THE REPUBLIC OF TURKEY

BAHCESEHIR UNIVERSITY

TIMELINE ANALYSIS BASED ON CRITICAL WINDOWS INCIDENT RESPONSE ARTIFACTS

Master's Thesis

OMER ALJOBOURY

ISTANBUL, 2020

THE REPUBLIC OF TURKEY
BAHCESEHIR UNIVERSITY

GRADUATE SCHOOL
MASTER OF SCIENCE IN CYBER SECURITY

TIMELINE ANALYSIS BASED ON CRITICAL WINDOWS INCIDENT RESPONSE ARTIFACTS

Master's Thesis

OMER AL JOBOURY

Supervisor: ASST. PROF. AHMET NACI UNAL

Istanbul, 2020

T.C.
BAHÇEŞEHİR UNIVERSITY
GRADUATE SCHOOL

...../...../.....

MASTER THESIS APPROVAL FORM

Program Name:	
Student's Name and Surname:	
Name Of The Thesis:	
Thesis Defense Date:	

This thesis has been approved by the Graduate School which has fulfilled the necessary conditions as Master thesis.

Assoc. Prof. Dr. Burak KÜNTAY
Institute Director

This thesis was read by us, quality and content as a Master's thesis has been seen and accepted as sufficient.

	Title/Name	Signature
Thesis Advisor's		
Member's		
Member's		

ACKNOWLEDGMENTS

First and foremost, I offer my sincerest and thorough gratitude to my academic supervisor, Professor Ahmet Unal for his patience, constant support, guidance, and encouragement during my master years. Furthermore, I would like to thank all the educational board and staff at Bahcesehir University for the hospitality and the wonderful opportunity to study in Istanbul – Turkey.

Secondly, I would like to express my deepest appreciation to Mr. Ibrahim Saruhan, the founder of Digisecure company, for his vital point of view and advice through the last year during my research, and for giving me the privilege to be part of his Research and Development team, last and not least giving me the permission to use their software ARTIFAST. Thanks for offering me all these facilities, which without it would not be possible to finish this thesis.

Thirdly, my thanks extend to both Ashraf Haddadeen lead researcher, and Mohammed Shiha lead Developer at Digisecure for their guidance, advice and for sharing their technical field knowledge.

Finally, I would like to seize this once in a lifetime opportunity to express my profound gratitude to my parents and brothers for their patience, warm encouragement, and their moral contribution during my educational period at Bahcesehir University.

Words fail me but you didn't – thank you for all you've done.

Omer Al Joboury

Istanbul, November 26, 2020

ABSTRACT

TIMELINE ANALYSIS BASED ON CRITICAL WINDOWS INCIDENT RESPONSE ARTIFACTS

Omer Al Joboury

Master of Science in Cyber Security

Supervisor: ASST. PROF. AHMET NACI UNAL

December 2020, Number of pages of main text 79

In today's enterprise world mainly windows operating systems are used in critical servers. These operating systems have internal features and also run many different applications to control both software and hardware. Users leave fingerprints while using different kind of applications on Windows systems. Files created based on user activities and also operating system files updated due to user activities are called artifacts. Digital forensic investigators collect these artifacts to verify incidents and consequently they use these artifacts to find answers after an incident to determine Who did What, When, Where and Why.

In this thesis, I plan to do research on artifacts from Scheduler Tasks, System Resource Usage Monitor, Win32 Service, Update Sequence Number Journal and Windows LogFile, in Windows Operating Systems (Window XP, Windows 7, and Windows 10).

The goal of this thesis to show how these artifacts are created, what information they contain, how to create timeline based on these artifacts, where to find and locate these artifacts, and how important they are during incident response and digital investigations.

Keywords: Timeline Analysis, Windows Artifacts, Incident Response, Digital Forensics

ÖZET

KRİTİK WINDOWS OLAY MÜDAHALESİ YAPIMLARINA DAYALI ZAMAN ÇİZELGESİ ANALİZİ

Omer Al Joboury

SİBER GÜVENLİKTE BİLİM Ustası

Danışman: ASST. PROF. AHMET NACI UNAL

Aralık 2020, Ana metnin sayfa sayısı 79

Günümüzün kurumsal dünyasında kritik sunucularda ağırlıklı olarak Windows işletim sistemleri kullanılmaktadır. Bu işletim sistemleri dahili özelliklere sahiptir ve ayrıca hem yazılımı hem de donanımı kontrol etmek için birçok farklı uygulamayı çalıştırır. Kullanıcılar Windows sistemlerinde farklı türden uygulamaları kullanırken parmak izleri bırakırlar. Kullanıcı etkinliklerine göre oluşturulan dosyalara ve ayrıca kullanıcı etkinlikleri nedeniyle güncellenen işletim sistemi dosyalarına yapay nesneler denir. Dijital adli tıp araştırmacıları, olayları doğrulamak için bu eserleri toplar ve sonuç olarak bu eserleri, bir olaydan sonra Kimin Ne, Ne Zaman, Nerede ve Neden yaptığını belirlemek için yanıtlar bulmak için kullanırlar.

Bu tezde, aşağıdakilerden eserler üzerine araştırma yapmayı planlıyorum: Zamanlayıcı Görevleri, Sistem Kaynağı Kullanım İzleyicisi, Win32 Hizmeti, Sıra Numarası Günlüğünü Güncelle, ve Windows LogFile, Windows İşletim Sistemlerinde (Windows XP, Windows 7 ve Windows 10).

Bu tezin amacı, bu eserlerin nasıl yaratıldığını, hangi bilgileri içerdiklerini, bu eserlere göre zaman çizelgesinin nasıl oluşturulacağını, bu eserlerin nerede bulunup konumlandırılacağını ve olay müdahalesi ve dijital araştırmalar sırasında ne kadar önemli olduklarını göstermektir.

Anahtar kelimeler: Zaman Çizelgesi Analizi, Windows Artefaktları, Olay Yanıtı, Dijital Adli Tıp

TABLE OF CONTENTS

TABLES

FIGURES

ABBREVIATIONS

APT	:	Advanced Persistent Threats
CPU	:	Central Processing Unit
DARPA	:	Defense Advanced Research Projects Agency
DF	:	Digital Forensic
ESE	:	Extensible Storage Engine
GUI	:	Graphical User Interface
GUID	:	Globally Unique Identifier
IR	:	Incident Response
MFT	:	Master File Table
NTFS	:	New Technology File System
OS	:	Operating System
PC	:	Personal Computer
RAT	:	Remote Access Trojan
RE	:	Reverse Engineering
SID	:	Security identifier
SRUM	:	System Resource Usage Monitor
SSID	:	Service Set Identifier
UsnJrnl	:	Update Sequence Number Journal
UUID	:	Universally Unique Identifiers

1. INTRODUCTION

Chapter one exhibit the introduction to this thesis, this chapter encloses the background on Timeline Analysis based on Critical Windows Incident Response Artifacts, the scope, and significance outlining why Timeline Analysis based on Critical Windows Incident Response Artifacts has a remarkable value to incident response and digital forensics domain. Four research questions were asked, and the limitation and delimitations for the present thesis are also listed.

1.1. BACKGROUND

Throughout history, Microsoft started slowly but surely to gain control over the market share of the personal computer industry (PCs) and expended its territory to the enterprise market. Microsoft gains more and more users over time, one of the most important reasons is its user-friendly features compared to its adversaries, which led to building an enormous fan base over the coming years until our present times.

Over a dozen versions of Windows operating system (OS) were released, including the current version, Windows 10 client, launched in July 2015, and Windows Server 2019 for server enterprise environment launched in October 2018.

Microsoft Windows became and stayed a dominating presence in the desktop operating system market since its debut in 1985. Despite a slight loss in market share in recent years, Windows operating system remains the frontrunner with a towering share of 77.74 percent, with Apple's macOS trailing as a distant second (Liu 2020).

Thus, Microsoft is globally well-known and more used than Linux and Apple's MAC OS X, therefore it is targeted more often by cybersecurity attacks.

Taking this into account, Incident Response (IR) and Digital Forensics (DF) Business concentrate their work and research on Microsoft windows artifacts to target the massive market share of Microsoft users for more financial profit, on the other hand, incident response

and digital forensics as science are closely related to cybersecurity, and right now there is and there will be a high demand on cybersecurity skills in these fields, "A recent survey by the International Data Corporation (IDC) revealed that organizations are prioritizing cybersecurity skills" (Day 2020), therefore choosing this topic for the thesis Timeline Analysis Based On Critical Windows Incident Response Artifacts", has a vast impact on my academic and future career achievement.

1.2. SCOPE

"Computer intrusions are more complex than ever before" (Luttgens et al. 2014, p.30), and when an incident happens, the incident response team needs to act swiftly, and knowing when an application or a file was executed is very critical in this case.

Providing this kind of information to IR team to perform timeline analysis on fingerprints left by users while using a different kind of applications on windows system needs a deep understanding on certain parts of system features, files created based on user activities and also operating system files updated due to user activities are called artifacts also known as Artifacts, providing all the information related to an artifact is part of a Digital Forensics research/investigator job, without researching and the understanding critical windows artifacts, investigating an incident will be a difficult task.

In this thesis I am going to refer to the selected windows artifacts that are crucial to the IR and DF investigation, I researched artifacts from: Scheduler Tasks, System Resource Usage Monitor (SRUM), Update Sequence Number Journal (UsnJrnl), Windows LogFile, Win32 Service.

Windows Operating System desktop clients are considered in this thesis, where tests are performed on Windows XP, 7, and 10.

1.3. SIGNIFICANCE

In addition to the academic enrichment, findings in this thesis will also help Digisecure, a digital forensics software development company based in Istanbul-Turkey, in their research

for critical windows artifacts. Understanding how each artifact is generated inside a Windows operating system and its structure helps in developing a suitable parser for that artifact. Artifacts are critical in real world-digital forensics investigations.

1.4. RESEARCH QUESTIONS

The primary goal of the proposed thesis is to answer the following questions:

a. Where is the artifact in the specific Windows operating system?
b. How is the artifact created by the specific Windows operating system?
c. What is the structure of the artifact?
d. How relevant is the artifact to digital forensics?

1.5. LIMITATIONS

a. In this thesis, I aim is to cover as many artifacts as possible
b. Deep understanding is required to research Windows operating system artifacts
c. Detailed information on some artifacts are rare to find online
d. In some cases, there is not even released information about the inner details of artifacts
e. Companies who create commercial digital forensics software which includes artifact parsers do not share critical research information for the artifacts they support
f. Microsoft itself does not share detailed information on some of its internal components
g. On top of my research, I demonstrated practical usage of timeline analysis based on three commercial digital forensics tools namely Artifast v4.0.10, developed by Digisecure, Axiom v3.9.0.18130, developed by MagnetForensics, finally I used ANJP v3.11.07, developed by Triforec in addition to FTK Imager v4.3.0.18, developed by AccessData
h. Tools used in this thesis will also include Virtual box v6.0, 010 Editor v10.0.2, Registry Explorer v1.5.2.0, DCode v5.1, XML Explorer v4.0.5.0, and ESEDatabaseViewer v1.65

1.6. SUMMARY

Chapter one presents the background of the timeline analysis based on critical windows artifacts, then it introduces the scope of this thesis and its significance, detailing why timeline analysis based on critical windows artifacts is important in the domain of digital

forensics. The current thesis included four questions that provide answers to them: (1) Where the artifact is located in the operating system? (2) How the artifact is being created by the Windows operating system? (3) What is the structure of the artifact? (4) How relevant is the artifact to digital forensics? The limitations and delimitations of the current research study have also been listed.

The next chapters included the literature review where I went through the following fields:(1) Incident Response, (2) Digital forensics, and (3) Reverse Engineering those three fields are used to research windows critical artifacts, after that, I spoke about how did I use each one of them in Data and Method to get to the required results in the Findings, finally in the Conclusion I discussed the results and future research.

2. LITERATURE REVIEW

To comprehend the topography of the literature and the appropriate methodology for this thesis, it is substantial to inspect research from the following bodies of literature: Incident Response, Digital forensics, and Reverse Engineering.

2.1. INCIDENT RESPONSE

The Computer Emergency Response Team Coordination Center (CERT/CC) is the first Incident Response team, it was created back in 1988 by Defense Advanced Research Projects Agency (DARPA), in response to the first internet worm attack, which was launched by Robert Tappan Morris in 1988.

Worms use facilities of an operating system that meant to be automatic and invisible to the user (Forristal 2001, p. 21).

This new kind of attack and its influence on the internet, plus the considerable challenges in coordinating a response in such a distributed environment led to the creation of a new cybersecurity field named Incident Response.

Incident Response has been described by Prosise et al. (2003) as:
We define a computer security incident as any unlawful, unauthorized, or unacceptable action that involves a computer system or a computer network. Such action can include any of the following events: Theft of trade secrets, Email spam or harassment, Unauthorized or unlawful intrusions into computing systems, Embezzlement, Possession or dissemination of child pornography, Denial-of-service (DoS) attacks, Tortious interference of business relations, Extortion, any unlawful action when the evidence of such action may be stored on computer media such as fraud, threats, and traditional crimes (p. 44).

The incident response aims to respond to security events as quickly as possible to contain, mitigate, and or prevent these events from occurring in the future to limit damage and reduce recovery time and costs, in today world IR team has become the backbone of the largest

government and enterprise organization, thus having an incident response capability has become increasingly a mandate regulation by the governments in the last years.

Computer and computer assisted incidents are nothing new. As soon as a new technology is developed, malicious people will find ways to abuse it (Rajnovic 2011, p. 22).

Taking this into account, incident response team need to be up to date when it comes to following the latest developments in this field, this requires a deep understanding of the new attack methods used by threat actors for example hackers, where they use Microsoft windows feature in their advantage to go under the radar without being detected.

Hacker incidents require a somewhat different response than do virus incidents. Some hackers are highly skilled, employ sophisticated techniques, and will go to great lengths to avoid being detected. To complicate matters further, a hacker can also be someone working for an organization (an insider) engaging in after-hours illegal activity, such as unauthorized access to sensitive information or perhaps password cracking. Whether they originate from the inside or outside, all hacker incidents need to be addressed as real threats to organizational computer systems (Schweitzer 2003, p. 195).

In my thesis I have researched artifacts from the most critical windows features that are related to attack incident launched by threat actors, for example using at.exe which is a genuine Windows tool to create a scheduled task, running this tool will leave fingerprints in different places in Windows one of these places is the registry, hence using at.exe tool is an indicator for a lateral movement technique.

Lateral movement is the act of the adversary moving from one system to another, inside your environment to expand their influence and access throughout the network. This is an area where the adversary will often spend a lot of time and during which we have a good opportunity to detect and respond to their attack; however, doing so requires bringing together the various skills (Anson 2020, p. 345), this technique is used by threats actors to launch Advanced Persistent Threats (APTs).

As it is mentioned above, incident response team must be versatile in term of skills and knowledge needed, to be able to handle incidents, digital forensics knowledge is essential in

these security events, in addition to that, there is a close similarity between both fields incident response and digital forensics, especially when it comes to steps taken to deal with incident, where every incident requires the collection, storage, and analysis of digital evidence, thus understanding digital forensic science is the next step in my thesis to undercover critical artifacts from Windows operating system features.

2.2. DIGITAL FORENSICS

In this section of the thesis I am going to give a glance on the history of digital forensics, its status nowadays, its definition and how it helps researchers in extracting artifacts from Windows operating systems, these results about artifact will be useful in investigating security events or digital crimes.

The world of crime is a complex place. Crime takes places everywhere and increasingly on the internet, "Criminal investigations that involve personal computer systems and their use for intrusion into other computers" (Parker 1976, p. 68), this was the first description of using digital information to investigate and prosecute crimes committed with the assistance of a computer, which was done by Donn Parker in his book Crime by computer in 1976 (Pan 2020).

The field of digital forensics is comparatively modern, while its history is short, however, in less than 3 decades, digital forensics has prospered from the seed of a thought to its current state, a complex and evolving field.

Digital Forensics now is no longer a linear process focused on recovering data, for example, the forensic analysis process has been identified by Eoghan Caseyas following: "The forensic analysis process involves taking factual observations from available evidence, forming and testing possible explanations for what caused the evidence, and ultimately developing deeper understanding of a particular item of evidence or the crime as a whole" (Casey 2010, p. 21), as it is mentioned above it is no longer a linear process, however, it is an evidence-based knowledge management process that will be incorporated into the investigation, intelligence analysis, information security, and electronic discovery, where there is career digital forensic educators and researchers in addition to practitioners and managers.

The definition of Digital forensics has been described by Dan Farmer as:
"Gathering and analyzing data in a manner as free from distortion or bias as possible to reconstruct data or what has happened in the past on a system" (Farmer et al. 2004, p. 193).

which simply can be summarized by the following: digital forensics is a science and process of collection, preserving, analyzing, and reporting legally admissible evidence to the court.

Building on the above definition Ayman Shaaban clarified that "the goals of the digital forensics, as a whole, are to answer these questions: What happened to the system under analysis? How was it compromised? During the analysis too, the analyst could answer some other questions based on their findings, such as the following: Who is the attacker? When did it happen? Where did it happen? Why did it happen? "(Shaaban et al 2016, p. 10).

Answering these questions to achieve the goals of digital forensics requires the reconstruction of data and knowing what happened in the past on an operating system, however, first the digital evidence needs to be acquired then analyzed, by undertaking specific methods and procedures similar to the one applied in a crime scene.

While most of the methods used in digital forensics are considered to be well-understood, operating systems are constantly changing and evolving and there may be new methods of data acquisition based on new ways of storing data. When filesystems change, for example, there will be new methods for accessing the information from the file system (Messier 2016, p. 6).

Methods and procedures that are applied to digital evidence are critical to any security event investigation, any alteration of digital evidence is not acceptable, as this will lead to a loss of credibility of the digital evidence, now let's briefly enumerate digital forensics investigation procedure:

a. Evidence Acquisition
b. Evidence Preservation
c. Evidence Analysis
d. Evidence Presentation

All of these procedures put to use on host-based evidence, and this is for the following reason as explained by Gerard Johansen:
"Host systems are far too often the target of malicious actions. They represent a possible initial target to gain a foothold in the network, pivot point, or the goal of threat actors. As a result, incident response analyst should be prepared to investigate these systems. Modern operating systems such as Microsoft Windows makes a number of changes during the execution of an application, changes to files, or the addition of user accounts" (Johansen 2017, p. 75).

Thus the acquisition of the evidence is performed on a host system in this thesis, and it will be locally however there are other variations of acquisition depending on the security event and this can be remotely, online and or offline acquisition in addition to locally which has been mentioned at earlier.

Preserving is done once evidence is acquired, where it is important to protect it from any sort of alteration, this can be achieved by creating a copy of the original evidence then creating another working copy for examining and analysis. This is because in some cases investigator can't have access to the original evidence for a long time and its copy will be used for future reference in case the working copy is corrupted or altered, it is noticeable to add that chain of custody is applied, were documenting the action taken on the digital evidence is mandatory in this stage event, even if case is the closed or solved.

Then digital forensics researcher in the next phase will choose what tool and digital forensics technique to use in extract the information which is in this case artifacts from the working copy evidence, next the digital forensics researcher will start analyzing the digital evidence, in this thesis I will only cover the windows operating system platform therefore digital forensics researcher needs to have the desired technical and investigation skills to extract the related artifacts from windows operating system internal and or features keeping in mind Microsoft constantly keep its OS up to date, these updates comes as new version of windows with new features and software or new patches.

Digital forensics researcher needs to understand what happens in case of a security event, here is where he or she needs to reconstruct the data and know what happened in the past on the OS, to answer the questions: Who is the attacker? When did it happen? Where did it

happen? Why did it happen? to achieve the goal of digital forensics investigation, however, because of the constant update on the software and feature of the OS, this task will be difficult to complete without understanding the internal parts of Windows OS, in most cases Microsoft Windows do not share in details all the information needed to complete the task, in that case, digital forensics researcher will have to rely on Reverse Engineering skill to extract the needed artifact information from the digital evidence.

Finally, the results are presented to the court, or in this thesis is delivered to the developers where they used it to create up-to-date investigation tools, which eventually will be used in the digital investigation.

2.3. REVERSE ENGINEERING

When there is no documented information on the structure of a windows internal feature then Reverse Engineering (RE) skills are the last resort for digital forensics researcher to analyze extracted data of digital evidence.

Eldad Eilam defined Reverse Engineering as:
Reverse engineering is a process where an engineered artifact (such as a car, a jet engine, or a software program) is deconstructed in a way that reveals its innermost details, such as its design and architecture. Reverse engineering is the process of extracting the knowledge or design blueprints from anything man-made. The concept has been around since long before computers or modern technology, and probably dates back to the days of the industrial revolution. Reverse engineering is usually conducted to obtain missing knowledge, ideas, and design philosophy when such information is unavailable. In some cases, the information is owned by someone who isn't willing to share them. In other cases, the information has been lost or destroyed (Eilam 2005, pp. xxiv-3-4).

This definition applies to many parts of Windows operating system features, Microsoft only documented some information thus it is either poor or incomplete, other important information for example to the structure of files and the meaning of its data is unknow. Other parties, like research companies interested in this field, will not share their knowledge, because of competition and business reasons.

The concept of reverse engineering by reconstructing data is applied in all the steps of digital forensics this is often when dealing with unknown parts of the operating system. Learning such techniques will help us to dive deep into any program, application, or file binaries.
The goal is very simple and can be summarized in one question as explained by Walter Bell: "Reverse engineering is the process of asking "how did they do that?" and then trying to do it yourself." (Bell 2007, p. 4).

Digital forensics researchers can apply different types of reverse engineering when investigating a case. Depending on what investigation we are dealing with and on what kind of data in the evidence, in my thesis, I am using file format reverse engineering also known as Binary File Format Analysis.

Having binary file format analysis skills under your belt is essential knowledge for reverse engineering, for several reasons as demonstrated by Andreas Pehnack:
"There are plenty of reasons why you could care about all the bits and bytes in binary files. Almost all files the average computer user reads and writes with applications like word processors, audio recorders, or video editing software, are binary files. However, computer specialists often need to dig deeper and want to extract, modify or simply understand the contents of such files" (Pehnack 2015, p. 3).

For example, when the digital forensics analyst is facing unknown data files, have no access to related documentation then it will be necessary to know the binary format of this unknow data. The understanding level must be to the bits and bytes level in binary files, this will help in extracting, modifying, and grasp the content of such files.

2.4. SUMMARY

Chapter two provides a review of the literature relevant to the Timeline Analysis based on the Critical Windows Incident Response Artifacts thesis. I demonstrated related fields to my thesis as follows: Incident response, digital forensics, and reverse engineering. Each one of these fields has a vast impact on extracting the needed artifact from the Windows operating system.

The next chapter outline the methodology for the current thesis, which examines the extraction of artifacts from critical Windows operating system Incident response artifacts, using different forensics tools, in addition to details on specifications for virtual environments used in this thesis.

3. DATA AND METHOD

Chapter three provides the methodology used in my thesis. As demonstrated in the introduction and literature review, Windows desktop has the most significant market share up to 77.74 percent globally compared to other operating systems.

Threat actor's primary target Windows operating system to set a foothold inside the networks and into the systems of a targeted environment. Incident response knowledge helps in identifying critical Windows operating system features that are used or targeted by threat actors.

Offensive actions on Windows operating system using its features will leave fingerprints, thus files created based on user activities and also operating system files updated due to user activities are called artifacts.

Then digital forensics knowledge is used to acquire evidence on the incident, extract artifacts from Windows operating system feature and finally report the results to the incident response team and/or developer team to solve the case and/or create a parser for these artifacts.

3.1. RESEARCH QUESTIONS

The primary objective of the proposed thesis is to answer the following questions:

a. Where is the artifact located in the specific Windows operating system?
b. How is the artifact created by the specific Windows operating system?
c. What is the structure of the artifact?
d. How relevant is the artifact to digital forensics?

The answer to these questions depends on the digital forensics knowledge after acquiring the specific data from evidence to extract the needed artifact based on the incident response selected critical Windows operating system feature.

In my thesis, I am trying to provide as much information as possible when extracting the artifact. This information is related to location, creation, structure of the artifacts and its affiliation to digital forensics. Finally, reverse engineering knowledge was used to understand undocumented Windows features.

The answer to the above questions helped in extracting artifacts from the following Windows operating system features/tools:

1. Scheduler Tasks
2. System Resource Usage Monitor
3. Win32 Service
4. Update Sequence Number Journal
5. Windows LogFile

3.2. RESEARCH DESIGN

3.2.1. Research Environments

To research for critical windows operating system artifacts, several virtual environments were set up as recommended by Ali and Meghanathan (2011). Virtualization provides multiple traits, like resource efficiency and low costs to carry out a configuration of several operating system environments.

The approach that I chose for my thesis is to create my virtual testing environment since I am researching specific features in the Windows operating system. I chose to use VirtualBox over VMware Workstation for two reasons, first Licensing Models for VirtualBox is free opensource software and second VirtualBox performance is similar to VMware as it was indicated by Amir Aghel Masjedi “Based on the results it was observed that, VMware Workstation and VirtualBox performed close to each other” (Masjedi 2012, p. 111).

