NVMe-ASSIST: A NOVEL THEORETICAL FRAMEWORK
FOR DIGITAL FORENSICS
A CASE STUDY ON NVME STORAGE DEVICES
AND RELATED ARTIFACTS ON WINDOWS 10

A Dissertation

Presented to

The Faculty of the Department of Computer Science

Sam Houston State University

In Partial Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

---

by

Ashar Neyaz

August, 2022

NVMe-ASSIST: A NOVEL THEORETICAL FRAMEWORK

FOR DIGITAL FORENSICS

A CASE STUDY ON NVME STORAGE DEVICES

AND RELATED ARTIFACTS ON WINDOWS 10

by

Ashar Neyaz

APPROVED:

Naramsimha Shashidhar, Ph.D.
Committee Chair

Cihan Varol, Ph.D.
Committee Member

Amar Rasheed, Ph.D.
Committee Member

John Pascarella, Ph.D.
Dean, COSET

ABSTRACT

Neyaz, Ashar, *Nvme-assist: A novel theoretical framework for digital forensics a case study on nvme storage devices and related artifacts on Windows 10*. Doctor of Philosophy (Digital and Cyber Forensic Science), August, 2022, Sam Houston State University, Huntsville, Texas.

With ever-advancing changes in technology come implications for the digital forensics community. In this document, we use the term *digital forensics* to denote the scientific investigatory procedure for digital crimes and attacks. Digital forensics examiners often find it challenging when new devices are used for nefarious activities. The examiners gather evidence from these devices based on supporting literature. Multiple factors contribute to a lack of research on a particular device or technology. The most common factors are that the technology is new to the market, and there has not been much time to conduct sufficient research. It is also likely that the technology is not popular enough to garner research attention. If an examiner encounters such a device, they are often requir-ed to develop impromptu solutions to investigate such a case. Sometimes, exami-ners have to review their examination processes on model devices that labs are necessi-tated to purchase to see if existing methods suffice. This ad-hoc approach adds time and additional expense before actual analysis can commence. In this research, we investigate a new storage technology called **Non-Volatile Memory Express** (**NVMe**). This technology uses **Peripheral Component Interconnect** (**PCIe**) mechanics for its working. Since this storage technology is relatively new, it lacks a substantial digital forensics foundation to draw upon to conduct a forensics investigation.

Additionally, to the best of our knowledge, there is an insufficient body of work to conduct sound forensics research on such devices. To this end, our framework, **NVMe-Assist** puts forth a strong theoretical foundation that empow-ers digital forensics examiners in conducting analysis on NVMe devices, includi-

ng wear-leveling, TRIM, Prefetch files, Shellbag, and BootPerfDiagLogger.etl.

Lastly, we have also worked on creating the NVMe-Assist tool using Python. This tool parses the partition tables in the boot sector and is the upgrade of the mmls tool of The Sleuth Kit command-line tools. Our tool currently supports E01, and RAW files of the physical acquisition of hard-disk drives (HDDs), solid-state drives (SSDs), NVMe SSDs, and USB flash drives as data source files. To add to that, the tool works on both the MBR (Master Boot Record) and GPT (GUID Partition Table) style partitions.

KEYWORDS: NVMe; PCIe; wear-leveling; TRIM; Prefetch files; Shellbag; BootPerfDiagLogger.etl.

ACKNOWLEDGMENTS

First and foremost, I would like to thank Almighty Allah for giving me the strength and guidance to pursue a doctoral program in Digital and Cyber Forensic Science at Sam Houston State University. Due to His divine guidance, I could become the best version of myself.

I owe a debt of gratitude to my advisor Dr. Narasimha Shashidhar. It has been my privilege to study and conduct research under his guidance, and this dissertation would not be possible without his constant support, encouragement, and motivation. I have benefited immensely from his patience and ability to breathe perfection into the most complex and challenging digital and cyber forensics concepts. In addition, he has constantly pushed me to achieve better things throughout my graduate study. I am grateful to him for his kindness and everything he has taught me.

I am incredibly grateful to my associate advisor, Dr. Cihan Varol, for giving me updated information on operating system technology and encouraging me to explore and add new technological advancements to my dissertation. Moreover, I have thoroughly enjoyed discussing exciting research avenues that can help new practitioners to conduct research. I would also like to thank Dr. Amar Rasheed for his valuable insights on hardware forensics and for teaching me how to understand the essentials of hardware pins and controller chips. Furthermore, I would like to express my gratitude to Dr. Bing Zhou and Dr. Peter Cooper for their insights into teaching and for giving me the creative control of the courses I would be teaching in the department. Their guidance helped me in becoming a much better instructor and teaching assistant.

I am fortunate to have been a part of the Cyber Forensic Intelligence Center, where I worked with the best colleagues, Dustin Thornton, Ashley

Miksch, and Ricky Malcom. They always stood by me throughout my journey in the department. Moreover, they inspired me to work harder and enjoy the process. Also, I would like to thank my Ph.D. cohorts Valentin Gazeau, and Alberto Ceballos for teaching and assisting me in complex topics of Python scripting and Khushi Gupta for pushing me to become a better scientific writer.

No amount of words is ever sufficient to express my deep gratitude to my parents, Asad and Urusa Neyaz; my sister, Nimra Neyaz; my best friend, Kamran Akhter; and my uncle Irshad Malik and aunt, Safura Ahmad; the six people who never lost faith in me and to whom I always turned for inspiration, and comfort.

Huntsville, Texas **Ashar Neyaz**

August, 2022

TABLE OF CONTENTS

ABSTRACT iii

ACKNOWLEDGMENTS v

LIST OF TABLES xii

LIST OF FIGURES xix

CHAPTER I: INTRODUCTION 1

History and Motivation 1

Prior Work 3

Challenges of SSDs in Digital Forensics Investigations 9

Challenges of Windows 10 artifacts (Prefetch, Shellbag, and BootPerfDiagLogger.etl) in Digital Forensics investigations . . . 10

CHAPTER II: PRELIMINARIES AND FRAMEWORK DESIGN 11

Terminology 11

NVMe-Assist Framework 14

CHAPTER III: FORENSICS ACQUISITION AND ANALYSIS BRIEFING . 18

Image Acquisition Procedure from FTK Imager 19

Image analysis procedure in AccessData FTK 24

Image Analysis Procedure in Autopsy 26

Image Header Analysis in WinHex 27

CHAPTER IV: WINDOWS PREFETCH FORENSICS 28

Windows Prefetch 28

Types of Prefetching 29
Prefetch Storage Location 32
Prefetch Naming Scheme 32
Prefetch Hash Algorithm Generation Steps 33
Prefetch Configuration 33
Contents of Prefetch Files 34
Signature of Prefetch Files 36
Prefetch File Header 36
Operating System Version Based on Prefetch Files 36
File Information from Prefetch File 39
Forensics Information in Sections A, B, C, D, and F from a Prefetch File 43
Tools for Comparative Prefetch Forensics Analysis 48
CHAPTER V: WINDOWS SHELLBAG FORENSICS 52
Windows Shellbag 52
Forensics Importance of Shellbag 53
Shellbag Registry and File Location 54
Interpreting BagMRU and Bags Subkeys in the Registry 56
Experiment Initiation 60
Shellbag Entries for Desktop Folder and in C:\ Drive 60
Shellbag Entries for USB Drives 61
Shellbag Entries for Compressed Files 62
Forensics Analysis using OSForensics 65
Forensics Analysis using ShellBags Explorer Tool 66
Forensics Analysis using ShellBagsView Tool 67
Comparative Findings from Tools used 68
CHAPTER VI: WINDOWS 10 ETL FILE FORENSICS 71

Circular ETL File Configuration 72
Forensics Analysis of BootPerfDiagLogger.etl with ETLParser.exe . . 73
Forensics Analysis of BootPerfDiagLogger.etl with PerfView.exe . . . 76
Forensics Analysis of BootPerfDiagLogger.etl with FullEventLogView 79
Forensics Analysis of BootPerfDiagLogger.etl with SVCLogViewer . 80
Forensics Analysis of BootPerfDiagLogger.etl with TraceFMT 80
Forensics Analysis of BootPerfDiagLogger.etl with Windows Performance Analyzer 82
Forensics Importance of Analyzing BootPerfDiagLogger.etl with different Tools 83
CHAPTER VII: DIGITAL FORENSICS IN USB NVME SSDS WITH WRITEBLOCKER 84
Experimental Setup with USB WriteBlocker 85
Specifics of SSDs 88
Methodology and Experiment Initiation 90
Experiment Results, Analysis, and Discussion 93
Samsung and Seagate TRIM ON Analysis 93
Samsung and Seagate TRIM OFF Analysis 103
Hash Analysis for Samsung and Seagate NVMe SSDs 109
Western Digital and Silicon Power TRIM ON Analysis 112
Western Digital and Silicon Power TRIM OFF Analysis 120
Hash Analysis for Western Digital and Silicon Power NVMe SSDs . . 126
CHAPTER VIII: DIGITAL FORENSICS IN USB NVME SSDS WITHOUT WRITEBLOCKER 129
Experimental Setup without USB WriteBlocker 129
Specifics of SSDs 131

Methodology and Experiment Initiation 134
Experiment Results, Analysis, and Discussion 137
Samsung and Seagate TRIM ON Analysis without WriteBlocker . . . 137
Samsung and Seagate TRIM OFF Analysis without WriteBlocker . . 147
Hash Analysis for Samsung and Seagate NVMe SSDs 153
Western Digital and Silicon Power TRIM ON Analysis without WriteBlocker 156
Western Digital and Silicon Power TRIM OFF Analysis without WriteBlocker 160
Hash Analysis for Western Digital and Silicon Power NVMe SSDs without WriteBlocker 168

CHAPTER IX: DIGITAL FORENSICS IN PCIE NVME SSDS WITH NVME WRITEBLOCKER 170

Experimental Setup with NVMe WriteBlocker 170
Specifications of SSDs 172
Methodology and Experiment Initiation 174
Experiment Results, Analysis, and Discussion 176
Samsung and Seagate TRIM ON Analysis with NVMe WriteBlocker 177
Samsung and Seagate TRIM OFF Analysis with NVMe WriteBlocker 184
Hash Analysis for Samsung and Seagate NVMe SSDs with NVMe WriteBlocker 193
Western Digital and Silicon Power TRIM ON Analysis with NVMe WriteBlocker 196
Western Digital and Silicon Power TRIM OFF Analysis with NVMe WriteBlocker 201

Hash Analysis for Western Digital and Silicon Power NVMe SSDs with NVMe WriteBlocker 211

CHAPTER X: NVMe-ASSIST PYTSK CODES 214

CHAPTER XI: REPORTING GUIDELINES AND INSTRUCTIONS 225

CHAPTER XII: CONCLUSION AND FUTURE WORK 233

Conclusion 233

Future Work 234

REFERENCES 235

VITA 245

LIST OF TABLES

2.1 The software tools used for analyzing Windows 10 artifacts. 16

4.1 The availability of NTOSBOOT_B00DFAAD.pf file in different Windows versions. 30

4.2 The EnablePrefetch REG_DWORD values for selection. 34

4.3 Prefetch file header. 37

4.4 Windows version from prefetch file. 37

4.5 Windows XP file information in prefetch file. 40

4.6 Windows Vista/7 file information in prefetch file. 41

4.7 Windows 8/8.1 file information in prefetch file. 42

4.8 Windows 10 file information in prefetch file. 43

4.9 Volume information from Section D in prefetch file. 46

4.10 Directory information from Section F in prefetch file. 47

5.1 The list of Windows Registry Hives and their supporting files. 55

5.2 Location of Shellbag entries inside Windows Registry, and NTUSER.DAT and UsrClass.DAT files. 56

5.3 Summary of the test experiments. 60

5.4 Tools comparison chart based on artifacts obtained. 70

7.1 Equipment used in the experiment. 86

7.2 Information of Samsung and Seagate NVMe SSDs used in the experiment. 89

7.3 Information of Western Digital and Silicon Power NVMe SSDs used in the experiment. 90

7.4 Timeline information of forensic file acquisition. 93

7.5 Timeline information of forensic file acquisition. 94
7.6 The number of files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 95
7.7 The number of files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 96
7.8 The number of files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 97
7.9 The number of files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 98
7.10 The number of files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 103
7.11 The number of files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 104
7.12 The number of files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 104
7.13 The number of files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 105
7.14 Digital forensics information about forensically acquired image files of Samsung and Seagate NVMe SSDs with USB WriteBlocker. 111
7.15 The number of files recovered from FTK in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 112
7.16 The number of files recovered from Autopsy in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 113
7.17 The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 114
7.18 The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case. 115

7.19 The number of files recovered from FTK in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 120
7.20 The number of files recovered from Autopsy in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 121
7.21 The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 121
7.22 The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case. 122
7.23 Digital forensics information about forensically acquired image files of Western Digital and Silicon Power NVMe SSDs with USB WriteBlocker. 128

8.1 Equipment used in the experiment. 129
8.2 Information of Samsung and Seagate NVMe SSDs used in the experiment. 133
8.3 Information of Western Digital and Silicon Power NVMe SSDs used in the experiment. 134
8.4 Timeline information of forensic file acquisition. 137
8.5 Timeline information of forensic file acquisition. 138
8.6 The number of files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 139
8.7 The number of files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 140
8.8 The number of files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 141

8.9 The number of files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 142
8.10 The files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 147
8.11 The files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 148
8.12 The files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 148
8.13 The files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 149
8.14 Digital forensics information about forensically acquired image files of Samsung and Seagate NVMe SSDs without USB WriteBlocker. . . 155
8.15 The number of files recovered from FTK in Western Digital (WD) NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 156
8.16 The number of files recovered from Autopsy in Western Digital (WD) NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 157
8.17 The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 158

8.18 The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case. 159
8.19 The number of files recovered from FTK in Western Digital NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 160
8.20 The number of files recovered from Autopsy in Western Digital NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 161
8.21 The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 162
8.22 The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case. 163
8.23 Digital forensics information about forensically acquired image files of Western Digital and Silicon Power NVMe SSDs without USB WriteBlocker. 169

9.1 Equipment used in the experiment with NVMe WriteBlocker. 171
9.2 Information of Samsung and Seagate NVMe SSDs used in the experiment. 173
9.3 Information of Western Digital and Silicon Power NVMe SSDs used in the experiment. 174
9.4 Timeline information of forensic file acquisition with NVMe WriteBlocker. 177
9.5 Timeline information of forensic file acquisition with NVMe WriteBlocker. 178

9.6 The number of files recovered using AccessData FTK in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 180
9.7 The number of files recovered using Autopsy in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 181
9.8 The number of files recovered using AccessData FTK in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 182
9.9 The number of files recovered using Autopsy in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 183
9.10 The number of files recovered using AccessData FTK in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 185
9.11 The number of files recovered using Autopsy in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 186
9.12 The number of files recovered using AccessData FTK in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 187
9.13 The number of files recovered using Autopsy in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 188
9.14 Digital forensics information about forensically acquired image files of Samsung and Seagate NVMe SSDs with NVMe WriteBlocker. . . . 195
9.15 The number of files recovered using AccessData FTK in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 197
9.16 The number of files recovered using Autopsy in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 198
9.17 The number of files recovered using AccessData FTK in SP NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 199
9.18 The number of files recovered using Autopsy in SP NVMe SSD as a primary boot device in Windows 10 TRIM ON case. 200

9.19 The number of files recovered using AccessData FTK in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 202
9.20 The number of files recovered using Autopsy in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 203
9.21 The number of files recovered using AccessData FTK in Silicon Power NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 204
9.22 The number of files recovered using Autopsy in Silicon Power NVMe SSD as a primary boot device in Windows 10 TRIM OFF case. 205
9.23 Digital forensics information about forensically acquired image files of Western Digital and Silicon Power NVMe SSDs with NVMe WriteBlocker. 213

LIST OF FIGURES

1.1 The seminal advances in storage and OS technology/forensic artifact. †- Topics covered under current research. 2

2.1 The NVMe-Assist Digital Forensics Framework. 17

3.1 AccessData FTK Imager step 1. 19

3.2 AccessData FTK Imager step 2. 20

3.3 AccessData FTK Imager step 3. 20

3.4 AccessData FTK Imager step 4. 21

3.5 AccessData FTK Imager step 5. 21

3.6 AccessData FTK Imager step 6. 22

3.7 AccessData FTK Imager step 7. 22

3.8 AccessData FTK Imager step 8. 23

3.9 AccessData FTK Imager step 9. 23

3.10 AccessData FTK Imager step 10. 24

3.11 Forensics image analysis in AccessData FTK step 1. 24

3.12 Forensics image analysis in AccessData FTK step 2. 25

3.13 Forensics image analysis in AccessData FTK step 3. 25

3.14 Forensics image analysis in Autopsy step 1. 26

3.15 Forensics image analysis in Autopsy step 2. 26

3.16 Forensics image analysis in Autopsy step 3. 27

3.17 Image header analysis in WinHex. 27

4.1 The presence of NTOSBOOT_B00DFAAD.pf in Windows XP and Vista. 31

4.2 The presence of NTOSBOOT_B00DFAAD.pf in Windows 7 compared to lack of it in Windows 8. 31

4.3 The Prefetch configuration in Windows 10 Registry Editor. 34
4.4 The little-endian representation of EnablePrefetch entry. 35
4.5 The Windows XP prefetch file header. 37
4.6 The Windows Vista prefetch file header. 38
4.7 The Windows 7 prefetch file header. 38
4.8 The Windows 8 prefetch file header. 38
4.9 The Windows 8.1 prefetch file header. 38
4.10 The Windows 10 prefetch file header after decompression. 39
4.11 The files referenced information from Windows XP prefetch file. . . 45
4.12 The files referenced information after decompressing Windows 10 prefetch file. 45
4.13 The Section D information after decompressing Windows 10 prefetch file. 47
4.14 The Section F information after decompressing Windows 10 prefetch file. 47
4.15 Prefetch Artifacts Viewer from OSForensics tool. 48
4.16 Windows Prefetch Parser open-source python script. 49
4.17 WinPrefetchView freeware tool. 50
4.18 PECmd tool by Eric Zimmerman. 50
4.19 Decompressing Windows 10 prefetch file using Francesco Picasso's python script. 51

5.1 Windows Registry Editor 54
5.2 Issuing **wmic** command to obtain SID of a user account. 55
5.3 The registry path translation to original path from Shellbag entries. . 57
5.4 The NodeSlot entry for txt_doc folder's view settings. 58
5.5 The Bags entry number 59 for txt_doc folder's view settings. 58
5.6 Obtaining name of folder using BagMRU entry. 59

5.7 The MRUListEx inside BagMRU key. 59
5.8 BagMRU Shellbag entry for a folder customization inside Desktop. . . 61
5.9 BagMRU Shellbag entry for a folder customization inside C:\Drive. . . 62
5.10 BagMRU Shellbag entry for a folder customization on a USB Drive with E:\drive letter. 63
5.11 BagMRU Shellbag entry for a ZIP file inside a folder on Desktop. . . 64
5.12 BagMRU Shellbag entry for a ZIP file inside a folder under C:\. . . . 64
5.13 BagMRU Shellbag entry for a ZIP file inside a folder in USB drive with E:\drive letter. 65
5.14 OSForensics analysis windows. 66
5.15 Shellbags Explorer analysis window for folders. 67
5.16 ShellBags Explorer analysis window for zip file. 67
5.17 ShellBagsView analysis window for folders. 68
5.18 ShellBagsView analysis window for zip files. 68

6.1 Events in BootPerfDiagLogger.etl file. 73
6.2 The parsing of BootPerfDiagLogger.etl using ETLParser.exe. 74
6.3 The parsing of BootPerfDiagLogger.etl using ETLParser.exe. 74
6.4 The output of ETLParser.exe in Microsoft Excel file. 75
6.5 The output of ETLParser.exe in DB Browser (SQLite) tool. 75
6.6 The output of Perfview.exe displaying headers from the parsed etl file. 76
6.7 The output of Perfview.exe displaying information of trace and machine. 77
6.8 The output of Perfview.exe displaying dead process summary. 77
6.9 The output of Perfview.exe displaying live process summary. 78
6.10 The output of Perfview.exe displaying event types and details. 78
6.11 The output of Perfview.exe showing event statistics. 79
6.12 The output of FullEventLogView. 79

6.13 The output of SvcLogViewer. 80
6.14 The output of TraceFMT. 81
6.15 The output of TraceFMT generated txt file of the parsed ETL file. . . 81
6.16 The output of Windows Performance Analyzer. 82

7.1 Samsung NVMe SSD attached with USB WriteBlocker. 86
7.2 Seagate NVMe SSD attached with USB WriteBlocker. 87
7.3 Western Digital NVMe SSD attached with USB WriteBlocker. 87
7.4 Silicon Power NVMe SSD attached with USB WriteBlocker. 88
7.5 The status of TRIM in Windows 10 using fsutil command. 91
7.6 File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM ON case, with using a USB WriteBlocker. 99
7.7 File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM ON case, with using a USB WriteBlocker. 100
7.8 File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM ON case, with using a USB WriteBlocker. 101
7.9 File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM ON case, with using a USB WriteBlocker. 102
7.10 File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 106
7.11 File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 107
7.12 File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 108
7.13 File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 109

7.14 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Samsung NVMe SSD. 110
7.15 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Seagate NVMe SSD. 110
7.16 File, Set-1-xml(296).xml, over 696 bytes in Western Digital NVMe SSD TRIM ON case, with using a USB WriteBlocker. 116
7.17 File, Set-1-xml(296).xml, under 696 bytes in Western Digital NVMe SSD TRIM ON case, with using a USB WriteBlocker. 117
7.18 File, Set-1-xml(296).xml, over 696 bytes in Silicon Power NVMe SSD TRIM ON case, with using a USB WriteBlocker. 118
7.19 File, Set-1-xml(296).xml, under 696 bytes in Silicon Power NVMe SSD TRIM ON case, with using a USB WriteBlocker. 119
7.20 File, Set-1-xml(296).xml, over 696 bytes in Western Digital NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 123
7.21 File, Set-1-xml(296).xml, under 696 bytes in Western Digital NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 124
7.22 File, Set-1-xml(296).xml, over 696 bytes in Silicon Power NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 125
7.23 File, Set-1-xml(296).xml, under 696 bytes in Silicon Power NVMe SSD TRIM OFF case, with using a USB WriteBlocker. 126
7.24 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Western Digital NVMe SSD. 127

7.25 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Silicon Power NVMe SSD. 127

8.1 Samsung NVMe SSD attached without USB WriteBlocker. 130

8.2 Seagate NVMe SSD attached without USB WriteBlocker. 130

8.3 Western Digital NVMe SSD attached without USB WriteBlocker. . . 131

8.4 Silicon Power NVMe SSD attached without USB WriteBlocker. 131

8.5 The status of TRIM in Windows 10 using fsutil command. 135

8.6 File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM ON case, without using a USB WriteBlocker. 143

8.7 File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM ON case, without using a USB WriteBlocker. 144

8.8 File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM ON case, without using a USB WriteBlocker. 145

8.9 File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM ON case, without using a USB WriteBlocker. 146

8.10 File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 150

8.11 File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 151

8.12 File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 152

8.13 File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 153

8.14 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Samsung NVMe SSD, without using a USB WriteBlocker. 154

8.15 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Seagate NVMe SSD, without using a USB WriteBlocker. 154
8.16 File, Set-1-xml(296).xml, over 696 bytes in Western Digital NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 164
8.17 File, Set-1-xml(296).xml, under 696 bytes in Western Digital NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 165
8.18 File, Set-1-xml(296).xml, over 696 bytes in Silicon Power NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 166
8.19 File, Set-1-xml(296).xml, under 696 bytes in Silicon Power NVMe SSD TRIM OFF case, without using a USB WriteBlocker. 167
8.20 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Western Digital NVMe SSD, without using a USB WriteBlocker. . 168
8.21 Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Silicon Power NVMe SSD, without using a USB WriteBlocker. . . . 168

9.1 Samsung NVMe SSD attached with NVMe WriteBlocker. 171
9.2 Seagate NVMe SSD attached with NVMe WriteBlocker. 171
9.3 Western Digital NVMe SSD attached with NVMe WriteBlocker. . . . 172
9.4 Silicon Power NVMe SSD attached with NVMe WriteBlocker. 172
9.5 The status of TRIM in Windows 10 using fsutil command issued from CMD. 175
9.6 Hexadecimal contents of xls-files(1).xls file in the original dataset from Samsung NVMe SSD. 189
9.7 Hexadecimal contents of xls-files(1).xls file after recovery from Samsung NVMe SSD TRIM ON case. 190

9.8 Hexadecimal contents of xls-files(1).xls file after recovery from Samsung NVMe SSD TRIM OFF case. 191
9.9 Hexadecimal contents of xls-files(1).xls file in the original dataset from Seagate NVMe SSD. 192
9.10 Hexadecimal contents of xls-files(1).xls file after recovery from Seagate NVMe SSD TRIM OFF case. 193
9.11 Hash values of xls-files(1).xls in Samsung NVMe SSD when using NVMe WriteBlocker. 194
9.12 Hash values of xls-files(1).xls in Seagate NVMe SSD when using NVMe WriteBlocker. 194
9.13 Hexadecimal contents of xls-files(1).xls file in the original dataset from Western Digital NVMe SSD. 206
9.14 Hexadecimal contents of xls-files(1).xls file after recovery from Western Digital NVMe SSD TRIM OFF case. 207
9.15 Hexadecimal contents of xls-files(1).xls file in the original dataset from Silicon Power NVMe SSD. 208
9.16 Hexadecimal contents of xls-files(1).xls file after recovery from Silicon Power NVMe SSD TRIM ON case. 209
9.17 Hexadecimal contents of xls-files(1).xls file after recovery from Silicon Power NVMe SSD TRIM OFF case. 210
9.18 Hash of xls-files(1).xls in Western Digital NVMe SSD using NVMe WriteBlocker. 211
9.19 Hash of xls-files(1).xls in Silicon Power NVMe SSD using NVMe WriteBlocker. 212

10.1 Manual page of NVMe-Assist Toolkit. 219
10.2 Running demonstration of NVMe-Assist step-1. 219
10.3 Running demonstration of NVMe-Assist step-2. 219

10.4 Running demonstration of NVMe-Assist step-3. 220
10.5 Running demonstration of NVMe-Assist step-4. 220
10.6 Running demonstration of NVMe-Assist step-5. 221
10.7 Running demonstration of NVMe-Assist step-6. 221
10.8 Manual page of GPT Sector Parser of NVMe-Assist Toolkit. 221
10.9 Running demonstration of GPT Sector Parser step-1. 222
10.10 Running demonstration of GPT Sector Parser step-2. 222
10.11 Running demonstration of GPT Sector Parser step-3. 223
10.12 Running demonstration of Logical Partition Parser step-1. 223
10.13 Running demonstration of Logical Partition Parser step-2. 224
10.14 Running demonstration of Logical Partition Parser step-3. 224
10.15 Running demonstration of Logical Partition Parser step-4. 224

CHAPTER I

Introduction

In this dissertation, we investigated the new computer storage technology, Non-Volatile Memory Express (NVMe). This storage technology uses Peripheral Computer Interconnect Express (PCIe) mechanism for its working. Due to the technology being relatively new, it lacks a sound digital forensics foundation to draw upon to conduct digital forensics investigations. Thus, in order to contribute to the digital forensics community, we designed a novel theoretical framework, **NVMe-Assist**. Using our framework, practitioners can make a sound digital forensics decision on which analytical processes to apply should they encounter cases regarding NVMe technology.

Furthermore, we also investigated useful forensics artifacts left on a Windows 10 operating system by user interactions. We mainly focused on Prefetch, Shellbag, and BootPerfDiagLogger.etl files. Due to numerous updates in the Windows operating systems, these components have also gone through numerous changes. As a result, we used proprietary and commercial tools to unearth valuable information.

History and Motivation

We live in an age where technology is rapidly changing, and new digital devices are being introduced to the market every day. One such technology that has made such significant advances is computer storage technology. To accommodate the ever-increasing demands of consumers and businesses, innovation in storage technology is a constant undertaking. In this section, we want to draw a parallel between advances in storage technology and the complementary advances in digital forensics framework over the years. Figure 1.1 talks about seminal advances in storage media technology over the years in conjunction with the development of different Microsoft Windows operating systems and corresponding forensics artifacts.

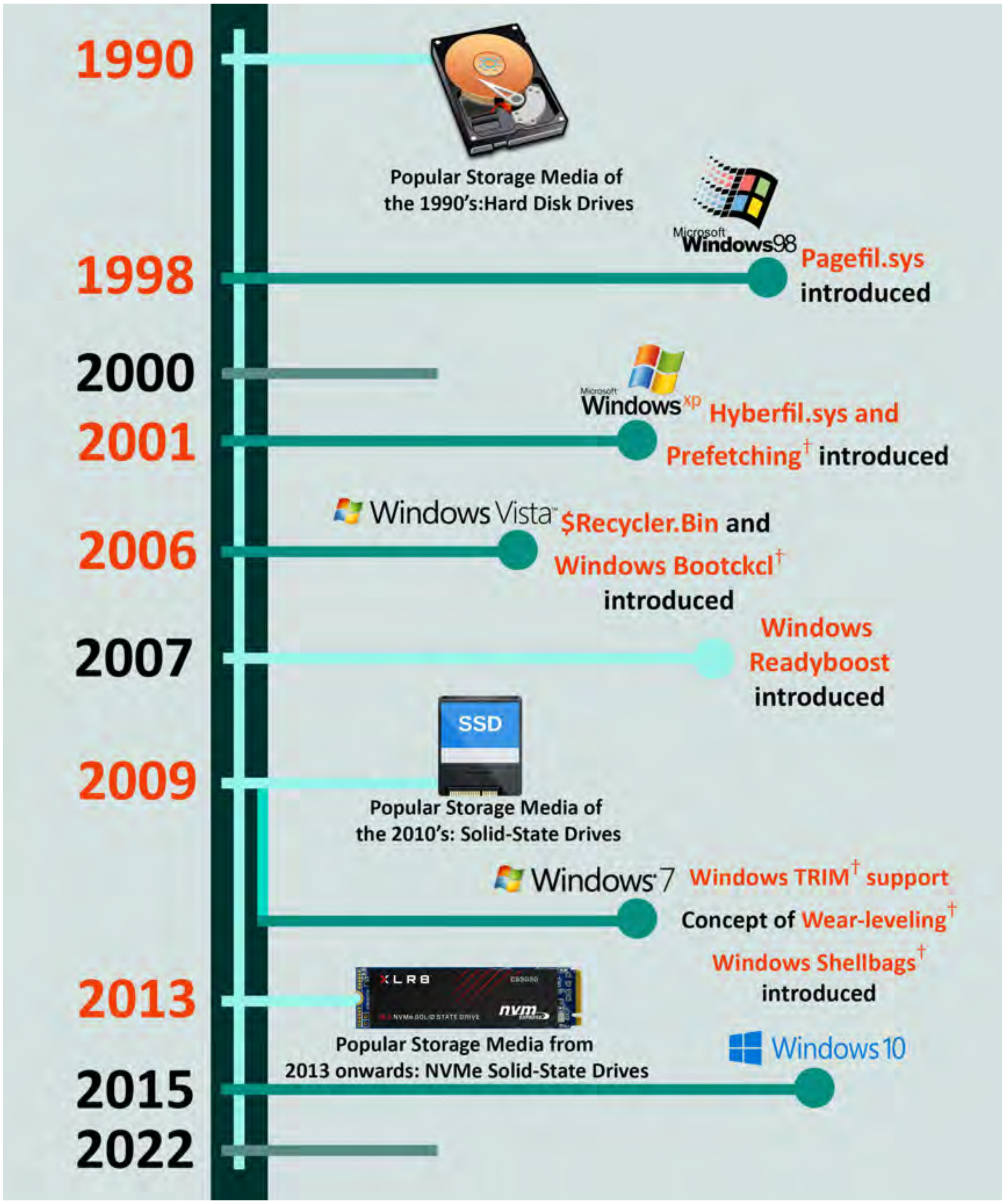


Fig. 1.1. The seminal advances in storage and OS technology/forensic artifact. †- Topics covered under current research.

In this figure, we also highlight the pivotal changes in the operating systems' designs to enhance the user experience and efficiency. These changes also have implications on forensics artifacts as well. In this work, we focus on the following artifacts: Prefetch files, Shellbag, and BootPerfDiagLogger.etl of the Windows 10 operating system because they are the source of vital forensics artifacts. This is because Windows 10 is predominantly the market leader in operating systems family. Although we showed Windows ReadyBoost, Pagefile.sys, Hyberfil.sys, and $Recycler.Bin, we did not dedicate much time to them in this study because we believe that those artifacts were not suitable for our NVMe-Assist framework. We concentrate on investigating the effect of wear-leveling, and TRIM in four different NVMe SSDs namely; of Samsung, Seagate, Western Digital, and Silicon Power.

Prior Work

Since its inception, numerous research initiatives have focused on solid-state drives (SSD) forensics. Unlike traditional mechanical hard drives, solid-state media tend to eliminate the data if not retrieved promptly. The dependence on the SSD controller, TRIM, and solid-state media garbage collection mechanism makes it challenging to conduct a sound forensics analysis than on the mechanical drives. Additionally, with the development of Windows 10, conducting forensics analysis on the operating system artifacts have become a new research area. The following literature review confirms that increasing SSDs has raised numerous challenges for digital forensic investigators during investigations. Furthermore, this review also highlights the forensic importance of the Windows operating system.

Hard disk drives, file recovery and carving have been challenging to effectuate. This is because the file system metadata information is required.

Pal and Memon [1] presented the evolution of file carving and described the techniques to recover files without needing metadata information from the file system. The authors have also predicted the prevalence of SSDs that will present difficulties in recovering data due to constant movement of data internally.

Bell and Boddington [2] further substantiated the challenges posed by SSDs in the evidence collection process. Moreover, the authors demonstrated that the conventional assumptions about the behavior of storage media devices are no longer valid in solid-state media. They exemplified their claim by conducting a series of experiments explaining the autonomous nature of SSD devices, including how evidence can be contaminated and complicate the forensic analysis process.

The author in [3] examines the file recovery process in SSDs by applying a forensics methodology. The article attempted to show a general overview of the file recovery process in SSDs. Ritola did a similar work of file recovery on solid-state drives [4]. Ritola not only aimed to forensically retrieve information from the storage area of SSD but also the spare area or free space.

Testing the forensic friendliness of an SSD has proven to be difficult, which can be seen from the research mentioned above. However, the article presented in [5] showed a series of steps to assess the forensic benevolence of an SSD. With the help of their methodology, the authors produced a decision that assists an analyst in deciding on taking a direct acquisition of memory cells of the SSD. They also issued a detailed description of the steps involved in conducting their experiment.

In [6], the article talked about favorable voltage measurement techniques to monitor the read-write operations on an SSD. Drawing a comparison between the HDD and SSD forensic analysis is challenging. The comprehensive research conducted by Joshi and Hubbard [7] provided the basic techniques needed to conduct forensics investigations in HDDs and SSD. The authors have also explained the self-corrosion concept and the limited life span of an SSD hindering forensic investigations. Finally, the article discussed methods that can mitigate the impact of forensic analysis due to TRIM and the garbage collection features of the device.

The author in [8] extended the work done in [2]. The thesis presented by the author identified key evidence in HDDs and SSDs along with the self-destructive nature of SSD which causes difficulties in forensics investigations. Along the same line, the authors in [9] compared the probability of file recovery in different flash media and a solid-state drive due to the self-destructive nature of the media. The article also presented the factors affecting the file recovery process by exhibiting a series of experiments to set a standard for digital forensics investigations. In addition to exploring the self-destructive nature of SSDs, studying the root cause of the autonomous behavior of SSDs is crucial. The authors in [10] studied two firmware versions of an SSD controller using current draw signals.

Damaged or crushed storage media can pose a risk for file recovery. The research in [11] discussed semantic carving and fragment recovery carving techniques for recovering files from a storage medium. In addition to this, the research highlighted in [12] was very similar to [10] in studying the TRIM operation of an SSD using a side-channel approach. The authors accurately inferred the TRIM operation with 99% accuracy. Consequently, this directly has implications for malware detection, digital forensics, and consumer privacy.

Presenting the digital evidence in a court of law is an accountable and strenuous task. Any tampering with the evidence will lead to the expulsion of the evidence. The research in [13], [14], and [15] demonstrated and offered guidelines for conducting forensics analysis in SSDs supporting TRIM, wear-leveling, and garbage collection features. The authors of the three research conducted experiments with different brands of SSDs for conducting forensics analysis and showed the impact of file recovery due to the different features of SSDs.

Understanding state-of-the-art technology is necessary in every field. For this purpose, a survey is needed to know the current standing of the body of work or contribution. Kumar, Neyaz, and Shashidhar [16] put together a survey article highlighting the work done in SSD forensics. They discussed various research that talk about data recovery in SSDs. Additionally, the authors present the latest forensics techniques and ideas in the literature in the field of SSD forensics.

Our research is dedicated to conducting research on NVMe SSDs and for that purpose, we conducted an exhaustive survey of the literature on NVMe SSDs. The gaps in the forensics framework in SSDs have enhanced after the invention of NVMe SSDs. These storage devices have different command sets, and thus, ATA/SCSI commands do not work on them. Nikkel [17] explored the NVMe technology and discussed its relevance to the digital forensics community. The author also mentioned the possible challenges when NVMe SSDs are involved, including the media's forensics acquisition.

With any type of SSD, SATA SSD, or NVMe SSD, there is the concept of TRIM and wear-leveling attached to them. Due to these concepts, the forensics recoverability of data always becomes uncertain. The research in [18] talked about the ability of data recovery tools to restore digital evidence on a TRIM enabled NVMe SSD. The authors concluded that the TRIM feature affects data recovery. A similar study in [19] talked about TRIM in NVMe SSDs in disabled and enabled cases. The authors affirmed that toggling TRIM on and off impacts the data recovery and forensics investigation.

Windows artifacts such as Prefetch files, BootPerDiagLogger.etl, Shellbag, RunKey, ShellLink, Windows Search History, etc., yield a lot of information having high forensic importance. The work done by Garcia in [20] talks about a scan_winprefetch tool. The tool scans for Windows Prefetch files in forensic disk images, analyzes them, and creates an XML report. In [21] the author shows the working of the prefetching process and highlights the changes in prefetching technology in various Windows operating systems. The author also showed the evidentiary value related to Prefetch files. The mechanism behind the creation and manipulation ofcreating and manipulating the Prefetch files is worth investigating. The research in [22] examined Prefetch files using the assembly code generated using IDA Pro software. The authors analyzed the Windows executable ntkrnlpa.exe to explore the kernel process responsible for creating prefetch files.

Process-related information is also a part of Windows artifacts. For example, the research conducted in [23] describes the System Resource Usage Monitor (SRUM) mechanism responsible for tracking process and network statistics in Windows 8 and above. The authors also compared the information presented by SRUM analysis, using their custom-made tool, to correlate information found in SRUM. The authors took a very similar approach in [24], where they examined the forensic information presented by Amcache.hve files with Prefetch, SRUDB.dat, and IconCache.db files to present useful forensics.

Hidden process is a serious threat when conducting a forensic investigation of a system. The author in [25] created a Hidden Processes Detector (HPD) program for detecting and revealing the hidden processes based on Windows Prefetch files.

Similarly, deleted user accounts also present a blockage in conducting forensics analysis. A user account contains a wealth of information related to a user, which is difficult to find when a user account is deleted. The authors in [26] researched this anti-forensic technique of deleting user accounts.

The research work conducted in [27], [28], [29], and [30] talked about the prefetching technology in Windows 10 operating system. In addition, they explored the changes that happened due to the operating system update from Windows 8.1. Moreover, the research also talked about the effect of malware on Windows 10 prefetch files and what happens when a prefetch file is manipulated and re-compressed to hide entries from the file system.

Other avenues explored by our framework: the Shellbag and BootPerfDiagLogger.etl file in Windows 10 operating system. The BootPerfDiagLogger.etl file contains information about a computer system's booting information, whereas the Shellbag entries contain user preference information for browsing folders. The research done in [31] briefly talked about the log entry of Western Digital hard disk drive files found in BootPerfDiagLogger.etl file. In [32], the authors introduced a novel method to analyze the Shellbag information from Windows XP Registry to reconstruct user activities by comparing successive registry snapshots.

A lot changed when Windows 7 was launched in 2009. Due to this change, the whole user-experience was redesigned and updated. This refinement led to changes in Shellbag as well. The authors of the research [33] investigated the Shellbag files in Windows 7 and created a timeline of user information by parsing the information from the files. Similarly, in [34], the author investigated Shellbag files in Windows 8, 8.1, and the introductory version of Windows 10 of 2015.

Anti-forensics is another challenge in the area of digital forensics. The purpose of anti-forensics is to obfuscate the digital forensics investigation. When this nefarious technique is used to modify the areas of forensics importance in the Windows operating system, the amount of evidence found in an investigation can diminish. The thesis in [35] talked about the anti-forensics wiping tools. The author tested the wiping tools on Shellbags, Prefetch, Jump Lists, etc., to check the impact of the wiping tools on these Windows artifacts. Muir, Leimich, and Buchanan [36] investigated the Tor browser and found user preference settings with the volatility tool plugin for Shellbag entries, ***shellbags***.

There were more changes introduced when Windows 10 had subsequent updates since 2015. The authors in [37] and [38] investigated the changes in Windows artifacts such as Shellbags, Prefetch, Shimcache, and associated timestamps. They thoroughly analyzed the changes so that the forensics examiners do not misinterpret any valuable data.

Challenges of SSDs in Digital Forensics Investigations

1. Uncertainty in finding the evidence due to wear-leveling.
2. Data fragmentation and changes in hash values due to the influence of wear-leveling mechanism.
3. Purging of data immediately after deletion.
4. The effect of TRIM on data from the operating system level.
5. Influence of physical write-blocker and auto-mounting of the device in the operating system impacting the forensic analysis.
6. Different forensic results due to the use of disparate NAND flash chips such as **Single-Level Cell** (**SLC**), **Multi-level Cell** (**MLC**), and **Triple-Level Cell** (**TLC**) on storage media.
7. Every manufacturer implements the storage mechanism differently, impacting the forensic analysis due to proprietary standards.

8. No procedure for forensic analysis in **Redundant Array of Independent Disks** (**RAID**) configuration of solid-state media.

9. Unpredictable data storage pattern.

10. The presence of hardware-level encryption and soldered storage that renders the forensic examination ineffective and unproductive.

Challenges of Windows 10 artifacts (Prefetch, Shellbag, and BootPerfDiagLogger.etl) in Digital Forensics Investigations

1. Conducting a comprehensive forensic analysis due to changes in Windows 10 prefetching technology due to operating system updates.

2. Analyzing information about users' preferences and behavior using Shellbag entries.

3. Decoding and analyzing the BootPerfDiagLogger.etl files of the boot process and obtaining useful information for forensics analysis.

CHAPTER II

Preliminaries and Framework Design

In this chapter, we are going to introduce the terminologies associated with computer storage technology. This will help the readers familiarize themselves with the terms so that they can easily comprehend the research subject. Furthermore, we are going to exhibit the design of our NVMe-Assist framework to help readers understand how our contributing framework tackles the challenges posed by NVMe-SSDs.

Terminology

1. **Hard disk drive**- A hard disk drive or HDD is a non-volatile data storage device. It stores data on one or more platters made of magnetic material [39].

2. **Platter**- A platter on an HDD is a disk coated with magnetic media. Platters are made of aluminum, glass, or ceramic and begin to rotate when a computer is turned on to read, write and seek data [40].

3. **Sector**- A sector is the smallest physical unit on a hard disk drive, usually 512 bytes in size, to store data [41].

4. **Cluster**- A combination of one or more consecutive sectors is called a cluster [41].

5. **Serial Advanced Technology Attachment**- SATA or Serial Advanced Technology Attachment is an industry-standard for connecting and transferring data from hard disk drives (HDDs) or solid-state drives (SSDs) to computer systems. Unlike Integrated Drive Electronics (IDE), SATA uses serial transfer technology to transmit data [42].

6. **Solid-state drive**- A solid-state drive (SSD) is a non-volatile storage device that stores data persistently using integrated circuit assemblies. SSDs do not have any moving parts such as read-write heads, platters to store data. Instead, it uses flash memory for data storage [43].

7. **Pages**- The smallest unit physical unit of data storage in an SSD is called a page. Typically, it is of 4KB in size. A page is sometimes referred to as a cell [44].

8. **Block**- A combination of several pages is called a block. Generally, there are 128 pages in a block; therefore, one block contains 512 bytes of storage space [44].

9. **Single-Level Cell NAND Flash**- A single-level cell or SLC is a type of flash chip that stores only one bit per cell. It offers the highest performance, reliability, and endurance. However, the downside of SLC is its price as it is considerably higher than other NAND flash types [45].

10. **Multi-Level Cell NAND Flash**- A multi-level cell or MLC is a type of flash chip that stores only two bits per cell. MLC offers good performance, reliability, and endurance and is cheaper than SLC [45].

11. **Triple-Level Cell NAND Flash**- A triple-level cell or TLC stores 3-bits per cell to store data. This NAND flash is commonly used for consumer-grade products. Compared to the previous two NAND flash, TLC has lower performance, reliability, and endurance [45].

12. **Solid-state drive Controller**- An SSD controller is a chip on a solid-state drive responsible for controlling the working of the storage device. The controller chip has the electronics for bridging the flash storage components to the SSD data transfer interface, i.e., the SATA interface [46].

13. **TRIM**- A TRIM command is a feature of an operating system such as Windows 10, macOS that notifies a solid-state device which block of data is no longer required to be utilized and can be safely be erased to be writable again [47].

14. **Wear-leveling**- Wear leveling is a technique employed by solid-state drive (SSD) controllers to increase the storage device's lifespan. The controller evenly distributes writing on all SSD blocks, so they evenly wear. All the memory blocks receive the same number of write frequency to avoid data writing too often on the same blocks [48].

15. **Peripheral Component Interconnect Express**- A peripheral component interconnect express (PCIe) is an interface in a computer motherboard connecting high-speed electronics components such as graphics card, Wi-Fi cards, RAID cards, SSDs [49].

16. **Non-Volatile Memory Express Solid-State Drive**- Non-Volatile Memory Express or NVMe is an interface for data communication and driver that defines a command and feature sets. The underlining principle behind NVMe is PCIe. The goals of NVMe are increased data transfer speed, efficient performance, and interoperability on an extensive range of enterprise and client systems. NVMe was designed for solid-state drives [50].

17. **Prefetch**- The Prefetching technology or Prefetch is a Windows feature implemented to decrease the load time of frequently used application. The speed of a application program execution increases due to the use of cache files for faster access [51].

18. **Shellbags**-They are valuable sources of information that include user

preferences while browsing folders for a specific user. Microsoft Windows records view settings of folders and the desktop of a user account. Therefore, when the user revisits the folder or desktop, Windows remembers the location of the folder, view, and positions of items, respectively. The setting values are stored in the Shellbags keys of the Windows registry [52].

19. **BootPerfDiagLogger.etl**- BootPerfDiagLogger.etl is a log file that includes boot trace information on computer booting. This etl file of circular kernel context logs type. The circular context of the file overwrites old events with new events when the max file buffer size is reached [53].

20. **File Recovery**-This technique makes use of the file system information that remains after deletion of a file. For this approach to work, the file system information needs to be correct. If not, the files can't be recovered. If a system is formatted, the file recovery techniques will not work either [54].

21. **File Carving**- File Carving deals with the raw data on the media and doesn't use the file system structure during its process. Although carving doesn't care about which file system is used to store the files, it could be very helpful to understand how a specific file system works. Carving makes use of the internal structure of a file. A file is a block of stored information like an image in a JPEG file [54].

NVMe-Assist Framework

In an effort to fill the gaps in solid-state media forensics, we designed a theoretical framework called **NVMe-Assist**. The goal of our framework was to address the changes and issues in forensics methodologies due to the development of NVMe SSDs. Furthermore, its objective is to assist digital forensics investigators in analyzing NVMe SSDs for forensic artifacts. In addition, it provided a sound framework for conducting a forensic investigation by adding to the existing literature. Our research experiment included the following four NVMe SSD storage media:

(a) **Samsung 970 EVO Plus NVMe SSD**
(b) **Western Digital SN550 NVMe SSD**
(c) **Seagate BarraCuda NVMe SSD**
(d) **Silicon Power NVMe SSD**

Using the above NVMe SSDs, we investigated the three core components of our NVMe-Assist framework. We have justified the reason behind choosing the core area for our framework below:

1. **Wear-leveling**: We investigated the effect of wear-leveling and understood the reasons for the change in hash values. The validation and verification of hash values are essential steps in proving data integrity in a court of law. Any change in hash values for any reason can render forensic evidence incompetent. For this reason, we put together a framework to investigate the change on a solid scientific basis.

2. **TRIM functionality**: We examined the TRIM functionality and its effect on deleted data and data recovery. Whenever data is deleted from a storage device using an operating system, it can be recovered if the operating system and hardware storage device provide a provision. Our framework offers assistance in recovering the deleted data from the NVMe SSDs. We also explained the factors involved when the data cannot be recovered or is beyond the scope of recovery and carving if we encounter any such case in our experiment.

3. **Prefetch files, Windows Shellbag, BootPerfDiagLogger.etl**: We deep-dived into the Windows 10 artifact files, Prefetch files, Windows Shellbag, and BootPerfDiagLogger.etl in our framework. We chose these files to investigate how these files behave when used under NVMe SSDs. We made use of proprietary, open-source, and freeware tools to parse the artifacts as shown in Table 2.1:

Table 2.1. The software tools used for analyzing Windows 10 artifacts.

Software Tools Used	
OSForensics	ShellBagsView
CQPrefetch Parser	ETLParser
PECmd	PerfView
Windows PrefetchParserMaster	FullEventLogView
WinPrefetchView	SvclogViewer
Win10 pf Decompression Tool	TraceFMT
ShellBagsExplorer	Windows Performance Analyzer

Outlined below are the reasons behind investigating particular artifacts of Windows 10:

(i) The **Prefetch** files contain valuable evidence of program execution. We analyzed these files in-depth since these files have gone through multiple changes with the development and subsequent updates of the Windows 10 operating system. Additionally, they contain valuable information if any nefarious activities have been performed to cover up any potential wrongdoings. In addition, we analyzed useful timeline information based on reverse-engineering the prefetch files.

(ii) We forensically analyzed the **Windows Shellbag** to find the user preference settings. The files are designed to hold information about a user's preferences while browsing folders, which can contain helpful incriminating evidence.

(iii) The **BootPerfDiagLogger.etl** file contains information related to system shutdown and restart. This information is beneficial in studying a user's computer interaction.

The high-level design of our NVMe-Assist framework is shown in the figure below.

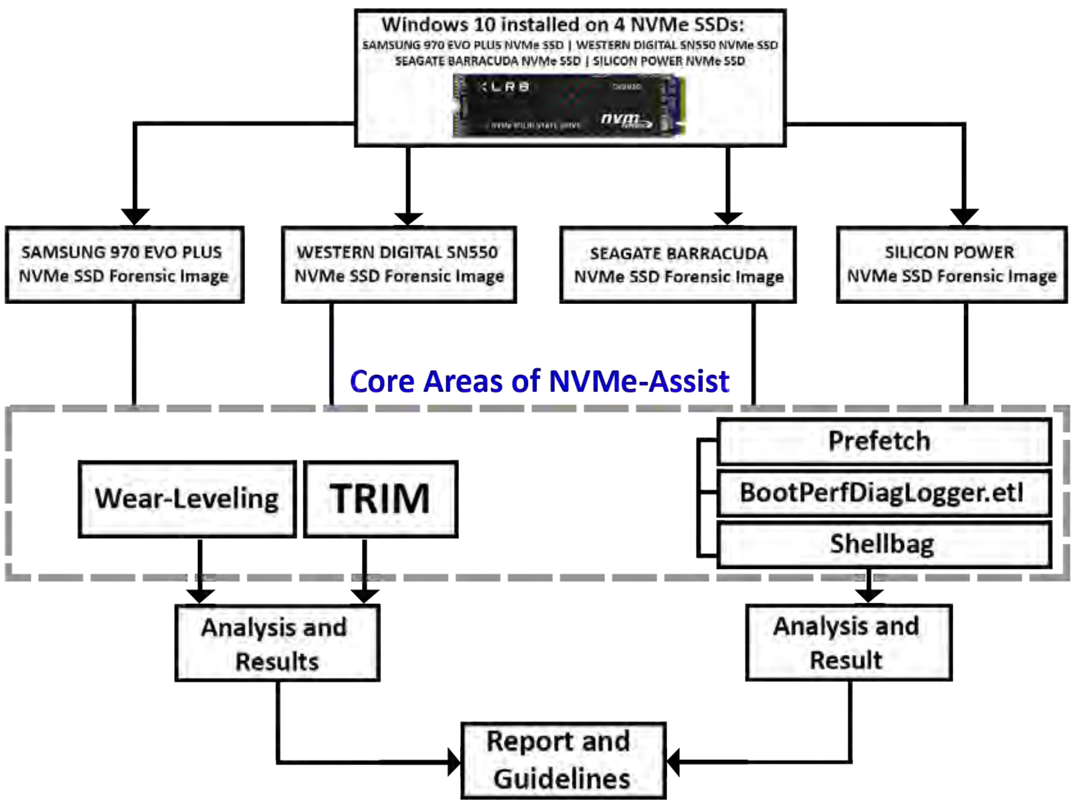


Fig. 2.1. The NVMe-Assist Digital Forensics Framework.

CHAPTER III

Forensics Acquisition and Analysis Briefing

In this chapter, we demonstrated the process of acquiring forensics images of SSDs used in our experiments using the imaging tool **AccessData FTK Imager**. These images were then analyzed using various tools such as **AccessData FTK** and **Autopsy**. This chapter also demonstrates the process for conducting image analysis employed throughout the dissertation. Lastly, this chapter also exhibits image header analysis that we performed using the tool **WinHex** of the acquired e01 format forensic images.

Figures 3.1 to 3.10 demonstrate the forensic image acquisition process using AccessData FTK Imager. Figure 3.1 shows the application screen when the tool is first initiated. In this step, we ran the FTK imager to acquire an image of an NVMe SSD. As shown in figure 3.2, on the top left of the FTK Imager window, we click on the **"File"** option and select the option **"Create Disk Image"**. We then choose the option of a physical drive as the source evidence type, as demonstrated in figure 3.2. We chose the option of physical acquisition because we wanted to acquire the **Master Boot Record** (**MBR**) and **GUID Partition Table** (**GPT**).

We then selected the drive we wanted to image, as shown in figure 3.4. A destination path for the created forensic image was then entered in the dialog box, after which we selected the format of the image we would like to acquire, as shown in Figures 3.5 and 3.6, respectively. Finally, we selected the e01 format, which stands for EnCase evidence file. It is a commonly used format for imaging, and it offers file compression. We then proceeded by adding the details of the image, such as unique description, examiner, etc., as displayed in figure 3.7. In the final steps, we added the image destination folder, the name of the image file, and specified the value for the image fragment size and compression, as seen in figure 3.8. We did not fragment the image in our case and used a compression value of 6 (optimum value). Once a destination path was added, we proceeded with imaging the drive, as shown in Figures 3.9 and 3.10.

Figures 3.11 to 3.13 demonstrate the steps used in the image analysis procedure in AccessData FTK. Figure 3.11 shows the application window where the different case files (NVMe SSD image files in our case) are listed and the date it was modified. Each case file can be differentiated using the metadata such as modified, accessed and created date, etc., provided by the FTK toolkit. Then, each case file can be further opened, and its contents can be viewed. As displayed in Figures 3.12 and 3.13, a specific image was opened to view the data it contains.

Figures 3.14 to 3.16 demonstrate the steps used in the image analysis procedure in Autopsy. Figure 3.14 shows the application window of Autopsy with an image file opened as shown in the hierarchy under **"Data Sources"**. Figures 3.15 and 3.16 show the image file's contents in great detail. The image header analysis using WinHex is shown in figure 3.17.

Image Acquisition Procedure from FTK Imager

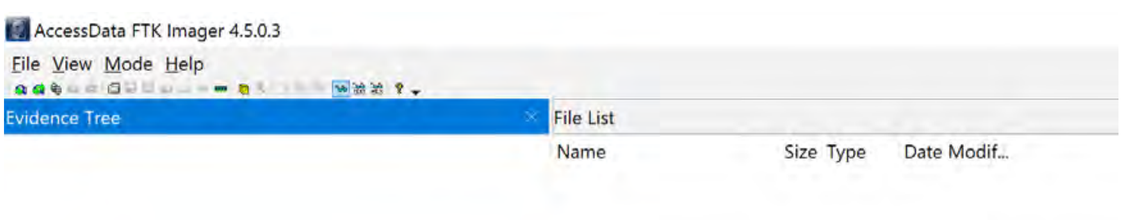


Fig. 3.1. AccessData FTK Imager step 1.

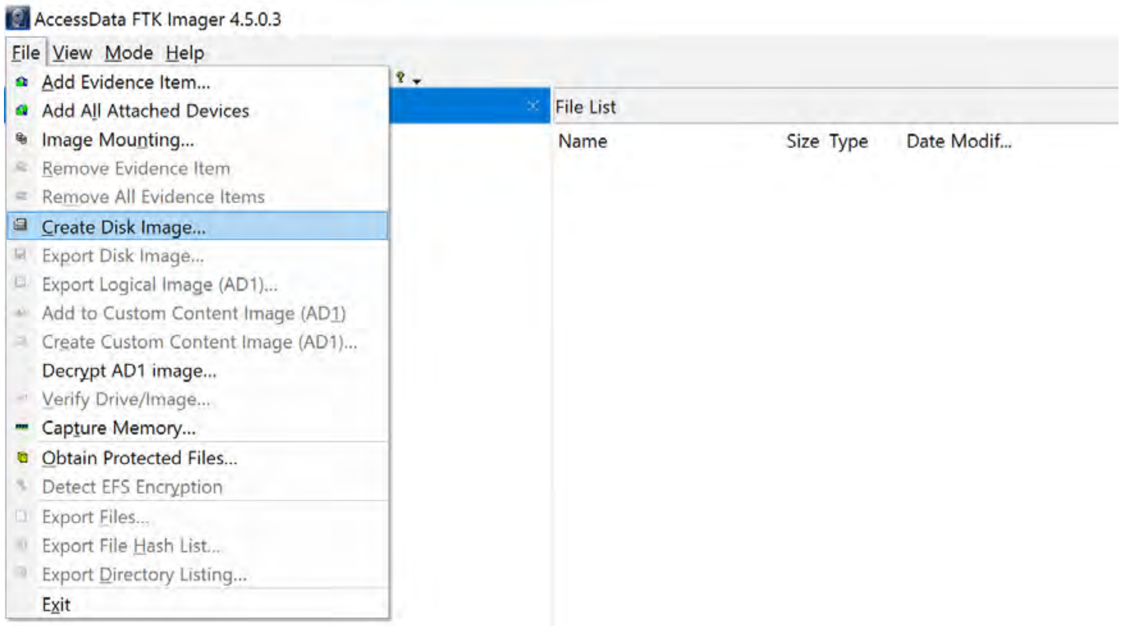


Fig. 3.2. AccessData FTK Imager step 2.

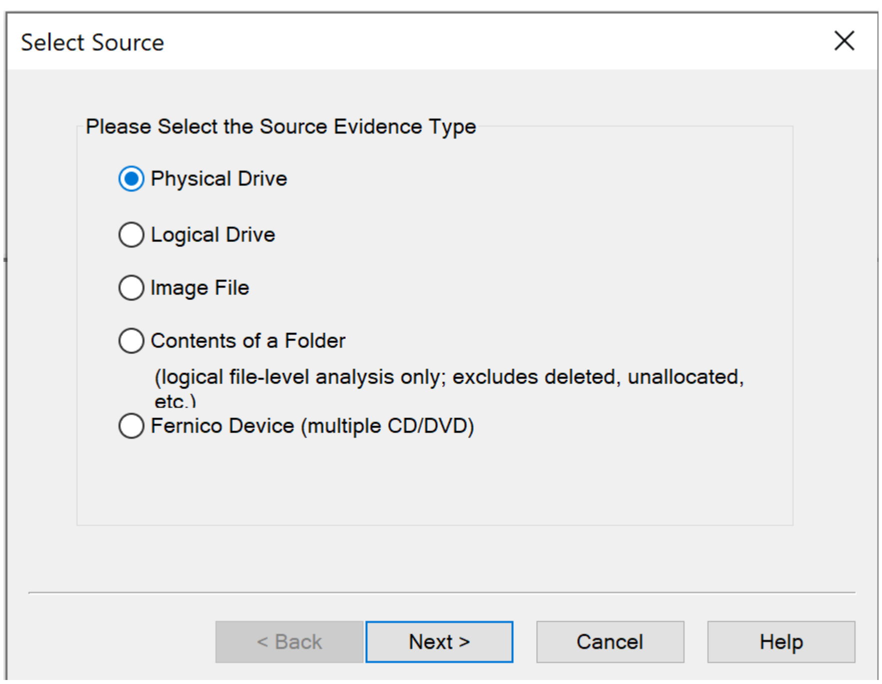


Fig. 3.3. AccessData FTK Imager step 3.

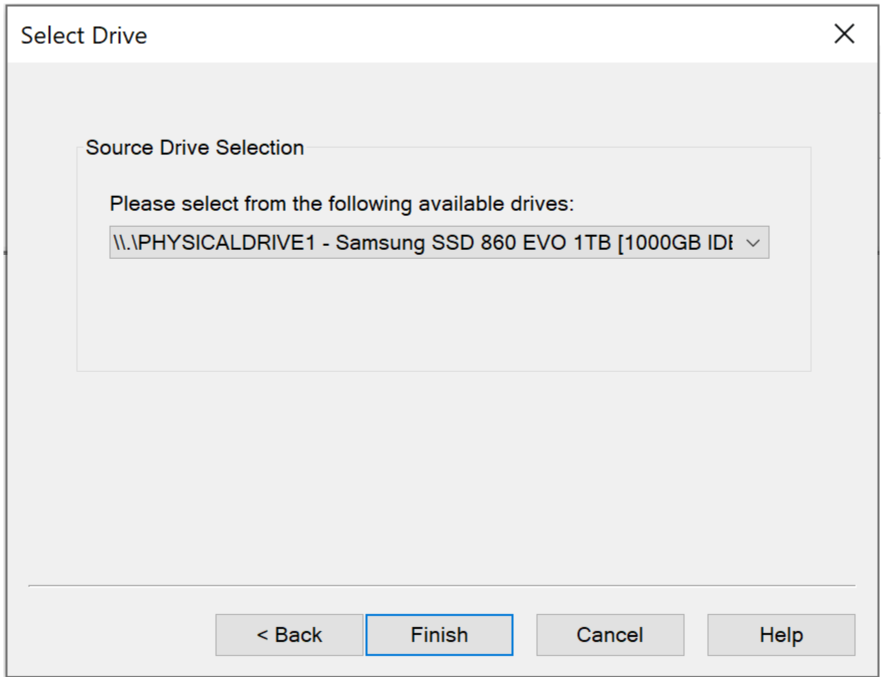


Fig. 3.4. AccessData FTK Imager step 4.

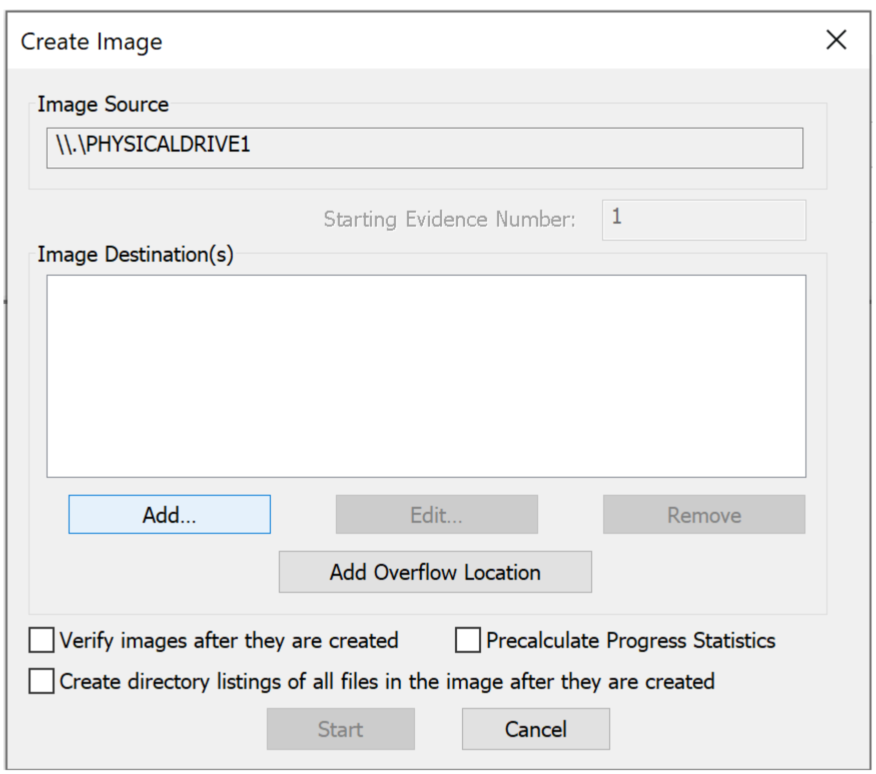


Fig. 3.5. AccessData FTK Imager step 5.

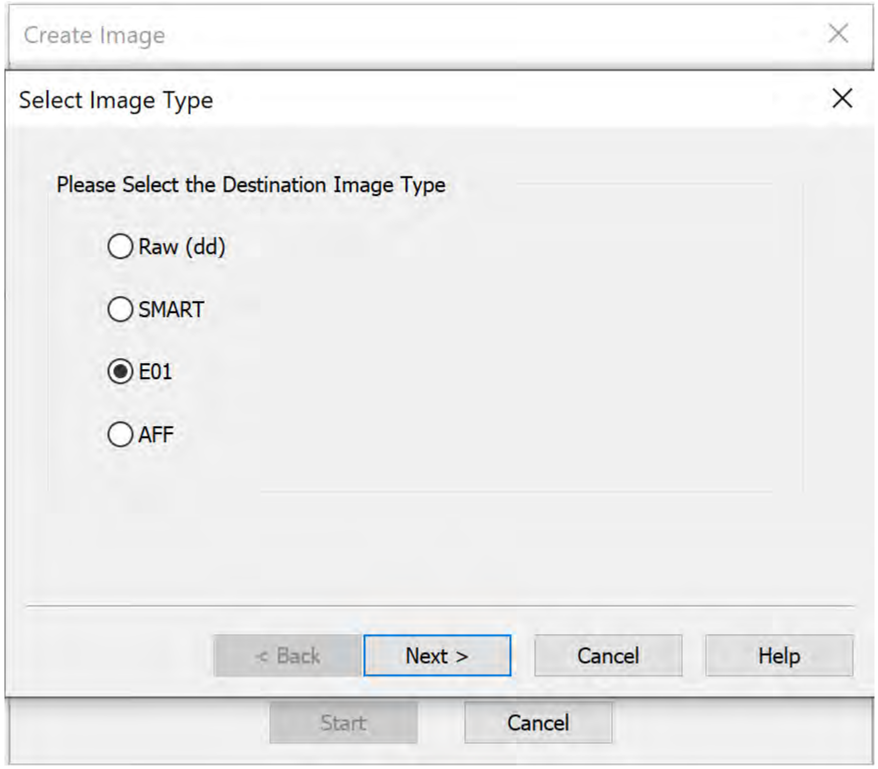


Fig. 3.6. AccessData FTK Imager step 6.

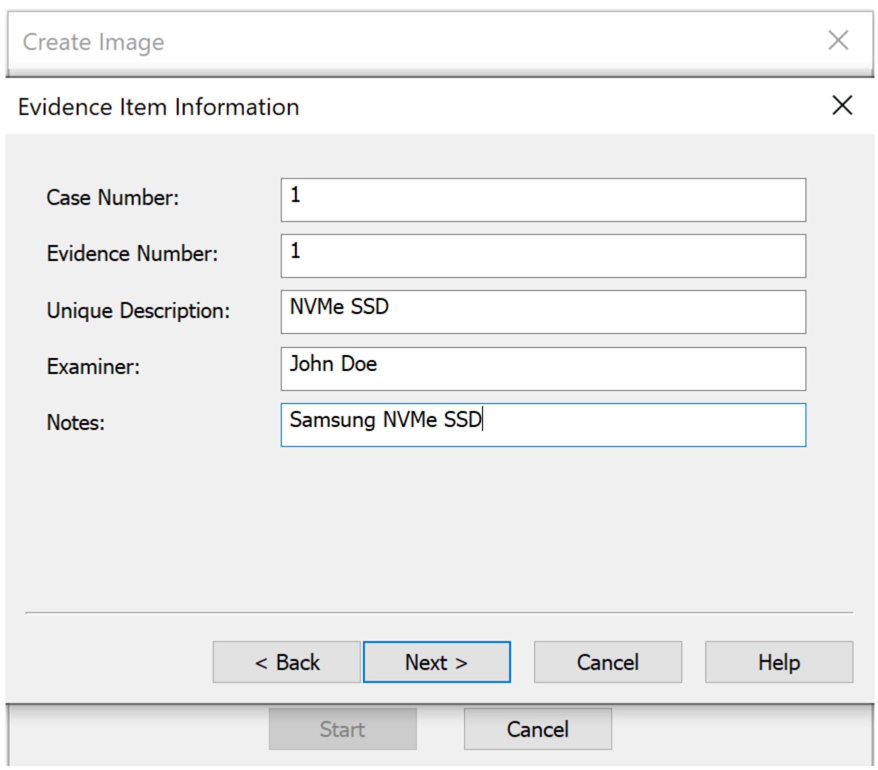


Fig. 3.7. AccessData FTK Imager step 7.

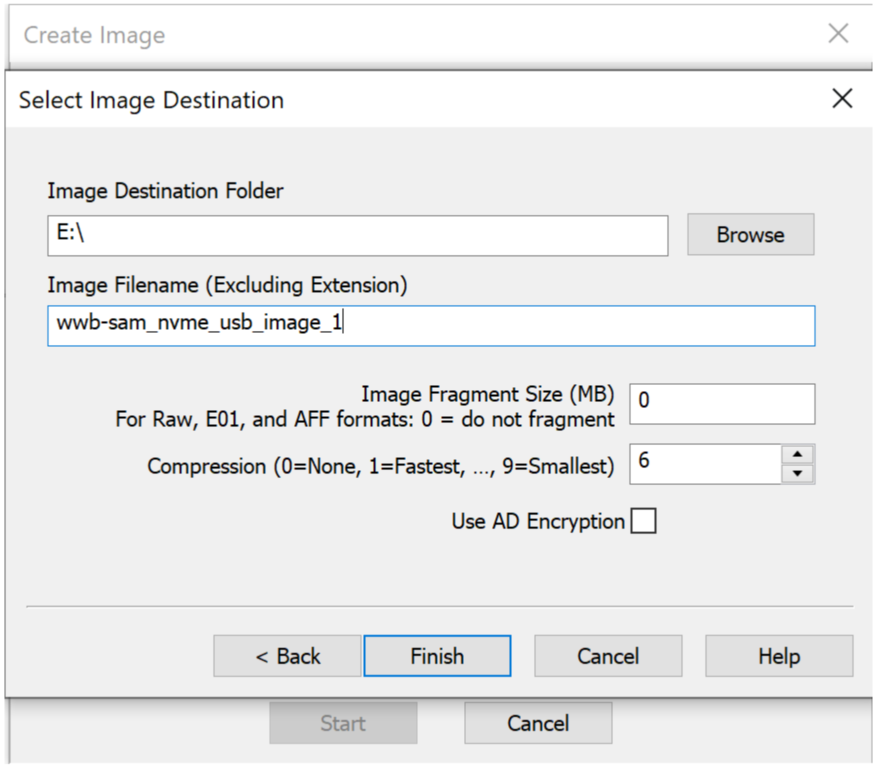


Fig. 3.8. AccessData FTK Imager step 8.