VirtualBox Graphical User Interface 6.0.24 r 139119 (Qt5.6.2) was installed on Windows host workstation and used as a virtualization client. A total of four environments were set up. The first two virtual environments were set up for Windows 7, the second Windows 10 virtual environment was set up for, and the final virtual environment was set up for Windows

XP. All operating systems in the virtual environment updated to the latest version possible. The hardware and software are discussed next.

3.3. HARDWARE AND SOFTWARE SPECIFICATIONS

3.3.1. Windows Host Workstation

The physical host workstation for the Windows environments was an ASUS G750JH-DB71 with the following specifications:

a. CPU: Intel(R)Core (TM) i7-4700HQ CPU (4Core/6MB L3 Cache) @ 2.40GHz up to 3.4 GHz
b. RAM: 4X8GB DDR3/1600 Dual Channel Memory
c. Hard Drive: 1 TB
d. GPU: GeForce GTX 780m
e. OS: Windows 7 Professional SP 1

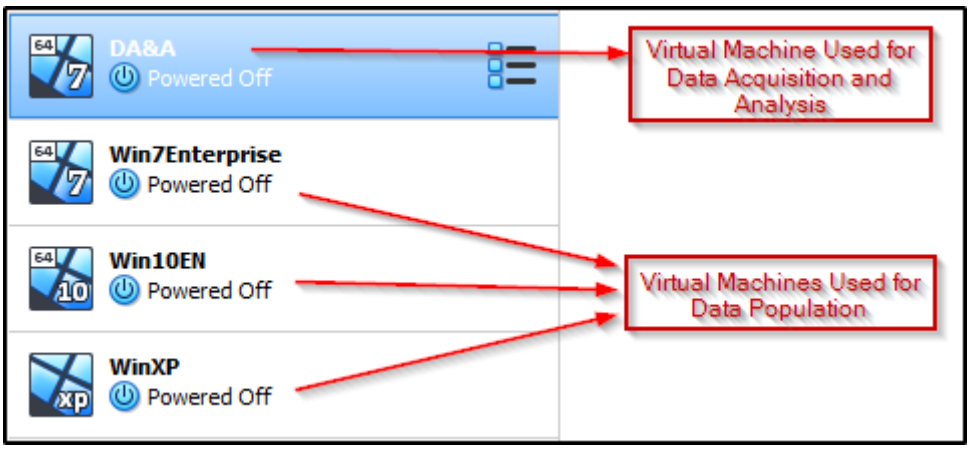
3.3.2. Windows Virtual Environment

There were four different virtualization environments for Windows operating system. All virtual machines had their entire disk space dynamically allocated. I used three flavors of the Windows operating system as follows:

a. First two virtual machines: Windows 7 Enterprise edition Evaluation 64bit SP1, with a disk space of 70 GB for each one of them
b. Second virtual machine: Windows 10 Enterprise edition Evaluation 64bit, Build 18362.19h1_release.190318-1202 with a disk space of 70 GB
c. Third virtual machine: Windows XP Professional Version 2002 Service Pack 3, disk space of 10 GB

I used one Windows 7 virtual machine as a primary machine to do all the data acquisition and analysis by installing FTM Imager, Artifast, and Axiom on it, I called DA&A, the reset of virtual machines I used them to populate the required data for the research as you can see in the below Figure 3.1.

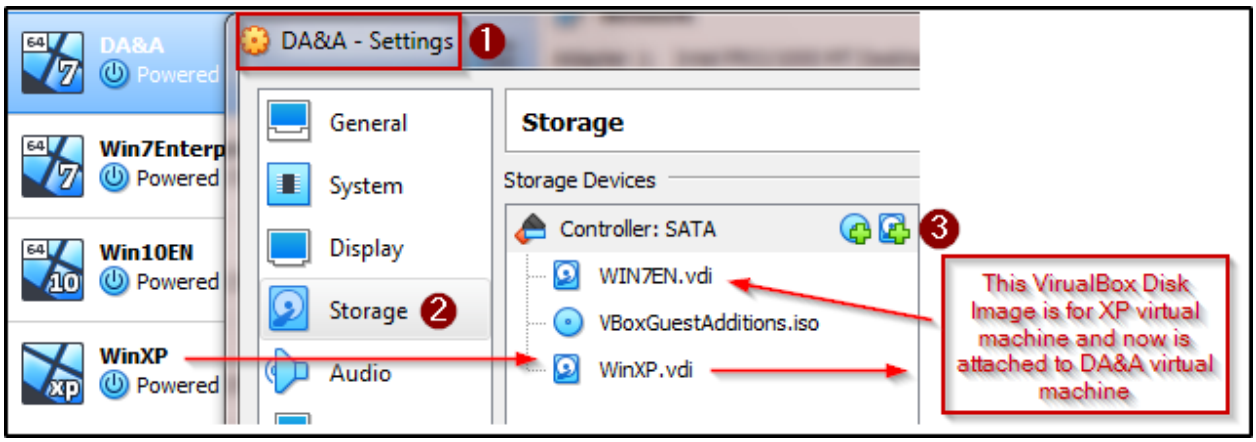
Figure 3.1: Virtual machines usage for acquisition, analysis and population of data



3.4. ACQUISITION OF DATA

The acquisition of data took place after the population step by using the Windows operating system tools on each of the selected artifact. Data was acquired from each virtual environment using FTK Imager version 4.3.0.18. Once data was populated for each Windows operating system tool the required data file was exported using FTK Imager. First, I attached the VirtualBox Disk Image of the virtual machine that contain the populated data to the acquisition and analysis DA&A virtual machine, which FTK Imager was installed on it, by following these steps: (1) access DA&A settings, then (2) navigate to Storage, and (3) press to add target Hard Disk as it is shown in the below Figure 3.2.

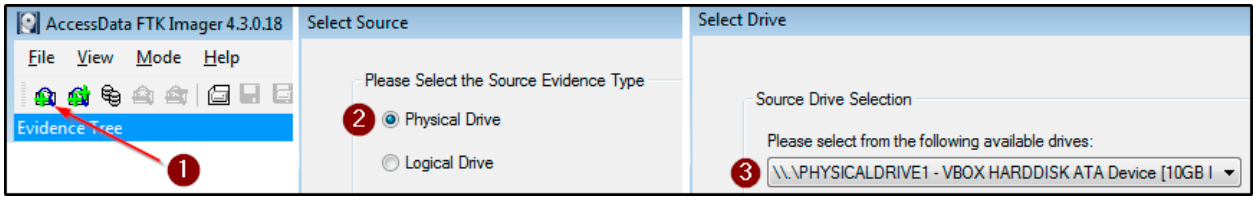
Figure 3.2: Data acquisition steps



After running DA&A virtual machine launch Computer management tool and to make sure that the newly added hard disk is online then Launch FTK Imager and I performed the

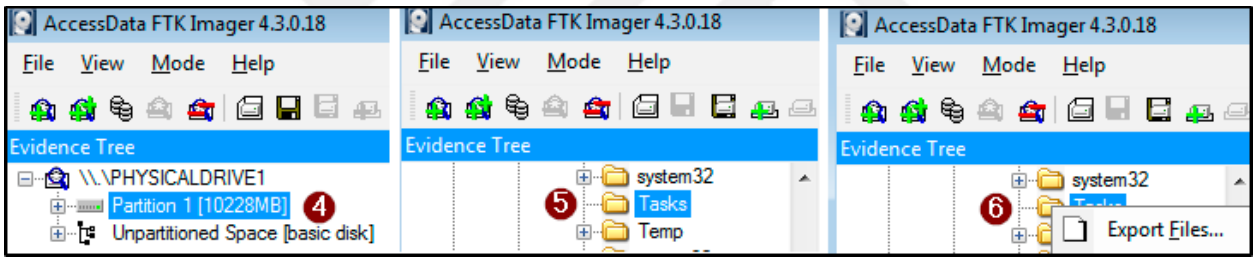
following steps to add the evidence and extract the needed data for it, (1) hit Add Evidence Item, (2) select Physical Drive and hit, (3) choose the data population Hard Drive as you can see in Figure 3.3.

Figure 3.3: FTK Imager Adding Evidence



The next step is to (4) navigate through the evidence to (5) the desired location and finally (6) right-click to export the needed data to the evidence folder in the DA&A virtual machine as presented in Figure 3.4.

Figure 3.4: Using FTK Imager to export needed data from evidence



The Same steps are performed on the other populated data and by this the acquisition of data is complete.

3.5. POPULATION OF DATA

Before Acquiring any data and perform analysis on critical windows incident response artifacts, first, a population of data had to take place. This was accomplished simply by using the tools of Windows operating systems which will produce or create and leave user's footprints. Files created based on user activities and operating system files updated due to user activities are called artifacts. The following Table 3.1 which I created based on my findings show how to populate these artifacts:

Table 3.1: Tools used to populate data

Critical Windows Incident Response Artifacts	Tools used to populate Data (Artifacts)
Task Scheduler	Task scheduler GUI Schtasks.exe At.exe
System Resource Usage Monitor	Automatically generated by Windows 10 SRUM extensions: -Windows Network Data Usage Monitor -WPN SRUM Provider -Application Resource Usage Provider -Windows Network Connectivity Usage Monitor -Energy Usage Provider
Win32 Service	Automatically Generated by any version of Windows operating system using: -Services.exe
Update Sequence Number Journal	Automatically Generated by Windows operating system, Windows versions include 7 8 and 10, in addition to that by creating restore point. In case of Windows XP this feature is disactivated by default and needs to be activated by using fsutil tool.
Windows LogFile	Automatically Generated by Windows operating system

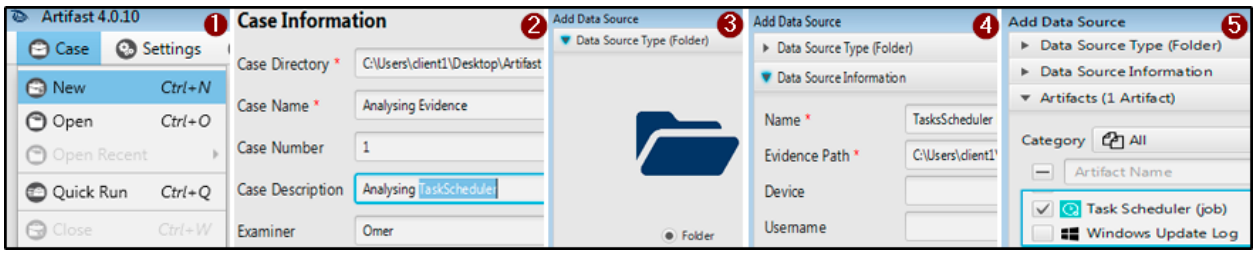
More details on the population of data of each artifact in the Finding Chapter where I explained in this thesis how each artifact can be created by the user when he or she use Windows operating system.

3.6. FORENSIC ANALYSIS OF DATA

There are several digital forensics tools that can be used to perform analysis on the evidence collected in the previous data acquisition step, in this thesis I chose Artifast v4.0.10, developed by Digisecure which is a local Turkish company located in Istanbul-Turkey, because I have the privilege and permission to use and test their software while working with their developers, and I chose Axiom v3.9.0.18130, developed by MagnetForensics, an industry-standard tool in the computer forensics field.

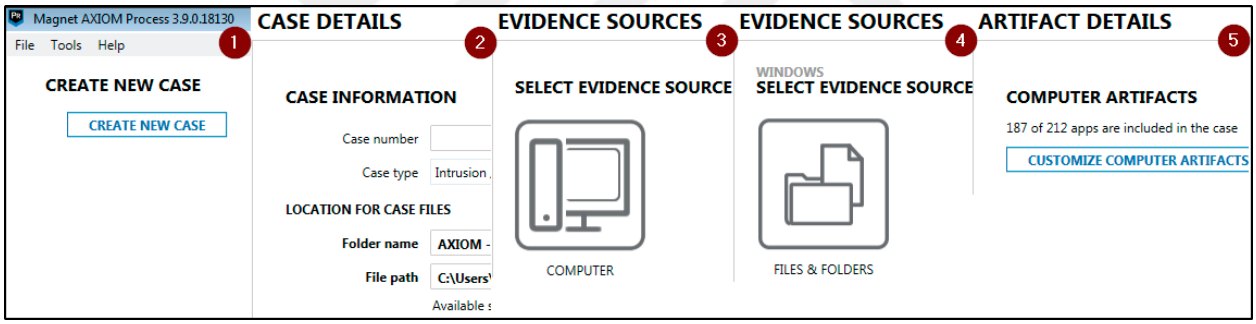
The following steps was used to perform analysis on evidence by Artifast: (1) Create New case, (2) Enter Case information, (3) Add Data Source Type in this thesis I worked with Folder type in most of the cases, however, there are other types that can be used such as L01, E01, and Raw Image, (4) Add Source Information, and the final step number (5) was choosing the targeted Artifact then press run as it is given in the below Figure 3.5: Steps used to analyze evidence by Artifast.

Figure 3.5: Steps used to analyze evidence by Artifast



The following steps were used to perform analysis on evidence by Axiom: (1) Create New case, (2) Enter Case Details, (3) Choose Evidence Source, (4) Choose Evidence source type, for Axiom I went with Folder in my thesis in most cases, however, there are other types like EnCaseImages E01, FTK Images AD1, Raw Images .dd, Virtual Machine Images .vdi, DMG Images .dmg and Archives .zip and finally in step (5) I chose the target artifact and then press run, as it is presented in the next Figure 3.6.

Figure 3.6: Steps used to analyze evidence by Axiom



I used other tools like 010 editor which serve as a hex editor to analyze the contents of any file. I used it to examine binary files, registry keys, and other types of files as needed. Registry Explorer was used to examine and analyze evidence from the registry, DCode was used to decoding Timestamp Data and lastly, XML Explorer was used to show the structure of XML files in a tree from.

Results using these mentioned software's are presented for each artifact in the findings chapter.

3.7. SUMMARY

Chapter three presented an overview of the research methodology of the thesis. This included the thesis questions and design. This chapter summarized the research environments, involving the hardware and software specifications for the host and four virtual environments.

The population of data was done using Windows operating system tools to create the needed artifacts. Forensics acquisitions were made of the virtual machines and then I used FTK imager, Artifast, and AXIOM to perform the digital forensics analysis on the acquired data. In the next chapter, I am going to demonstrate the outcome of my thesis on each artifact extracted after using Windows operating system tools.

4. FINDINGS

4.1. ARTIFACTS

The thesis covers Artifact extracted from the following Windows XP/7/8/10 internals:

1. Scheduler Tasks
2. System Resource Usage Monitor (SRUM)
3. Update Sequence Number Journal ($UsnJrnl)
4. Windows LogFile ($LogFile)
5. Win32 Service

4.1.1. Task Scheduler

Windows Vista, Windows 2008, and Windows 7 systems manage scheduled tasks a bit differently from previous versions of Windows. Starting with Windows Vista, Microsoft introduced Task Scheduler 2.0 (Carvey2012, p. 138).

Several Windows components are implemented as services, such as the Print Spooler, EventLog, Task Scheduler, and various networking components (Solomon et al. 2012, p. 75).

In this thesis, I am going to demonstrate the location, creation, and the structure of task scheduler Artifact in addition to its relevance to digital forensics.

The task scheduler is an application that is delivered with every Windows operation system which allows in advance defined actions to be automatically executed whenever a group of terms is met. It was first introduced in Windows 95 as System Agent then renamed to Task Scheduler in internet explorer 4.0 and windows 98, where the windows event log must be running before Task Scheduler starts up. Since Windows Vista Task Scheduler is running based on Microsoft Management Console (MMC) and tasks can be created using three tools:

a. The GUI Windows Task Scheduler
b. Schtasks.exe (command line tool)
c. At.exe (only used as command line tool)

Tasks can be scheduled to execute:

a. At a specific time/time on daily/weekly/monthly/monthly day-of-week schedule
b. When a specific system event occurs, when computer enters idle state, when task is registered, when system is booted, when user logs on, when a terminal server session changes state

All three task-scheduling tools can work locally or remotely against other hosts.

4.1.1.1. Location of artifact

In my research, I have found that Task Scheduler artifacts are located in the following areas within the Windows registry and files, and I have noticed that there are differences between the old flavor of windows the legacy XP and the new Windows 7 (out of support in 2020) and Windows 10 as follows where I am listing the type of files used for task scheduler and its location:

Windows XP task scheduler locations as it can be seen in Figure 4.1, Figure 4.2, and Figure 4.3:

a. .job files are in C:\Windows\Tasks
b. Registry: HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Schedule
c. SchedLgU.txt, this log file is in C:\Windows

Figure 4.1: Windows XP task scheduler folder and .job file type

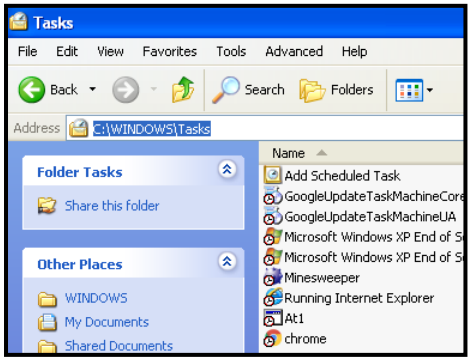


Figure 4.2: Windows XP task scheduler location in registry

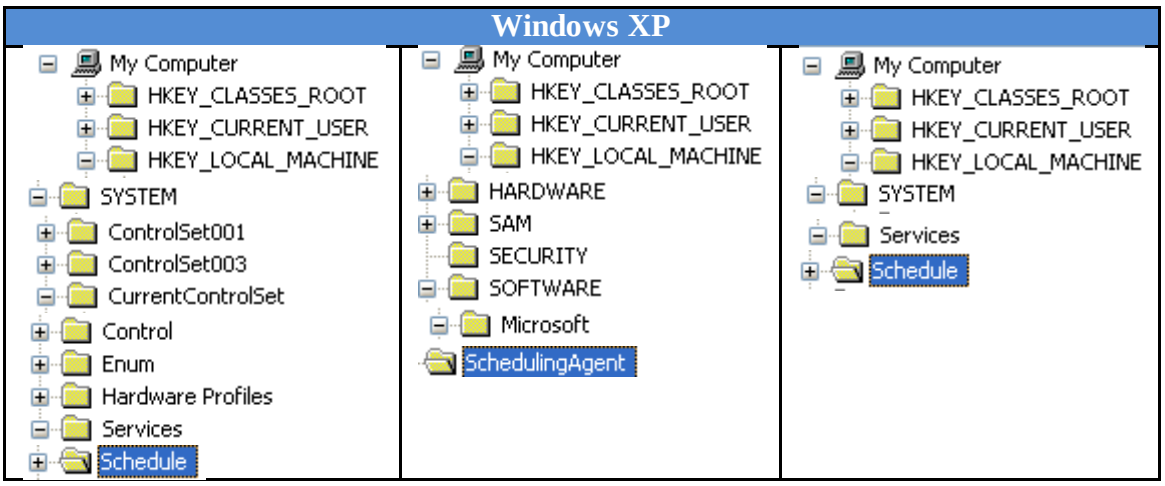
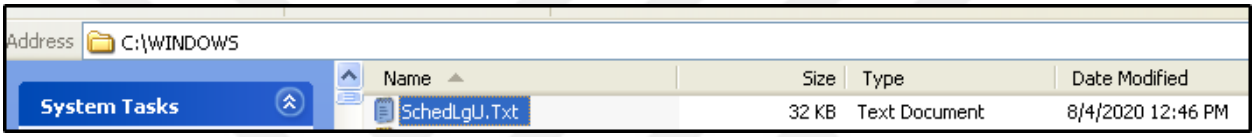


Figure 4.3: Task scheduler log file location



Windows 7/10 task scheduler location as seen in Figure 4.4, Figure 4.5and Figure 4.6:

a. .job and XML files are located in C:\Windows\System32\Tasks and C:\Windows\Tasks

b. .job and XML files are located in Registry

- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\Taskcache\Tasks
- HKLM\Software\Microsoft\Windows NT\CurrentVersion\Schedule\Taskcache\Tree

Figure 4.4: Windows 10, task scheduler folder

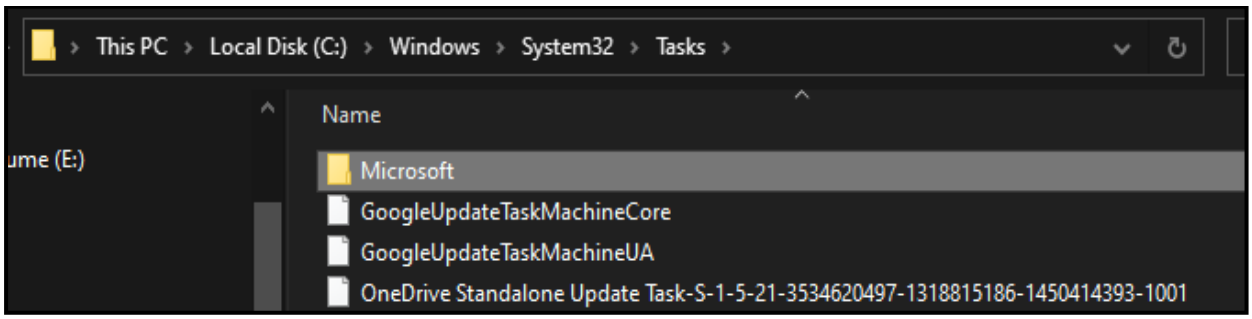


Figure 4.5: Windows 7, task schedueler folder

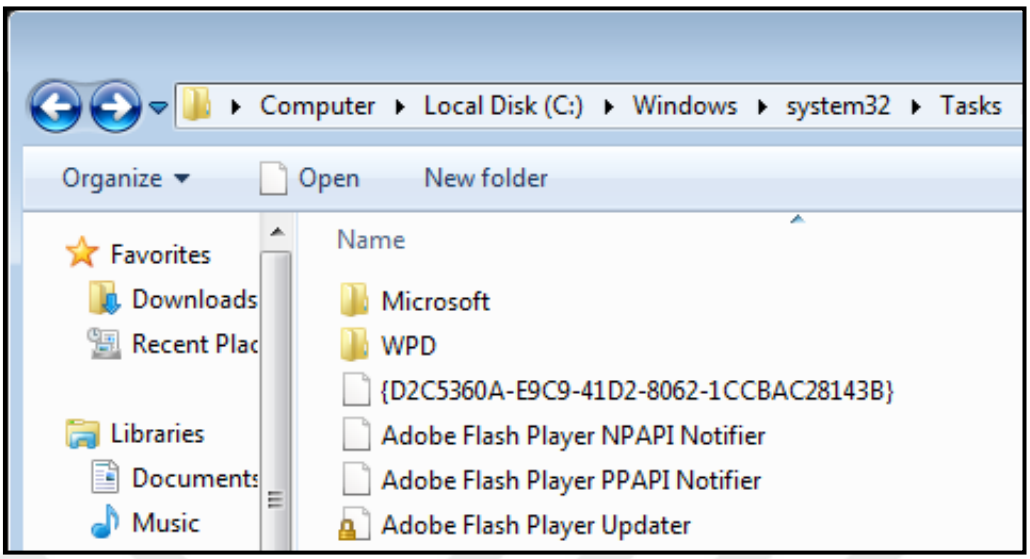
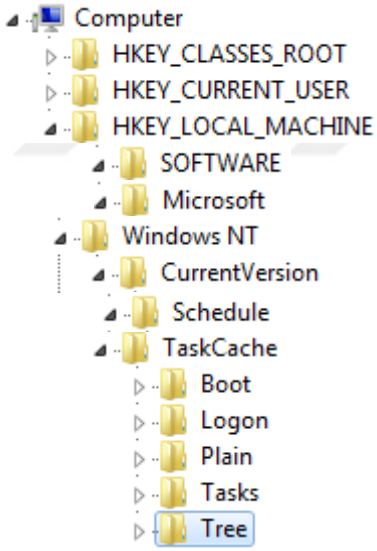
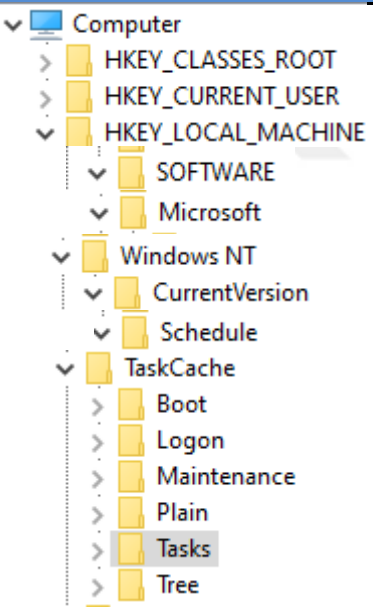


Figure 4.6: Windows 7/10 task scheduler registry location

Windows 7 Enterprise Evaluation sp1 64bit	Windows 10 Enterprise Evaluation 64bit
	

4.1.1.2. Artifact creation

I have found that depending on the Windows version the user can create Scheduled Tasks one of the following tools:

a. Graphical User Interface GUI / Microsoft Management Console available in Windows 7/10
b. Schtasks.exe (command line tool) available in Windows 7/10
c. At.exe (command line tool) available in Windows XP/7
d. Scheduled Tasks service available in Windows XP

4.1.1.3. Structure of artifact

I have found that when a user using Windows XP, all task scheduler created tasks are .job file type, this type of file is also used in Windows 7 if at.exe is used to create the task, whereas this type of file is not used anymore in Windows 10, both Windows 7 and 10 use XML file type for the tasks created using the GUI or schtasks.exe.

In the following paragraph, I will explain the difference between those two-file types starting with .job file format then XML format.

4.1.1.3.1. *Job file*

I have found that Windows .JOB file specifies the task configuration. A .JOB file consists of two main sections, (1) fixed-length, and (2) variable-length, all fields in the .JOB file format must use little-endian byte ordering unless otherwise stated. All extra padding bytes are a value of zero unless otherwise stated and all are ignored upon receipt.

I have found that the fixed length size is 68 bytes in total, and the binary structure is arranged as follows as it can be seen in the following Table 4.1, starting at offset 0 with a size of 2 bytes these initial bytes represents the product version like Windows XP or any other windows flavor as it is given in Table 4.2, offset 4 with a size of 16 bytes is for job Universally Unique Identifiers (UUID) or Globally Unique Identifier (GUID)next important offset is number 32 with 4 bytes in size which is the priority of the job, then offset 44 with a size of 4 bytes for .job status, after that we have offset 48 with 4 bytes size representing the flags and finally, the last important one is offset number 52 with a size of 2 bytes for system time.

Table 4.1: Fixed length binary map for .job files

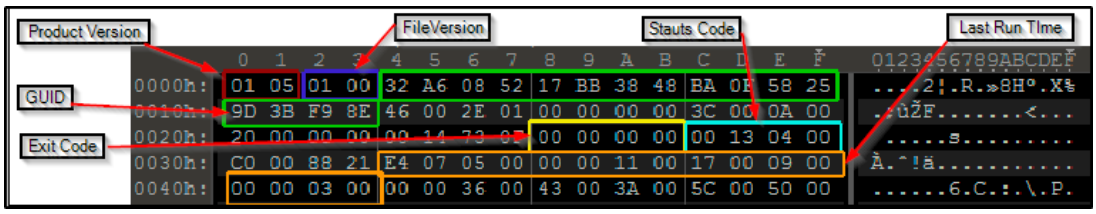
Offset Order	Size	Value	Description
0	2		Product version
2	2	1	File (format) version
4	16		Job UUID (or GUID)
20	2		Application name size offset, the offset is relative from the start of the file.
22	2		Trigger offset, the offset is relative from the start of the file.
24	2		Error Retry Count
26	2		Error Retry Interval
28	2		Idle Deadline
30	2		Idle Wait
32	4		Priority
36	4		Maximum Run Time
40	4		Exit Code
44	4		Status
48	4		Flags
52	16		Last run time, Consists of a SYSTEMTIME

Table 4.2: Product version

Offset	Size	Description	Type
0x0400	2	Windows NT 4.0	Products
0x0500	2	Windows 2000	
0x0501	2	Windows XP	
0x0600	2	Windows Vista	
0x0601	2	Windows 7	
0x0602	2	Windows 8	
0x0603	2	Windows 8.1	
0x0a00	2	Windows 10	

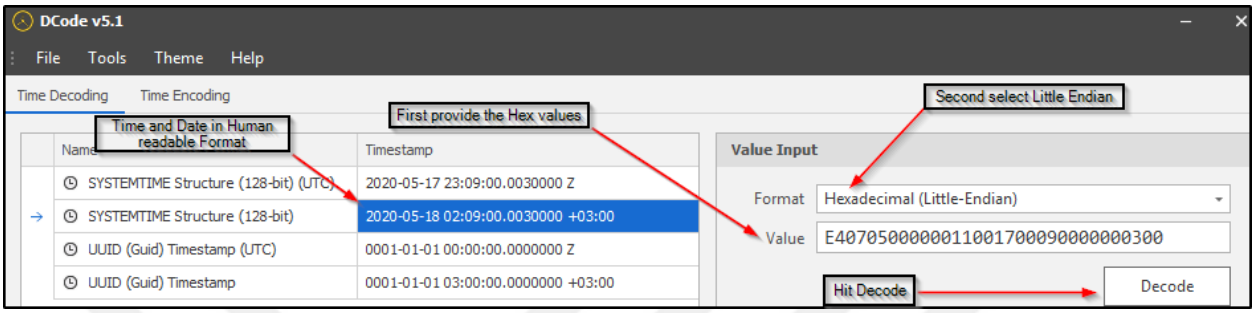
The following testing sample from the Windows XP .job file that I have prepared, shows how to apply the binary map on the offset of .job file using any Hex editors it is given in Table 4.1, we can get the following information: Product version which is in this case 0105 since all the hex are little Endian, reverse it to big Endian and this code will be 0501 for Windows XP as you can see in Table 4.2, File version is 1, GUID of task, Exit Code is 00000000 which means the task has completed successfully as it is shown in Table 4.4, Status Code is 00130400 little Endian and 00041300 in big Endian which means the task is ready to run at its next scheduled time and the at the end we have Last run time of the running task as shown in Figure 4.7.

Figure 4.7: Hex representation of Windows XP .job file using a Hex Editor



Using a time decoder we can get the last run time in a human readable format, so again by applying the binary map on our example, the time offset is located at offset 52 with a size of 16 bytes little Endian E4 07 05 00 00 00 11 00 17 00 09 00 00 00 03 00 and using the decoder the result will be 2020-05-18 02:09:00.0030000 +03:00 as seen in Figure 4.8.

Figure 4.8: Windows XP Decoding last run tine little endian to human readable



The Next example is for windows 7 .job files, keep in mind that this type of task scheduler job was create using the At.exe tool, this leads to creating a .job file similar in its binary structure to Windows XP .job file, as you can see in Figure 4.9, we can notice that Product version is 0106 little Endian and 0601 big Endian for windows 7 as you can see in Table 4.2, by applying the same technique to this case we can get the rest of the information need for the investigation.

Figure 4.9: Hex representation of Windows 7 .job file using a Hex Editor

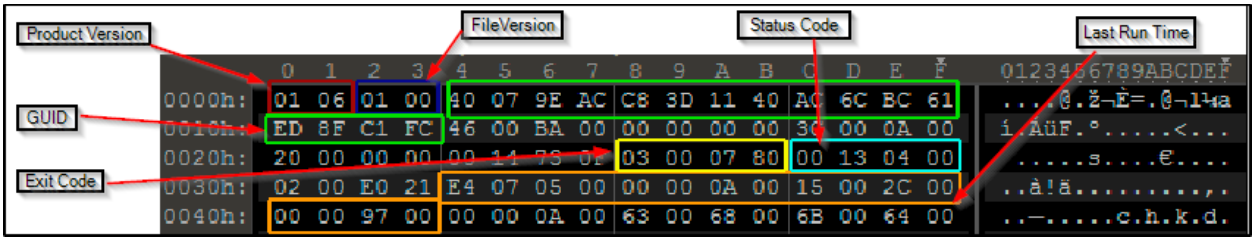
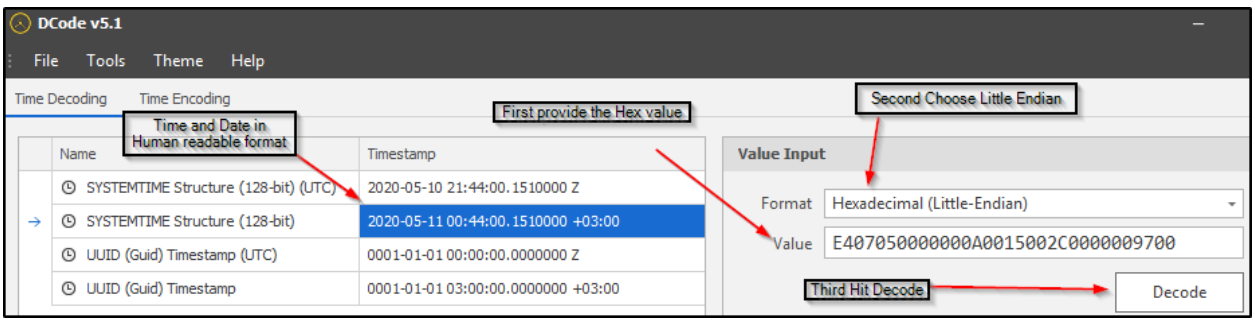


Figure 4.10:Windows 7 Decoding last run time little endian to human readable



Exit code is 03000780 which means The system cannot find the path specified, status code 00130400 means Task is ready to run at its next scheduled time as it is mentioned in Table 4.4, and the Last run time in little Endian is E407050000000A0015002C0000009700 decoded to a human readable format to get 2020-05-11 00:44:00.1510000 +03:00 as you can see in Figure 4.10.

Variable length structure is variable in size and comes exactly after fixed length, it consists of the following of components, starting with offset 0 with size 2 bytes representing running instance count as it is given in Table 4.3, next at offset 2 we have the application name with variable size, the reset of the component have variable offset number and size and this applies for parameters, working directory, author, comment and user data. One more thing to add is that in these fields all data are stored as Unicode strings.

Table 4.3: Variable length binary map

Offset	Size	Description
0	2	Running Instance Count
2	...	Application Name Consists of a Unicode string
...	...	Parameters Consists of a Unicode string
...	...	Working Directory Consists of a Unicode string
...	...	Author Consists of a Unicode string
...	...	Comment Consists of a Unicode string
...	...	User Data

After running the task job in Windows XP, the output of this process will be stored in "SchedLgU.txt" which is the task scheduler log file that is provided in version 1.0. SchedlgU.txt size is 32 Kilobytes and in general, this log includes information about the task name, date and time when the task started, finished and the task process output in the form of code, as you can see in Figure 4.11, it is also worth mentioning that the same code is used within Task Scheduler GUI in Windows 7 and 10, I mention each code with its meaning as it is shown in Table 4.4 you can see the code numbers in the right column, and its description on the left column.

Figure 4.11: Schedlgu.exe file

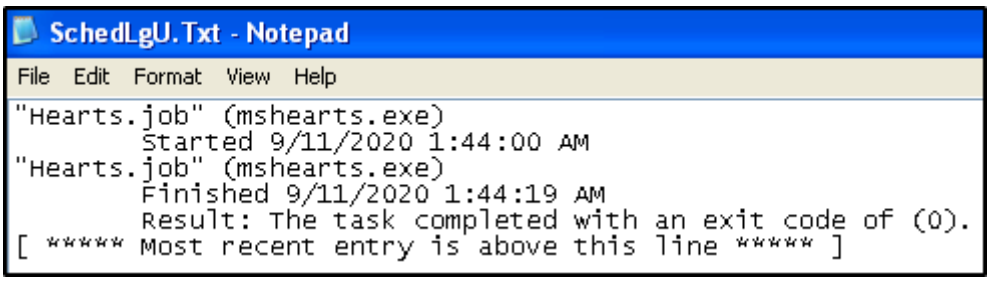


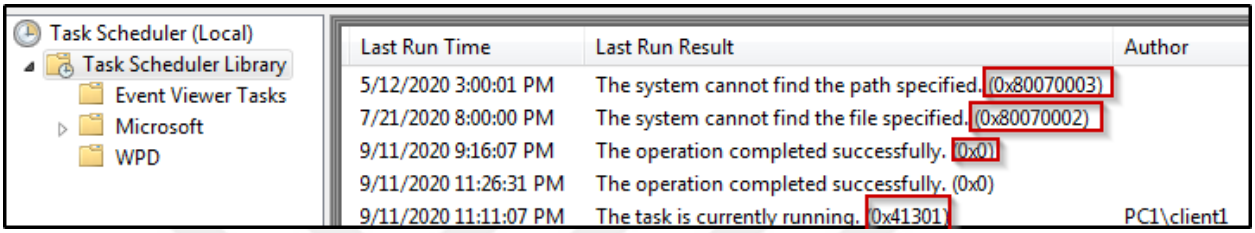
Table 4.4: Last result codes used in Windows XP log, Windows 7 and 10 Gui

Code	Description
0 or 0x0	The operation completed successfully.
1 or 0x1	Incorrect function called or unknown function called.
2 or 0x2	File not found.
10 or 0xa	The environment is incorrect.
0x00041300	Task is ready to run at its next scheduled time.
0x00041301	The task is currently running.
0x00041302	The task has been disabled.
0x00041303	The task has not yet run.
0x00041304	There are no more runs scheduled for this task.
0x00041305	One or more of the properties that are needed to run this task have not been set.
0x00041306	The last run of the task was terminated by the user.
0x00041307	Either the task has no triggers, or the existing triggers are disabled or not set.
0x00041308	Event triggers do not have set run times.
0x80010002	Call was canceled by the message filter
0x80041309	A task's trigger is not found.
0x8004130A	One or more of the properties required to run this task have not been set.
0x8004130B	There is no running instance of the task.
0x8004130C	The Task Scheduler service is not installed on this computer.
0x8004130D	The task object could not be opened.
0x8004130E	The object is either an invalid task object or is not a task object.
0x8004130F	No account information could be found in the Task Scheduler security database for the task indicated.
0x80041310	Unable to establish existence of the account specified.
0x80041311	Corruption was detected in the Task Scheduler security database
0x80041312	Task Scheduler security services are available only on Windows NT.
0x80041313	The task object version is either unsupported or invalid.
0x80041314	The task has been configured with an unsupported combination of account settings and run time options.
0x80041315	The Task Scheduler Service is not running.
0x80041316	The task XML contains an unexpected node.
0x80041317	The task XML contains an element or attribute from an unexpected namespace.
0x80041318	The task XML contains a value which is incorrectly formatted or out of range.
0x80041319	The task XML is missing a required element or attribute.
0x8004131A	The task XML is malformed.
0x0004131B	The task is registered, but not all specified triggers will start the task.
0x0004131C	The task is registered but may fail to start. Batch logon privilege needs to be enabled for the task principal.
0x8004131D	The task XML contains too many nodes of the same type.
0x8004131E	The task cannot be started after the trigger end boundary.
0x8004131F	An instance of this task is already running.
0x80041320	The task will not run because the user is not logged on.
0x80041321	The task image is corrupt or has been tampered with.
0x80041322	The Task Scheduler service is not available.
0x80041323	The Task Scheduler service is too busy to handle your request. Please try again later.
0x80041324	The Task Scheduler service attempted to run the task, but the task did not run due to one of the constraints in the task definition.

0x00041325	The Task Scheduler service has asked the task to run.
0x80041326	The task is disabled.
0x80041327	The task has properties that are not compatible with earlier versions of Windows.
0x80041328	The task settings do not allow the task to start on demand.
0x80070003	The system cannot find the path specified.
0xC000013A	The application terminated as a result of a CTRL+C.
0xC0000142	The application failed to initialize properly.

As you can see in the below Figure 4.12, both Windows 7 and 10 use the same last run result code in their GUI as like as window XP.

Figure 4.12: Last run codes in the GUI of windows 7 and 10



4.1.1.3.2. *Xml file*

XML file has a hierarchical format and can be translated as a tree structure, called an XML tree. It must contain a root element that is master of all other elements. All elements can contain subparts, text, and attributes. XML file demonstrates its structure as a tree which starts at the root element and divaricated to the lowest level of elements.

When the user creates a scheduled task using the GUI in Windows 7 or 10 an XML file is generated, it is a tree represented visually which can be is ASCII chart or a more visually complex hierarchy.

The task created will generate an XML file, that it contains the following tree branches, starting from the root is Task version which contains the version of task in use currently it is version 1.2, then followed by the main branches RegistrationInfo, Triggers, Principles, Settings, Actions Context ="Author" and Data, the final branch Data if it exist in the XML file it will contain data type as seen in Figure 4.13, we can notice that each branch include additional information related to the task created for example the Registration info contains date/time, Author information that has a great importance to a digital forensics investigation, other important information reside in triggers, principals, settings and Action as it is shown in Figure 4.14 and Figure 4.15.

Figure 4.13: Main xml file structure

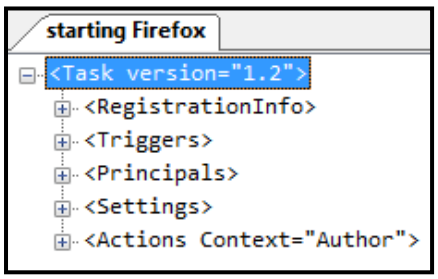


Figure 4.14: Xml file structure

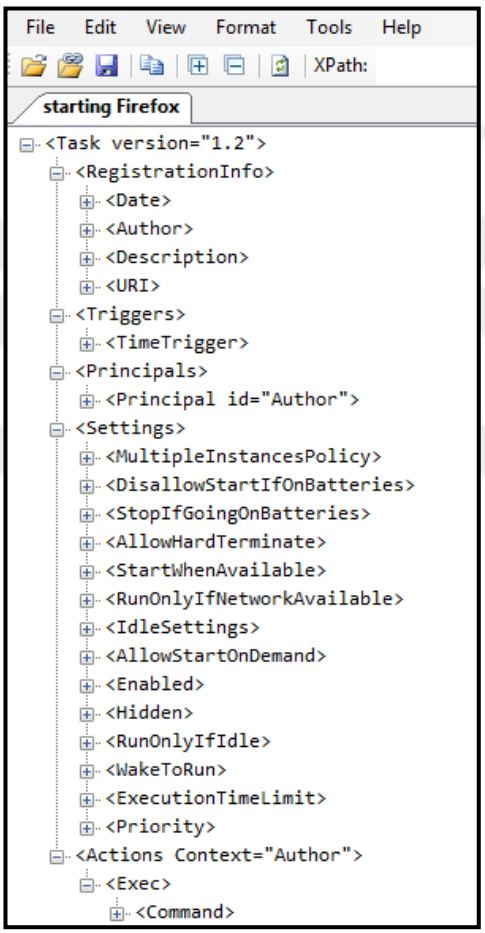
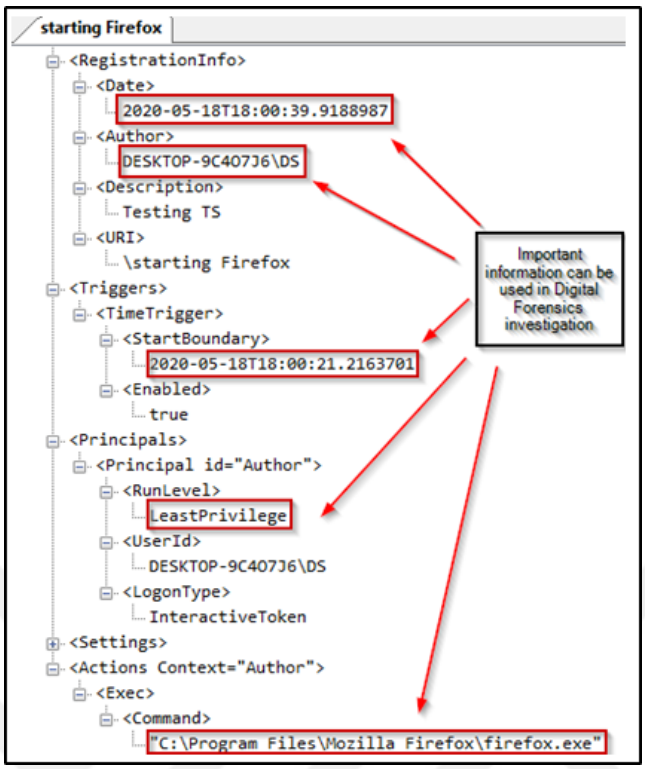


Figure 4.15: Structure of task schedule xml file



4.1.1.3.3. *Registry key, subkey and values*

Creating a task using the task scheduler in windows 7 and 10 will generate artifacts in the registry which also can be very valuable to a digital forensic investigation, these artifacts are created inside the registry key "Tree" and "Tasks", the artifact will take the name given by the user when creating the task for example "starting firefox" this is a subkey to "Mytask" and contain the following registry values Names: "Id", "Index" and "SD", as it is shown in Figure 4.16, the most important value name is "Id", it is a REG_SZ type which is a string, more specifically a UTF-16, and since it contains the GUID as its data, which is related for the task that has been started and finished, this GUID will help us navigate to the subkey inside "Tasks" as it is given in Figure 4.17. After Navigating through Task until we reach the specified GUID we will find the following value names: Author, Date, Description, Path, and Hash all of them are REG_SZ type which is a string UTF-16 except Hash which is a Reg_Binary type and this value Name is so valuable since it can be used to verify the integrity of XML task file.

Figure 4.16: Task GUID

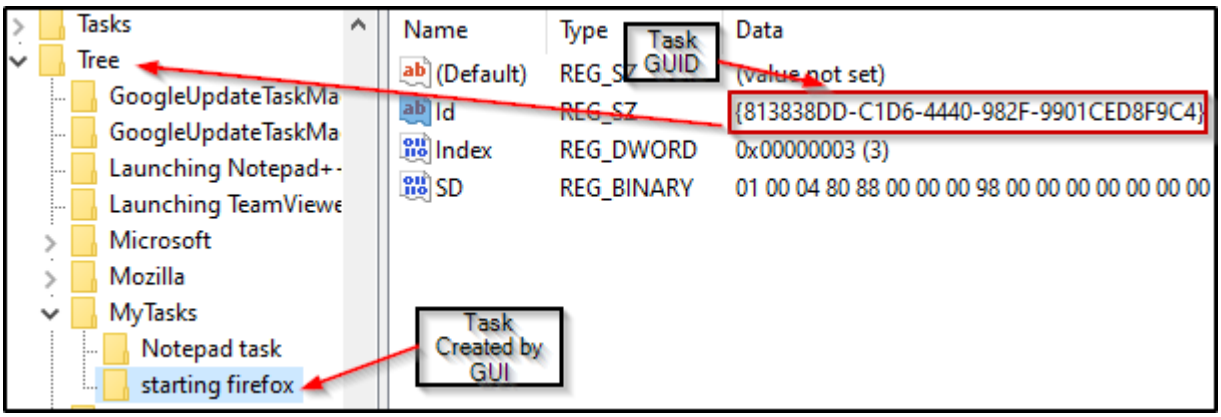
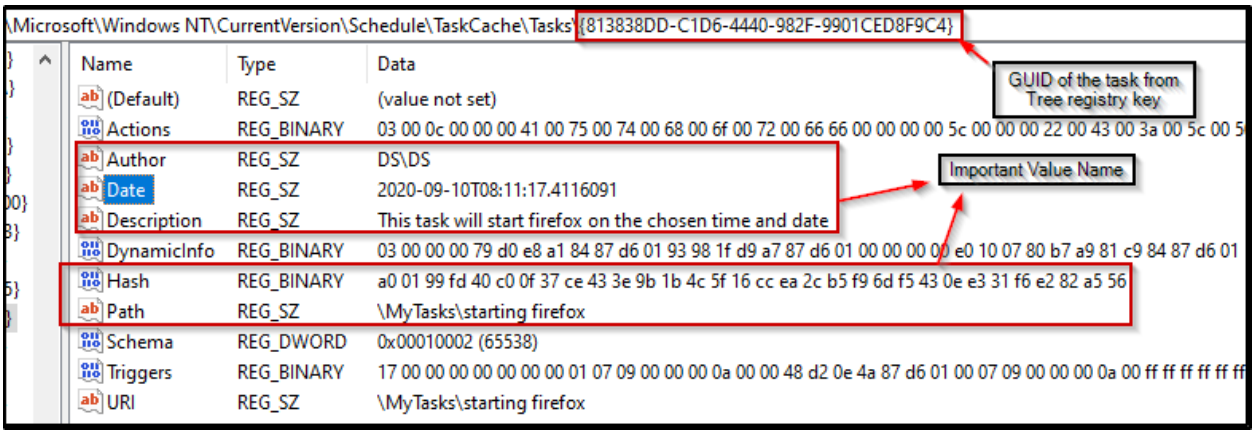


Figure 4.17: Navigating through subkey Tasks using GUID



4.1.1.4. Relevance to digital forensics

Digital Forensic investigator can get much critical information after analyzing Task Scheduler artifacts for the investigation and I can list the most important ones as follow:

a. Name of the executable file or application that is used to perform the task
b. Path of the file (location on Hard Drive)
c. Time and date of task creation, execution and its status (complete, canceled or failed)
d. Name of the user account used to create the task and the computer name used in that

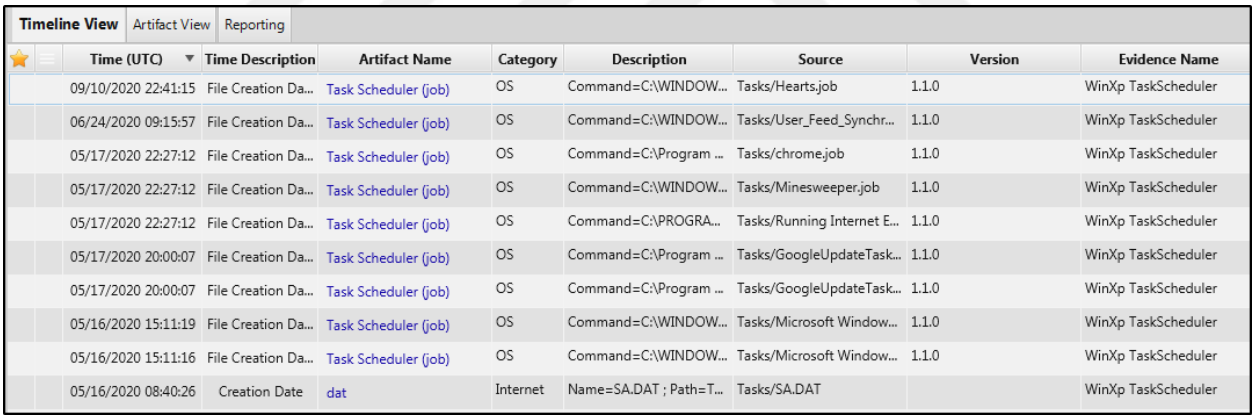
This information can help the digital forensics Investigator in detecting Lateral Movement techniques used by an intruder to move between the systems in a compromised infrastructure which allows them to use for example the freely available PSExec.exe tool to run attack commands.