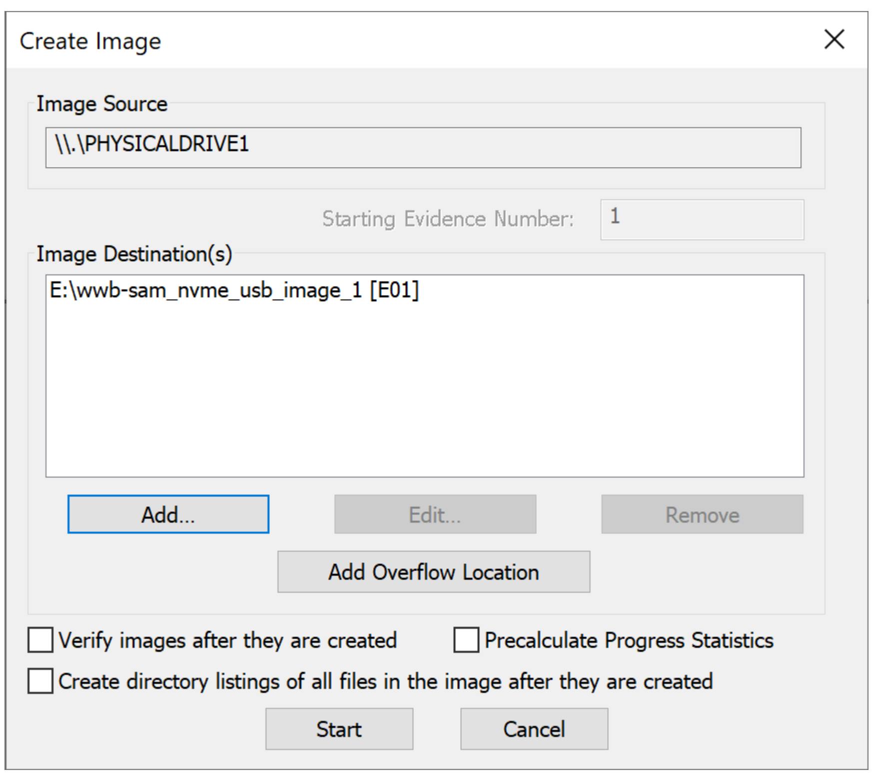


Fig. 3.9. AccessData FTK Imager step 9.

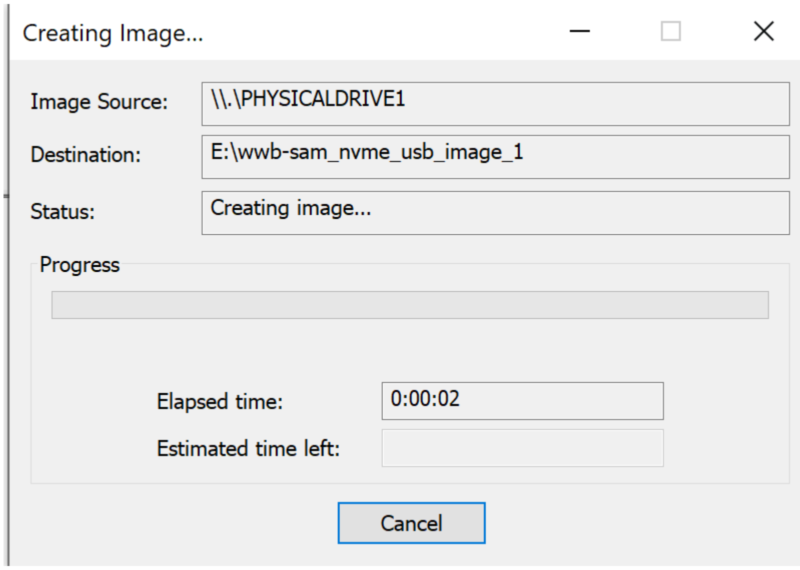


Fig. 3.10. AccessData FTK Imager step 10.

Image analysis procedure from AccessData FTK

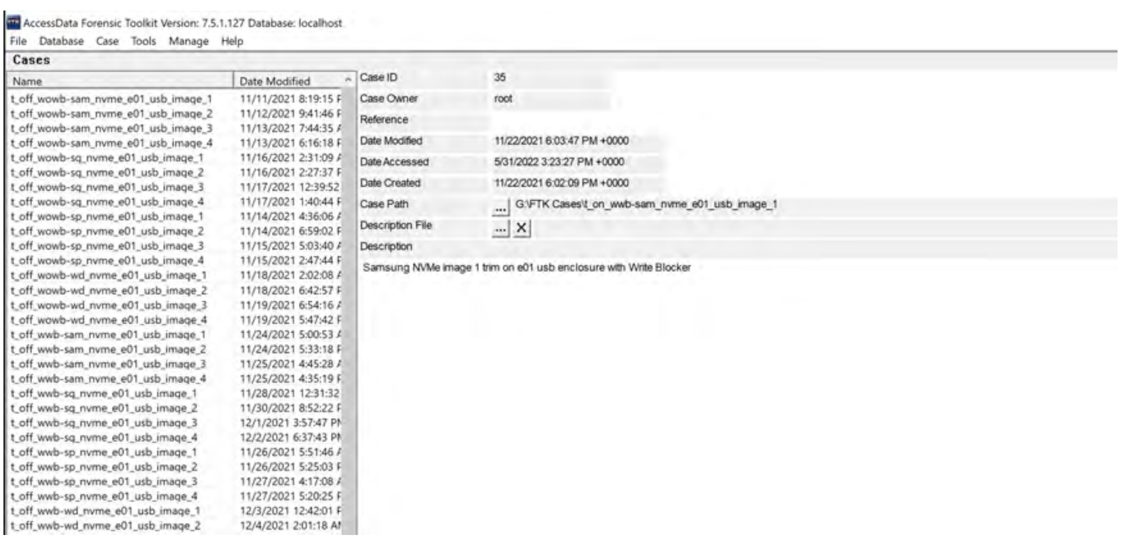


Fig. 3.11. Forensics image analysis in AccessData FTK step 1.

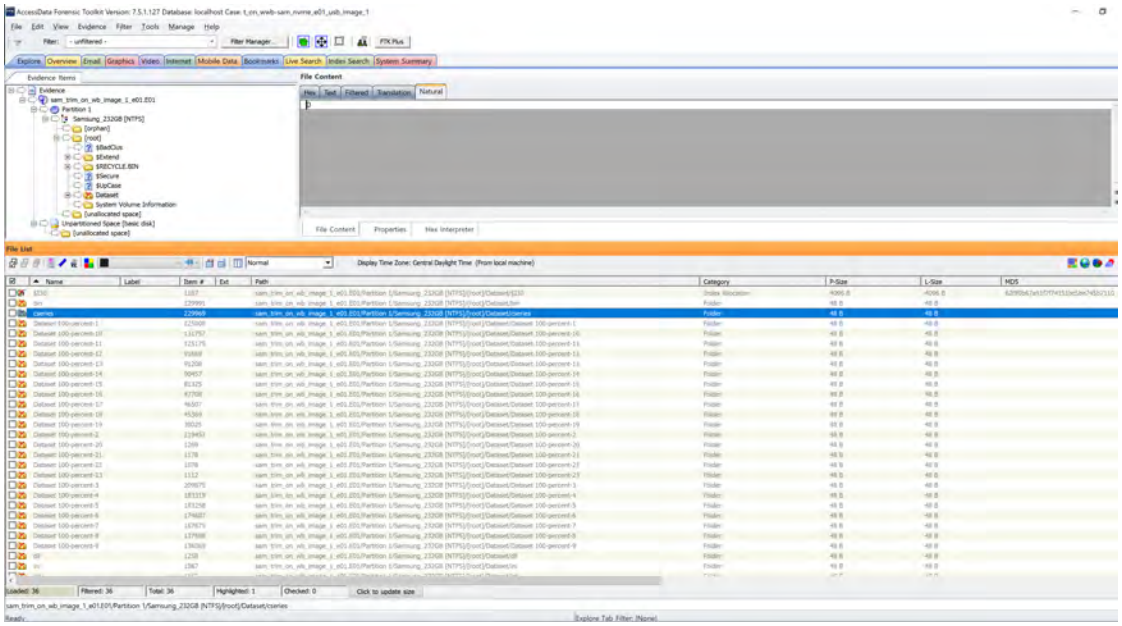


Fig. 3.12. Forensics image analysis in AccessData FTK step 2.

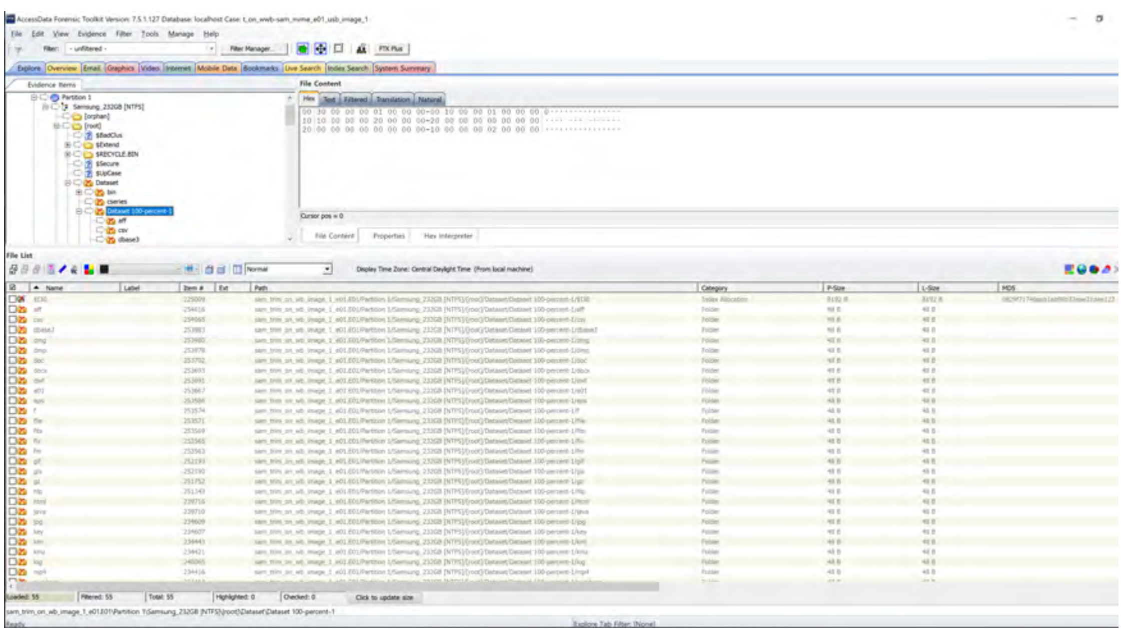


Fig. 3.13. Forensics image analysis in AccessData FTK step 3.

Image Analysis Procedure in Autopsy

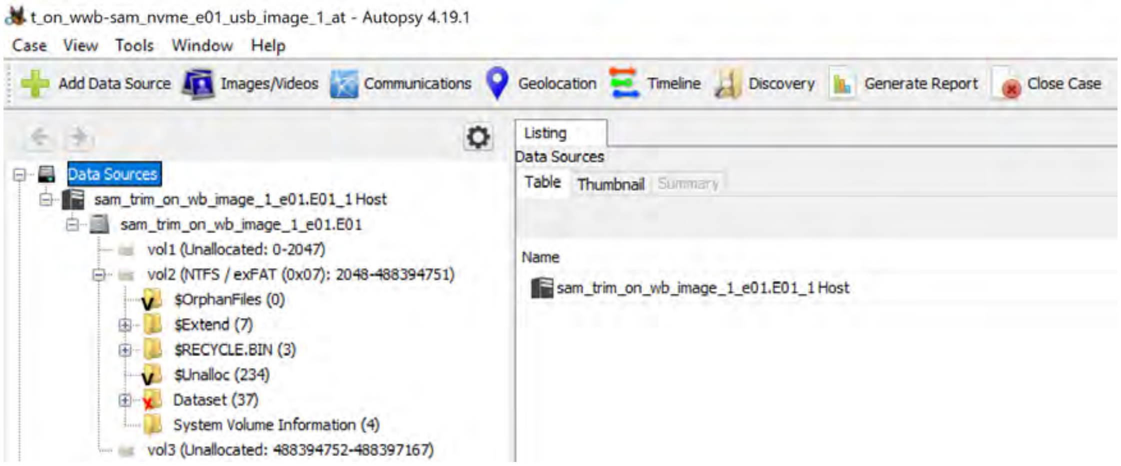


Fig. 3.14. Forensics image analysis in Autopsy step 1.

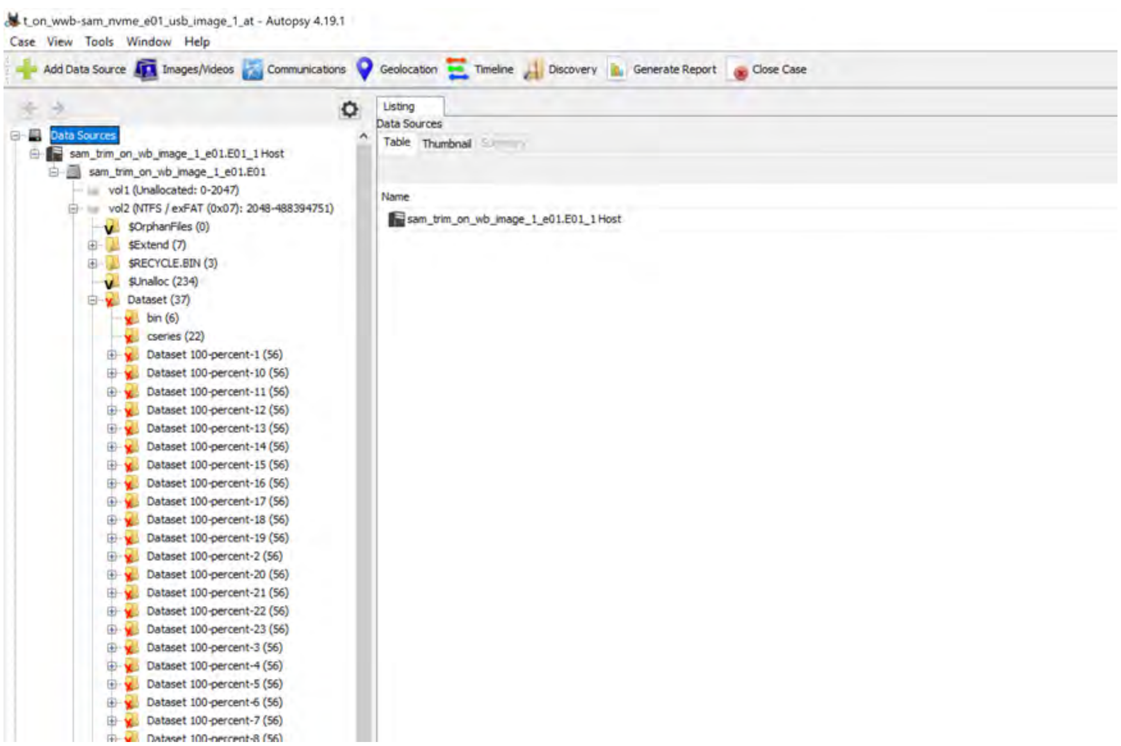


Fig. 3.15. Forensics image analysis in Autopsy step 2.

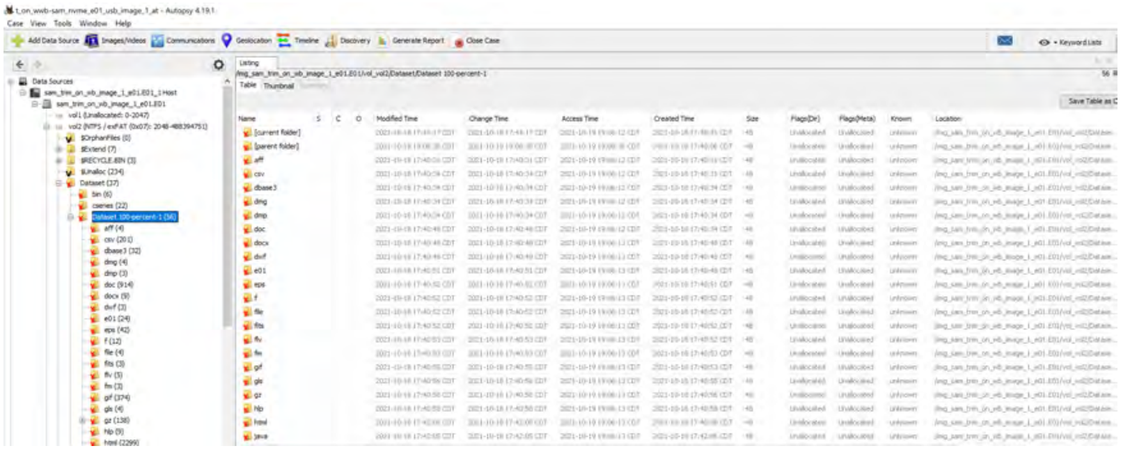


Fig. 3.16. Forensics image analysis in Autopsy step 3.

Image Header Analysis in WinHex

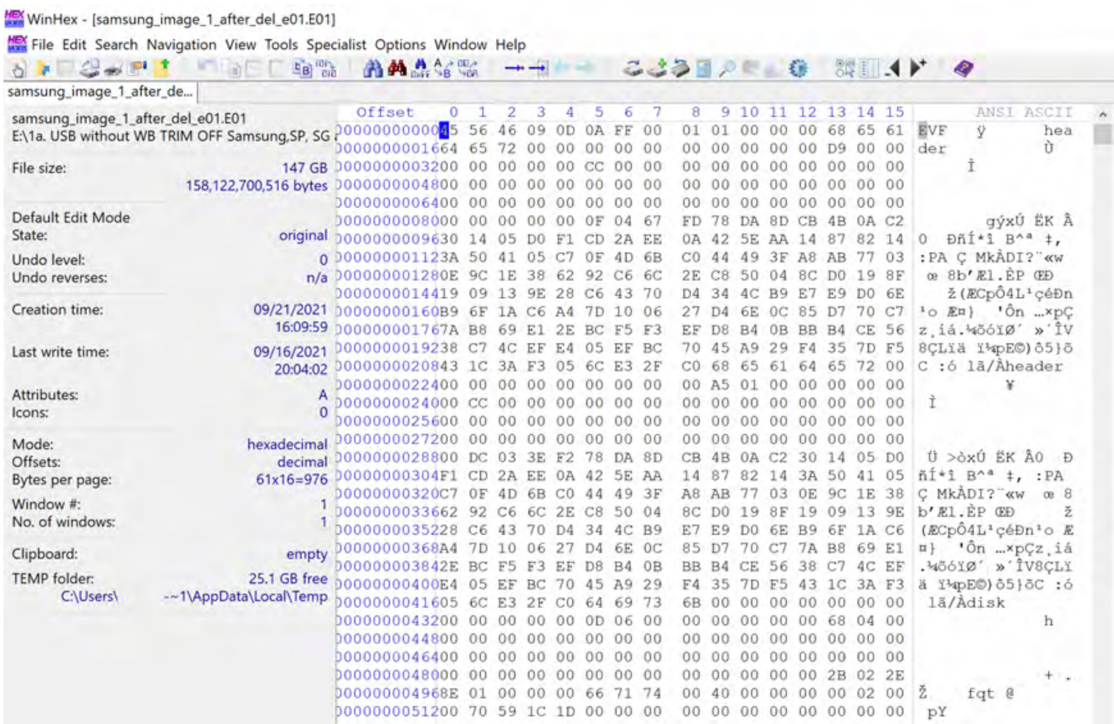


Fig. 3.17. Image header analysis in WinHex.

CHAPTER IV

Windows Prefetch Forensics

Windows Prefetch

Windows operating systems produce a substantial amount of artifacts. These artifacts have high forensics value. These artifacts contain information that can be used as incriminating evidence when conducting digital forensics examinations, thus, they have a high forensics value. One such artifact is the Windows Prefetch file. Windows prefetch or simply Prefetcher is one of the components of the Windows operating systems. It was introduced with the inception of Windows XP in the year 2001 [30].

Prefetcher is a part of the Windows Memory Manager. This component aims to speed up the operating system booting process and lessen the startup time of program applications including stand-alone executables like cmd.exe, conhost.exe, etc. and .COM binaries such as format.com. In order to attain this, Prefetcher caches files to the Random Access Memory (RAM) when an application is launched. Hence, unifying disk reads and thereby lowering disk seeks[1]. Microsoft's Prefetching technology is covered by the US patent: 6,633,968 [55].

With the aim of improving Windows user experience, Microsoft enhanced the algorithm functionality of Prefetch technology from Windows Vista onwards; Superfetch and ReadyBoost have extended it [56]. Superfetch aims to expedite application launch times by observing and adapting to the application pattern of use over time. Hence, it caches most of the dependent files and data needed by the program in advance for speedy access.

---

1 Disk seeks only happen in hard-disk drives (HDDs). The hardware implementation of disk seek is not present in the regular solid-state drives (SSDs) or the newer Non-Volatile Memory Express (NVMe) SSDs.

On the other hand, ReadyBoost uses USB flash memory to extend the system cache memory beyond the memory (RAM) installed on the computer. In addition, the ReadyBoot component of ReadyBoost decreases operating system boot time by preloading dependent booting files and startup programs into cache [55].

In this chapter, we thoroughly analyzed Windows prefetch files from Windows XP till Windows 10. A comparitive illustration has been drawn of the prefetch files in different Windows operating systems. For this purpose, we installed different Windows operating systems in HDDs, SSDs, and NVMe SSDs to draw a comparison and explain our findings in detail. In addition, we demonstrated a few prefetch forensics analysis tools, both open-source and proprietary, for presenting forensics analysis information.

Types of Prefetching

There are two types of prefetching: **Application prefetching** and **boot prefetching** [55]. Application prefetching works by monitoring approximately the first ten seconds of application program startup (4 to 8 seconds in SSDs or NVMe SSDs) to record the useful dependent files and information for future execution of programs. Also, prefetching depends on the size and complexity of the program and storage devices such as HDDs, SSDs, NVMe SSDs. For example, prefetching MATLAB will take longer compared to MS Paint.

In contrast, the boot prefetching observes the necessary files, registry hives, crucial data from the Master File Table (MFT) from the NTFS filesystem. Hence, the future booting process then uses information from the boot prefetch file for swift booting [57].

As far as the boot prefetching is concerned, we observed that the boot trace file which is **NTOSBOOT_B00DFAAD.pf** [55], exists only between

Windows XP and Windows 7. This prefetch file is used to cache the necessary dependencies for faster booting of a computer system working on HDDs. This file also exists on computer systems having SSDs and NVMe SSDs. The name of the file did not change with the different versions of Windows operating systems (XP to 7), i.e., the boot prefetch trace file has the same name without any executable path hash change (prefetch hash is mentioned in detail later in the chapter). However, Microsoft has removed this file from Windows 8 onwards, i.e., it is not found in the Prefetch file directory located in **C:\Windows** [55].

We have shown the availability of **NTOSBOOT_B00DFAAD.pf**[2] in different Windows operating systems in table 4.1 . Figures 4.1 and 4.2 show the NTOSBOOT_B00DFAAD.pf from Windows XP to Windows 8.

Table 4.1. The availability of NTOSBOOT_B00DFAAD.pf file in different Windows versions.

NTOSBOOT_B00DFAAD.pf	XP	Vista	7	8	8.1	10
in Hard-disk drives (HDDs)	✓	✓	✓	×	×	×
in Solid-state drives (SSDs)	×	✓	✓	×	×	×
in NVMe SSDs	✘	✘	✘	×	×	×

Task Scheduler needs to run for prefetch to work. It is responsible for parsing the trace data collected by the Prefetcher and writing files to the prefetch directory [55] [56].

In addition, the Prefetcher is enabled by default from Windows Vista to Windows 10. Moreover, in our experiment we have seen that the service name of Prefetcher is **Superfetch** from Windows Vista to 8.1. However, Microsoft changed the service's name to **SysMain** from Windows 10 onwards. The

[2] **Note**: Windows XP cannot be installed on an SSD or NVMe SSD, so the presence of the file could not be confirmed. Also, Windows Vista, 7, and 8 cannot be installed on an NVMe SSD, so the existence of NTOSBOOT_B00DFAAD.pf cannot be established.
✘ - NVMe (Non-Volatile Memory Express) SSD is not natively supported while installing the operating system.

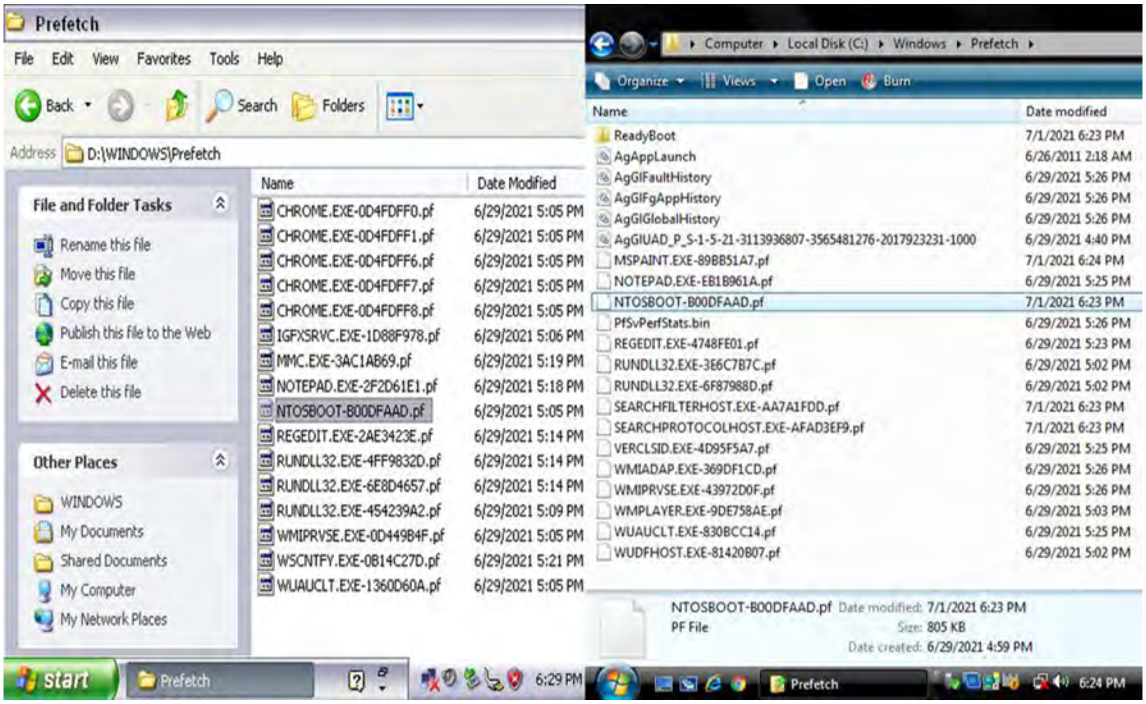


Fig. 4.1. The presence of NTOSBOOT_B00DFAAD.pf in Windows XP and Vista.

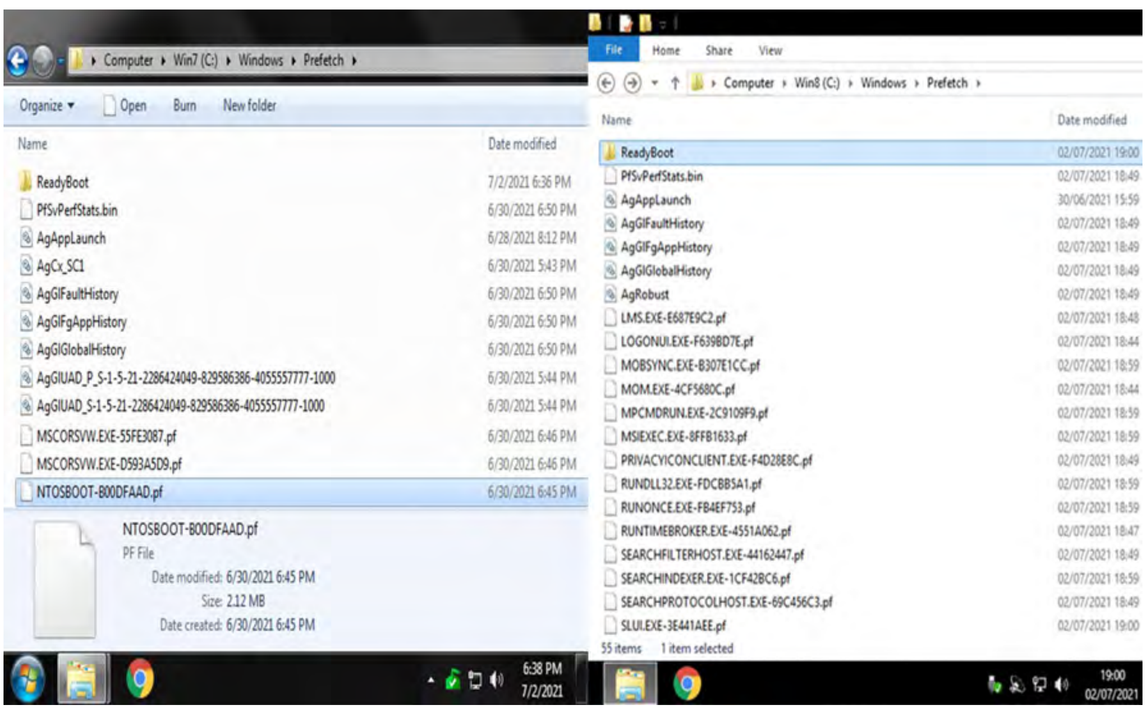


Fig. 4.2. The presence of NTOSBOOT_B00DFAAD.pf in Windows 7 compared to lack of it in Windows 8.

prefetcher service can be found in the Service snap-in of Windows, which can be invoked by typing **services.msc** in the Run box or from the Control Panel's Administrative Tools.

Also, contrary to the popular belief, prefetch is not disabled by default in SSDs or NVMe SSDs since we have confirmed this fact by installing Windows in both types of SSDs. Moreover, application prefetch in SSDs is enabled by default, but boot prefetch is not. The reason is that SSDs are faster than HDDs, so there is no need for boot prefetching. In addition, SSDs do not perform mechanical disk seeks, unlike HDDs.

Prefetch Storage Location

As briefly mentioned earlier, prefetch files are stored in **C:\Windows\Prefetch**. The prefetch file name is always in uppercase and there can be only 128 files in the Prefetch folder from Windows XP, Vista and 7. From Windows 8 onwards, however, the maximum number of prefetch files in the Prefetch folder is up to 1024 [56]. If the limit of prefetch files in the prefetch directory is reached, the oldest prefetch files are removed first. Additionally, it is beneficial to note that whenever a program is uninstalled from a system, its associated pf file is not deleted from the Prefetch folder.

Prefetch Naming Scheme

The naming scheme for the prefetch file is as follows, which is in order: *Application name, a dot, EXE followed by a dash or hyphen, 8-letter hexadecimal prefetch algorithm hash of the program's full path of execution, and a .pf extension* [56], [22]. **For example:** CHROME.EXE-039F1FCB.pf, CALCULATOR.EXE-DD323BEE.pf

In addition, we subtantiated a claim made in [57] and observed that when we open a program from command prompt (cmd) or execute it directly, the computation of the prefetch file is not affected, i.e., there will not be two different prefetch files for the same program executed differently. However, there are certain exceptions to this assertion. For instance, chrome.exe, svchost.exe, dllhost.exe, mmc.exe, and rundll32.exe, in particular, will have different prefetch files generated based on different command line parameters and functions requirements by these executables. For different functions, a different set of dependency files are needed. In addition, variations in both cases and spaces within the parameters and path will also affect the prefetch hash[3] [57].

Prefetch Hash Algorithm Generation Steps

– Full path for file is determined (e.g. C:\Windows\Notepad.exe).
– Path is converted to a unicode string.
– Path is converted to a device path.
  (e.g., \DEVICE\HARDDISKVOLUME*x*\WINDOWS\NOTEPAD.EXE. *x* is some volume number given by the operating system.)
– Prefetch hashing is applied.
– Prefetch filename is generated (e.g. NOTEPAD.EXE-XXXXXXXX.pf).

Prefetch Configuration

Prefetch configuration is stored in the Windows Registry at:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters

From our experimental analysis, we noted that the Registry path is the same across all the Windows versions, i.e., from Windows XP to Windows 10. Figure 4.3 shows the registy path of Prefetch in Windows 10.

3 **Note**: Prefetcher is only enabled on Windows workstations by default, and not on server computers.

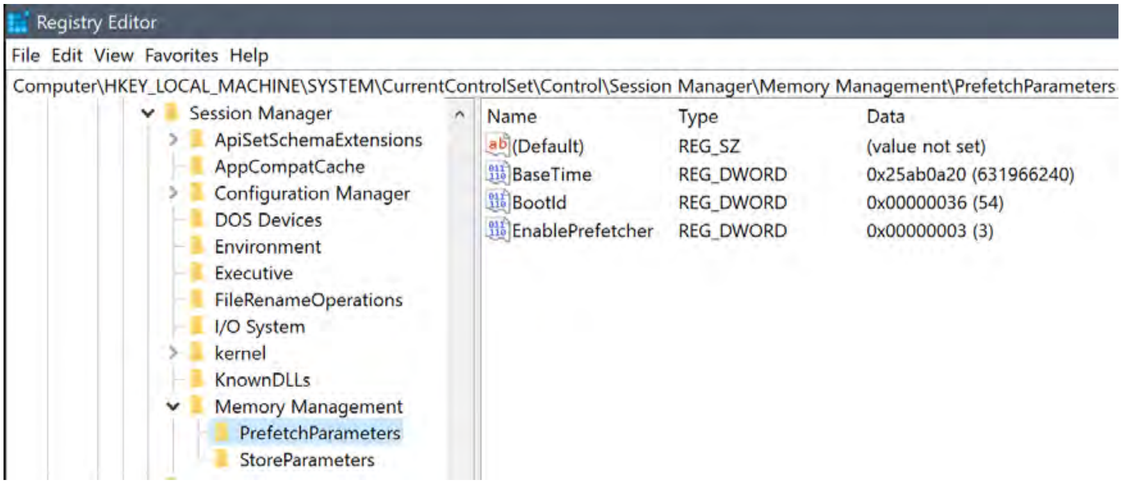


Fig. 4.3. The Prefetch configuration in Windows 10 Registry Editor.

The **EnablePrefetcher** entry (REG_DWORD i.e. Registry DWORD value), a 32-bit value that can be set to one of the following as shown in table 4.2 [4] for modifying Prefetcher setting:

Table 4.2. The EnablePrefetch REG_DWORD values for selection.

Value (in decimal)	**Value** (in hexadecimal)	**Action**
0	0x 00 00 00 00	Disabled
1	0x 00 00 00 01	Application prefetching enabled
2	0x 00 00 00 02	Boot prefetching enabled
3	0x 00 00 00 03	Application and Boot prefetching enabled

The **EnablePrefetcher** registry value is shown as **big-endian** in the registry as seen in the figure 4.3 (upper right side of the figure). However, when we right-click **EnablePrefetcher** value and then click on ***Modify Binary Data***, we see the value shown as **little-endian**. This can be seen from the figure 4.4.

[4] **Note**: The little-endian hexadecimal is displayed as **0x 03 00 00 00**, while big-endian hexadecimal is displayed as **0x 00 00 00 03**

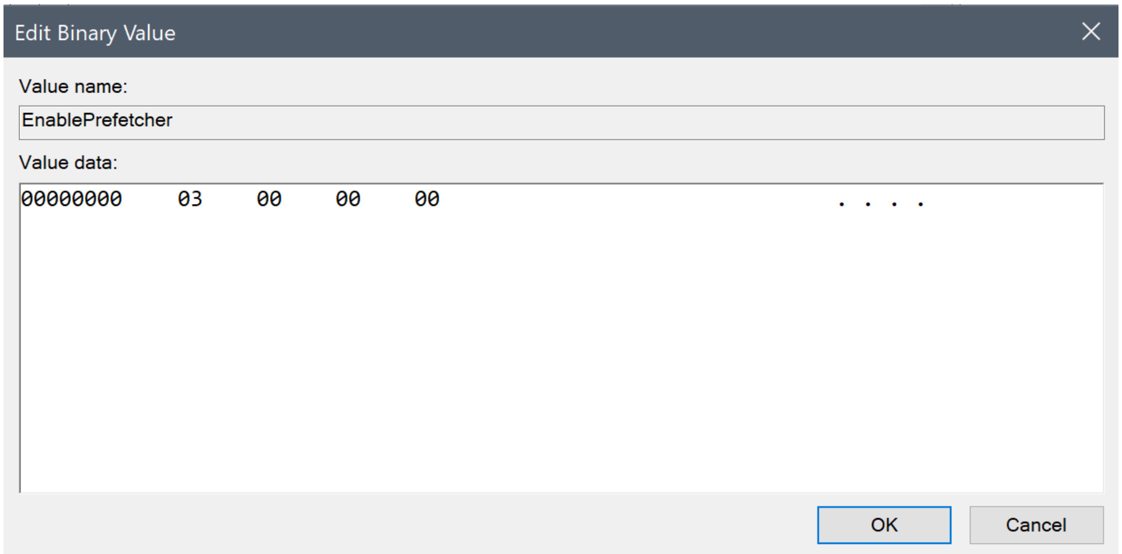


Fig. 4.4. The little-endian representation of EnablePrefetch entry.

Contents of Prefetch Files

Prefetch files keep track of programs that have been executed on the system even if the original application program is no longer present. These files can specifically tell us when the program was executed and the number of times it was executed [22]. It also gives us the path of the execution of the program and files loaded by the application in the first ten seconds of program execution, in HDDs (or nearly first four to eight seconds in SSDs and NVMe SSDs). Below are some of the basic contents of a Prefetch file that are useful while conducting forensic analysis [57].

1. Name of the executable (.exe) or program executable.
2. A unicode list of DLL (dynamic link library) used by the program.
3. A count of how many times the executable ran (program run counter).
4. Last run timestamp of the executable (last run time).
5. Prefetch file size.
6. Program executable path in the operating system.
7. Created time of the prefetch file.

8. Modified time of the prefetch file.
9. Path of DLL described as device path.

Signature of Prefetch Files

A prefetch file has a **4-byte** signature, **"SCCA"** or in hexadecimal **0x53 0x43 0x43 0x41** starting at offset 4, when viewed in any hex-editor tool such as WinHEX, HexWorkshop, HxD, etc [22]. This signature can only be seen as **"SCCA"** up to Windows 8.1. From Windows 10 onwards, the prefetch files are compressed with the **XPRESS HUFFMAN** algorithm. Therefore, the signature of the compressed prefetch file is **"MAM"** [30]. The compressed prefetch file needs to be decompressed[5] before it can be read for forensic analysis.

Prefetch File Header

For deep-diving into prefetch file header analysis, we used WinHex hex editor tool and noted some interesting forensics information. The prefetch file header is **84 bytes** long [59] and consists of the following information shown in table 4.3. The length of file header is the same across all the Windows versions, i.e., from Windows XP to Windows 10.

Operating System Version Based on Prefetch Files

The prefetch file indicates which version of the Windows operating system the prefetch file belongs to. Windows version can be determined from the offset 0 to 3 when viewed in any hex editor tool. Table 4.4 lists out the Windows version from the prefetch file[6].

---

5 The Windows API responsible for decompressing MAM file is RtlDecompressBuffer. Also, a python code by **Francesco Picasso** [58] can help in decompressing the Windows 10 prefetch file. It is hosted on GitHub under the code page named **Windows 10 Prefetch (native) Decompress**. The python file is named "w10pfdecomp.py". **Link to the GitHub page**: https://gist.github.com/dfirfpi/113ff71274a97b489dfd

6 **Note**: The **Signature value (in hexadecimal)** column in table 4.4 is in big-endian. The value in hex-editor shown in subsequent figures, 4.5, 4.6, 4.7, 4.8, 4.9 and 4.10, is in litte-endian.

Table 4.3. Prefetch file header.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of Information**
0x 00 00	4	Format version
0x 00 04	4	Signature "SCCA"
0x 00 08	4	Can be considered BOOTLDR (bootloader) version **0F = BOOTLDR Version 5.0 for Windows XP &** **11 = BOOTLDR Version 6.0 for Windows Vista & above**
0x 00 0C	4	Prefetch file size
0x 00 10	60	Name of the executable
0x 00 4C	4	Prefetch file hash
0x 00 50	4	NTOSBOOT_B00DFAAD identifier. 0 for all prefetch files and 1 for NTOSBOOT_B00DFAAD.pf file **Note**:(NTOSBOOT_B00DFAAD.pf present in XP/ /Vista/7 only)

Table 4.4. Windows version from prefetch file.

Signature Value (in decimal)	**Signature Value** (in hexadecimal)	**Windows version**
17	0x 00 00 00 11	Windows XP
23	0x 00 00 00 17	Windows Vista/7
26	0x 00 00 00 1a	Windows 8
26	0x 00 00 00 1a	Windows 8.1
30	0x 00 00 00 1e	Windows 10

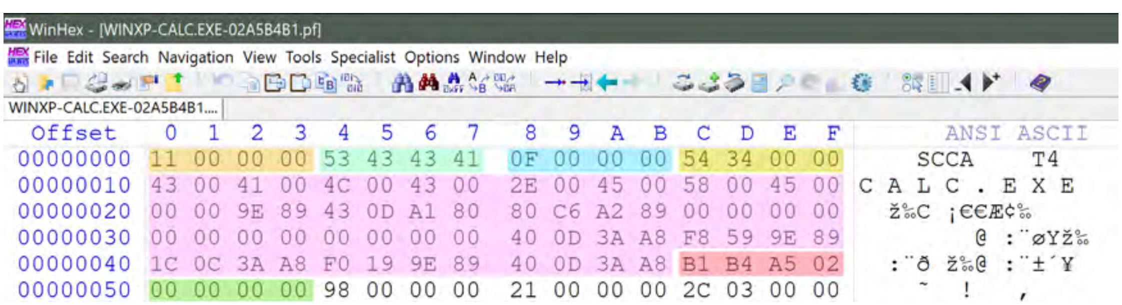


Fig. 4.5. The Windows XP prefetch file header.

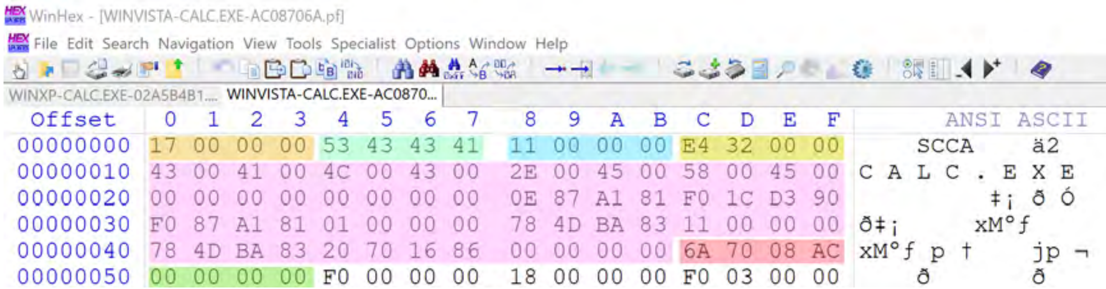


Fig. 4.6. The Windows Vista prefetch file header.

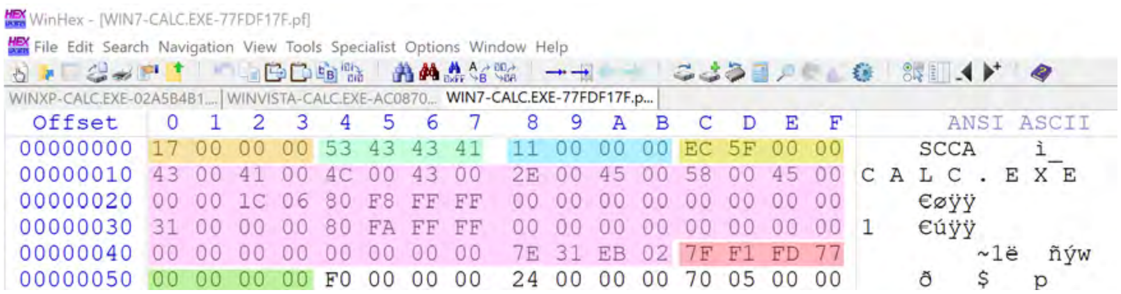


Fig. 4.7. The Windows 7 prefetch file header.

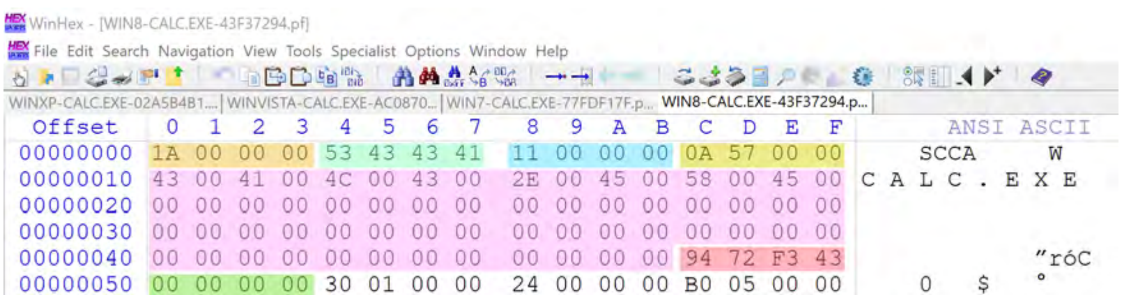


Fig. 4.8. The Windows 8 prefetch file header.

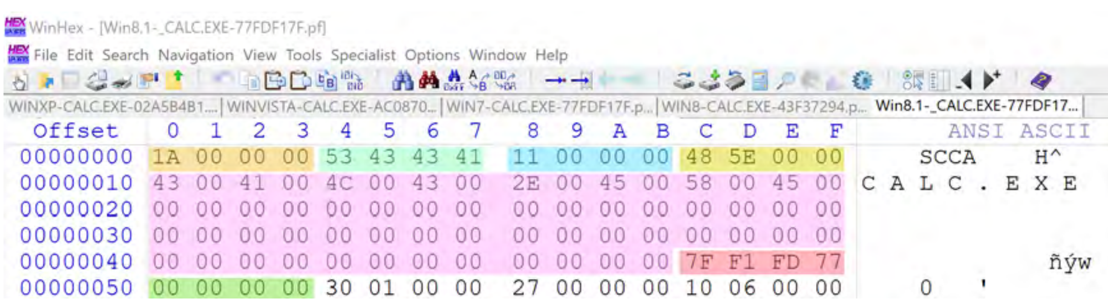


Fig. 4.9. The Windows 8.1 prefetch file header.

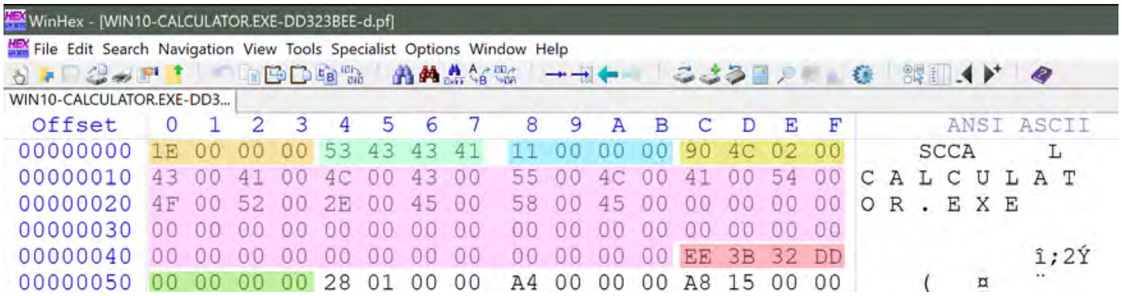


Fig. 4.10. The Windows 10 prefetch file header after decompression.

File Information from Prefetch File

After calculating offsets for sections A, B, C, and D, from the tables talked in subsequent pages, navigate to the calculated offset from the beginning. For example: When we go to offset 0x 00 54 to calculate section A's offset, after finding the actual offset number for section A, navigate to section A's offset from the offset 0x 00 00. The above sections, A, B, C, and D, are used to find useful forensics information from prefetch files.

For Windows XP: The file information in the Windows XP prefetch file is **68 bytes** in size. However, there is certain information in the prefetch file that is unknown or unresolved [59]. Therefore, for forensics relevancy, we have described only the important information in table 4.5.

For Windows Vista and 7: The file information in Windows Vista and 7 prefetch file is **156 bytes** in size. However, there is certain information in the prefetch file that is unknown or unresolved [59]. Therefore, for forensics relevancy, we have described only the important information in table 4.6.

For Windows 8 and 8.1: The file information in Windows 8 and 8.1 prefetch file is **224 bytes** in size. However, there are certain information in the prefetch file that is unknown or unresolved [59]. Therefore, for forensics relevancy, we have described only the important information in table 4.7.

For Windows 10: The file information in Windows 10 prefetch file is of **224 bytes** in size. However, there are certain information in the prefetch file that is unknown or unresolved. Moreover, we have analyzed the prefetch file in Windows 10 version 21H1 at the time of conducting our experiment [59]. Therefore, only relevant forensics information pertaining to Windows 10 v21H1 prefetch file has been described in table 4.8.

Table 4.5. Windows XP file information in prefetch file.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of Information**
0x 00 54	4	Offset to section A. **Note: The offset is relative from the start of the file.**
0x 00 58	4	The number of entries in section A.
0x 00 5C	4	Offset to section B. **Note: The offset is relative from the start of the file.**
0x 00 60	4	The number of entries in section B.
0x 00 64	4	Offset to section C. **Note: The offset is relative from the start of the file.**
0x 00 68	4	Length of section C.
0x 00 6C	4	Offset to section D. **Note: The offset is relative from the start of the file.**
0x 00 70	4	The number of entries in section D
0x 00 74	4	Length of section D
0x 00 78	8	Latest execution time/ run time of executable. **Note: Only one run-time observed in Windows XP.**
0x 00 90	4	Execution counter of the program.

Table 4.6. Windows Vista/7 file information in prefetch file.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of Information**
0x 00 54	4	Offset to section A. **Note: The offset is relative from the start of the file.**
0x 00 58	4	The number of entries in section A.
0x 00 5C	4	Offset to section B. **Note: The offset is relative from the start of the file.**
0x 00 60	4	The number of entries in section B.
0x 00 64	4	Offset to section C. **Note: The offset is relative from the start of the file.**
0x 00 68	4	Length of section C.
0x 00 6C	4	Offset to section D. **Note: The offset is relative from the start of the file.**
0x 00 70	4	The number of entries in section D
0x 00 74	4	Length of section D
0x 00 80	8	Latest execution time/ run time of executable. **Note: Only one run-time observed in Windows Vista/7.**
0x 00 98	4	Execution counter of the program.

Table 4.7. Windows 8/8.1 file information in prefetch file.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of** **Information**
0x 00 54	4	Offset to section A. **Note: The offset is relative from the start of the file.**
0x 00 58	4	The number of entries in section A.
0x 00 5C	4	Offset to section B. **Note: The offset is relative from the start of the file.**
0x 00 60	4	The number of entries in section B.
0x 00 64	4	Offset to section C. **Note: The offset is relative from the start of the file.**
0x 00 68	4	Length of section C.
0x 00 6C	4	Offset to section D. **Note: The offset is relative from the start of the file.**
0x 00 70	4	The number of entries in section D
0x 00 74	4	Length of section D
0x 00 80	8	Latest execution time/ run time of executable.
0x 00 88	56 (8 bytes x 7)	Older/most recent 7 execution/ run times of executable. **Note: 7 run-times observed in Windows 8/8.1.**
0x 00 D0	4	Execution counter of the program.

Table 4.8. Windows 10 file information in prefetch file.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of Information**
0x 00 54	4	Offset to section A. **Note: The offset is relative from the start of the file.**
0x 00 58	4	The number of entries in section A.
0x 00 5C	4	Offset to section B. **Note: The offset is relative from the start of the file.**
0x 00 60	4	The number of entries in section B.
0x 00 64	4	Offset to section C. **Note: The offset is relative from the start of the file.**
0x 00 68	4	Length of section C.
0x 00 6C	4	Offset to section D. **Note: The offset is relative from the start of the file.**
0x 00 70	4	The number of entries in section D
0x 00 74	4	Length of section D
0x 00 80	8	Latest execution time/ run time of executable.
0x 00 88	56 (8 bytes x 7)	Older/most recent 7 execution/ run times of executable. **Note: 7 run-times observed in Windows 10.**
0x 00 C8	4	Execution counter of the program.

Forensics Information in Sections A, B, C, D, and F from a Prefetch File

The data structure lengths of sections A and B depend on the Windows operating system version. While, the length of sections C, D, and F depend on the application size of the operating systems. We can find information like files referenced, directories referenced, volume serial number, volume creation date, device path, etc. Calculating the offsets for sections A, B, C, D, E, and F will be the same across all the versions of Windows' prefetch files.

Section A and **Section B:** Information in Section A will give us details about start time and duration of run of the application in millisecond (ms). Also, filename string and filename string number of characters without end-of-character string and lastly, NTFS file reference (only in prefetch files of Windows Vista and higher) can be obtained [59]. Information in section B talks about trace chain array which is responsible for calculating next array entry index and number of blocks loaded [59].

Section C: Information in Section C will give us details about files referenced by the application. We jump to the starting offset of section C after calculating it from the offset number **0x 00 64** from all the versions of the Windows prefetch file. Reaching the offset of section C will show the relevant information regarding the files referenced. Figures 4.11 and 4.12 show the snippet of Section C information from Windows XP and Windows 10 prefetch files [59].

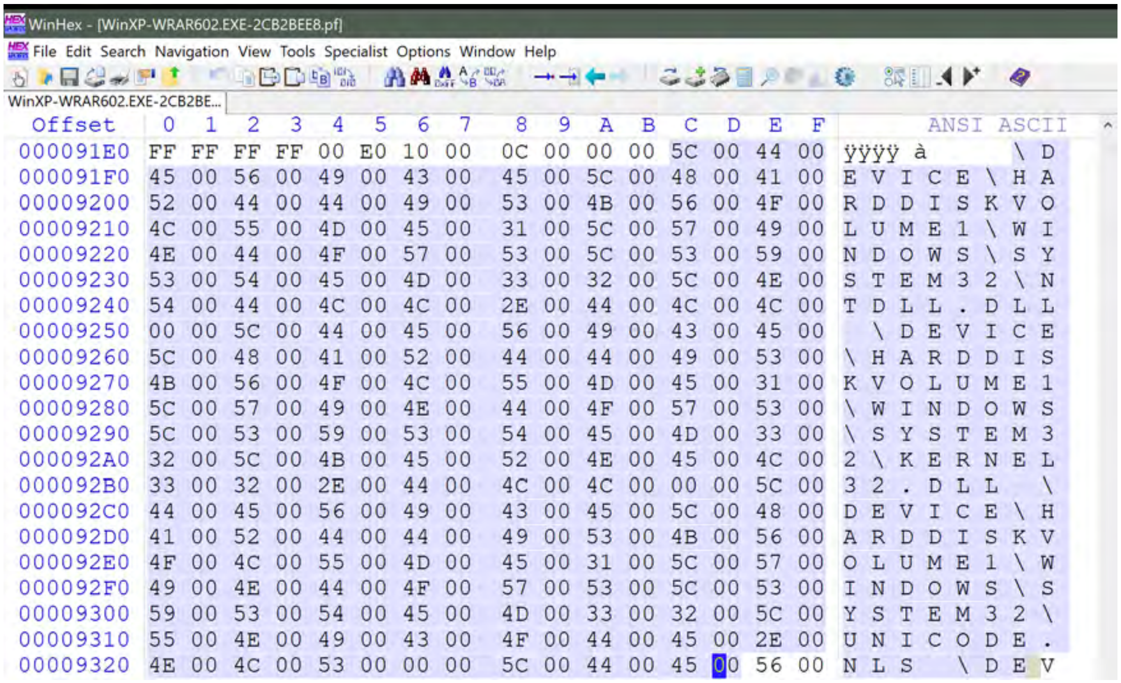


Fig. 4.11. The files referenced information from Windows XP prefetch file.

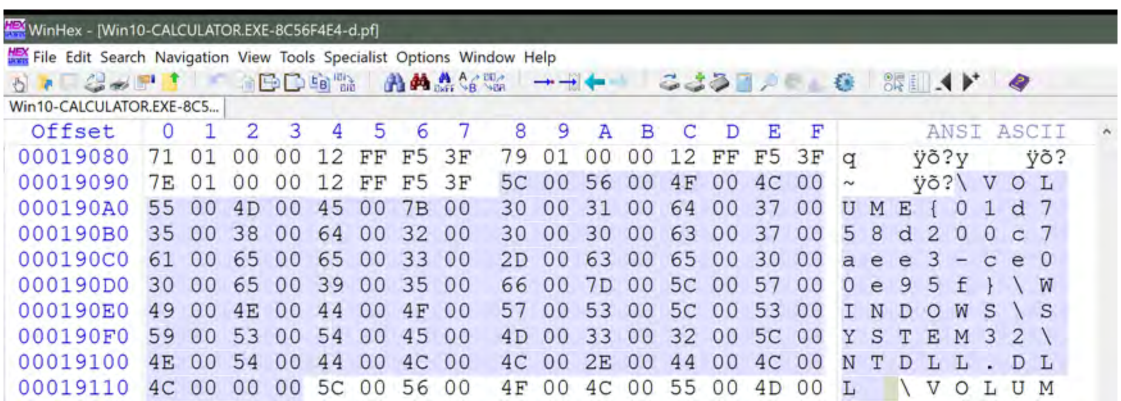


Fig. 4.12. The files referenced information after decompressing Windows 10 prefetch file.

Section D and Section F: Information in Section D will give us details about offset to device path, length of volume device path, volume creation time, and volume serial number. Additionally, Section D gives information regarding offset to Section E and Section F. Whereas Section F will give information such as the number of characters of the directory name and the names of directories.

Table 4.9 and 4.10 describe information in Section D and F respectively [59]. We jump to the starting offset of section D after calculating it from the offset number **0x 00 6C** from all the versions of the Windows prefetch files. Reaching the offset of section D will show the relevant information. Figures 4.13 and 4.14 show the snippet of Section D and Section F information from decompressed Windows 10 prefetch files.

Table 4.9. Volume information from Section D in prefetch file.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of** **Information**
0x 00 00	4	Offset to volume device path.
0x 00 04	4	Length of volume device path. **Note: Number of characters for volume device.**
0x 00 08	4	Volume creation time.
0x 00 10	4	Serial number of volume
0x 00 14	4	Offset to section E. **Note: Must be calculated from the ”offset to volume device path”.**
0x 00 18	4	Length of section E.
0x 00 1C	4	Offset to section F. **Note: Must be calculated from the ”offset to volume device path”.**
0x 00 20	4	Length of section F.

Table 4.10. Directory information from Section F in prefetch file.

Offset (in hexadecimal)	**Length** (in bytes)	**Type of Information**
0x 00 00	2	Number of characters of the directory name.
0x 00 02		Directory name presented as Unicode (UTF-16) little-endian string.

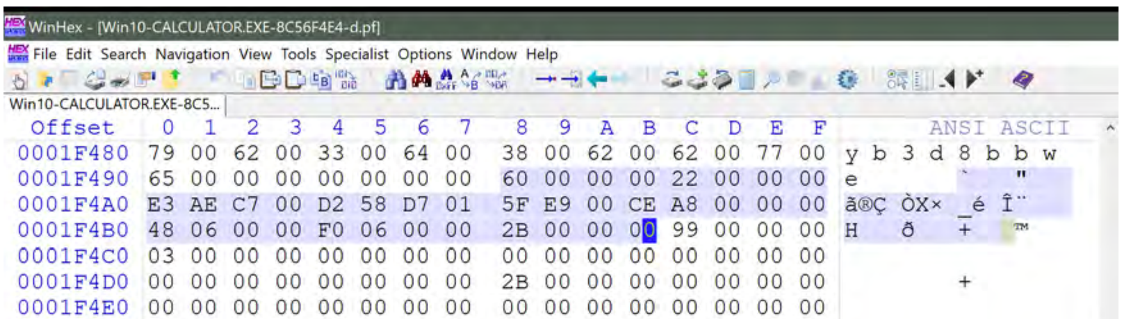


Fig. 4.13. The Section D information after decompressing Windows 10 prefetch file.

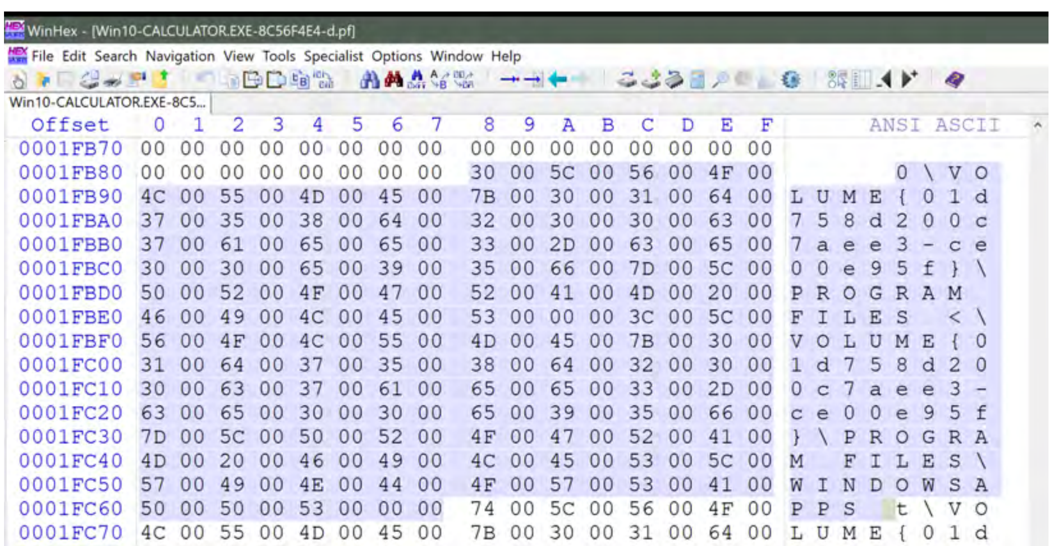


Fig. 4.14. The Section F information after decompressing Windows 10 prefetch file.

Tools for Comparative Prefetch Forensics Analysis

In this section we have used both open-source and commercial tools available for analyzing and parsing prefetch files to draw a comparison. These tools can only display up to a maximum of eight last run times after parsing the prefetch files.

1. **OSForensics** - This convenient proprietary digital forensics tool by PassMark Software [60] has a dedicated Prefetch Viewer. The tool window shows the application name, run count, prefetch file size, prefetch file, prefetch hash, and last run time with seven other run times, if they exist, of applications. It also shows mapped files and directories in the two bottom tabs. We have used the full version of the software as a student license having the same features as that of a full regular licensed software. We have demonstrated the use of OSForensics in figure 4.15. OSForensics is available on a 30-day free trial. A full-version student license can be obtained an affordable price.

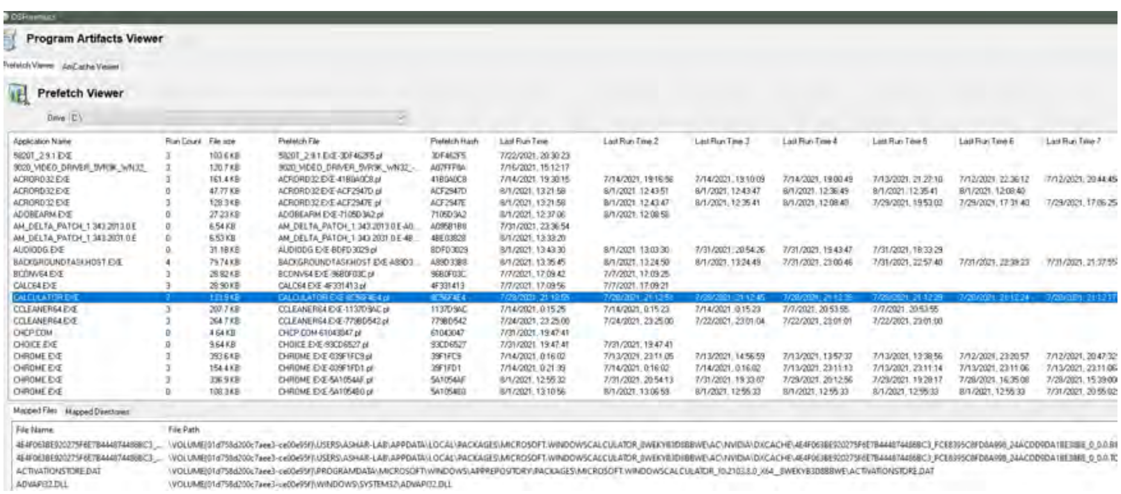


Fig. 4.15. Prefetch Artifacts Viewer from OSForensics tool.

2. **Windows Prefetch Parser** - This proficient open-source python script by

Adam Witt parses Windows Prefetch (pf) files [61]. The script code supports prefetch files from Windows XP up to Windows 10. The best feature of the script is that it can support a directory of prefetch files for parsing. The tool also displays the volume name, creation, and serial number from the pf file. Also, it can output the result in CSV format for better readability. We have demonstrated the use of this tool in the command prompt in figure 4.16.

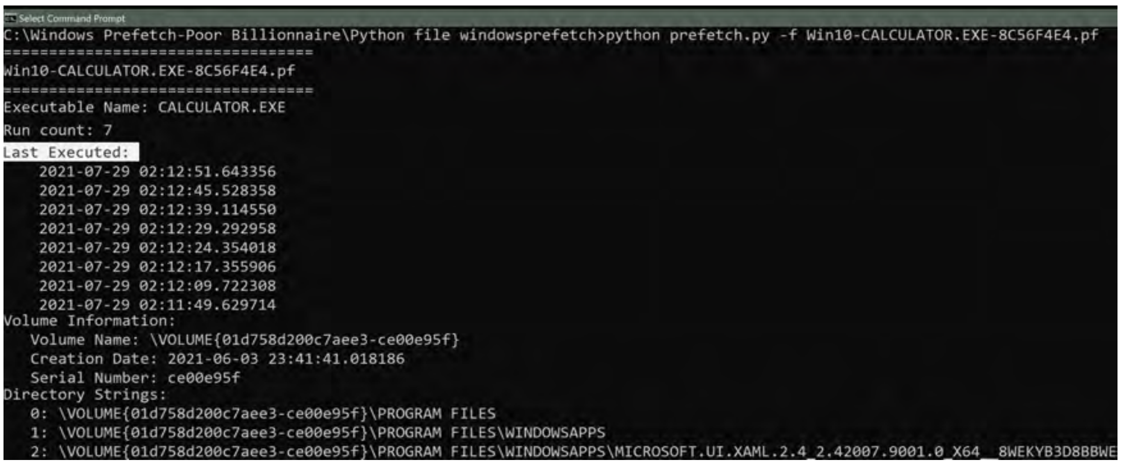


Fig. 4.16. Windows Prefetch Parser open-source python script.

3. **WinPrefetchView** - We have used this freeware tool by NirSoft [62]. The tool is very similar to OSForensics. Along with displaying all the information as OSForensics, WinPrefetch displays the latest modified and created time, with accuracy, of prefetch files whenever the NTFS file system's MFT record is updated. We have demonstrated the use of this tool in figure 4.17.

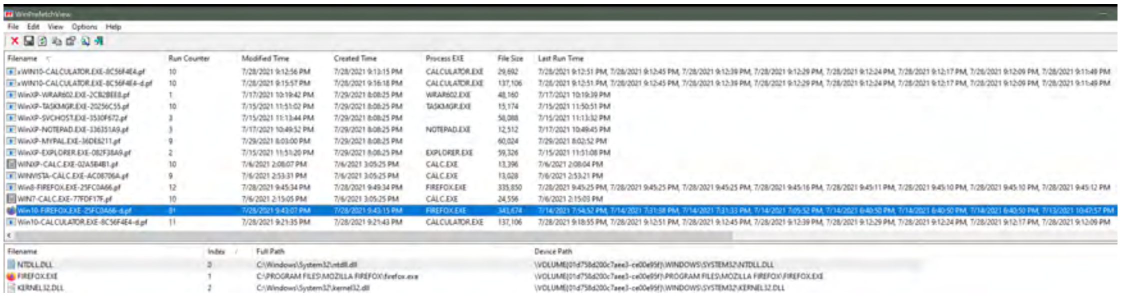


Fig. 4.17. WinPrefetchView freeware tool.

4. **PECmd** - This free stand-alone executable tool by Eric Zimmerman parses prefetch files from Windows XP to Windows 10 [63]. Like the Windows Prefetch Parser tool, this tool can also output the results in many different formats such as JSON, HTML, CSV, CSVF, and JSONF. PECmd displays the latest modified, accessed, and created time, with accuracy, of prefetch files whenever the NTFS file system's MFT record is updated. Moreover, the prefetch file header information displays the Windows version from a pf file. Additionally, it can process volume shadow copies to parse prefetch files if present. We have demonstrated the use of this tool in figure 4.18.

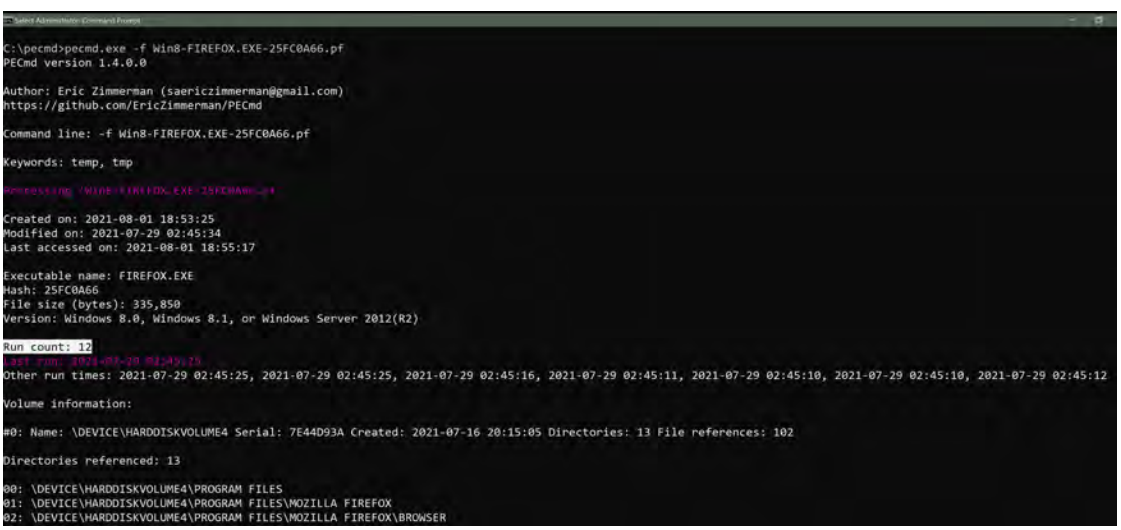


Fig. 4.18. PECmd tool by Eric Zimmerman.