4.1.1.5. Timeline analysis

After collecting this information from its location and analyzing it by the researcher as I did in this thesis, the results will be handed over to the developers where it will be used to create an application (Parser) to display the results related to the task scheduler artifact based on when they occurred in chronological order, in other words in a timeline format. The output will be analyzed by a digital forensics investigator to determine Who did What, When, Where, and Why.

In the following example, I have used "Artifast" which a professional digital forensics software which developed by Digiseure for digital forensics investigation, these results I gathered from the test experiments that I performed on windows XP testing operating system, where it shows all the task scheduler .job files in a Timeline format as it is shown in Figure 4.18.

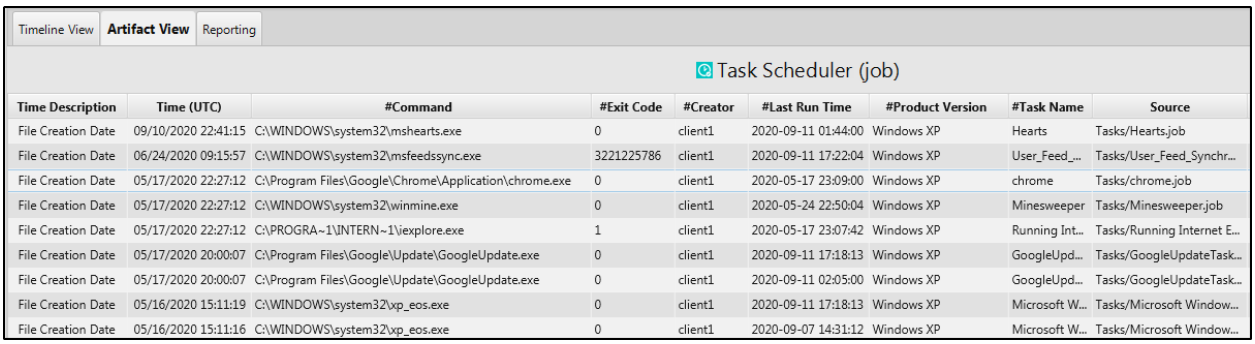
Figure 4.18: Timeline Analysis for .job file



Time (UTC)	Time Description	Artifact Name	Category	Description	Source	Version	Evidence Name
09/10/2020 22:41:15	File Creation Da...	Task Scheduler (job)	OS	Command=C:\WINDOW...	Tasks/Hearts.job	1.1.0	WinXp TaskScheduler
06/24/2020 09:15:57	File Creation Da...	Task Scheduler (job)	OS	Command=C:\WINDOW...	Tasks/User_Feed_Synchr...	1.1.0	WinXp TaskScheduler
05/17/2020 22:27:12	File Creation Da...	Task Scheduler (job)	OS	Command=C:\Program ...	Tasks/chrome.job	1.1.0	WinXp TaskScheduler
05/17/2020 22:27:12	File Creation Da...	Task Scheduler (job)	OS	Command=C:\WINDOW...	Tasks/Minesweeper.job	1.1.0	WinXp TaskScheduler
05/17/2020 22:27:12	File Creation Da...	Task Scheduler (job)	OS	Command=C:\PROGRA...	Tasks/Running Internet E...	1.1.0	WinXp TaskScheduler
05/17/2020 20:00:07	File Creation Da...	Task Scheduler (job)	OS	Command=C:\Program ...	Tasks/GoogleUpdateTask...	1.1.0	WinXp TaskScheduler
05/17/2020 20:00:07	File Creation Da...	Task Scheduler (job)	OS	Command=C:\Program ...	Tasks/GoogleUpdateTask...	1.1.0	WinXp TaskScheduler
05/16/2020 15:11:19	File Creation Da...	Task Scheduler (job)	OS	Command=C:\WINDOW...	Tasks/Microsoft Window...	1.1.0	WinXp TaskScheduler
05/16/2020 15:11:16	File Creation Da...	Task Scheduler (job)	OS	Command=C:\WINDOW...	Tasks/Microsoft Window...	1.1.0	WinXp TaskScheduler
05/16/2020 08:40:26	Creation Date	dat	Internet	Name=SA.DAT ; Path=T...	Tasks/SA.DAT		WinXp TaskScheduler

Pressing on Artifact view will show more detailed information in timeline format as you can see in Figure 4.19.

Figure 4.19: Artifact view will show more details



Timeline View | **Artifact View** | Reporting

Task Scheduler (job)

Time Description	Time (UTC)	#Command	#Exit Code	#Creator	#Last Run Time	#Product Version	#Task Name	Source
File Creation Date	09/10/2020 22:41:15	C:\WINDOWS\system32\mshearts.exe	0	client1	2020-09-11 01:44:00	Windows XP	Hearts	Tasks/Hearts.job
File Creation Date	06/24/2020 09:15:57	C:\WINDOWS\system32\msfeedssync.exe	3221225786	client1	2020-09-11 17:22:04	Windows XP	User_Feed_...	Tasks/User_Feed_Synchr...
File Creation Date	05/17/2020 22:27:12	C:\Program Files\Google\Chrome\Application\chrome.exe	0	client1	2020-05-17 23:09:00	Windows XP	chrome	Tasks/chrome.job
File Creation Date	05/17/2020 22:27:12	C:\WINDOWS\system32\winmine.exe	0	client1	2020-05-24 22:50:04	Windows XP	Minesweeper	Tasks/Minesweeper.job
File Creation Date	05/17/2020 22:27:12	C:\PROGRA~1\INTERN~1\iexplore.exe	1	client1	2020-05-17 23:07:42	Windows XP	Running Int...	Tasks/Running Internet E...
File Creation Date	05/17/2020 20:00:07	C:\Program Files\Google\Update\GoogleUpdate.exe	0	client1	2020-09-11 17:18:13	Windows XP	GoogleUpd...	Tasks/GoogleUpdateTask...
File Creation Date	05/17/2020 20:00:07	C:\Program Files\Google\Update\GoogleUpdate.exe	0	client1	2020-09-11 02:05:00	Windows XP	GoogleUpd...	Tasks/GoogleUpdateTask...
File Creation Date	05/16/2020 15:11:19	C:\WINDOWS\system32\xp_eos.exe	0	client1	2020-09-11 17:18:13	Windows XP	Microsoft W...	Tasks/Microsoft Window...
File Creation Date	05/16/2020 15:11:16	C:\WINDOWS\system32\xp_eos.exe	0	client1	2020-09-07 14:31:12	Windows XP	Microsoft W...	Tasks/Microsoft Window...

The next example will be from Windows 7 testing operating system however I am going to use "Magnet AXIOM" another digital forensics investigation software, parsing the evidence using this software will show task scheduler .job file and XML files in a timeline format, for a digital forensics investigator it will be very easy to answer the questions related to the investigation so, for example, a job name AT2.job was triggered to run chkdsk for driver D on 12/5/2020 by the system as you can see in Figure 4.20 and Figure 4.21**Error! Reference source not found.**.

Figure 4.20: Windows 7 task scheduler timeline Analysis parsing results

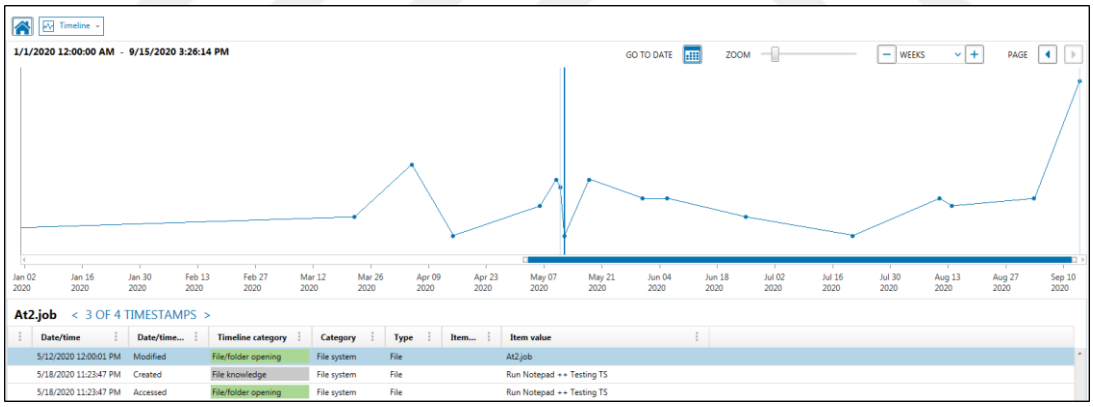
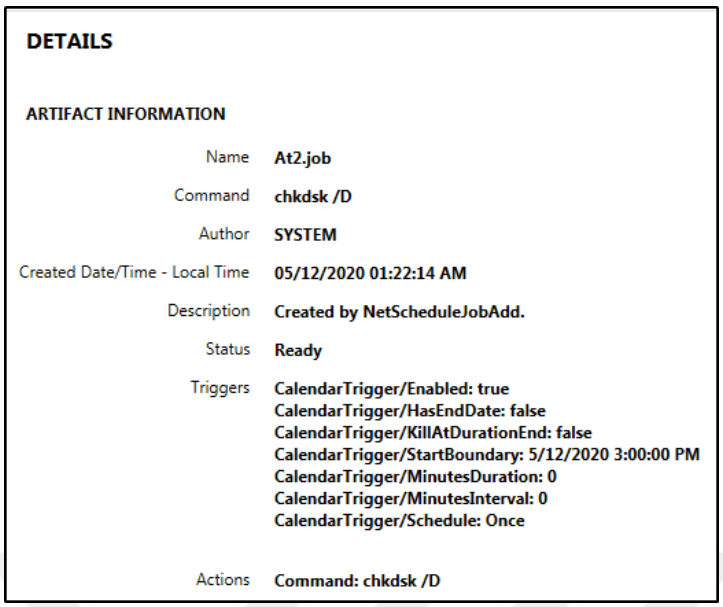


Figure 4.21: Windows 7 task scheduler detailed results

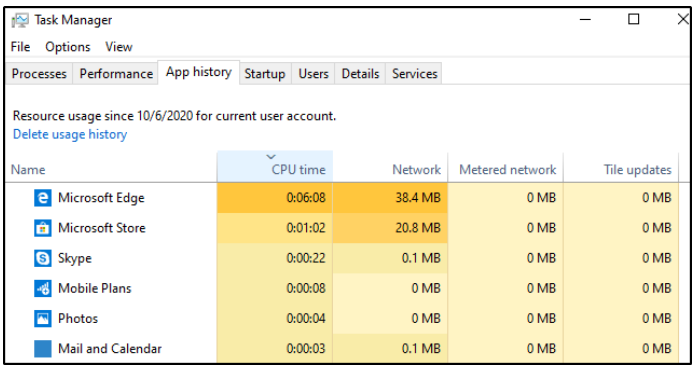


4.1.2. System Resource Usage Monitor

System Resource Usage Monitor (SRUM) was first seen in Windows 8 and is still exist in Windows 10. This is a new modern feature integrated within the new version of Windows operating system since Microsoft has pulled the plug on Windows 8 life support, thus in my thesis, I have decided to cover only Windows 10, besides, this feature is not available in the older versions such as Windows 7 and XP.

SRUM is a new technology that traces down user details, operating system resources usage, and network transactions, most of this information is hidden from the operating system end-users, thus end user only can access a fraction of this data using the App history tab in the Task Manager, this data consist of mainly Name of the application, CPU time and Network as it is demonstrated in Figure 4.22, however, it is stored all the other information in a database within the Windows 10 internals.

Figure 4.22: Portion of SRUM data shown in App history tab of Task Manager



Extracting information or artifacts from SRUM, will serve and provide digital forensics examiner an edge during a digital investigation. Artifacts stored within SRUM database helps in tracing down user activity and network processes. Digital forensics investigators will gain a historical view over the past events on the operating system by linking recovered data together.

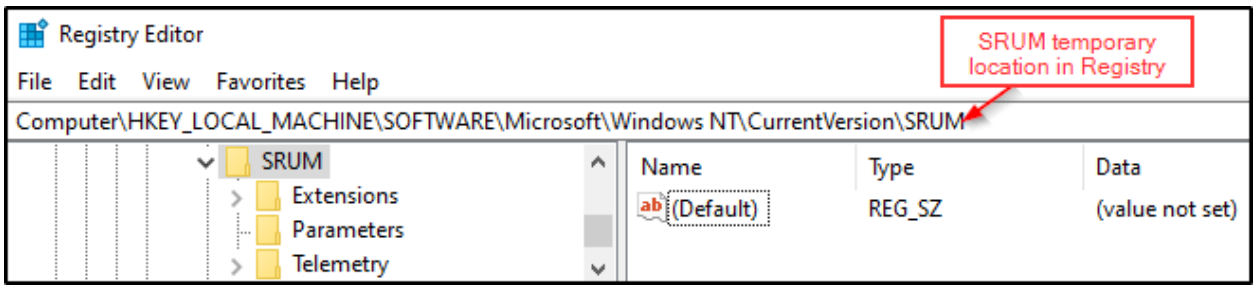
4.1.2.1. Location of artifact

In my research I have found that SRUM files are located only within Windows 8 and 10, and precisely there are two places, the first place is temporary inside Windows operating system registry key, and the last place is permanent within Windows Internals C drive folders as it is demonstrated in the following Figure 4.23 and Figure 4.24, here as you can see SRUM is stored inside "sru" folder inside System32 directory of Drive C:

a. /HKLM\SOFTWARE\Microsoft\WindowsNT\CurrentVersion\SRUM\
b. C:\Windows\System32\sru

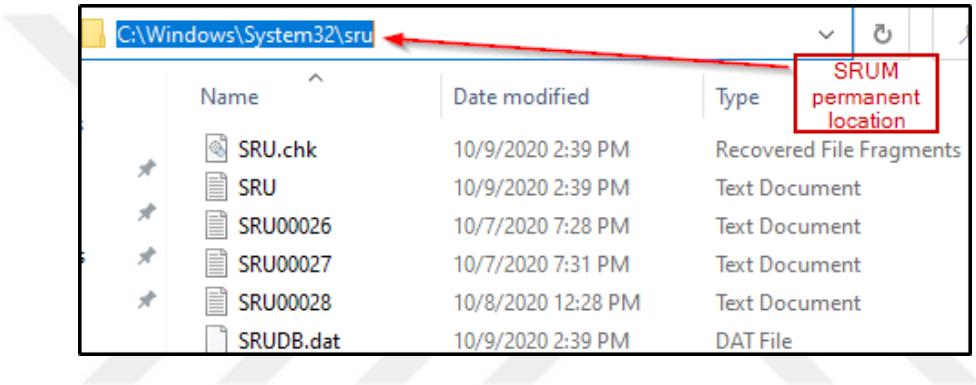
I have found that SRUM stores its temporary data in registry hive KEY_LOCAL_MACHINE inside SRUM subkey of CurrentVersion subkey of Windows NT subkey of Microsoft.

Figure 4.23: SRUM temporary location in registry



Here as you can see SRUM is stored inside sru folder inside System32 directory of Driver C.

Figure 4.24: SRUM permanent location



4.1.2.2. Artifact creation

SRUM is integrated with Windows operating system, meaning that it is part of the operating system once it is installed it will start functioning as soon as Windows end-user logs in as it is stated by Yogesh Khatri: This service is enabled by default and configured to start automatically upon system startup on all Windows versions, including Enterprise versions (Khatri 2015, p. 54).

What happened behind the scenes inside the Windows operating system while it is running, is that it uses several extensions to observe multiple operating system resources. These extensions include:

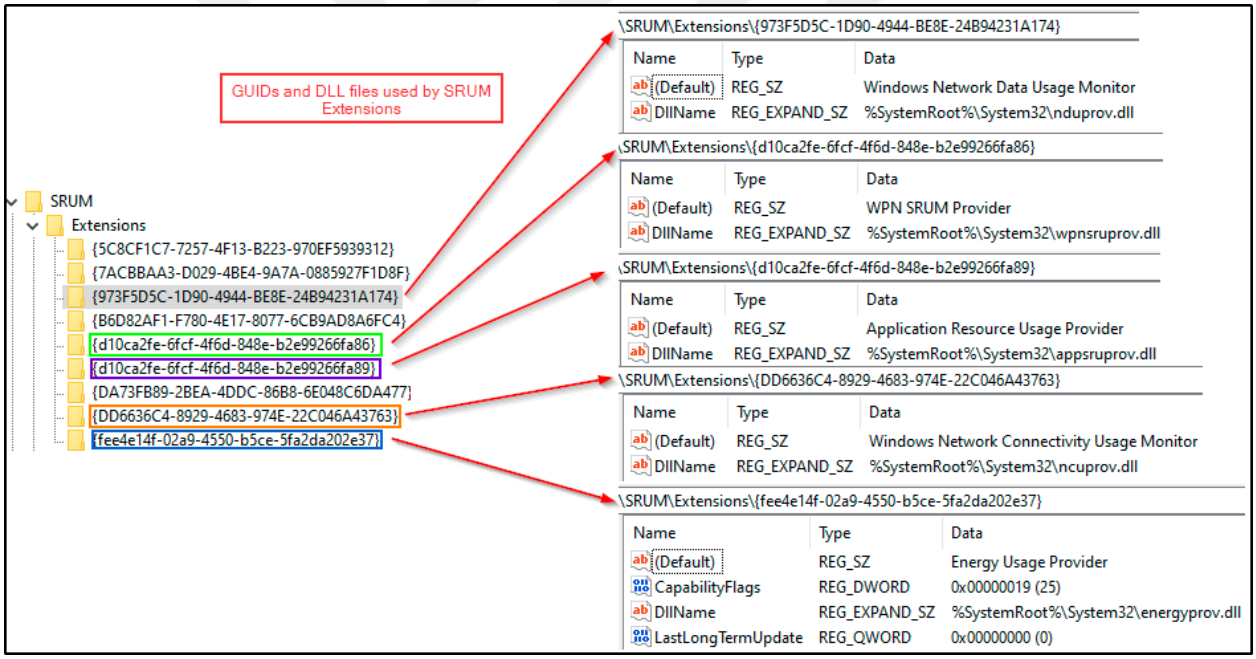
a. Windows Network Data Usage Monitor
b. WPN SRUM Provider
c. Application Resource Usage Provider
d. Windows Network Connectivity Usage Monitor

e. Energy Usage Provider

The following Figure 4.25 that I have prepared shows the main extensions used by Windows 10 SRUM in the registry as you can see each extension has its GUID with the associated DLL files from system 32 folder.

Data collected by Windows 10 SRUM are stored temporarily in these extensions, then it will be moved to from temporary location in the registry hive software to be stored permanently in Windows 10 system32 directory. Data are stored periodically, exactly every hour and on shutdown, permanent stored data considered to be a very critical artifact to any digital forensic examiner.

Figure 4.25: SRUM GUIDs and DLL files in Windows 10 operating system



The following subsection give more details about the type of data collected in both locations by Windows 10 SRUM feature.

4.1.2.3. Structure of artifact

4.1.2.3.1. *Registry SRUM subkey*

As mentioned above in this thesis, Windows 10 SRUM feature store data temporarily in registry hive Software inside SRUM subkey. this subkey SRUM uses number of extensions and its associated DLL files to collect data about user activity.

This registry artifact, as described by Antun Duracnec: Its records user's Security Identifier (SID) that executed the program. Furthermore, it records the names of all networks on which system has been connected, length of connection and how many bites were written to or read from the hard drive by the specific application. Its records application battery usage and Central Processing Unit (CPU) usage (Duracnec et al. 2019, p. 1417).

I have Created subsequent Table 4.5 SRUM that shows GUID, extensions name, and its associated DLL files with its registry type.

Table 4.5: Srum Guid, extensions name and associated dll files

GUID	Extension name	DLL files	Registry File type
973F5D5C-1D90-4944-BE8E-24B94231A174	Windows Network Data Usage Monitor	nduprov.dll	REG_SZ REG_EXPAND_SZ both are string UTF-16
d10ca2fe-6fcf-4f6d-848e-b2e99266fa86	WPN SRUM Provider	wpnsruprov.dll	
d10ca2fe-6fcf-4f6d-848e-b2e99266fa89	Application Resource Usage Provider	appsruprov.dll	
DD6636C4-8929-4683-974E-22C046A43763	Windows Network Connectivity Usage Monitor	ncuprov.dll	
fee4e14f-02a9-4550-b5ce-5fa2da202e37	Energy Usage Provider	energyprov.dll	

The collected information is saved to the file every hour and on operating system shutdown this imply that digital forensics analyzer will not know the precis time of application execution. This data then saved in registry then moved to database inside system32 directory.

4.1.2.3.2. *SRUDB.dat*

Extensible Storage Engine (ESE), also known as JET Blue, is a database engine from Microsoft that does not speak SQL (Talyor 2017).

Extensible Storage Engine (ESE) database is a technology developed by Microsoft, it is used in different Microsoft applications such as Exchange, Active Directory, and multiple features within Windows 10 operating system such as SRUM. SRUM use this technology to store all the information collected in registry by SRUM extensions into SRUDB.dat which is an ESE database file that contains multiple tables.

SRUM perform the same operation using extensions, GIUD and its associated DLL files to save the information on SRUDB.dat, as it was elaborated by Steve Anson: These same GUIDs are used as the names for the associated tables in the SRUM ESE database (Anson 2020, p. 324).

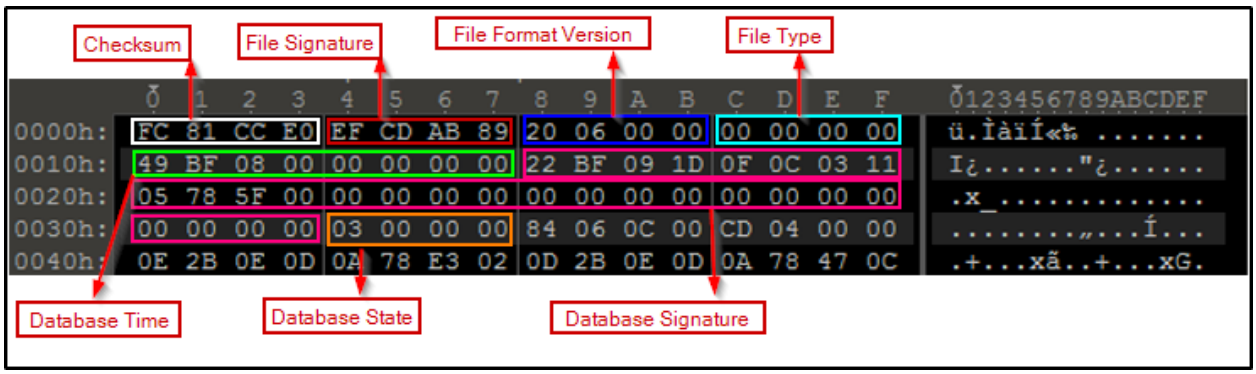
This artifact stores information that have been already mentioned in the above subsection and can be summarized in the following table.

Table 4.6: Type of data in Srudb.dat

Activity Name	Stored Data
Application activity	Full path to executable User Security identifier (SID) Volume of accessed data
Network Activity	Network connections Starting time / Duration Service Set Identifier (SSID) Data sent / received CPU time

The File structure of SRUDB.dat shows that is consist of two parts (1) File header and (2) Fixed size pages, the byte order inside SRUDB.dat is in little-endian, and database is stored inside pages. I used 010 editor tool to show the file structure of SRUDB.dat file header in Hexadecimal as you can see in the following Figure 4.26.

Figure 4.26: Srudb.dat file structure



I have created the below Table 4.7: Offset order of Srudb.dat with more details on the binary structure of the SRUDB.dat file and I have found that: checksum is at offset 0 with a size of 4 bytes, file signature is at offset 4 sizes of 4 bytes, file format version is at offset 8 with a size of 4 bytes, the file type is at offset 12 if its value is all zero then SRUDB.dat file contain database however if this value is 1 then SRUDB.dat contain the streaming file.

At offset 16 we have database time with 8 bytes size, then offset 24 to 51 is for Database signature which contains a random number, log and NetBIOS name, last but not least we have in this table database state at offset 52 with 4 bytes size, in this case it is 3 which means clean state.

Table 4.7: Offset order of Srudb.dat

Offset	Size	Description
0	4	Checksum
4	4	File Signature
8	4	File Format Version
12	4	File Type: 0 for database 1 for Streaming file
16	8	Database Time First 2 bytes: Hours Second 2 bytes: Minutes Third 2 bytes: Seconds Last 2 bytes: Padding
24	28	Database Signature Offset 0 random assigned number Offset 4 creation date and time Offset 12 NetBIOS computer name
52	4	Database State 1 was just created 2 require recovery 3 clean state 4 upgrading

In my thesis have chosen to use the ESE database viewer and Magnet Axiom to examine the content of the SRUDB.dat file, ESE database viewer shows artifacts from SRUM extensions which are stored in several tables as you can see in the below Figure 4.27 and Figure 4.28 we can see the content of Windows Network Data Usage Monitor table.

Figure 4.27: Srudb.dat Tables

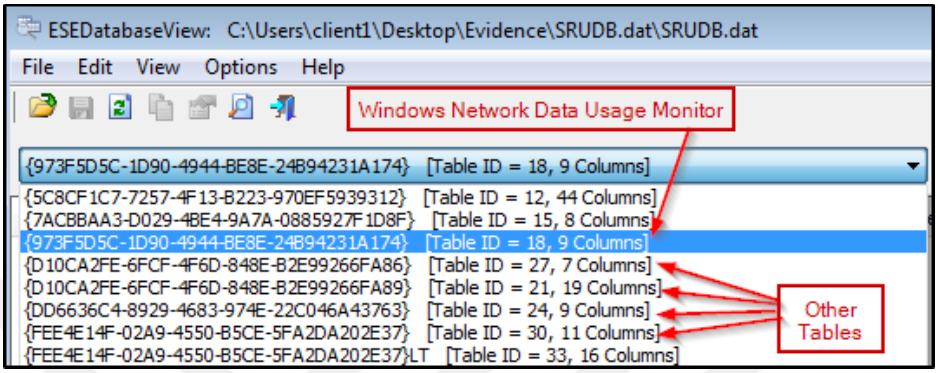
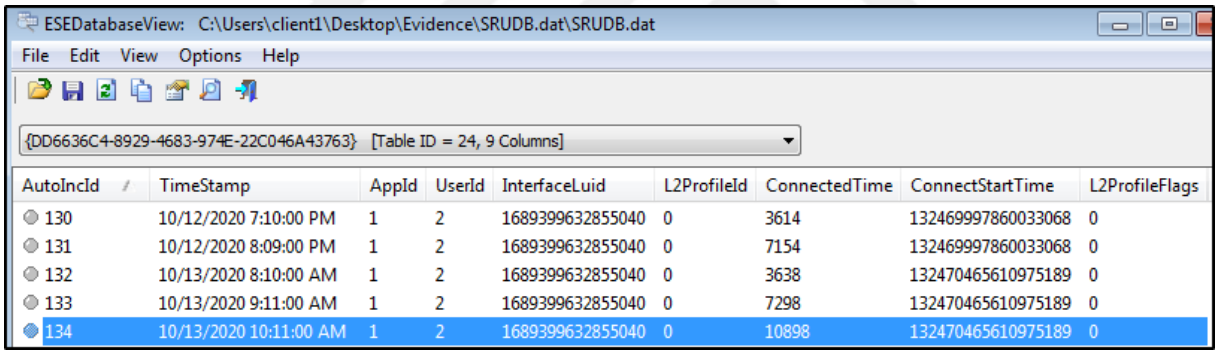


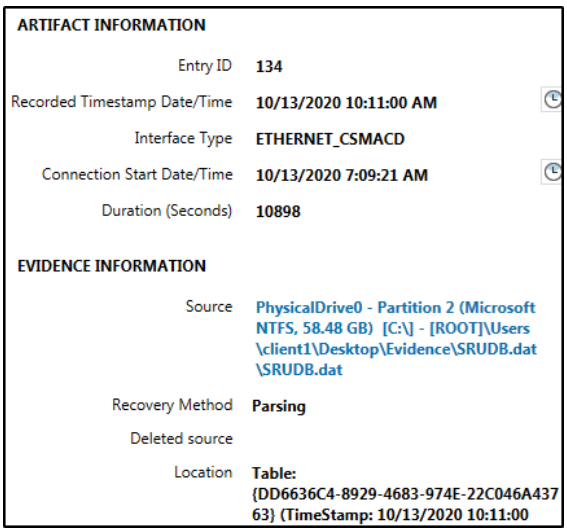
Figure 4.28: Network activity data inside Srudb.dat



AutoIncId	TimeStamp	AppId	UserId	InterfaceLuid	L2ProfileId	ConnectedTime	ConnectStartTime	L2ProfileFlags
130	10/12/2020 7:10:00 PM	1	2	1689399632855040	0	3614	132469997860033068	0
131	10/12/2020 8:09:00 PM	1	2	1689399632855040	0	7154	132469997860033068	0
132	10/13/2020 8:10:00 AM	1	2	1689399632855040	0	3638	132470465610975189	0
133	10/13/2020 9:11:00 AM	1	2	1689399632855040	0	7298	132470465610975189	0
134	10/13/2020 10:11:00 AM	1	2	1689399632855040	0	10898	132470465610975189	0

Using Magnet Axiom shows more precise details extracted from these Artifacts, we can see the application ID, Interface type ETHERNET_CSMACD recorded on 13/10/2020 at 10:11:00 by the operating system with a duration of 3 hours. This evidence is from my virtual machine Windows 10 where I used it to browse the internet for 3 hours and as you can see it is recorded in the SRUM database.

Figure 4.29: Artifact Information using Axiom



4.1.2.4. Relevance to digital forensics

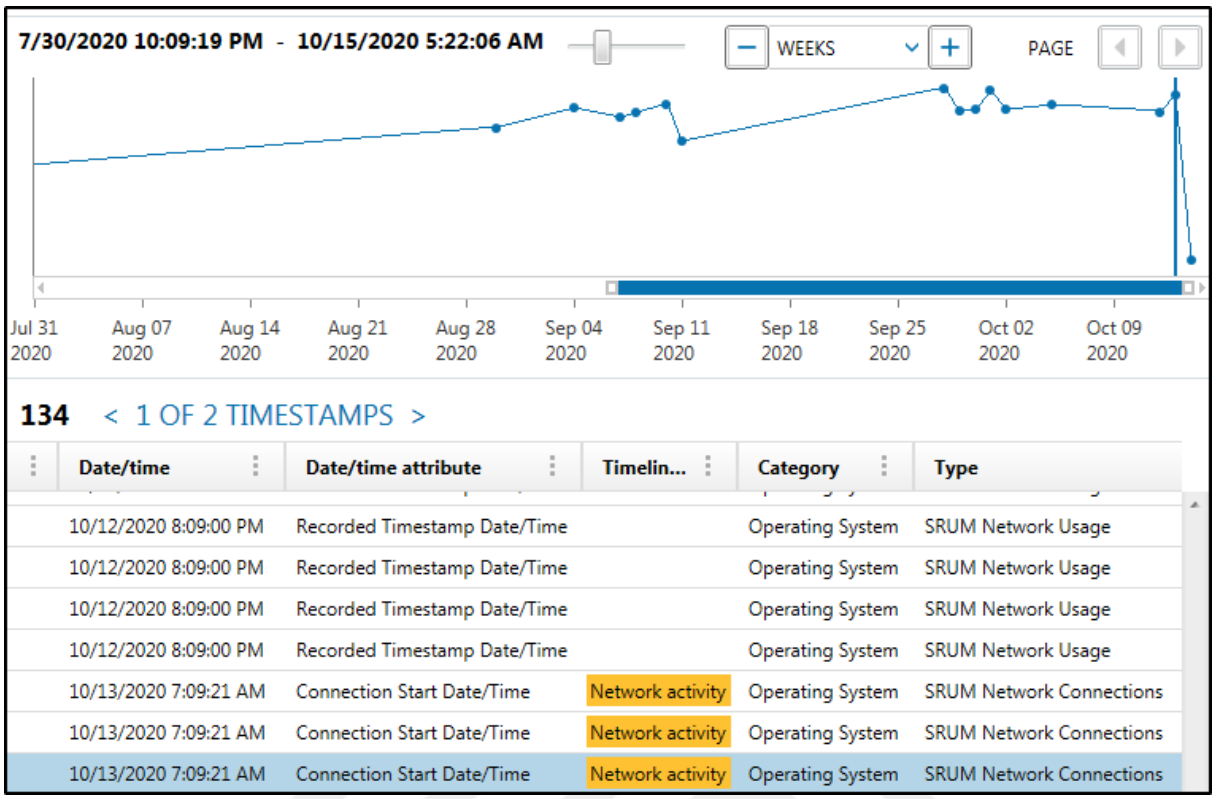
SRUM is one of the essential resources in a digital forensic investigation, it offers forensics investigator a historical sight into the previous operating system usage. It used to collect evidence of program executions such as phishing emails that deliver malware the digital forensics investigator wants to know if the user clicks it, did the malware run and if the email was sent to anyone else.

SRUM can be used as a last resort in case of a security event, in rare cases when there are no logs of any kind such as Syslog, event logs, centralized long-term logging, and no IDS.
Determine which user launches a specific process data upload and download keeps data of programs that were deleted installed uninstalled run from external media.

4.1.2.5. Timeline analysis

For timeline analysis I have used “Magnet Axiom” to parse the evidence collected by FTK Imager, Axiom will show SRUDB.dat files in a timeline format, for a digital forensics investigator it will be able to answer the questions related to the investigation, as you can see in Figure 4.30.

Figure 4.30: Srum Timeline Analysis using Axiom



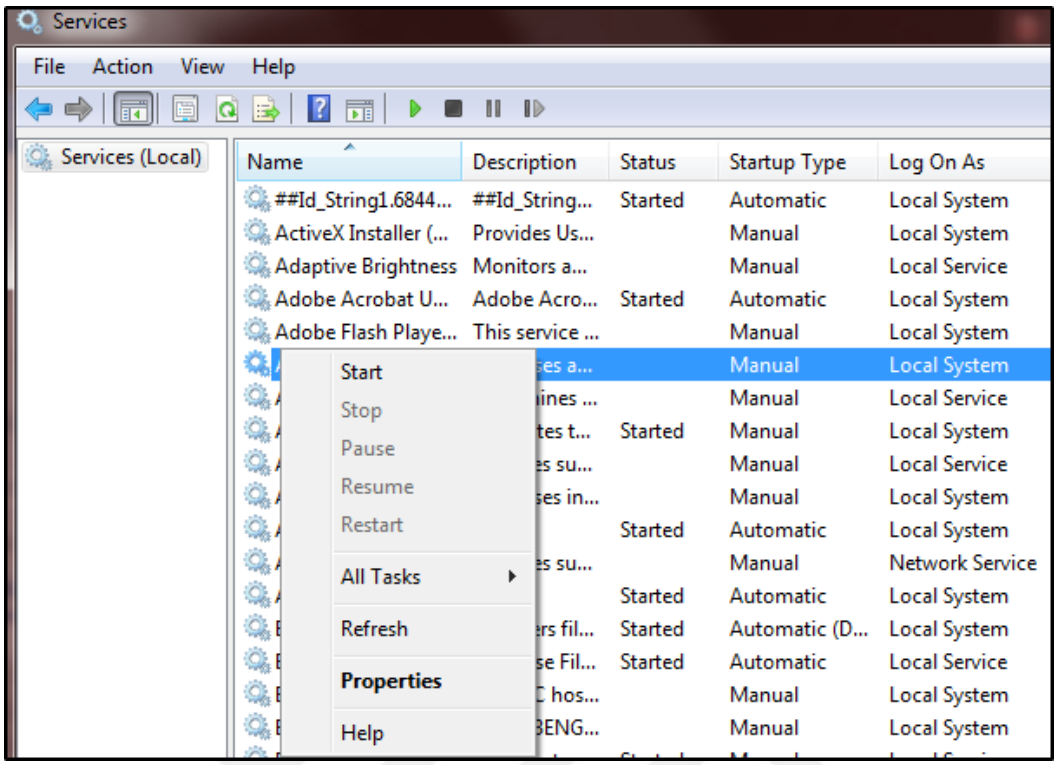
4.1.3. Win32 Service

Win32 Service also known as Windows Services is the backbone of the Microsoft Windows operating system which allows the creation of processes. On the contrary to usual software that is initiated by the end-user and exclusively runs when the end-user logged in, Windows services start without the end-user interaction and can continue to run even after logging off.

Furthermore, Windows Services running in the background will be triggered when the Windows operating system is booted.

Windows Services is a component of every version of the Windows operating system, and as an end-user you can access and manage to start, stop or disable them through multiple tools such as Microsoft Management Console (MMC) or Windows Services Manager, running one of them the end-user can change the state of a service as it is shown in the following Figure 4.31Figure 4.31: Windows services manager.

Figure 4.31: Windows services manager



These changes on Windows services are stored within Windows internals and has a great value in case of a security incident and will help the digital investigator in revealing such threats where threat actors use Windows services as a persistent technique that grants threat such as malware, worm, backdoor or remote access Trojan (RAT) to carry on functioning across rebooting, and it will be considered as an Advanced Persistence Threats.

In the following subsections I am going to demonstrate the location, creation, and structure of Windows Services, and end it with its relevance to digital forensics and timeline analysis of artifacts extracted from this feature of Windows.

4.1.3.1. Location of artifact

In my research, I have found that Windows Services artifacts are located in the following areas within the Windows registry and folder directories of %SystemRoot%\, and I have noticed that there is no difference regarding the location of artifact between the old version of Windows the legacy XP, Windows 7 (out of support in 2020) and the new Windows 10 as it is shown in the following Table 4.8 that I have created:

Table 4.8: Windows Service locations

Windows Version	Location of Artifact	Description
Windows XP, 7 and 10	C:\WINDOWS\system32	Contains two files: • Services.exe • Services.msc
	HKLM\SYSTEM\CurrentControlSet\Services	Registry location where all data from running and stopping Windows services are stored

The following Figure 4.33, and Figure 4.33, shows that there are no differences between all Windows version as you can see for registry and directories.

Figure 4.32: Windows Services in registry for different Windows versions

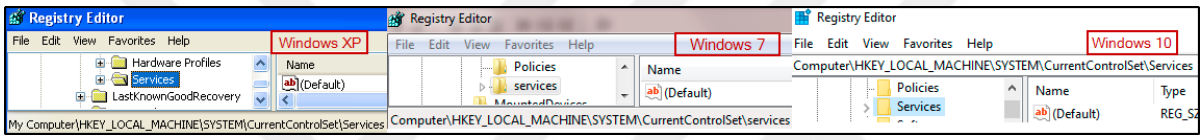
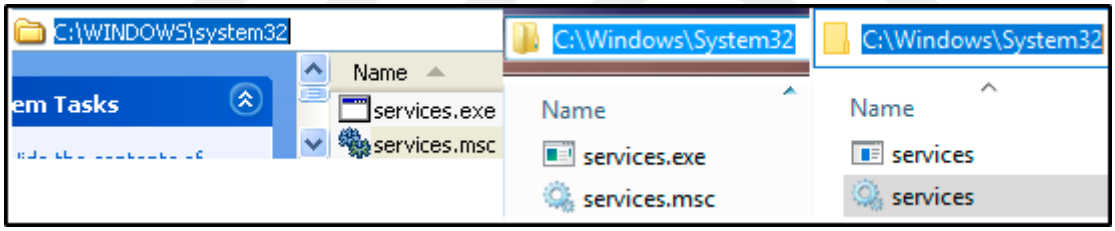


Figure 4.33: Windows Services directory for different Windows versions



4.1.3.2. Artifact creation

Windows Services come integrated with every version of Windows operating system, and will start automatically as soon as the OS is booted as explained by Harlan Carvey:
Windows services are programs that run automatically when the system is booted and are started by the system and with no interaction from the user (however, users with the appropriate privileges can install, start, and stop services) (Carvey 2016,p. 82).

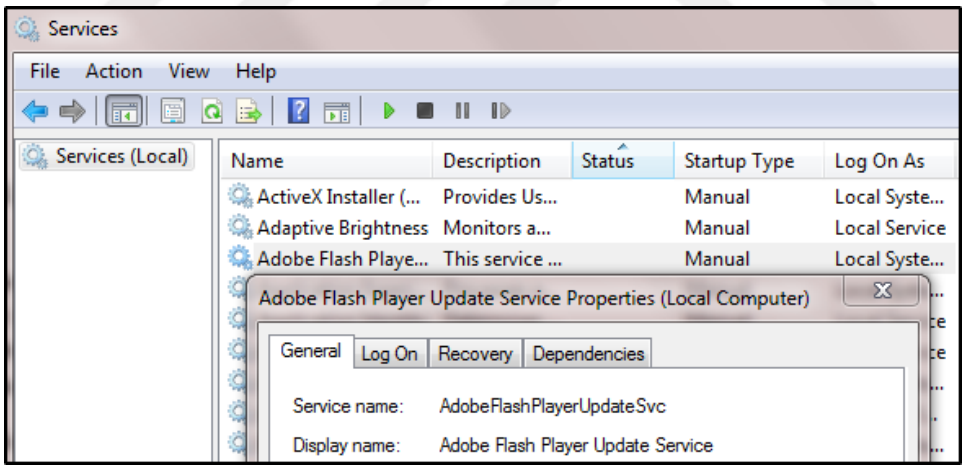
4.1.3.3. Structure of artifact

Windows Services are managed by Windows Services Control Manager, performing changes on this manager will change the status of services, this information is stored in Windows registry hive HKEY_LOCAL_MACHINE. In the following subsection, I am elaborating on both components of this feature.

4.1.3.3.1. *Windows Services Control Manager*

Windows Services are administrated by a tool integrated with each version of the Windows operating system called Windows Services Control Manager, the manager shows a list of services names, descriptions, status which can be either running, stopped or paused, and service type. Double-clicking on any service presents the properties with more details and options as you can see in Figure 4.34.

Figure 4.34: Windows service properties



Windows Services consist of four categories as it was clarified by Shanker Naik:
Local Services, Network Services, and System. Third-party applications such as antivirus software may also install their services (Naik et al 2019, p. 1).

Windows Service Manager runs using:

a. Service.msc is the tool that the end-user can use to launch Windows Service Control Manager

b. Service.exe is a pre-Windows installation executable and is a Windows application that is responsible for loading and initializing auto-start device drivers and Windows Services

The next subsection demonstrates the structure of the final location where the information of each service is stored inside the “SYSTEM” hive “Services” subkey of the Windows registry.

4.1.3.3.2. *Registry Services subkey*

Windows Services store all information in registry Hive “SYSTEM” more specifically in:

a. HKLM\SYSTEM\CurrentControlSet\Services

This Subkey is followed by several service names each one of them contain different registry attributes where all information needed for digital forensics analyst lies. There are no differences between versions of Windows operating systems regarding the registry Services subkey.

Jerry Honeycutt described the Services content:

The Services subkeys contain entries for standard and optional Windows services, such as network drivers and services. Although the values of the entries differ for each service, most Services subkeys have the same subkeys and entries (Honeycutt 2005, p. 468).

Every Service subkey carries the name of the service that uses it. Often, this is also the name of the file from which the service is loaded.

As said above most of the Services subkeys have the same subkey and entries or attributes and the following Figure 4.35 and Figure 4.36 show a sample of the general similarities between two services.

Figure 4.35: FileInfo service attributes

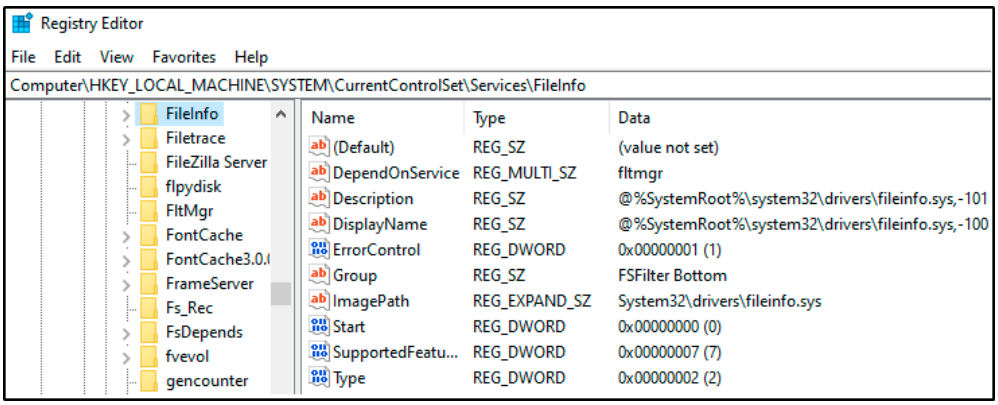
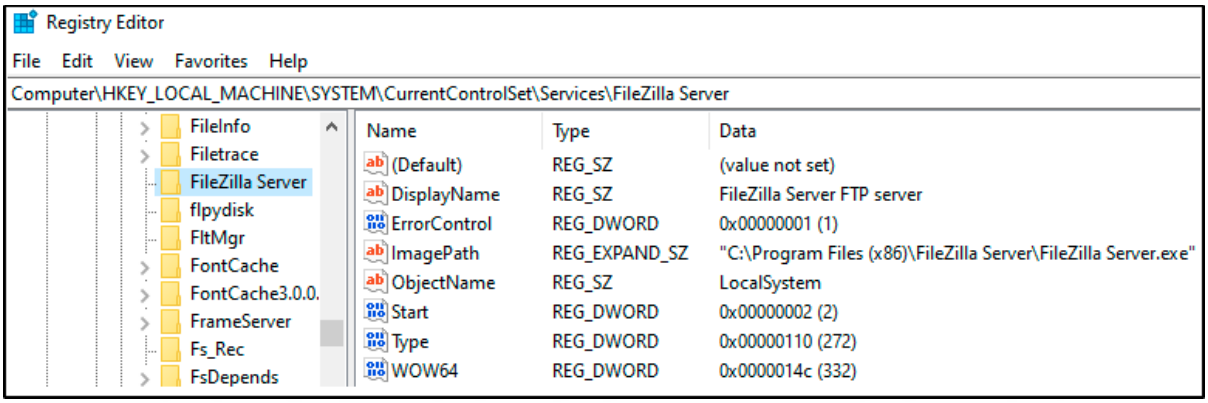


Figure 4.36: FileZilla service attributes



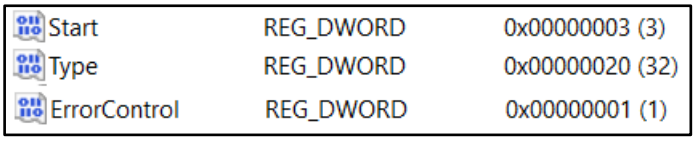
I have collected most of the common entries also called attributes in the following table, such attributes are very important to digital forensics.

Table 4.9: Windows services registry attributes

Attribute	Description	Type
Service Name	Registry Subkey that bear the same name of the Service	key
DisplayName	Display Name in Windows Services Manager	UTF-16
Description	Displays brief description of hosted service	UTF-16
Type	This defines the kind of service or driver	Binary
Group	A collection of similar services that are loaded together at startup	UTF-16
Status	Displays the status of the service	UTF-16
DependOnService	The dependency of the specific service.	UTF-16
ErrorControl	Actions to control specific error.	Binary
ImagePath	Path to executable	UTF-16
ObjectName	Displays User Account; LocalSystem/Local Service/Network Service	UTF-16
RequiredPrivileges	Required privileges to run this specific service.	UTF-16
Start	This defines when in the boot sequence the service should be started	Binary
FailureActions	represents the configured actions on failure of the service	Binary
DelayedAutostart	A value that indicates whether the service should be delayed from starting until other automatically started services are running.	Binary
Tag	The value of Tag specifies a number that is unique within the group of which the service is a member.	Binary
Last Modification Date	This attribute stores the date and time of the last changes done on the service; it is hidden can be revealed for every service by exporting the targeted service as a text file	Time

Each binary attribute such as Start, Type, and ErrorControl stores information about the service in the form of code, as it is shown in the below Figure 4.37.