5. **Windows 10 Prefetch Decompress tool** - This open-source python utility

by Francesco Picasso [58] decompresses the compressed Windows 10 prefetch files. The decompressed files thus obtained can be used by any prefetch parsing tool or can be manually examined to understand the working of the Windows 10 pf files. We used this tool to decompress the Windows 10 pf file and study it in the WinHex editor tool. The demonstration of Windows 10 Prefetch Decompress tool is shown in figure 4.19.

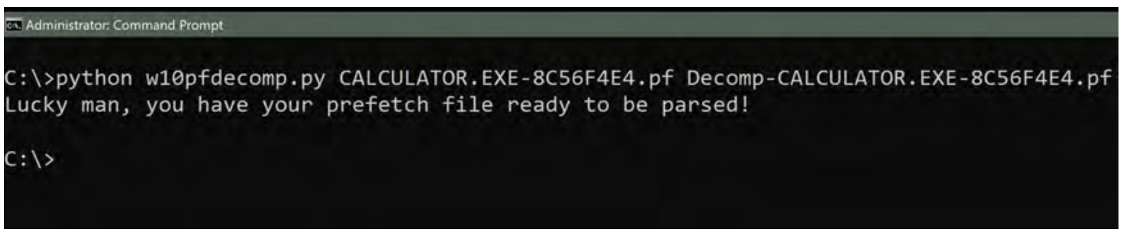


Fig. 4.19. Decompressing Windows 10 prefetch file using Francesco Picasso's python script.

CHAPTER V

Windows Shellbag Forensics

Windows Shellbag

A huge number of forensically useful artifacts are produced by Windows operating systems. The artifacts used in digital forensic examinations contain information that could be used as incriminating evidence. One such artifact is the Windows Shellbag. Shellbag, often known as Windows Shellbag, is a Microsoft Windows operating system component. It was first introduced in 2001 with the release of Windows XP, and it has since become a fundamental element of Windows.

Shellbag is a Windows Registry Key and is also the oddest named artifact in the Microsoft Windows operating system. Shellbag allows you to alter the way you see folders in Windows Explorer [32], [64]. For example, changing the view options of any folder to extra large or small icons, adjusting the sort order of a folder's contents, adding more columns, expanding the folder's window, and so forth. After a user alters a style in Windows Shellbag, all these customizations remain intact. Furthermore, these customization settings stay the same even after a user shuts down his computer.

Every time the user adjusts folder settings, the corresponding update triggers a change in the Shellbag entry. These Shellbag entries are vital and forensically significant. When performing a digital forensics examination on Shellbag entries, queries about when and which folder a culprit viewed can be promptly answered; for example, suppose a corporation accuses an employee of leaking private and sensitive trade secrets housed on a computer system. In that case, the individual's computer may include the essential Shellbag entries confirming that he did indeed enter the folder to leak the secret to a competitor.

Shellbag entries contain useful timestamp information. Therefore, fully comprehending the activities that create and update Shellbag entries becomes challenging. Many factors are considered, such as the version of the Windows operating system used, folder settings, folder types, etc. In this paper, we thoroughly analyze Windows Shellbag in the latest version of Windows 11. In addition, we demonstrate our findings using both open-source and proprietary tools for a holistic understanding of the Shellbag component.

Forensics Importance of Shellbag

All the files on a local system, network system, and attached external devices like USB devices are tracked using Shellbag [32]. Shellbag data is user-specific and user-driven. The changes made by a user will stay intact for that user. Therefore, any Shellbag evidence is an indication of user activity that happened on a system. This valuable evidence is helpful to a digital forensics examiner. For example, traversing a directory, modifying window size, timestamps information, etc. These are crucial artifacts for a forensics investigation. The records in the Shellbag entries are updated as well, which then can translate into suggestive information of the date and time a particular user visited a specific folder. Each user on a system will have their Shellbag information related to them.

The most remarkable aspect of Shellbag is that it remains in the registry record even if the folder is deleted from a local system or USB media or device has been unmounted from the network [65]. The reason for this persistency is that Shellbag information is local to the machine, and the Windows operating system is constantly gathering and recording useful data on the local system. Due to this information, while conducting a forensic examination, a forensic investigator can uncover answers to many important questions such as:

1. Did a user traverse within the local machine?
2. Did a user plug a removable USB media to access files for malicious purposes?
3. Did a user plug the USB media to exfiltrate the company's critical information?

Shellbag Registry and File Location

Shellbag entries are registry keys stored in a specific location in the Windows Registry. These keys can be viewed using the Windows Registry Editor (RegEdit) tool. The information presented in RegEdit, as shown in figure 5.1, is exhibited as hives. A hive is a group of keys, subkeys, and values in the Windows Registry. The Windows Registry has a fixed number of supportive files loaded in computer memory when the Windows operating system is booted or when a user logs in [66]. Hence, one can quickly determine when a particular folder was first visited or last updated based on the timestamps mentioned. Table 5.1 lists these registry hives and their supporting files.

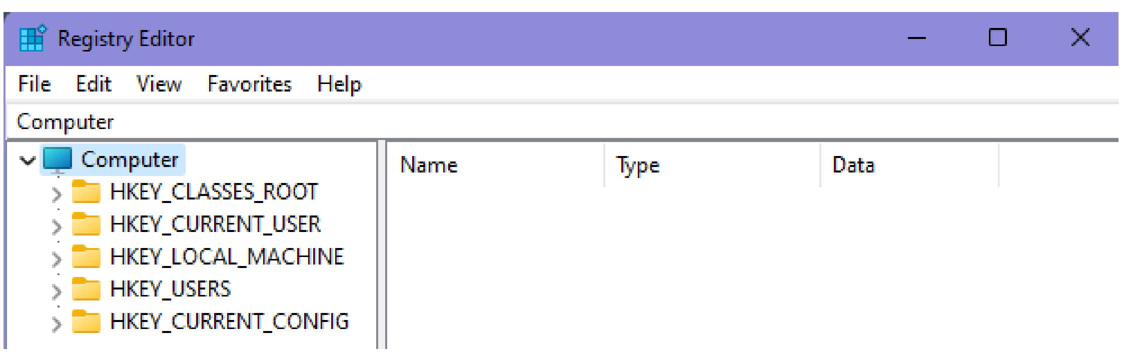


Fig. 5.1. Windows Registry Editor

Table 5.1. The list of Windows Registry Hives and their supporting files.

Registry Hive	Supporting files
HKEY_CURRENT_CONFIG	System, System.alt, System.log, System.sav
HKEY_CURRENT_USER	NTUSER.DAT, ntuser.dat.log
HKEY_CURRENT_USER\Sofware\Classes	UsrClass.DAT
HKEY_LOCAL_MACHINE\SAM	Sam, Sam.log, Sam.sav
HKEY_LOCAL_MACHINE\Security	Security, Security.log, Security.sav
HKEY_LOCAL_MACHINE\Software	Software, Software.log, Software.sav
HKEY_LOCAL_MACHINE\System	System, System.alt, System.log, System.sav
HKEY_USERS\.DEFAULT	Default, Default.log, Default.sav

Furthermore, there are user-specific files connected with the registry keys in the case of Shellbag. **NTUSER.DAT** and **UsrClass.DAT** are those associated files. Both **NTUSER.DAT** and **UsrClass.DAT** are user-specific files, while the latter stores a user's registry information separate from the main registry hives [67]. NTUSER.DAT and UsrClass.DAT are presented in a unified view as **HKEY_CURRENT_USER** (HKCU) in a live system. The UsrClass.DAT file is plugged in **HKCU****Software****Classes**, while NTUSER.DAT is mapped to HKCU [68]. Table 5.2 talks about Shellbag entries corresponding to SID. The SID can be found by issuing a **wmic** command in the **Windows Command Prompt** (**cmd**) as shown in figure 5.2.

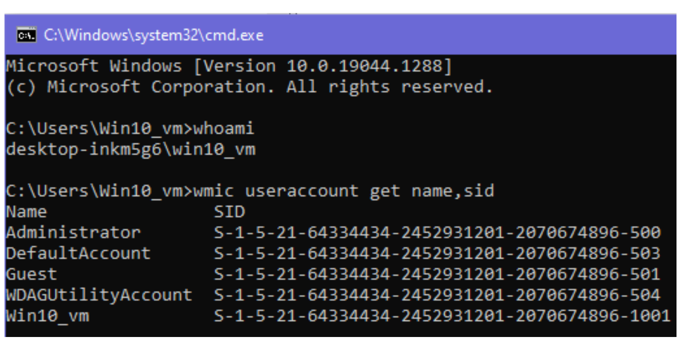


Fig. 5.2. Issuing **wmic** command to obtain SID of a user account.

Table 5.2. Location of Shellbag entries inside Windows Registry, and NTUSER.DAT and UsrClass.DAT files.

Registry Location of NTUSER.DAT Shellbag entries
*HKCU\Software\Microsoft\Windows\Shell**BagMRU*** *HKCU\Software\Microsoft\Windows\Shell**Bags***
File Location of NTUSER.DAT
C:\Users\username\NTUSER.DAT
Registry Location of UsrClass.DAT Shellbag entries
*HKCU\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell**BagMRU*** *HKCU\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell**Bags***
File Location of UsrClass.DAT
C:\Users\username\AppData\Local\Microsoft\Windows\UsrClass.DAT
Other Important Shellbag entries location
HKCU\Software\Microsoft\Windows\CurrentVersion\Explorer\RecentDocs\Folder *HKCU\Software\Microsoft\Windows\Shell**BagMRU*** *HKCU\Software\Microsoft\Windows\Shell**Bags**\1\Desktop* *HKEY_USERS\sid\Software\Microsoft\Windows\Shell**BagMRU*** *HKEY_USERS\sid\Software\Microsoft\Windows\Shell**Bags***

Interpreting BagMRU and Bags Subkeys in the Registry

NTUSER.DAT and **UsrClass.DAT** files contain two important subkeys, BagMRU and Bags as shown in table 5.2 listing.

1. **BagMRU** and **Bags**: **BagMRU** subkey shows the directory structures of the folders that were interacted with within the numbered subkey/value hierarchy format. Figure 5.3 demonstrates the **BagMRU** subkey as it shows the registry path translation from the Shellbag entry to the corresponding original path in the computer. On the other hand, the **Bags** subkey constitutes numbered subkeys for each hierarchical corresponding child subkey under **BagMRU** [69].

    (a) For example: Windows 11 registry path:

HKCU\SOFTWARE\Classes\Local Settings\Software\Microsoft\Windows\Shell\ BagMRU\1\1\1\0 will have the actual translation to **Desktop\ThisPC\C:\shell_test\txt_doc**

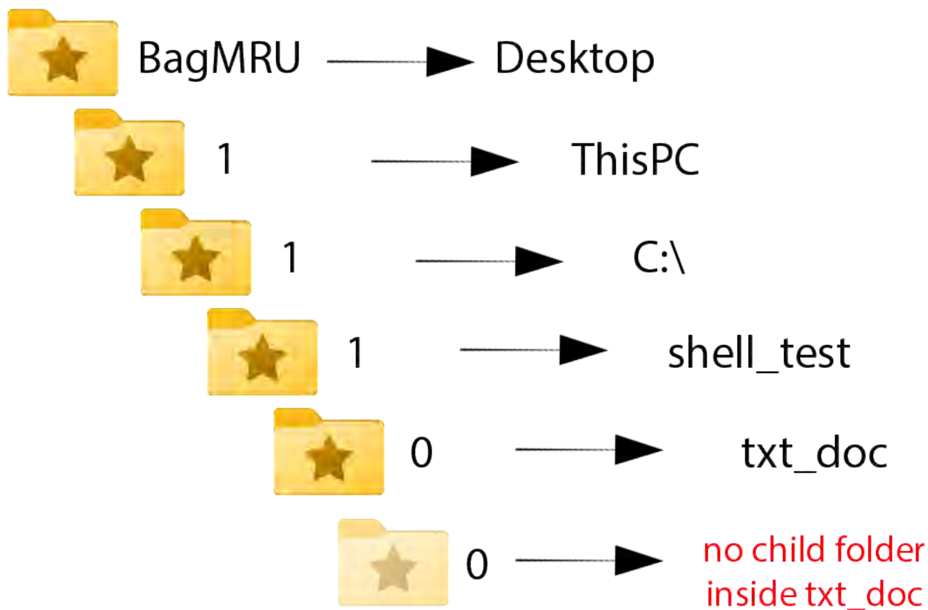


Fig. 5.3. The registry path translation to original path from Shellbag entries.

(b) Each particular numbered subkey contains an entry called **NodeSlot**. The decimal number obtained from this entry points to the folder's customization settings in the **Bags** subkey, such as group view, icon size, sort order, etc. So, to view the personalization choice values, we have to see the **NodeSlot** value in decimal and move to a particular numbered entry in the **Bags** subkey. Figures 5.4 and 5.5 show an example of the **NodeSlot** entry for the **txt_doc** folder's view settings. The **txt_doc** view settings are in the **59th entry** in the **Bags** subkey, as shown in figure 5.4. Figure 5.5 shows an elaborated view of bags entry number 59 of the **txt_doc** folder's view settings. It is also possible to obtain the folder's name using the **BagMRU** numbered entry [69], as demonstrated from the highlighted parts in figure 5.6. NOTE: **MRUListEx**, shown in figure 5.7, is a 4-byte value indicating the order in which folders were accessed. It shows the most recent access first. The color-coding exhibited in figure 5.7 and the numbers preceded by a # sign show the order in which the folders customization was done.

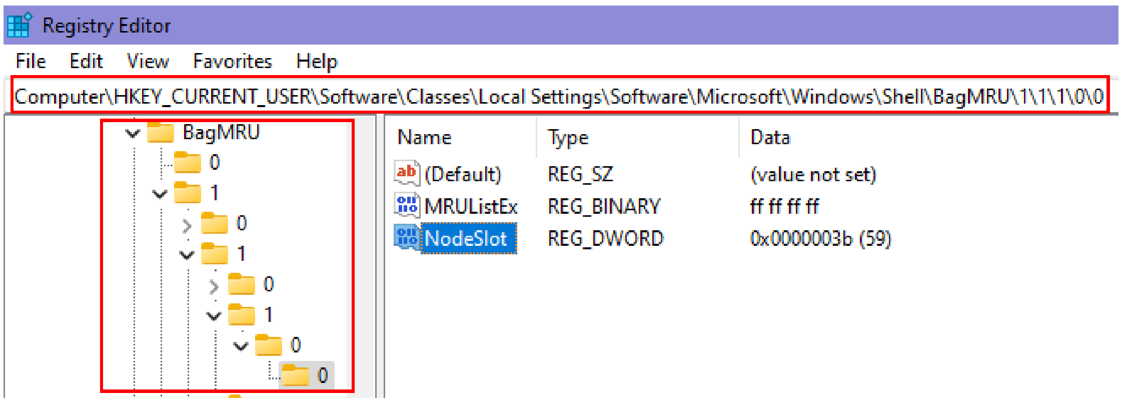


Fig. 5.4. The NodeSlot entry for txt_doc folder's view settings.

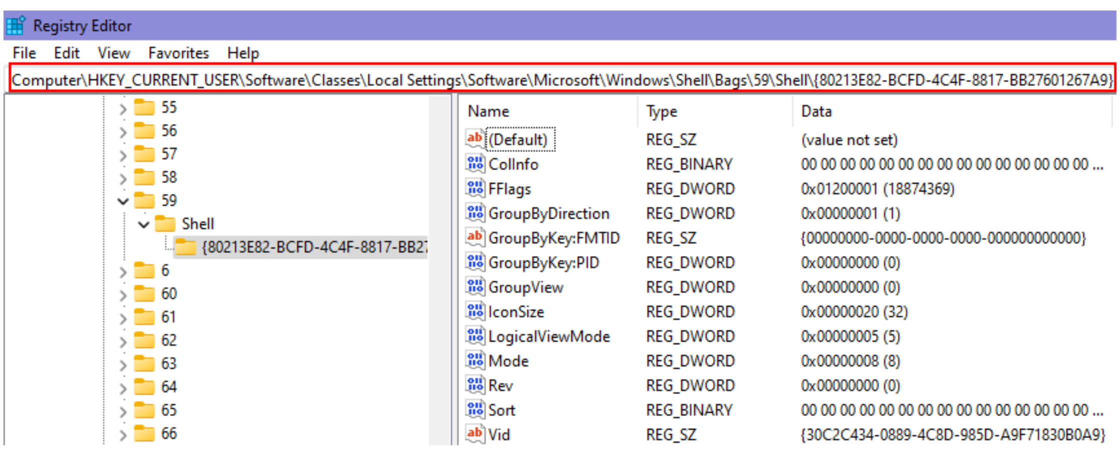


Fig. 5.5. The Bags entry number 59 for txt_doc folder's view settings.

(c) Windows Registry stores the **Last Write Time** of **keys** and **subkeys**; however, it does not do so for the Last Write Time of individual values inside [69].

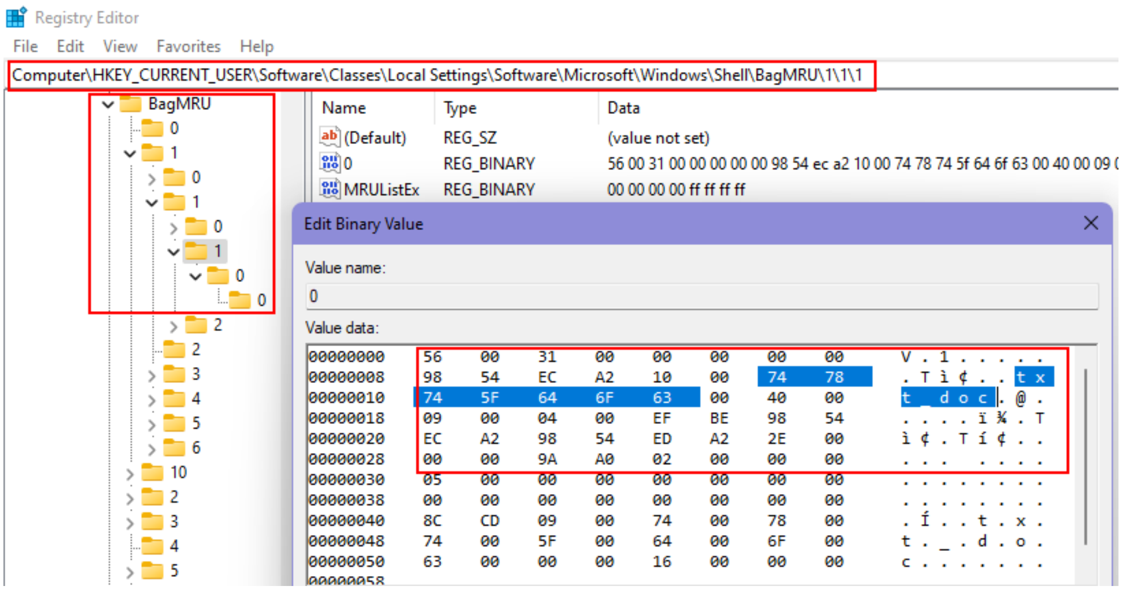


Fig. 5.6. Obtaining name of folder using BagMRU entry.

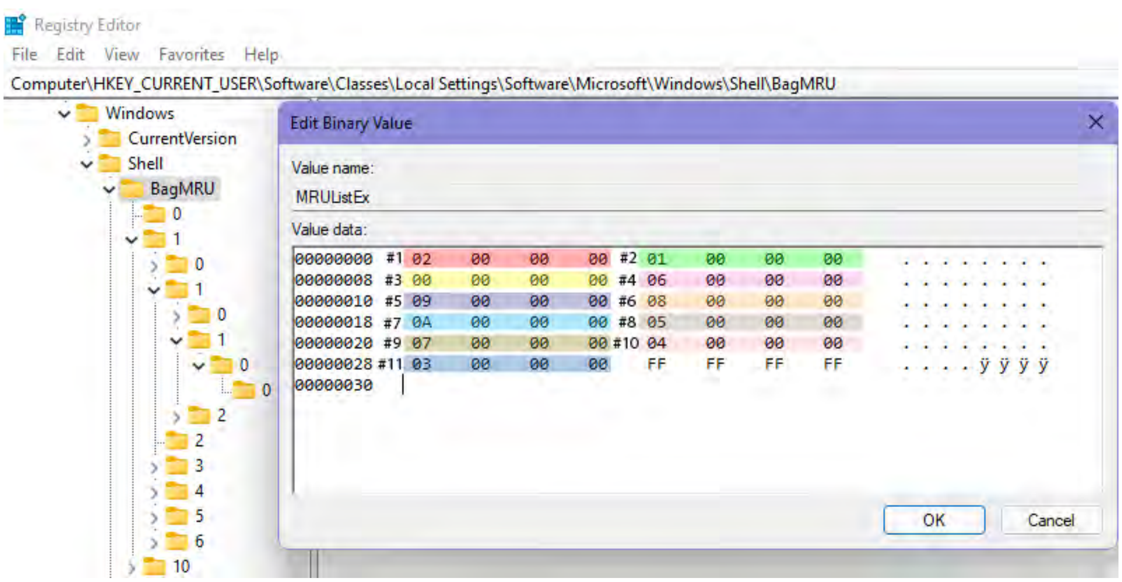


Fig. 5.7. The MRUListEx inside BagMRU key.

Experiment Initiation

We conducted our experiment on a Windows 10 v21H2 system running on Samsung 970 Evo Plus NVMe SSD. Our test experiment focused on Shellbag entries within a local machine and a USB media used on the local machine. Table 5.3 summarizes the different test scenarios used throughout the investigation.

Table 5.3. Summary of the test experiments.

Test Scenario	**Summary of Test Experiments**
Local Machine	Shellbag entries for a folder on Desktop
Local Machine	Shellbag entries for a folder inside C:\ drive
Within a USB	Shellbag entries for a folder inside a USB drive
Local Machine	Shellbag entries for compressed files

Shellbag Entries for Desktop Folder and in C:\ Drive

In this scenario, we created a regular folder on a user's Desktop called **sh_test1.** Then inside this folder, we created three nested folders, **sh_test2**, **sh_test3**, and **sh_test4**, respectively. Similarly, under **C:**\ drive, we created **shell_test** folder with **txt_doc** directory under it. The Shellbag entries for these two subcases are shown in figures 5.8, 5.9. The corresponding **BagMRU** entry for the **sh_test1** folder was **ten (10)** and **BagMRU** entry for the **C:**\ was **one (1)**, respectively.

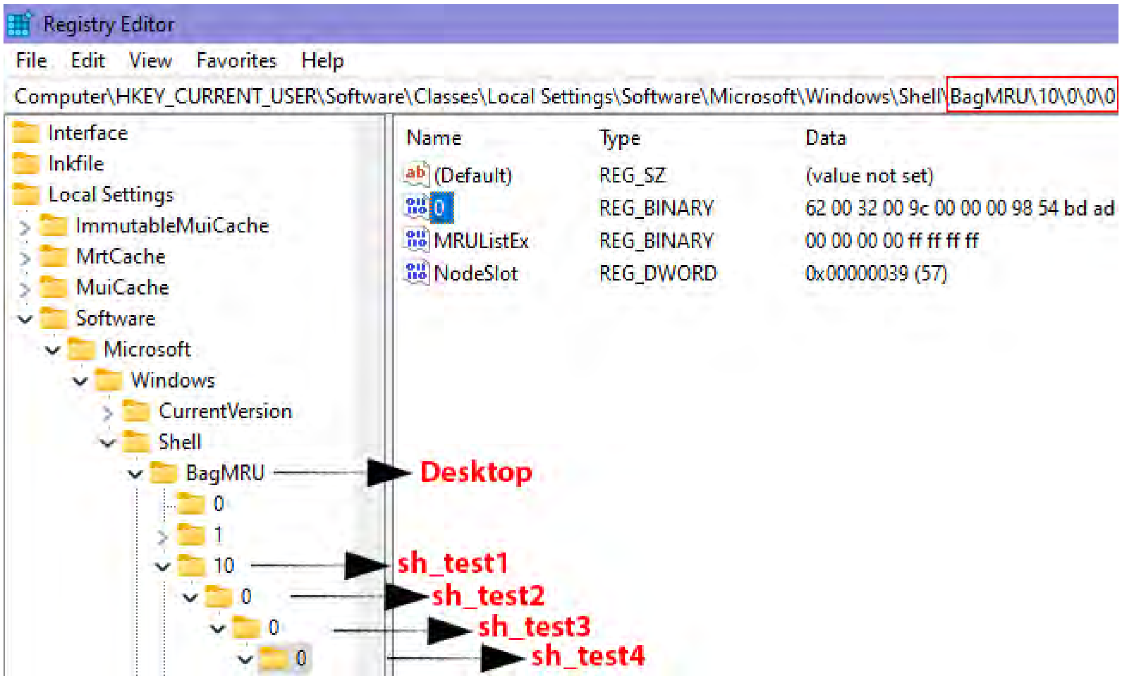


Fig. 5.8. BagMRU Shellbag entry for a folder customization inside Desktop.

Shellbag Entries for USB Drives

In our experiment described in this subsection, we created four folders recursively inside our USB flash drive. These folders were called, **folder1**, **folder2**, **folder3**, and **folder4**, respectively. The USB drive letter associated with our device was **E:**\ as can been from figure 5.10.

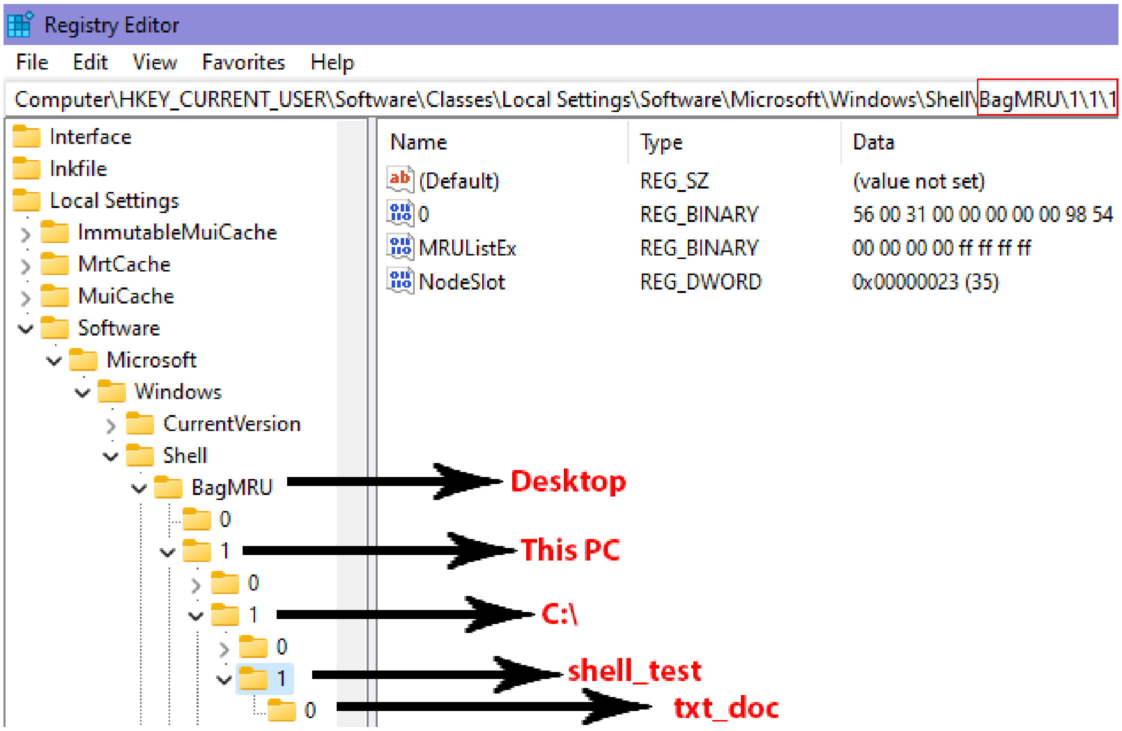


Fig. 5.9. BagMRU Shellbag entry for a folder customization inside C:\Drive.

Upon connecting the USB device to the USB port of our workstation, Windows automatically assigned the letter **E:** to our storage device. The operating system did so because drive letters, **C:**\ and **D:**\ were already assigned to the two partition entries. One for operating system partition and the other for the optical drive. The corresponding **BagMRU** entry for the **drive letter**, **E:**\, of our USB device was **five** (**5**).

Shellbag Entries for Compressed Files

Creating a zip file inside folder: Windows Registry keeps track of Shellbag entries for zip (.zip) file. We created zip files for three places. One was inside **sh_test_zip** which was created under **sh_test4** on the user's desktop. Windows created a separate entry specially for the zip file created. The name of the zip file was **zip_txt.zip**. The corresponding **BagMRU** entry for this zip file was **zero** (**0**) as it was only zip file inside the **sh_test_zip** folder.

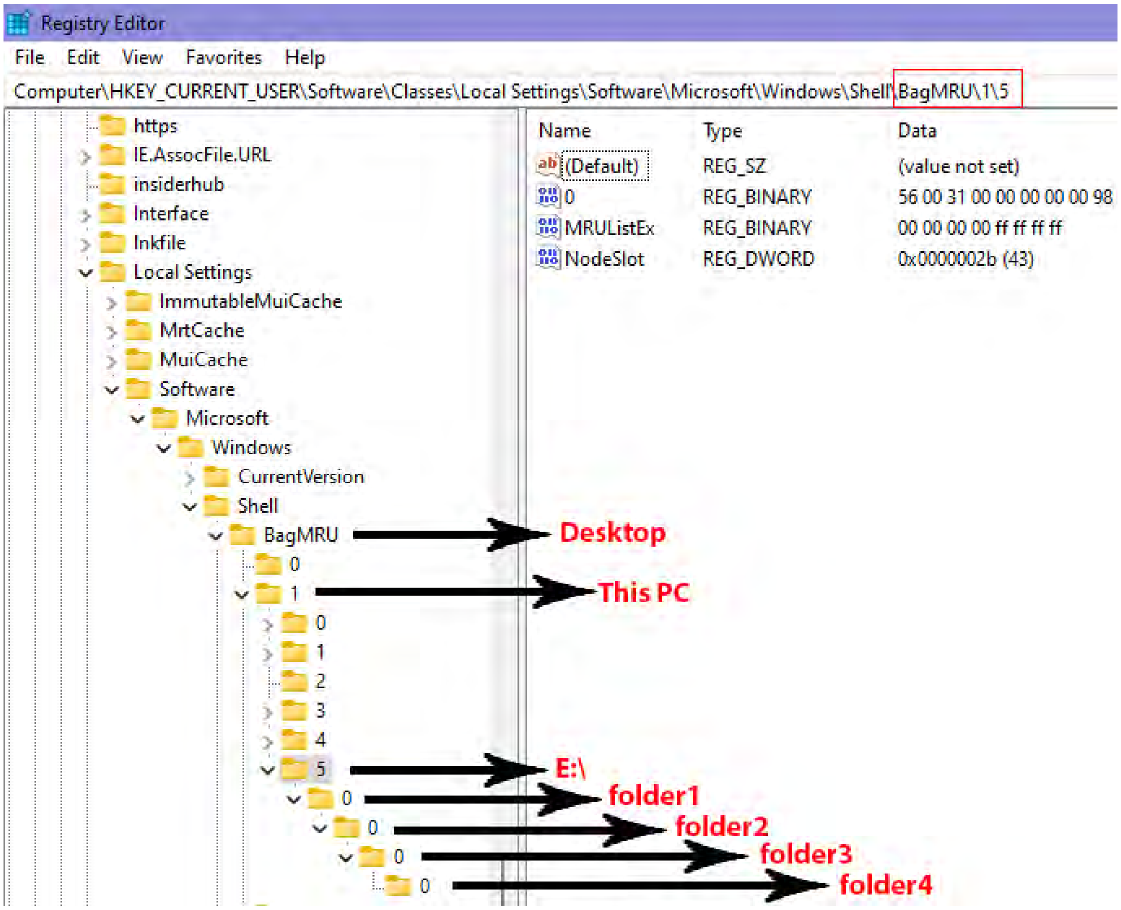


Fig. 5.10. BagMRU Shellbag entry for a folder customization on a USB Drive with E:\drive letter.

The second zip file, **txt.zip**, was created under **txt_doc** directory (which was inside the folder of **C:** drive), **shell_test**). The **BagMRU** entries for **txt_doc** was **one** (**1**), and **txt.zip** was **zero** (**0**), respectively.

Lastly, the third zip file, **e_drive_txt.zip**, was created under **folder4** inside the USB drive. The corresponding **BagMRU** entry for this zip file was **zero** (**0**) as it was only zip file inside the **folder4** directory. Figures 5.11, 5.12, 5.13 showcased our findings.

We extended this experiment to see if Windows 10 supports WinRar (.rar) and 7zip (.7z) Shellbag entries. Unfortunately, we did not find any record for the two file types even after changing the view options for the .rar and .7z.

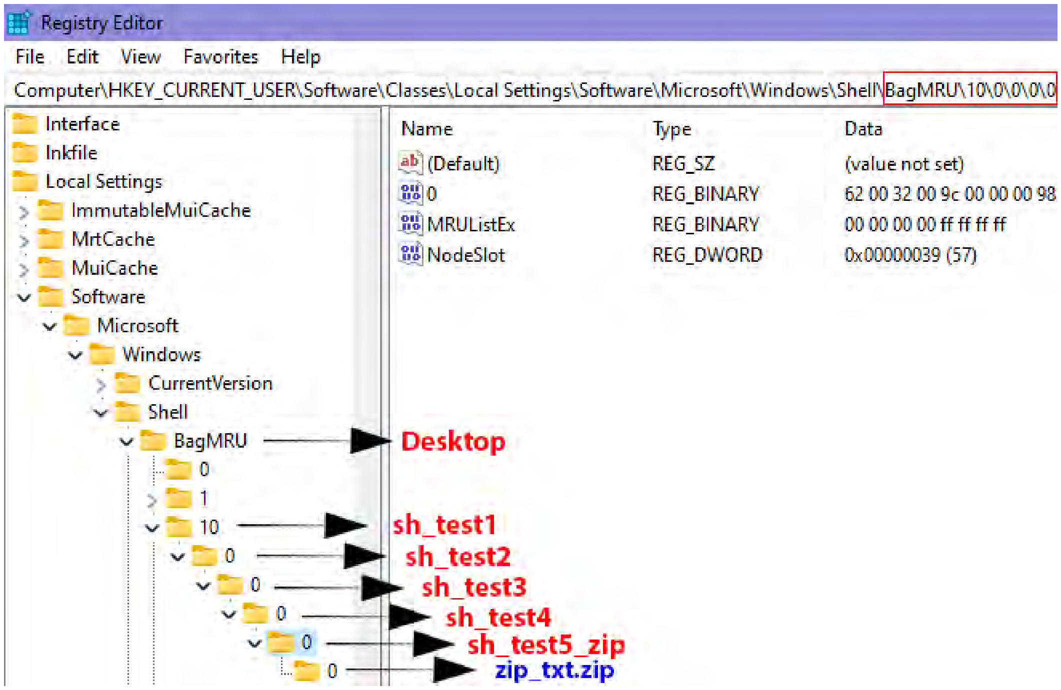


Fig. 5.11. BagMRU Shellbag entry for a ZIP file inside a folder on Desktop.

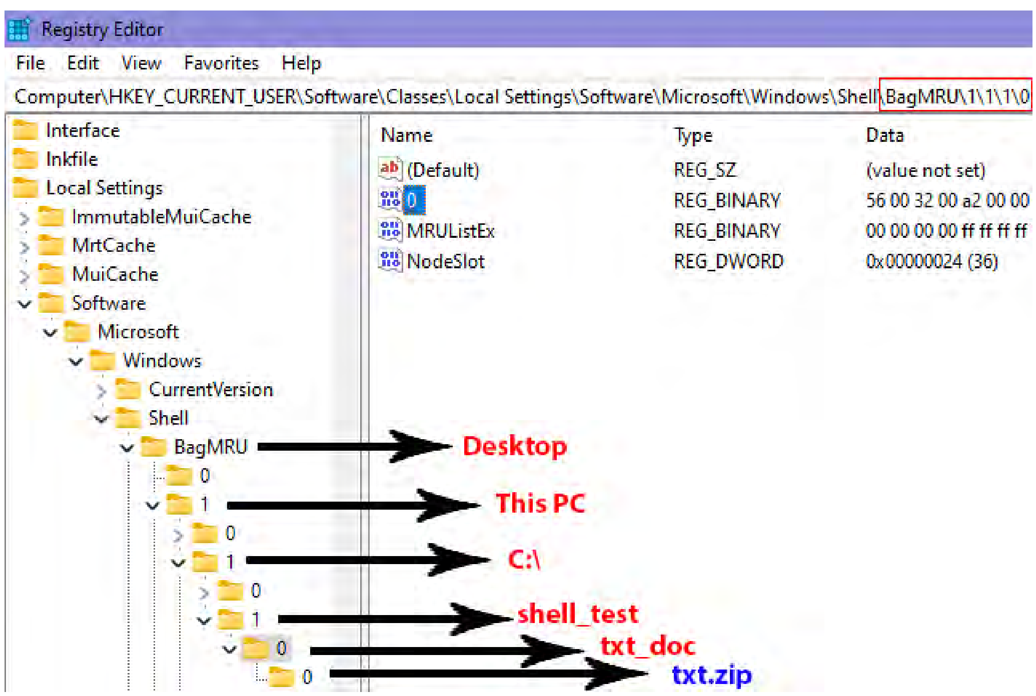


Fig. 5.12. BagMRU Shellbag entry for a ZIP file inside a folder under C:\.

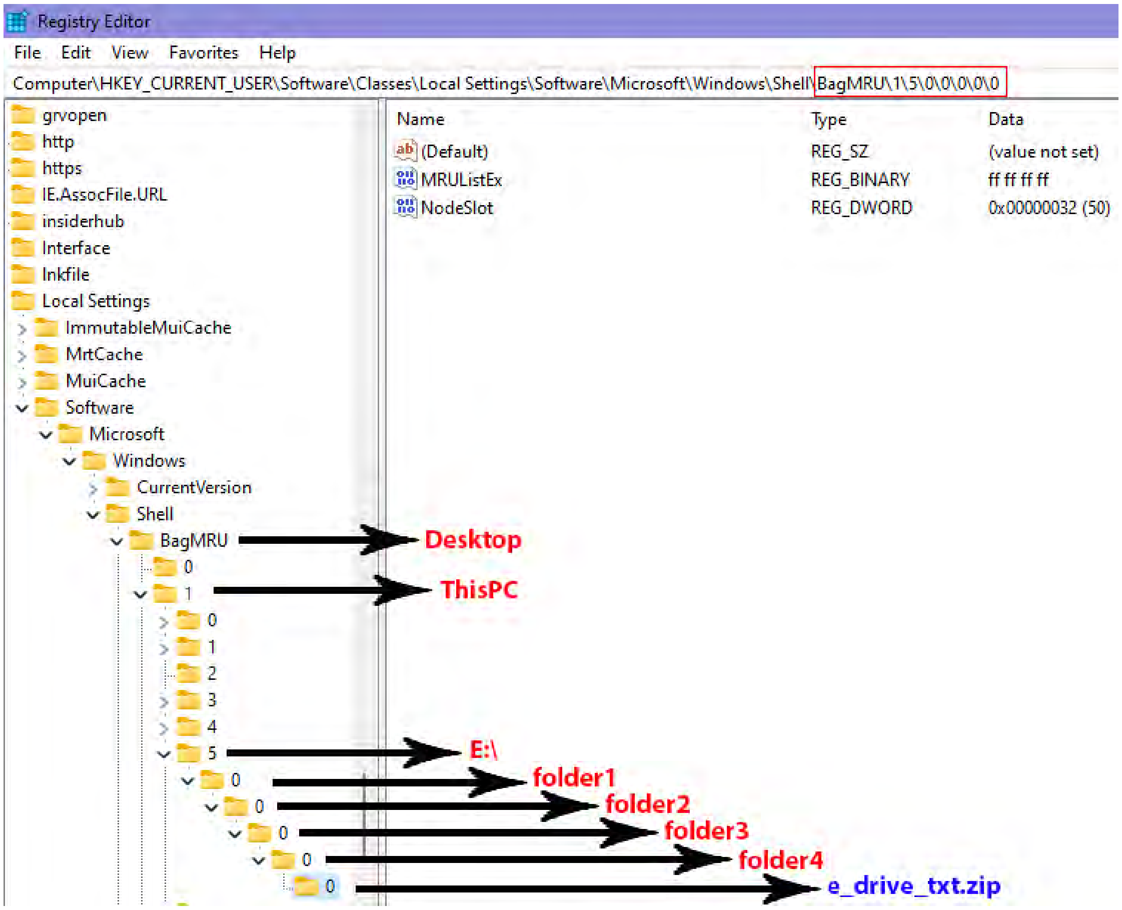


Fig. 5.13. BagMRU Shellbag entry for a ZIP file inside a folder in USB drive with E:\drive letter.

Forensics Analysis using OSForensics

The **OSForensics** [60] tool by PassMark software is a proprietary tool that we used to forensically analyze our Shellbag entries for all folder customizations. This tool does a tremendous job is fetching all the Shellbag entries information from the Windows 10 Registry. The display items for Shellbag entries include items' names and their full path starting from the Desktop. Then followed by the timestamps information, i.e., created, modified, and accessed dates. Figure 5.14 displays the information of **OSForensics**.

OSForensics[1] tool can scan both the live operating system and offline registry hives to parse the Shellbag information. Furthermore, it also can export registry files for later analysis. A detailed comparison of OSForensics with other tools used in the study is shown in table 5.4.

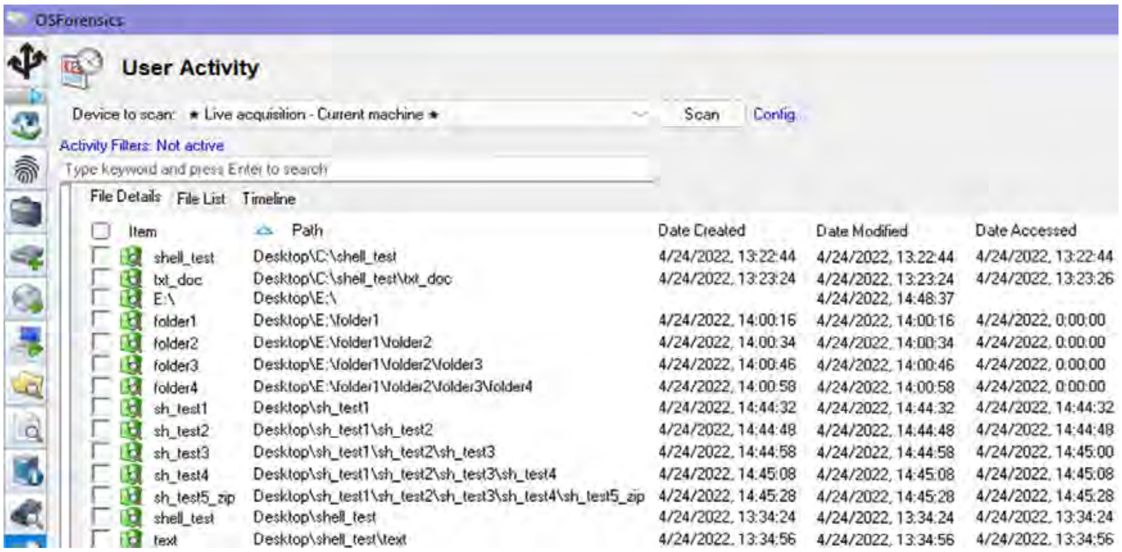


Fig. 5.14. OSForensics analysis windows.

Forensics Analysis using ShellBags Explorer Tool

ShellBags Explorer [70], an open-source, easy-to-use tool authored by Eric Zimmerman, is the most comprehensive tool for forensically analyzing Windows Shellbag information. The tool parsed relevant forensics information from the Registry. It even showed the parent-child relationship of a folder with another folder or a zip file. Like the OSForensics tool, **ShellBags Explorer** can analyze offline and online registries. A detailed comparison of **ShellBags Explorer**[2] with other tools is shown in table 5.4. Figures 5.15 and 5.16 display the interface for ShellBags Explorer tool, which demonstrates the different forensics analysis information such as the **absolute path**, **last write time** and **modified-access-created timestamps**.

1 OSForensics is available at: https://www.osforensics.com/osforensics.html

2 ShellBags Explorer is available at: https://ericzimmerman.github.io/#!index.md

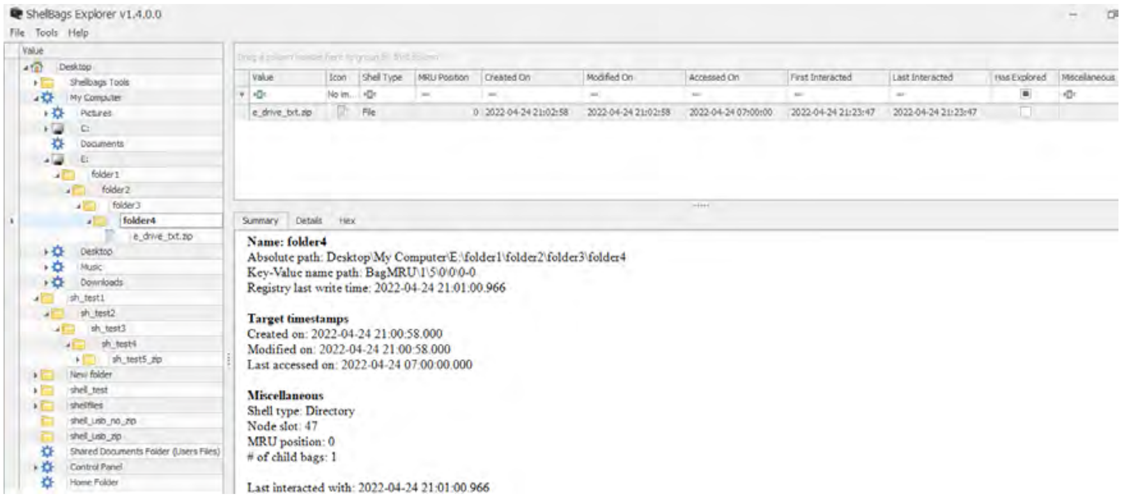


Fig. 5.15. Shellbags Explorer analysis window for folders.

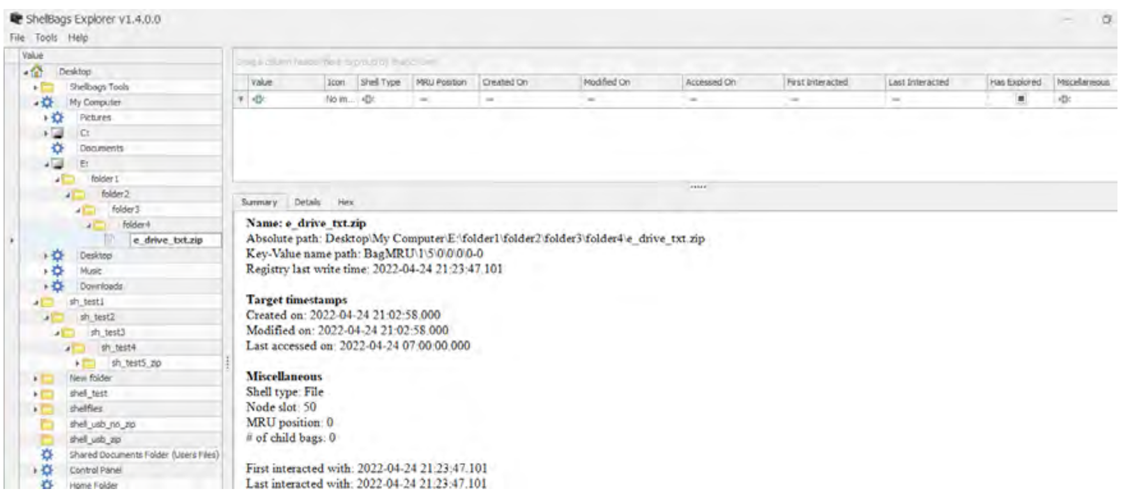
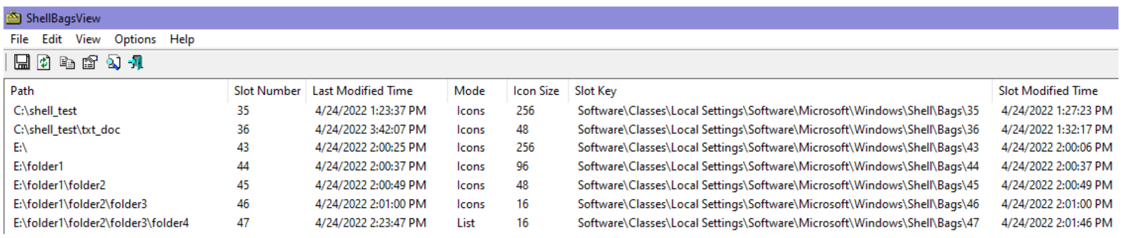


Fig. 5.16. ShellBags Explorer analysis window for zip file.

Forensics Analysis using ShellBagsView Tool

The freeware tool called, **ShellBagsView** is a lightweight software by Nirsoft. This tool thoroughly analyzes Shellbag entries from the Windows Registry. Unfortunately, unlike ShellBagsExplorer, **ShellBagsView** cannot handle offline registry analysis for Shellbags as this feature is not supported. The tool by default displays seven default columns: **path**, **slot number**, **last**

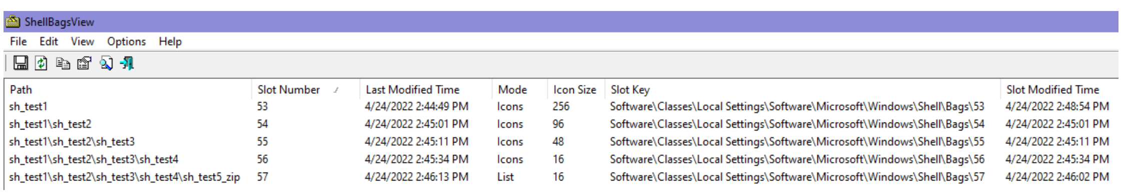
modified time, **mode**, **icon size**, **slot key**, **slot modified time**, **windows position**, **windows size**, **type**, and **username**. However, not all column entries will be filled after **ShellBagsView** parses the evidence from the Registry. A detailed comparison of **ShellBagsView**[3] with other tools is shown in Table 5.4. Figure 5.17 presents the forensics information extracted using **ShellBagsView** based on the created folders in the experiment.. In contrast, figure 5.18 shows the extracted forensic information retrieved from the creation of zipped files.



ShellBagsView
File Edit View Options Help

Path	Slot Number	Last Modified Time	Mode	Icon Size	Slot Key	Slot Modified Time
C:\shell_test	35	4/24/2022 1:23:37 PM	Icons	256	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\35	4/24/2022 1:27:23 PM
C:\shell_test\txt_doc	36	4/24/2022 3:42:07 PM	Icons	48	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\36	4/24/2022 1:32:17 PM
E:\	43	4/24/2022 2:00:25 PM	Icons	256	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\43	4/24/2022 2:00:06 PM
E:\folder1	44	4/24/2022 2:00:37 PM	Icons	96	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\44	4/24/2022 2:00:37 PM
E:\folder1\folder2	45	4/24/2022 2:00:49 PM	Icons	48	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\45	4/24/2022 2:00:49 PM
E:\folder1\folder2\folder3	46	4/24/2022 2:01:00 PM	Icons	16	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\46	4/24/2022 2:01:00 PM
E:\folder1\folder2\folder3\folder4	47	4/24/2022 2:23:47 PM	List	16	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\47	4/24/2022 2:01:46 PM

Fig. 5.17. ShellBagsView analysis window for folders.



ShellBagsView
File Edit View Options Help

Path	Slot Number	Last Modified Time	Mode	Icon Size	Slot Key	Slot Modified Time
sh_test1	53	4/24/2022 2:44:49 PM	Icons	256	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\53	4/24/2022 2:48:54 PM
sh_test1\sh_test2	54	4/24/2022 2:45:01 PM	Icons	96	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\54	4/24/2022 2:45:01 PM
sh_test1\sh_test2\sh_test3	55	4/24/2022 2:45:11 PM	Icons	48	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\55	4/24/2022 2:45:11 PM
sh_test1\sh_test2\sh_test3\sh_test4	56	4/24/2022 2:45:34 PM	Icons	16	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\56	4/24/2022 2:45:34 PM
sh_test1\sh_test2\sh_test3\sh_test4\sh_test5_zip	57	4/24/2022 2:46:13 PM	List	16	Software\Classes\Local Settings\Software\Microsoft\Windows\Shell\Bags\57	4/24/2022 2:46:02 PM

Fig. 5.18. ShellBagsView analysis window for zip files.

Comparative Findings from Tools used

Table 5.4 shows a detailed comparison of different tools used in the experiment, i.e., proprietary, open-source, and freeware. We have used **OSForensics** from *Passmark Software* [60], **ShellBags Explorer** [70] by *Eric Zimmerman*, and **ShellBagsView** [71] from *NirSoft*. Every tool is unique and displays evidence based on the ability to parse data from Shellbag entries from Windows Registry.

3 ShellBagsView is available at: https://www.nirsoft.net/utils/shell_bags_view.html

We have listed the features of all the tools to showcase their competency when conducting forensics analysis on the Windows 10 v21H2 operating system. The tools used in Windows 10 did not show any software conflict and performed their tasks smoothly. The comparison chart aims to help digital forensics practitioners to have a holistic view when conducting a forensics examination on Shellbag. Table 5.4 summarizes the forensic information recovered from Shellbag entries from the three different tools used. From the comparison shown in table 5.4, **ShellBag Explorer** stood out because it provides much more information than other tools.

Table 5.4. Tools comparison chart based on artifacts obtained.

Artifacts Obtained	**OSForensics** Proprietary Tool	**ShellBags Explorer** Open-source Tool	**ShellBagsView** Freeware Tool
Item/Value Name	✔	✔	✖
Absolute/ Full File Path	✔	✔	✔
Shell Type	✖	✔	✖
BagMRU/Registry Position Exhibition	✔	✔	✖
Node Slot/Bag subkey Position	✔*	✔	✔
View Mode	✖	✖	✔
Icon Size Value	✖	✖	✔
Key Value Name Path	✔*	✔	✔
Modified Timestamp	✔	✔	✔
Accessed Timestamp	✔	✔	✖
Created Timestamp	✔	✔	✖
First Interacted Timestamp	✖	✔	✖
Last Interacted Timestamp	✖	✔	✖
Registry Last Write Time	✔**	✔	✔
OS Identifier from Shellbag entry	✖	✔	✖
MFT Entry & Sequence Numbers	✖	✔	✖
File System Hints	✖	✔	✖
Display Long and Short File Names	✖	✔	✖

* Shown only after double-clicking the Shellbag entries.
** Presented as key edit time upon double-clicking Shellbag entries.

CHAPTER VI

Windows 10 ETL File Forensics

Event Trace Logs (ETLs) are traces from Event Tracing for Windows (ETW) that are saved to storage media. ETW was first launched with Windows 2000 and is still included in recent Windows operating systems. ETL files can store a snapshot of events relating to state information at a specific time or events relating to state information over time [53].

Event Tracing is a critical step for maintaining the well-being of a system. As a result, both Windows operating system and application developers use it. Some of the applications or processes that generate events are Microsoft Office, Windows Shutdown, Windows Booting, Windows SleepStudy, Skype, Lync, OneDrive, Power Efficiency Diagnostics, Explorer Start-up [53]. Windows ETW is enabled by default; however, several factors include the version of the operating system, software installed, dictate when the tracing will start, what it will consist of, etc.

The file extension of Event Trace Logs files is .etl. ETL files stored on storage devices vary in their data and volatility. When first configuring a trace session, the ETW settings are used to decide how log files will be stored and what information will be stored. For example, some log files are circular, which overwrites the present file content with new information when the maximum file size is reached. Other settings of ETL contain log file's contents starting from scratch. Settings of ETL files can also include multiple log files for each instance that the event trace information saved to the disk. Windows stores information into ETL files when the system is shut down, booted, a second user logged into the system when performing updates, etc. A wealth of forensics information can be determined when parsing an ETL file.

In this chapter, we have talked about the BootPerfDiagLogger.etl file, which is found at **C:\Windows\System32\WDI\LogFiles**. BootPerfDiagLogger.etl is a Circular Kernel Context Logs (CKCL) file containing information about the system that the event trace session knows when it booted. In addition, we have used multiple ETL file parsing tools, open-source and freeware, to exhibit a comprehensive understanding of the subject.

- ETLParser
- PerfView
- FullEventLogView
- SvclogViewer
- Windows Performance Analyzer
- TraceFMT by Windows Development Toolkit

NOTE: It is worth mentioning that when decoding an ETL on a system other than the source system, the information needed to decode event data properly may be unavailable. When a system registers an event provider, it records the information required to interpret the event data. The tool will be unable to correctly parse the events if the event provider is not listed on the system you are using to decode an ETL file [53].

Circular ETL File Configuration

When the maximum file/buffer size is reached, the circular event trace sessions will overwrite existing events with new ones. Therefore, the previous events in the ETL will be overwritten and become unrecoverable. This is why this kind of file is called **Circular Kernel Context Logs** (**CKCL**). BootPerfDiagLogger.etl and ShutdownPerfDiagLogger.etl are all examples of circular log files in Windows 10 [53]. Moreover, when a maximum file size is reached in ETL files that use the new file option, a new file is created with an incrementing value as the new file's name. WdiContextLog.etl.001, WdiContextLog.etl.002, WdiContextLog.etl.003, are the examples of this kind of file. Figure 6.1 displays the important events associated with BootPerfDiagLogger.etl file [1].

Event Name	Description	Fields
DiskIO/Read	A list of disk reads at the time the trace occurred that the trace session has recorded.	DiskNumber, IrpFlags, Priority, TransferSize, ByteOffset, Irp, ElapsedTimeMSec, DiskServiceTimeMSec, FileKey, FileName
DiskIO/Write	A list of disk writes at the time the trace occurred that the trace session has recorded.	DiskNumber, IrpFlags, Priority, TransferSize, ByteOffset, Irp, ElapsedTimeMSec, DiskServiceTimeMSec, FileKey, FileName
FileIO/FileCreate	A list of files created at the time the trace occurred that the trace session has recorded.	FileKey, FileName
FileIO/FileDelete	A list of files deleted at the time the trace occurred that the trace session has recorded.	FileKey, FileName
FileIO/FileRundown	A list of open file handles at the time the trace occurred that the trace session has recorded.	FileKey, FileName
Image/DCStart	A list of images (.dll, .sys, .exe) and the processes they were loaded into at the time the trace started.	ImageBase, ImageSize, ImageChecksum, TimeDateStamp, DefaultBase, BuildTime, FileName
Image/DCEnd	A list of images (.dll, .sys, .exe) and the processes they were loaded into at the time the trace ended.	ImageBase, ImageSize, ImageChecksum, TimeDateStamp, DefaultBase, BuildTime, FileName
Image/Load	A list of images (.dll, .sys, .exe) and the processes they were loaded into at the time the trace occurred that the trace session has recorded.	ImageBase, ImageSize, ImageChecksum, TimeDateStamp, DefaultBase, BuildTime, FileName
Process/DCStart	A list of running processes at the time the trace started.	ProcessID, ParentID, ImageFileName, PageDirectoryBase, Flags, SessionID, ExitStatus, UniqueProcessKey, CommandLine, PackageFullName, ApplicationID
Process/DCEnd	A list of running processes at the time the trace ended.	ProcessID, ParentID, ImageFileName, PageDirectoryBase, Flags, SessionID, ExitStatus, UniqueProcessKey, CommandLine, PackageFullName, ApplicationID
Process/Start	A list of running processes at the time the trace started.	ProcessID, ParentID, ImageFileName, PageDirectoryBase, Flags, SessionID, ExitStatus, UniqueProcessKey, CommandLine, PackageFullName, ApplicationID
Process/End	A list of running processes at the time the trace ended.	ProcessID, ParentID, ImageFileName, PageDirectoryBase, Flags, SessionID, ExitStatus, UniqueProcessKey, CommandLine, PackageFullName, ApplicationID
Thread/DCStart	A list of running process threads at the time the trace started.	StackBase, StackLimit, UserStackBase, UserStackLimit, StartAddr, Win32StartAddr, TebBase, SubProcessTag, ParentThreadID, ParentProcessID
Thread/DCEnd	A list of running process threads at the time the trace ended.	StackBase, StackLimit, UserStackBase, UserStackLimit, StartAddr, Win32StartAddr, TebBase, SubProcessTag, ParentThreadID, ParentProcessID
Thread/Start	A list of running process threads at the time the trace started.	StackBase, StackLimit, UserStackBase, UserStackLimit, StartAddr, Win32StartAddr, TebBase, SubProcessTag, ParentThreadID, ParentProcessID
Thread/End	A list of running process threads at the time the trace ended.	StackBase, StackLimit, UserStackBase, UserStackLimit, StartAddr, Win32StartAddr, TebBase, SubProcessTag, ParentThreadID, ParentProcessID

Fig. 6.1. Events in BootPerfDiagLogger.etl file.

Forensics Analysis of BootPerfDiagLogger.etl with ETLparser.exe

This section talks about the parsing of BootPerfDiagLogger.etl using **ETLparser.exe**. This simple command-line utility developed by Forensic Lunch[2] is advantageous in parsing ETL files. Its usage is straightforward, and it generates output in two formats, namely a CSV file and an SQLlite DB file. The figures below show the use and output of the ETLparse.exe tool.

Figure 6.2 shows the execution of parsing from ETLParser.exe tool. The command takes three arguments. First is the case folder name preceded

1 https://tinyurl.com/56hce3fd
2 https://github.com/forensiclunch/ETLParser

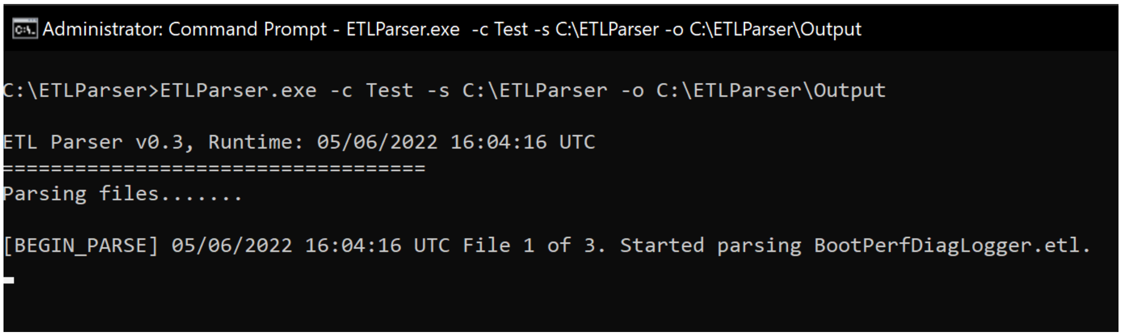


Fig. 6.2. The parsing of BootPerfDiagLogger.etl using ETLParser.exe.

by -c switch, then the source directory where ETL files resides preceded by -s switch and lastly, an output directory path preceded by -o switch.

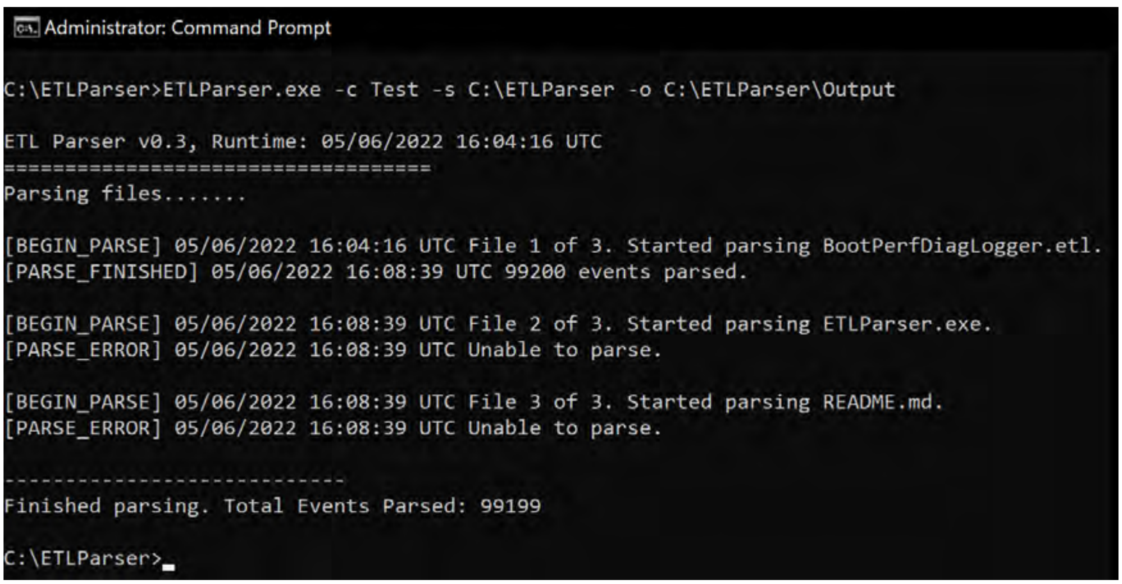


Fig. 6.3. The parsing of BootPerfDiagLogger.etl using ETLParser.exe.

Figure 6.3 shows the completion of the parsing procedure from the tool. It shows the total number of events parsed in three steps. It will report the start and end times of parsing. The concluding message displays the total events parsed.

Figure 6.4 shows the granular output from the CSV file generated by ETLParser.exe. The output shows the name of the logfile, timestamp of the event recording in UTC format, event name that triggered the event, provider name,

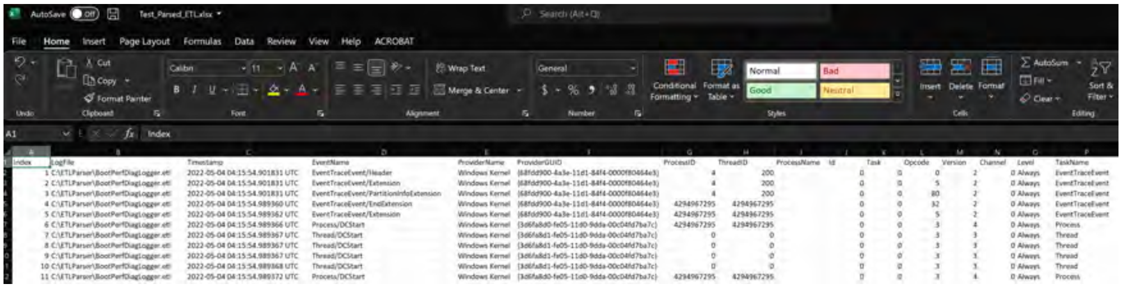


Fig. 6.4. The output of ETLParser.exe in Microsoft Excel file.

GUID, process ID, thread ID, process name, task, opcode, version, channel, level, task name. From a forensics standpoint, finding out any traces of malicious activity and persistence left by virulent software is extremely convenient.

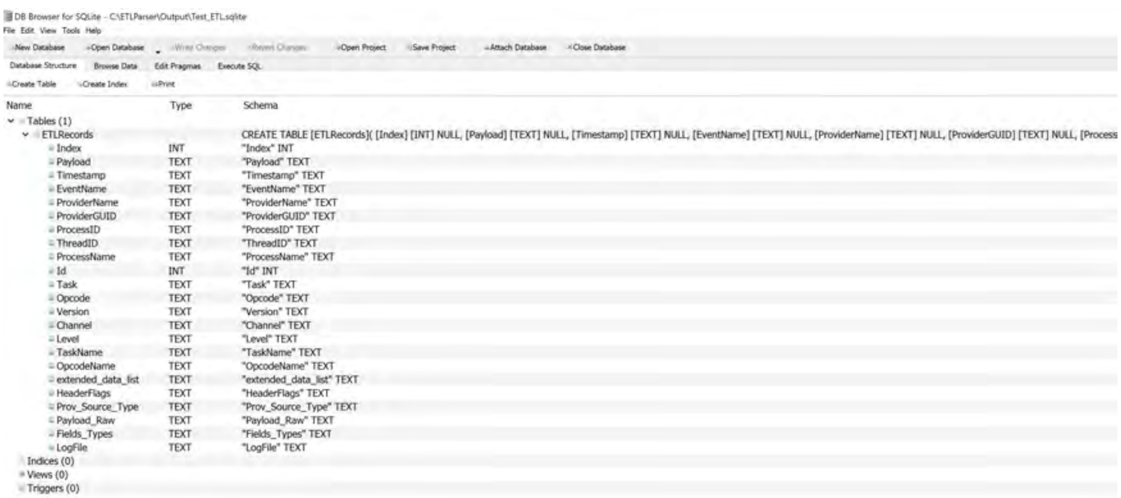


Fig. 6.5. The output of ETLParser.exe in DB Browser (SQLite) tool.

Figure 6.5 displays the DB Browser's SQLite file output generated by ETLParser.exe for the fields present in BootPerfDiagLogger.etl file. The output shows the name of the table, the type of fields, and the table's schema.

Forensics Analysis of BootPerfDiagLogger.etl with PerfView.exe

This section talks about the parsing of BootPerfDiagLogger.etl using **PerfView.exe**, developed by Microsoft[3]. This simple GUI tool parses ETL files efficiently and displays the parsed contents in the tool window itself. Figure 6.6 shows the output of PerfView displaying the heading of the parsed contents from our BootPerfDiagLogger.etl file. Figure 6.7 shows the details of trace and machine. Figures 6.8 and 6.9 exhibit the process summary information including the command-line execution of static and dynamic traces. Figures 6.10 and 6.11 displays the process details and event statistics.

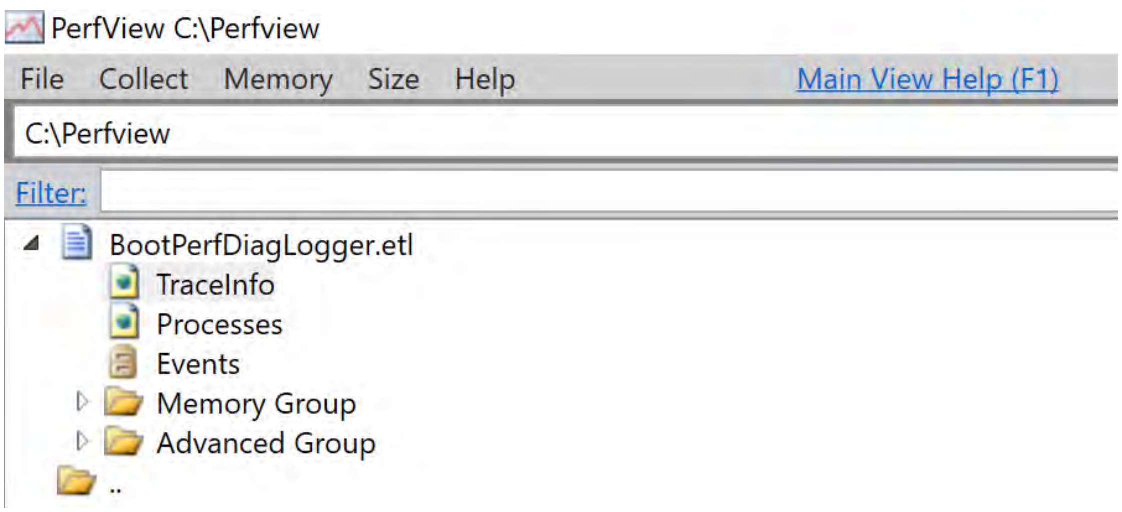


Fig. 6.6. The output of Perfview.exe displaying headers from the parsed etl file.

3 https://www.microsoft.com/en-us/download/details.aspx?id=28567

TraceInfo for BootPerfDiagLogger.etl in Perfview (C:\Perfview\BootPerfDiagLo

Back Forward

Information on the Trace and Machine

Machine Name	
Operating System	
OS Build Number	
UTC offset where data was collected	-7.00
UTC offset where PerfView is running	-5.00
Delta of Local and Collection Time	2.00
OS Boot Time	05/03/2022 23:15:54.500
Trace Start Time	05/03/2022 23:15:54.901
Trace End Time	05/03/2022 23:17:37.410
Trace Duration (Sec)	102.5
CPU Frequency (Mhz)	3,600
Number Of Processors	4
Memory Size (Meg)	0
Pointer Size	8
Sample Profile Interval (MSec)	1.00
Total Events	82,911
Lost Events	0
ETL File Size (MB)	16.8
No data collection log file found	

Fig. 6.7. The output of Perfview.exe displaying information of trace and machine.

Processes for BootPerfDiagLogger.etl in Perfview (C:\Perfview\BootPerfDiagLogger.etl)

Back Forward

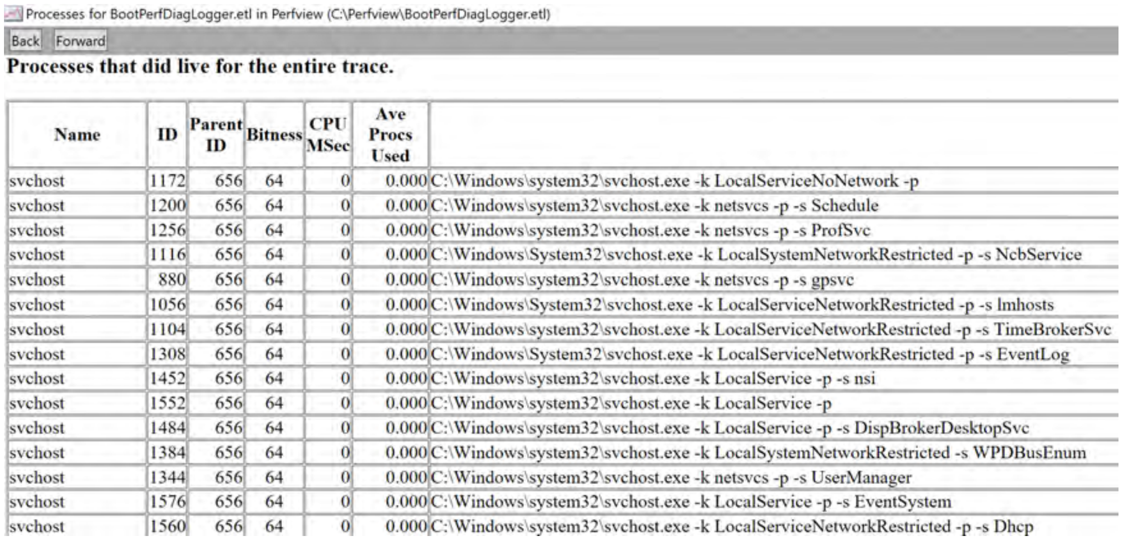
Process Summary

- View Process Data in Excel
- View Process Modules in Excel

Processes that did not live for the entire trace.

Name	ID	Parent ID	Bitness	CPU MSec	Ave Procs Used	Duration MSec	Start MSec	Exit Code	Command Line
taskhostw	1668	1200	64	0	0.000	80.177	73,082.736	0x0	taskhostw.exe $(Arg0)
dllhost	7028	856	64	0	0.000	5,096.019	46,461.136	0x0	C:\Windows\system32\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}
wlrmdr	5672	684	64	0	0.000	28.531	41,431.754	0x0	-c -s 0 -f 0 -t Empty -m Empty -a 0 -u Empty
taskhostw	7136	1200	64	0	0.000	132.583	40,805.983	0x0	taskhostw.exe SyncFromCloud
rundll32	5048	856	64	0	0.000	77.650	38,854.697	0x0	C:\Windows\System32\rundll32.exe C:\Windows\System32\shell32.dll,SHCreateLocalServerRunDll {9aa4600
dllhost	2168	856	64	0	0.000	5,647.280	36,785.911	0x0	C:\Windows\system32\DllHost.exe /Processid:{AB8902B4-09CA-4BB6-B78D-A8F59079A8D5}
WmiApSrv	3860	656	64	0	0.000	66,561.349	35,947.622	?	C:\Windows\system32\wbem\WmiApSrv.exe
svchost	2660	656	64	0	0.000	71,676.900	30,832.071	?	C:\Windows\system32\svchost.exe -k LocalServiceAndNoImpersonation -p -s SSDPSRV
svchost	6956	656	64	0	0.000	71,751.286	30,757.685	?	C:\Windows\System32\svchost.exe -k netsvcs -p -s BITS
vmtoolsd	4536	4520	64	0	0.000	72,037.519	30,471.452	?	"C:\Program Files\VMware\VMware Tools\vmtoolsd.exe" -n vmusr
SecurityHealthService	6300	656	64	0	0.000	72,432.654	30,076.317	?	C:\Windows\system32\SecurityHealthService.exe
SecurityHealthSystray	6396	4520	64	0	0.000	72,508.809	30,000.163	?	"C:\Windows\System32\SecurityHealthSystray.exe"
smartscreen	6200	856	64	0	0.000	72,576.440	29,932.531	?	C:\Windows\System32\smartscreen.exe -Embedding
WmiPrvSE	7160	856	64	0	0.000	72,823.392	29,685.579	?	C:\Windows\system32\wbem\wmiprvse.exe -Embedding
conhost	7084	7076	64	0	0.000	71.294	28,254.659	0x0	\??\C:\Windows\system32\conhost.exe 0xffffffff -ForceV1

Fig. 6.8. The output of Perfview.exe displaying dead process summary.

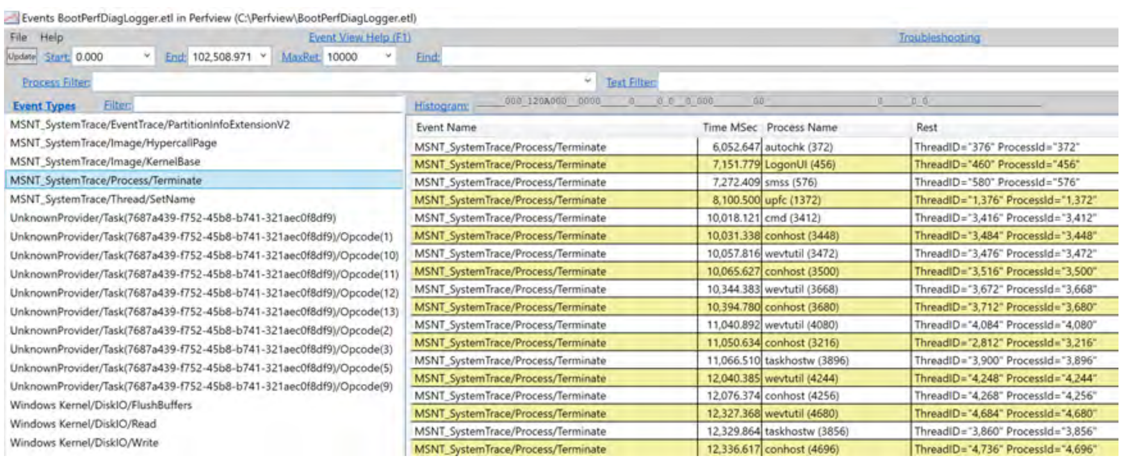

Processes for BootPerfDiagLogger.etl in Perfview (C:\Perfview\BootPerfDiagLogger.etl)

Back Forward

Processes that did live for the entire trace.

Name	ID	Parent ID	Bitness	CPU MSec	Ave Procs Used	
svchost	1172	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalServiceNoNetwork -p
svchost	1200	656	64	0	0.000	C:\Windows\system32\svchost.exe -k netsvcs -p -s Schedule
svchost	1256	656	64	0	0.000	C:\Windows\system32\svchost.exe -k netsvcs -p -s ProfSvc
svchost	1116	656	64	0	0.000	C:\Windows\System32\svchost.exe -k LocalSystemNetworkRestricted -p -s NcbService
svchost	880	656	64	0	0.000	C:\Windows\system32\svchost.exe -k netsvcs -p -s gpsvc
svchost	1056	656	64	0	0.000	C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p -s lmhosts
svchost	1104	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s TimeBrokerSvc
svchost	1308	656	64	0	0.000	C:\Windows\System32\svchost.exe -k LocalServiceNetworkRestricted -p -s EventLog
svchost	1452	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalService -p -s nsi
svchost	1552	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalService -p
svchost	1484	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalService -p -s DispBrokerDesktopSvc
svchost	1384	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalSystemNetworkRestricted -s WPDBusEnum
svchost	1344	656	64	0	0.000	C:\Windows\system32\svchost.exe -k netsvcs -p -s UserManager
svchost	1576	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalService -p -s EventSystem
svchost	1560	656	64	0	0.000	C:\Windows\system32\svchost.exe -k LocalServiceNetworkRestricted -p -s Dhcp

Fig. 6.9. The output of Perfview.exe displaying live process summary.


Events BootPerfDiagLogger.etl in Perfview (C:\Perfview\BootPerfDiagLogger.etl)

File Help Event View Help (F1) Troubleshooting

Update Start: 0.000 End: 102,508.971 MaxRet: 10000 Find:

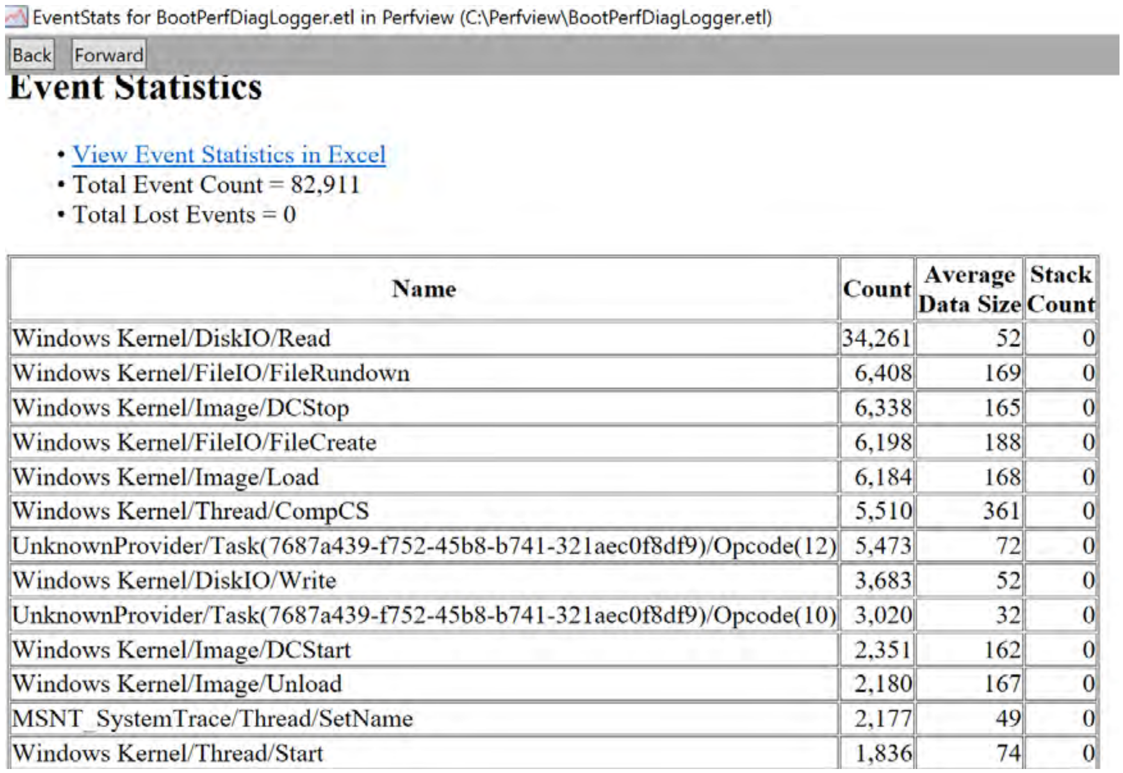
Process Filter: Text Filter:

Event Types Filter: Histogram:

MSNT_SystemTrace/EventTrace/PartitionInfoExtensionV2
MSNT_SystemTrace/Image/HypercallPage
MSNT_SystemTrace/Image/KernelBase
MSNT_SystemTrace/Process/Terminate
MSNT_SystemTrace/Thread/SetName
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(1)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(10)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(11)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(12)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(13)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(2)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(3)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(5)
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(9)
Windows Kernel/DiskIO/FlushBuffers
Windows Kernel/DiskIO/Read
Windows Kernel/DiskIO/Write

Event Name	Time MSec	Process Name	Rest
MSNT_SystemTrace/Process/Terminate	6,052.647	autochk (372)	ThreadID="376" ProcessId="372"
MSNT_SystemTrace/Process/Terminate	7,151.779	LogonUI (456)	ThreadID="460" ProcessId="456"
MSNT_SystemTrace/Process/Terminate	7,272.409	smss (576)	ThreadID="580" ProcessId="576"
MSNT_SystemTrace/Process/Terminate	8,100.500	upfc (1372)	ThreadID="1,376" ProcessId="1,372"
MSNT_SystemTrace/Process/Terminate	10,018.121	cmd (3412)	ThreadID="3,416" ProcessId="3,412"
MSNT_SystemTrace/Process/Terminate	10,031.338	conhost (3448)	ThreadID="3,484" ProcessId="3,448"
MSNT_SystemTrace/Process/Terminate	10,057.816	wevtutil (3472)	ThreadID="3,476" ProcessId="3,472"
MSNT_SystemTrace/Process/Terminate	10,065.627	conhost (3500)	ThreadID="3,516" ProcessId="3,500"
MSNT_SystemTrace/Process/Terminate	10,344.383	wevtutil (3668)	ThreadID="3,672" ProcessId="3,668"
MSNT_SystemTrace/Process/Terminate	10,394.780	conhost (3680)	ThreadID="3,712" ProcessId="3,680"
MSNT_SystemTrace/Process/Terminate	11,040.892	wevtutil (4080)	ThreadID="4,084" ProcessId="4,080"
MSNT_SystemTrace/Process/Terminate	11,050.634	conhost (3216)	ThreadID="2,812" ProcessId="3,216"
MSNT_SystemTrace/Process/Terminate	11,066.510	taskhostw (3896)	ThreadID="3,900" ProcessId="3,896"
MSNT_SystemTrace/Process/Terminate	12,040.385	wevtutil (4244)	ThreadID="4,248" ProcessId="4,244"
MSNT_SystemTrace/Process/Terminate	12,076.374	conhost (4256)	ThreadID="4,268" ProcessId="4,256"
MSNT_SystemTrace/Process/Terminate	12,327.368	wevtutil (4680)	ThreadID="4,684" ProcessId="4,680"
MSNT_SystemTrace/Process/Terminate	12,329.864	taskhostw (3856)	ThreadID="3,860" ProcessId="3,856"
MSNT_SystemTrace/Process/Terminate	12,336.617	conhost (4696)	ThreadID="4,736" ProcessId="4,696"

Fig. 6.10. The output of Perfview.exe displaying event types and details.



EventStats for BootPerfDiagLogger.etl in Perfview (C:\Perfview\BootPerfDiagLogger.etl)

Back Forward

Event Statistics

- View Event Statistics in Excel
- Total Event Count = 82,911
- Total Lost Events = 0

Name	Count	Average Data Size	Stack Count
Windows Kernel/DiskIO/Read	34,261	52	0
Windows Kernel/FileIO/FileRundown	6,408	169	0
Windows Kernel/Image/DCStop	6,338	165	0
Windows Kernel/FileIO/FileCreate	6,198	188	0
Windows Kernel/Image/Load	6,184	168	0
Windows Kernel/Thread/CompCS	5,510	361	0
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(12)	5,473	72	0
Windows Kernel/DiskIO/Write	3,683	52	0
UnknownProvider/Task(7687a439-f752-45b8-b741-321aec0f8df9)/Opcode(10)	3,020	32	0
Windows Kernel/Image/DCStart	2,351	162	0
Windows Kernel/Image/Unload	2,180	167	0
MSNT_SystemTrace/Thread/SetName	2,177	49	0
Windows Kernel/Thread/Start	1,836	74	0

Fig. 6.11. The output of Perfview.exe showing event statistics.

Forensics Analysis of BootPerfDiagLogger.etl with FullEventLogView

This section talks about the parsing of BootPerfDiagLogger.etl using Nirsoft's **FullEventLogView**[4]. This efficient and easy to use tool parses ETL files conveniently. Figure 6.12 shows the detailed output of the file parsed using FullEventLogView.

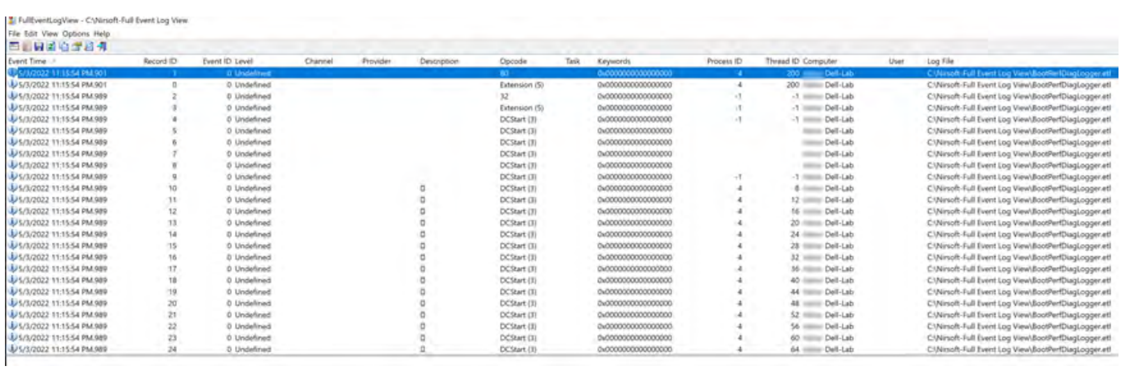


Fig. 6.12. The output of FullEventLogView.

[4] https://www.nirsoft.net/utils/fulleventlogview-x64.zip

Forensics Analysis of BootPerfDiagLogger.etl with SVCLogViewer

This section talks about the parsing of BootPerfDiagLogger.etl using **Svclogviewer** developed by Martijn Stolk[5]. Figure 6.13 displays the output of the tool after parsing the BootPerfDiagLogger.etl file.

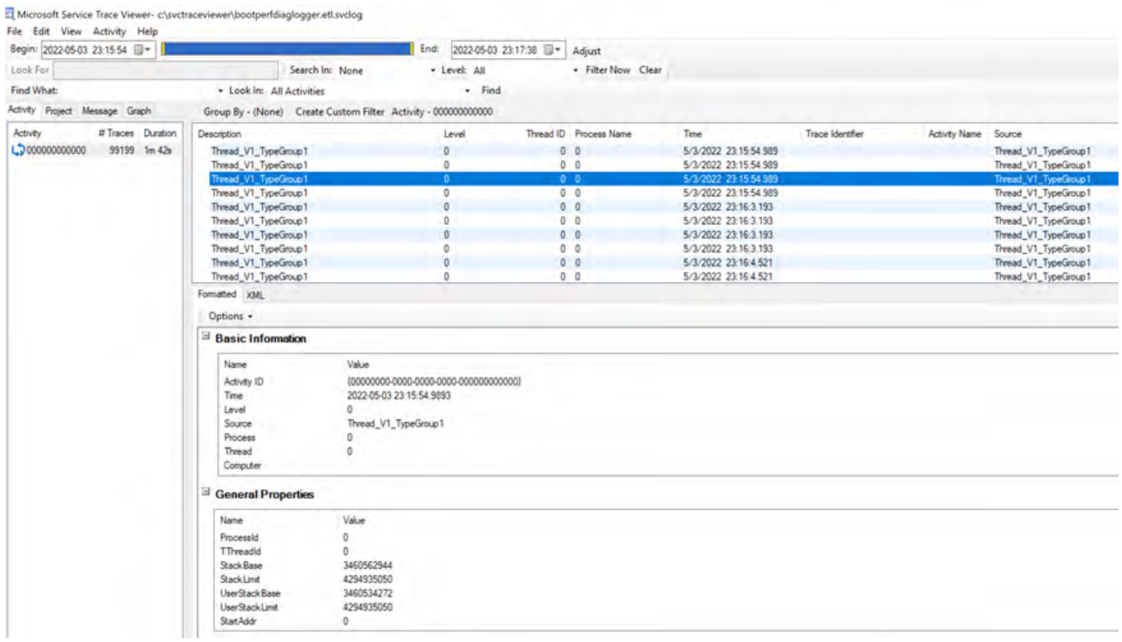


Fig. 6.13. The output of SvcLogViewer.

Forensics Analysis of BootPerfDiagLogger.etl with TraceFMT

TraceFMT is another simple command-line utility for parsing ETL files from Windows Development Toolkit (WDK) by Microsoft[6]. Its use is straightforward; the name of the ETL file has to be supplied as an argument after typing traceFMT in the command prompt (CMD). The tool creates two txt files. The first file shows the summary of the event parsed, whereas the second txt file has the content of parsed BootPerfDiagLogger ETL file. Figures 6.14 and 6.15 show the output TraceFMT tool.

[5] https://github.com/martijns/SvclogViewer
[6] https://docs.microsoft.com/en-us/windows-hardware/drivers/other-wdk-downloads

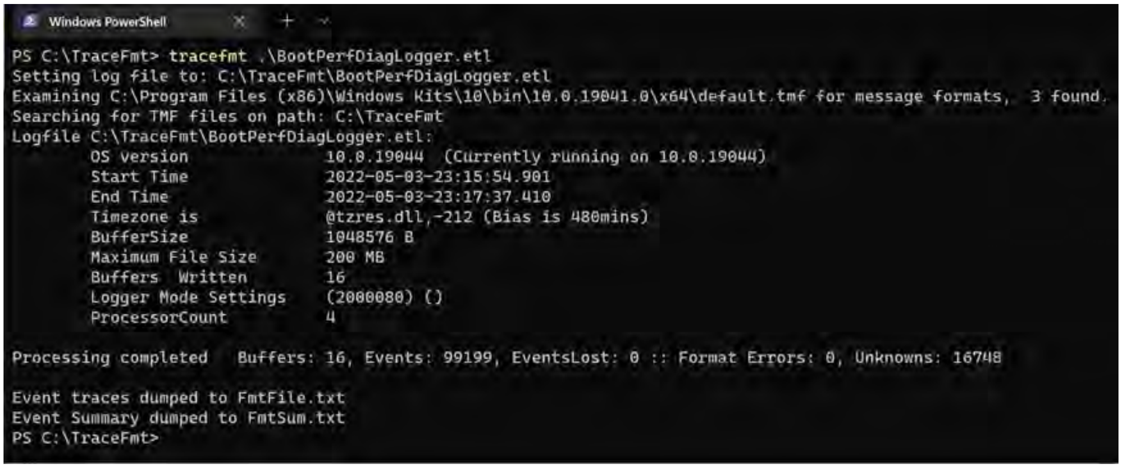


Fig. 6.14. The output of TraceFMT.

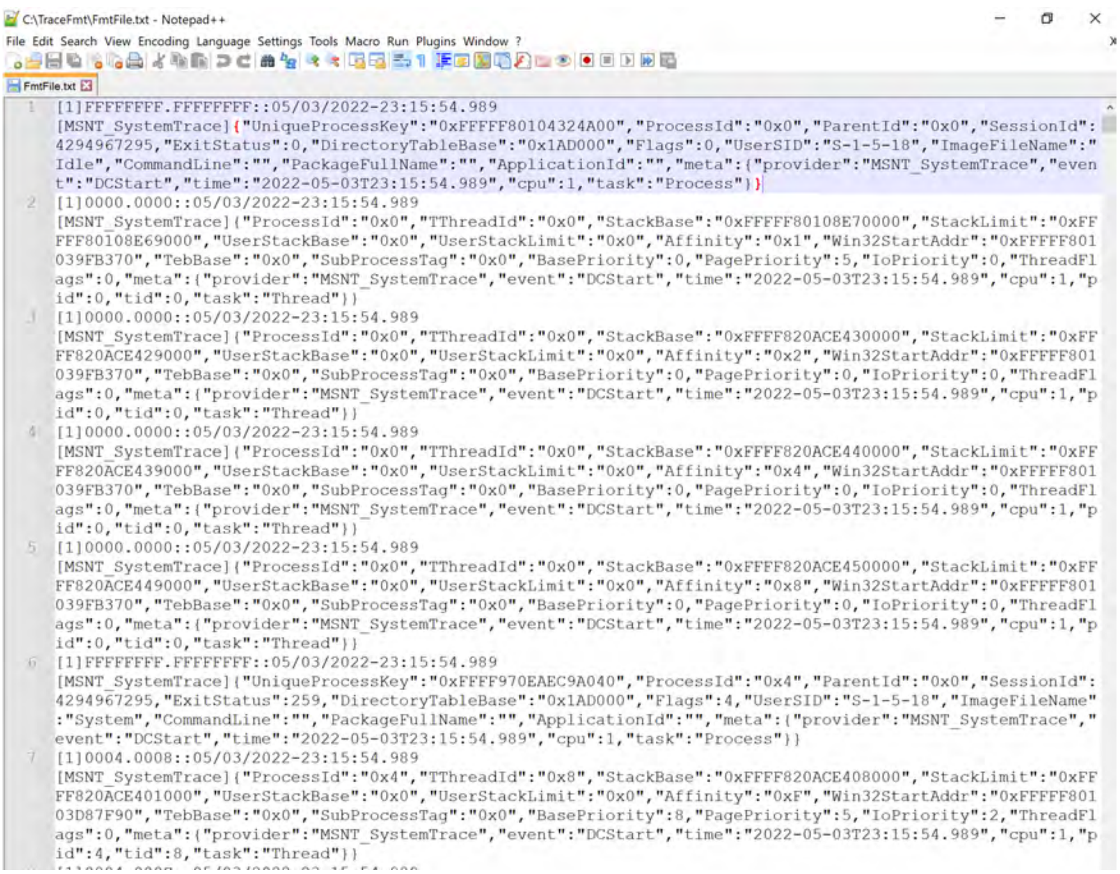


Fig. 6.15. The output of TraceFMT generated txt file of the parsed ETL file.

Forensics Analysis of BootPerfDiagLogger.etl with Windows Performance Analyzer

This section talks about the parsing of BootPerfDiagLogger.etl using **Windows Performance Analyzer**[7]. This GUI application parses BootPerfDiagLogger ETL file and shows system activity, processes, images, computation information, process name, event name, duration of the event and processes. Figure 6.16 shows the detailed output of the file parsed using Windows Performance Analyzer.

Fig. 6.16. The output of Windows Performance Analyzer.

7 https://docs.microsoft.com/en-us/windows-hardware/test/wpt/windows-performance-analyzer

Forensics Importance of Analyzing BootPerfDiagLogger.etl with different Tools

We used different tools to conduct a comprehensive forensics analysis on BootPerfDiagLogger.etl file (previously BootCKCL.etl till Windows 10 earlier versions). All this information will help a forensics examiner get a complete picture for drafting a report. We can obtain the following information using the tools:

- Total number of event in the ETL file
- Process IDs
- Process name
- Computer name
- DLL files associated with the malicious process
- Duration for which a process ran
- Operating system version
- Time zone information
- Event trace time start and end
- Command-line executable for the process

CHAPTER VII

Digital Forensics in USB NVMe SSDs with WriteBlocker

Storage is a vital mechanism that enables a computer system to temporarily or permanently retain data. Computer storage devices store digital information on itself. These devices are ubiquitous, and a fundamental component of most digital devices since they allow users to store all kinds of digital media [72].

There are currently numerous computer storage devices available in the market such as hard drives (HDDs), memory cards, USB flash drives, solid-state drives (SSDs), and non-volatile memory express solid-state drives (NVMe SSDs), to name a few. Until the late 2000s, HDDs had the highest market share in terms of storage devices, but recently there has been a gradual shift towards SSDs [73]. The shift from using HDDs to NVMe SSDs in digital devices is primarily driven by latter's performance, durability, and reliability, to name a few. Therefore, a user can get his task done ten times faster when using an NVMe SSD compared to an HDD [74].

Since the NVMe SSD technology is relatively new; there is not much prior sound digital forensic research in this field. Unlike HDD, which stores data on fundamental data storage units called sectors and can be written and rewritten multiple times, NVMe SSD stores data on flash chips internally controlled by the controller chip on the storage device. There are significant discrepancies in the underpinnings between the two forms of storage media, which have serious implications for security and digital forensics. When it comes to conducting file recovery on HDD, we have the certainty of finding the data as it stays on the device's storage unit. Since only the address reference to the stored data is removed after deletion [54], it is not always guaranteed that deleted data is erased from the hard drive. So, when we take a forensics image of an HDD and recover data, we would find the data as long it is not overwritten. On the contrary, this is not the case for NVMe SSD, as the controller chip inside the device is constantly moving data around the flash chips to prolong the life of the storage device. The constant movement of data for elongating the life of an NVMe SSD is achieved by the concept of wear-leveling, which the controller implements autonomously [75]. Even if they are not connected, SSDs can sometimes delete data independently. As a result, standard techniques aimed at preserving forensics data on solid-state drives are ineffective. In addition, they could also result in potential evidence being lost, destroyed, or corrupted, thus, making evidence inadmissible in court. [76].

To address the problem of file recovery in NVMe SSDs, we conducted a sound forensic analysis on four NVMe SSDs: Samsung, Seagate, Western Digital, and Silicon Power. These storage devices were used inside USB enclosure adapters. We aimed to determine the number of files recovered after they were deleted from these devices. We prepared the NVMe SSDs by installing Windows 10 operating system on them. To recover the files and conduct the forensic analysis, we used AccessData FTK [77], Autopsy, and WinHex [78] disk editor. In addition, we explained our forensic findings based on the observations from four the different brands of SSDs with varying controller chips.

Experimental Setup with USB WriteBlocker

Table 7.1 below shows the technical specifications of the equipment used throughout the experiment, including the digital forensic workstation and various hardware and software tools.

Table 7.1. Equipment used in the experiment.

Tools	Name
NVMe SSD 1	Samsung V-NAND SSD 970 Evo Plus
NVMe SSD 2	Seagate Barracuda 510 250GB NVMe SSD
NVMe SSD 3	Western Digital SN550 250GB NVMe SSD
NVMe SSD 4	Silicon Power 3D-NAND NVMe SSD
Operating System	Windows 10 Pro v21H2
Forensic Analysis Tool	AccessData FTK 7.5 and WinHex
Forensics Acquisition Tool	AccessData FTK Imager 4.7
WriteBlocker	Wiebetech USB 3.0 WriteBlocker
Workstation	CPU: Intel Xeon W-2123 — RAM : 80GB

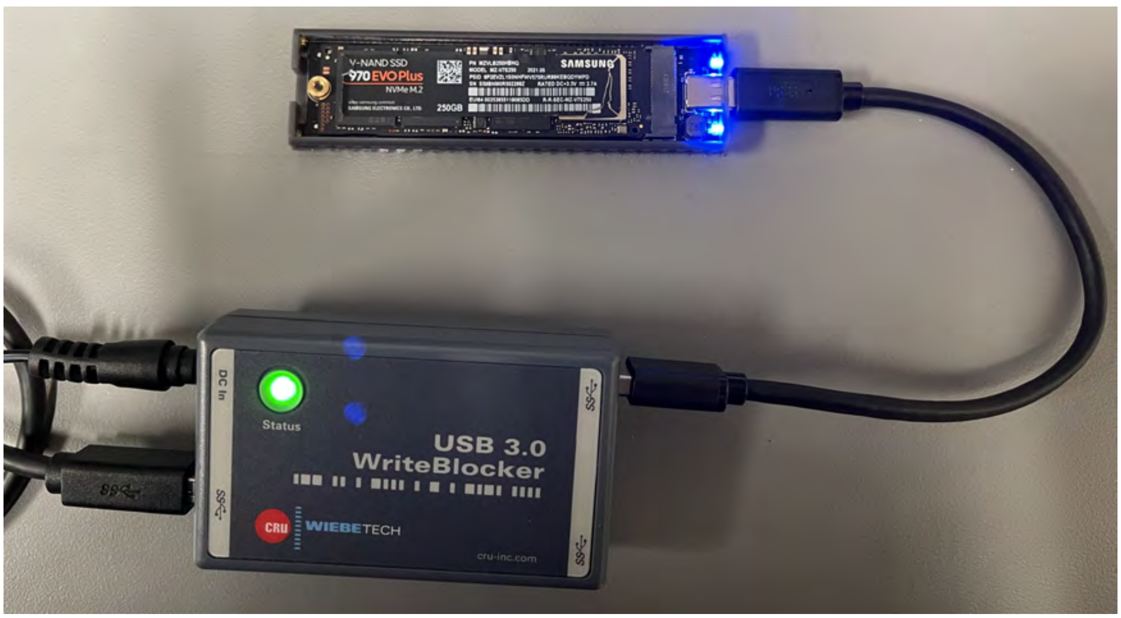


Fig. 7.1. Samsung NVMe SSD attached with USB WriteBlocker.

Fig. 7.2. Seagate NVMe SSD attached with USB WriteBlocker.

Fig. 7.3. Western Digital NVMe SSD attached with USB WriteBlocker.

Fig. 7.4. Silicon Power NVMe SSD attached with USB WriteBlocker.

Specifics of SSDs

The experiment was based on four brands of NVMe SSDs, namely Samsung, Seagate, Western Digital, and Silicon Power. This was due to their significant market dominance and reliability [79]. The four brands and the specific models were chosen to reflect a real-life scenario since the specifications of these SSDs mimic the specifications of a typical SSD a user might own. Additionally, the choice of SSDs makes the experiment more meaningful to the digital forensic community as these are the most prominent specifications of SSDs embedded in a laptop or desktop computer. Tables 7.2 and 7.3, list down the name, model, product number (P/N), storage capacity, number of flash chips, type of NVMe flash chip, and the controller information of the NVMe SSDs.

Table 7.2. Information of Samsung and Seagate NVMe SSDs used in the experiment.

SSD Information	Samsung NVMe Specification 1.3
Name	Samsung NVMe V-NAND SSD 970 Evo Plus NVMe M.2
Model	MZ-V7S250
P/N	MZVLB250HBHQ
Storage Capacity	250 GB
Number of flash chips inside	2
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Samsung S4LR020 — 2117 ARM — Pheonix

SSD Information	Seagate NVMe Specification 1.3
Name	Seagate Barracuda 510 250GB NVMe SSD
Model	ZP250CM30001
P/N	2NS312-300
Storage Capacity	250 GB
Number of flash chips inside	4
Type of NVMe NAND Flash	3D TLC NAND
Controller information	SKHynix - H5AN4G6NBJR

Table 7.3. Information of Western Digital and Silicon Power NVMe SSDs used in the experiment.

SSD Information	WD NVMe Specification 1.4
Name	Western Digital SN550 250GB NVMe SSD
Model	WDS250G2B0C-00PXH0/21146P801302
P/N	87161901478830731375399388282263
Storage Capacity	250 GB
Number of flash chips inside	4
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Sandisk 20-82-10023-A1 — 1015ZKLY0KN

SSD Information	Silicon Power NVMe Specification 1.3
Name	Silicon Power 3D-NAND NVMe SSD
Model	A-60
P/N	SP256GBP34A60M28
Storage Capacity	256 GB
Number of flash chips inside	2
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Phison PS5013-E13-31—C02102E— TB5V79/ 001BB

Methodology and Experiment Initiation

This section lists down and explains the procedures and configurations we followed and assigned throughout the experiment.

1. The partition scheme used for the NVMe SSDs inside the USB enclosure adapters: **MBR (Master Boot Record)**
2. The number of partitions in each NVMe SSD: **1**

3. The file system of the one partition: **NTFS**
4. Prior to copying the files to the devices from Digital Corpora [80], we checked the **TRIM** status in Windows 10 by issuing the following command through the command prompt.

> **fsutil behavior query DisableDeleteNotify**
>
> **If the output is 1, then TRIM is disabled. If the output is 0, then TRIM is enabled.*
>
> **To enable TRIM**: fsutil behavior set DisableDeleteNotify 0
>
> **To disable TRIM**: fsutil behavior set DisableDeleteNotify 1

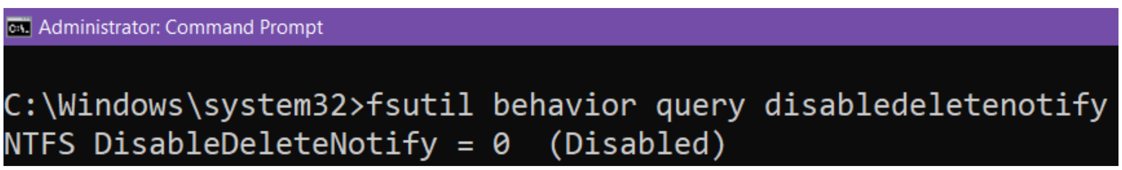


Fig. 7.5. The status of TRIM in Windows 10 using fsutil command.

Case scenario: TRIM ON from Windows 10 operating system with the USB WriteBlocker

1. We copied the commonly used file types from the Digital Corpora dataset [80] to the four NVMe SSDs. We used large file sizes to exhaust the storage drives' capacity.
2. We then kept the files for one day with no user activity by keeping the drive attached to the USB port.
3. Next, we deleted (shift+delete) the files from the devices and waited for one day before taking four forensic images of the four NVMe SSDs respectively using the physical USB writeblocker.
   (a) We took four forensic images: three consecutive images with one day gap and last image after a span of four days from the third acquisition.

4. We analyzed the images in AccessData FTK and Autopsy for the NVMe storage devices.
5. We performed file recovery of the deleted files from the forensics images in TRIM ON case.
6. Based on our results from the file recovery and WinHex analysis we documented the effects of wear-leveling.

Case scenario: TRIM OFF from Windows 10 operating system with the USB WriteBlocker

1. Firstly, we disabled TRIM using Windows 10 command prompt before copying the files.
2. We copied the commonly used file types from the Digital Corpora dataset [80] to the four NVMe SSDs. We used large file sizes to exhaust the storage drives' capacity.
3. We then kept the files for one day with no user activity by keeping the drive attached to the USB port.
4. Next, we deleted (shift+delete) the files from the devices and waited for one day before taking four forensic images of the four NVMe SSDs respectively using the physical USB writeblocker.
   (a) We took four forensic images: three consecutive images with one day gap and last image after a span of four days from the third acquisition.
5. We analyzed the images in AccessData FTK and Autopsy for the NVMe storage devices.
6. We performed file recovery of the deleted files from the forensics images in TRIM OFF case.
7. Like the TRIM ON case, based on our results from the file recovery and WinHex analysis we documented the effects of wear-leveling.

Experiment Results, Analysis, and Discussion

In this section, we provided the results of the file recovery performed using the AccessData FTK and Autopsy tools. Tables 7.4 and 7.5 show the timeline information of forensic image acquisition in both TRIM ON and TRIM OFF cases of Samsung, Seagate, Western Digital (WD), and Silicon Power (SP) NVMe SSDs, respectively.

Table 7.4. Timeline information of forensic file acquisition.

TRIM ON information			
Samsung NVMe	**Time**	**Seagate NVMe**	**Time**
Copy file date	7:06 pm 10/18/21	Copy file date	5:38 pm 10/18/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	7:06 pm 10/19/21	Delete files	5:38 pm 10/19/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	7:06 pm 10/20/21	1st image	5:38 pm 10/20/21
2nd image	7:06 pm 10/21/21	2nd image	5:38 pm 10/21/21
3rd image	7:06 pm 10/22/21	3rd image	5:38 pm 10/22/21
4th image	7:06 pm 10/26/21	4th image	5:38 pm 10/26/21
TRIM OFF information			
Samsung NVMe	**Time**	**Seagate NVMe**	**Time**
Copy file date	7:51 pm 9/29/21	Copy file date	7:51 pm 9/29/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	7:51 pm 9/30/21	Delete files	7:51 pm 9/30/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	7:51 pm 10/1/21	1st image	7:51 pm 10/1/21
2nd image	7:51 pm 10/2/21	2nd image	7:51 pm 10/2/21
3rd image	7:51 pm 10/3/21	3rd image	7:51 pm 10/3/21
4th image	7:51 pm 10/7/21	4th image	7:51 pm 10/7/21

Table 7.5. Timeline information of forensic file acquisition.

TRIM ON information			
WD NVMe	**Time**	**SP NVMe**	**Time**
Copy file date	11:22 pm 10/18/21	Copy file date	9:02 pm 10/10/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	11:22 pm 10/19/21	Delete files	9:02 pm 10/11/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	11:22 pm 10/20/21	1st image	9:02 pm 10/12/21
2nd image	11:22 pm 10/21/21	2nd image	9:02 pm 10/13/21
3rd image	11:22 pm 10/22/21	3rd image	9:02 pm 10/14/21
4th image	11:22 pm 10/26/21	4th image	9:02 pm 10/18/21
TRIM OFF information			
WD NVMe	**Time**	**SP NVMe**	**Time**
Copy file date	10:51 pm 9/29/21	Copy file date	10:48 pm 9/28/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	10:51 pm 9/30/21	Delete files	10:48 pm 9/29/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	10:51 pm 10/1/21	1st image	10:48 pm 9/30/21
2nd image	10:51 pm 10/2/21	2nd image	10:48 pm 10/1/21
3rd image	10:51 pm 10/3/21	3rd image	10:48 pm 10/2/21
4th image	10:51 pm 10/7/21	4th image	10:48 pm 10/6/21

Samsung and Seagate TRIM ON Analysis

The TRIM command allows the operating system to tell the SSD that specific sections are no longer needed. As a result, the SSD controller can now undertake many of the processes required to clear data well ahead of any request from the operating system. These internal procedures could even be carried out when the SSD is under low load, hiding or masking the activity from the user.

Despite this, the TRIM ON analysis on both the Samsung NVMe and Seagate NVMe SSDs' forensics images showed that all files were recovered using the AccessData FTK and Autopsy tools. However, the controller chip did not act on files under 693 bytes in Samsung NVMe SSD and 696 bytes in Seagate NVMe

SSD, respectively. As a result, they were all intact without any content wiped or corrupted. In addition, files greater than 693 bytes in Samsung NVMe, and 696 bytes in Seagate NVMe SSD were all corrupted, i.e., their file contents were all zeroed out and hence were rendered unusable. Tables 7.6, 7.7, 7.8, and 7.9 give the statistics of the different files used from the Digital Corpora dataset and the files recovered from Samsung and Seagate NVMe SSDs in TRIM on case.

Table 7.6. The number of files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Samsung FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 693 bytes were intact after recovery in Samsung NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 7.7. The number of files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Samsung Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 693 bytes were intact after recovery in Samsung NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 7.8. The number of files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Seagate FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***

*: All files recovered but corrupted.
** : All files recovered but 8280 corrupted + 92 not corrupted.
*** : All files recovered but 69 corrupted + 46 not corrupted.
Note:
1) Files under 696 bytes were intact after recovery in Seagate NVMe SSD.
2) For corrupted files, the size of the files was the same, but the contents were wiped out.

Table 7.9. The number of files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Seagate Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 696 bytes were intact after recovery in Seagate NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

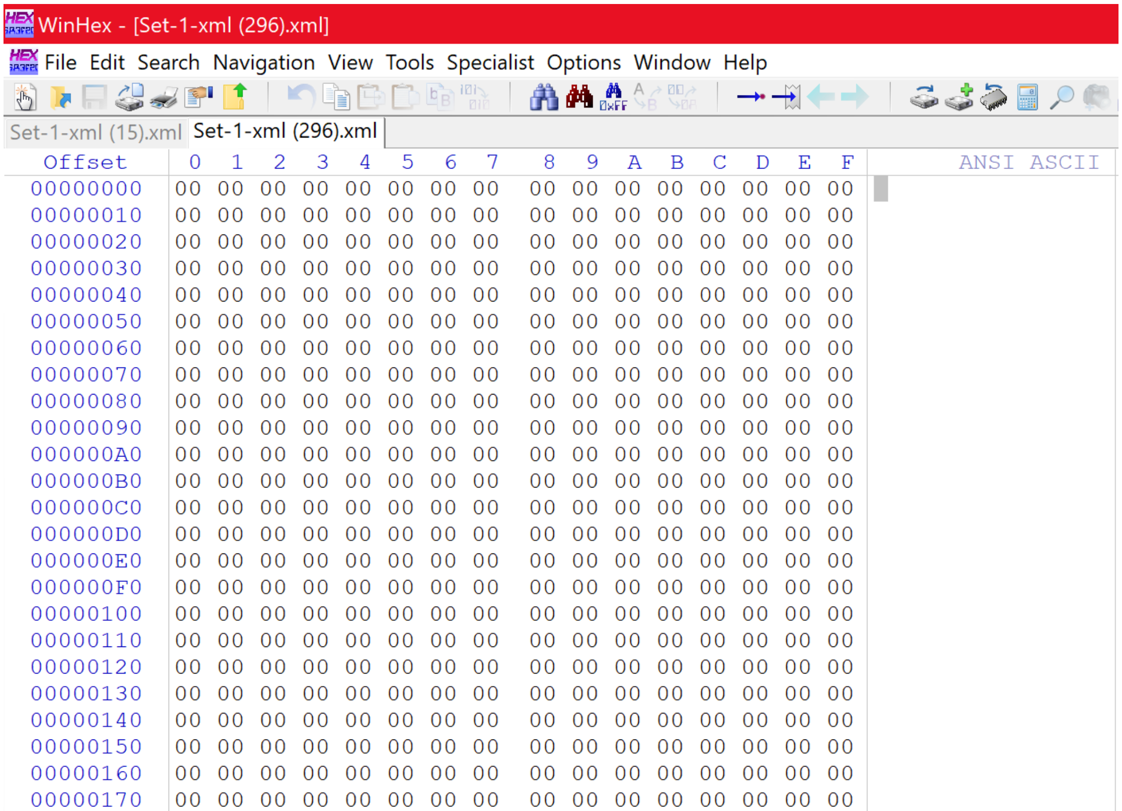


Fig. 7.6. File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.6 shows a snippet of an XML file with regards to the Samsung NVMe SSD TRIM ON case using a USB WriteBlocker. The file which over 693 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, but the contents of the file were corrupted, making the file unusable, as shown by the zeroes.

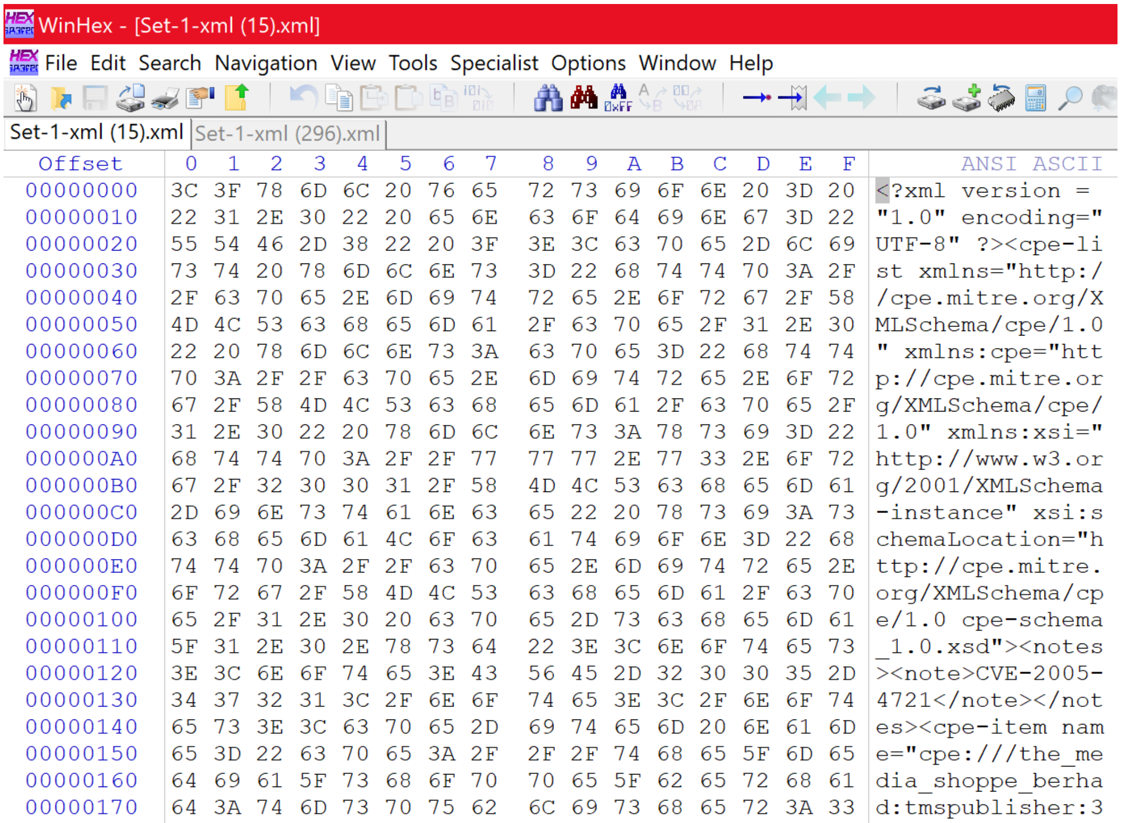


Fig. 7.7. File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.7 shows a snippet of the Set-1-xml(15).xml file with regards to the Samsung NVMe SSD TRIM ON case using a USB WriteBlocker. The file under 693 bytes was opened in the Win Hex tool. As seen from the experimental results, the file was recovered, and the contents of the file were intact.

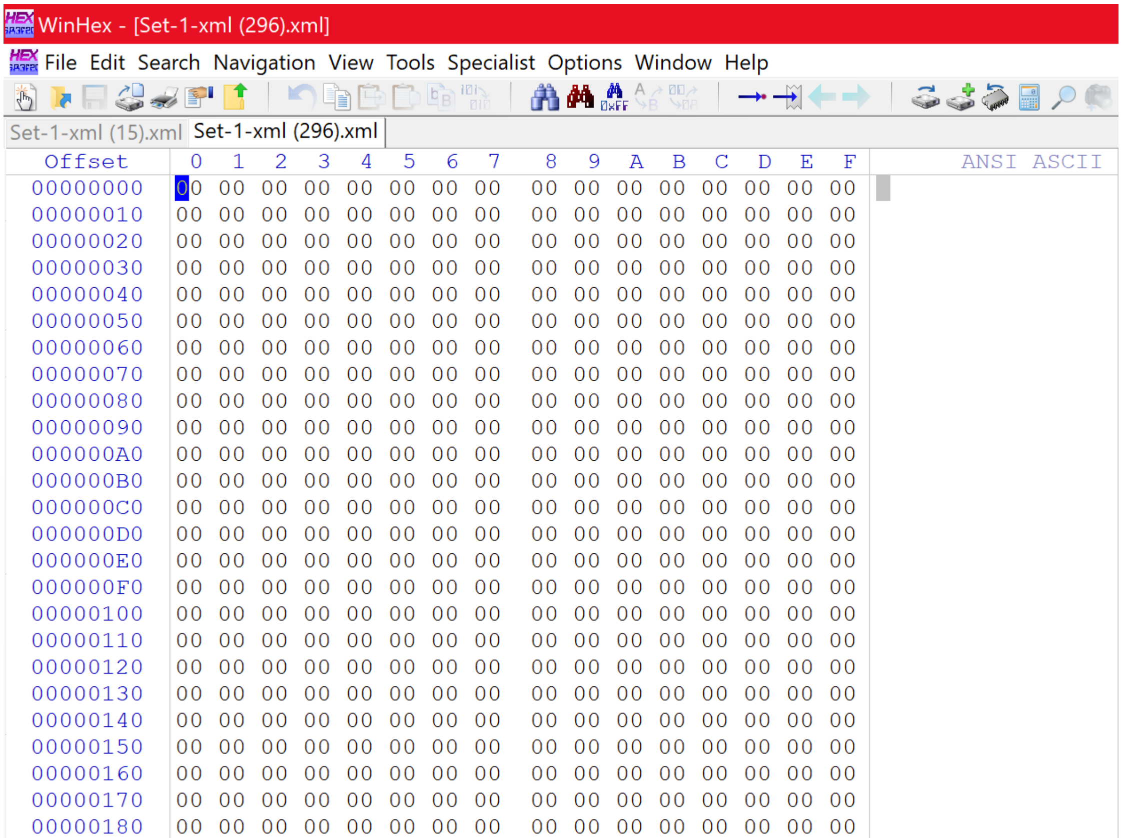
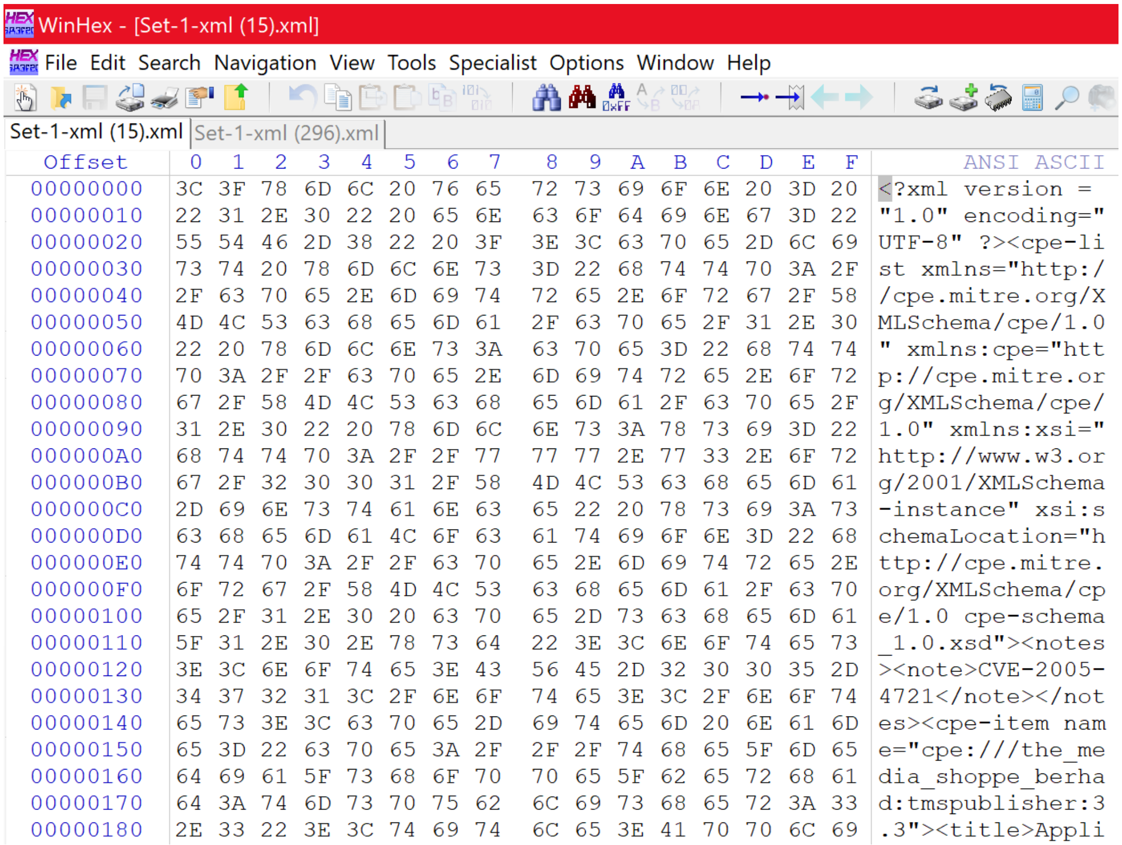


Fig. 7.8. File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.8 shows a snippet of an XML file with regards to the Seagate NVMe SSD TRIM ON case using a USB WriteBlocker. The file over 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, but the contents of the file were corrupted, making the file unusable, as shown by the zeroes.



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	ANSI ASCII
00000000	3C	3F	78	6D	6C	20	76	65	72	73	69	6F	6E	20	3D	20	<?xml version =
00000010	22	31	2E	30	22	20	65	6E	63	6F	64	69	6E	67	3D	22	"1.0" encoding="
00000020	55	54	46	2D	38	22	20	3F	3E	3C	63	70	65	2D	6C	69	UTF-8" ?><cpe-li
00000030	73	74	20	78	6D	6C	6E	73	3D	22	68	74	74	70	3A	2F	st xmlns="http:/
00000040	2F	63	70	65	2E	6D	69	74	72	65	2E	6F	72	67	2F	58	/cpe.mitre.org/X
00000050	4D	4C	53	63	68	65	6D	61	2F	63	70	65	2F	31	2E	30	MLSchema/cpe/1.0
00000060	22	20	78	6D	6C	6E	73	3A	63	70	65	3D	22	68	74	74	" xmlns:cpe="htt
00000070	70	3A	2F	2F	63	70	65	2E	6D	69	74	72	65	2E	6F	72	p://cpe.mitre.or
00000080	67	2F	58	4D	4C	53	63	68	65	6D	61	2F	63	70	65	2F	g/XMLSchema/cpe/
00000090	31	2E	30	22	20	78	6D	6C	6E	73	3A	78	73	69	3D	22	1.0" xmlns:xsi="
000000A0	68	74	74	70	3A	2F	2F	77	77	77	2E	77	33	2E	6F	72	http://www.w3.or
000000B0	67	2F	32	30	30	31	2F	58	4D	4C	53	63	68	65	6D	61	g/2001/XMLSchema
000000C0	2D	69	6E	73	74	61	6E	63	65	22	20	78	73	69	3A	73	-instance" xsi:s
000000D0	63	68	65	6D	61	4C	6F	63	61	74	69	6F	6E	3D	22	68	chemaLocation="h
000000E0	74	74	70	3A	2F	2F	63	70	65	2E	6D	69	74	72	65	2E	ttp://cpe.mitre.
000000F0	6F	72	67	2F	58	4D	4C	53	63	68	65	6D	61	2F	63	70	org/XMLSchema/cp
00000100	65	2F	31	2E	30	20	63	70	65	2D	73	63	68	65	6D	61	e/1.0 cpe-schema
00000110	5F	31	2E	30	2E	78	73	64	22	3E	3C	6E	6F	74	65	73	_1.0.xsd"><notes
00000120	3E	3C	6E	6F	74	65	3E	43	56	45	2D	32	30	30	35	2D	><note>CVE-2005-
00000130	34	37	32	31	3C	2F	6E	6F	74	65	3E	3C	2F	6E	6F	74	4721</note></not
00000140	65	73	3E	3C	63	70	65	2D	69	74	65	6D	20	6E	61	6D	es><cpe-item nam
00000150	65	3D	22	63	70	65	3A	2F	2F	2F	74	68	65	5F	6D	65	e="cpe:///the_me
00000160	64	69	61	5F	73	68	6F	70	70	65	5F	62	65	72	68	61	dia_shoppe_berha
00000170	64	3A	74	6D	73	70	75	62	6C	69	73	68	65	72	3A	33	d:tmspublisher:3
00000180	2E	33	22	3E	3C	74	69	74	6C	65	3E	41	70	70	6C	69	.3"><title>Appli

Fig. 7.9. File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.9 shows a snippet of the Set-1-xml(15).xml file with regards to the Seagate NVMe SSD TRIM ON case using a USB WriteBlocker. The file under 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered and the contents of the file were intact.

Samsung and Seagate TRIM OFF Analysis

Interestingly, all files were recovered successfully in the TRIM OFF analysis from the four forensics Samsung NVMe and Seagate NVMe SSD images, respectively. This is because the TRIM OFF feature prevents the computer's operating system from notifying the SSD to erase useless data blocks. Thus, the SSD controller no longer manages all of the available storage space. Hence, in our experiment, the controller chip did not wipe/clear the pages of the storage devices. Therefore, this time the contents of all the files were intact. i.e., files could be opened and worked on regularly. Furthermore, there was no instance of file corruption in the case. Tables 7.10, 7.11, 7.12, and 7.13 show the statistics of different files used and the files that were recovered.

Table 7.10. The number of files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Samsung FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
All files recovered and intact in TRIM OFF case of Samsung NVMe SSD					

Table 7.11. The number of files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Samsung Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
All files recovered and intact in TRIM OFF case of Samsung NVMe SSD					

Table 7.12. The number of files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Seagate FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
All files recovered and intact in TRIM OFF case of Seagate NVMe SSD					

Table 7.13. The number of files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Seagate Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
All files recovered and intact in TRIM OFF case of Seagate NVMe SSD					

Figure 7.10 shows a snippet of the Set-1-xml(296).xml file in the Samsung NVMe SSD TRIM OFF case using a USB WriteBlocker. The file over 693 bytes in size was opened in the WinHex tool to be analyzed. As seen from the experimental results, the file was recovered, and the contents of the file were not wiped.

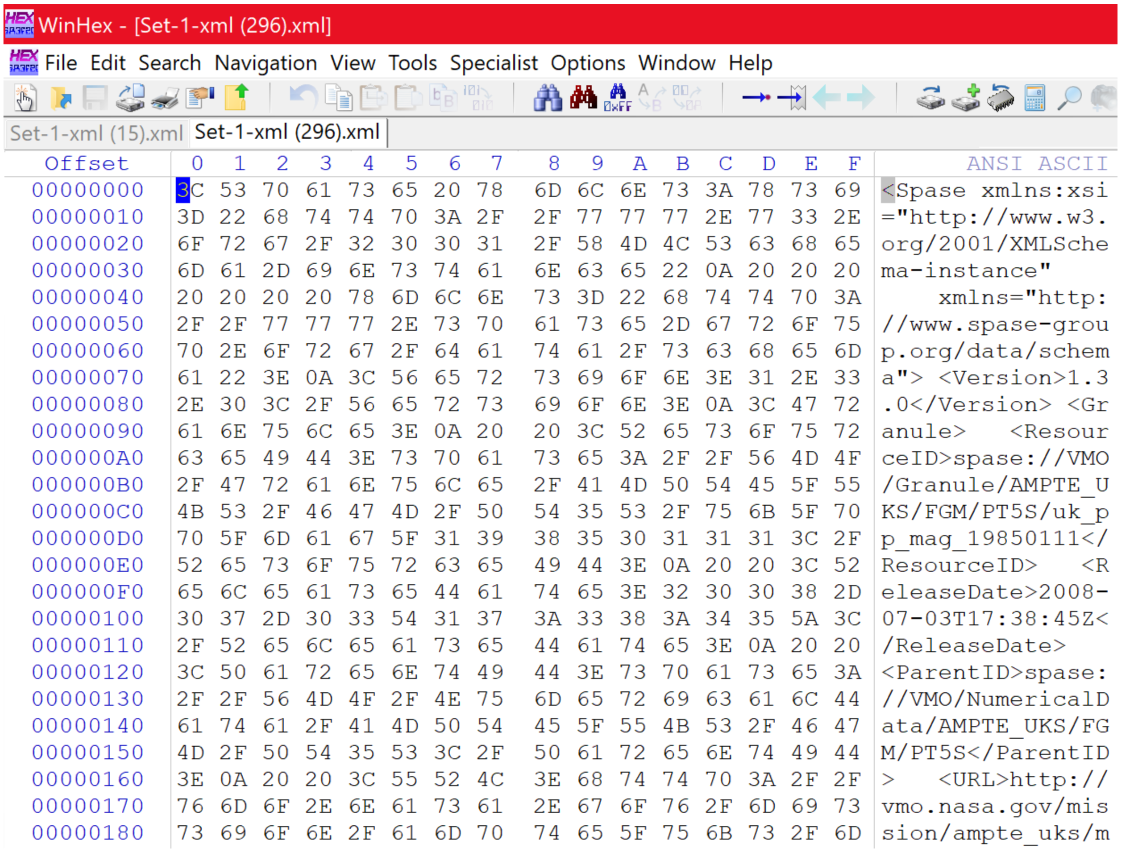


Fig. 7.10. File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Figure 7.11 shows a snippet of the Set-1-xml(15).xml file with regards to the Samsung NVMe SSD TRIM ON case using a USB WriteBlocker. The file under 693 bytes was opened in the Win Hex tool. The file was recovered and intact. Moreover, the file contents were not wiped, as shown in WinHex.

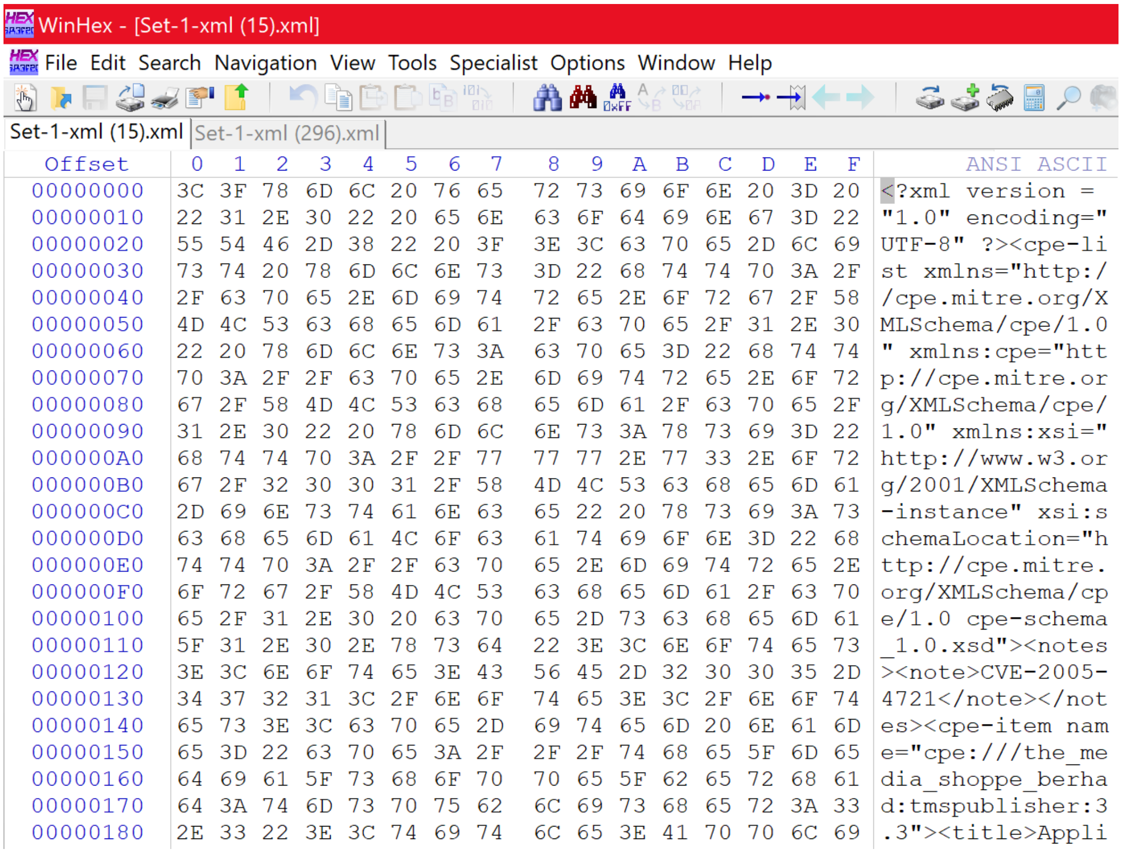


Fig. 7.11. File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Figure 7.12 shows a snippet of the Set-1-xml(296).xml file with regards to the Seagate NVMe SSD TRIM OFF case using a USB WriteBlocker. The file over 696 bytes in size was opened in the WinHex tool. As seen from the experimental results, the file was recovered, and the contents of the file were not wiped out.

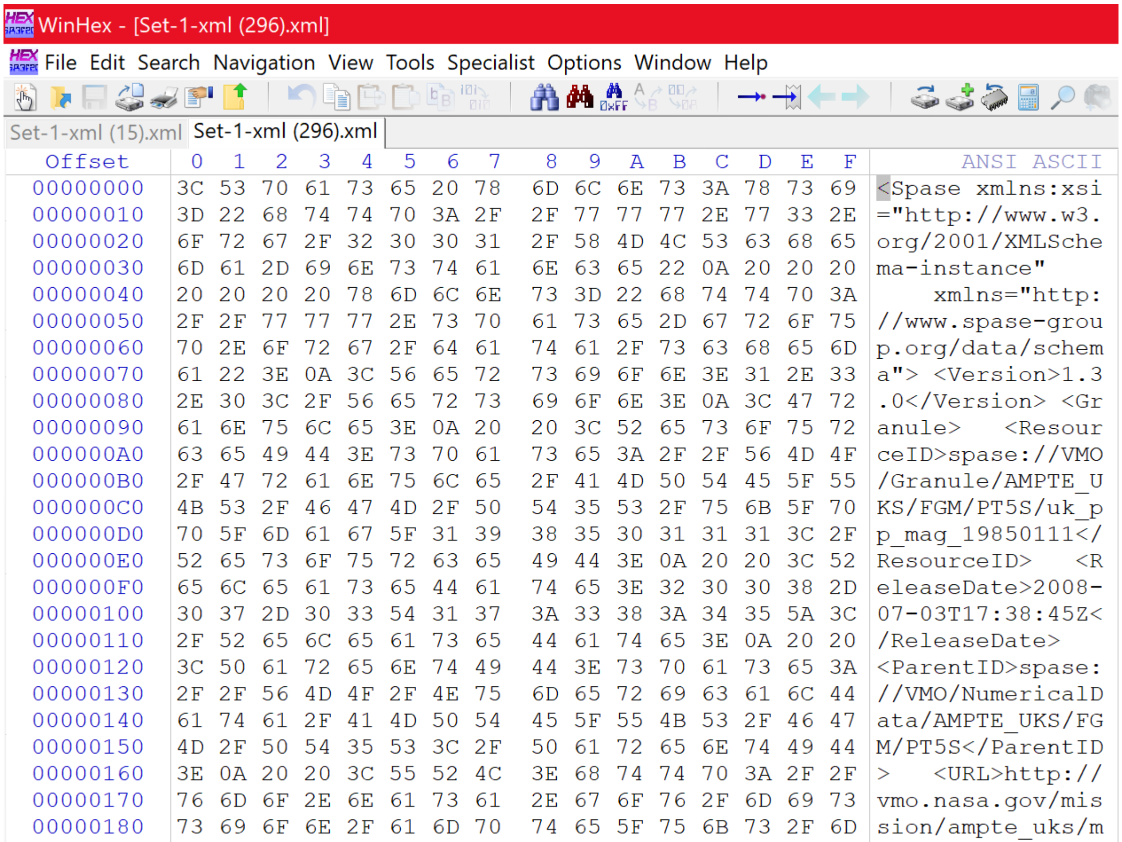


Fig. 7.12. File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Figure 7.13 shows a snippet of the Set-1-xml(15).xml file with regards to the Seagate NVMe SSD TRIM ON case using a USB WriteBlocker. The fil under 696 bytes was opened in the Win Hex tool. The file was recovered and intact. Moreover, the file contents were not wiped, as shown by the hexadecimal characters.