Figure 4.37: Binary code in registry



Different codes have different meanings and the following tables demonstrate the code numbers for a binary attribute of a service and its description.

a. Start Attribute: it is used to specify when the service should be started during the boot sequence, codes and its related descriptions as it is seen in Table 4.10

Table 4.10: Start attribute codes and description

Value	Start Type	Description
0x00	Boot	The kernel loaded will load this driver first as its needed to use the boot volume device
0x01	System	This is loaded by the I/O subsystem
0x02	Autoload	The service is always loaded and run
0x03	Manual	This service does not start automatically and must be manually started by the user

b. Type Attribute: indicate service or device type as it is listed in Table 4.11

Table 4.11: Type attribute codes and description

Value	Description
0x01	Kernel-mode device driver
0x02	Kernel-mode device driver that implements the file system
0x04	Information used by the Network Adapter
0x10	A Win32 service that should be run as a stand-alone process
0x20	A Win32 service that can share address space with other services of the same type

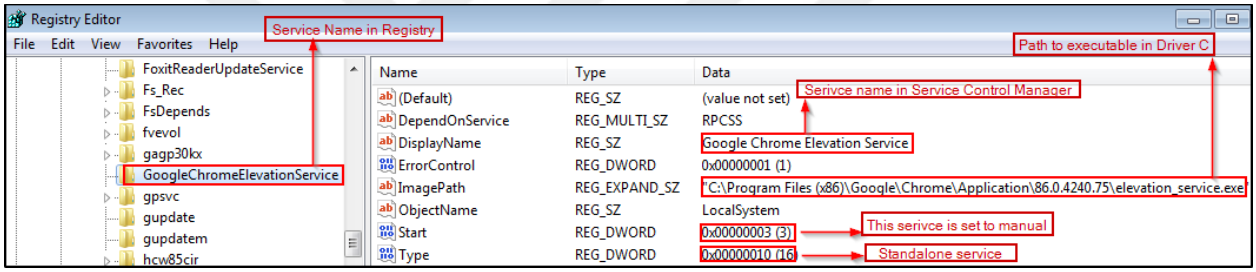
c. ErrorControl Attribute: it shows messages in case of failure to startup the service during boot as it is explained in Table 4.12

Table 4.12: ErrorControl codes and description

Value	Description
0x00	If this driver can't be loaded or started ignoring the problem and display no error
0x01	If the driver fails produce a warning but let bootup continue
0x02	Panic, if the current config is last known good continue, if not switch to last known well
0x03	Record the current startup as a failure. If this is last known good run diagnostic, if not switch to last known good and reboot

In the following sample service example, I demonstrated the above-mentioned information on "GoogleChromeElevationService" which is the service name used in the registry, DisplayName shows the Name of the service control manager, and the path to executable show in ImagePath, the last two attributes shows service is set to manual and as a standalone service as it is demonstrated in the following Figure 4.38.

Figure 4.38: Stored information in a service



d. Last modification date Attribute: is very important to digital forensics it contains date and time, this information is hidden, however, it can be revealed by exporting the targeted service and saving it as a text file as you can see in the following Figure 4.39, Registry Explorer is another tool that can be used is to show the hidden date and time and I noticed that there is 2 hours difference in time between these two methods to show the date and time.

Figure 4.39: Revealing last modification date attribute for a service

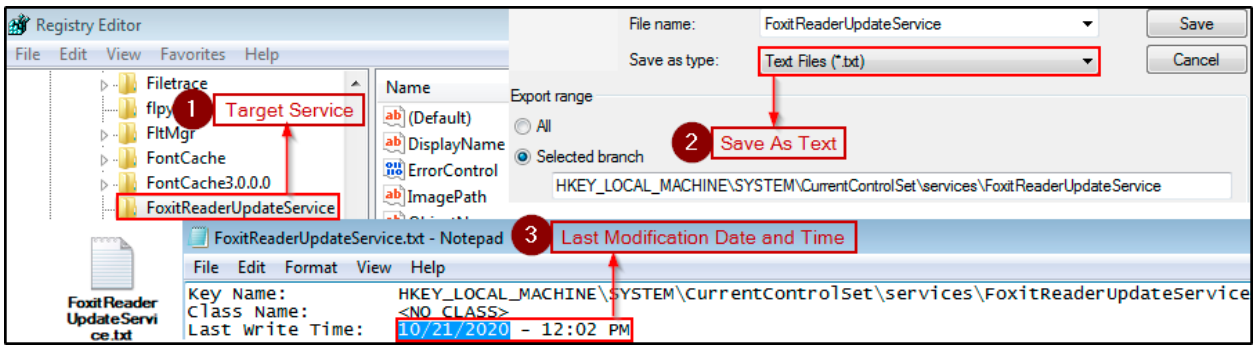
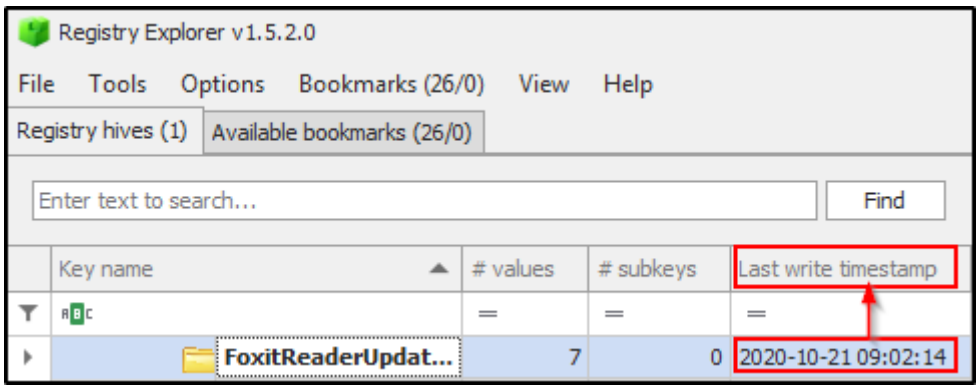


Figure 4.40: Using registry explorer tool to show hidden time for services



4.1.3.4. Relevance to digital forensics

The real threat of Windows Services comes from its ability to be used as a persistent mechanism by the attacker to keep their attacks immune to operating system reboots, independent from end-user interaction and hidden because it is using an integrated tool from Microsoft Windows and this is was clearly explained by Cameron H. Malin:
Microsoft Windows services are long-running executable applications that run in their own Windows sessions; they do not require user initiation or interaction. Services can be configured to automatically start when a computer is booted up, paused, and restarted without showing up in any user interface. Malware can manifest on a victim system as a service, silently running in the background, unbeknownst to the user (Malin et al. 2012, p. 24).

We can summarize the relevance into the next points:

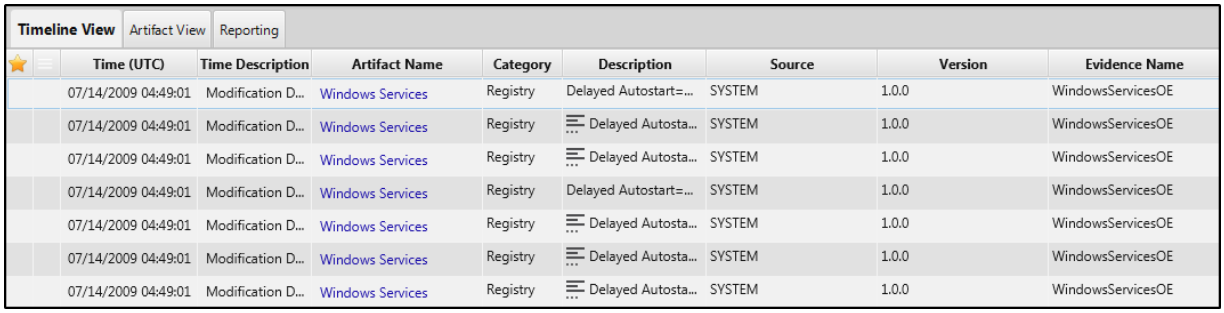
a. Digital forensic examiner can gather the following information from Windows Services: Service Name, Display Name, Status, Startup configuration, Process ID, Executable Path, and Username associated with the service
b. Detect persistent mechanism services used by advanced persistent threat (APT) such as malware, worm, backdoor or remote access Trojan (RAT)

4.1.3.5. Timeline Analysis

After mining Evidence using FTK imager for Windows Survives feature I used in my thesis two forensics tools Artifast and Axiom to examine artifacts extracted from it in a timeline format.

In the following example, I have used "Artifast" for investigation, the results I gathered from the test experiments that I performed on Windows 7 testing operating system, where it shows all the Windows services files in a Timeline format as it is shown in Figure 4.41.

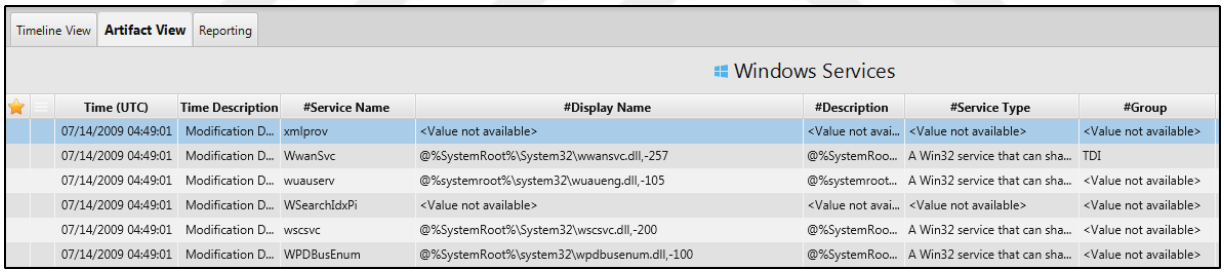
Figure 4.41: Artifast parsing Windows services in a Timeline format



Time (UTC)	Time Description	Artifact Name	Category	Description	Source	Version	Evidence Name
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autostart=...	SYSTEM	1.0.0	WindowsServicesOE
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autosta...	SYSTEM	1.0.0	WindowsServicesOE
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autosta...	SYSTEM	1.0.0	WindowsServicesOE
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autostart=...	SYSTEM	1.0.0	WindowsServicesOE
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autosta...	SYSTEM	1.0.0	WindowsServicesOE
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autosta...	SYSTEM	1.0.0	WindowsServicesOE
07/14/2009 04:49:01	Modification D...	Windows Services	Registry	Delayed Autosta...	SYSTEM	1.0.0	WindowsServicesOE

Clicking on the Artifact name icon navigate the examiner to a more detailed Window of the software where it shows Service Name in registry and in-Windows Service Control panel, Description and Type in addition to other details as you can see in the below Figure 4.42.

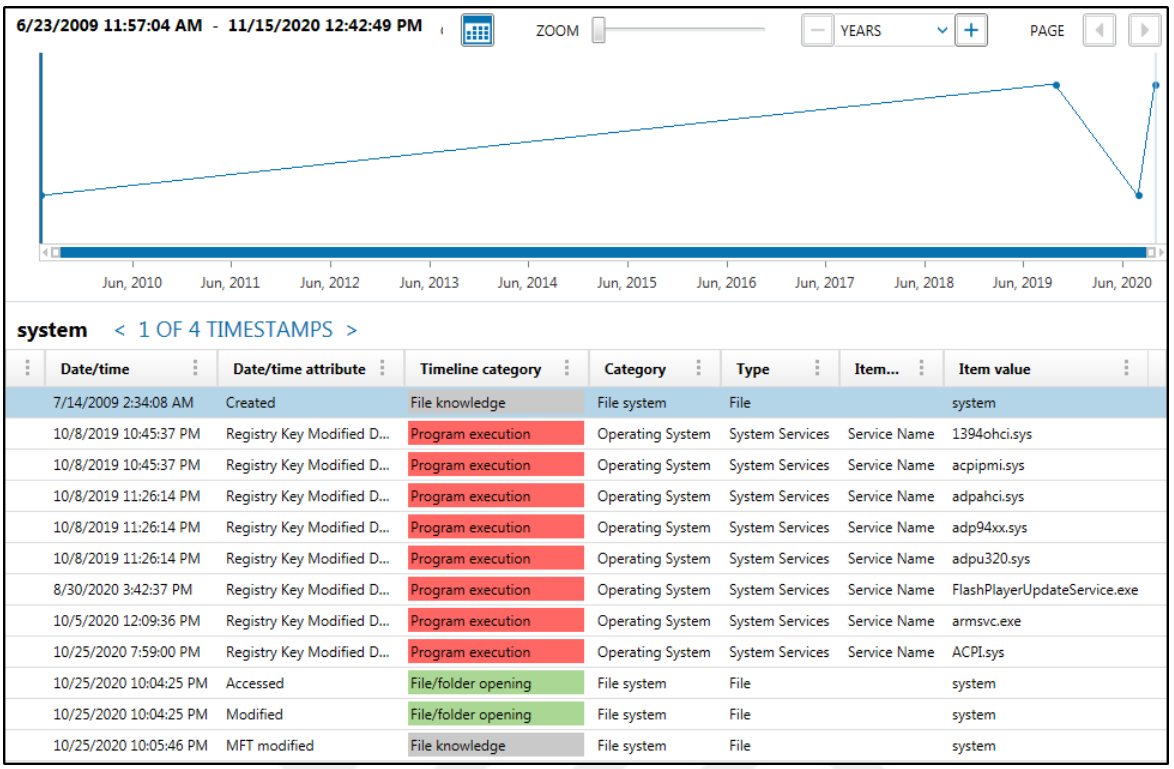
Figure 4.42: Artifast detailed view for Windows services artifact



Time (UTC)	Time Description	#Service Name	#Display Name	#Description	#Service Type	#Group
07/14/2009 04:49:01	Modification D...	xmlprov	<Value not available>	<Value not avai...	<Value not available>	<Value not available>
07/14/2009 04:49:01	Modification D...	WwanSvc	@%SystemRoot%\System32\wwansvc.dll,-257	@%SystemRoo...	A Win32 service that can sha...	TDI
07/14/2009 04:49:01	Modification D...	wuauserv	@%systemroot%\system32\wuaueng.dll,-105	@%systemroot...	A Win32 service that can sha...	<Value not available>
07/14/2009 04:49:01	Modification D...	WSearchIdxPi	<Value not available>	<Value not avai...	<Value not available>	<Value not available>
07/14/2009 04:49:01	Modification D...	wscsvc	@%SystemRoot%\System32\wscsvc.dll,-200	@%SystemRoo...	A Win32 service that can sha...	<Value not available>
07/14/2009 04:49:01	Modification D...	WPDBusEnum	@%SystemRoot%\system32\wpdbusenum.dll,-100	@%SystemRoo...	A Win32 service that can sha...	<Value not available>

The next test was conducted using Magnet Axiom, performing the same steps on the evidence collected by FTK Imager from Windows 7 testing virtual machine to parse Windows services artifacts will show results in Timeline format, and the following Figure 4.43, and Figure 4.44 show is the parsing results which will be used in digital forensics investigation.

Figure 4.43: Axiom Timeline Analysis parsing results for Windows Services



Date/time	Date/time attribute	Timeline category	Category	Type	Item...	Item value
7/14/2009 2:34:08 AM	Created	File knowledge	File system	File		system
10/8/2019 10:45:37 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	1394ohci.sys
10/8/2019 10:45:37 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	acpipmi.sys
10/8/2019 11:26:14 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	adpahci.sys
10/8/2019 11:26:14 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	adp94xx.sys
10/8/2019 11:26:14 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	adpu320.sys
8/30/2020 3:42:37 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	FlashPlayerUpdateService.exe
10/5/2020 12:09:36 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	armsvc.exe
10/25/2020 7:59:00 PM	Registry Key Modified D...	Program execution	Operating System	System Services	Service Name	ACPI.sys
10/25/2020 10:04:25 PM	Accessed	File/folder opening	File system	File		system
10/25/2020 10:04:25 PM	Modified	File/folder opening	File system	File		system
10/25/2020 10:05:46 PM	MFT modified	File knowledge	File system	File		system

Figure 4.44: Axiom details results after parsing the targeted artifact

DETAILS

ARTIFACT INFORMATION

Service Name	**ACPI.sys**
Service Type	**KernelDriver**
Start Type	**Boot**
Service Location	**system32\drivers \ACPI.sys**
Group	**Boot Bus Extender**
Display Name	**Microsoft ACPI Driver**
Service Details	**ImagePath: system32 \drivers\ACPI.sys DriverPackageId: acpi.inf_amd64_neutral_ 2a841284c9de8962**
Hosted	**No**
Registry Key Modified Date/Time	**10/25/2020 7:59:00 PM**
Error Control	**Critical**
Tag	**1**

4.1.4. Update Sequence Number Journal

New Technology File System (NTFS) includes a large number of features, one of which is called filesystem journaling, this technology allows the operating system to maintain a transactional record of all changes made to a volume such that in event of a crash or power failure the system can roll back the changes or continue where it left off the goal is to maintain the filesystem integrity and hopefully prevent catastrophic events from occurring. Information gathered from NTFS can be used to tell if and when something was deleted from its volume, this goal can be achieved by extracting information from two of its journals: (1) Update Sequence Number Journal and (2) Transitions Log.

Update Sequence Number Journal in short known as $UsnJml, this unique feature exists on NTFS volumes on Windows operating system only, and its main purpose is to track changes on disk as it was described by Brian Carrier: The change journal is a file that records when changes are made to files and directories. It can exist in versions 3.0+ of NTFS and can be used by an application to determine which files have changed in a certain time span (Carrier 2005, p. 333).

When $UsnJrnl feature is activated or in the most recent case enabled by default the operating system store all alterations made to files and directories on the system volume inside $UsnJml feature, these changes include: (1) Creation, (2) Deletion, (3) Modifications, (4) Overwrite, (5) Transaction and last and not least (6) Compression. From a digital forensics point of view, these changes considered very valuables and treated as critical artifacts that need to be extracted from $UsnJml to be used in case of a security incident or analyzed for research purposes where it could be used by a developer to create a parsing tool for investigation purposes.

Update Sequence Number Journal is located inside system volume, it is hidden within the file system and cannot be viewed by the end-user directly. In the following subsections, I demonstrated the location, creation, and structure of the artifact, and at the end, I explained its relevance to digital forensics, and I showed the timeline analysis using two digital forensics tools: Magnet Axiom and ANJP.

4.1.4.1. Location of artifact

$UsnJml is located within NTFS system file precisely inside Entry number 11 inside $Extend metadata file, as it is seen in the following Table 4.13, using a special tool it is in:

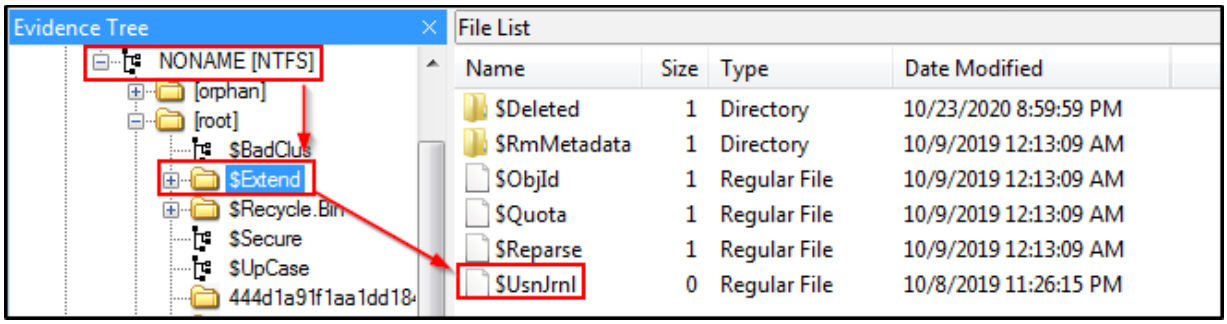
a. NTFS → $Extend → $UsnJrnl

Table 4.13: UsnJrnl location inside NTFS

Entry Number	File Name	Description
0	$MFT	MFT entry
1	$MFTMirr	Back file for the first 4 MFT entries
2	$LogFile	Store Metadata Transaction log
…	…	…
11	**$Extend**	**Store files like $Quota, $ObjId, $Reparse or $UsnJrnl, for optional extensions**

$UsnJml is hidden by default, and to demonstrate the exact location of this system file and its where about within Windows operating system I used FTK Imager tool since this tool is able to show hidden system files as it is shown in the following Figure 4.45. we must browse through NTFS then $Extend to find $UsnJrnl artifacts as it was clarified by Irfan Shakeel: $Extend: A file system directory containing various optional extensions, such as $Quota, $ObjId, $Reparse or $UsnJrnl (Shakeel 2015, p. 44).

Figure 4.45: $UsnJrnl location revealed using FTK Imager



It's worth mentioning that older copies of $Usnjrnl can be found in the volume shadow copy.

4.1.4.2. Artifact creation

Update Sequence Number Journal feature comes integrated within Windows operating system installation, however in Windows XP it is deactivated compared to Windows 7 and 10 where it is enabled by default, the following Figure 4.46 was taken from Windows XP where it shows system files with no $UsnJml. To activate this feature in Windows XP we can use "fsutil" built-in tool and run this command "fsutil usn createjournal m=3000 a=300 C:" in cmd, where "m" is for maximum size in bytes and "a" for memory allocation in bytes and the last attribute is for volume name. Figure 4.47 shows the same Windows XP system file after activating $UsnJrnl with the mentioned command.

Figure 4.46: Windows XP NTFS with $UsnJrnl feature not activated

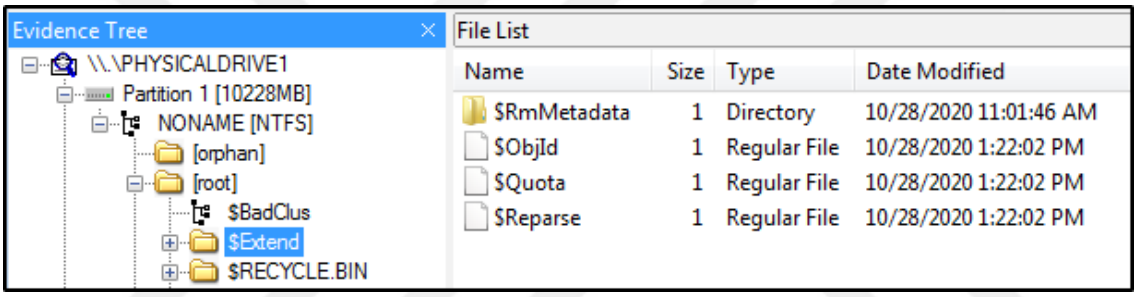
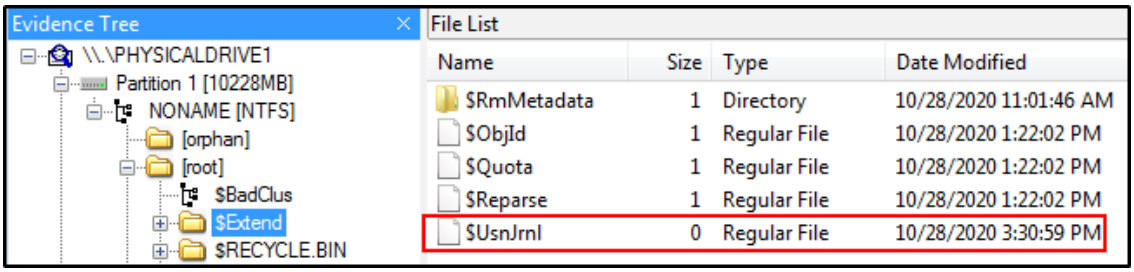


Figure 4.47: Windows XP NTFS with activated $UsnJrnl feature



When $UsnJrnl is enabled and activated it will be generated inside $Extend and it will contain two $DATA attributes:

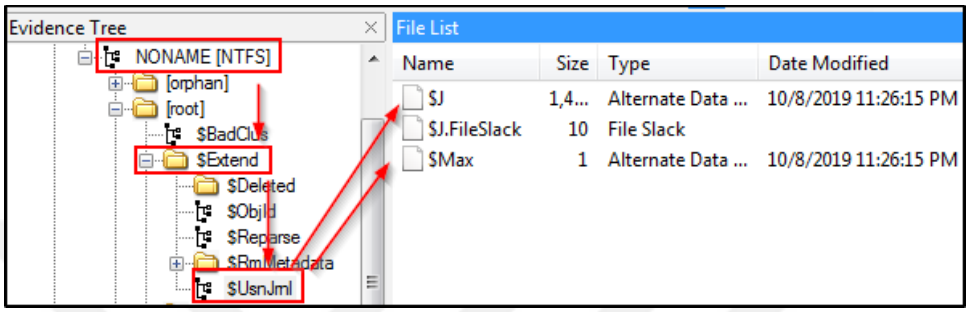
a. $J
b. $MAX

It is important to highlight that the other location that stores the same information, is in Master File Table (MFT), which keeps information related to update sequence number (USN)

in $STANDARD_INFORMATION at offset 0x40. The other valuable artifact is $LogFile sequence number (LSN) within MFT header which I explained in the next subsection.

The following Figure 4.48 was taken from the Windows 7 operating system where I also used FTK Imager as you can see it shows these files location within $UsnJrnl.

Figure 4.48: $MAX and $J are created when $UsnJrnl is enabled



4.1.4.3. Structure of artifact

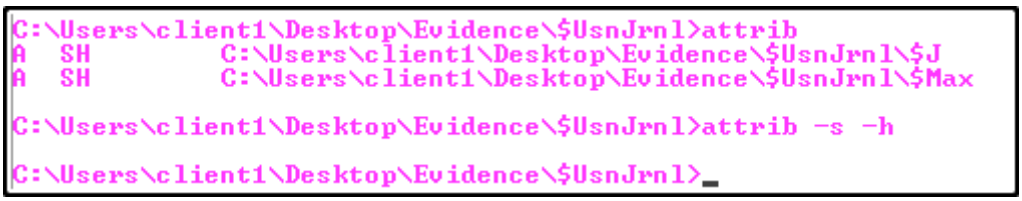
Update sequence number journal is an NTFS metafile, it contains two data stream files:

a. $MAX: is the metadata of the change log, it contains basic information on $UsnJrnl.
b. $J: is the actual change log records, $J involve Update Sequence Number (USN) and Master File Table (MFT) reference number

In fact, there are four attributes within $Usnjrnl: at offset 0x10 $STANDARD_INFORMATION followed by offset 0x20 $FILE_NAME which keeps the name of the file in this case it is $UsnJml and finally on offset 0x80 we have two $Data stream $J and $ MAX.