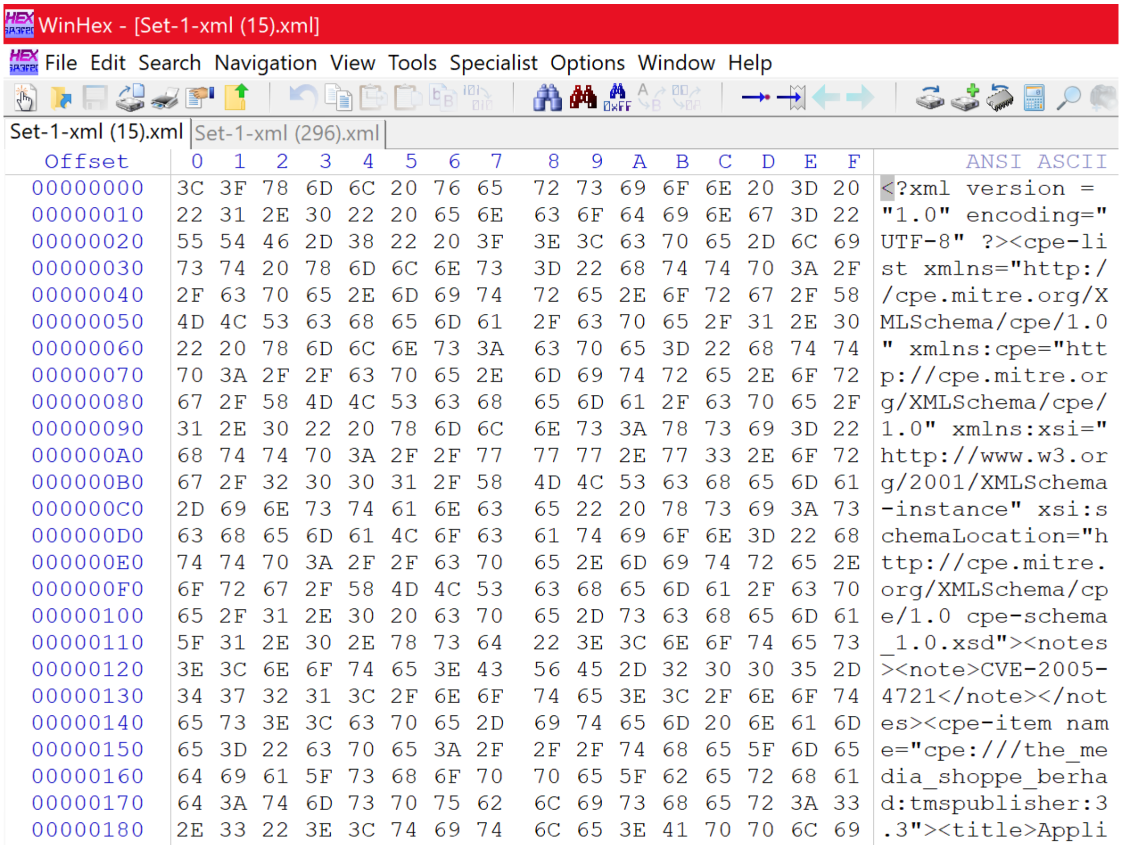


Fig. 7.13. File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Hash Analysis for Samsung and Seagate NVMe SSDs

In this section, we exhibited our findings via MD5 hash values of the files following the TRIM ON and OFF recovery operations. We used the QuickHash hashing tool to generate hash values.

Initially, the hash value of the original file is displayed, followed by TRIM ON and TRIM OFF MD5 hashes, and file size for Samsung NVMe SSD, shown in figure 7.14. Similarly, figure 7.15 shows the hash values of the original file, followed by TRIM ON and TRIM OFF MD5 hashes, and file size in the Seagate NVMe SSD case. These figures aim to validate and verify the claims made due to experimental observation.

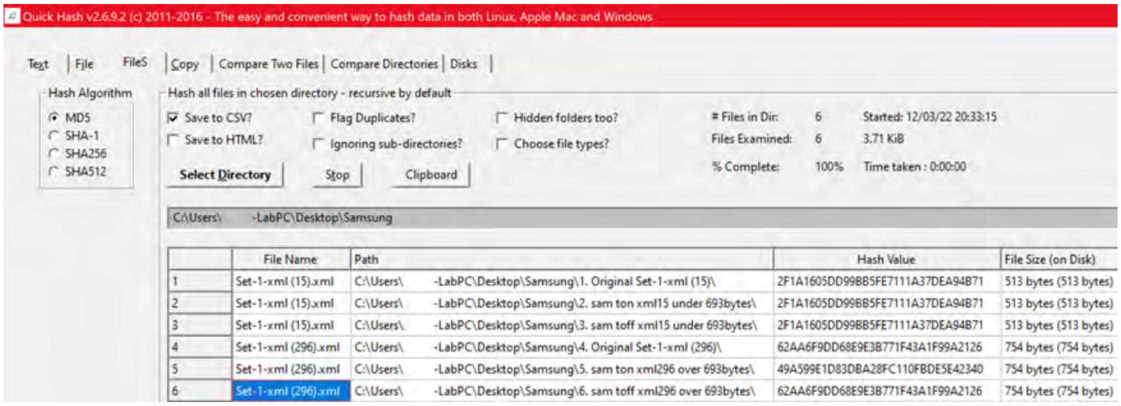


Fig. 7.14. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Samsung NVMe SSD.

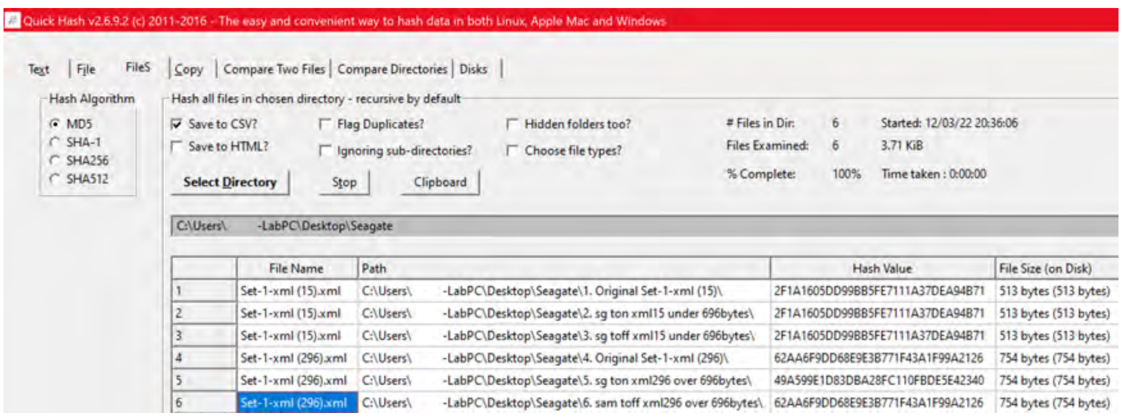


Fig. 7.15. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Seagate NVMe SSD.

Table 7.14 shows the names of all forensically acquired images, image type, size in kilobytes, MD5 and SHA1 hashes. The hash of all the image files constantly changed through the steps of the experiment. For example, in the case of TRIM OFF, the hash values of all the forensic images changed even though exact files were recovered. In the case of TRIM ON for Samsung and Seagate NVMe SSDs, the hash values of all the forensics images were different. Furthermore, file recovery was impossible when the file size was greater than 693 bytes in Samsung and 696 in Seagate, respectively.

Table 7.14. Digital forensics information about forensically acquired image files of Samsung and Seagate NVMe SSDs with USB WriteBlocker.

File Names	Image Type	Image Size (KB)	MD5 Hash	SHA1 Hash
Imaging TRIM ON Samsung NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-sam_nvme_usb_image_1	e01	475 381	0fcf46557dd96ca090736a8d83a810ab	0a1e689519d4187fe2d9557d058e5dc53c91e966
wwb-sam_nvme_usb_image_2	e01	475 380	634d99ae8749160b1d448f4e2711fceb	35db9a9bde217666553130ed7525a530633ddece
wwb-sam_nvme_usb_image_3	e01	475 375	86e9e536ab1287d8f924b5cdc46cb787	e763ad799563d00c4d564abecaab7eab32d75f77
wwb-sam_nvme_usb_image_4	e01	475 370	7acaa78d12c10e77a759faf02da1daba	262c45b09581762453a8da7876a8fce64e9c1401
Imaging TRIM ON Seagate NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-sg_nvme_usb_image_1	e01	486 133	b8adcbe881cc289d48588677e4d3e058	de89e807b89a298d6dc7837d7a3ae03547694cea
wwb-sg_nvme_usb_image_2	e01	486 130	3425bc4e3067e2d8260e417b986be443	3a9e6bfa0f95c9580394bb5e7e1f285f5fe19e9a
wwb-sg_nvme_usb_image_3	e01	486 127	cdd68873ec87d7fa4c238b91f03c19b9	e010d7b88f4de114d84675f173dc777bc9f87a40
wwb-sg_nvme_usb_image_4	e01	486 118	a9b9815c7100720b7a9a1612df6e3bf9	dceada2448129964a6a4ab6f6dc38060df493f23
Imaging TRIM OFF Samsung NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-sam_nvme_usb_image_1	e01	154 417 284	c5dfce8d2373ea10db669247d961b6fd	35e2b78047d03e7102d0534b30eff1ffc31c565e
wwb-sam_nvme_usb_image_2	e01	154 417 284	72401002a3282952a955baa0eb25c4fb	b34a92e79c733c7182f370fb610b77e72fded536
wwb-sam_nvme_usb_image_3	e01	154 417 282	617a56356cb1bec21429f5ffc497794d	58671d7860765abcd2a9dd43ed5e31ee3c486894
wwb-sam_nvme_usb_image_4	e01	154 417 281	8ddb9e6ca6836ca3a168729add43aba0	70fbc297993fd9454d2ff7413851949ef05e5600
Imaging TRIM OFF Seagate NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-sg_nvme_usb_image_1	e01	154 417 193	2a5aab29392eea665c945c728f29e51c	3240d787b6d3763a037694b07beeafe218a99742
wwb-sg_nvme_usb_image_2	e01	154 417 191	036baa93abc61cdf38137ce7e530e6b6	25bd325ba7d2278743ad787289dfa32fdde75e8a
wwb-sg_nvme_usb_image_3	e01	154 417 190	acff78b2ea6f39ec1fd622ae868a47af	c95987513a4747bae35d7910d3dff675a9ec173b
wwb-sg_nvme_usb_image_4	e01	154 417 189	251e3f565a640ace74b7edb77a216d70	69fdef2fb40ceaaa6abe77508149625819833d59

Western Digital and Silicon Power TRIM ON Analysis

The analysis of TRIM ON cases in Western Digital and Silicon Power shows a similar trend in file recovery procedures as seen in the Seagate NVMe SSD. The controller chip did not act on files under 696 bytes in Western Digital and Silicon Power storage devices. As a result, all files under 696 bytes were intact without any file content corruption. However, when the files' size was greater than 696 bytes in Western Digital and Silicon Power NVMe SSDs, the files' contents were cleared or zeroed out and hence rendered unusable. Tables 7.15, 7.16, 7.17, and 7.18 show the statistics of different files used and the files that were recovered.

Table 7.15. The number of files recovered from FTK in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Western Digital FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 693 bytes were intact after recovery in Western Digital NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 7.16. The number of files recovered from Autopsy in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Western Digital Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***

*: All files recovered but corrupted.
** : All files recovered but 8280 corrupted + 92 not corrupted.
*** : All files recovered but 69 corrupted + 46 not corrupted.
Note:
1) Files under 693 bytes were intact after recovery in Western Digital NVMe SSD.
2) For corrupted files, the size of the files was the same, but the contents were wiped out.

Table 7.17. The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Silicon Power FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13152	13152*	13152*	13152*	13152*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115*
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 696 bytes were intact after recovery in Silicon Power NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 7.18. The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM ON case.

Silicon Power Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13152	13152*	13152*	13152*	13152*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115*

*: All files recovered but corrupted.
** : All files recovered but 8280 corrupted + 92 not corrupted.
*** : All files recovered but 69 corrupted + 46 not corrupted.
Note:
1) Files under 696 bytes were intact after recovery in Silicon Power NVMe SSD.
2) For corrupted files, the size of the files was the same, but the contents were wiped out.

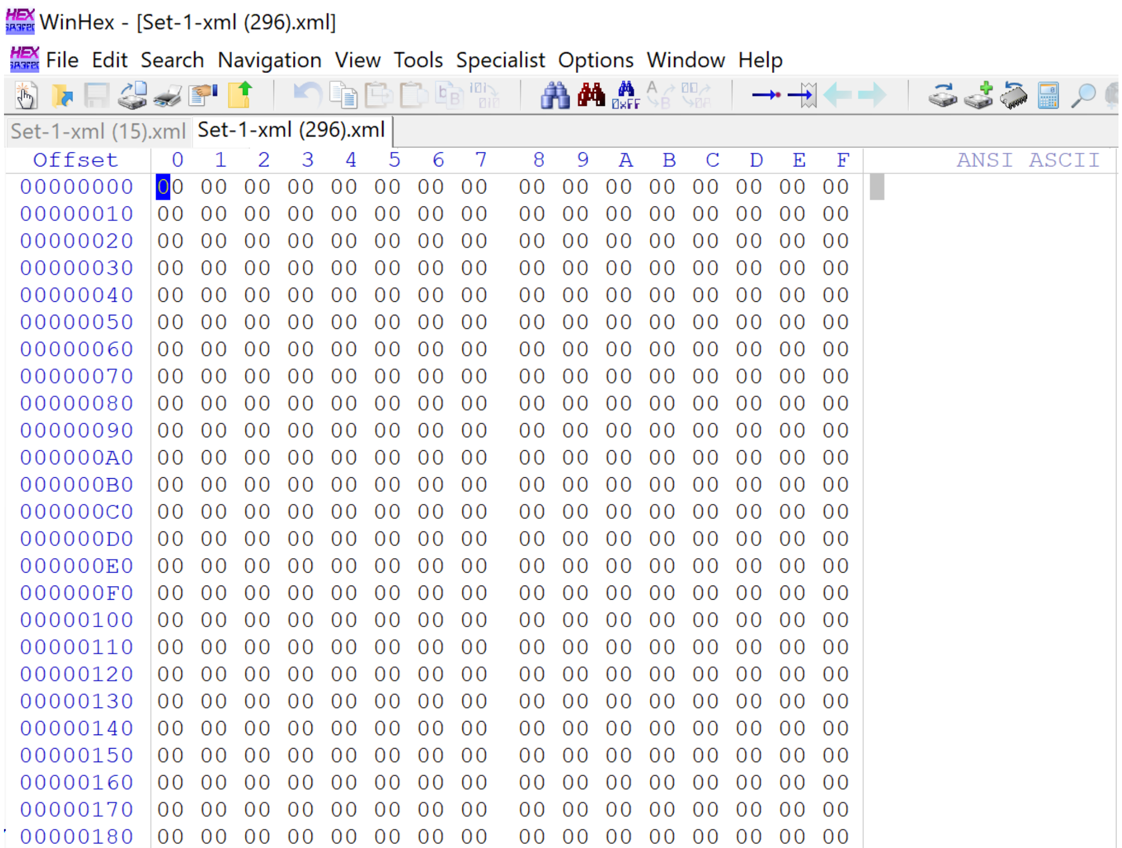


Fig. 7.16. File, Set-1-xml(296).xml, over 696 bytes in Western Digital NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.16 shows a snippet of an XML file with regards to the Western Digital NVMe SSD TRIM ON case using a USB WriteBlocker. The file over 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, and the contents of the file were wiped out.

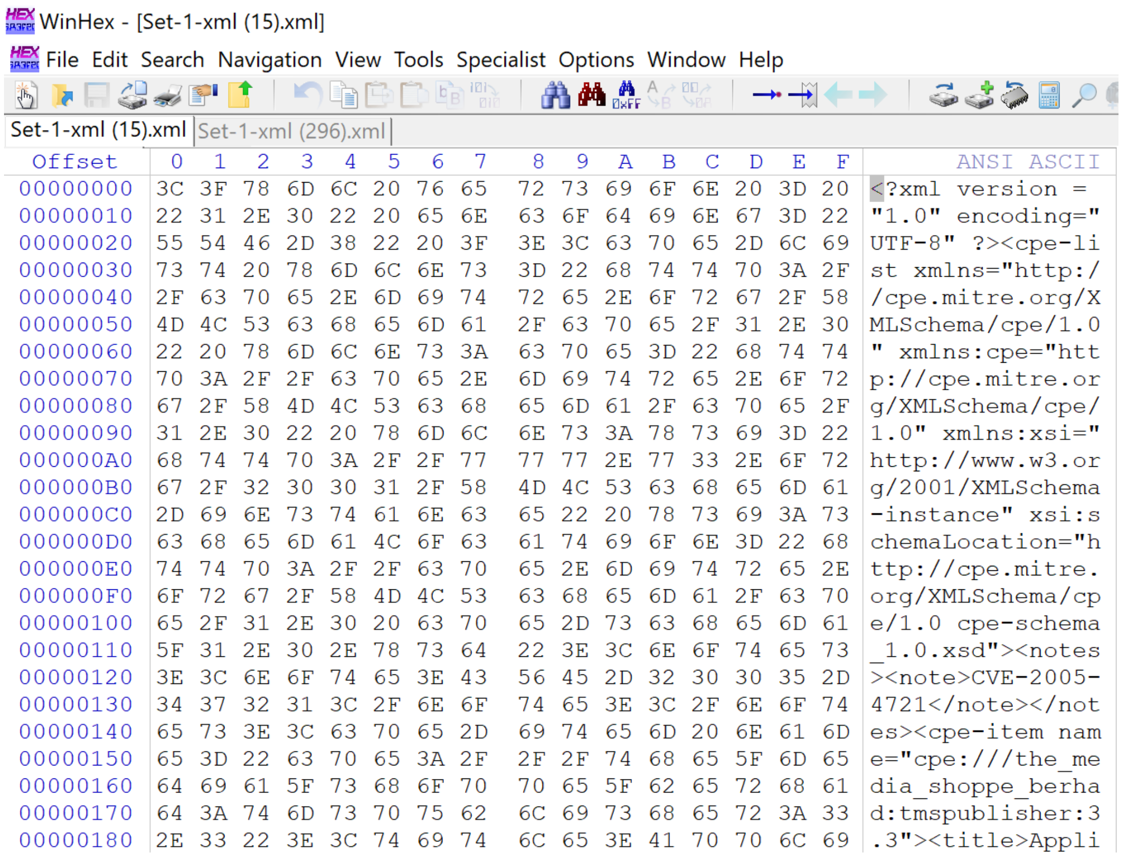


Fig. 7.17. File, Set-1-xml(296).xml, under 696 bytes in Western Digital NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.17 shows a snippet of an XML file with regards to the Western Digital NVMe SSD TRIM ON case using a USB WriteBlocker. The file under 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered and the contents of the file were not wiped.

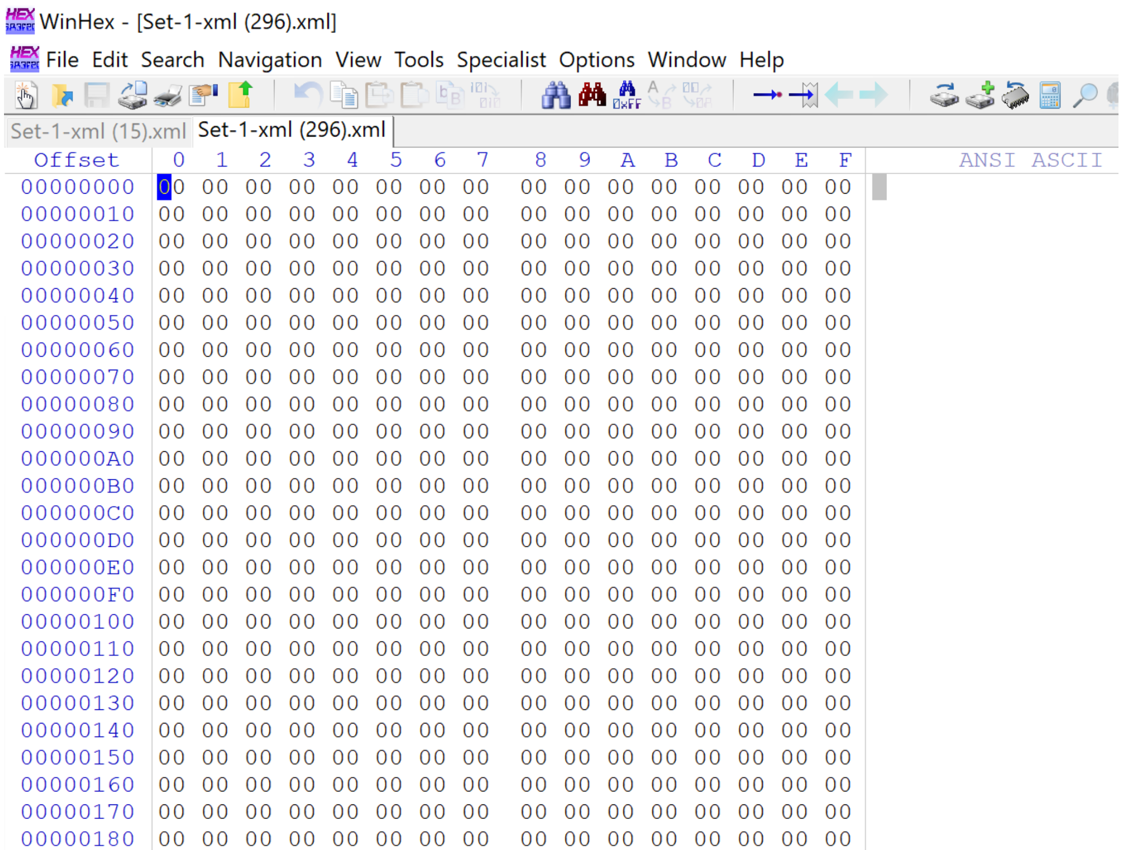


Fig. 7.18. File, Set-1-xml(296).xml, over 696 bytes in Silicon Power NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.18 shows a snippet of an XML file with regards to the Silicon Power NVMe SSD TRIM ON case using a USB WriteBlocker. The file over 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, and the contents of the file were wiped out.

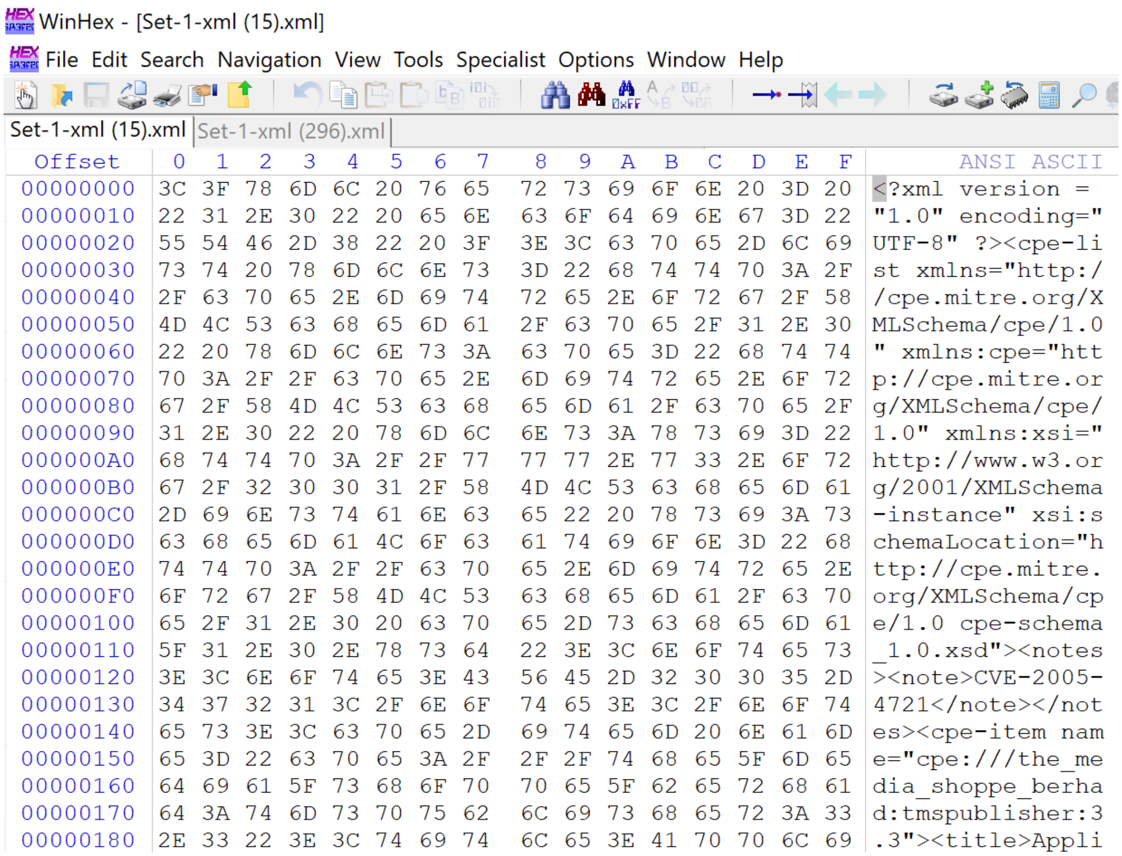


Fig. 7.19. File, Set-1-xml(296).xml, under 696 bytes in Silicon Power NVMe SSD TRIM ON case, with using a USB WriteBlocker.

Figure 7.19 shows a snippet of an XML file with regards to the Silicon Power NVMe SSD TRIM ON case using a USB WriteBlocker. The file under 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, and the contents of the file were not wiped out.

Western Digital and Silicon Power TRIM OFF Analysis

File recovery with TRIM OFF of four forensics Western Digital and Silicon Power NVMe SSD images, using AccessData FTK and Autopsy tools was successful. This happened because the TRIM OFF feature stops the operating system from informing the SSD to erase unusable data blocks. Hence, the NVMe SSD controller no longer oversees the available storage space to its full potential. Therefore, the controller chip did not clear out the pages so the contents of all the files were intact i.e., files could be viewed, opened, and worked on consistently. Furthermore, there was no instance of file corruption in the case. Tables 7.19, 7.20, 7.21, and 7.22 show the statistics of different files used and the files that were recovered.

Table 7.19. The number of files recovered from FTK in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Western Digital FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13152	13152	13152	13152	13152
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	150	150	150	150	150
All files recovered and intact in TRIM OFF case of Western Digital NVMe SSD					

Table 7.20. The number of files recovered from Autopsy in Western Digital NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Western Digital Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13152	13152	13152	13152	13152
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	150	150	150	150	150
All files recovered and intact in TRIM OFF case of Western Digital NVMe SSD					

Table 7.21. The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Silicon Power FTK Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13152	13152	13152	13152	13152
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	150	150	150	150	150
All files recovered and intact in TRIM OFF case of Silicon Power NVMe SSD					

Table 7.22. The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter in Windows 10 TRIM OFF case.

Silicon Power Autopsy Case Statistics in Windows 10 with WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13152	13152	13152	13152	13152
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	150	150	150	150	150
All files recovered and intact in TRIM OFF case of Silicon Power NVMe SSD					

Figure 7.20 shows a snippet of an XML file with regards to the Western Digital NVMe SSD TRIM OFF case using a USB WriteBlocker. The file over 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered and the contents of the file were not wiped out.

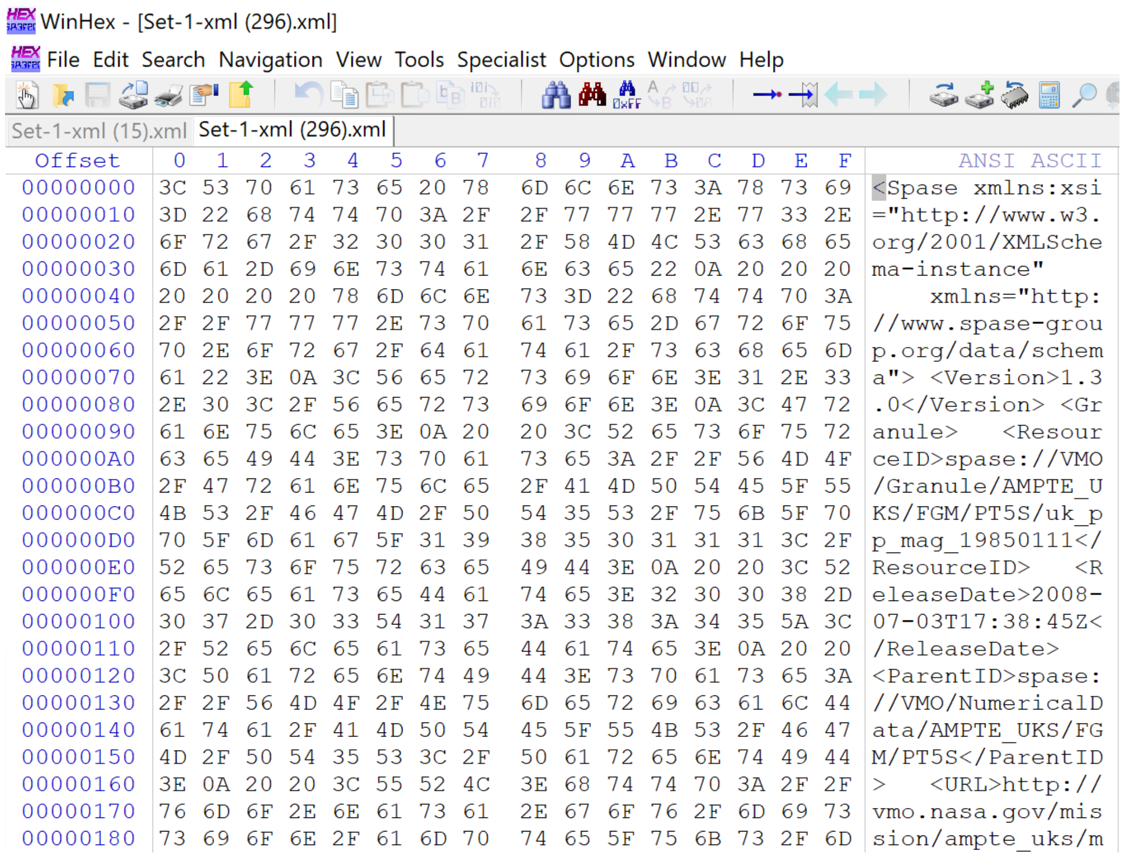


Fig. 7.20. File, Set-1-xml(296).xml, over 696 bytes in Western Digital NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Figure 7.21 shows a snippet of an XML file with regards to the Western Digital NVMe SSD TRIM OFF case using a USB WriteBlocker. The file under 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered and the contents of the file were not wiped out.

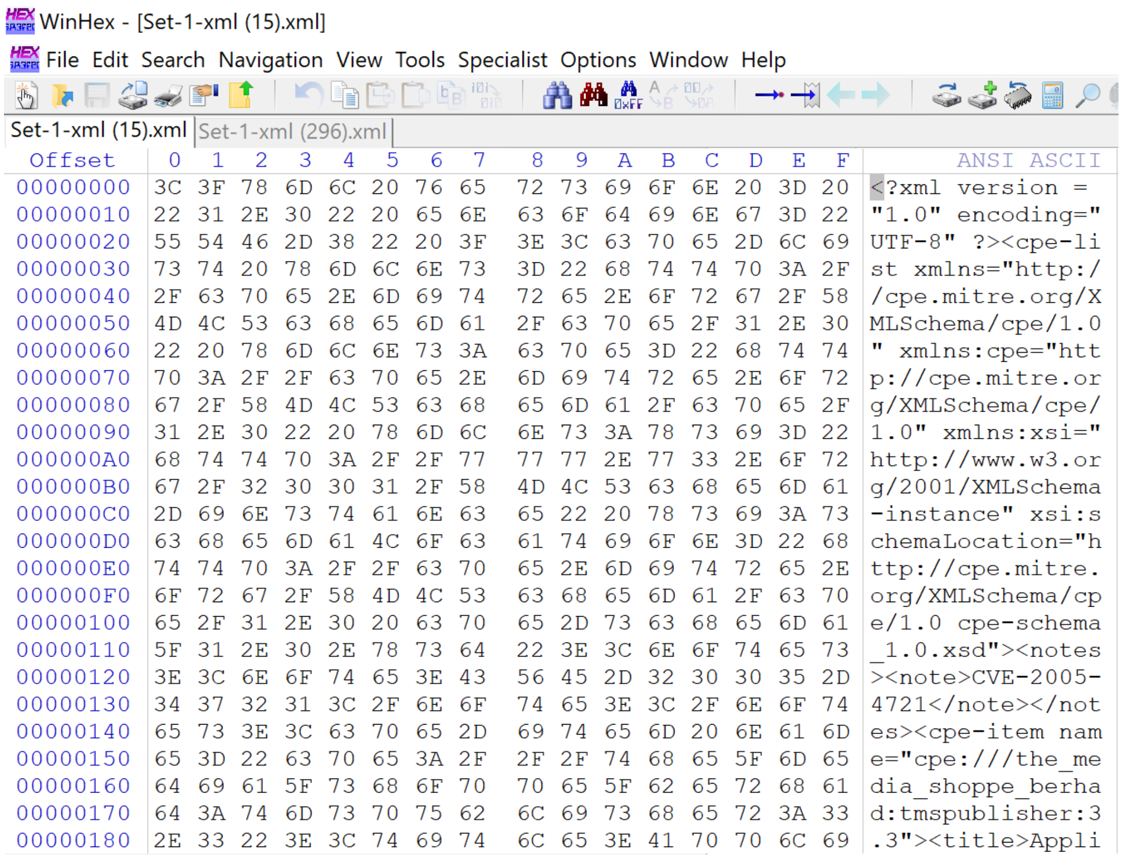


Fig. 7.21. File, Set-1-xml(296).xml, under 696 bytes in Western Digital NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Figure 7.22 shows a snippet of an XML file with regards to the Silicon Power NVMe SSD TRIM OFF case using a USB WriteBlocker. The file over 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered and the contents of the file were not wiped out.

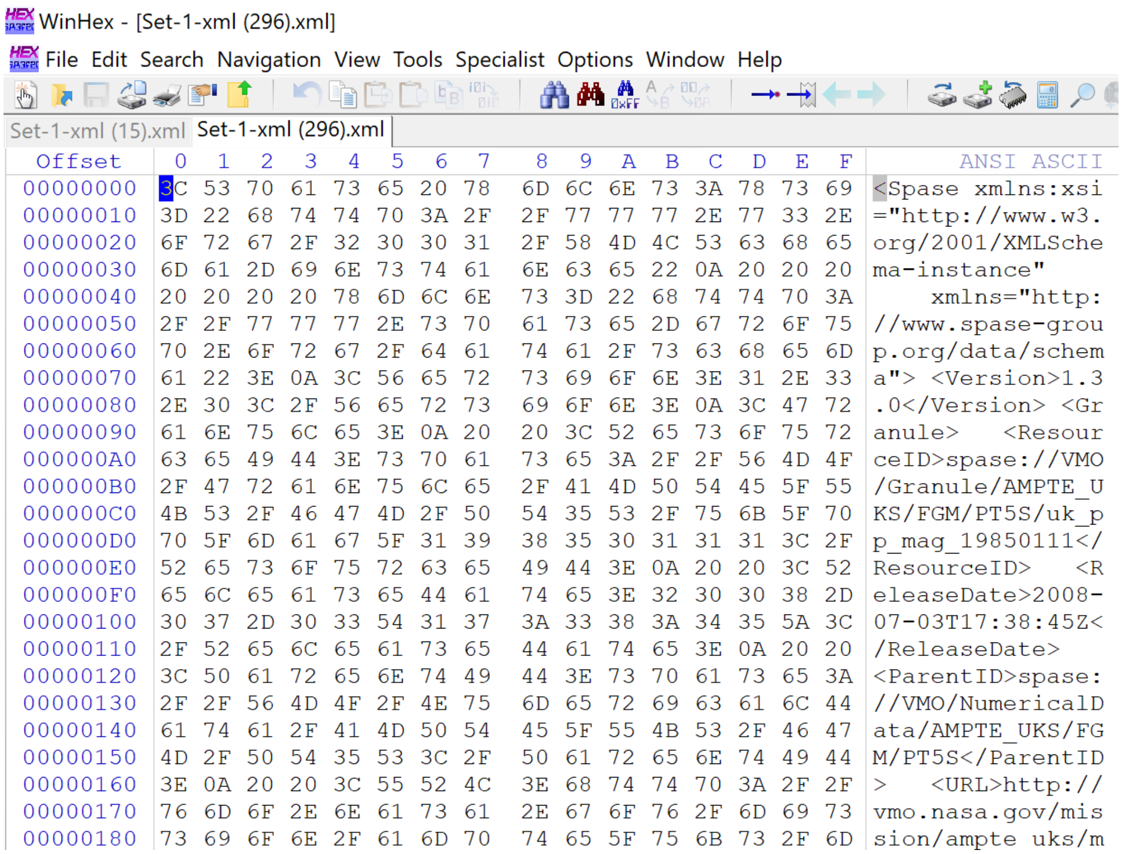


Fig. 7.22. File, Set-1-xml(296).xml, over 696 bytes in Silicon Power NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Figure 7.23 shows a snippet of an XML file with regards to the Silicon Power NVMe SSD TRIM OFF case using a USB WriteBlocker. The file under 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered and the contents of the file were not wiped out.

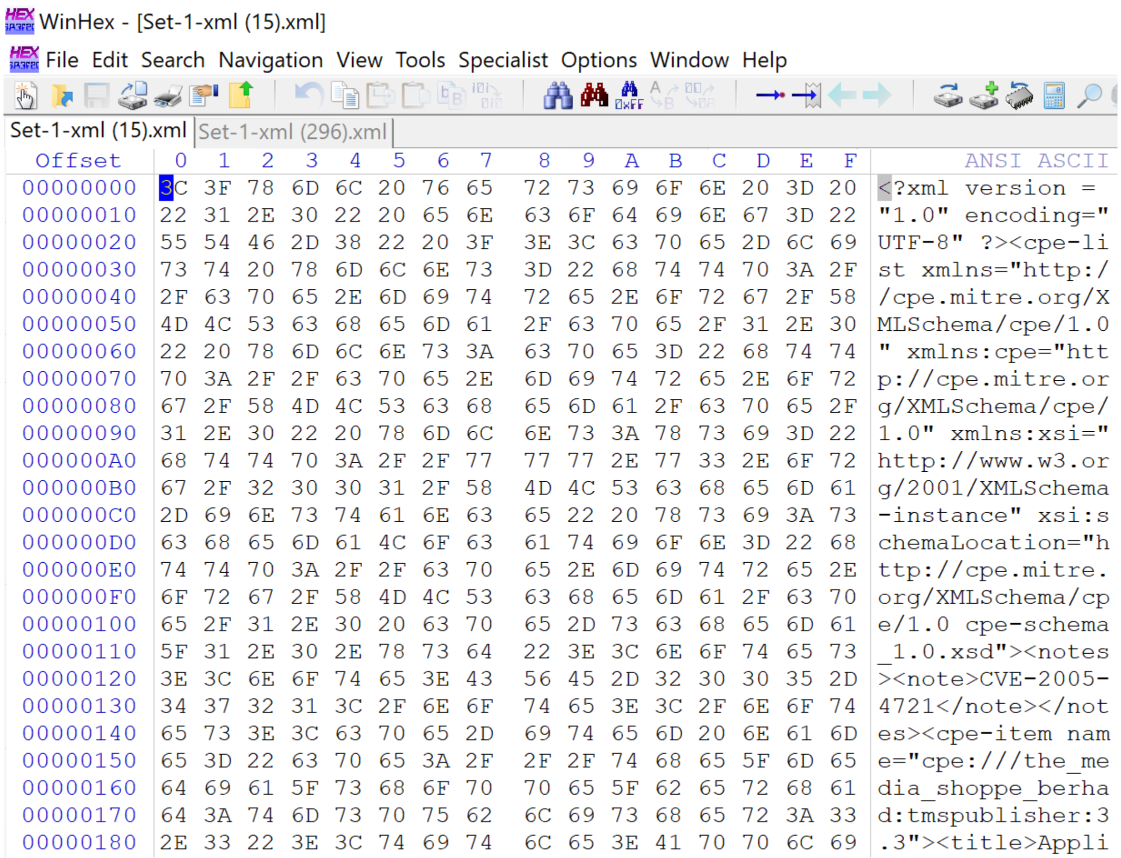


Fig. 7.23. File, Set-1-xml(296).xml, under 696 bytes in Silicon Power NVMe SSD TRIM OFF case, with using a USB WriteBlocker.

Hash Analysis for Western Digital and Silicon Power NVMe SSDs

In this section, we presented our findings via MD5 hash values of the files following the TRIM ON and OFF recovery operations. We used the QuickHash hashing tool to generate the hash values.

The hash value of the original file is displayed, followed by TRIM ON and TRIM OFF MD5 hashes, and file size for Western Digital NVMe SSD, as shown in figure 7.24. Similarly, Figure 7.25 shows the hash values of the original file, followed by TRIM ON and TRIM OFF MD5 hashes, and file size in the Silicon Power NVMe SSD case. The figures aim to validate and verify the claims which were made due to experimental observation.

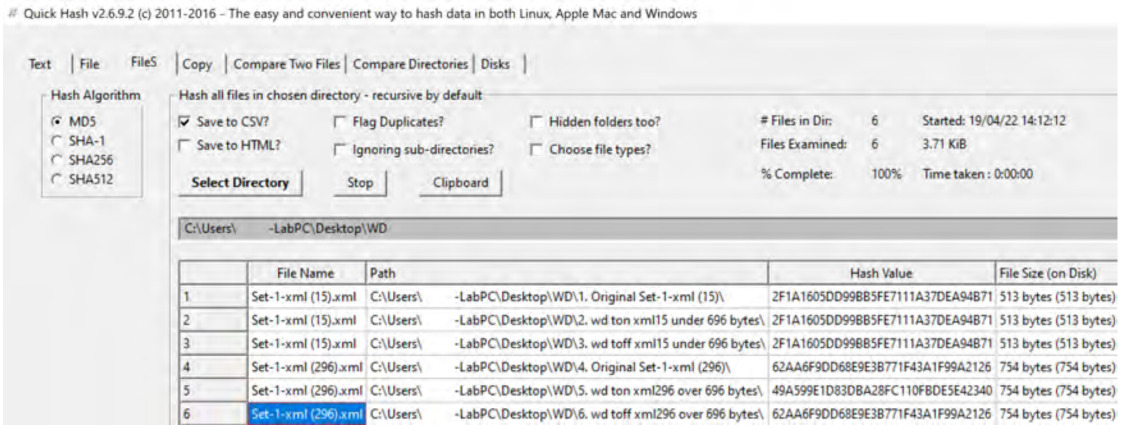


Fig. 7.24. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Western Digital NVMe SSD.

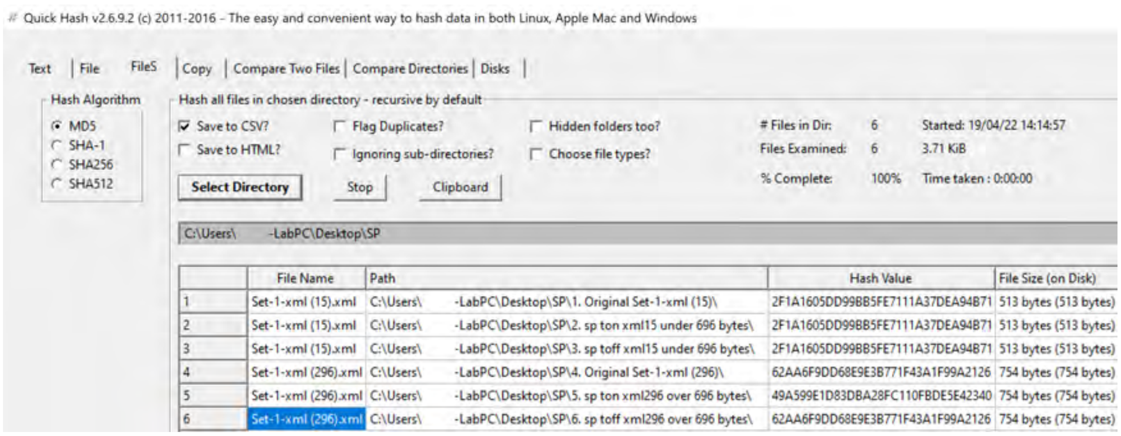


Fig. 7.25. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Silicon Power NVMe SSD.

Table 7.23 contains the names of all forensically obtained images, as well as their types, sizes in kilobytes, and MD5 and SHA1 hashes. Throughout the experiment, the hash of all the image files changed regularly. In the case of TRIM OFF, for example, the hash values of all the forensic images were altered even if exact files were recovered. However, for the TRIM ON case for WD and SP NVMe SSDs, the hash values of all forensics images were different, and file recovery was not feasible when the file size exceeded 696 bytes.

Table 7.23. Digital forensics information about forensically acquired image files of Western Digital and Silicon Power NVMe SSDs with USB WriteBlocker.

File Names	Image Type	Image Size (KB)	MD5 Hash	SHA1 Hash
Imaging TRIM ON Western Digital NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-wd_nvme_usb_image_1	e01	475 384	9747ef574e5ea691286f92d6ed6f0b1b	893cec583ef5a2720fa7251175934e94de45fc16
wwb-wd_nvme_usb_image_2	e01	475 378	12c96ff6c39731e85b56d8ed07470e49	3a8d2511f168e197bfcab109d848bd70d837a622
wwb-wd_nvme_usb_image_3	e01	475 378	bfc80533e9af9c30129b12038f2e62de	7eb3edbf2a180518dfc03196921eb58e36e8afd1
wwb-wd_nvme_usb_image_4	e01	475 373	60fad6107bc9e02f7ca549eaef3acda0	23778f71989835db54979c50f11589221e22f5e6
Imaging TRIM ON Silicon Power NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-sp_nvme_usb_image_1	e01	470 826	6401fd22ee10bfe1dd576d29bf1f71d6	d4a39202c36897cce7c9df7cfac90486dd310db9
wwb-sp_nvme_usb_image_2	e01	470 825	f15f747abf8e905beb3598981befb61f	1b359933b070df95eb609cc1be5131140783a6bd
wwb-sp_nvme_usb_image_3	e01	470 825	22f26a6baafb98a6f0c34a3d898a89b5	b8658f796451adfdc826d548699da1bba3479212
wwb-sp_nvme_usb_image_4	e01	470 825	8c140fb4b880631c38b0006809d33bc9	64b6e25581451316d799f27a3a217f3e036a0dce
Imaging TRIM OFF Western Digital NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-wd_nvme_usb_image_1	e01	154 417 179	1157dda1e4ea07f5014361ab09bd14cc	d33654f553c58ff2170c3107f133dd6849e0c437
wwb-wd_nvme_usb_image_2	e01	154 417 176	e4c7a7369b2180322f5b11118ec93ea2	49bef3ab38d9434a1e5f31a0dde935b0ae2cb3a1
wwb-wd_nvme_usb_image_3	e01	154 417 175	6201a0dee3e92e1fb7df09353c6f7b19	afa9fe58f2abab8aa4f9ce2c38b909eeaa7b92eb
wwb-wd_nvme_usb_image_4	e01	154 417 179	418a0d5725c6c79cc48a1ccd99722c28	b3b56d65c2bd90cd70a512ab39953137be9ecb5a
Imaging TRIM OFF Silicon Power NVMe SSD with USB WriteBlocker using FTK Imager				
wwb-sp_nvme_usb_image_1	e01	160 276 543	8b397574e606ea5bb405b000dca33203	1eb2f504e52362c654b0052efa8c01a3ad6fd008
wwb-sp_nvme_usb_image_2	e01	160 276 540	3ab94d65dc06be9bc81f2f21209607bd	20e2f2cf3af64e2a429ca7d54f9f39b1e71ebad9
wwb-sp_nvme_usb_image_3	e01	160 276 539	d90952fcb67c4b9190fc6c48ad7627db	3522e70fbf715ccef8d7f0d2e91c8c4fc1249b27
wwb-sp_nvme_usb_image_4	e01	160 276 539	89ea36f99bb3e8b90afe1031e508153d	82b239130ad0dcbc48f95b373c03af1356a63278

CHAPTER VIII

Digital Forensics in USB NVMe SSDs without WriteBlocker

This chapter is the continuation of digital forensics analysis in USB NVMe SSDs. We enhanced our research method and modified our approach to investigate the behavior of the four NVMe SSDs enclosed in USB adapters when no write blocker is used. This chapter aimed to find the number of files recovered after they were deleted from four NVMe storage devices connected to computer systems. However in this case, the forensics images are taken without using a USB WriteBlocker.

Similar to the work done in Chapter VII, we installed Windows 10 operating system on four NVMe SSDs. We used AccessData FTK [77], Autopsy, and WinHex [78] tools to recover and conduct forensics examination. Lastly, we explained the forensics observation based on the findings with varying controller chips of the four NVMe SSD devices.

Experimental Setup without USB WriteBlocker

Table 8.1 below shows the technical specifications of the equipment we have used for the experiment in this chapter.

Table 8.1. Equipment used in the experiment.

Tools	Name
NVMe SSD 1	Samsung V-NAND SSD 970 Evo Plus
NVMe SSD 2	Seagate Barracuda 510 250GB NVMe SSD
NVMe SSD 3	Western Digital SN550 250GB NVMe SSD
NVMe SSD 4	Silicon Power 3D-NAND NVMe SSD
Operating System	Windows 10 Pro v21H2
Forensic Analysis Tool	AccessData FTK 7.5 and WinHex
Forensics Acquisition Tool	AccessData FTK Imager 4.7
Workstation	CPU: Intel Xeon W-2123 — RAM : 80GB

Fig. 8.1. Samsung NVMe SSD attached without USB WriteBlocker.

Fig. 8.2. Seagate NVMe SSD attached without USB WriteBlocker.

Fig. 8.3. Western Digital NVMe SSD attached without USB WriteBlocker.

Fig. 8.4. Silicon Power NVMe SSD attached without USB WriteBlocker.

Specifics of SSDs

The test included four different NVMe SSD brands: Samsung, Seagate, Western Digital, and Silicon Power. These devices were picked due to their substantial market share and dependability. The four manufacturers and models used in the experiment were chosen to reflect a real-world scenario as the specifications of the SSDs used in the experiment closely resemble those of a common SSD that a regular user may own. Furthermore, because these are the most common characteristics of SSDs incorporated in a laptop or desktop computer, the choice of SSDs makes the experiment more meaningful to the digital forensic community. The name, model, product number (P/N), storage capacity, number of flash chips, kind of NVMe flash chip, and controller information of the NVMe SSDs used are all listed in the tables 8.2 and 8.3.

Table 8.2. Information of Samsung and Seagate NVMe SSDs used in the experiment.

SSD Information	Samsung NVMe Specification 1.3
Name	Samsung NVMe V-NAND SSD 970 Evo Plus NVMe M.2
Model	MZ-V7S250
P/N	MZVLB250HBHQ
Storage Capacity	250 GB
Number of flash chips inside	2
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Samsung S4LR020 — 2117 ARM — Pheonix

SSD Information	Seagate NVMe Specification 1.3
Name	Seagate Barracuda 510 250GB NVMe SSD
Model	ZP250CM30001
P/N	2NS312-300
Storage Capacity	250 GB
Number of flash chips inside	4
Type of NVMe NAND Flash	3D TLC NAND
Controller information	SKHynix - H5AN4G6NBJR

Table 8.3. Information of Western Digital and Silicon Power NVMe SSDs used in the experiment.

SSD Information	WD NVMe Specification 1.4
Name	Western Digital SN550 250GB NVMe SSD
Model	WDS250G2B0C-00PXH0/21146P801302
P/N	87161901478830731375399388282263
Storage Capacity	250 GB
Number of flash chips inside	4
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Sandisk 20-82-10023-A1 — 1015ZKLY0KN

SSD Information	Silicon Power NVMe Specification 1.3
Name	Silicon Power 3D-NAND NVMe SSD
Model	A-60
P/N	SP256GBP34A60M28
Storage Capacity	256 GB
Number of flash chips inside	2
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Phison PS5013-E13-31—C02102E— TB5V79/ 001BB

Methodology and Experiment Initiation

The protocols and setups we followed and assigned during the experiment are listed and explained in this section.

1. The partition scheme used for the NVMe SSDs inside the USB enclosure adapters: **MBR (Master Boot Record)**
2. The number of partitions in each NVMe SSD: **1**

3. The file system of the one partition: **NTFS**
4. Prior to copying the files to the devices from Digital Corpora [80], we checked the **TRIM** status in Windows 10 by issuing the following command through the command prompt.

fsutil behavior query DisableDeleteNotify
**If the output is 1, then TRIM is disabled. If the output is 0, then TRIM is enabled.*
To enable TRIM: fsutil behavior set DisableDeleteNotify 0
To disable TRIM: fsutil behavior set DisableDeleteNotify 1

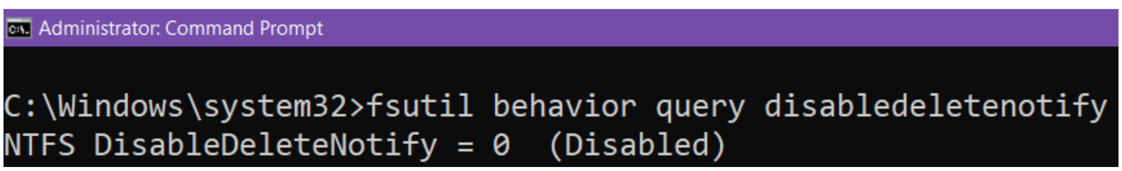


Fig. 8.5. The status of TRIM in Windows 10 using fsutil command.

Case scenario: TRIM ON from Windows 10 operating system without WriteBlocker

1. We copied the commonly used file types from the Digital Corpora dataset [80] to the four NVMe SSDs. We used large file sizes to exhaust the storage drives' capacity.
2. We then kept the files for one day with no user activity by keeping the drive attached to the USB port of the computer system.
3. Next, we deleted (shift+delete) the files from the devices and waited for one day before acquiring four forensic images of the four NVMe SSDs respectively.
   (a) We took four forensic images: three consecutive images with one day gap and last image after a span of four days from the third acquisition.

4. We analyzed the images in AccessData FTK and Autopsy for the NVMe storage devices.
5. We performed file recovery of the deleted files from the forensics images in the TRIM ON case.
6. Based on our results from the file recovery and WinHex analysis we documented the effects of wear-leveling.

Case scenario: TRIM OFF from Windows 10 operating system without WriteBlocker

1. Firstly, we disabled TRIM using Windows 10 command prompt before copying the files.
2. We copied the commonly used file types from the Digital Corpora dataset [80] to the four NVMe SSDs. We used large file sizes to exhaust the storage drives' capacity just like we talked about in chapter 6.
3. We then kept the files for one day with no user activity by keeping the drive attached to the USB port of the computer system.
4. Next, we deleted (shift+delete) the files from the devices and waited for one day before acquiring four forensic images of the four NVMe SSDs respectively.
   (a) We took four forensic images: three consecutive images with one day gap and last image after a span of four days from the third acquisition.
5. We analyzed the images in AccessData FTK and Autopsy for the NVMe storage devices.
6. We performed file recovery of the deleted files from the forensics images in the TRIM OFF case.
7. Like the TRIM ON case, based on our results from the file recovery and WinHex analysis, we documented the effects of wear-leveling.

Experiment Results, Analysis, and Discussion

The results of the file recovery utilizing the AccessData FTK and Autopsy tools are presented in this section. We began by populating the NVMe SSDs with the most frequently used files from the Digital Corpora dataset [80]. We then used the forensically acquired images of the four NVMe SSDs to undertake the file recovery operation. Tables 8.4 and 8.5, respectively, present the forensic image acquisition timeline information in both TRIM ON and TRIM OFF scenarios of Samsung, Seagate, Western Digital (WD), and Silicon Power (SP) NVMe SSDs.

Table 8.4. Timeline information of forensic file acquisition.

TRIM ON information without WriteBlocker			
Samsung NVMe	**Time**	**Seagate NVMe**	**Time**
Copy file date	11:20 pm 8/19/21	Copy file date	11:47 pm 8/22/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	11:20 pm 8/20/21	Delete files	11:47 pm 8/23/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	11:20 pm 8/21/21	1st image	11:47 pm 8/24/21
2nd image	11:20 pm 8/22/21	2nd image	11:47 pm 8/25/21
3rd image	11:20 pm 8/23/21	3rd image	11:47 pm 8/26/21
4th image	11:20 pm 8/27/21	4th image	11:47 pm 8/30/21
TRIM OFF information without WriteBlocker			
Samsung NVMe	**Time**	**Seagate NVMe**	**Time**
Copy file date	8:17 pm 9/14/21	Copy file date	11:45 pm 9/14/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	8:17 pm 9/15/21	Delete files	11:45 pm 9/15/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	8:17 pm 9/16/21	1st image	11:45 pm 9/16/21
2nd image	8:17 pm 9/17/21	2nd image	11:45 pm 9/17/21
3rd image	8:17 pm 9/18/21	3rd image	11:45 pm 9/18/21
4th image	8:17 pm 9/22/21	4th image	11:45 pm 9/22/21

Table 8.5. Timeline information of forensic file acquisition.

TRIM ON information without WriteBlocker			
WD NVMe	**Time**	**SP NVMe**	**Time**
Copy file date	12:15 pm 8/20/21	Copy file date	9:40 pm 8/22/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	12:15 pm 8/21/21	Delete files	9:40 pm 8/23/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	12:15 pm 8/22/21	1st image	9:40 pm 8/24/21
2nd image	12:15 pm 8/23/21	2nd image	9:40 pm 8/25/21
3rd image	12:15 pm 8/24/21	3rd image	9:40 pm 8/26/21
4th image	12:15 pm 8/28/21	4th image	9:40 pm 8/30/21
TRIM OFF information without WriteBlocker			
WD NVMe	**Time**	**SP NVMe**	**Time**
Copy file date	10:04 pm 9/14/21	Copy file date	6:10 pm 9/15/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	10:04 pm 9/15/21	Delete files	6:10 pm 9/16/21
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	10:04 pm 9/16/21	1st image	6:10 pm 9/17/21
2nd image	10:04 pm 9/17/21	2nd image	6:10 pm 9/18/21
3rd image	10:04 pm 9/18/21	3rd image	6:10 pm 9/19/21
4th image	10:04 pm 9/22/21	4th image	6:10 pm 9/23/21

Samsung and Seagate TRIM ON Analysis without WriteBlocker

The TRIM command allows the operating system to tell the SSD that specific sections are no longer needed. As a result, the SSD controller can now undertake many of the processes required to clear data well ahead of any request from the operating system. These internal procedures could even be carried out when the SSD is under low load, hiding or masking the activity from the user.

Despite this, the TRIM ON analysis on both the Samsung NVMe and Seagate NVMe SSDs' forensics images showed that all files were recovered using the AccessData FTK and Autopsy tools. However, the controller chip did not act on files under 693 bytes in Samsung NVMe SSD and 696 bytes in Seagate NVMe

SSD, respectively. As a result, they were all intact without any content wiped or corrupted. In addition, files greater than 693 bytes in Samsung NVMe, and 696 bytes in Seagate NVMe SSD were all corrupted, i.e., their file contents were all zeroed out and hence were rendered unusable. Tables 8.6, 8.7, 8.8, and 8.9 give the statistics of the different files used from the Digital Corpora dataset and the files recovered from Samsung and Seagate NVMe SSDs in TRIM on case.

Table 8.6. The number of files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

Samsung FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3*	3*	3*	3*
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 693 bytes were intact after recovery in Samsung NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 8.7. The number of files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

Samsung Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3*	3*	3*	3*
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. Note: 1) Files under 693 bytes were intact after recovery in Samsung NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 8.8. The number of files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

Seagate FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3^{+}	3^{+}	3^{+}	3^{+}

*: All files recovered but corrupted.
** : All files recovered but 8280 corrupted + 92 not corrupted.
*** : All files recovered but 69 corrupted + 46 not corrupted.
+: Recovered all but one file of out the three was corrupted.
Note:
1) Files under 696 bytes were intact after recovery in Seagate NVMe SSD.
2) For corrupted files, the size of the files was the same, but the contents were wiped out.

Table 8.9. The number of files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

Seagate Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3^+	3^+	3^+	3^+
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. +: Recovered all but one file of out the three was corrupted. Note: 1) Files under 696 bytes were intact after recovery in Seagate NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

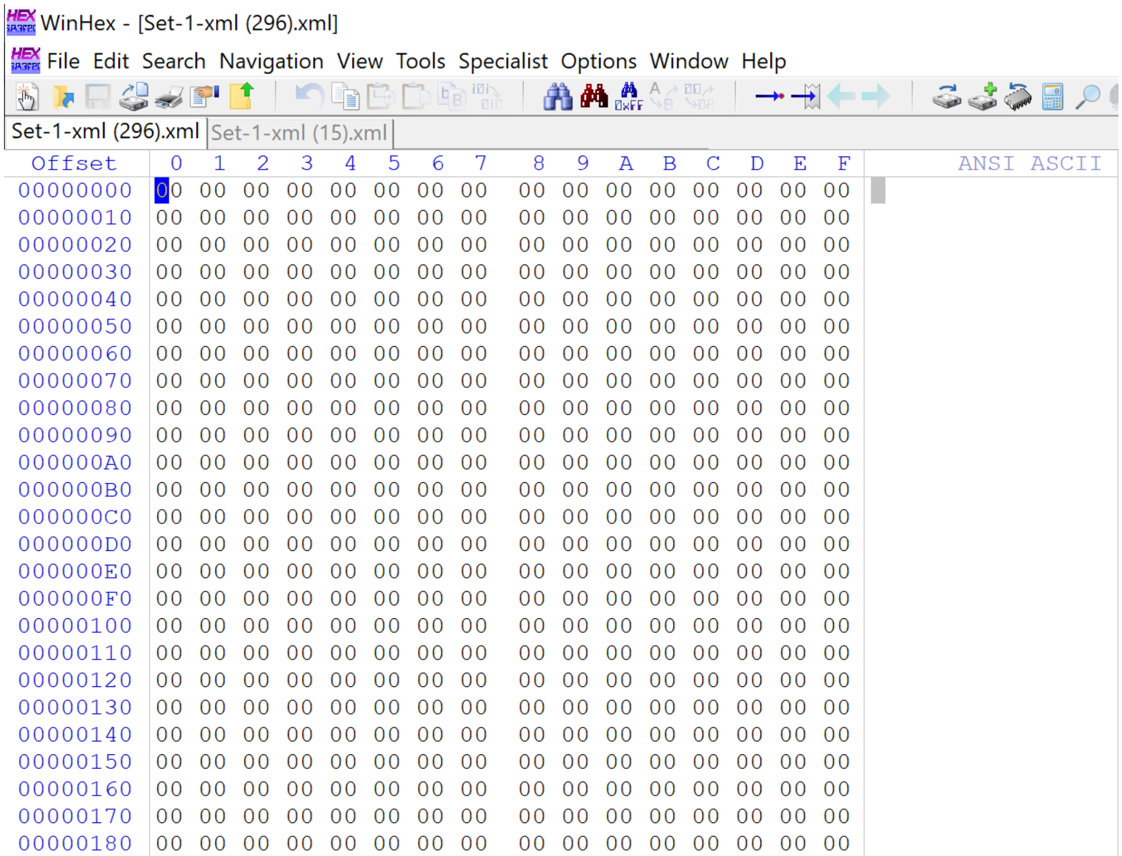


Fig. 8.6. File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM ON case, without using a USB WriteBlocker.

Figure 8.6 shows a snippet of an XML file with regards to the Samsung NVMe SSD TRIM ON case without using a USB WriteBlocker. The file over 693 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, but the contents of the file were corrupted, making the file unusable, as shown by the zeroes.

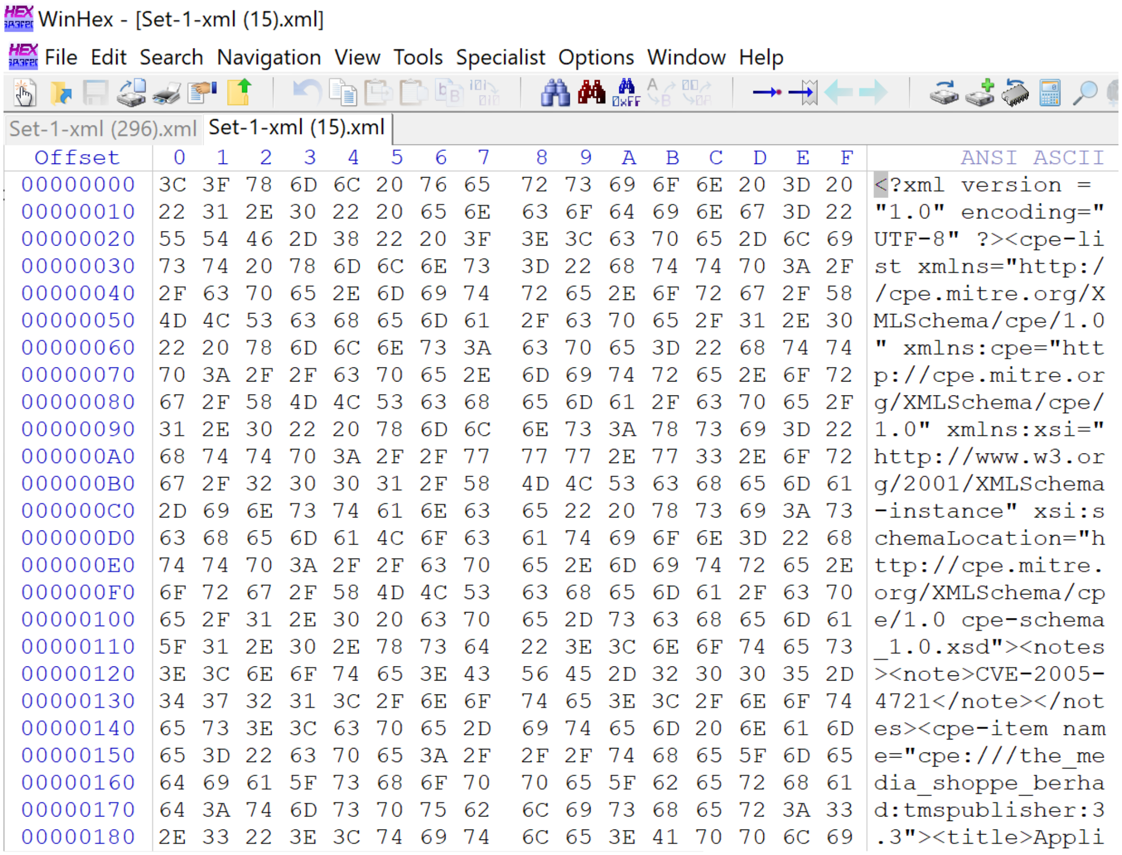


Fig. 8.7. File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM ON case, without using a USB WriteBlocker.

Figure 8.7 shows a snippet of the Set-1-xml(15).xml file with regards to the Samsung NVMe SSD TRIM ON case without using a USB WriteBlocker. The file under 693 bytes was opened in the Win Hex tool. As seen from the experimental results, the file was recovered, and the contents of the file were intact.

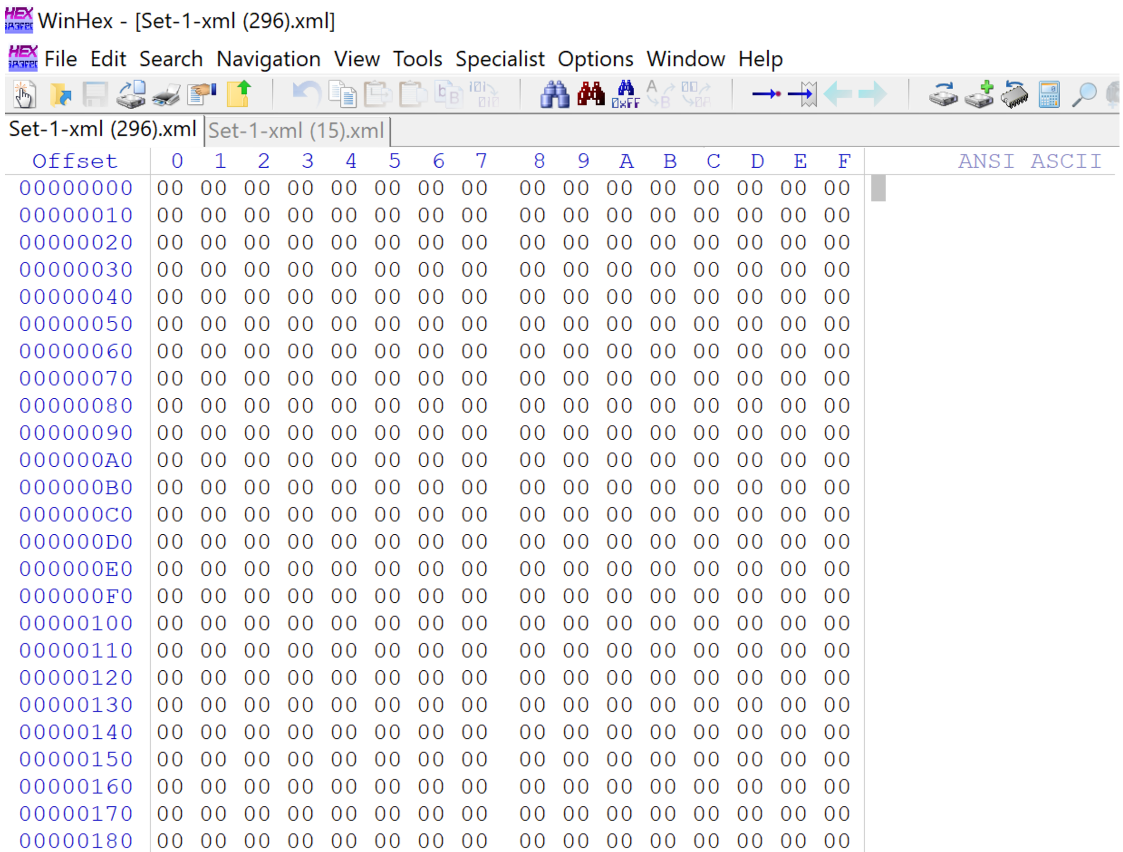


Fig. 8.8. File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM ON case, without using a USB WriteBlocker.

Figure 8.8 shows a snippet of an XML file with regards to the Seagate NVMe SSD TRIM ON case without using a USB WriteBlocker. The file over 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, but the contents of the file were corrupted, making the file unusable, as shown by the zeroes.

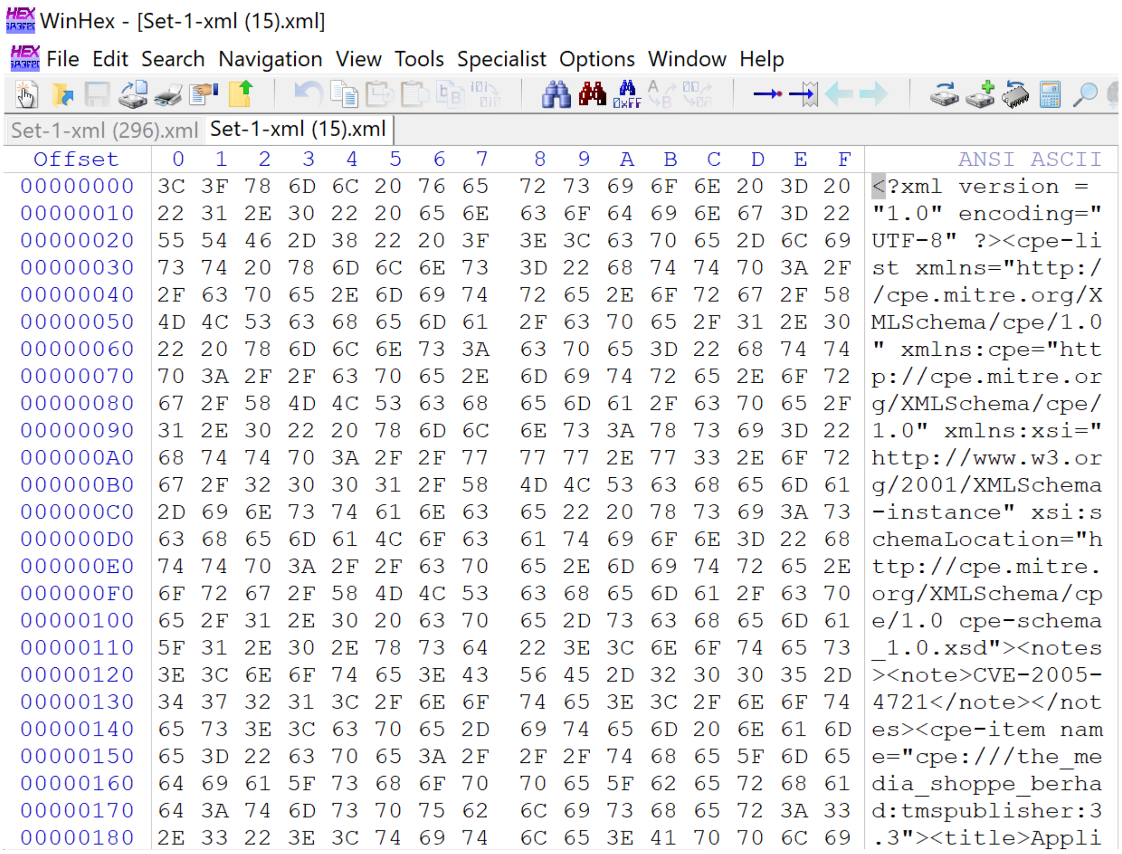


Fig. 8.9. File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM ON case, without using a USB WriteBlocker.

Figure 8.9 shows a snippet of the Set-1-xml(15).xml file with regards to the Seagate NVMe SSD TRIM ON case without using a USB WriteBlocker. The file under 696 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, and the contents of the file were intact.

Samsung and Seagate TRIM OFF Analysis without WriteBlocker

Interestingly, all files were recovered successfully in the TRIM OFF analysis from the four forensics Samsung NVMe and Seagate NVMe SSD images, respectively. This is because the TRIM OFF feature restricts the computer's operating system to inform the SSD to erase useless data blocks. Thus, the SSD controller no longer manages the available storage space to its full potential. Hence, in our experiment, the controller chip did not wipe/clear the pages of the storage devices. Therefore, this time the contents of all the files were intact. i.e., files could be opened and worked on regularly. Furthermore, there was no instance of file corruption in the case. Tables 8.10, 8.11, 8.12, and 8.13 show the statistics of different files used and the files that were recovered.

Table 8.10. The files recovered from FTK in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

Samsung FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
.bin	3	1^+	1^+	1^+	1^+
1. Some extra NTFS metadata files were theretoo 2. 66 pdf folders are created for some pdffiles 3. 4 zip files extracted inside folders +original files 4. + = Only one bin file was recovered + no traces of the two files.					

Table 8.11. The files recovered from Autopsy in Samsung NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

Samsung Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
.bin	3	1^{+}	1^{+}	1^{+}	1^{+}
1. + = Only one bin file was recovered + no traces of the two files.					

Table 8.12. The files recovered from FTK in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

Seagate FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115	115	115	115
.bin	3	1^{+}	1^{+}	1^{+}	1^{+}
1. + = Only one bin file was recovered + no traces of the two files.					

Table 8.13. The files recovered from Autopsy in Seagate NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

Seagate Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115***	115***	115***	115***
.bin	3	1^{+}	1^{+}	1^{+}	1^{+}
1. + = Only one bin file was recovered + no traces of the two files.					

Figure 8.10 shows a snippet of the Set-1-xml(296).xml file with regards to the Samsung NVMe SSD TRIM OFF case without using a USB WriteBlocker. The file over 693 bytes was opened in the WinHex tool. As seen from the experimental results, the file was recovered, and the contents of the file were not wiped, as shown by the hexadecimal characters.

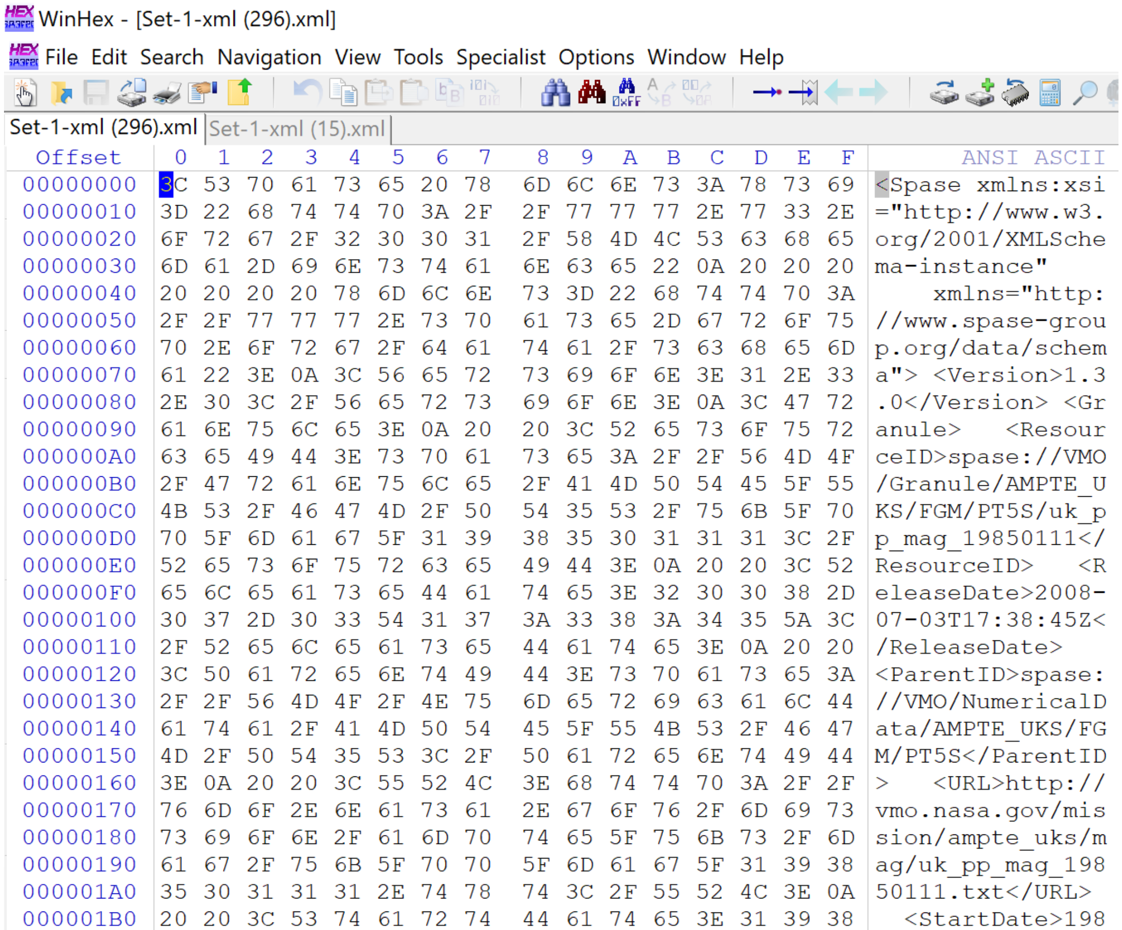


Fig. 8.10. File, Set-1-xml(296).xml, over 693 bytes in Samsung NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.11 shows a snippet of the Set-1-xml(15).xml file with regards to the Samsung NVMe SSD TRIM ON case without using a USB WriteBlocker. The file under 693 bytes was opened in the Win Hex tool. The file was recovered and intact. Moreover, the file contents were not wiped, as shown in the figure.

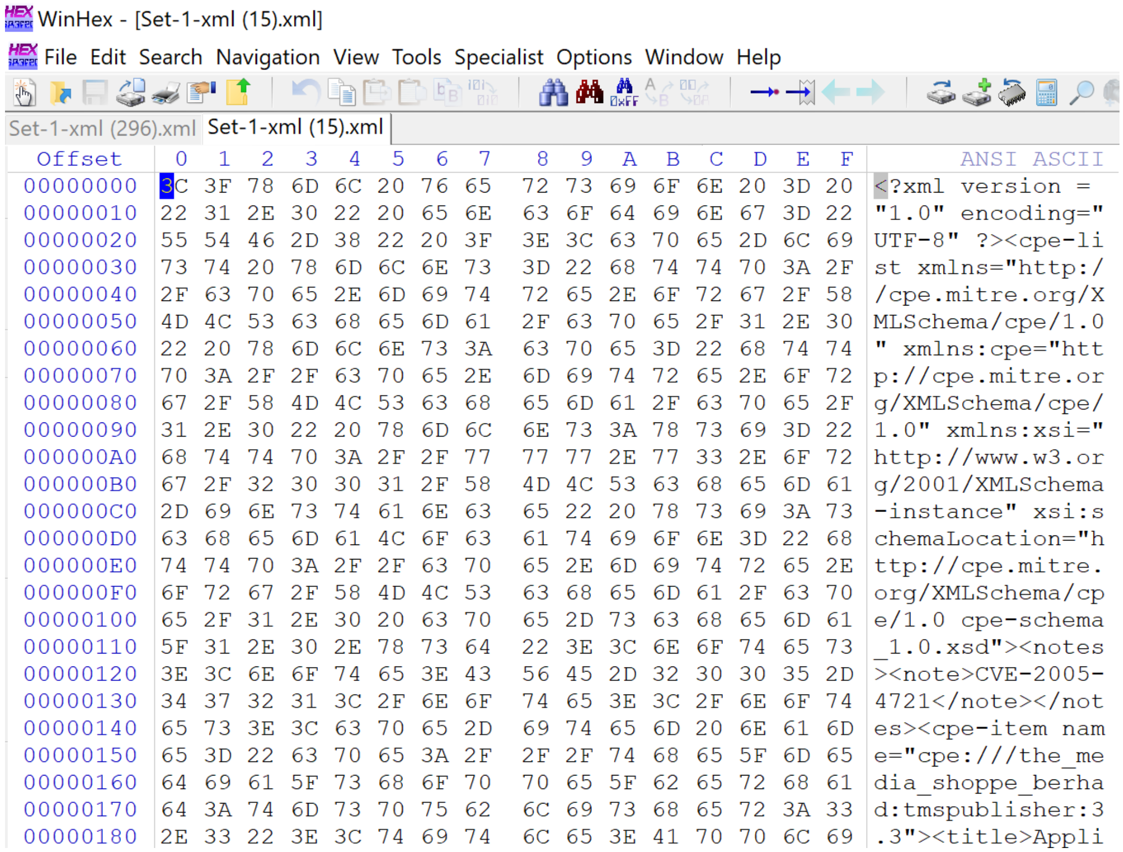


Fig. 8.11. File, Set-1-xml(15).xml, under 693 bytes in Samsung NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.12 shows a snippet of the Set-1-xml(296).xml file with regards to the Seagate NVMe SSD TRIM OFF case without using a USB WriteBlocker. The file over 696 bytes was opened in WinHex. As seen from the experimental results, the file was recovered, and the contents of the file were not wiped out.

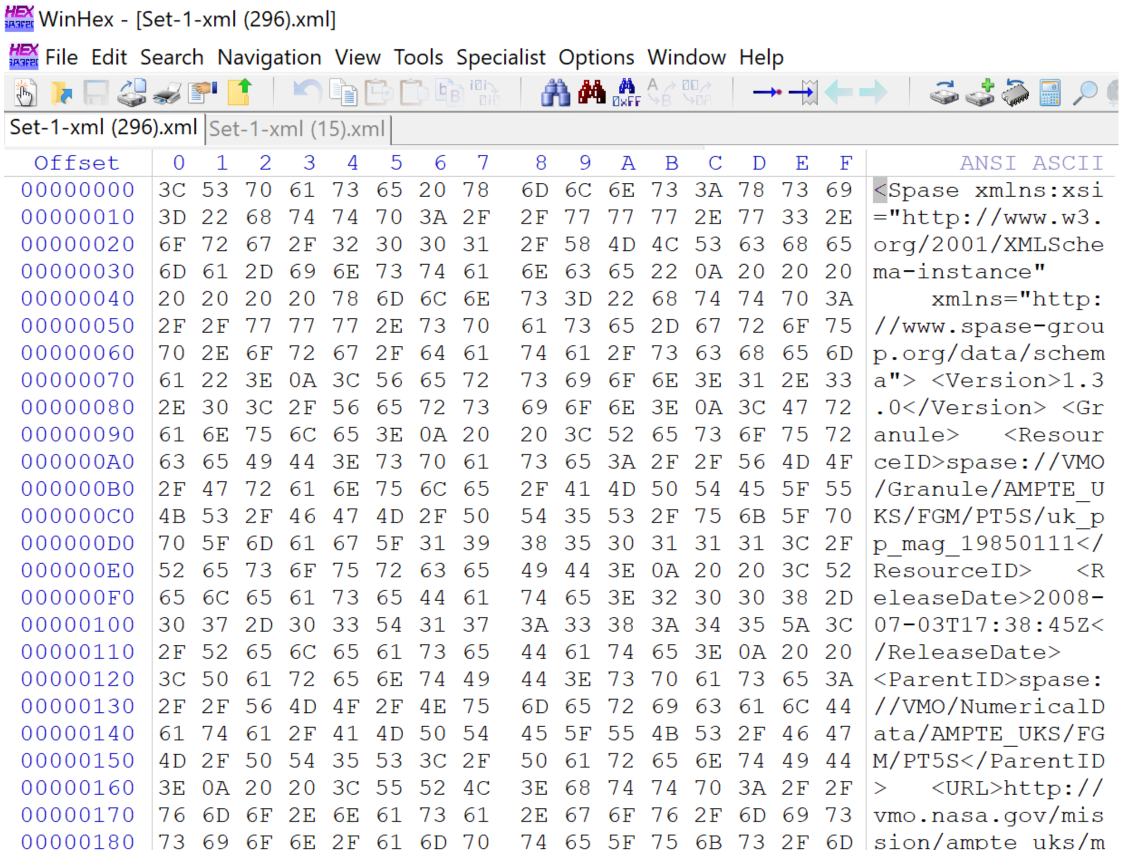


Fig. 8.12. File, Set-1-xml(296).xml, over 696 bytes in Seagate NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.13 shows a snippet of the Set-1-xml(15).xml file with regards to the Seagate NVMe SSD TRIM ON case without using a USB WriteBlocker. The file under 696 bytes was opened in WinHex and was fully recovered and intact. Moreover, the file contents were not wiped out, as shown by the hexadecimal characters.

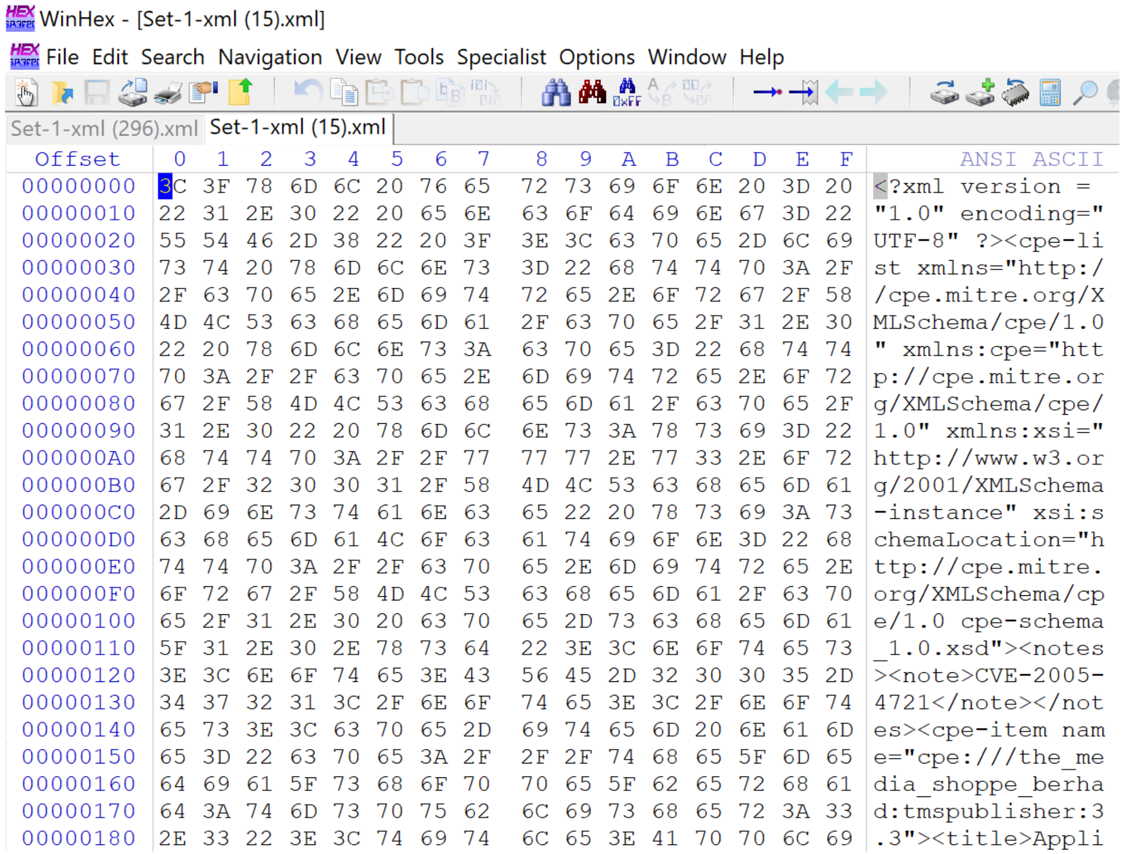


Fig. 8.13. File, Set-1-xml(15).xml, under 696 bytes in Seagate NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Hash Analysis for Samsung and Seagate NVMe SSDs without WriteBlocker

In this section, we exhibited our findings via MD5 hash values of the files following the TRIM ON and OFF recovery operations. We used the QuickHash hashing tool to generate the hash values.

Initially, the hash value of the original file is displayed, followed by TRIM ON and TRIM OFF MD5 hashes, and file size for Samsung NVMe SSD, as shown in figure 8.14. Figure 8.15 shows the original file hash values, TRIM ON and TRIM OFF MD5 hashes, and file size in the Seagate NVMe SSD case. These figures aim to validate and verify the claims made due to experimental observation without using a USB WriteBlocker.

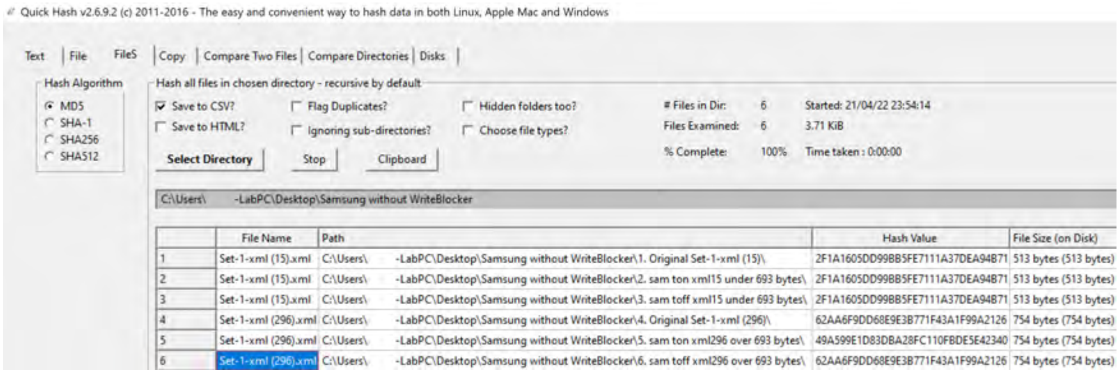


Fig. 8.14. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Samsung NVMe SSD, without using a USB WriteBlocker.

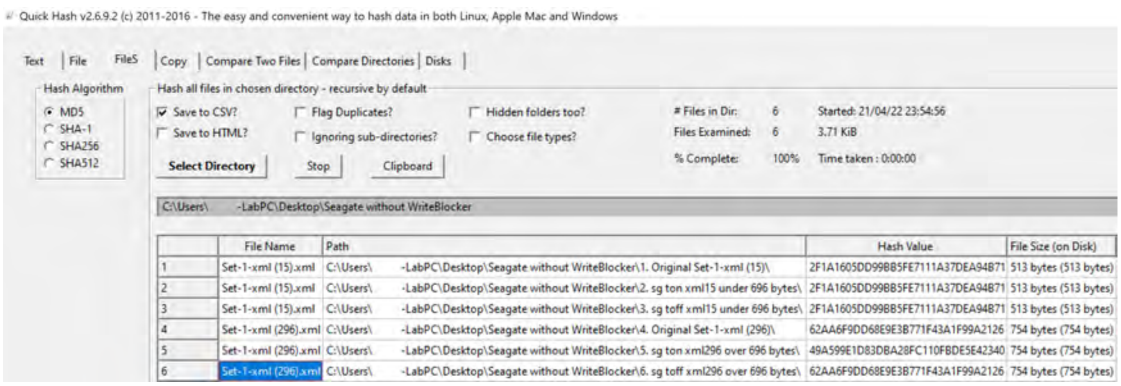


Fig. 8.15. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Seagate NVMe SSD, without using a USB WriteBlocker.

Table 8.14 shows names of all forensically acquired images, image type, size in kilobytes, MD5 and SHA1 hashes. The hash of all the image files constantly changed through the steps of the experiment. For example, in the case of TRIM OFF, the hash values of all the forensic images changed even though exact files were recovered. But in the case of TRIM ON for Samsung and Seagate NVMe SSDs, the hash values of all the forensics images were different, and file recovery was not possible when the file size was greater than 693 bytes in Samsung and 696 bytes in Seagate, respectively.

Table 8.14. Digital forensics information about forensically acquired image files of Samsung and Seagate NVMe SSDs without USB WriteBlocker.

File Names	**Image Type**	**Image Size (KB)**	**MD5 Hash**	**SHA1 Hash**
Imaging TRIM ON Samsung NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-sam_nvme_usb_image_1	e01	475 355	caa14bef0c7d14a9eb74dc5d25398d3b	c0c975cb9481ccb2cb07b9e6a769e622bd4daac8
wowb-sam_nvme_usb_image_2	e01	475 345	fd24d4766a662d6b5f6dd727ec003465	1f4826b5b9772fc09f30a58efab1526f3dec2b77
wowb-sam_nvme_usb_image_3	e01	464 929	800e62134ff8cb3857623e64a538a98b	d1f08981213c0351dc3860f9e56adcab7d3f0392
wowb-sam_nvme_usb_image_4	e01	464 938	79d60c51690d70884d06754d9f27e24c	f475f439d9ddca837989ceb1279b6790c280ba4e
Imaging TRIM ON Seagate NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-sg_nvme_usb_image_1	e01	475 291	fbc4779010c46231235fe743ef7496a2	5995419bd912727c30a7ce17d97023cdb9eca45f
wowb-sg_nvme_usb_image_2	e01	475 285	84b7d1820203e3e0db4d0fd2b44dbc2e	8248a1efcd9f1772ff7590a803960fd9d46c7724
wowb-sg_nvme_usb_image_3	e01	475 274	5f38ba9a16cd0c3e21ec31be08f7fbde	4eb13aa792194064e8687165b3e05e33c52f9616
wowb-sg_nvme_usb_image_4	e01	475 239	b97401f22b37038833e98a4f86f3bad5	7a47670619cc8f0dd11a270c17c0935934c8caa9
Imaging TRIM OFF Samsung NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-sam_nvme_usb_image_1	e01	154 416 700	bdb8dba80ba6ed739e7859a33d2a71db	bbdddd59629006ac182d40e9b76389e95aaf7175
wowb-sam_nvme_usb_image_2	e01	154 416 696	5b21f56dcbe770128c973655f8a014de	c4e5df22f31cb9169c197d5204d0ca37017723d9
wowb-sam_nvme_usb_image_3	e01	154 416 691	245aa23ad114d924c3f38ce9ef81aa22	76287dbe8413c181b428637788e4be61dd68563b
wowb-sam_nvme_usb_image_4	e01	154 416 684	d705b9e4db3b0db42f1d2300a6b18b87	16582e106d9867f96de74ec6c5537aff7536cfa2
Imaging TRIM OFF Seagate NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-sg_nvme_usb_image_1	e01	154 417 191	fee0b2d53bf09e7fe740dc96c9805580	f9e0b8c27216d979e39b64383e7cb466265d89e9
wowb-sg_nvme_usb_image_2	e01	154 417 184	4b5b749b9b7a0386f4cdb9a983eff1d6	2cd6ce0c9529995d5fdd65d4ad964bcbed352bdb
wowb-sg_nvme_usb_image_3	e01	154 417 182	744899aa01d83bbcce0c57b051aef940	06f16e22d7b749e41129a620efa90432aa03d40c
wowb-sg_nvme_usb_image_4	e01	154 417 177	4aed8747238c684e4950cbeb974e42c2	399caf5b1a10b8136f2cc1c913266c4481f003d3

Western Digital and Silicon Power TRIM ON Analysis without WriteBlocker

The analysis of TRIM ON cases in Western Digital and Silicon Power shows a similar trend in file recovery procedures as was seen in Seagate NVMe SSD. The controller chip did not act on files under 696 bytes in Western Digital and Silicon Power storage devices. As a result, they were all intact without any file content corruption. Tables 8.15, 8.16, 8.17, and 8.18 show the statistics of different files used and the files that were recovered.

Table 8.15. The number of files recovered from FTK in Western Digital (WD) NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

WD FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3*	3*	3*	3*
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. +: Recovered all but one file of out the three was corrupted. Note: 1) Files under 696 bytes were intact after recovery in Western Digital (WD) NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 8.16. The number of files recovered from Autopsy in Western Digital (WD) NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

WD Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3*	3*	3*	3*
*: All files recovered but corrupted. ** : All files recovered but 8280 corrupted + 92 not corrupted. *** : All files recovered but 69 corrupted + 46 not corrupted. +: Recovered all but one file of out the three was corrupted. Note: 1) Files under 696 bytes were intact after recovery in Western Digital (WD) NVMe SSD. 2) For corrupted files, the size of the files was the same, but the contents were wiped out.					

Table 8.17. The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

SP FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3*	3*	3*	3*

*: All files recovered but corrupted.
** : All files recovered but 8280 corrupted + 92 not corrupted.
*** : All files recovered but 69 corrupted + 46 not corrupted.
+: Recovered all but one file of out the three was corrupted.
Note:
1) Files under 696 bytes were intact after recovery in Silicon Power (SP) NVMe SSD.
2) For corrupted files, the size of the files was the same, but the contents were wiped out.

Table 8.18. The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM ON case.

SP Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976*	20976*	20976*	20976*
.docx	161	161*	161*	161*	161*
.ppt	13524	13524*	13524*	13524*	13524*
.pptx	23	23*	23*	23*	23*
.xls	14881	14881*	14881*	14881*	14881*
.xlsx	46	46*	46*	46*	46*
.pdf	59432	59432*	59432*	59432*	59432*
.xml	8372	8372**	8372**	8372**	8372**
.jpg	27577	27577*	27577*	27577*	27577*
.png	920	920*	920*	920*	920*
.mp4	92	92*	92*	92*	92*
.zip	115	115***	115***	115***	115***
.bin	3	3*	3*	3*	3*

*: All files recovered but corrupted.
** : All files recovered but 8280 corrupted + 92 not corrupted.
*** : All files recovered but 69 corrupted + 46 not corrupted.
+: Recovered all but one file of out the three was corrupted.
Note:
1) Files under 696 bytes were intact after recovery in Silicon Power (SP) NVMe SSD.
2) For corrupted files, the size of the files was the same, but the contents were wiped out.

Western Digital and Silicon Power TRIM OFF Analysis without WriteBlocker

File recovery successfully took place in the TRIM OFF analysis from the four forensics Western Digital and Silicon Power NVMe SSD images, respectively, using AccessData FTK and Autopsy tools. All the files were intact as the functionality of SSD controller chip was limited by TRIM OFF feature of the operating system. Therefore, the controller chip did not clear out the pages having deleted data from the storage devices in our experiment. Tables 8.19, 8.20, 8.21, and 8.22 show the statistics of files recovery.

Table 8.19. The number of files recovered from FTK in Western Digital NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

WD FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115***	115***	115***	115***
.bin	3	1^+	1^+	1^+	1^+
1. Some extra NTFS metadata files were there too. 2. 66 pdf folders are created for some pdf files. 3. ***4 zip files extracted inside folders +original files. 4. + = Only one bin file was recovered + no traces of the two files.					

Table 8.20. The number of files recovered from Autopsy in Western Digital NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

WD Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115***	115***	115***	115***
.bin	3	1^+	1^+	1^+	1^+
1. Some extra NTFS metadata files were there too. 2. 66 pdf folders are created for some pdf files. 3. ***4 zip files extracted inside folders +original files. 4. + = Only one bin file was recovered + no traces of the two files.					

Table 8.21. The number of files recovered from FTK in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

SP FTK Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115***	115***	115***	115***
.bin	3	3	3	3	3
1. Some extra NTFS metadata files were there too. 2. 66 pdf folders are created for some pdf files. 3. ***4 zip files extracted inside folders +original files.					

Table 8.22. The number of files recovered from Autopsy in Silicon Power NVMe SSD in USB enclosure adapter without using WriteBlocker in Windows 10 TRIM OFF case.

SP Autopsy Case Statistics in Windows 10 without WriteBlocker					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.doc	20976	20976	20976	20976	20976
.docx	161	161	161	161	161
.ppt	13524	13524	13524	13524	13524
.pptx	23	23	23	23	23
.xls	14881	14881	14881	14881	14881
.xlsx	46	46	46	46	46
.pdf	59432	59432	59432	59432	59432
.xml	8372	8372	8372	8372	8372
.jpg	27577	27577	27577	27577	27577
.png	920	920	920	920	920
.mp4	92	92	92	92	92
.zip	115	115***	115***	115***	115***
.bin	3	3	3	3	3
1. Some extra NTFS metadata files were there too. 2. 66 pdf folders are created for some pdf files. 3. ***4 zip files extracted inside folders +original files.					

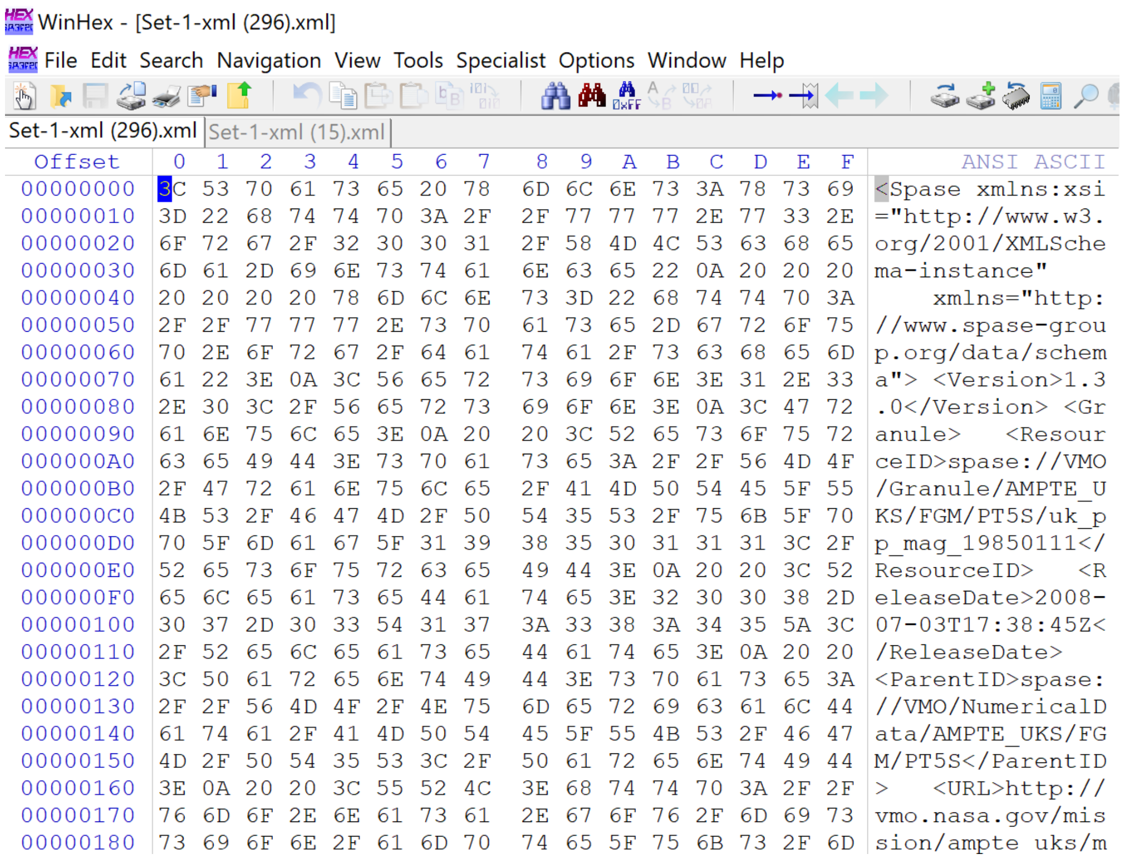


Fig. 8.16. File, Set-1-xml(296).xml, over 696 bytes in Western Digital NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.16 shows a snippet of an XML file with regards to the Western Digital NVMe SSD TRIM OFF case without using a USB WriteBlocker. The file over 696 bytes was opened in WinHex. As seen from the experimental results, the file was recovered and the contents of the file were not wiped, as shown by the hexadecimal characters.

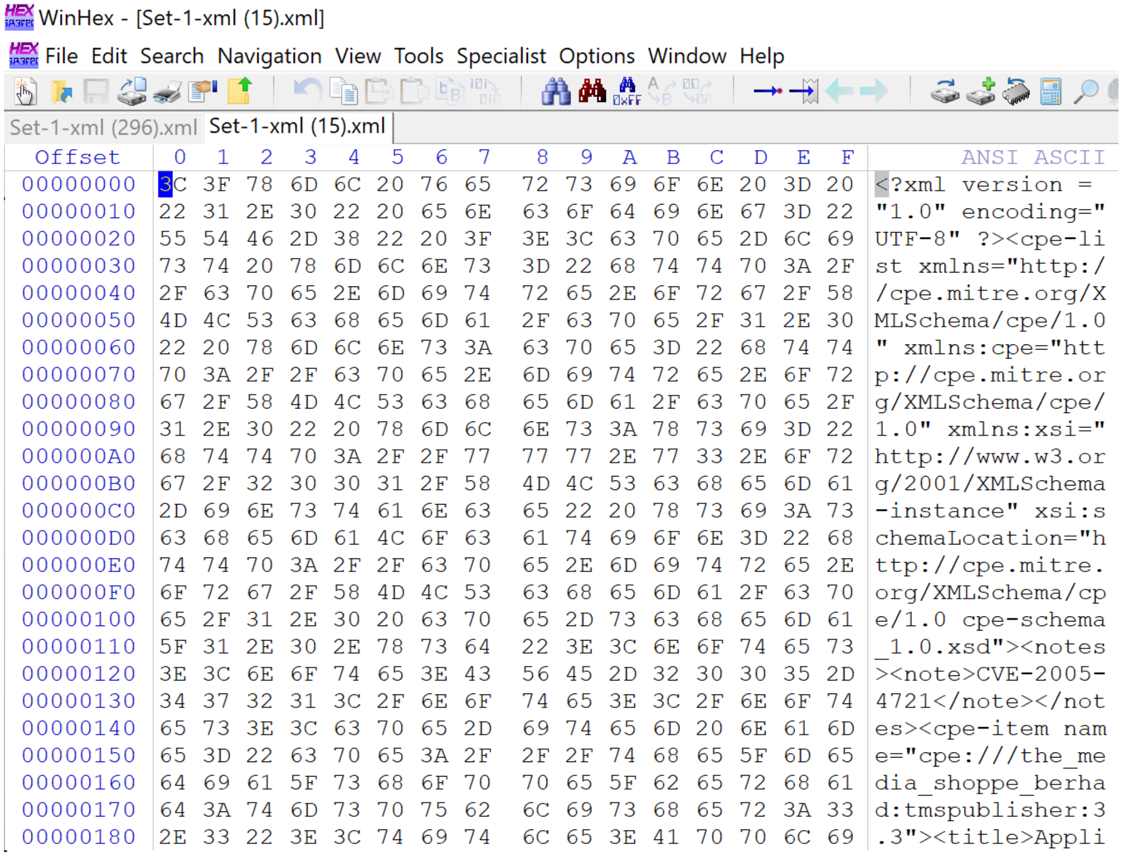


Fig. 8.17. File, Set-1-xml(296).xml, under 696 bytes in Western Digital NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.17 shows a snippet of an XML file with regards to the Western Digital NVMe SSD TRIM OFF case without using a USB WriteBlocker. The file under 696 bytes was opened in WinHex. As seen from the experimental results, the file was recovered and the contents of the file were not wiped, as shown by the hexadecimal characters.

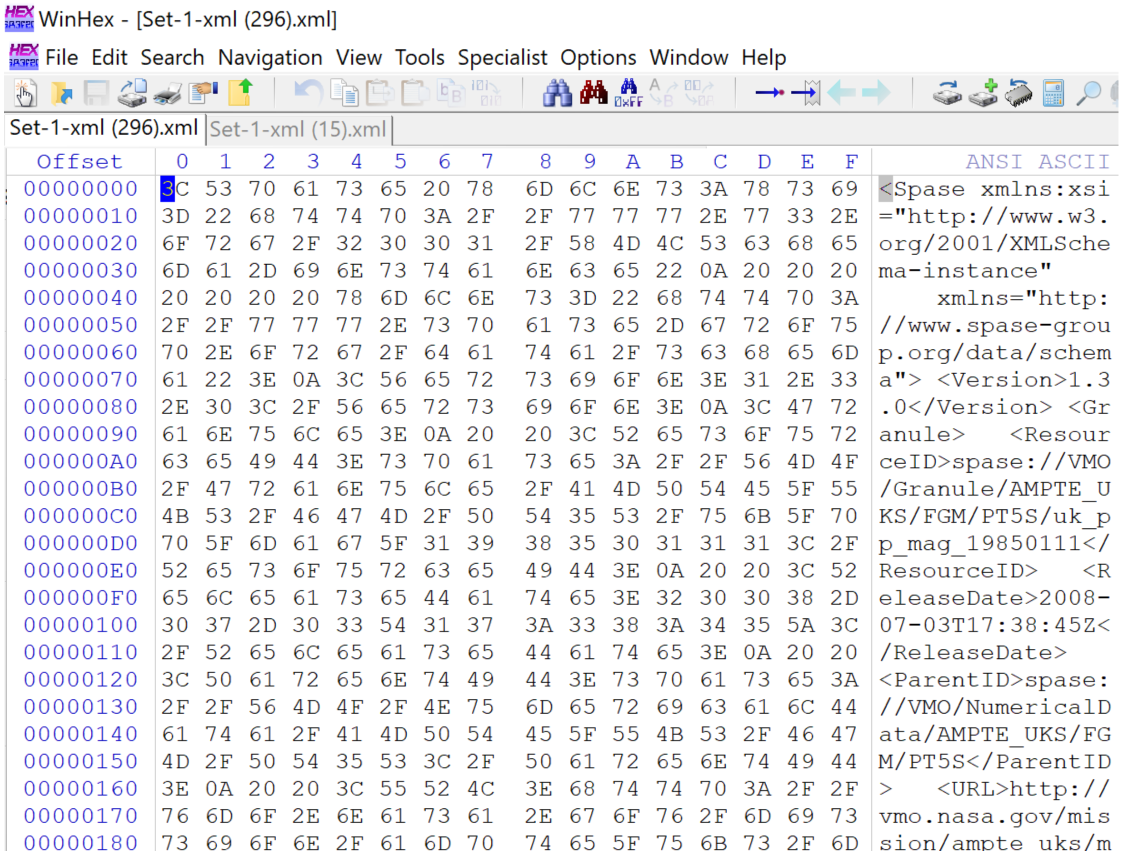


Fig. 8.18. File, Set-1-xml(296).xml, over 696 bytes in Silicon Power NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.18 shows a snippet of an XML file with regards to the Silicon Power NVMe SSD TRIM OFF case without using a USB WriteBlocker. The file over 696 bytes was opened in WinHex. As seen from the experimental results, the file was recovered and the contents of the file were not wiped out.

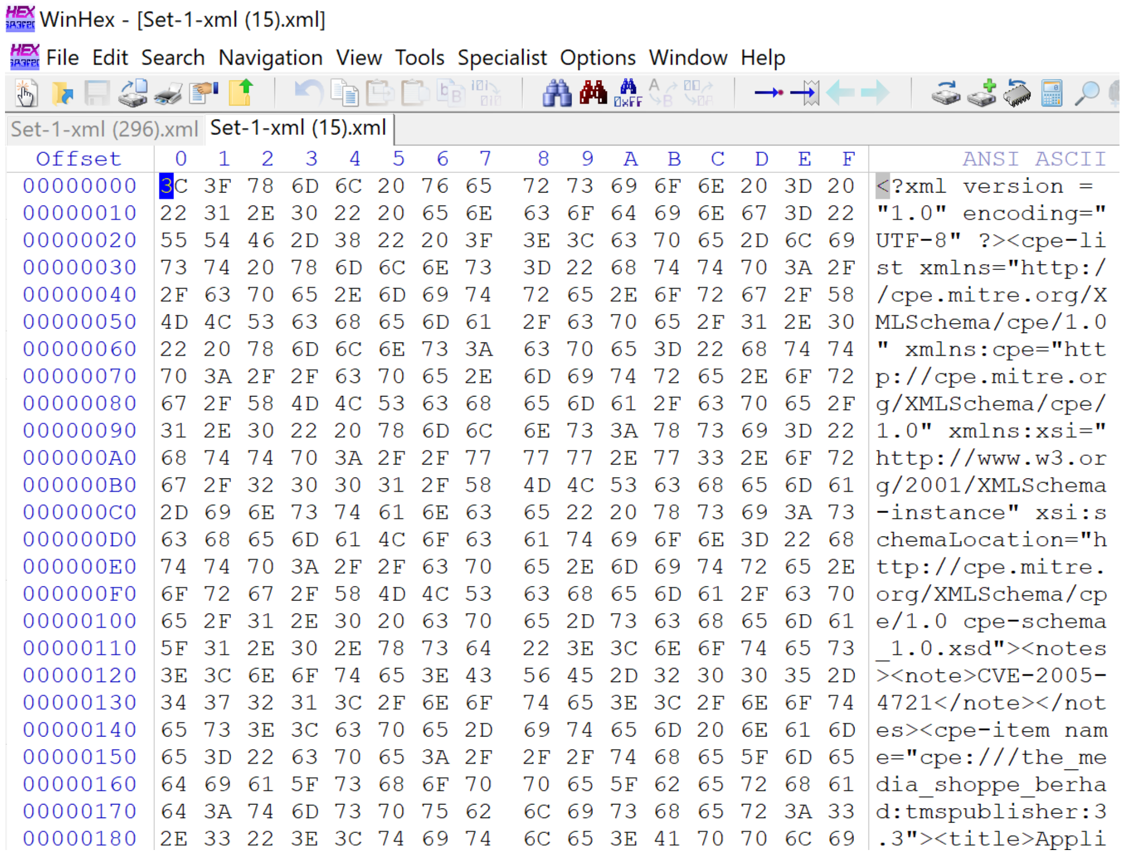


Fig. 8.19. File, Set-1-xml(296).xml, under 696 bytes in Silicon Power NVMe SSD TRIM OFF case, without using a USB WriteBlocker.

Figure 8.19 shows a snippet of an XML file with regards to the Silicon Power NVMe SSD TRIM OFF case without using a USB WriteBlocker. The file under 696 bytes was opened in WinHex. As seen from the experimental results, the file was recovered and the contents of the file were not wiped, as shown by the hexadecimal characters.

Hash Analysis for Western Digital and Silicon Power NVMe SSDs without WriteBlocker

In this section, we exhibited the MD5 hash values for the TRIM ON and OFF cases using QuickHash. The hash values of the original file, followed by TRIM ON and TRIM OFF MD5 hashes, and file size for Western Digital and Silicon Power NVMe SSDs are shown in figures 8.20, 8.21. The figures aim to validate and verify the claims made due to experimental observation when a USB WriteBlocker was not used.

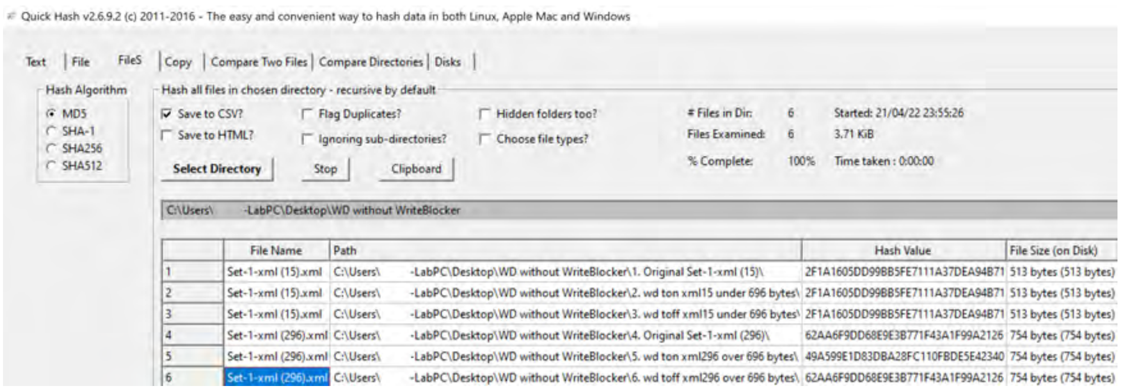


Fig. 8.20. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Western Digital NVMe SSD, without using a USB WriteBlocker.

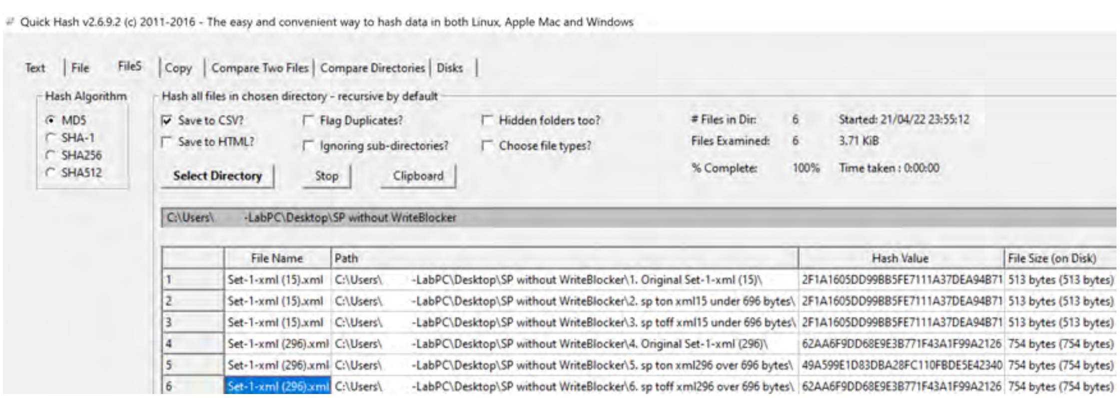


Fig. 8.21. Hash values of Set-1-xml(15).xml and Set-1-xml(296).xml files in original dataset, and after recovery from TRIM ON and OFF cases in Silicon Power NVMe SSD, without using a USB WriteBlocker.

Table 8.23. Digital forensics information about forensically acquired image files of Western Digital and Silicon Power NVMe SSDs without USB WriteBlocker.

File Names	Image Type	Image Size (KB)	MD5 Hash	SHA1 Hash
Imaging TRIM ON Western Digital NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-wd_nvme_usb_image_1	e01	475 306	0582ad4002ad5ce34aae34b78fed3722	ad6f3e913439d85779e3bf1070f285c564126f45
wowb-wd_nvme_usb_image_2	e01	475 299	4a18adfc9a920bf6a6c2fc054ef485f8	dc7a663f4e2e5ac2bb7209caffad955170af0132
wowb-wd_nvme_usb_image_3	e01	464 928	ba4322c3bd2f8e0bb13ec08b64227d0a	b9f292ff1769868a44bb5836c9c0590186616d4d
wowb-wd_nvme_usb_image_4	e01	464 936	dda9ef260b077c7aac76c133a63d8180	f702d88df3f8ab1e8a249c9460e432f2dfe035fd
Imaging TRIM ON Silicon Power NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-sp_nvme_usb_image_1	e01	486 033	4f718d16b26eb07e3b5de64fdce73425	eca0d91389c9f6941d890b7f130618acadead308
wowb-sp_nvme_usb_image_2	e01	486 027	de52b7b0f4389a94f3282a7bd21b6c1d	8b9aa4266a8f6dd6842b277466c3bef32fefbd0c
wowb-sp_nvme_usb_image_3	e01	486 016	06923b858229c610091da1d0afa7b0fc	1373dc05e1874bec5f38e61ec36028920fd7a205
wowb-sp_nvme_usb_image_4	e01	485 948	ec2615f04b4f91b5609fb5455b99d2da	f72b8a959254fbdfa0818d0c4c3f7df634da2909
Imaging TRIM OFF Western Digital NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-wd_nvme_usb_image_1	e01	154 416 713	c2627eccde4010871bedb368fe7515fb	0e4e2d30159285bf4978e7f3ccb50a36bdce2347
wowb-wd_nvme_usb_image_2	e01	154 416 708	f9bbff78205f7e5363876daf84af4545	380a67e713dbae2af795e64a6deae41cd5e8b7b5
wowb-wd_nvme_usb_image_3	e01	154 416 704	973af13c1e82ef0021ea4eed06da438d	51a452b587b272daef602e9739a2125aa1b45d9a
wowb-wd_nvme_usb_image_4	e01	154 416 698	c84f017d98a2c339340dc1ceb6bf089e	ab4d01f3ea593c0d40a02d3767b8116a29f91926
Imaging TRIM OFF Silicon Power NVMe SSD without USB WriteBlocker using FTK Imager				
wowb-sp_nvme_usb_image_1	e01	160 278 230	595cc00fa3aa9a1c6d31b1b0ca6fd9f3	9bb411ea6bab251eed064d20dcd0e8cdd68009f8
wowb-sp_nvme_usb_image_2	e01	160 276 206	bebab4cfefb700f24b18dad67da369f2	e49254783e27a902027fa4ceddcee7f3081e7196
wowb-sp_nvme_usb_image_3	e01	160 276 206	36c58edb5af5f984896ce1889984729d	1bc35bd1991f11dc03ec750a738e97543b71cbf7
wowb-sp_nvme_usb_image_4	e01	160 276 204	3d74d5457243b0636ebacf4eb3b84780	836589536518c853f1eeeed0543ec3d12e627a6c

CHAPTER IX

Digital Forensics in PCIe NVMe SSDs with NVMe WriteBlocker

PCIe NVMe SSDs are becoming popular in computer systems. They are now gradually replacing the regular SATA SSDs as a primary boot device. A primary boot device is a storage device with an operating system installed on it. Due to the advanced protocol of NVMe, the speed of reading and writing operations in NVMe SSDs is far greater than in SATA SSDs. In this chapter, we have worked on NVMe SSDs as a primary boot device. We have installed Windows 10 v21H2 operating system on the SSD devices. The work in this chapter is similar to chapters VII and VIII. However, the results produced by NVMe SSDs are distinct when it comes to hashing and file recovery when a dedicated NVMe WriteBlocker is used when acquiring forensics images.

We used Autopsy, AccessData FTK [77], and WinHex [78] tools to recover and conduct a digital forensics examination. Lastly, we explained the forensics observations based on the findings with various controller chips of the four NVMe SSD devices.

Experimental Setup with NVMe WriteBlocker

Table 9.1 below enumerates the technical specifications of the equipment we have used for the experiment in this chapter. The equipment used for this experiment is the same as the one used in chapter VII but we used Wiebetech NVMe WriteBlocker instead of the USB WriteBlocker. Lastly, figures 9.1, 9.2, 9.3, and 9.4 show the NVMe SSDs attached to the NVMe WriteBlocker.

Table 9.1. Equipment used in the experiment with NVMe WriteBlocker.

Tools	Name
NVMe SSD 1	Samsung V-NAND SSD 970 Evo Plus
NVMe SSD 2	Seagate Barracuda 510 250GB NVMe SSD
NVMe SSD 3	Western Digital SN550 250GB NVMe SSD
NVMe SSD 4	Silicon Power 3D-NAND NVMe SSD
Operating System	Windows 10 Pro v21H2
Forensic Analysis Tool	AccessData FTK 7.5 and WinHex
Forensics Acquisition Tool	AccessData FTK Imager 4.7
WriteBlocker	Wiebetech NVMe WriteBlocker
Workstation	CPU: Intel Xeon W-2123 — RAM : 80GB

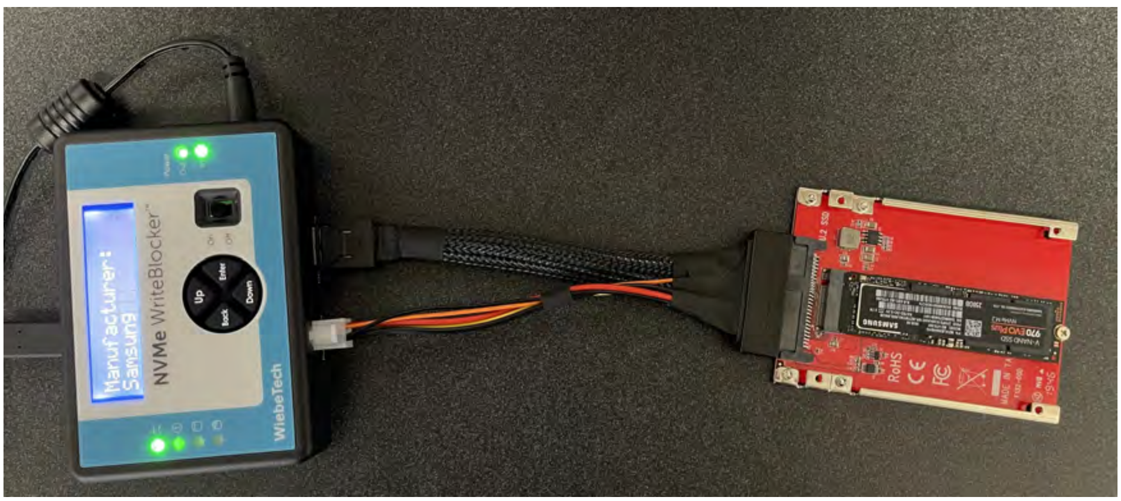


Fig. 9.1. Samsung NVMe SSD attached with NVMe WriteBlocker.

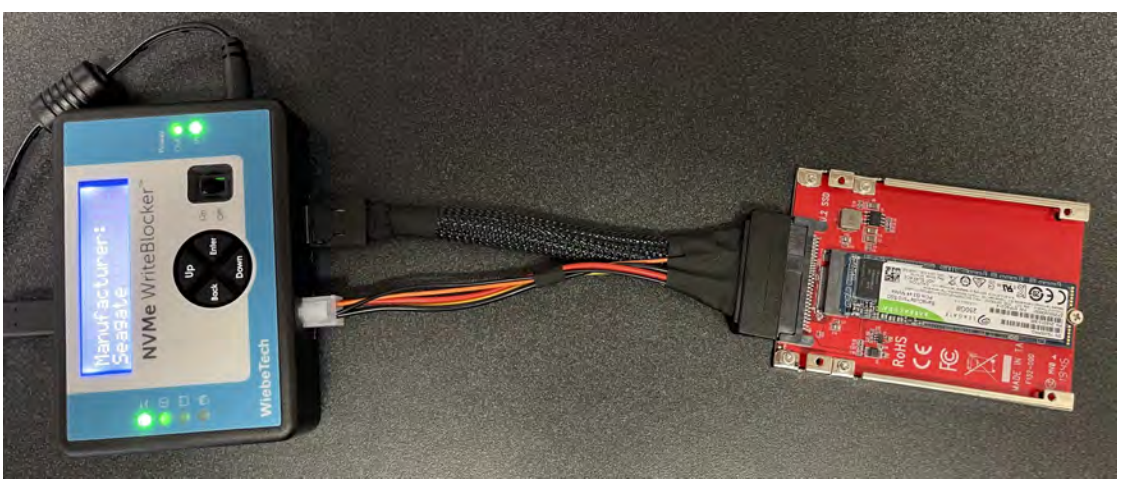


Fig. 9.2. Seagate NVMe SSD attached with NVMe WriteBlocker.

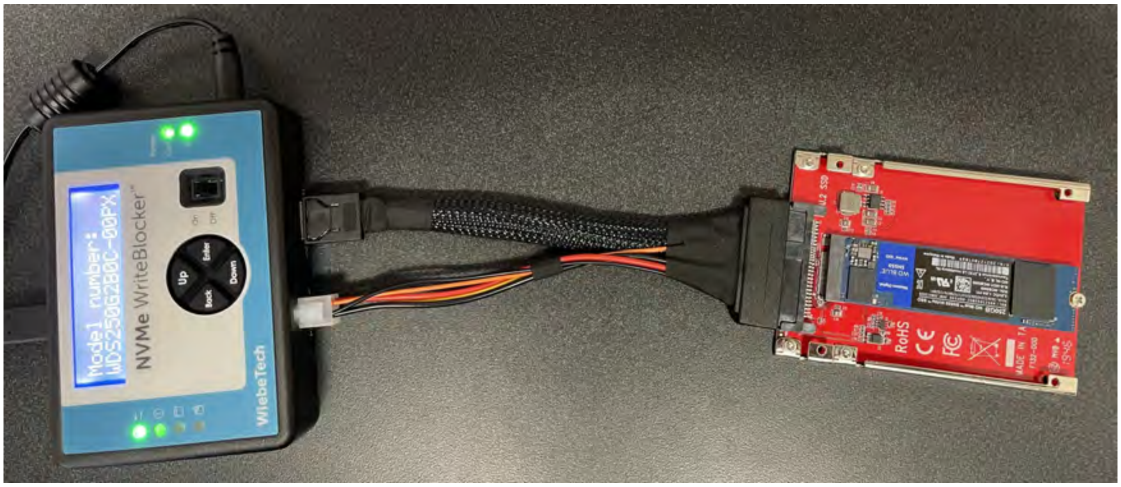


Fig. 9.3. Western Digital NVMe SSD attached with NVMe WriteBlocker.

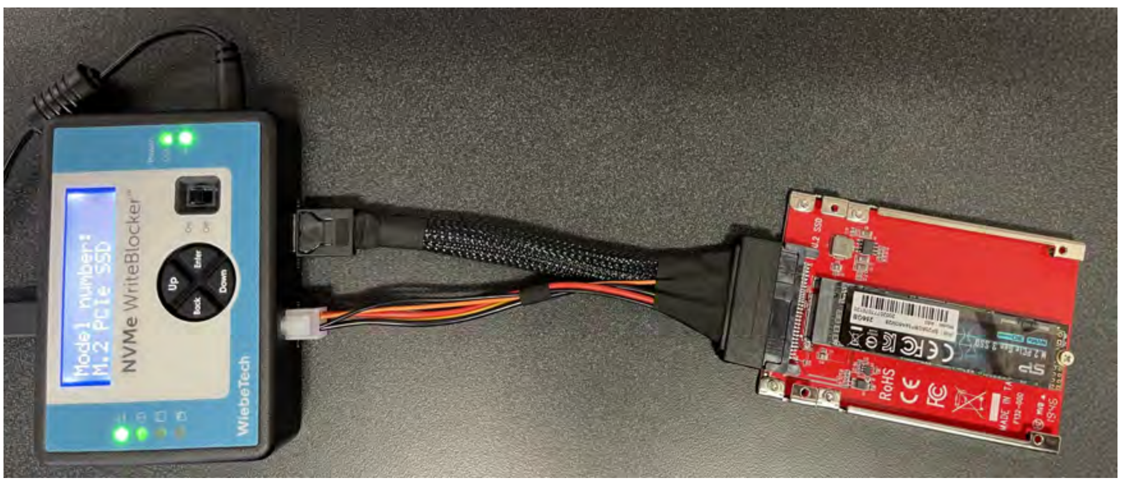


Fig. 9.4. Silicon Power NVMe SSD attached with NVMe WriteBlocker.

Specifications of SSDs

The test in this chapter included the same four NVMe SSD brands: Samsung, Seagate, Western Digital (WD), and Silicon Power (SP). We chose these drives due to their dense popularity, market share, and dependability. We chose them because the parameters of the SSDs used in the experiment closely mimic those of a standard SSD that a regular user may own, so they will reflect a real-world scenario. Furthermore, the experiment is more relevant to the digital forensic community because these are the most frequent properties of SSDs found in laptops and desktop computers. The tables 9.2 and 9.3 detail the name, model, product number (P/N), storage capacity, number of flash chips, kind of NVMe flash chip, and controller information for NVMe SSDs.

Table 9.2. Information of Samsung and Seagate NVMe SSDs used in the experiment.

SSD Information	Samsung NVMe Specification 1.3
Name	Samsung NVMe V-NAND SSD 970 Evo Plus NVMe M.2
Model	MZ-V7S250
P/N	MZVLB250HBHQ
Storage Capacity	250 GB
Number of flash chips inside	2
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Samsung S4LR020 — 2117 ARM — Pheonix

SSD Information	Seagate NVMe Specification 1.3
Name	Seagate Barracuda 510 250GB NVMe SSD
Model	ZP250CM30001
P/N	2NS312-300
Storage Capacity	250 GB
Number of flash chips inside	4
Type of NVMe NAND Flash	3D TLC NAND
Controller information	SKHynix - H5AN4G6NBJR

Table 9.3. Information of Western Digital and Silicon Power NVMe SSDs used in the experiment.

SSD Information	WD NVMe Specification 1.4
Name	Western Digital SN550 250GB NVMe SSD
Model	WDS250G2B0C-00PXH0/21146P801302
P/N	87161901478830731375399388282263
Storage Capacity	250 GB
Number of flash chips inside	4
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Sandisk 20-82-10023-A1 — 1015ZKLY0KN

SSD Information	Silicon Power NVMe Specification 1.3
Name	Silicon Power 3D-NAND NVMe SSD
Model	A-60
P/N	SP256GBP34A60M28
Storage Capacity	256 GB
Number of flash chips inside	2
Type of NVMe NAND Flash	3D TLC NAND
Controller information	Phison PS5013-E13-31—C02102E— TB5V79/ 001BB

Methodology and Experiment Initiation

The methodology followed and configuration assigned during the experiment are listed and explained in this section.

1. The partition scheme used for the NVMe SSDs: **GPT** (**GUID Partition Table**)
2. The number of partitions in each NVMe SSD: **1**

3. The file system of the one partition: **NTFS**
4. Before copying the files to the primary boot devices from Digital Corpora [80], we checked the **TRIM** status in Windows 10 by issuing the following command through the Windows command prompt (CMD).

fsutil behavior query DisableDeleteNotify

**If the output is 1, then TRIM is disabled. If the output is 0, then TRIM is enabled.*

To enable TRIM: fsutil behavior set DisableDeleteNotify 0

To disable TRIM: fsutil behavior set DisableDeleteNotify 1

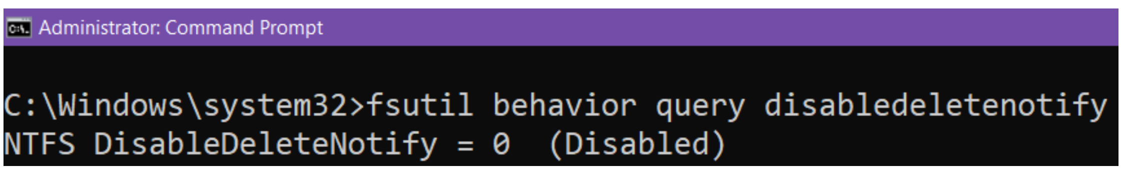


Fig. 9.5. The status of TRIM in Windows 10 using fsutil command issued from CMD.

Case scenario: TRIM ON from Windows 10 operating system with NVMe WriteBlocker

1. We copied the commonly used file types having 160GB of total size, from the Digital Corpora dataset [80] to the four NVMe SSDs.
2. We then kept the system powered on for one day with no user activity.
3. Next, we deleted (shift+delete) the files from the devices and waited for one day before acquiring four forensic images of the four NVMe SSDs respectively.
   (a) We took four forensic images: three consecutive images with one day gap and last image after a span of four days from the third acquisition.

4. We analyzed the images in AccessData FTK and Autopsy for the NVMe storage devices.
5. We performed file recovery of the deleted files from the forensics images in the TRIM ON case.
6. Based on our results from the file recovery and WinHex analysis we documented the effects of wear-leveling.

Case scenario: TRIM OFF from Windows 10 operating system with NVMe WriteBlocker

1. First and foremost, we disabled TRIM using Windows 10 command prompt (CMD) before copying the files.
2. We copied the commonly used file types having 160GB of total size, from the Digital Corpora dataset [80] to the four NVMe SSDs.
3. We then kept the system powered on for one day with no user activity.
4. Next, we deleted (shift+delete) the files from the devices and waited for one day before acquiring four forensic images of the four NVMe SSDs respectively.
   (a) We took four forensic images: three consecutive images with one day gap and last image after a span of four days from the third acquisition.
5. We analyzed the images in AccessData FTK and Autopsy for the NVMe storage devices.
6. We performed file recovery of the deleted files from the forensics images in the TRIM OFF case.
7. Like the TRIM ON case, based on our results from the file recovery and WinHex analysis, we documented the effects of wear-leveling.

Experiment Results, Analysis, and Discussion

The results of the file recovery utilizing the AccessData FTK and Autopsy tools are presented in this section. We began by populating the NVMe SSDs with the most frequently used files from the Digital Corpora dataset [80]. We then used the forensically acquired images of the four NVMe SSDs using the NVMe WriteBlocker to undertake the file recovery operation. Tables 9.4 and 9.5 present the timeline information of forensic image acquisition in both TRIM ON and TRIM OFF scenarios of Samsung, Seagate, Western Digital (WD), and Silicon Power (SP) NVMe SSDs.

Table 9.4. Timeline information of forensic file acquisition with NVMe WriteBlocker.

TRIM ON information with NVMe WriteBlocker			
Samsung NVMe	**Time**	**Seagate NVMe**	**Time**
Copy file date	11:49 pm 2/11/22	Copy file date	5:30 pm 2/20/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	11:49 pm 2/12/22	Delete files	5:30 pm 2/21/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	11:49 pm 2/13/22	1st image	5:30 pm 2/22/22
2nd image	11:49 pm 2/14/22	2nd image	5:30 pm 2/23/22
3rd image	11:49 pm 2/15/22	3rd image	5:30 pm 2/24/22
4th image	11:49 pm 2/19/22	4th image	5:30 pm 2/28/22
TRIM OFF information with NVMe WriteBlocker			
Samsung NVMe	**Time**	**Seagate NVMe**	**Time**
Copy file date	11:09 pm 2/28/22	Copy file date	10:23 pm 3/1/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	11:09 pm 3/1/22	Delete files	10:23 pm 3/2/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	11:09 pm 3/2/22	1st image	10:23 pm 3/3/22
2nd image	11:09 pm 3/3/22	2nd image	10:23 pm 3/4/22
3rd image	11:09 pm 3/4/22	3rd image	10:23 pm 3/5/22
4th image	11:09 pm 3/8/22	4th image	10:23 pm 3/9/22

Table 9.5. Timeline information of forensic file acquisition with NVMe WriteBlocker.

TRIM ON information with NVMe WriteBlocker			
WD NVMe	**Time**	**SP NVMe**	**Time**
Copy file date	9:27 pm 2/22/22	Copy file date	1:18 pm 2/25/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	9:27 pm 2/23/22	Delete files	1:18 pm 2/26/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	9:27 pm 2/24/22	1st image	1:18 pm 2/27/22
2nd image	9:27 pm 2/25/22	2nd image	1:18 pm 2/28/22
3rd image	9:27 pm 2/26/22	3rd image	1:18 pm 3/1/22
4th image	9:27 pm 3/2/22	4th image	1:18 pm 3/5/22
TRIM OFF information with NVMe WriteBlocker			
WD NVMe	**Time**	**SP NVMe**	**Time**
Copy file date	8:43 pm 3/2/22	Copy file date	9:59 pm 3/4/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
Delete files	8:43 pm 3/3/22	Delete files	9:59 pm 3/5/22
Wait for 24 hrs	Waited	Wait for 24 hrs	Waited
1st image	8:43 pm 3/4/22	1st image	9:59 pm 3/6/22
2nd image	8:43 pm 3/5/22	2nd image	9:59 pm 3/7/22
3rd image	8:43 pm 3/6/22	3rd image	9:59 pm 3/8/22
4th image	8:43 pm 3/10/22	4th image	9:59 pm 3/12/22

Samsung and Seagate TRIM ON Analysis with NVMe WriteBlocker

The TRIM ON analysis of Samsung NVMe SSD with NVMe WriteBlocker (WB) shows that most files become unrecoverable even after one day of deletion. As we have seen in chapters VII and VIII, in the case of Samsung NVMe SSD used under a USB enclosure, files with file size under 693 bytes stay intact even though they were deleted in the TRIM ON case scenario. However, this is not the case for Samsung NVMe SSDs used as primary boot devices. Recovery with AccessData FTK and Autopsy show similar results. Tables 9.6 and 9.7 give the recovery statistics from AccessData FTK and Autopsy of the different files from Samsung NVMe SSD in the TRIM ON case. Surprisingly, all the files were irrecoverable in the TRIM ON case of Seagate NVMe SSD. The Seagate controller chip acted instantly on the deleted file as soon as the files were deleted from the device. The recovery operation from AccessData FTK and Autopsy showed the same results, i.e., there was no recovery possible in the case of Seagate. Tables 9.8 and 9.9 give the recovery statistics from AccessData FTK and Autopsy of the different files from Seagate NVMe SSD in the TRIM ON case.