To be able to study these artifacts one more step need to be taken, since they are hidden by Windows operating system, thus they are not visible to the end-user, therefore after collecting them with FTK Imager, we need to reveal them by using the following command line in cmd "attirb -s -h" where "s" is to clear system file attribute and "h" to clear hidden file attribute as seen in the following Figure 4.49.

Figure 4.49: Revealing hidden data streams $J and $MAX



4.1.4.3.1. *$MAX data stream file*

$MAX is a 32 bytes data stream and all its attributes have an 8 bytes size. $MAX contains general information about $UsnJrnl and $J, in particular the maximum size of the log data which is the size of $UsnJrnl at offset 0x00 as it is shown in Table 4.14, followed by Allocation size at offset 0x08, then $Usnjrnl creation time at offset 0x10 and last we found the lowest valid USN at offset 0x18 which can be used to determine the beginning record in $J.

Table 4.14: Detailed $MAX data stream structure

#	Offset	Size	Data	Description	Type
1	0x00	8	Maximum Size	Log data Maximum size	UInt64
2	0x08	8	Allocation Size	Allocated zone size after storing new log data	UInt64
3	0x10	8	USN ID	$Usnjrnl file creation time	FILETIME
4	0x18	8	Lowest Valid USN	USN lowest value in present records, help determining the starting record offset inside $J	UInt64

4.1.4.3.2. *$J data stream file*

$J contain the actual records of the journal, each record stores information about:

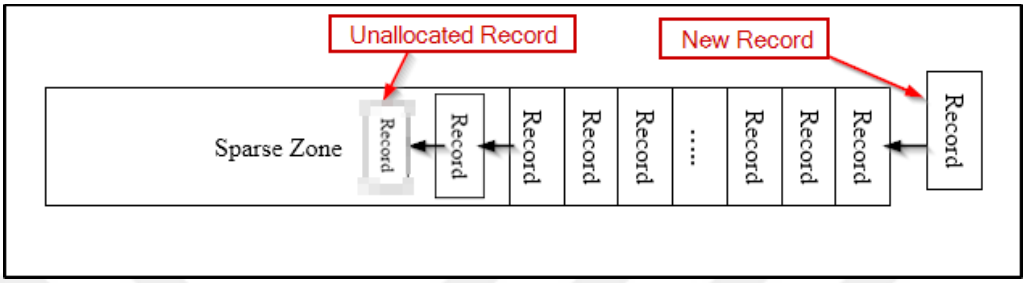
a. File name
b. Time of change
c. Type of change
d. Update sequence number (USN) with 64bit size

The length of the record is equal to the length of the file name, and Update sequence number is stored in the $STANDARD_INFORMATION of the modified file. USN represents the byte offset in the journal.

$UsnJml is a circular log, when the journal reaches its maximum size, it turns records into sparse data and continues adding records at the end of the file as it is shown in Figure

4.50Figure 4.50: Structure overview of $J data stream. When it assigns a new cluster at the end of the file, it turns the first cluster into sparse files, therefore the number of the clusters is not changing however the USN number is rising because it is matching the byte offset from the beginning of the file.

Figure 4.50: Structure overview of $J data stream



I created the following Table 4.15 to dissect the $J data stream in more details, in the below list I am mentioning only the important offset from a forensics point of view:

a. **Parent MFT Reference Number** at offset 0x10 which can help in acquiring the full path information of the object (File or Directory) when combined with $MFT
b. **Timestamp** when the change happened at offset 0x20
c. **Reason Flag** which tells what is it that case the change at offset 0x20
d. **File Attributes,** generally used for categorizing the object into a file or a directory at offset 0x34
e. **Filename** contains the name of the object (File or Directory), Its known that All files within NTFS are using Little Endian byte ordering however there are some exceptions and this attribute uses Big Endian ordering. Also, it is located at offset 0x3C with a variable size depending on the length of the name.

Table 4.15: Detailed $J data stream structure

#	Offset	Size	Data	Description	Type
1	0x00	4	Size of Record	Journal record Size	UInt32
2	0x04	2	Major Version	Major version = 2	UInt16
3	0x06	2	Minor Version	Minor version = 0	UInt16
4	0x08	8	MFT Reference Number	Object (File or Directory) MFT Reference Number	UInt64
5	0x10	8	**Parent MFT Reference Number**	Parent directory MFT Reference Number of an object	UInt64
6	0x18	8	USN	Update Sequence Number entry	UInt64
7	0x20	8	**Timestamp** (FILETIME)	Event Time (UTC +0)	FILETIME
8	0x28	4	**Reason Flag**	Change event flags	UInt32
9	0x2C	4	Source Information	Change event source	UInt32

10	0x30	4	Security ID	Security identifier SID for the record	UInt32
11	0x34	4	**File Attributes**	Attribute that define object features	UInt32
12	0x38	2	Filename size	Object name byte size	UInt16
13	0x3A	2	Filename offset	Object name offset	UInt16
14	0x3C	V	**Filename**	Object name effected by present event	UTF-16 Big Endian
15	V+0x3C	P	**Padding**	Goes up to 8 bytes	

Offset 0x28 include the change reason flag of journal record Table 4.15, this attribute is a group of flags, and it could be more than one cause for the change as it is seen in the below Table 4.16.

Table 4.16: Reason flags values used within $J data stream

#	Reason Flag Value	Description
1	0x0000001	File was overwritten.
2	0x0000002	Object (file or directory) was added
3	0x0000004	Object (file or directory) was truncated.
4	0x0000010	File named data streams was overwritten.
5	0x0000020	File named data streams was added
6	0x0000040	File named data streams was truncated
7	0x0000100	Object (file or directory) was created for the first time
8	0x0000200	Object was deleted
9	0x0000400	Change on Object (file or directory) extended attributes
10	0x0000800	Change on Object (file or directory) access permission
11	0x0001000	Change on Object name, current record is the old name
12	0x0002000	Change on Object name, current record is the new name
13	0x0004000	Change on Content indexed feature state of an object
14	0x0008000	Change on general object (file or directory) attributes
15	0x00010000	Object (file or directory) hard link was added or omitted
16	0x00020000	Object (file or directory) compression state was changed (Compressed/Decompressed)
17	0x00040000	Object (file or directory) Encryption state was changed (Encrypted/Decrypted)
18	0x00080000	Object (file or directory) ID was changed
19	0x00100000	Object (file or directory) reparse point value was changed
20	0x00200000	Named data stream was added, removed or changed
21	0x80000000	Object (file or directory) was closed.

Information source values start at offset 0x2C Table 4.15, generally, it is set to 0, however, it can have other values in case the source of record is not the end-user such as the OS as it is seen in the following Table 4.17.

Table 4.17: Source information values used within $J data stream

#	Source Information value	Description
1	0x00000000	Standard Event
2	0x00000001	Information about Change supplied by Operating system
3	0x00000002	Operation adds a private data stream to a file or directory.
4	0x00000004	Operation creates or updates the contents of a replicated file.

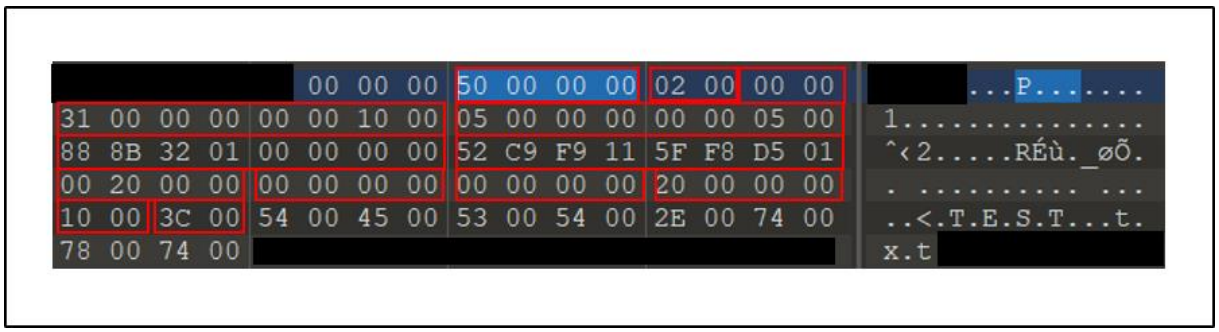
File attribute values begin at offset 0x34 Table 4.15, this field is a set of flags that determine the object (file or directory) features such as making the file a read-only file, other attributes are listed in the below Table 4.18.

Table 4.18: File attribute values used within $J data stream

#	File Attribute value	Description
1	0x01	A file that is read-only
2	0x02	The file or directory is hidden
3	0x04	A file or directory that the operating system uses a part of or uses exclusively.
4	0x10	The handle that identifies a directory
5	0x20	An archive Object (file or directory)
6	0x40	This value is reserved for system use
7	0x80	A file that does not have other attributes set
8	0x100	A file that is being used for temporary storage
9	0x200	A file that is a sparse file
10	0x400	A file or directory that has an associated reparse point, or a file that is a symbolic link.
11	0x800	A file or directory that is compressed
12	0x1000	This attribute indicates that the file data is physically moved to offline storage
13	0x2000	The file or directory is not to be indexed by the content indexing service
14	0x4000	A file or directory that is encrypted
15	0x8000	The directory or user data stream is configured with integrity (only supported on ReFS volumes)
16	0x10000	0 This value is reserved for system use
17	0x20000	The user data stream not to be read by the background data integrity scanner (AKA scrubber)

The following record example elaborates on how to analyze $Usnjml artifact by applying the above-mentioned $J data stream structure, as you can notice in the following Figure 4.51.

Figure 4.51: Record example inside $J data stream file



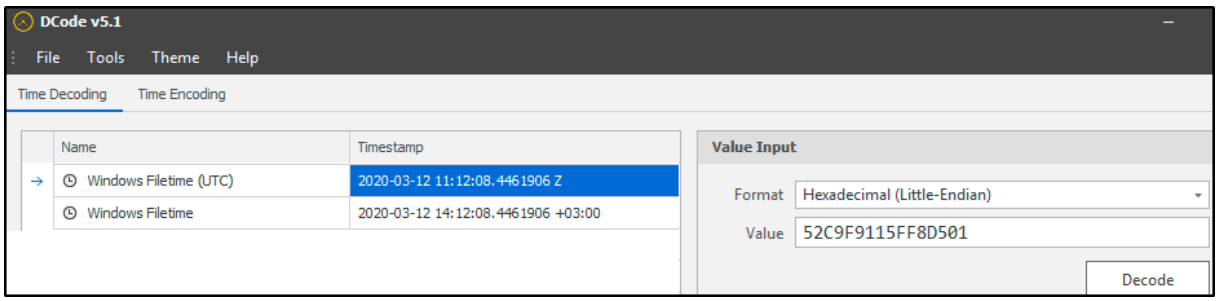
I have created this Table 4.19 of the results which shows every offset from the record and its meaning, therefore, I can understand that the record size is 50 00 00 00 in Little Endian ordering and its value in decimal is 80.

Table 4.19: Record analysis

#	Offset	Size	Data	Value
1	0x00	4	Size of Record	50 00 00 00
2	0x04	2	Major Version	02 00
3	0x06	2	Minor Version	00 00
4	0x08	8	MFT Reference Number	31 00 00 00 00 00 10 00 (Only 31 and 10 would be used for reference 31 in decimal is 49 and 10 in decimal is 16 so the reference would be 49-16 in ANJP Parser)
5	0x10	8	Parent MFT Reference Number	05 00 00 00 00 00 05 00 (Only 05 and 05 would be used for reference, 05 in decimal is 05, so the reference would be 5-5 in ANJP Parser)
6	0x18	8	USN	88 8b 32 01 00 00 00 00
7	0x20	8	Timestamp (FILETIME)	52 C9 F9 11 5F F8 D5 01
8	0x28	4	Reason Flag	00 20 00 00
9	0x2C	4	Source Information	00 00 00 00
10	0x30	4	Security ID	00 00 00 00
11	0x34	4	File Attributes	20 00 00 00
12	0x38	2	Size of Filename	10 00
13	0x3A	2	Offset to Filename	3C 00
14	0x3C	V	Filename	TEST.txt

For timestamp, it is also in little endian ordering and it is using UTC time converting it to human reading time using DCode tool we can tell when this file was created on 2020-03-12 at 11:12:08 as it is shown in Figure 4.52.

Figure 4.52: Record timestamp to human readable time using DCode tool



Reason flag is 20 which means File named data streams was added, source information is a standard event and attribute is set to archive. Finally, you can see at the end of the record is the file name and its size are set V in this case it is equal to 16 bytes little endian UTF-16.

4.1.4.4. Relevance to digital forensics

Analyzing $UsnJrnl can supply information about creation time, rename, move, or change on a file, with renaming both old and new name are recorded with their timestamp, if we take into account when a threat actor is able to set a foothold inside a system using malicious

software, rename it, move it, then delete it, therefore this artifact is a valuable root of timeline information for solving issues such as:

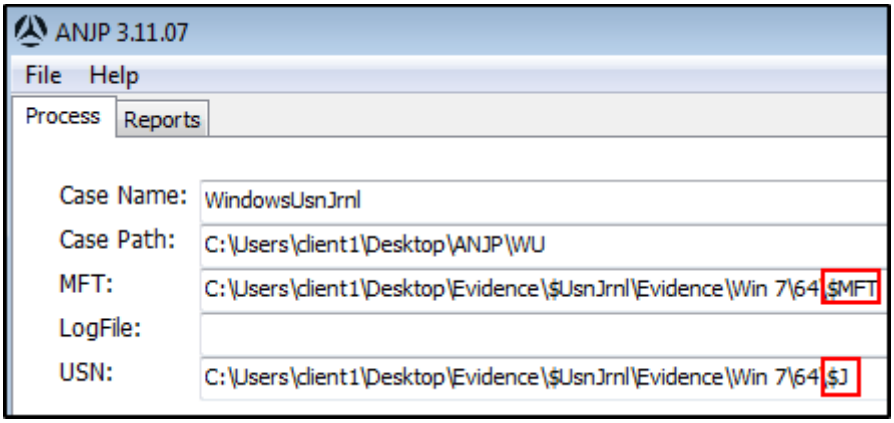
a. Malware attack
b. Incident response investigation
c. Timestamping anti-forensics activities
d. Used with combination of $LogFile and $MFT files to display more valuable information

4.1.4.5. Timeline analysis

After harvesting artifacts using FTK Imager from its location and revealing them as I explained in the above artifact structure subsection, then analyzing it, after that the results will be handed over to the developers where it will be used to create an application (Parser) to display the results related to $UsnJrnl artifact based on when they occurred in chronological order, in a timeline format. Or the output will be analyzed by a digital forensics investigator to determine Who did What, When, Where, and Why.

In the following example, I used "ANJP" to examine and analyze $UsnJrnl, the software is developed by Triforce for a digital forensics investigation, the figure below shows the results I gathered from the test experiments that I performed on the Windows 7 operating system, where I provided $J data stream files and $MFT into the software as it is shown in Figure 4.53.

Figure 4.53: Supplying ANJP with $J data stream and $MFT file



The below Figure show the output of 28998 created records, 28991 deleted records, and 43 Rename_moves records.

Figure 4.54: $UsnJrnl output by ANJP

```
Sat Nov 21 14:26:17 2020 - [Starting] Looking for USN Events
Creations                              28998
Deletions                              28991
Renames_Moves                          43
Sat Nov 21 14:26:30 2020 - [Finished] Looking for USN Events
```

The output consists of three tables: (1) creation, (2) deletion, and (3) rename/move, Figure 4.55 shows the creation table which includes record information on USN, file name, date and time, and reason. You can see that the file name "New Text Document.txt" is newly created in the reason column and this creation happened on 12-3-2020 at 11:12:05.

Figure 4.55: Creation record table output from ANJP

Export | Filter | <-- | -->

	USN Rcd USN	USN Evt ID	USN Rcd File Name	USN Evt Hit	USN Rcd Time	USN Rcd Reason	iN Ext illnan	USN Rcd File Ref #	N I te	USN Rcd Off	USN Rcd Trans Count	USN Rcd Rcd Len
3997	20087400	Creations	spin.mp4	28995	2020-03-12 11:11:54.668	File_Create	\spin.r	46-1355	5-	20087400	58489	80
3998	20087800	Creations	cryptography_tutorial.pdf	28996	2020-03-12 11:11:54.693	File_Create	\crypt	47-6	5-	20087800	58490	112
3999	20088696	Creations	$IZBUJUU.pdf	28997	2020-03-12 11:11:58.044	File_Create	\$REC	48-10	44	20088696	58491	88
4000	20089424	Creations	New Text Document.txt	28998	2020-03-12 11:12:05.713	File_Create	\New	49-16	5-	20089424	58494	104

The next table that I would like to present is the rename/remove table, it contains records that have been rename and I can notice that the file name "New Text Document.txt" has been rename to "Text.txt" and this happened on 12-3-2020 at 11:12:08.

Figure 4.56: Rename/Remove table output from ANJP

Export | Filter | <-- | -->

	USN Rcd Off	USN Rcd File Name	USN Rcd Reason	USN Rcd Time	USN Rcd File Ref #	USN Rcd P Ref #	USN Rcd USN	USN Rcd Trans Coun	USN Rcd Rcd Len	iN R : Rc	ISN Rc e Seq	USN Rcd P Rcd #	USN Rcd P Seq #
83	20088960	cryptography_tutorial.pdf	Rename_Old_Name	2020-03-12 11:11:58.045	47-6	5-5	20088960	58492	112	47	6	5	5
84	20089072	$RZBUJUU.pdf	Rename_New_Name	2020-03-12 11:11:58.045	47-6	44-14	20089072	58492	88	47	6	44	14
85	20089632	New Text Document.txt	Rename_Old_Name	2020-03-12 11:12:08.446	49-16	5-5	20089632	58495	104	49	16	5	5
86	20089736	TEST.txt	Rename_New_Name	2020-03-12 11:12:08.446	49-16	5-5	20089736	58495	80	49	16	5	5

4.1.5. Windows LogFile

I have already mentioned above New Technology File System (NTFS) includes a many number of features, one of which is known as filesystem journaling, this technology allows the operating system to maintain a transactional record of all changes made to a volume such that in event of a crash or power failure the system can roll back the changes or continue

where it left off the goal is to maintain the filesystem integrity and hopefully prohibit distressing events from occurring.

Information gathered from NTFS can be used to tell if and when a record was deleted from its volume, this objective can be carried out by extracting information from two of its journals: (1) Update Sequence Number Journal which I explained in the above subsection and (2) Transitions Log also known as $Logfile.

Unlike $Unsjml which can be found in NTFS and ReFS, the metadata journal or $LogFile is an NTFS specific. Transaction log "$LogFile" help NTFS to take in account all modifications to a file system, this feature provides file system recovery by logging and recoding the operations required for every transaction that changes critical file system data structures.

Windows $LogFile is located inside system volume, it is hidden within file system and cannot be viewed by the end-user directly. In the following subsections I demonstrated the location, creation and the structure of the artifact, and at the end I explained its relevance to digital forensics, and I showed the timeline analysis using two digital forensics tools: Magnet Axiom and ANJP.

4.1.5.1. Location of artifact

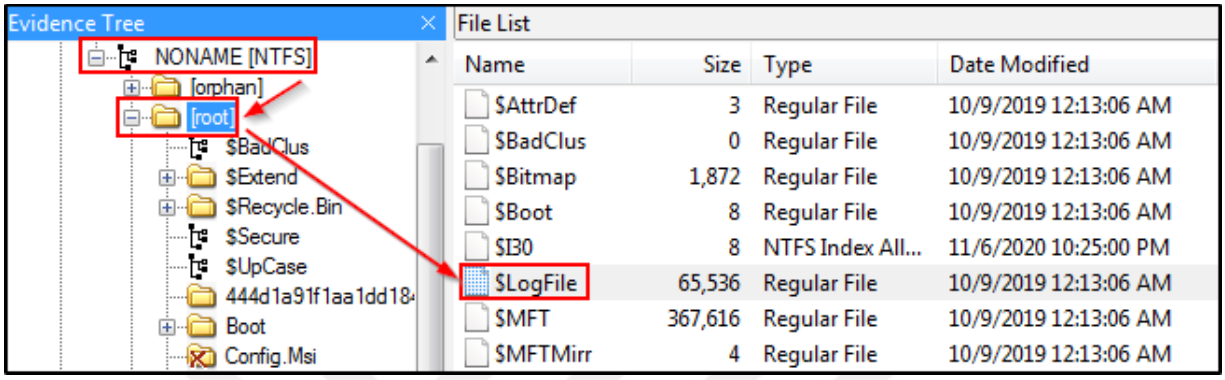
Windows LogFile is located at the core of NTFS, it is located at entry number 2 as you can see in the following Table 4.20, which represents the standard NTFS file system metadata files, using a special tool it is in: NTFS → $LogFile

Table 4.20: $LogFile location inside NTFS Structure

Entry Number	File Name	Description
0	$MFT	MFT entry
1	$MFTMirr	Back file for the first 4 MFT entries
2	**$LogFile**	**Store Metadata Transaction log**
3	$Volume	Volume Information
4	$AttDef	Attribute Information
5	.	File system Root directory
6	$Bitmap	Cluster Allocation state inside file system
7	$Boot	Boot sector and software
8	$BadClus	Bad sector clusters
9	$Secure	Security and Permission control
10	$Upcase	Unicode character uppercase
11	$Extend	Optional Extension

Like $UsnJrnl, $Logfile is hidden by default, and to present the exact position of this system file within Windows operating system, I utilized FTK Imager tool to be able to show it as it is shown in the following Figure 4.57, I navigated to NTFS then click on "root" folder to find $LogFile artifacts.

Figure 4.57: $LogFile Location revealed using FTK Imager



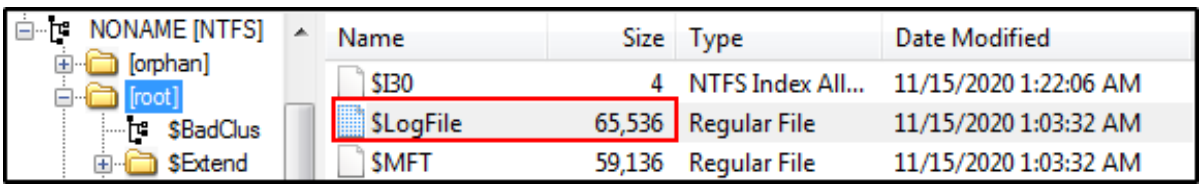
4.1.5.2. Artifact creation

Windows LogFile was designed by Microsoft as a special feature for NTFS to upgrade and enhance the reliability of an OS, it enables Windows operating system to rapidly fetch file system in a neat status. $Logfile is an NTFS file system metadata which comes integrated within Windows operating system installation, and this feature is included in Windows XP,7 and 10.

Once Windows operating system is freshly installed, a 64 MB Windows LogFile is created as it is displayed in Figure 4.58, and it consists of two main parts:

a. Restart area
b. Logging area

Figure 4.58: Windows LogFile is already created by freshly installed Windows 7



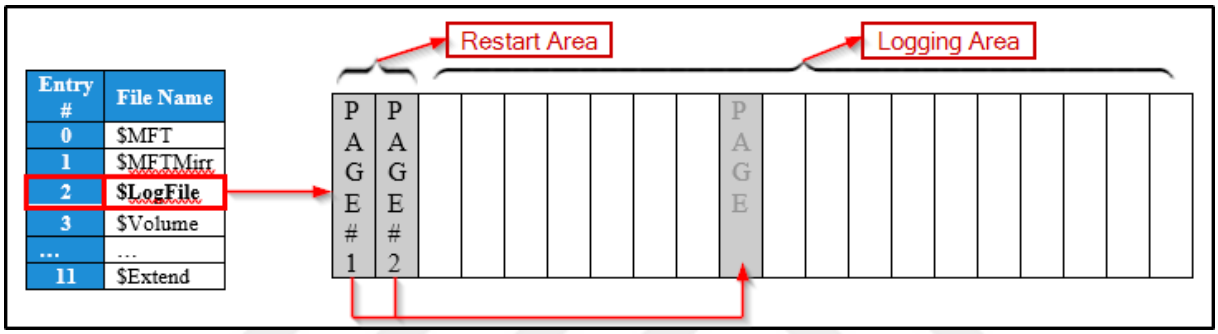
In the next subsection, I will dissect the structure of Windows Logfile with its core content.

4.1.5.3. Structure of artifact

Undetailed information was published on Windows LogFile by Microsoft concerning its precise content; however, Gyu-Sang Cho and Marcus K. Rogers elaborated that "The $LogFile consists of two parts: one is the restart area, the other is the logging area" (Cho 2012 et al, p. 213), two main portions that form its structure:

a. Restart area: contain information about last operation and record
b. Logging area: contain actual operation records

Figure 4.59: Overview Structure for $LogFile



Before jumping into the details of these artifacts that form the structure of Windows Logfile additional steps need to be performed. By default, $Logfile is hidden, so it is not visible to the end-user, therefore after collecting it with FTK Imager, I needed to reveal it by using the same command line used with $UsnJrnl in the above subsection. In cmd run "attirb -s -h" where "s" is to clear system file attribute and "h" to clear hidden file attribute as it is seen in the following Figure 4.60.