Table 9.6. The number of files recovered using AccessData FTK in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: Samsung FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	16792*	16792*	16792*	16792*
.key	16	16*	16*	16*	16*
.kml	192	189*	189*	189*	189*
.kmz	320	317*	317*	317*	317*
.log	1680	974*	974*	974*	974*
.mp4	64	30*	30*	30*	30*
.numbers	16	10*	10*	10*	10*
.odt	16	16*	16*	16*	16*
.pages	16	16*	16*	16*	16*
.pcap	32	1*	1*	1*	1*
.pdf	41344	40630*	40630*	40630*	40630*
.png	640	640*	640*	640*	640*
.pps	176	176*	176*	176*	176*
.ppt	9408	9406*	9406*	9406*	9406*
.pptx	16	16*	16*	16*	16*
.xls	10352	10004*	10004*	10004*	10004*
.xlsx	32	32*	32*	32*	32*
*: All files recovered but corrupted.					

Table 9.7. The number of files recovered using Autopsy in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: Samsung Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	0	0	0	0
.key	16	16*	16*	16*	16*
.kml	192	64	64	64	64
.kmz	320	317*	317*	317*	317*
.log	1680	19	19	19	19
.mp4	64	0	0	0	0
.numbers	16	0	0	0	0
.odt	16	0	0	0	0
.pages	16	0	0	0	0
.pcap	32	0	0	0	0
.pdf	41344	0	0	0	0
.png	640	0	0	0	0
.pps	176	0	0	0	0
.ppt	9408	0	0	0	0
.pptx	16	0	0	0	0
.xls	10352	0	0	0	0
.xlsx	32	0	0	0	0
*: All files recovered but corrupted.					

Table 9.8. The number of files recovered using AccessData FTK in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: Seagate FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	0	0	0	0
.key	16	0	0	0	0
.kml	192	0	0	0	0
.kmz	320	0	0	0	0
.log	1680	0	0	0	0
.mp4	64	0	0	0	0
.numbers	16	0	0	0	0
.odt	16	0	0	0	0
.pages	16	0	0	0	0
.pcap	32	0	0	0	0
.pdf	41344	0	0	0	0
.png	640	0	0	0	0
.pps	176	0	0	0	0
.ppt	9408	0	0	0	0
.pptx	16	0	0	0	0
.xls	10352	0	0	0	0
.xlsx	32	0	0	0	0
None of the files were recovered from AccessData FTK.					

Table 9.9. The number of files recovered using Autopsy in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: Seagate Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	0	0	0	0
.key	16	0	0	0	0
.kml	192	0	0	0	0
.kmz	320	0	0	0	0
.log	1680	0	0	0	0
.mp4	64	0	0	0	0
.numbers	16	0	0	0	0
.odt	16	0	0	0	0
.pages	16	0	0	0	0
.pcap	32	0	0	0	0
.pdf	41344	0	0	0	0
.png	640	0	0	0	0
.pps	176	0	0	0	0
.ppt	9408	0	0	0	0
.pptx	16	0	0	0	0
.xls	10352	0	0	0	0
.xlsx	32	0	0	0	0
None of the files were recovered from Autopsy.					

Samsung and Seagate TRIM OFF Analysis with NVMe WriteBlocker

There was a promising sign of file recovery from AccessData FTK in the TRIM OFF case of Samsung NVMe SSD as a primary boot device. All the files were recovered successfully except for .bin, .vhd, .ps2, .aff, and .csv files. However, in the case of .doc, .flv, .numbers, .odt, .pcap, .pdf, .png, .ppt remainder of the files from the original count were corrupted or zeroed out. Unfortunately, Autopsy could not recover any files even from the TRIM OFF case. Tables 9.10 and 9.11 show the statistics.

Tables 9.12 and 9.13 show the statistics of file recovery from AccessData FTK and Autopsy from Seagate NVMe SSD with NVMe WriteBlocker. The following special trend was seen from the AccessData FTK recovery process for the files below (refer to table 9.12 for statistics):

- **.csv**: Recovered all 3184 files but file size greater than 391 bytes had content zeroed out.
- **.dbase3**: Recovered all 480 files but file size greater than 418 bytes had content zeroed out.
- **.gif**: Recovered all 5952 files but 92 files were zeroed out.
- **.jpg**: Recovered all 19184 files but 114 files were zeroed out.
- **.png**: Recovered all 626 files but 14 files were zeroed out.

Controller chips of both Samsung and Seagate NVMe SSD restrict their operation, respectively, in the case of TRIM OFF. The similar behavior gave us a surety of finding data with success. However, this trend is not valid for all types of files, as the tables 9.10, 9.11, 9.12, and 9.13 below demonstrate.

Table 9.10. The number of files recovered using AccessData FTK in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: Samsung FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	480	480	480	480
.dmg	32	32	32	32	32
.doc	14592	14590*	14590*	14590*	14590*
.docx	112	112	112	112	112
.dwf	16	16	16	16	16
.e01	352	352	352	352	352
.eps	640	640	640	640	640
.f	160	160	160	160	160
.file	32	32	32	32	32
.fits	16	16	16	16	16
.flv	48	47*	47*	47*	47*
.fm	16	16	16	16	16
.gif	5952	5952	5952	5952	5952
.gls	32	32	32	32	32
.gz	2176	2176	2176	2176	2176
.hlp	112	112	112	112	112
.java	80	80	80	80	80
.jpg	19184	19184	19184	19184	19184
.key	16	16	16	16	16
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1680	1680	1680	1680
.mp4	64	64	64	64	64
.numbers	16	8*	8*	8*	8*
.odt	16	13*	13*	13*	13*
.pages	16	16	16	16	16
.pcap	32	26*	26*	26*	26*
.pdf	41344	41338*	41338*	41338*	41338*
.png	640	626*	626*	626*	626*
.pps	176	176	176	176	176
.ppt	9408	9335*	9335*	9335*	9335*
.pptx	16	16	16	16	16
.xls	10352	10327	10327	10327	10327
.xlsx	32	32	32	32	32
*: Remainder of the files got recovered but were corrupted/zeroed out.					

Table 9.11. The number of files recovered using Autopsy in Samsung NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: Samsung Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	0	0	0	0
.key	16	0	0	0	0
.kml	192	0	0	0	0
.kmz	320	0	0	0	0
.log	1680	0	0	0	0
.mp4	64	0	0	0	0
.numbers	16	0	0	0	0
.odt	16	0	0	0	0
.pages	16	0	0	0	0
.pcap	32	0	0	0	0
.pdf	41344	0	0	0	0
.png	640	0	0	0	0
.pps	176	0	0	0	0
.ppt	9408	0	0	0	0
.pptx	16	0	0	0	0
.xls	10352	0	0	0	0
.xlsx	32	0	0	0	0
None of the files were recovered from Autopsy.					

Table 9.12. The number of files recovered using AccessData FTK in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: Seagate FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	1*	1*	1*	1*
.vhd	1	1	1	1	1
.ps2	2	2	2	2	2
.aff	16	16	16	16	16
.csv	3184	3184	3184	3184	3184
.dbase	480	480	480	480	480
.dmg	32	32	32	32	32
.doc	14592	14592	14592	14592	14592
.docx	112	112	112	112	112
.dwf	16	16	16	16	16
.e01	352	352	352	352	352
.eps	640	640*	640*	640*	640*
.f	160	160	160	160	160
.file	32	32	32	32	32
.fits	16	16	16	16	16
.flv	48	48	48	48	48
.fm	16	16	16	16	16
.gif	5952	5860	5860	5860	5860
.gls	32	32	32	32	32
.gz	2176	2176	2176	2176	2176
.hlp	112	112	112	112	112
.java	80	80	80	80	80
.jpg	19184	19070	19070	19070	19070
.key	16	16	16	16	16
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1680	1680	1680	1680
.mp4	64	64	64	64	64
.numbers	16	16	16	16	16
.odt	16	16	16	16	16
.pages	16	16	16	16	16
.pcap	32	32	32	32	32
.pdf	41344	41344	41344	41344	41344
.png	640	626	626	626	626
.pps	176	176	176	176	176
.ppt	9408	9408*	9408*	9408*	9408*
.pptx	16	16	16	16	16
.xls	10352	10352*	10352*	10352*	10352*
.xlsx	32	32*	32*	32*	32*
* : Recovered all but hash of some files were different with wiped out contents.					

Table 9.13. The number of files recovered using Autopsy in Seagate NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: Seagate Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	1	1	1	1
.ps2	2	1	1	1	1
.aff	16	16*	16*	16*	16*
.csv	3184	3181*	3181*	3181*	3181*
.dbase	480	480*	480*	480*	480*
.dmg	32	32*	32*	32*	32*
.doc	14592	14591*	14591*	14591*	14591*
.docx	112	112*	112*	112*	112*
.dwf	16	16	16	16	16
.e01	352	347	347	347	347
.eps	640	640*	640*	640*	640*
.f	160	160*	160*	160*	160*
.file	32	32	32	32	32
.fits	16	16	16	16	16
.flv	48	48	48	48	48
.fm	16	16	16	16	16
.gif	5952	5871	5871	5871	5871
.gls	32	32	32	32	32
.gz	2176	2176	2176	2176	2176
.hlp	112	112	112	112	112
.java	80	80	80	80	80
.jpg	19184	19183	19183	19183	19183
.key	16	16	16	16	16
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1680	1680	1680	1680
.mp4	64	64	64	64	64
.numbers	16	16	16	16	16
.odt	16	16	16	16	16
.pages	16	16	16	16	16
.pcap	32	32	32	32	32
.pdf	41344	41344	41344	41344	41344
.png	640	627	627	627	627
.pps	176	176	176	176	176
.ppt	9408	9408	9408	9408	9408
.pptx	16	16	16	16	16
.xls	10352	10348*	10348*	10348*	10348*
.xlsx	32	32*	32*	32*	32*
* : Recovered all but hash of some files were different with wiped out contents.					

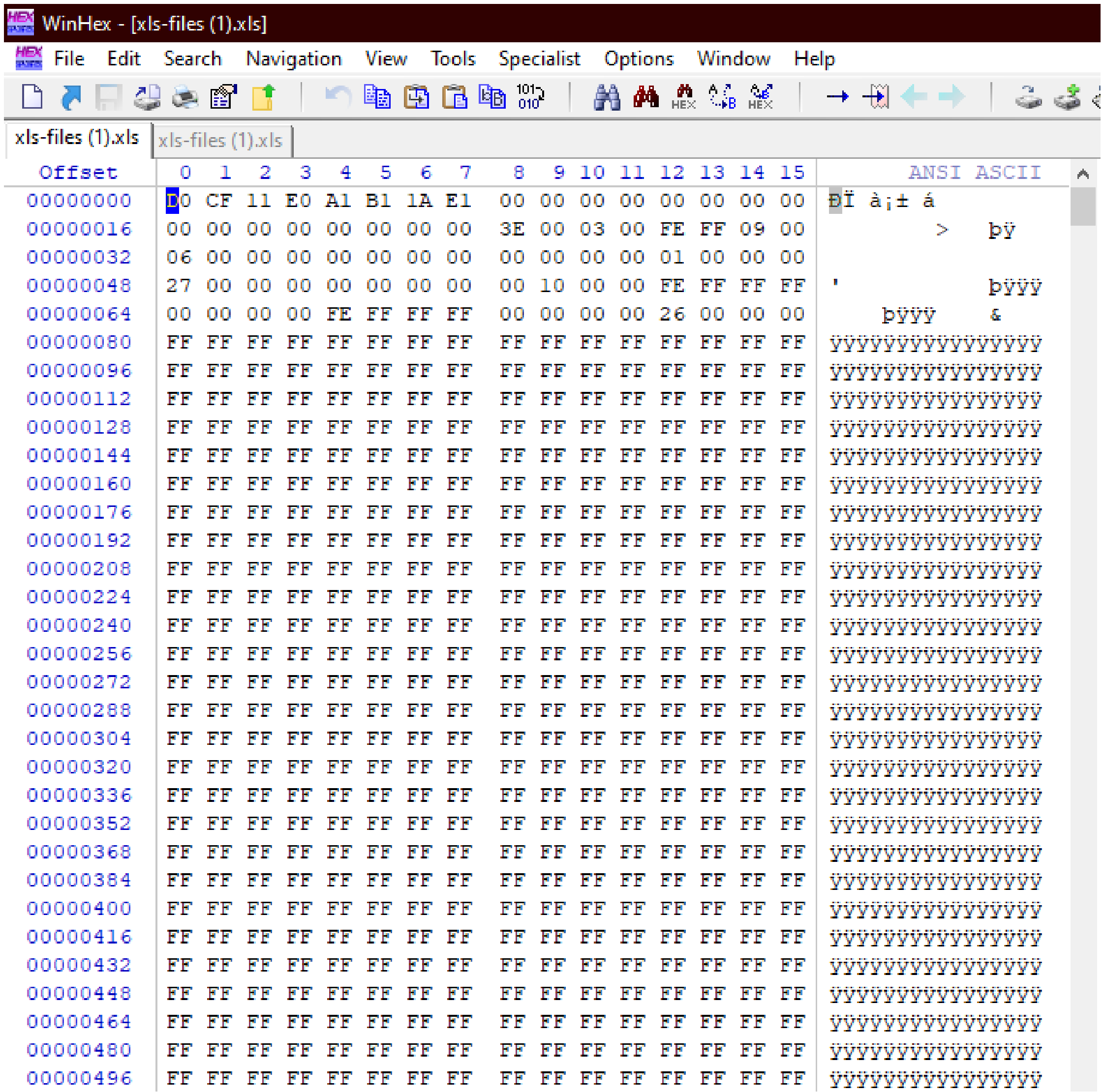


Fig. 9.6. Hexadecimal contents of xls-files(1).xls file in the original dataset from Samsung NVMe SSD.

Figure 9.6 shows a snippet of the original xls-files(1).xls file with regards to the Samsung NVMe SSD TRIM ON case in the original dataset. The hexadecimal contents are shown along with ASCII value when a file is opened in a disk editor such as WinHex. In this case, the original contents of the file are shown in the figure.

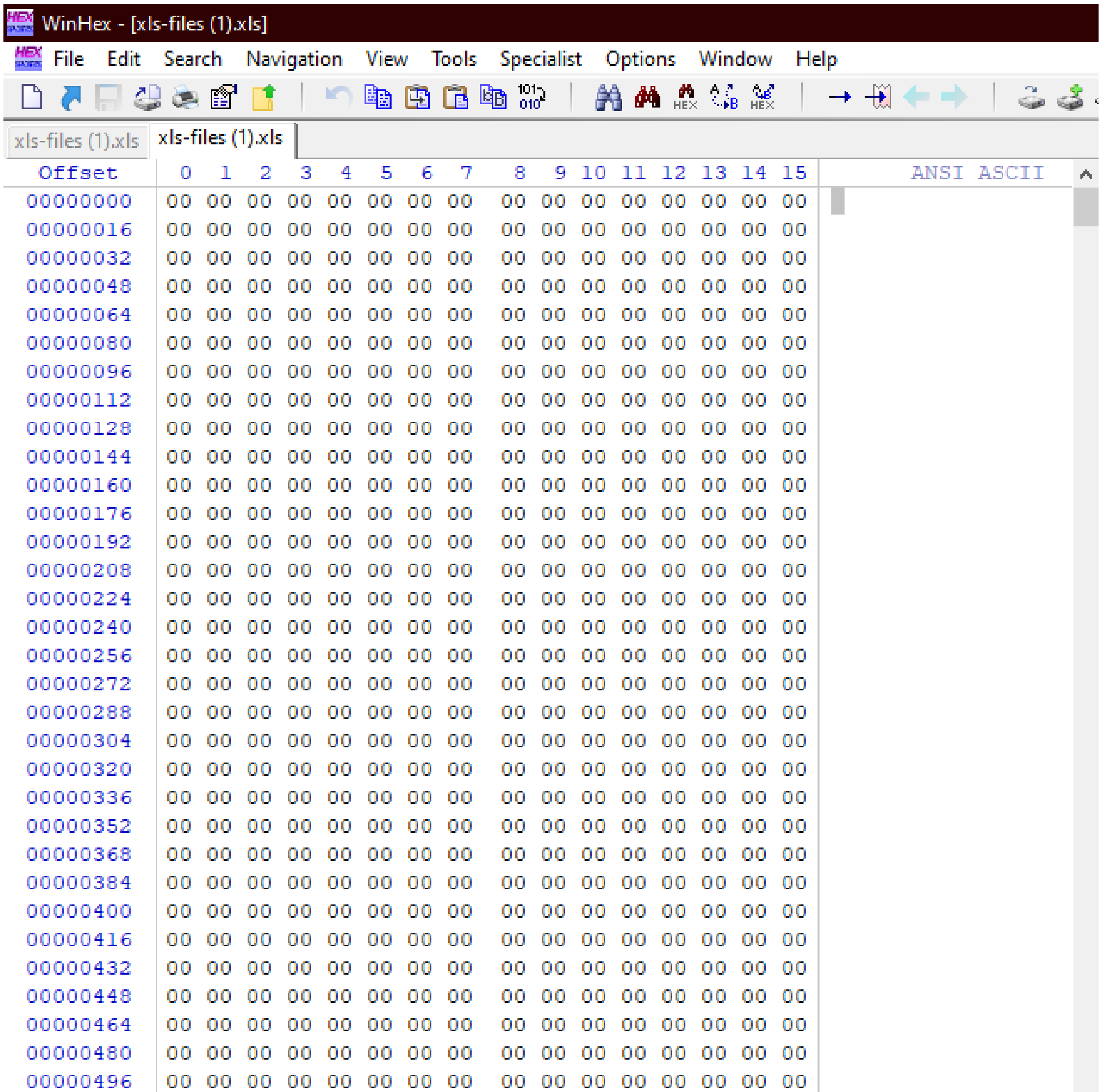


Fig. 9.7. Hexadecimal contents of xls-files(1).xls file after recovery from Samsung NVMe SSD TRIM ON case.

Figure 9.7 shows a snippet of the xls-files(1).xls file after recovery from Samsung NVMe SSD in the TRIM ON case. In this case, the file contents are wiped out for the file as shown by zeroes in the figure.

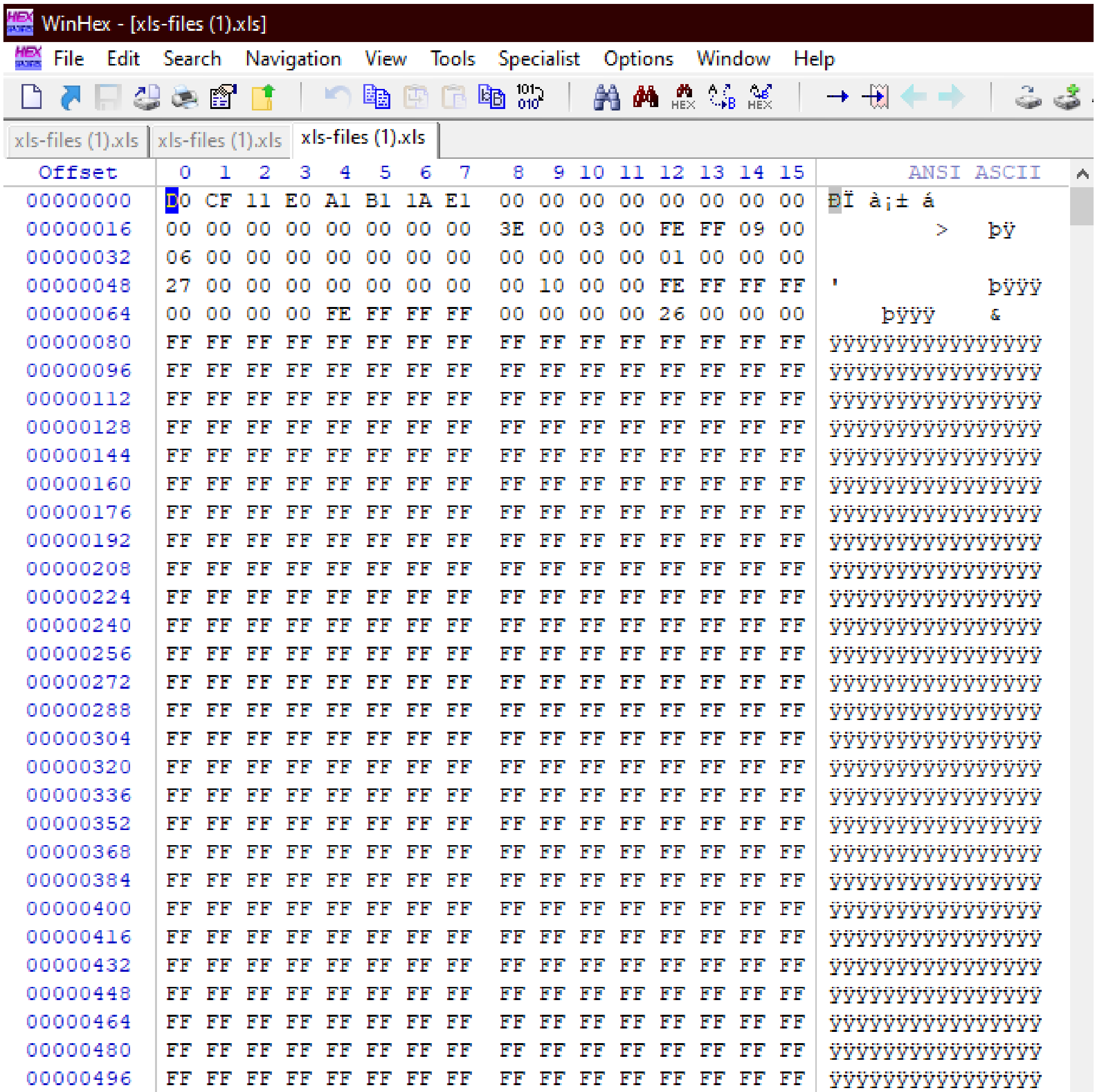


Fig. 9.8. Hexadecimal contents of xls-files(1).xls file after recovery from Samsung NVMe SSD TRIM OFF case.

Figure 9.8 shows a snippet of the xls-files(1).xls file after recovery from Samsung NVMe SSD in the TRIM OFF case. In this case, the file contents are wiped out for the file as shown by zeroes in the figure.

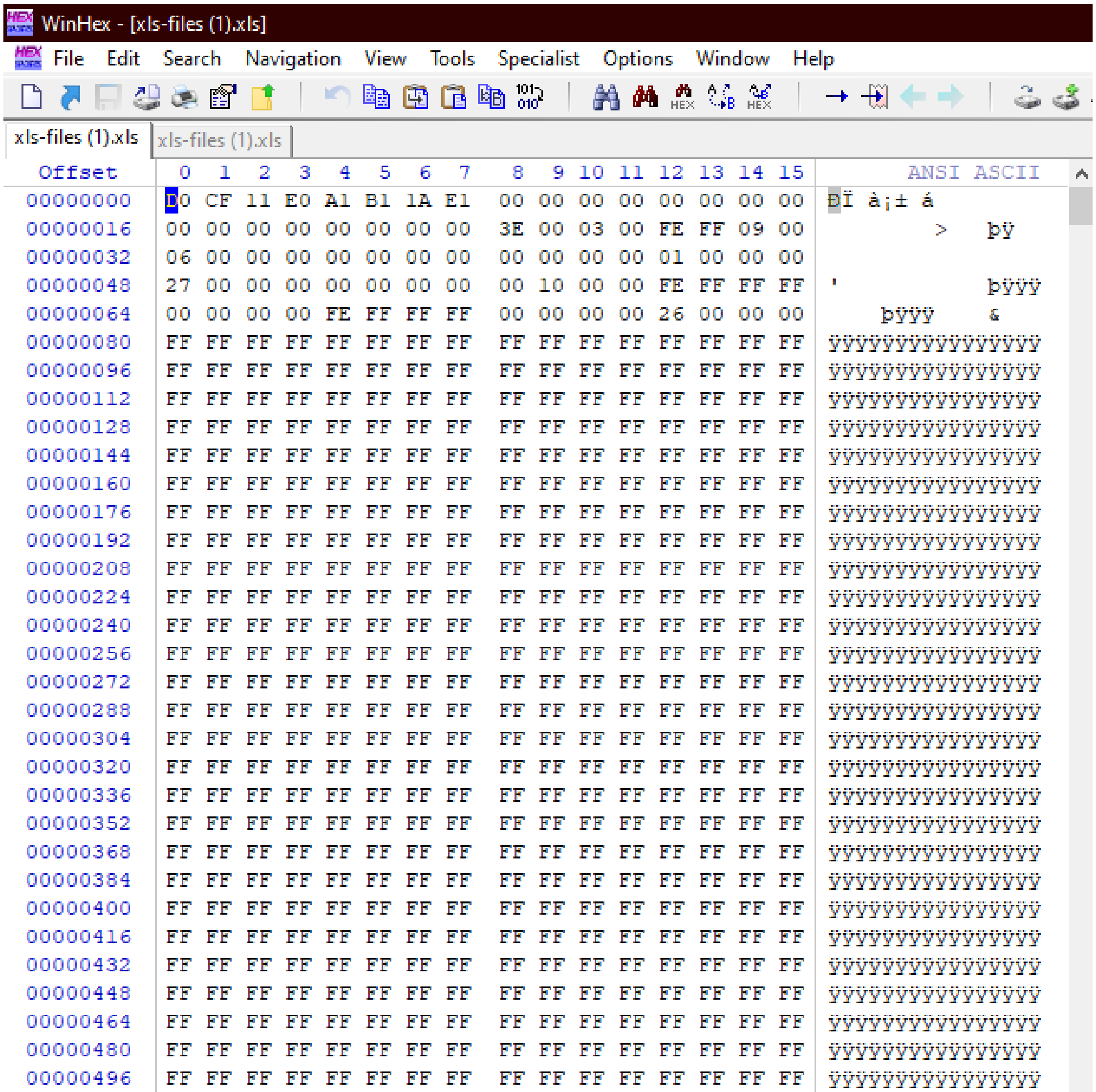


Fig. 9.9. Hexadecimal contents of xls-files(1).xls file in the original dataset from Seagate NVMe SSD.

Figure 9.9 shows a snippet of the original xls-files(1).xls file with regards to the Seagate NVMe SSD TRIM ON case in the original dataset. In this case, the original contents of the file are shown in the figure. Since there was no recovery of xls-files(1).xls from Seagate NVMe SSD, we could not show the hexadecimal contents of the recovered file.

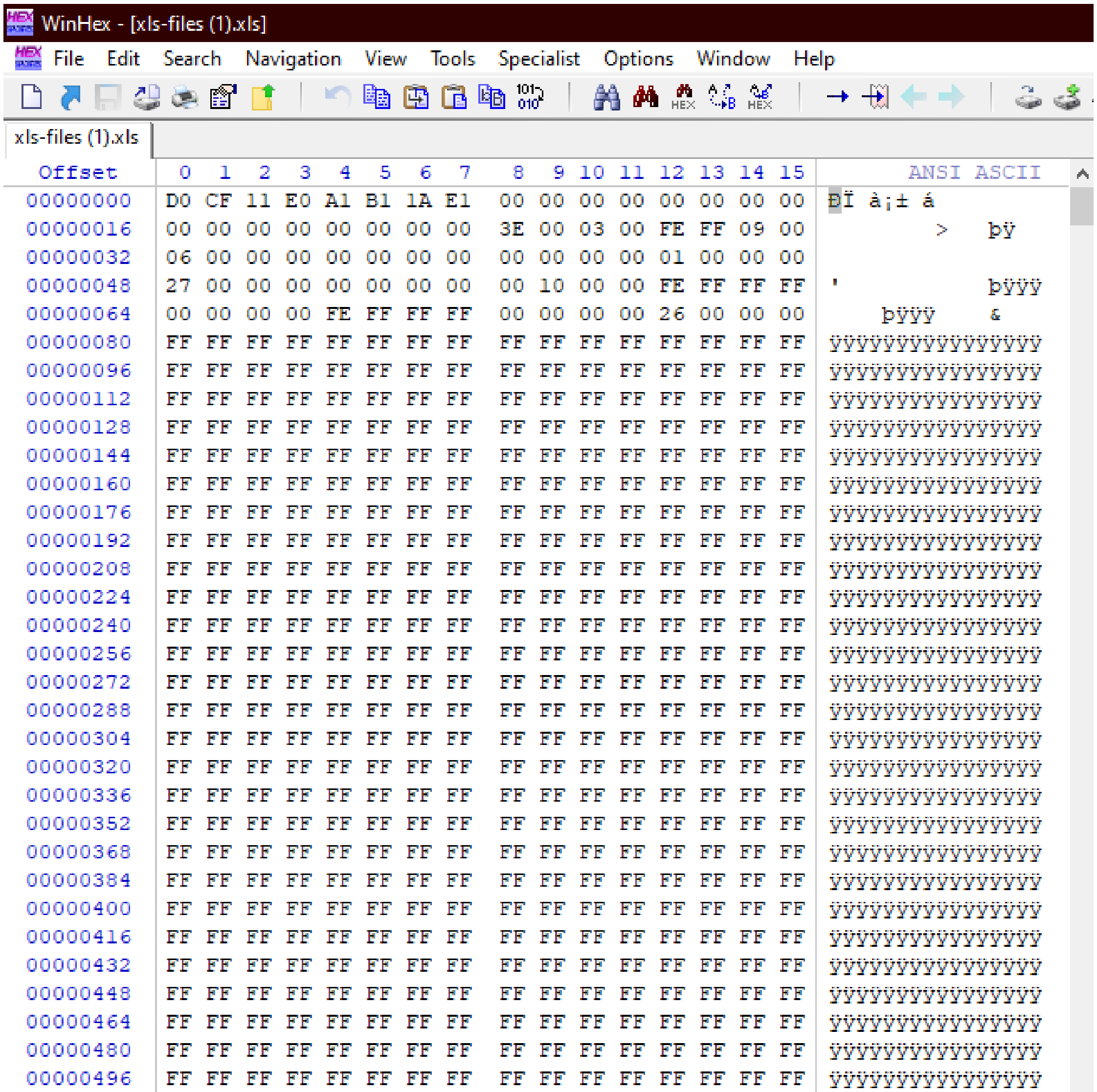


Fig. 9.10. Hexadecimal contents of xls-files(1).xls file after recovery from Seagate NVMe SSD TRIM OFF case.

Figure 9.10 shows a snippet of the xls-files(1).xls file after recovery from Seagate NVMe SSD in the TRIM OFF case. The original contents of the file are shown in the figure above.

Hash Analysis for Samsung and Seagate NVMe SSDs with NVMe WriteBlocker

In this section, we presented our findings via MD5 hash values of the files following the TRIM ON and OFF recovery operations from Samsung and Seagate NVMe SSDs. We used the QuickHash hashing tool to generate the hash values. The MD5 hash value of the original file, followed by TRIM ON and TRIM OFF MD5 hash, and the file size, for Samsung NVMe SSD, are shown in figure 9.11. However, figure 9.12 shows the hash value of the original file, followed by TRIM OFF MD5 hash, and file size in the Seagate NVMe SSD case. Since xls-files(1).xls file was not recovered in the TRIM ON case, we could not show its hash value in 9.12. The figures aim to validate and verify the claims made due to experimental observation when using an NVMe WriteBlocker.

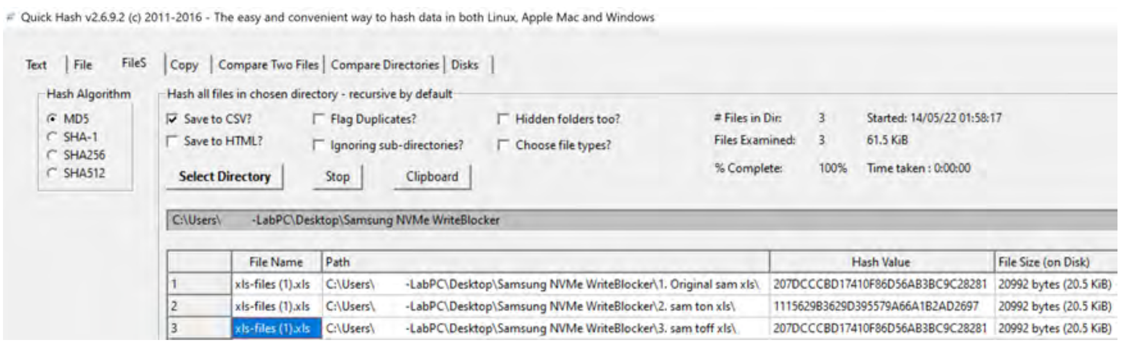


Fig. 9.11. Hash values of xls-files(1).xls in Samsung NVMe SSD when using NVMe WriteBlocker.

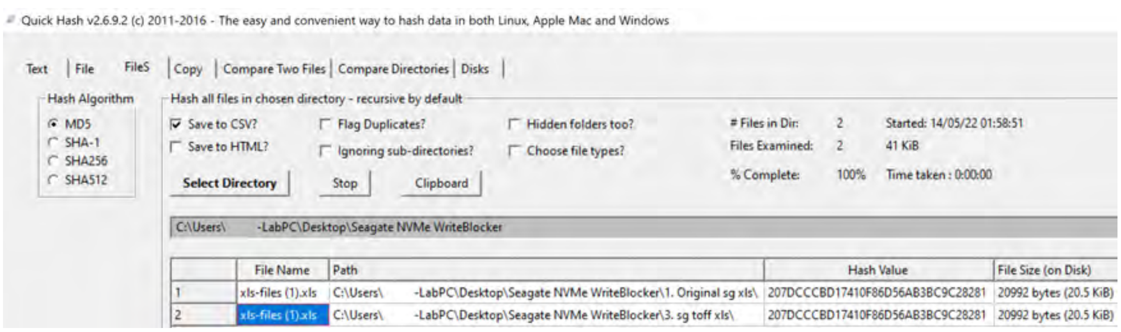


Fig. 9.12. Hash values of xls-files(1).xls in Seagate NVMe SSD when using NVMe WriteBlocker.

Table 9.14. Digital forensics information about forensically acquired image files of Samsung and Seagate NVMe SSDs with NVMe WriteBlocker.

File Names	Image Type	Image Size (KB)	MD5 Hash	SHA1 Hash
Imaging TRIM ON Samsung NVMe SSD with PCIe WriteBlocker using FTK Imager				
ton_wwb-sam_e01_pcie_img_1	e01	12 012 969	db57eed1616f5f6aac5ae9f75b1f2f33	966ce9ae480c72e96012e964b716480474241f83
ton_wwb-sam_e01_pcie_img_2	e01	12 012 969	db57eed1616f5f6aac5ae9f75b1f2f33	966ce9ae480c72e96012e964b716480474241f83
ton_wwb-sam_e01_pcie_img_3	e01	12 012 969	db57eed1616f5f6aac5ae9f75b1f2f33	966ce9ae480c72e96012e964b716480474241f83
ton_wwb-sam_e01_pcie_img_4	e01	12 012 969	db57eed1616f5f6aac5ae9f75b1f2f33	966ce9ae480c72e96012e964b716480474241f83
Imaging TRIM ON Seagate NVMe SSD with PCIe WriteBlocker using FTK Imager				
ton_wwb-sg_e01_pcie_img_1	e01	17 075 465	9e8f73e6ab6f9c135536900ee5a5a037	206bd3674088623a174a6ec46046acf0f76c1b88
ton_wwb-sg_e01_pcie_img_2	e01	17 075 465	9e8f73e6ab6f9c135536900ee5a5a037	206bd3674088623a174a6ec46046acf0f76c1b88
ton_wwb-sg_e01_pcie_img_3	e01	17 075 465	9e8f73e6ab6f9c135536900ee5a5a037	206bd3674088623a174a6ec46046acf0f76c1b88
ton_wwb-sg_e01_pcie_img_4	e01	17 075 465	9e8f73e6ab6f9c135536900ee5a5a037	206bd3674088623a174a6ec46046acf0f76c1b88
Imaging TRIM OFF Samsung NVMe SSD with PCIe WriteBlocker using FTK Imager				
toff_wwb-sam_e01_pcie_img_1	e01	125 889 025	d4152d87f93ad8fdfee2c97e1d7e7aee	fc848104d67c636cf2e32bbe2f45274391b1f631
toff_wwb-sam_e01_pcie_img_2	e01	125 889 025	d4152d87f93ad8fdfee2c97e1d7e7aee	fc848104d67c636cf2e32bbe2f45274391b1f631
toff_wwb-sam_e01_pcie_img_3	e01	125 889 025	d4152d87f93ad8fdfee2c97e1d7e7aee	fc848104d67c636cf2e32bbe2f45274391b1f631
toff_wwb-sam_e01_pcie_img_4	e01	125 889 025	d4152d87f93ad8fdfee2c97e1d7e7aee	fc848104d67c636cf2e32bbe2f45274391b1f631
Imaging TRIM OFF Seagate NVMe SSD with PCIe WriteBlocker using FTK Imager				
toff_wwb-sg_e01_pcie_img_1	e01	123 818 402	282e7fc9c54203ba40fd7264e0c16cc1	d47a9cdc320f7fe4c98039d2da92aaccb1ab2ab1
toff_wwb-sg_e01_pcie_img_2	e01	123 818 402	282e7fc9c54203ba40fd7264e0c16cc1	d47a9cdc320f7fe4c98039d2da92aaccb1ab2ab1
toff_wwb-sg_e01_pcie_img_3	e01	123 818 402	282e7fc9c54203ba40fd7264e0c16cc1	d47a9cdc320f7fe4c98039d2da92aaccb1ab2ab1
toff_wwb-sg_e01_pcie_img_4	e01	123 818 402	282e7fc9c54203ba40fd7264e0c16cc1	d47a9cdc320f7fe4c98039d2da92aaccb1ab2ab1

Western Digital and Silicon Power TRIM ON Analysis with NVMe WriteBlocker

The TRIM ON analysis of Western Digital (WD) NVMe SSD with NVMe WriteBlocker (WB) shows that none of the files could be recovered even after one day of deletion with both AccessData FTK and Autopsy tools. Tables 9.15 and 9.16 give the recovery statistics from AccessData FTK and Autopsy of the different files from Western Digital NVMe SSD in the TRIM ON case. In addition, the results of file recovery using AccessData FTK and Autopsy on Silicon Power (SP) NVMe SSD were identical. Tables 9.17 and 9.18 show the statistics from Silicon Power NVMe SSD file recovery using AccessData FTK and Autopsy tools.

The behavior of the controller chips on WD and SP NVMe SSDs exhibited unique results. There were no files recovered from Western Digital NVMe SSD using both AccessData FTK and Autopsy. However, in the case of Silicon Power, file types specifically .csv, .dbase3, .doc, .docx, .eps, .f, .file, .flv, .gif, .gz, .hlp, .jpg, .kml, .kmz, .log, .pages, .pdf, .png, .xls, .xlsx, under 12KB were intact as the controller chip did not clear them out. However, files greater than 12KB were all zeroed out.

Table 9.15. The number of files recovered using AccessData FTK in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: WD FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	0	0	0	0
.key	16	0	0	0	0
.kml	192	0	0	0	0
.kmz	320	0	0	0	0
.log	1680	0	0	0	0
.mp4	64	0	0	0	0
.numbers	16	0	0	0	0
.odt	16	0	0	0	0
.pages	16	0	0	0	0
.pcap	32	0	0	0	0
.pdf	41344	0	0	0	0
.png	640	0	0	0	0
.pps	176	0	0	0	0
.ppt	9408	0	0	0	0
.pptx	16	0	0	0	0
.xls	10352	0	0	0	0
.xlsx	32	0	0	0	0
None of the files were recovered from AccessData FTK.					

Table 9.16. The number of files recovered using Autopsy in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: WD Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	0	0	0	0
.dbase	480	0	0	0	0
.dmg	32	0	0	0	0
.doc	14592	0	0	0	0
.docx	112	0	0	0	0
.dwf	16	0	0	0	0
.e01	352	0	0	0	0
.eps	640	0	0	0	0
.f	160	0	0	0	0
.file	32	0	0	0	0
.fits	16	0	0	0	0
.flv	48	0	0	0	0
.fm	16	0	0	0	0
.gif	5952	0	0	0	0
.gls	32	0	0	0	0
.gz	2176	0	0	0	0
.hlp	112	0	0	0	0
.java	80	0	0	0	0
.jpg	19184	0	0	0	0
.key	16	0	0	0	0
.kml	192	0	0	0	0
.kmz	320	0	0	0	0
.log	1680	0	0	0	0
.mp4	64	0	0	0	0
.numbers	16	0	0	0	0
.odt	16	0	0	0	0
.pages	16	0	0	0	0
.pcap	32	0	0	0	0
.pdf	41344	0	0	0	0
.png	640	0	0	0	0
.pps	176	0	0	0	0
.ppt	9408	0	0	0	0
.pptx	16	0	0	0	0
.xls	10352	0	0	0	0
.xlsx	32	0	0	0	0
None of the files were recovered from Autopsy.					

Table 9.17. The number of files recovered using AccessData FTK in SP NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: SP FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	4*	4*	4*	4*
.vhd	1	1*	1*	1*	1*
.ps2	2	2*	2*	2*	2*
.aff	16	16*	16*	16*	16*
.csv	3184	3184	3184	3184	3184
.dbase3	480	480	480	480	480
.dmg	32	32*	32*	32*	32*
.doc	14592	14539	14539	14539	14539
.docx	112	112	112	112	112
.dwf	16	16*	16*	16*	16*
.e01	352	352*	352*	352*	352*
.eps	640	640	640	640	640
.f	160	160	160	160	160
.file	32	32	32	32	32
.fits	16	16*	16*	16*	16*
.flv	48	48	48	48	48
.fm	16	16*	16*	16*	16*
.gif	5952	5943	5943	5943	5943
.gls	32	0	0	0	0
.gz	2176	1940	1940	1940	1940
.hlp	112	112	112	112	112
.java	80	80*	80*	80*	80*
.jpg	19184	19184	19184	19184	19184
.key	16	16*	16*	16*	16*
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1676	1676	1676	1676
.mp4	64	64*	64*	64*	64*
.numbers	16	16*	16*	16*	16*
.odt	16	16*	16*	16*	16*
.pages	16	16	16	16	16
.pcap	32	32*	32*	32*	32*
.pdf	41344	41296	41296	41296	41296
.png	640	640	640	640	640
.pps	176	176*	176*	176*	176*
.ppt	9408	9408*	9408*	9408*	9408*
.pptx	16	16*	16*	16*	16*
.xls	10352	10347	10347	10347	10347
.xlsx	32	32	32	32	32
* All files were recovered from AccessData FTK but corrupted.					

Table 9.18. The number of files recovered using Autopsy in SP NVMe SSD as a primary boot device in Windows 10 TRIM ON case.

TRIM ON: SP Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	4*	4*	4*	4*
.vhd	1	1*	1*	1*	1*
.ps2	2	2*	2*	2*	2*
.aff	16	16*	16*	16*	16*
.csv	3184	3184	3184	3184	3184
.dbase3	480	480	480	480	480
.dmg	32	32*	32*	32*	32*
.doc	14592	14539	14539	14539	14539
.docx	112	112	112	112	112
.dwf	16	16*	16*	16*	16*
.e01	352	352*	352*	352*	352*
.eps	640	640	640	640	640
.f	160	160	160	160	160
.file	32	32	32	32	32
.fits	16	16*	16*	16*	16*
.flv	48	48	48	48	48
.fm	16	16*	16*	16*	16*
.gif	5952	5943	5943	5943	5943
.gls	32	0	0	0	0
.gz	2176	1940	1940	1940	1940
.hlp	112	112	112	112	112
.java	80	80*	80*	80*	80*
.jpg	19184	19184	19184	19184	19184
.key	16	16*	16*	16*	16*
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1676	1676	1676	1676
.mp4	64	64*	64*	64*	64*
.numbers	16	16*	16*	16*	16*
.odt	16	16*	16*	16*	16*
.pages	16	16	16	16	16
.pcap	32	32*	32*	32*	32*
.pdf	41344	41296	41296	41296	41296
.png	640	640	640	640	640
.pps	176	176*	176*	176*	176*
.ppt	9408	9408*	9408*	9408*	9408*
.pptx	16	16*	16*	16*	16*
.xls	10352	10347	10347	10347	10347
.xlsx	32	32	32	32	32
* All files were recovered from Autopsy but corrupted.					

Western Digital and Silicon Power TRIM OFF Analysis with NVMe WriteBlocker

In this section, we have analyzed forensics images taken using NVMe WriteBlocker in TRIM OFF cases of Western Digital (WD) and Silicon Power (SP) NVMe SSDs. The controller chips on WD and SP NVMe SSDs behaved in a distinctive way for this case. Except for a few, most of the files were recovered from Western Digital and Silicon Power devices. Tables 9.19, 9.20, 9.21, and 9.22 show the statistics of file recovery from AccessData FTK and Autopsy.

The controller chip on Western Digital NVMe SSD mostly targeted .bin, .vhd, .ps2, .aff specifically and there were no traces of recovery from AccessData FTK in all of the four forensics images. Furthermore, even though some files were fully recovered, there were found to be corrupted or content wiped out, which happened in the case of, .csv, .dbase3, .dmg, .dmp, .e01, .eps, .f, .hlp, .jpg, .png, .ppt, .xls, and .xlsx. In addition, the recovery process from Autopsy was not up to mark. The tool recovered the files, but their contents were all jumbled up, except for .gif, .jpg. and .key files.

For the controller chip of Silicon Power, the trend of recovery looked quite similar to Western Digital. Files such as .bin, .vhd, .ps2, .aff, .csv, .dbase3, .dmg, .dmp, .doc, .fits, .fm, .java, .numbers, .odt, .pages, .txt could not be said to be fully recovered as they were corrupted, after recovery from AccessData FTK. The recovery from Autopsy showed similar results as shown in the case Western Digital Autopsy recovery. File types such as .bin, .vhd, .ps2, .aff, .csv, .dmg, .dmp, .doc, .eps, .f, .fits, .fm, .jpg, .numbers, .odt, .pages, .png, .ppt, and .xls got mostly affected by the deletion process as their contents were totally jumbled even after full recovery.

Table 9.19. The number of files recovered using AccessData FTK in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: WD FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	2991*	2991*	2991*	2991*
.dbase	480	480*	480*	480*	480*
.dmg	32	32*	32*	32*	32*
.doc	14592	14592	14592	14592	14592
.docx	112	112	112	112	112
.dwf	16	16	16	16	16
.e01	352	352*	352*	352*	352*
.eps	640	640*	640*	640*	640*
.f	160	160*	160*	160*	160*
.file	32	17	17	17	17
.fits	16	16	16	16	16
.flv	48	48	48	48	48
.fm	16	16	16	16	16
.gif	5952	5952	5952	5952	5952
.gls	32	32	32	32	32
.gz	2176	2176	2176	2176	2176
.hlp	112	112*	112*	112*	112*
.java	80	80	80	80	80
.jpg	19184	19184*	19184*	19184*	19184*
.key	16	16	16	16	16
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1680	1680	1680	1680
.mp4	64	64	64	64	64
.numbers	16	16	16	16	16
.odt	16	16	16	16	16
.pages	16	16	16	16	16
.pcap	32	32	32	32	32
.pdf	41344	41344	41344	41344	41344
.png	640	640*	640*	640*	640*
.pps	176	176	176	176	176
.ppt	9408	9408*	9408*	9408*	9408*
.pptx	16	16	16	16	16
.xls	10352	10279*	10279*	10279*	10279*
.xlsx	32	32*	32*	32*	32*
*: Recovered all but some files were corrupted or contents wiped out with different hash values.					

Table 9.20. The number of files recovered using Autopsy in Western Digital NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: WD Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	0	0	0	0
.vhd	1	0	0	0	0
.ps2	2	0	0	0	0
.aff	16	0	0	0	0
.csv	3184	2991*	2991*	2991*	2991*
.dbase	480	480*	480*	480*	480*
.dmg	32	32*	32*	32*	32*
.doc	14592	14592*	14592*	14592*	14592*
.docx	112	112*	112*	112*	112*
.dwf	16	16*	16*	16*	16*
.e01	352	352*	352*	352*	352*
.eps	640	640*	640*	640*	640*
.f	160	160*	160*	160*	160*
.file	32	32*	32*	32*	32*
.fits	16	16*	16*	16*	16*
.flv	48	48*	48*	48*	48*
.fm	16	16*	16*	16*	16*
.gif	5952	5949	5949	5949	5949
.gls	32	32*	32*	32*	32*
.gz	2176	2176*	2176*	2176*	2176*
.hlp	112	112*	112*	112*	112*
.java	80	80*	80*	80*	80*
.jpg	19184	19184*	19184*	19184*	19184*
.key	16	16*	16*	16*	16*
.kml	192	192*	192*	192*	192*
.kmz	320	320*	320*	320*	320*
.log	1680	1680*	1680*	1680*	1680*
.mp4	64	64*	64*	64*	64*
.numbers	16	16*	16*	16*	16*
.odt	16	16*	16*	16*	16*
.pages	16	16*	16*	16*	16*
.pcap	32	32*	32*	32*	32*
.pdf	41344	41344*	41344*	41344*	41344*
.png	640	640*	640*	640*	640*
.pps	176	176*	176*	176*	176*
.ppt	9408	9408*	9408*	9408*	9408*
.pptx	16	16*	16*	16*	16*
.xls	10352	10352*	10352*	10352*	10352*
.xlsx	32	32*	32*	32*	32*
*: Files recovered but their contents were jumbled.					

Table 9.21. The number of files recovered using AccessData FTK in Silicon Power NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: SP FTK Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	3*	3*	3*	3*
.vhd	1	1*	1*	1*	1*
.ps2	2	2*	2*	2*	2*
.aff	16	16*	16*	16*	16*
.csv	3184	3184*	3184*	3184*	3184*
.dbase	480	480*	480*	480*	480*
.dmg	32	32*	32*	32*	32*
.doc	14592	14590*	14590*	14590*	14590*
.docx	112	112	112	112	112
.dwf	16	16	16	16	16
.e01	352	352	352	352	352
.eps	640	640	640	640	640
.f	160	160	160	160	160
.file	32	32	32	32	32
.fits	16	16*	16*	16*	16*
.flv	48	48	48	48	48
.fm	16	16	16	16	16
.gif	5952	5952	5952	5952	5952
.gls	32	32	32	32	32
.gz	2176	2176	2176	2176	2176
.hlp	112	112	112	112	112
.java	80	80	80	80	80
.jpg	19184	19184	19184	19184	19184
.key	16	16	16	16	16
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1680	1680	1680	1680
.mp4	64	64	64	64	64
.numbers	16	16*	16*	16*	16*
.odt	16	16*	16*	16*	16*
.pages	16	16*	16*	16*	16*
.pcap	32	32	32	32	32
.pdf	41344	41344	41344	41344	41344
.png	640	626	626	626	626
.pps	176	176*	176*	176*	176*
.ppt	9408	9403	9403	9403	9403
.pptx	16	16*	16*	16*	16*
.xls	10352	10352	10352	10352	10352
.xlsx	32	32	32	32	32
* : Recovered all but some files were corrupted or contents wiped out with different hash values					

Table 9.22. The number of files recovered using Autopsy in Silicon Power NVMe SSD as a primary boot device in Windows 10 TRIM OFF case.

TRIM OFF: SP Autopsy Statistics in Windows 10 with NVMe WB					
File Type	Original Image	Image-1	Image-2	Image-3	Image-4
.bin	4	4*	4*	4*	4*
.vhd	1	1*	1*	1*	1*
.ps2	2	2*	2*	2*	2*
.aff	16	16*	16*	16*	16*
.csv	3184	3184*	3184*	3184*	3184*
.dbase	480	480	480	480	480
.dmg	32	32*	32*	32*	32*
.doc	14592	14592*	14592*	14592*	14592*
.docx	112	112	112	112	112
.dwf	16	16	16	16	16
.e01	352	352	352	352	352
.eps	640	640*	640*	640*	640*
.f	160	160*	160*	160*	160*
.file	32	32	32	32	32
.fits	16	16	16	16	16
.flv	48	48	48	48	48
.fm	16	16*	16*	16*	16*
.gif	5952	5949	5949	5949	5949
.gls	32	32	32	32	32
.gz	2176	2176	2176	2176	2176
.hlp	112	112	112	112	112
.java	80	80	80	80	80
.jpg	19184	19184*	19184*	19184*	19184*
.key	16	16	16	16	16
.kml	192	192	192	192	192
.kmz	320	320	320	320	320
.log	1680	1680	1680	1680	1680
.mp4	64	64	64	64	64
.numbers	16	16*	16*	16*	16*
.odt	16	16*	16*	16*	16*
.pages	16	16*	16*	16*	16*
.pcap	32	32	32	32	32
.pdf	41344	41344	41344	41344	41344
.png	640	621	621	621	621
.pps	176	176	176	176	176
.ppt	9408	9403*	9403*	9403*	9403*
.pptx	16	16	16	16	16
.xls	10352	10352	10352	10352	10352
.xlsx	32	32	32	32	32
* : Files recovered but their contents were jumbled or wiped out.					

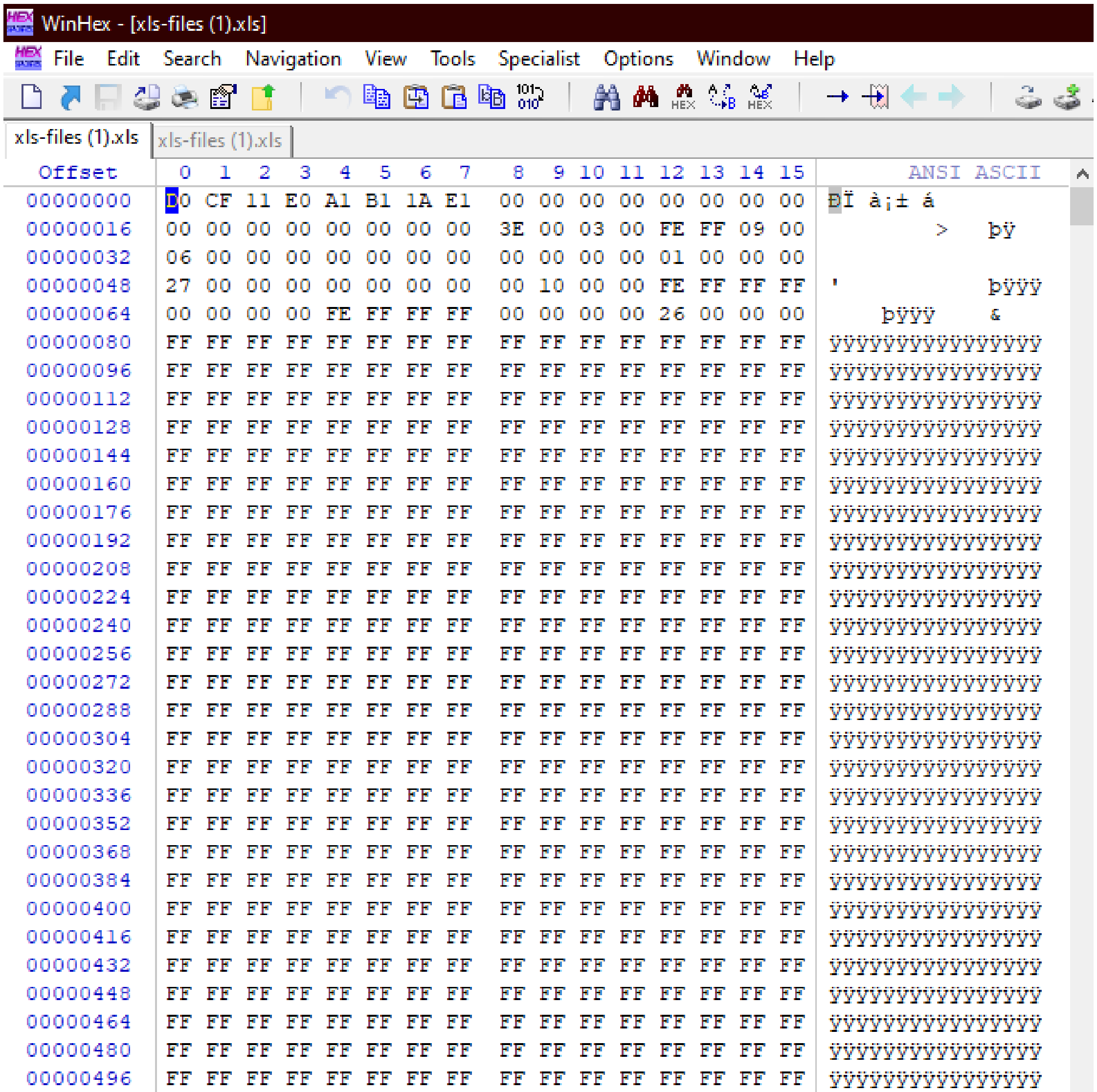


Fig. 9.13. Hexadecimal contents of xls-files(1).xls file in the original dataset from Western Digital NVMe SSD.

Figure 9.13 shows a snippet of the original xls-files(1).xls file with regards to the Western Digital NVMe SSD TRIM ON case in the original dataset. The hexadecimal contents are shown along with ASCII value when a file is opened in a disk editor such as WinHex. In this case, original contents of the file are shown in the figure.

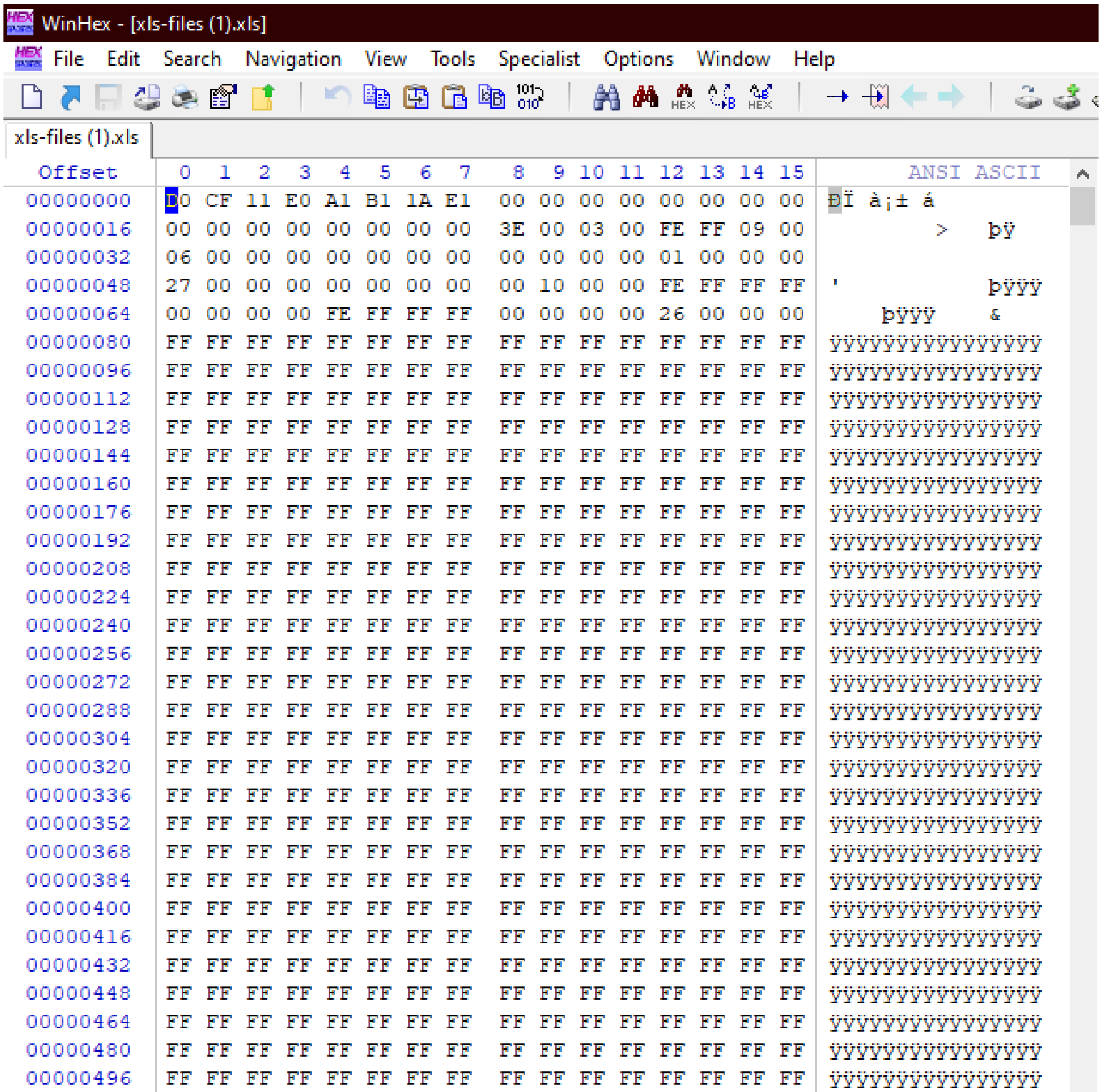


Fig. 9.14. Hexadecimal contents of xls-files(1).xls file after recovery from Western Digital NVMe SSD TRIM OFF case.

Figure 9.14 shows a snippet of the xls-files(1).xls file after recovery from Western Digital NVMe SSD in the TRIM OFF case. In this case, the file contents were not wiped out for the file as shown in the figure.

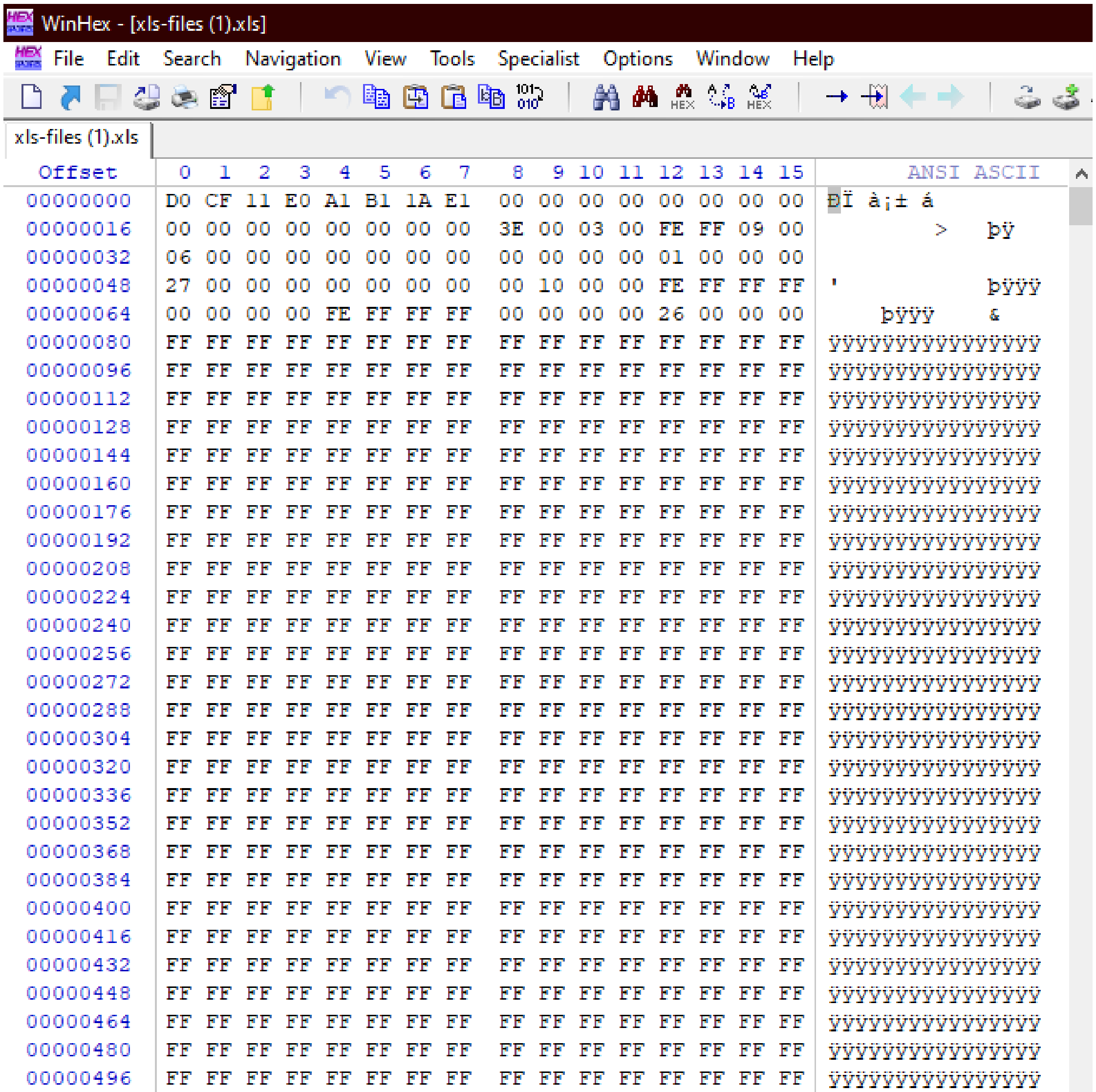


Fig. 9.15. Hexadecimal contents of xls-files(1).xls file in the original dataset from Silicon Power NVMe SSD.

Figure 9.15 shows a snippet of the original xls-files(1).xls file with regards to the Silicon Power NVMe SSD TRIM ON case in the original dataset. The hexadecimal contents are shown along with ASCII value when a file is opened in a disk editor such as WinHex. In this case, original contents of the file are shown in the figure.

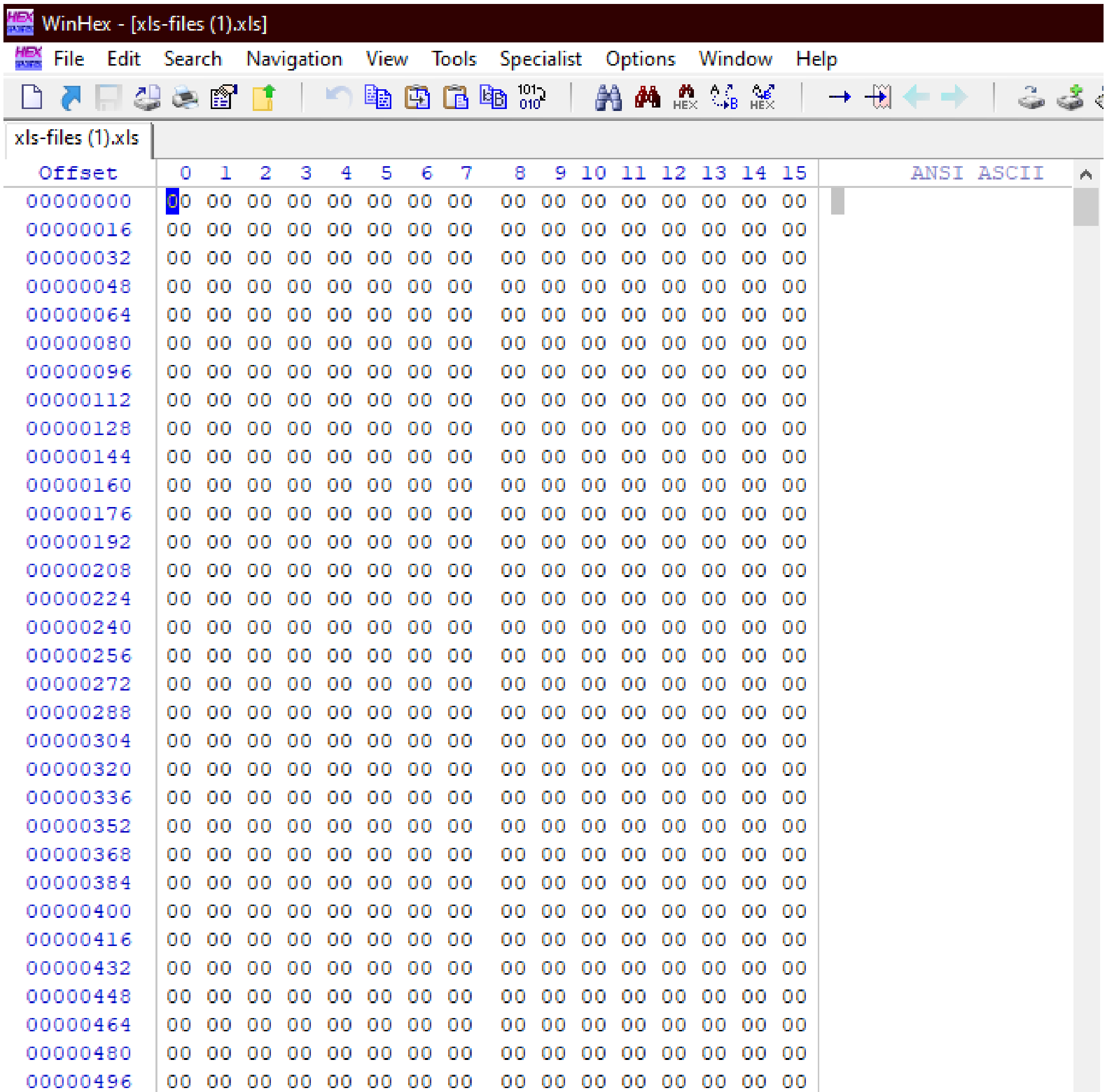


Fig. 9.16. Hexadecimal contents of xls-files(1).xls file after recovery from Silicon Power NVMe SSD TRIM ON case.

Figure 9.16 shows a snippet of the xls-files(1).xls file after recovery from Silicon Power NVMe SSD in the TRIM ON case. In this case, the file contents were wiped out for the file as shown by zeroes in the figure.

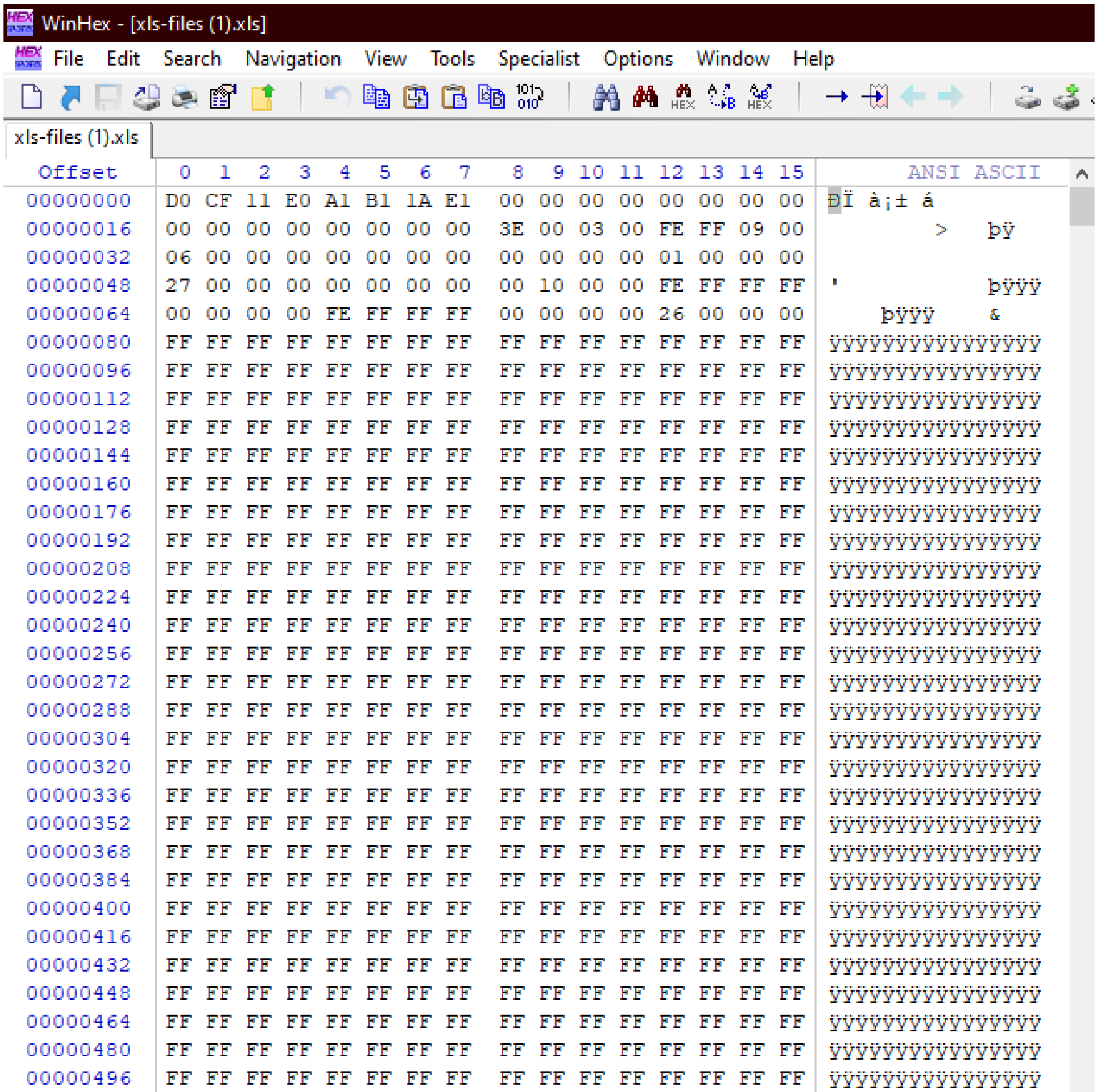


Fig. 9.17. Hexadecimal contents of xls-files(1).xls file after recovery from Silicon Power NVMe SSD TRIM OFF case.

Figure 9.17 shows a snippet of the xls-files(1).xls file after recovery from Silicon Power NVMe SSD in the TRIM OFF case. The original contents of the file are shown in the figure.

Hash Analysis for Western Digital and Silicon Power NVMe SSDs with NVMe WriteBlocker

The MD5 hash values of the files following the TRIM ON and OFF recovery operations from Western Digital and Silicon Power NVMe SSDs are displayed in this part to demonstrate our findings. We used the QuickHash hashing tool to generate hash values. The MD5 hash value of the original file, followed by TRIM OFF MD5 hash, and the file size for Western Digital NVMe SSD, are shown in figure 9.18. Unfortunately, we could not show the TRIM ON hash value due to the absence of recovery of xls-files(1).xls file. However, figure 9.19 shows the hash values of the original file, followed by TRIM ON and OFF MD5 hash values and file size in the Silicon Power NVMe SSD case as shown in figure 9.19. These figures aim to validate and verify the claims made due to experimental observation when using an NVMe WriteBlocker.

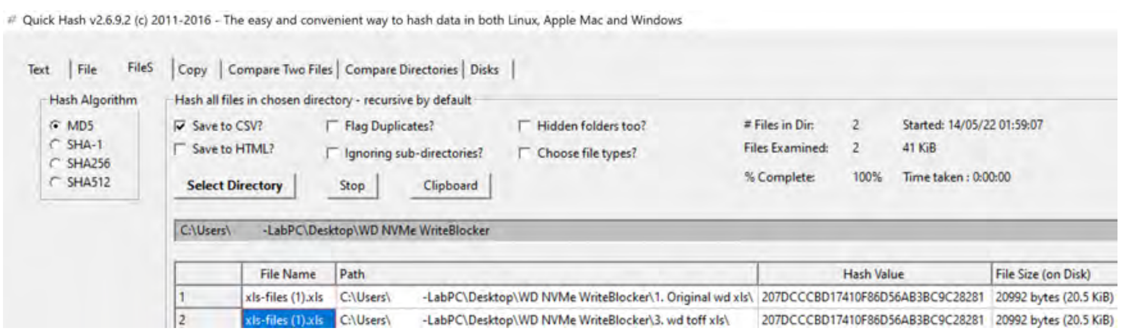


Fig. 9.18. Hash of xls-files(1).xls in Western Digital NVMe SSD using NVMe WriteBlocker.

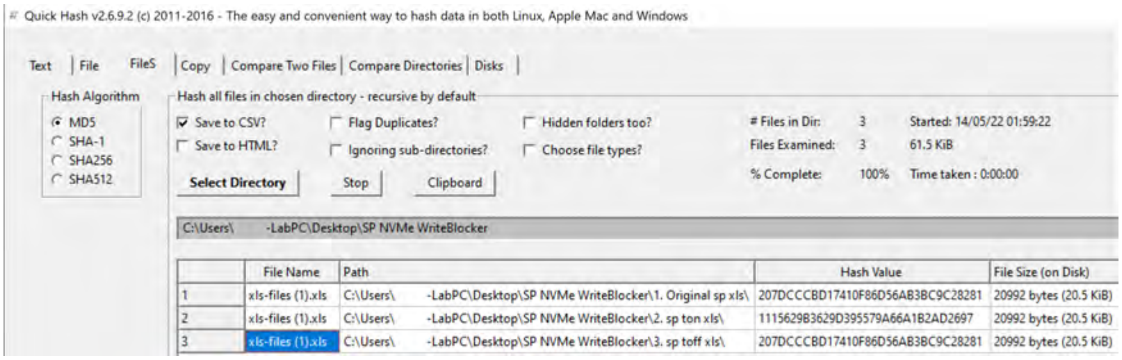


Fig. 9.19. Hash of xls-files(1).xls in Silicon Power NVMe SSD using NVMe WriteBlocker.

Table 9.23. Digital forensics information about forensically acquired image files of Western Digital and Silicon Power NVMe SSDs with NVMe WriteBlocker.

File Names	Image Type	Image Size (KB)	MD5 Hash	SHA1 Hash
Imaging TRIM ON Western Digital NVMe SSD with PCIe WriteBlocker using FTK Imager				
ton_wwb-wd_e01_pcie_img_1	e01	19 373 246	e612c339d9b3001c18b13a8ba3250093	77521c6fdf0767be9b9840b9517438d58e015828
ton_wwb-wd_e01_pcie_img_2	e01	19 373 246	e612c339d9b3001c18b13a8ba3250093	77521c6fdf0767be9b9840b9517438d58e015828
ton_wwb-wd_e01_pcie_img_3	e01	19 373 246	e612c339d9b3001c18b13a8ba3250093	77521c6fdf0767be9b9840b9517438d58e015828
ton_wwb-wd_e01_pcie_img_4	e01	19 373 246	e612c339d9b3001c18b13a8ba3250093	77521c6fdf0767be9b9840b9517438d58e015828
Imaging TRIM ON Silicon Power NVMe SSD with PCIe WriteBlocker using FTK Imager				
ton_wwb-sp_e01_pcie_img_1	e01	16 531 698	14d8b304d966ac894322e359f33cd601	6a2ed6ea5d7d42554dc2f050d72816bbe9ed18d3
ton_wwb-sp_e01_pcie_img_2	e01	16 531 698	14d8b304d966ac894322e359f33cd601	6a2ed6ea5d7d42554dc2f050d72816bbe9ed18d3
ton_wwb-sp_e01_pcie_img_3	e01	16 531 698	14d8b304d966ac894322e359f33cd601	6a2ed6ea5d7d42554dc2f050d72816bbe9ed18d3
ton_wwb-sp_e01_pcie_img_4	e01	16 531 698	14d8b304d966ac894322e359f33cd601	6a2ed6ea5d7d42554dc2f050d72816bbe9ed18d3
Imaging TRIM OFF Western Digital NVMe SSD with PCIe WriteBlocker using FTK Imager				
toff_wwb-wd_e01_pcie_img_1	e01	125 161 110	025181d55629d0876c881b479c0be4cf	74b4f195df8faf66997b28d30644018baa048396
toff_wwb-wd_e01_pcie_img_2	e01	125 161 110	025181d55629d0876c881b479c0be4cf	74b4f195df8faf66997b28d30644018baa048396
toff_wwb-wd_e01_pcie_img_3	e01	125 161 110	025181d55629d0876c881b479c0be4cf	74b4f195df8faf66997b28d30644018baa048396
toff_wwb-wd_e01_pcie_img_4	e01	125 161 110	025181d55629d0876c881b479c0be4cf	74b4f195df8faf66997b28d30644018baa048396
Imaging TRIM OFF Silicon Power NVMe SSD with PCIe WriteBlocker using FTK Imager				
toff_wwb-sp_e01_pcie_img_1	e01	124 938 768	59ec02930b9df63922d4396c4509c00d	5ccf99e78bcb89d07d921ee5c874ce4422536bca
toff_wwb-sp_e01_pcie_img_2	e01	124 938 768	59ec02930b9df63922d4396c4509c00d	5ccf99e78bcb89d07d921ee5c874ce4422536bca
toff_wwb-sp_e01_pcie_img_3	e01	124 938 768	59ec02930b9df63922d4396c4509c00d	5ccf99e78bcb89d07d921ee5c874ce4422536bca
toff_wwb-sp_e01_pcie_img_4	e01	124 938 768	59ec02930b9df63922d4396c4509c00d	5ccf99e78bcb89d07d921ee5c874ce4422536bca

CHAPTER X

NVMe-Assist PyTsk Codes

In this chapter, we have explained and defined the Python libraries used to develop the NVMe-Assist toolkit. A library in computer programming is a collection of files, programs, routines, scripts, or procedures that can be referenced in the code. The library is a collection of pre-written code that users can utilize to speed up their work. We have used the following Python libraries to achieve our task. Figures 10.1 to 10.11 exhibit the working of our NVMe-Assist framework toolkit. Also, we have hosted our code on GitHub link: https://github.com/asharneyaz/nvme-assist

1. Python ***os*** library: This module allows you to use operating system-dependent functions on the go. If you only want to read or write a file, use open(), the os.path module if you want to change paths, and the fileinput module if you want to read all the lines in all the files on the command line. The tempfile module can be used to create temporary files and directories, and the shutil module can be used to handle high-level file and directory operations.
2. Python ***sys*** library: This module gives you access to some variables that the interpreter uses or maintains, as well as functions that have a lot of interaction with it. It is available at all times.
3. Python ***pytsk3*** library: This is a Python binding for the libtsk library (SleuthKit library). The goal is to make the binding as close to the TSK API as possible in terms of capabilities, while still providing a pleasant Pythonic interface.
4. Python ***datetime*** library: The datetime module supplies classes for manipulating dates and times. While date and time arithmetic is

supported, the focus of the implementation is on efficient attribute extraction for output formatting and manipulation.

5. Python ***hashlib*** library: This module provides a standardised interface to a variety of secure hash and message digest methods. The SHA1, SHA224, SHA256, SHA384, and SHA512 secure hash algorithms (specified in FIPS 180-2) are included, as well as RSA's MD5 algorithm (defined in internet RFC 1321). The phrases "message digest" and "secure hash" are synonymous. Message digests were the name for older algorithms. Secure hash is the modern phrase for it.
6. Python ***itertools*** library: This module implements a set of iterator building pieces based on APL, Haskell, and SML principles. Each has been recast in a Python-friendly format. The module standardizes a core collection of quick, memory-efficient utilities that can be used alone or in tandem. They constitute a "iterator algebra" when combined, making it easy to build specialized tools in pure Python quickly and effectively.
7. Python ***tabulate*** library: tabulate is a module that allows you to present table data in a visually appealing manner. Because tabulate is not included in the standard Python library, it must be installed separately.
8. Python ***pyfiglet*** library: pyfiglet transforms ASCII text into ASCII art fonts. ASCII text is converted to ASCII art fonts using the **figlet_format** technique.
9. Python ***art*** library: The art package is used to print attractive art on the display.
10. Python ***pathlib*** library: This module contains classes that represent filesystem paths and provide semantics for various operating systems. Pure paths, which allow purely computational operations without I/O, and concrete paths, which inherit from pure pathways but additionally provide I/O operations, are the two types of path classes.
11. Python ***simple-colors*** library: Exhibits colorful output in terminal.

Algorithm 1 NVMe-Assist Toolkit Algorithm

Requirement: User runs the NVMe-Assist code using **python nvme_df1.py**

```
if no. of arguments is two and the first argument is either –help or -h or /? then
    print manual page of NVMe-Assist Toolkit.
else proceed with code execution
    check for operating system (os) family: Linux, Windows, macOS then
        if os is Linux then
            clear the screen, print NVMe-Assist banner, and Linux machine
        if os is Apple macOS then
            clear the screen, print NVMe-Assist banner, and macOS machine
        if os is Windows then
            clear the screen, print NVMe-Assist banner, and Windows machine
        User chooses the file path then
        Change the default location to file path then
        List the contents of the file path then
        Choose forensics image between: .dd/.raw/.img/.001/.e01 then
        Show the file chosen to the user then
            if file chosen is .dd/.raw/.img/.001 then
                call nvme_df2.py then
                    print modified and created times of the image file then
                    print the partition scheme from the image file then
                    print the partition table from the image file then
                    print the MD5 and SHA1 hashes of the image file then
            else file chosen is .e01 then
                call nvme_df3.py then
                    print modified and created times of the image file then
                    print the partition scheme from the image file then
                    print the partition table from the image file then
                    print the MD5 and SHA1 hashes of the image file then
                    Ask the user for program continuation: option of Y or N
            end if
end if
```

Algorithm 2 GPT Sector Parser Algorithm

Requirement: User runs the GPT sector parser code using **python nvme_df4_gpt_sector_parser.py**

if *no. of arguments is two and the first argument is either –help or -h or /?* **then**
    print *manual page of GPT Sector Parser Toolkit.*
else *proceed with code execution*
    check for operating system (os) family: Linux, Windows, macOS **then**
        if *os is Linux* **then**
          clear the screen, print NVMe-Assist banner, and Linux machine
        if *os is Apple macOS* **then**
          clear the screen, print NVMe-Assist banner, and macOS machine
        if *os is Windows* **then**
          clear the screen, print NVMe-Assist banner, and Windows machine
        User chooses the file path **then**
        Change the default location to file path **then**
        List the contents of the file path **then**
        Choose forensics image between: ***.dd/.raw/.img/.001/.e01*** **then**
        Show the file chosen to the user **then**
            if *file chosen is .dd/.raw/.img/.001* **then**
                call **nvme_df5_gpt_partition_parser.py then**
                  print basic information of the image file **then**
                  print the partition scheme from the image file **then**
                  print the total partitions present from the image file **then**
                  print the GPT Header from the image file **then**
                  print the GPT partition table from the image file **then**
                  Ask the user for program continuation: option of Y or N
            end if
end if

Algorithm 3 Logical Partition OEM Checker

Requirement: User runs the GPT sector parser code using **python nvme_df4_gpt_sector_parser.py**

```
if User chooses the file path then
    Change the default location to file path then
    List the contents of the file path then
    Choose logical partition forensics image between: .dd/.raw/.img/.001/.e01 then
    Show the file chosen to the user then
        if file chosen is .dd/.raw/.img/.001 then
            read the imagefile as binary file then
            read the three bytes of the jump instructions from the imagefile then
            read the eight bytes of the OEM from the imagefile then
            print the OEM identifier of the partition then
            decode the hexadecimal value to utf-8 encoding from the imagefile then
                if decoded value == "4E54465320202020" then
                    print NTFS partition.
                else if decoded value == "4D5357494E342E31" then
                    print FAT-16 partition.
                else if decoded value == "4D53444F53352E30" then
                    print FAT-32 partition.
                end if
        Ask the user for program continuation: option of Y or N
        end if
end if
```

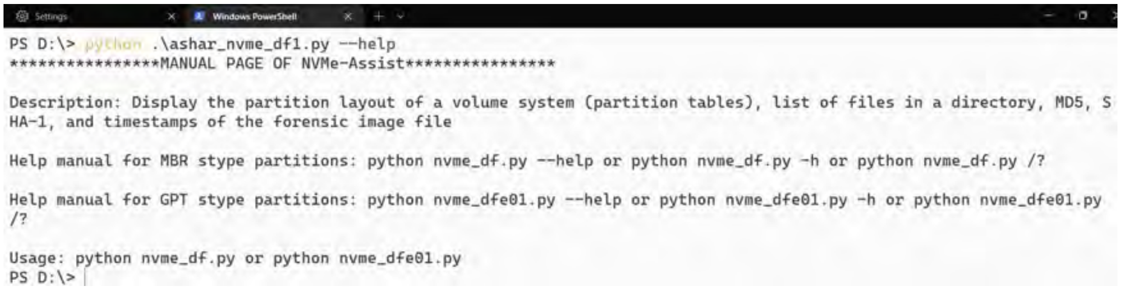


Fig. 10.1. Manual page of NVMe-Assist Toolkit.

Fig. 10.2. Running demonstration of NVMe-Assist step-1.

Fig. 10.3. Running demonstration of NVMe-Assist step-2.

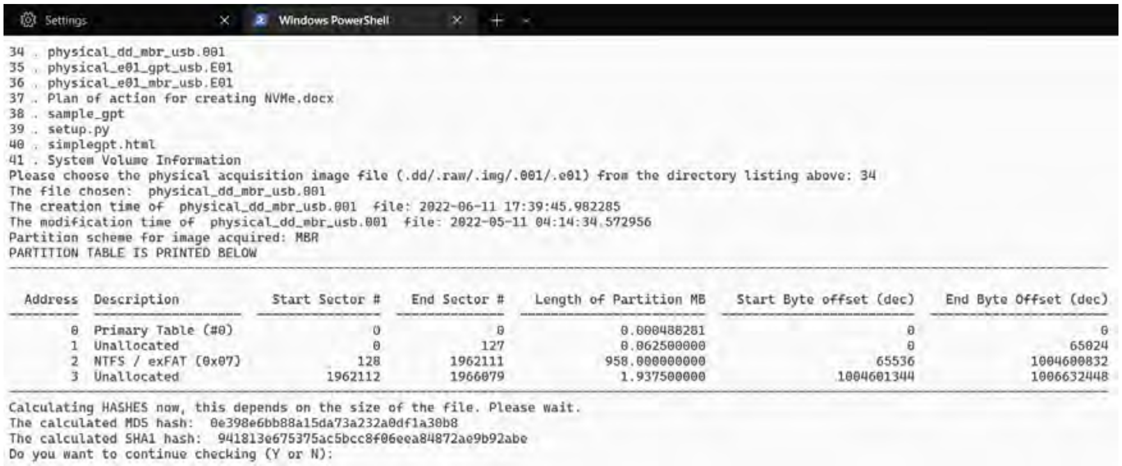


Fig. 10.4. Running demonstration of NVMe-Assist step-3.

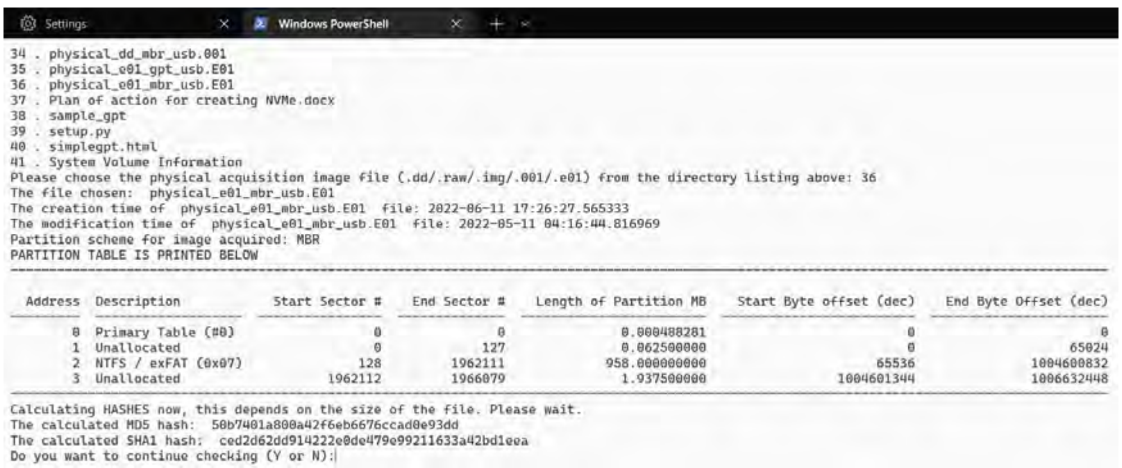
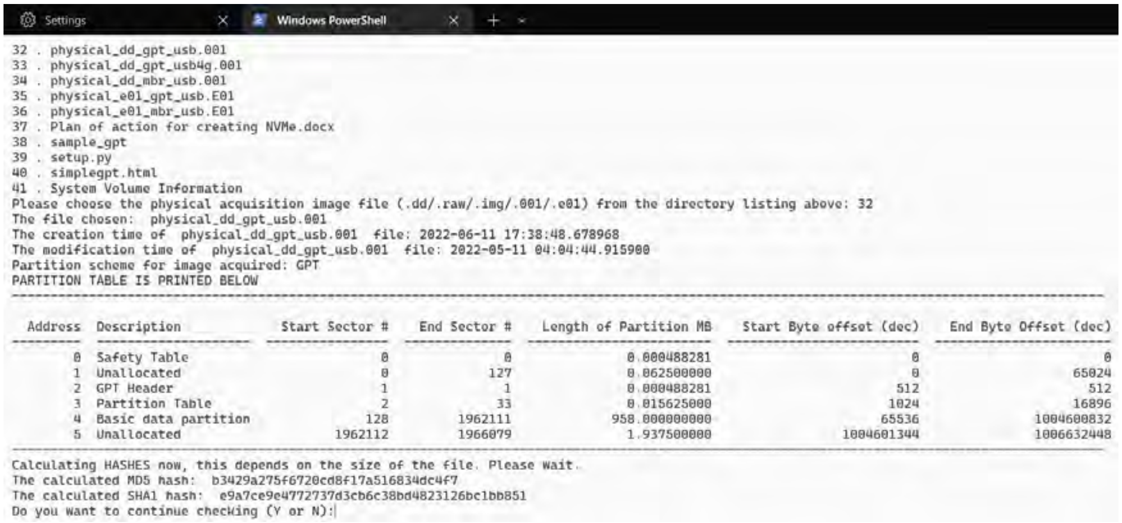


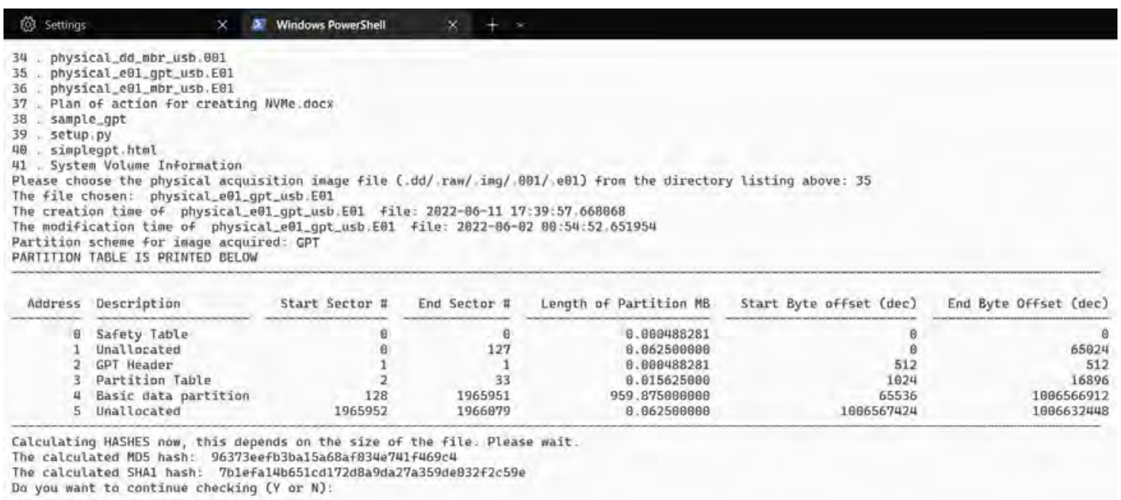
Fig. 10.5. Running demonstration of NVMe-Assist step-4.



```
32 . physical_dd_gpt_usb.001
33 . physical_dd_gpt_usb4g.001
34 . physical_dd_mbr_usb.001
35 . physical_e01_gpt_usb.E01
36 . physical_e01_mbr_usb.E01
37 . Plan of action for creating NVMe.docx
38 . sample_gpt
39 . setup.py
40 . simplegpt.html
41 . System Volume Information
Please choose the physical acquisition image file (.dd/.raw/.img/.001/.e01) from the directory listing above: 32
The file chosen:  physical_dd_gpt_usb.001
The creation time of  physical_dd_gpt_usb.001  file: 2022-06-11 17:38:48.678968
The modification time of  physical_dd_gpt_usb.001  file: 2022-05-11 04:04:44.915900
Partition scheme for image acquired: GPT
PARTITION TABLE IS PRINTED BELOW
--------------------------------------------------------------------------------------------------------------------------------

 Address  Description            Start Sector #   End Sector #   Length of Partition MB   Start Byte offset (dec)   End Byte Offset (dec)
 -------  ---------------------  ---------------  -------------  -----------------------  ------------------------  ---------------------
       0  Safety Table                         0              0              0.000488281                         0                      0
       1  Unallocated                          0            127              0.062500000                         0                  65024
       2  GPT Header                           1              1              0.000488281                       512                    512
       3  Partition Table                      2             33              0.015625000                      1024                  16896
       4  Basic data partition               128        1962111            958.000000000                     65536             1004600832
       5  Unallocated                    1962112        1966079              1.937500000                1004601344             1006632448
--------------------------------------------------------------------------------------------------------------------------------
Calculating HASHES now, this depends on the size of the file. Please wait.
The calculated MD5 hash:  b3429a275f6720cd8f17a516834dc4f7
The calculated SHA1 hash:  e9a7ce9e4772737d3cb6c38bd4823126bc1bb851
Do you want to continue checking (Y or N):
```

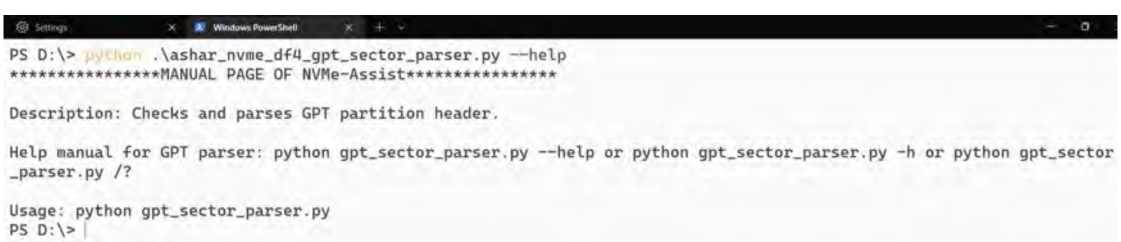
Fig. 10.6. Running demonstration of NVMe-Assist step-5.



```
34 . physical_dd_mbr_usb.001
35 . physical_e01_gpt_usb.E01
36 . physical_e01_mbr_usb.E01
37 . Plan of action for creating NVMe.docx
38 . sample_gpt
39 . setup.py
40 . simplegpt.html
41 . System Volume Information
Please choose the physical acquisition image file (.dd/.raw/.img/.001/.e01) from the directory listing above: 35
The file chosen:  physical_e01_gpt_usb.E01
The creation time of  physical_e01_gpt_usb.E01  file: 2022-06-11 17:39:57.668068
The modification time of  physical_e01_gpt_usb.E01  file: 2022-06-02 00:54:52.651954
Partition scheme for image acquired: GPT
PARTITION TABLE IS PRINTED BELOW
--------------------------------------------------------------------------------------------------------------------------------

 Address  Description            Start Sector #   End Sector #   Length of Partition MB   Start Byte offset (dec)   End Byte Offset (dec)
 -------  ---------------------  ---------------  -------------  -----------------------  ------------------------  ---------------------
       0  Safety Table                         0              0              0.000488281                         0                      0
       1  Unallocated                          0            127              0.062500000                         0                  65024
       2  GPT Header                           1              1              0.000488281                       512                    512
       3  Partition Table                      2             33              0.015625000                      1024                  16896
       4  Basic data partition               128        1965951            959.875000000                     65536             1006566912
       5  Unallocated                    1965952        1966079              0.062500000                1006567424             1006632448
--------------------------------------------------------------------------------------------------------------------------------
Calculating HASHES now, this depends on the size of the file. Please wait.
The calculated MD5 hash:  96373eefb3ba15a68af034e741f469c4
The calculated SHA1 hash:  7b1efa14b651cd172d8a9da27a359de032f2c59e
Do you want to continue checking (Y or N):
```

Fig. 10.7. Running demonstration of NVMe-Assist step-6.



```
PS D:\> python .\ashar_nvme_df4_gpt_sector_parser.py --help
****************MANUAL PAGE OF NVMe-Assist****************

Description: Checks and parses GPT partition header.

Help manual for GPT parser: python gpt_sector_parser.py --help or python gpt_sector_parser.py -h or python gpt_sector
_parser.py /?

Usage: python gpt_sector_parser.py
PS D:\>
```

Fig. 10.8. Manual page of GPT Sector Parser of NVMe-Assist Toolkit.

Fig. 10.9. Running demonstration of GPT Sector Parser step-1.

Fig. 10.10. Running demonstration of GPT Sector Parser step-2.

```
Please choose the physical acquisition image file (.dd/.raw/.img/.001) from the directory listing above: 33
The file chosen:  physical_dd_gpt_usb4g.001
Proceeding with physical_dd_gpt_usb4g.001 file.
Information:physical_dd_gpt_usb4g.001 acquired from GPT style storage device.
--------------------------------------Counting Total Number of Partitions--------------------------------------
Total partitions in physical_dd_gpt_usb4g.001: 4
---------------------------------------------------GPT HEADER---------------------------------------------------
GPT version:                                            Revision 1.0 for UEFI 2.8.
Size of GPT Header:                                     92 bytes
CRC32 Checksum of GPT Header:                           C7D09751
SECTOR OF GPT Header Structure:                         1
SECTOR OF BACKUP GPT Header Structure:                  7827455
SECTOR OF LBA OF START OF PARTITION AREA:               34
SECTOR OF LBA OF END OF PARTITION AREA:                 7827422
DISK Globally Unique Identifier (GUID):                 84B44480BE3DF644BC6C6C06DD473FFB
SECTOR OF LBA OF START OF THE PARTITION TABLE:          2
NUMBER OF ENTRIES IN THE PARTITION TABLE:               128
SIZE OF EACH ENTRY IN THE PARTITION TABLE:              128 bytes
CRC32 Checksum of partition table:                      0A4917EE
-----------------------------------------------GPT PARTITION TABLE-----------------------------------------------
---------------------------------------------1st GPT PARTITION ENTRY--------------------------------------------
Partition Type GUID:                                    a2a0d0ebe5b9334487c068b6b72699c7
Unique Partition GUID:                                  1c8b684459153e48a1ca7ec9821d6853
Starting LBA SECTOR of the Partition:                   128
**********This Basic Partition Type is NTFS File System.
Ending LBA SECTOR of the Partition:                     1955967
---------------------------------------------2nd GPT PARTITION ENTRY--------------------------------------------
Partition Type GUID:                                    a2a0d0ebe5b9334487c068b6b72699c7
Unique Partition GUID:                                  1ddb31167201d041867a0093b55d2f4a
Starting LBA SECTOR of the Partition:                   1955968
**********This Basic Partition Type is FAT File System.
Ending LBA SECTOR of the Partition:                     3911807
---------------------------------------------3rd GPT PARTITION ENTRY--------------------------------------------
Partition Type GUID:                                    a2a0d0ebe5b9334487c068b6b72699c7
Unique Partition GUID:                                  25b784f13a8d85408ac7183a29d3fed7
Starting LBA SECTOR of the Partition:                   3911808
**********This Basic Partition Type is FAT File System.
Ending LBA SECTOR of the Partition:                     5867647
---------------------------------------------4th GPT PARTITION ENTRY--------------------------------------------
Partition Type GUID:                                    a2a0d0ebe5b9334487c068b6b72699c7
Unique Partition GUID:                                  472527e87305fe4d927ebb2e6a5ae3af
Starting LBA SECTOR of the Partition:                   5867648
**********This Basic Partition Type is NTFS File System.
Ending LBA SECTOR of the Partition:                     7823487
No other partition exists.
----------------------------------------------------------------------------------------------------------------
Do you want to continue checking (Y or N):
```

Fig. 10.11. Running demonstration of GPT Sector Parser step-3.

```
18 . gpt_header_sample
19 . HowTo-Hash_Function.txt
20 . HowTo-Location of Python on Windows 10.txt
21 . HowTo-Make an exe from py.txt
22 . logical_image_01_fat16.dd
23 . logical_image_02_fat32.dd
24 . logical_image_03_fat.dd
25 . logical_image_04_ntfs.dd
Please choose the file to check for logical acquisition image: 22
The file chosen:  logical_image_01_fat16.dd
OEM ID of the partition:  MSWIN4.1
This is a stand-alone FAT-16 logical image.
Do you want to continue checking (Y or N):
```

Fig. 10.12. Running demonstration of Logical Partition Parser step-1.

```
18 . gpt_header_sample
19 . HowTo-Hash_Function.txt
20 . HowTo-Location of Python on Windows 10.txt
21 . HowTo-Make an exe from py.txt
22 . logical_image_01_fat16.dd
23 . logical_image_02_fat32.dd
24 . logical_image_03_fat.dd
25 . logical_image_04_ntfs.dd
Please choose the file to check for logical acquisition image: 23
The file chosen:  logical_image_02_fat32.dd
OEM ID of the partition:  MSDOS5.0
This is a stand-alone FAT-32 logical image.
Do you want to continue checking (Y or N):
```

Fig. 10.13. Running demonstration of Logical Partition Parser step-2.

```
18 . gpt_header_sample
19 . HowTo-Hash_Function.txt
20 . HowTo-Location of Python on Windows 10.txt
21 . HowTo-Make an exe from py.txt
22 . logical_image_01_fat16.dd
23 . logical_image_02_fat32.dd
24 . logical_image_03_fat.dd
25 . logical_image_04_ntfs.dd
Please choose the file to check for logical acquisition image: 25
The file chosen:  logical_image_04_ntfs.dd
OEM ID of the partition:  NTFS
This is a stand-alone NTFS logical image.
Do you want to continue checking (Y or N):
```

Fig. 10.14. Running demonstration of Logical Partition Parser step-3.

```
18 . gpt_header_sample
19 . HowTo-Hash_Function.txt
20 . HowTo-Location of Python on Windows 10.txt
21 . HowTo-Make an exe from py.txt
22 . logical_image_01_fat16.dd
23 . logical_image_02_fat32.dd
24 . logical_image_03_fat.dd
25 . logical_image_04_ntfs.dd
Please choose the file to check for logical acquisition image: 24
The file chosen:  logical_image_03_fat.dd
OEM ID of the partition:
This is not a logical image.
Do you want to continue checking (Y or N):
```

Fig. 10.15. Running demonstration of Logical Partition Parser step-4.

CHAPTER XI

Reporting Guidelines and Instructions

In this chapter, we talk about framing guidelines for digital forensics practitioners. The instructions present in this chapter will serve as a precursor to conducting digital forensics investigations when a non-volatile memory express solid-state drive (NVMe SSD) is acquired in any case. Moreover, this will also assist in examining forensics artifacts of Windows 10 operating systems. The instructions in the chapter are listed in a question-answer format for better understanding.

Q1. **What is Windows Prefetch?**

A. Windows Prefetch is a component of Microsoft Windows operating systems that aims to speed up the application launch and booting process. Also, prefetching depends on the size and complexity of the program and storage devices such as hard-disk drives (HDDs), solid-state drives (SSDs), non-volatile memory express solid-state drives (NVMe SSDs). For example, prefetching MATLAB will take longer compared to MS Paint. Refer chapter IV for a detailed explanation.

Q2. **Is Windows Prefetch still available in Windows 10?**

A. Yes, Microsoft still has this feature available for quicker execution of the programs. However, the Windows service related to Prefetch is now called **SysMain**, which can be found in the **Service** snap-in on Windows or executing **services.msc** command. Refer chapter IV for a detailed explanation.

Q3. **What is the endianness of the registry values for EnablePrefetcher key in Windows Registry for Windows Prefetch?**

A. **EnablePrefetcher** is represented in a **big-endian** format. However, upon right-clicking **EnablePrefetcher** value and then selecting **Modify Binary Data**, the values are represented as **little-endian**.

Q4. **What is the header signature of the Windows Prefetch file?**

A. The header signature is **SCCA**. However, from Windows 10 onwards, contents of Windows Prefetch are compressed using **XPRESS HUFFMAN** algorithm. Hence, the compressed file header signature is **MAM**. Decompression is required to analyze Windows 10 Prefetch files forensically. Refer chapter IV for a detailed explanation.

Q5. **Does Windows Prefetch still work to speed the execution of programs when a Windows operating system is installed on a solid-state drive?**

A. Yes, Prefetch is always running irrespective of the storage media type. It also works on a non-volatile memory express solid-state drive (NVMe SSD) with Windows installed.

Q6. **What are the contents of a prefetch file?**

A. Name of the executable, list of DLLs, program run counter, last run time, prefetch file size, the executable path of the program, created and modified times of prefetch file, and the path of DLL described a device path.

Q7. **What is Windows Shellbag?**

A. Shellbag is a Windows Registry Key that stores information about directory customization, such as changing view options, adding more columns, altering sort order, etc.

Q8. **What is the forensic importance of Shellbag?**

A. All files on a local computer system, network system, and attached external devices such as USB flash drives, external HDDs, and SSD are tracked using Windows Shellbag. The records of files are also updated, which suggests information regarding timestamps of visiting a directory.

Q9. **What are the two user-specific files for Windows Shellbag?**

A. **UsrClass.DAT** and **NTUSER.DAT** are the user-specific files.

Q10. **Are shellbag entries only created for zip files?**

A. Yes, Windows track the customization changes only for zip files. Unfortunately, Windows does not do so for rar and 7zip files. Refer V for more details.

Q11. **What is the file location of NTUSER.DAT file in Windows 10?**

A. ***C:\Users\⟨username⟩\NTUSER.DAT***

Q12. **What is the file location of UsrClass.DAT file in Windows 10?**

A. ***C:\Users\⟨username⟩\AppData\Local\Microsoft\Windows\UsrClass.DAT***

Q13. **Out of the three software tools, OSForensics, ShellBags Explorer, and ShellBagView, which one of these produces the most forensics information?**

A. **ShellBags Explorer** by Eric Zimmerman produces the most relevant information regarding Shellbag entries.

Q14. **What are ETL files?**

A. **Event Trace Logs (ETL)** files store a snapshot of events relating to a system's state information at a specific time or event. Refer chapter VI for a more detailed explanation.

Q15. **What kind of information can be found from ETL files?**

A. Information about the system shut down, startup, restarting, user logon, secondary user logon, etc.

Q16. **Is BootCKCL.etl file the same as BootPerfDiagLogger.etl?**

A. Yes, Microsoft has changed the name of **BootCKCL.etl** file to **BootPerfDiagLogger.etl**.

Q17. **Where is BootPerfDiagLogger.etl file located in Windows 10 operating system?**

A. It is located in: **C:\Windows\System32\WDI\LogFiles**.

Q18. **What information is displayed by ETLParser?**

A. The information displayed by **ETLParser**: log file name, timestamp of event recording in UTC format, triggering event, provider name, GUID, process ID, thread ID, process name, task, opcode, version, channel, level, task name.

Q19. **What information is displayed by PerfView?**

A. **PerfView** exhibits process summary information that includes command-line execution of static and dynamic traces. Furthermore, it displays the trace machine's details, including the trace start time, trace end time, operating system information, total number of events, live process summary, event types, and details. Refer chapter VI for a more detailed explanation.

Q20. **What information is exhibited by FullEventLogview?**

A. **FullEventLogview** shows event time, record ID, event ID, level, opcode, keywords, process ID, thread ID, computer name, user, log file, etc.

Q21. **What information is presented by SVCLog Viewer?**

A. Information regarding event description, process name, event time, source, basic information, and general properties of events are displayed.

Q22. **What information is put forth by TraceFMT?**

A. Information about the operating system version, the event's start time, end time of the event, timezone information, maximum file size information, total events, etc., are parsed by **TraceFMT** from an ETL file.

Q23. **What information of forensics importance is reported by Windows Performance Analyzer (WPA)?**

A. WPA shows system activity, processes, images, computation information, process name, event name, duration of the event, and processes.

Q24. **When using Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs enclosed in a USB enclosure, what behavior did they exhibit in the TRIM ON case when images were acquired using a USB WriteBlocker?**

A. In the TRIM on case of Samsung and Seagate NVMe SSDs, files under 693 bytes and 696 bytes were found to be intact after file recovery. However, files over 693 and 696 bytes had their contents wiped out. Whereas in Western Digital and Silicon Power NVMe SSDs, files under 696 bytes were found to be intact after file recovery. But there was no recovery possible in Western Digital and Silicon Power NVMe SSDs when the file size was over 696 bytes. Refer chapter VII for more details.

Q25. **When using Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs enclosed in a USB enclosure, what behavior did they exhibit in the TRIM OFF case when images were acquired using a USB WriteBlocker?**

A. In the TRIM off case of Samsung Seagate, Western Digital, and Silicon Power NVMe SSDs all the files were recovered successfully. File contents and hash values of the files were intact. Refer chapter VII for more details.

Q26. **Did the hash values change when consecutive forensics images were acquired of Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs enclosed in a USB enclosure with WriteBlocker connected?**

A. Yes, MD5 and SHA-1 hash values were all different for all the four NVMe SSDs brand types in the both cases of TRIM on and TRIM off cases. Refer chapter VII for more details.

Q27. **When using Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs enclosed in a USB enclosure, what behavior did they exhibit in the TRIM ON case when images were acquired without using a USB WriteBlocker?**

A. The drives exhibited a very similar behavior as shown in chapter VII. However, files with larger sizes were impacted the most by the NVMe SSDs' controller chips. Refer chapter VIII for further information.

Q28. **When using Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs enclosed in a USB enclosure, what behavior did they exhibit in the TRIM OFF case when images were acquired without using a USB WriteBlocker?**

A. All the files were recovered successfully in the TRIM off case of Samsung Seagate, Western Digital, and Silicon Power NVMe SSDs without using a USB WriteBlocker. However, only one .bin file out of three was fully recovered from all drives. This indicated that controller chips of the drives acted in .bin files the most. Refer chapter VIII for more details.

Q29. **Did the hash values change when consecutive forensics images were acquired of Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs enclosed in a USB enclosure when USB WriteBlocker was**

not used for forensics image acquisition purposes?

A. Yes, all hash values were different for the respective NVMe SSDs. Refer chapter VIII for more details.

Q30. **When using Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs as a primary boot device, what behavior did they exhibit in the TRIM ON case when images were acquired using an NVMe WriteBlocker?**

A. Only the controller chips of Samsung and Silicon Power NVMe SSDs behaved intelligently to a certain extent for the file recovery operation. Most of the file types were irrecoverable in the case of Samsung NVMe SSD, while in the case of Silicon Power, most file types were recovered but, unfortunately, corrupted. Refer chapter IX for more details.

Q31. **When using Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs as a primary boot device, what behavior did they exhibit in the TRIM OFF case when images were acquired using an NVMe WriteBlocker?**

A. Most of the files were recovered from all four NVMe SSDs. However, even after recovery, some files were corrupted, or their contents were zeroed out. Refer chapter IX for more details.

Q32. **Did the hash values change when consecutive forensics images were acquired of Samsung, Seagate, Western Digital, and Silicon Power NVMe SSDs used as a primary boot device using NVMe WriteBlocker?**

A. No, hash values for all the NVMe SSDs did not change till seven days, which is extremely surprising. The autonomous movement of data around pages of NVMe SSDs was stopped due to NVMe WriteBlocker for seven days. This was true for both TRIM on and TRIM off cases of the experiment. Refer chapter IX for more details.

Q33. **Based on the experiment's results, are controller chips more intelligently designed to retain data after deletion for file recovery purposes?**

A. Yes. Storage manufacturers have considered the factor of file recovery for better reliability of user data, which makes data retention capability better compared to controller chips of SATA SSDs, which do not have complete reliability.

Q34. **Based on the experimental scenario and results, which brand of NVMe SSDs can be considered the most reliable?**

A. As can be seen from our experimental analysis, Samsung NVMe SSD was the most reliable. Silicon Power came second when it comes to data retention after deletion and file recovery.

Q35. **What type of device usage is affected by wear-leveling the most?**

A. NVMe SSD used as a primary boot device shows the affect of wear-leveling to the maximum. The chances of file recovery is slim when the device is used a primary boot medium.

CHAPTER XII

Conclusion and Future Work

Conclusion

In this dissertation, we have conducted a rigorous study of NVMe SSD forensics, namely on four brands, Samsung, Seagate, Western Digital, and Silicon Power, respectively for our NVMe-Assist framework. In addition, we conducted exhaustive experiments to find out the behavior pattern of wear-leveling in our four different storage media. The objective of our investigation was to find a trend of deletion patterns based on the four different controller chips used in the media. We developed steps for conducting qualitative research and empirical procedure to support our hypothesis. We filled our NVMe storage devices with files from the Digital Corpora dataset. The unique thing about our experiment was that we used two different WriteBlockers from WiebeTech, one for USB protocol and another for PCIe mechanism.

The design approach for recovering deleted files from the NVMe SSD was simple. First, we populated the devices with files from Digital Corpora. After copying the files, we kept the computer system powered on with the storage media attached, and then after that, we deleted the files after twenty-four hours. We started acquiring full physical images one day after deletion. Altogether we took four forensics images, with three taken at a gap of one day and the last one taken four days from the third image. Lastly, we used AccessData FTK and Autopsy tools to recover files and note down the deletion pattern of our four different devices.

In addition to investigating deletion patterns and thereby conducting file recovery from AccessData FTK and Autopsy tools, we also touched upon Windows 10 artifacts. Particularly, we investigated Prefetch, Shellbag, and the new BootPerfDiafLogger.etl files of Windows 10 v21H2 operating system. We applied different open-source, proprietary, and freeware tools to formulate a decisive result and contribute to the digital forensics community.

Future Work

In this work, we have focused on file deletion patterns of four brands of NVMe SSD having varying controller chips and conducting recovery along with investigating Windows 10 artifacts. In addition, our NVMe-Assist framework accommodated two popular digital forensics tools of different, AccessData FTK, which is proprietary, and Autopsy, which is open-source. Finally, we also used other tools to investigate artifacts generated by Windows 10 forensically and present the result. The ability to conduct a forensics investigation with only a specific selected toolset can be expanded, thereby improving the quality of the inquiry. This will help practitioners answer further questions which might have been left unanswered by only using selective tools. Findings from our experiment can be expanded by using tools such as Belkasoft X, Magnet AXIOM, Cellebrite Inspector, and Oxygen Forensic Detective, to name a few. Using these tools and investigating file deletion patterns for different brands of NVMe SSDs will only enhance the literature in storage media forensics.

REFERENCES

[1] A. Pal and N. Memon, "The evolution of file carving," *IEEE Signal Processing Magazine*, vol. 26, no. 2, pp. 59–71, 2009.

[2] G. B. Bell and R. Boddington, "Solid state drives: the beginning of the end for current practice in digital forensic recovery?" *Journal of Digital Forensics, Security and Law*, vol. 5, no. 3, p. 1, 2010.

[3] F. Yohannes and S. Zanero, "Solid State Drive (SSD) Digital Forensics Construction," Ph.D. dissertation, Polytechnic University of Milan, 2011.

[4] R. Ritola, "Forensic investigation of Solid State Drive," Ph.D. dissertation, Dalarna University, 2012.

[5] G. Bonetti, M. Viglione, A. Frossi, F. Maggi, and S. Zanero, "A comprehensive black-box methodology for testing the forensic characteristics of solid-state drives," in *Proceedings of the 29th Annual Computer Security Applications Conference*, ser. ACSAC '13. New York, NY, USA: Association for Computing Machinery, 2013, p. 269–278. [Online]. Available: https://doi.org/10.1145/2523649.2523660

[6] J. Canclini, J. McMasters, J. Shey, O. Walker, R. Rakvic, H. Ngo, and K. D. Fairbanks, "Inferring read and write operations of solid-state drives based on energy consumption," in *2016 IEEE 7th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)*, 2016, pp. 1–6.

[7] B. R. Joshi and R. Hubbard, "Forensics analysis of solid state drive (ssd)," in *2016 Universal Technology Management Conference (UTMC)*, researchgate.

net. The Society of Digital Information and Wireless Communications (SDIWC) Wilmington, New Castle, DE 19801, USA, 2016, pp. 1–12.

[8] S. S. R. Marupudi, “Solid State Drive: New Challenge for Forensic Investigation,” St. Cloud State University, Tech. Rep., 2017. [Online]. Available: https://repository.stcloudstate.edu/msia_etds/30

[9] A. Neyaz, N. Shashidhar, and U. Karabiyik, “Forensic analysis of wear leveling on solid-state media,” in *2018 17th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/ 12th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2018, pp. 1706–1710.

[10] Z. Johnson, A. Varon, J. Blanco, R. Rakvic, J. Shey, H. Ngo, D. Brown, and O. Walker, “Classifying solid state drive firmware via side-channel current draw analysis,” in *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech)*, 2018, pp. 943–948.

[11] G. Ninahualpa, C. Perez, S. G. Yoo, T. Guarda, J. Diaz, and D. Piccirilli, “Restoring data in solid state devices damaged by crushing and falling, using file carving technique,” in *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 2018, pp. 1–4.

[12] J. Shey, J. A. Blanco, O. Walker, T. W. Tedesso, H. T. Ngo, R. Rakvic, and K. D. Fairbanks, “Monitoring device current to characterize trim operations of solid-state drives,” *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1296–1306, 2019.

[13] A. Nisbet and R. Jacob, "Trim, wear levelling and garbage collection on solid state drives: A prediction model for forensic investigators," in *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*, 2019, pp. 419–426.

[14] S. Kandala, "Analyzing the Trimming Activity of Solid-State Drives in Digital Forensics," St. Cloud State University, Tech. Rep. 81, 2019. [Online]. Available: https://repository.stcloudstate.edu/msia_etds/81

[15] A. Neyaz, B. Zhou, and N. Karpoor, "Comparative study of wear-leveling in solid-state drive with ntfs file system," in *2019 IEEE International Conference on Big Data (Big Data)*, 2019, pp. 4294–4298.

[16] A. Kumar, A. Neyaz, and N. Shashidhar, "A survey on solid-state drive forensic analysis techniques." *International Journal of Computer Science and Security (IJCSS)*, vol. 14, no. 2, pp. 13–22, 2020.

[17] B. Nikkel, "Nvm express drives and digital forensics," *Digital Investigation*, vol. 16, pp. 38–45, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287616000025

[18] I. Riadi and A. Hadi, "Analysis of Digital SSD NVMe Evidence on Proprietary Operating Systems Using the Static Forensics Method," 2019.

[19] I. Riadi, S. Sunardi, and A. Hadi, "Analysis of Digital Evidence Trim Enable NVME SSD Using Static Forensics Method," *JUITA: Jurnal Informatika*, vol. 8, no. 1, pp. 65–74, 2020. [Online]. Available: http://jurnalnasional.ump.ac.id/index.php/JUITA/article/view/6584

[20] I. Garcia and E. Luis, "Bulk Extractor Windows Prefetch Decoder," Naval Postgraduate School Monterey, CA, Tech. Rep., 2011.

[21] A. Heriyanto, "Forensic Examination and Analysis of the Prefetch Files on the Banking Trojan Malware Incidents," 12 2014.

[22] N. K. Shashidhar and D. Novak, "Digital Forensic Analysis on Prefetch Files," *International Journal of Information Security Science*, vol. 4, no. 2, pp. 39–49, 2015.

[23] Y. Khatri, "Forensic Implications of System Resource Usage Monitor (SRUM) Data in Windows 8," *Digital Investigation*, vol. 12, pp. 53–65, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287615000031

[24] B. Singh and U. Singh, "Leveraging the Windows Amcache.hve File in Forensic Investigations," *Journal of Digital Forensics, Security and Law*, vol. 11, pp. 37–54, 12 2016.

[25] Z. Abdulelah, "Design and Implement a Hidden Processes Detector (HPD) based on Windows Prefetch Files," *International Journal of Computer Applications*, vol. 171, pp. 37–39, 08 2017.

[26] M. Saleh and M. Shamaileh, "Forensic Artifacts Associated with Intentionally Deleted User Accounts," *International Journal of Electronic Security and Digital Forensics*, vol. 9, p. 1, 01 2017.

[27] D. Hintea, R. Bird, and M. Green, "An Investigation into the Forensic Implications of the Windows 10 Operating System: Recoverable Artefacts and Significant Changes from Windows 8.1," *International Journal of Electronic Security and Digital Forensics*, vol. 9, p. 326, 01 2017.

[28] B. Singh, "Program Execution Analysis in Windows: A Study of Data Sources, their Format and Comparison of Forensic Capability," *Computers and Security*, vol. 74, 01 2018.

[29] V. Vouvoutsis, “Manipulating and Generating Windows 10 Prefetch files,” University of Piraeus, Tech. Rep., 2019. [Online]. Available: https://dione.lib.unipi.gr/xmlui/handle/unipi/12017

[30] B. Alsulami, “Behavioral Malware Detection and Classification Using Windows Prefetch Files,” Ph.D. dissertation, 2019.

[31] D. J. Ohana and N. Shashidhar, “Do private and portable web browsers leave incriminating evidence? a forensic analysis of residual artifacts from private and portable web browsing sessions,” in *2013 IEEE Security and Privacy Workshops*, 2013, pp. 135–142.

[32] Y. Zhu, P. Gladyshev, and J. James, “Using shellbag information to reconstruct user activities,” *Digital Investigation*, vol. 6, pp. S69–S77, 2009, the Proceedings of the Ninth Annual DFRWS Conference. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1742287609000413

[33] D. A. Duncan, “Exploring the Uses of ShellBag Data within the Windows 7 Registry.” East Tennessee State University, Honors Thesis - Open Access P-136, 2012. [Online]. Available: https://dc.etsu.edu/honors/136

[34] M. P. Mbatha, “Windows Registry Forensic Artifacts; Shellbags for Computer Security,” Ph.D. dissertation, University of Nairobi, 2016.

[35] N. Panwar, “Anti Forensics Analysis of File Wiping Tools,” Sardar Patel University of Police, Security and Criminal Justice, Jodhpur, Tech. Rep., 2016. [Online]. Available: https://www.idrbt.ac.in/assets/publications/Reports/Summer%20Interns%202016/Narendra%20Panwar.pdf

[36] M. Muir, P. Leimich, and W. Buchanan, “A forensic audit of the tor browser bundle,” *Digital Investigation*, vol. 29, 03 2019.

[37] A. Duranec, D. Topolcic, K. Hausknecht, and D. Delija, "Investigating file use and knowledge with Windows 10 Artifacts," in *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, 2019, pp. 1213–1218.

[38] A. Sanker, "Analysis of shellbags date time update conditions," Ph.D. dissertation, Utica College, 2020.

[39] A. Gillis, "Hard Disk Drive," Available at https://www.techtarget.com/searchstorage/definition/hard-disk-drive, 2021.

[40] Datarecovery.com, "What Does Hard Drive Platter Damage Look Like?" Available at https://datarecovery.com/2015/07/hard-drive-platter-damage/, 2015.

[41] NTFS.com, "Hard Drive Basics," Available at https://www.ntfs.com/hard-disk-basics.htm, 2017.

[42] G. Kranz, "Serial ATA (serial advanced technology attachment or sata)," Available at https://searchstorage.techtarget.com/definition/Serial-ATA, 2021.

[43] A. Gillis, "SSD (Solid-State Drive)," Available at https://searchstorage.techtarget.com/definition/SSD-solid-state-drive, 2021.

[44] O and O Software GmbH, "About Solid State Drives (SSD)," Available at https://www.oo-software.com/en/docs/whitepaper/whitepaper_ssd.pdf, 2011.

[45] R. Allen, "NAND and Cells: SLC, QLC, TLC and MLC Explained," Available at https://www.techradar.com/news/nand-and-cells-slc-qlc-tlc-and-mlc-explained, 2021.

[46] StorageReview.com, "SSD Controller," Available at https://rb.gy/yixjbb, 2020.

[47] M. Huculak, "How to Ensure TRIM is Enabled on Windows 10 to Keep an SSD at Top Performance," Available at https://www.windowscentral.com/how-ensure-trim-enabled-windows-10-speed-ssd-performance, 2016.

[48] A. Valette, "Overview of 'Wear Leveling' With SSD Controllers," Available at https://www.ontrack.com/en-us/blog/wear-leveling, 2016.

[49] S. Harding, "What Is PCIe? A Basic Definition," Available at https://rb.gy/xbbacl, 2020.

[50] K. Technology, "Understanding SSD Technology: NVMe, SATA, M.2," Available at https://www.kingston.com/unitedstates/us/community/articledetail/articleid/48543, 2017.

[51] F. Picasso, "A First Look at Windows 10 Prefetch Files," Available at https://blog.digital-forensics.it/2015/06/a-first-look-at-windows-10-prefetch.html, 2015.

[52] V. Lo, "Windows Shellbag Forensics in Depth," *SANS Institute*, 2014.

[53] N. Ibrahim, "ETW Event Tracing for Windows and ETL Files - Hacking Exposed Computer Forensics Blog," 2018. [Online]. Available: https://www.hecfblog.com/2018/06/etw-event-tracing-for-windows-and-etl.html

[54] B. Carrier, *File System Forensic Analysis*. Addison-Wesley Professional, 2005.

[55] Forensics Wiki, "Prefetch Forensics," Available at https://forensicswiki.xyz/wiki/index.php?title=Prefetch, 2020.

[56] L. Rocha, "Digital Forensics - Prefetch Artifacts," Available at https://countuponsecurity.com/2016/05/16/digital-forensics-prefetch-artifacts/, 2016.

[57] HexCorn, "Prefetch Hash Calculator and Hash Lookup Table for Windows XP/Windows Vista/Windows 7/Windows 2003/Windows 2008," Available at https://www.hexacorn.com/blog/2012/06/13/prefetch-hash-calculator-a-hash-lookup-table-xpvistaw7w2k3w2k8/, 2012.

[58] Francesco Picasso, *Windows 10 Prefetch Decompressor*, RealityNet, Genoa, Italy, 2015. [Online]. Available: https://gist.github.com/dfirfpi/113ff71274a97b489dfd

[59] Forensics Wiki, "Prefetch File Format and File Information," Available at https://forensicswiki.xyz/wiki/index.php?title=Windows_Prefetch_File_Format, 2020.

[60] PassMark Software, *OSForensics v9*, PassMark Software Inc., Redwood City, CA, USA, 2022. [Online]. Available: https://www.osforensics.com/

[61] Adam Witt, *Windows Prefetch Parser*, GitHub, USA, 2021. [Online]. Available: https://github.com/PoorBillionaire/Windows-Prefetch-Parser/blob/master/windowsprefetch/windowsprefetch.py

[62] Nir Sofer, *WinPrefetchView*, Nirsoft, USA, 2020. [Online]. Available: https://www.nirsoft.net/utils/win_prefetch_view.html

[63] Eric Zimmerman, *PECmd*, Indianapolis, IN, USA, 2020. [Online]. Available: https://ericzimmerman.github.io/#!index.md

[64] C. Eastwood, "Shellbags Analysis," Available at https://medium.com/ce-digital-forensics/shellbag-analysis-18c9b2e87ac7, Nov. 17, 2020.

[65] J. McQuaid, "Forensic Analysis of Windows Shellbags," Available at https://www.magnetforensics.com/blog/forensic-analysis-of-windows-shellbags/, Aug. 17, 2014.

[66] S. White, K. Sharkey, and M. Satran, "Registry Hives," Available at https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-hives, Jan. 1, 2021.

[67] R. Chandel, "Forensic Investigation: Shellbags," Available at https://www.hackingarticles.in/forensic-investigation-shellbags/, Oct. 26, 2020.

[68] LiFars, "Windows ShellBags Forensics, Investigative Value of Windows ShellBags," Available at https://lifars.com/knowledge-center/windows-shellbags-forensics-investigative-value-of-windows-shellbags/, Apr. 2, 2020.

[69] "Shellbags Forensics: Addressing a Misconception," Available at https://www.4n6k.com/2013/12/shellbags-forensics-addressing.html, Dec. 4, 2013.

[70] Eric Zimmerman, *ShellBags Explorer*, Indianapolis, IN, USA, 2022. [Online]. Available: https://ericzimmerman.github.io/#!index.md

[71] Nir Sofer, *Shellbags View*, NirSoft, USA, 2020. [Online]. Available: https://www.nirsoft.net/utils/shell_bags_view.html

[72] S. Robert, G. Kranz, and D. Raffo, "Computer Storage," Available at https://www.techtarget.com/searchstorage/definition/storage, 2021.

[73] C. Mellor, "Hard Disk Drive Shipments Fell 50 percent Between 2012 and 2019," Available at https://blocksandfiles.com/2020/01/14/disk-drive-shipments-50-per-cent-fallfrom-2012-to-2019/, 2020.

[74] H. Riggs, S. Tufail, I. Parvez, and A. Sarwat, "Survey of solid state drives, characteristics, technology, and applications," in *2020 SoutheastCon*, 2020, pp. 1–6.

[75] A. Valette, "Overview of 'Wear Leveling' With SSD Controllers and What is it?" Available at https://www.ontrack.com/en-us/blog/wear-leveling, 2016.

[76] P. Bednar and V. Katos, "Ssd: New challenges for digital forensics," 10 2011.

[77] AccessData, "AccessData FTK," Exterro, Portland, OR, USA, 2022. [Online]. Available: https://www.exterro.com/forensic-toolkit

[78] X-Ways Forensics, *WinHex*, X-Ways Software Technology AG, Germany, 2022. [Online]. Available: https://www.x-ways.net/winhex/

[79] "Market share of Solid-State Drives," Available at https://www.tomshardware.com/news/ssd-market-shares-q1-2021-trendfocus, 2021.

[80] S. Garfinkel, P. Farrell, V. Roussev, and G. Dinolt, "Bringing science to digital forensics with standardized forensic corpora. digital investigation, 6," Available at https://digitalcorpora.org/, 2009.

VITA

Ashar Neyaz

Education

Ph.D. Digital and Cyber Forensic Science **August 2022**
Sam Houston State University, Huntsville, TX
Dissertation: NVMe-Assist: A Novel Theoretical Framework for Digital Forensics
Advisor: Dr. Narasimha Shashidhar
GPA: 4.00

Master of Science Digital Forensics **December 2017**
Sam Houston State University, Huntsville, TX
Thesis: Understanding and Predicting Wear-leveling by Performing Forensics in Solid-State Media
Advisor: Dr. Narasimha Shashidhar
GPA: 4.00

Bachelor of Engineering Computer Science **June 2014**
Siddaganga Institute of Technology, Tumakuru, India
Graduated First Class with Distinction
GPA: 3.45

Teaching Experience

Course Instructor (DFSC 1316), Undergraduate
Course: Digital Forensics and Information Assurance-I
Semester Taught: Fall 2021, Spring 2022
Department of Computer Science, Sam Houston State University

Course Instructor (CSTE 1331), Undergraduate
Course: Visual Computing using Python
Semester Taught: Spring 2021
Department of Computer Science, Sam Houston State University

Teaching Assistant (DFSC 7358), Graduate
Course: Memory Forensics
Semester Taught: Spring 2020
Department of Computer Science, Sam Houston State University

Course Instructor (COSC 1436), Undergraduate
Course: Java Programming Fundamentals-I
Semester Taught: Fall 2019, 2020
Department of Computer Science, Sam Houston State University

Teaching Interests

- </> Digital Forensics and Information Assurance
- File Systems, Operating Systems, & Registry Forensics
- Network Forensics Fundamentals
- Web Browser Forensics
- Linux Forensics
- >_ Memory Forensics with Volatility
- Java and Python Fundamentals

Research Interests

- Operating Systems Forensics
- Storage Media Forensics
- Memory Forensics
- Mobile Device Forensics

Presentations

2020: Security, Privacy and Steganographic Analysis of FaceApp and TikTok. *Business, Energy, Technology, and Health Webinar*

2019: Comparative Study of Wear-leveling in Solid-State Drive with NTFS File System. *IEEE Big Data: The 3rd International Workshop on Big Data Analytic for Cyber Crime Investigation and Prevention Conference, Los Angeles, California, USA*

2018: Forensic Analysis of Wear Leveling on Solid-State Media. *IEEE Trustcom Conference, New York, USA*

Reviewer

Current (2022): **Scientific Committee Member** *International Symposium of Digital Forensics & Security*

From 2020 - Current: **Reviewer Board Member** *International Journal of Security*

From 2020 - Current: **Journal Article Reviewer** *International Journal of Security*

From 2020 - Current: **Journal Article Reviewer** *International Journal of Cyber Criminology*

2020: **Conference Article Reviewer** *International Conference on Math & Computing*

Awards and Memberships

Student Excellence in Teaching Award, 2022
College of Science & Engineering Technology, Sam Houston State University

Outstanding Ph.D. Student Award, 2022
Department of Computer Science, Sam Houston State University

Member of the Scientific Committee, 2022
International Symposium on Digital Forensics and Security

Best Reviewer Award, 2020
International Journal of Security (IJS)

Member, 2018
Institute of Electrical and Electronic Engineers

Member, 2015-2017
International Student Organization, Sam Houston State University

Technical Skills

- **Programming Languages:** Java, Python, Powershell
- **Software:** Microsoft Office, Adobe Suite, LaTeX, VMware, Virtual Box, Wireshark, Cherwell System
- **Operating Systems:** Microsoft Windows, Apple macOS, Ubuntu, Kali
- **Forensics Tools:** AccessData FTK, Autopsy, OSForensics, Cellebrite Inspector, open-source and freeware digital forensics tools

Research Publications

Journals

J1. **Ashar Neyaz**, Narasimha Shashidhar, Cihan Varol, and Amar Rasheed. **"Digital Forensics in NVMe SSDs with NVMe WriteBlocker"**. International Journal of Security (IJS) (in press). 2022

J2. Sundar Krishnan, **Ashar Neyaz**, and Qingzhong Liu. **"IoT Network Attack Detection using Supervised Machine Learning"**. International Journal of Artificial Intelligence and Expert Systems (IJAE), Volume (10): Issue (2), 2021

J3. **Ashar Neyaz**, Avinash Kumar, Sundar Krishnan, Jessica Placker, and Qingzhong Liu.**"Security, Privacy and Steganographic Analysis of FaceApp and TikTok"**. International Journal of Computer Science and Security (IJCSS), 2020

J4. Avinash Kumar, **Ashar Neyaz**, and Narasimha Shashidhar. **"A Survey On Solid-State Drive Forensic Analysis Techniques"**. International Journal of Computer Science and Security (IJCSS), 2020

J5. **Ashar Neyaz**, and Narasimha Shashidhar. **"USB Artifact Analysis Using Windows Event Viewer, Registry and File System Logs"**. MPDI Electronics Volume (8): Issue (11), 2019

J6. Sundar Krishnan, **Ashar Neyaz** and Narasimha Shashidhar, **"A Survey of Security and Forensic Features in Popular eDiscovery Software Suites"**. International Journal of Security (IJS), Volume (10): Issue (2), 2019

J7. **Ashar Neyaz**, and Cihan Varol. **"Audio Steganography via Cloud Services: Integrity Analysis of Hidden File"**. International Journal of Cyber-Security and Digital Forensics (IJCSDF) 7(1): 79-86. The Society of Digital Information and Wireless Communications (SDIWC), 2018

Conference Proceedings

CP1. **Ashar Neyaz**, Narasimha Shashidhar, Cihan Varol, and Amar Rasheed. **"Digital Forensics Analysis of Windows 11 Shellbag with Comparative Tools"**. 10th International Symposium on Digital Forensics and Security (ISDFS), Istanbul, Turkey. 2022

CP2. Khushi Gupta, **Ashar Neyaz**, Narasimha Shashidhar, and Cihan Varol. **"Digital Forensics Lab Design: A Framework"**. 10th International Symposium on Digital Forensics and Security (ISDFS), Istanbul, Turkey. 2022

CP3. **Ashar Neyaz**, Bing Zhou, and Narasimha Shashidhar. **"Comparative Study of Wear-leveling in Solid-State Drive with NTFS File System"**. IEEE Big Data: The 3rd International Workshop on Big Data Analytic for Cyber Crime Investigation and Prevention, Los Angeles, California, USA. 2019

CP4. **Ashar Neyaz**, Narasimha Shashidhar, and Umit Karabiyik, **"Forensic Analysis of Wear Leveling on Solid-State Media"**. 17th IEEE International Conference on Trust, Security and Privacy in Computing and Communications, New York, NY, USA. 2018

Book Chapters

BC1. **Ashar Neyaz**, and Narasimha Shashidhar. (in press). **"Windows Prefetch Forensics"**. Breakthroughs in Digital Biometrics and Forensics. Springer. 2022. (**Editors**: Kevin Daimi, Guillermo A. Francia, and Luis Hernández Encinas)

BC2. **Ashar Neyaz**, Narasimha Shashidhar, Cihan Varol, Amar Rasheed. (in press). **"Digital Forensics Analysis in NVMe SSDs inside USB Enclosure Adapters"**. Breakthroughs in Digital Biometrics and Forensics. Springer. 2022. (**Editors**: Kevin Daimi, Guillermo A. Francia, and Luis Hernández Encinas)

Work Experience

Doctoral Research Assistant

September 2018 - Present
Dept. of Computer Science
Sam Houston State University

- Taught courses of Java, Python, Fundamentals of Digital Forensics and Information Assurance as a teaching assistant.
- Conducted research work, writing articles and journal papers.
- Contributed to research project for academic conferences.
- Assisted in organizing academic teaching documents required by faculty members and dissertation supervisor.
- Collaborated and coordinated seminars, discussion groups, and laboratory sessions.
- Conducted surveys, laboratory experiments, and other research for use in scholarly publications.

Computer Systems Technician

May 2018 - August 2018
IT Client Services
Sam Houston State University

- Supported and maintained level one computer systems, desktops, and peripherals.
- Performed installation, diagnosing, repairing, maintaining, and upgrading all minor hardware and equipment while ensuring optimal workstation performance.
- Installed, configured, tested, and troubleshot workstations' hardware and software.
- Troubleshot level-one issues and minor problem areas in a timely and accurate fashion.
- Accurately identified and escalated large-scale problems to the proper group(s) for resolution.
- Responded to service requests regarding PC and hardware issues.
- Provided initial contact, troubleshooting, and supporting, conveying resolutions to client issues.

Graphics Designer

June 2017 - August 2017
John R. Ragsdale Visitor & Alumni Center
Sam Houston State University

- Developed media pieces, program brochures, banners, flyers and websites.
- Performed as a student ambassador, served in-person visitor and gave campus tours.

Graduate Assistant

February 2016 - May 2017
Texas Research Institute for Environmental Studies
Sam Houston State University

- Designed digital maps in Esri ArcMap.
- Analyzed data and geo-coordinates for invasive species.