Figure 4.60: Revealing hidden $LogFile after extracting with FTK Imager

```
C:\Users\client1\Desktop\Evidence\$LogFile\Win7EN>attrib
  SH        C:\Users\client1\Desktop\Evidence\$LogFile\Win7EN\$LogFile

C:\Users\client1\Desktop\Evidence\$LogFile\Win7EN>attrib -s -h

C:\Users\client1\Desktop\Evidence\$LogFile\Win7EN>_
```

This step is also essential and required in Timeline analysis, where I have to use the $LogFile itself to show results in Timeline format.

4.1.5.3.1. *Restart Area*

$Logfile Restart Area has the following characteristics:

a. Hold last operation and present record data “Current LSN”

b. Divided to two pages, each one of them is 0x1000 in size

c. Restart area location within $LogFile structure, is between 0x0000 and 0x2000 offset

d. Both pages start with “RSTR”, the second page is backup, an exact copy of the first page

I have created the following Table 4.21, to show the structure of the restart area.

Table 4.21: Restart area structure

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	"RSTR" Header Signature				Update Seq. Offset		Update Seq. Count		Check Disk LSN							
0x10	System Page Size				Log Page Size				Restart Offset		Minor Version		Major Version			
0x20	Update Sequence Array															
0x30	Current LSN								Log Client		Client List		Flags			
0x40	UNKNOWN															
0x50																
0x60																
0x70	Oldest LSN								Restart LSN							

The next Table 4.22: Offset order of restart areaTable 4.22, shows a more detailed structure in relation to the offset order.

Table 4.22: Offset order of restart area

Offset	Attribute	Size/Range
0x00	RSTR Header Signature	4 bytes/0-3
0x04	Update Seq. Offset	2 bytes/4-5
0x06	Update Seq. Count	2 bytes/6-7
0x08	Check Disk LSN	8 bytes/8-f
0x10	System Page Size	4 bytes/16-19
0x14	Log Page Size	4 bytes/20-23
0x18	Restart Offset	2 bytes/24-25
0x1A	Minor Version	2 bytes/26-27
0x1C	Major Version	2 bytes/28-29
0x1E	Update Sequence Array	18 bytes/30-47
0x30	Current LSN	8 bytes/48-55
0x38	Log Client	2 bytes/56-57
0x3A	Client List	2 bytes/58-59
0x3C	Flags	4 bytes/60-63
0x70	Oldest LSN	8 bytes/112-119
0x78	Restart LSN	8 bytes/120-127

4.1.5.3.2. *Logging Area*

$Logfile Logging Area has the following characteristics:

a. Hold actual information related to operation records
b. Divided into “Buffer Page Area” and “Normal Page Area”

Buffer Page Area:

a. Buffer page area is divided in two pages, first page offset starts at 0x2000 or 0x3FF0, second page offset starts at 0x4000 or 0x5FF0
b. Second page act as a mirror for the first page
c. Sequential order is used to keep operation records
d. When this page is saturated, the content is moved to "Normal Page Area"
e. Buffer page area holds the last operation

Normal Page Area:

a. Normal page area offset starts exactly at the end of page two of the buffer page area
b. Sequential order is used to keep operation records
c. When this page is saturated, new records overwrite old records and this process takes place at the beginning of the normal page area

In the following Table 4.23, and Table 4.24, where I demonstrated the page structure, this structure is used for both buffer and normal page areas. I noticed that single header is tailed by versions operation records, and if the page can’t contain all the final records content, then the rest of the record contents will be stored on the following page.

Table 4.23: Page header format for logging area

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	**“RCRD” (signature)**				Update Seq. Offset		Update Seq. Count		**Last LSN or File Offset**							
0x10	Flags				Page Count		Page Position		**Next Record Offset**		Word Align		DWord Align			
0x20	**Last End LSN**															
0x30	Update Sequence Array															

Table 4.24: Detailed page header offset order

Offset	Attribute	Description	Size\Range
0x00	**RCRD**	**Signature**	4 bytes\0-3
0x04	Update Seq. Offset		2 bytes\4-5
0x05	Update Seq. Count		2 bytes\6-7
0x08	**Last LSN or File Offset**	The highest LSN between all records this includes the record that is content in the next page	8 bytes\8-F
0x10	Flags		4 bytes\16-19
0x14	Page Count	Number of pages that are used for the transaction run	2 bytes\20-21
0x16	Page Position	The current page number of a transaction run	2 bytes\22-23
0x18	**Next Record Offset**	Offset of last LSN on the page which has the highest LSN in the page	2 bytes\24-25
0x1A	Word Align		2 bytes\26-27
0x1C	DWord Align		4 bytes\28-31
0x20	**Last End LSN**	Last complete LSN on the page, the one that all its content in the same page (excluding the LSN of last record that has some content in the next page)	8 bytes\32-39
0x30	Update Sequence Array	Array containing the update sequences for replacement. The first two bytes of the value is the Update Sequence Value. These are used every 512 bytes	16 bytes\48-63

4.1.5.3.3. *Operation Records*

Operation records are considered to be a critical part of the logging area and are used in both buffer page and normal page areas; therefore, they share the same characteristics. Each operation record holds the real transaction content, and it hold one of the following records:

a. Checkpoint : beginning of transaction record
b. Update : middle of transaction record
c. Commit : end of transaction record

Each operation record contains information on the previous operation except the first one which is the checkpoint record. The structure of each operation record is dissected into two main parts:

a. Header : with a restricted size of 0x58 that contain the metadata of the record
b. Data : can be either Undo (data before operation) or Redo (data after operation)

The following Table 4.25, Table 4.26, and Table 4.27 that I created demonstrate the structure of the operation record format in more details.

Table 4.25: Operation record structure

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0x00	Current LSN								Previous LSN							
0x10	Client Undo LSN								Client Data Length				Client ID			
0x20	Record Type				Transaction ID				Flags		Alignment Reserved					
0x30	Redo OP		Undo OP		Redo Offset		Redo Length		Undo Offset		Undo Length		Target Attribute		LCNs to Follow	
0x40	Record Offset		Attr Offset		MFT Cluster Index		Alignment or Reserved		Target VCN				Alignment or Reserved			
0x50	Target LCN				Alignment or Reserved											

Table 4.26: Offset order for operation record

Offset	Attribute	Description	Size\Range
0x00	**Current LSN**	LSN of current record	8 bytes\0-7
0x08	**Previous LSN**	LSN of previous record	8 bytes\8-F
0x10	Client Undo LSN	Usually the same as Previous LSN, in case of recovery, LSN info of a record has Undo OP	8 bytes\16-23
0x18	Client Data Length	Record size which starts from “Redo OP" field to the end of record	4 bytes\24-27
0x1C	Client ID		4 bytes\28-31
0x20	Record Type	0x02 is a Check Point Record, 0x01 is the rest of Record	4 bytes\32-35
0x24	Transaction ID		4 bytes\36-39
0x28	Flags	0X00 Record does not overlap in next page, 0x01 Record does overlap in next page	2 bytes\40-41
0x2A	Alignment Reserved		6 bytes\42-47
0x30	Redo OP	Redo operation code	2 bytes\48-49
0x32	Undo OP	Undo operation code	2 bytes\50-51
0x34	Redo Offset	Offset of "Redo" data (From "Redo OP" Field)	2 bytes\52-53
0x36	Redo Length	Size of "Redo" data	2 bytes\54-55
0x38	Undo Offset	Offset of "Undo" data (From "Undo OP" Field)	2 bytes\56-57
0x3A	Undo Length	Size of "Undo" data	2 bytes\58-59
0x3C	Target Attribute		2 bytes\60-61
0x3E	LCNs to Follow	0x00 There is no next record (no LCNs follow LSN Header), 0x01There is a next record (LCNs follow LSN Header)	2 bytes\62-63
0x40	Record Offset	The MFT record offset if changed, this will affect the MFT record by applying Redo/Undo data, otherwise it is set to 0x00.	2 bytes\64-65
0x42	Attr Offset	If changes apply to MFT record, this will be applied Redo/Undo data within attribute, otherwise Redo/Undo data applies within the cluster	2 bytes\66-67
0x44	MFT Cluster Index	when changes occur to MFT record, Location of record applied Redo/Undo data within cluster are: First 0x0000, Second 0x0002, Third 0x0003, Forth 0x0006	2 bytes\68-69
0x46	Alignment or Reserved		2 bytes\70-71
0x48	Target VCN	Virtual Cluster Number of "$MFT" file applied Redo/Undo data	4 bytes\72-75
0x4C	Alignment or Reserved		4 bytes\76-79
0x50	Target LCN	Logical Cluster Number LCN of the disk applied Redo/Undo data	4 bytes\80-83
0x54	Alignment or Reserved		4 bytes\84-87

Table 4.27: Explaining Undo/Redo operation

#	NTFS Operation	Hex Value
1	Noop	0x00
2	CompensationLogRecord	0x01
3	InitializeFileRecordSegment	0x02
4	DeallocateFileRecordSegment	0x03
5	WriteEndOfFileRecordSegment	0x04
6	CreateAttribute	0x05
7	DeleteAttribute	0x06
8	UpdateResidentValue	0x07
9	UpdateNonresidentValue	0x08
10	UpdateMappingPairs	0x09
11	DeleteDirtyClusters	0x0A
12	SetNewAttributeSizes	0x0B
13	AddIndexEntryRoot	0x0C
14	DeleteIndexEntryRoot	0x0D
15	AddIndexEntryAllocation	0x0E
16	DeleteIndexEntryAllocation	0x0F
17	WriteEndOfIndexBuffer	0x10
18	SetIndexEntryVcnRoot	0x11
19	SetIndexEntryVcnAllocation	0x12
20	UpdateFileNameRoot	0x13
21	UpdateFileNameAllocation	0x14
22	SetBitsInNonresidentBitMap	0x15
23	ClearBitsInNonresidentBitMap	0x16
24	HotFix	0x17
25	EndTopLevelAction	0x18
26	PrepareTransaction	0x19
27	CommitTransaction	0x1A
28	ForgetTransaction	0x1B

29	OpenNonresidentAttribute	0x1C
30	OpenAttributeTableDump	0x1D
31	AttributeNamesDump	0x1E
32	DirtyPageTableDump	0x1F
33	TransactionTableDump	0x20
34	UpdateRecordDataRoot	0x21
35	UpdateRecordDataAllocation	0x22
36	UpdateRelativeDataIndex	0x23
37	UpdateRelativeDataAllocation	0x24
38	ZeroEndOfFileRecord	0x25

4.1.5.4. Relevance to digital forensics

Windows LogFile is indeed a great source of information when it comes to digital forensics investigation. Vast value of data stored into $LogFile since its records operate on files/folders which will help the investigator to perform a forensic rebuild of timestamp events based on:

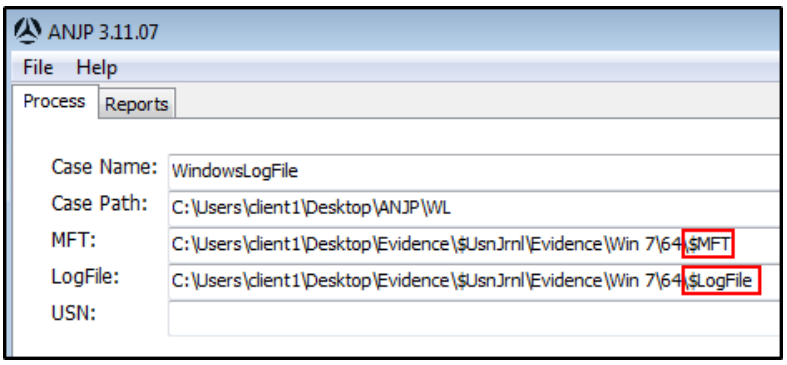
a. Files/Folders name
b. Timestamp
c. Recover deleted files/folders
d. Used with a combination of $UsnJrnl and $MFT files to display more valuable information

4.1.5.5. Timeline analysis

Timeline analysis is the last step in every digital forensic investigation, where the investigator demonstrates the chronology of changes on the artifact. To achieve this, I used two software “ANJP” and “Magnet Axiom”. I applied the same method used with $UsnJrnl on $LogFile where I collected the artifact using FTK Imager and disclose it using “attrib” command as I mentioned and elaborated in the above subsections.

Starting with "ANJP" to examine and analyze $LogFile, the figures below shows the results I gathered from the test experiments that I performed on windows 7 operating system. First step taken is that I provided $LogFile and $MFT to the software process page as it is shown in Figure 4.61.

Figure 4.61: Providing ANJP with $LogFile and $MFT evidence



The following Figure 4.62 shows an overview on the results with 128 created event/records, 121 deleted record and 2 Renamed records.

Figure 4.62: Initial parsing Results

```
Thu Nov 19 13:51:33 2020 - [Starting] Looking for LogFile Events
Creations                              126
Deletions                              123
Renames_Moves                          2
Thu Nov 19 13:51:34 2020 - [Finishing] Looking for LogFile Events
```

On the report page of ANJP to show more details on the records findings, Figure 4.63, below present the transaction records of created file or directory with its MAC time, here we have the new created text file named “New Text Document” and was created on 12-03-2020 at 11:12.

Figure 4.63: Create transaction results for $LogFile

LogFile Rcd Name	IDX Attr Modified Time	IDX Attr Accessed Time	IDX Attr Created Time
\New Text Document.txt	2020-03-12 11:12:05.713	2020-03-12 11:12:05.713	2020-03-12 11:12:05.713
\$RECYCLE.BIN\S-1-5-21-285092204-3887931873-1764890242-1000\$IZBUJUU.pdf	2020-03-12 11:11:58.043	2020-03-12 11:11:58.043	2020-03-12 11:11:58.043
\cryptography_tutorial.pdf	2020-03-12 11:11:54.692	2020-03-12 11:11:54.692	2020-03-12 11:11:54.692

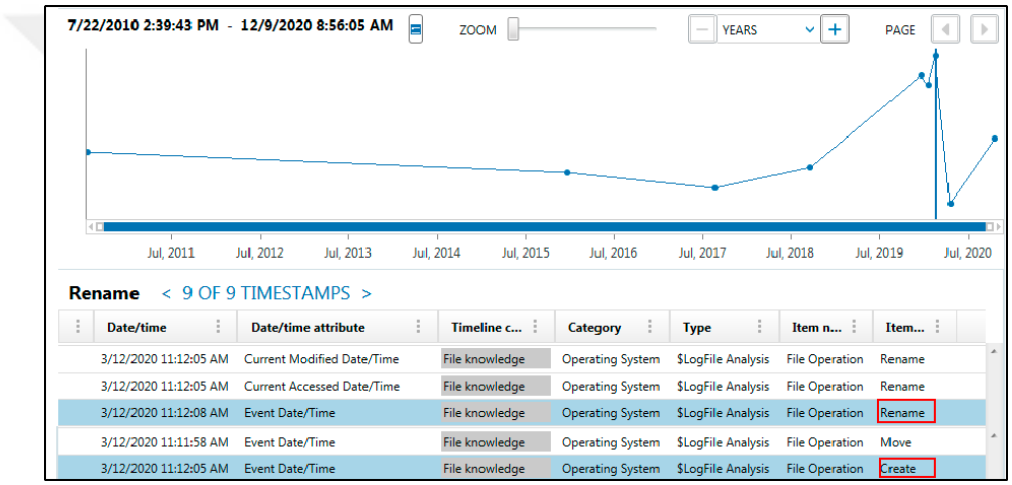
In the following Rename_Move table, we can notice that the original name of the file is “New Text Document” where it was changed to “TEST”, and the indicator for this change is undo and redo where undo represent the original data before the change and redo form the data after the change.

Figure 4.64: Rename transaction results for $LogFile

LogFile Rcd Name	IDX Attr Modified Time	IDX Attr Accessed Time	IDX Attr Created Time	LogFile Rcd Data
\New Text Document.txt	2020-03-12 11:12:05.713	2020-03-12 11:12:05.713	2020-03-12 11:12:05.713	undo
\TEST.txt	2020-03-12 11:12:05.713	2020-03-12 11:12:05.713	2020-03-12 11:12:05.713	redo
\cryptography_tutorial.pdf	2017-09-19 19:57:25.350	2020-03-12 11:11:54.692	2020-03-12 11:11:54.692	undo
\$RECYCLE.BIN\S-1-5-21-285092204-3887931873-1764890242-1000\$RZBUJUU.pdf	2017-09-19 19:57:25.350	2020-03-12 11:11:54.692	2020-03-12 11:11:54.692	redo

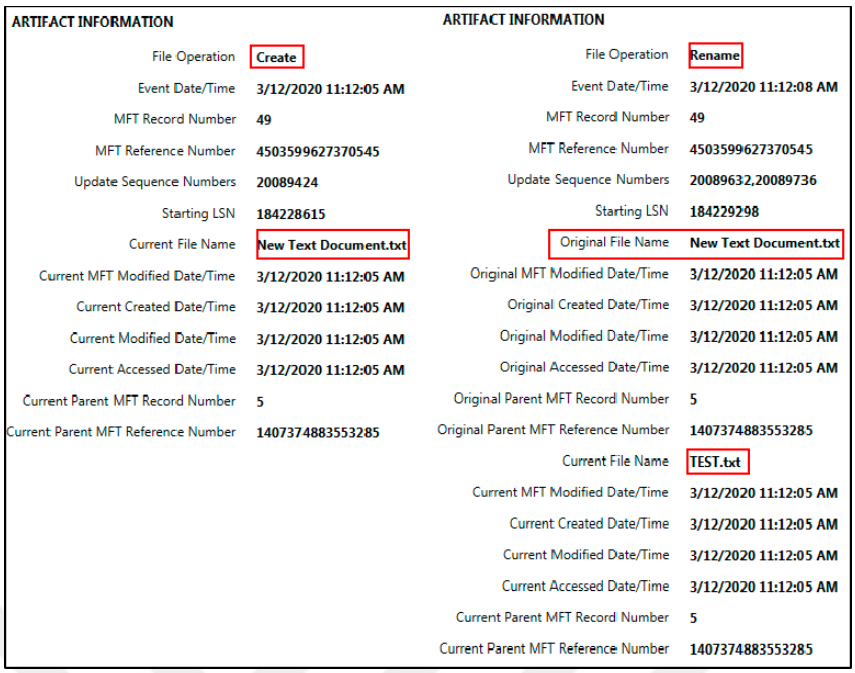
Next, I used "Magnet Axiom" to parse the same evidence. The software shows the results in a timeline graph format, and again if we search for the same .txt file from the above example we can notice that there are two transition records create and rename as it is shown in Figure 4.65.

Figure 4.65: $LogFile timeline graph using Axiom



Further below by clicking on each one of them, it will show more details on the transaction record that toke place and I can conclude that the original file name and type was "New Text Document.txt" which was created on 12/3/2020 at 11:12:05am then was renamed to "TEST.txt" on the same date at 11:12:08 as it is shown in Figure 4.66.

Figure 4.66: $LogFile detailed information using Axiom



ARTIFACT INFORMATION

File Operation	Create
Event Date/Time	3/12/2020 11:12:05 AM
MFT Record Number	49
MFT Reference Number	4503599627370545
Update Sequence Numbers	20089424
Starting LSN	184228615
Current File Name	New Text Document.txt
Current MFT Modified Date/Time	3/12/2020 11:12:05 AM
Current Created Date/Time	3/12/2020 11:12:05 AM
Current Modified Date/Time	3/12/2020 11:12:05 AM
Current Accessed Date/Time	3/12/2020 11:12:05 AM
Current Parent MFT Record Number	5
Current Parent MFT Reference Number	1407374883553285

ARTIFACT INFORMATION

File Operation	Rename
Event Date/Time	3/12/2020 11:12:08 AM
MFT Record Number	49
MFT Reference Number	4503599627370545
Update Sequence Numbers	20089632,20089736
Starting LSN	184229298
Original File Name	New Text Document.txt
Original MFT Modified Date/Time	3/12/2020 11:12:05 AM
Original Created Date/Time	3/12/2020 11:12:05 AM
Original Modified Date/Time	3/12/2020 11:12:05 AM
Original Accessed Date/Time	3/12/2020 11:12:05 AM
Original Parent MFT Record Number	5
Original Parent MFT Reference Number	1407374883553285
Current File Name	TEST.txt
Current MFT Modified Date/Time	3/12/2020 11:12:05 AM
Current Created Date/Time	3/12/2020 11:12:05 AM
Current Modified Date/Time	3/12/2020 11:12:05 AM
Current Accessed Date/Time	3/12/2020 11:12:05 AM
Current Parent MFT Record Number	5
Current Parent MFT Reference Number	1407374883553285

5. DISCUSSION AND CONCLUSION

The main purpose behind this research is to identify critical Windows artifacts from incident response point of view, and to apply digital forensics to acquire artifacts, understand their nature in term of their location, creation, structure, and then finally present this information in timeline format.

Artifacts covered by this research are within each of Windows operating system XP, 7, and 10 which includes: Task Scheduler, System Resource Usage Monitor, Win32 Service, Update Sequence Number Journal, and Windows LogFile.

The conclusion of researching critical Windows incident response artifacts reveals the testing virtual environment used to perform the research, the different artifacts location within the operating system in addition to methods used to gather and extracting the targeted artifacts using specialized Digital Forensics tools, such as FTK Imager especially when dealing with hidden Windows operating systems like $J, $LogFile and $MFT.

Furthermore, this research elaborated on how to populate each of the targeted artifacts and if they need user interaction in order to populate the needed data or it is done automatically by the operating system.

The most important and valuable part of the research is dissecting the structure of each artifact which is explained in the finding section in the most detailed possible, where I explained that some parts of the artifact use specific byte ordering called Endianness either little Endian or big endian and how it is helpful when it comes to converting this information to a human readable format to get the time information by using DCode.

The outcome of this research will help Incident Response and Digital Forensics investigators in answering Who did What, When, Where, and Why. It is also a valuable research that can be used by programmers to develop parsers for specific artifacts.

I think the biggest achievement of this academic thesis is that it is used practically in developing the “Artifast” software by Digisecure development team. I am proud to say that Task Scheduler, Win32 services, and LogFile researched artifacts has been already used in the development, and other artifacts will follow the same path.

This is only the start and soon will see the final product with international impact and financial benefit too.

REFERENCES

Books

Anson, S., 2020. Applied IncidentResponse, 1st Edition. Indianapolis, Indiana: John Wiley & Sons, Inc.

Bell, W., 2007. Reverse Engineering, 1st Edition. 1819, Bhagirath Palace,Chandni Chowk, Delhi-110 006: Global Media.

Casey, E., 2009. Handbook of Digital Forensicsand Investigation, 1st Edition. 30 Corporate Drive, Suite 400, Burlington, MA 01803, USA:Academic Press.

Carrier, B., 2005. FILE SYSTEM FORENSIC ANALYSIS, 1st Edition. One Lake Street Upper Saddle River, NJ 07458: Pearson Education, Inc.

Carvey, H., 2012. Windows Forensic Analysis Toolkit: Advanced Analysis Techniques for Windows 7, 3rd edition. Kolde, J.ed. 225 Wyman Street, Waltham, MA 02451, USA: Syngress.

Carvey, H., 2016. WINDOWS REGISTRY FORENSICS: Advanced Digital Forensic Analysis of the Windows Registry, 2nd edition. 50 Hampshire Street, 5th Floor, Cambridge, MA 02139, USA: Syngress.

Eilam, E. 2005, Reversing: Secrets of Reverse Engineering, 1st Edition. 10475 Crosspoint Boulevard Indianapolis, IN 46256: Wiley Publishing, Inc.

Farmer, D. and Venema, W., 2004. Forensic Discovery, 1st Edition. One Lake Street Upper Saddle River, NJ 07458: Addison-WesleyProfessional.

Forristal, J., 2001. Hack Proofing Your Web Applications. 800 Hingham Street Rockland, MA 02370: Syngress Publishing, Inc.

Honeycutt, J., 2005. Microsoft WINDOWS REGISTRY GUIDE, 2nd Edition. Redmond, Washington 98052-6399: Microsoft Press.

Luttgens, J. and Pepe, M. and Mandia, K., 2014. Incident Response & Computer Forensics. 3rd editon. New York, Chicago, San Francisco, Athens, London, Madrid, Mexico City, Milan, New Delhi, Singapore, Sydney, Toronto: McGrawHill Education.

Malin, C., and Casey, E., and Aquilina, J., 2012. Malware Forensics Field Guide for Windows Systems Digital Forensics Field Guides, 1st Edition. 225 Wyman Street, Waltham, MA 02451, USA: Syngress.

Messier, R., 2016. OPERATING SYSTEM FORENSICS, 1st Edition. 225 Wyman Street, Waltham, MA 02451, USA: Syngress.

Parker, D., 1986. Computer Crime: Criminal Justice Resource Manual. Washington, D.C.: National Institute of Justice.

Pehnack, A., 2015. How to Approach Binary File Format Analysis: Essential knowledge for reverse engineering. PO Box 803338 #62838, Chicago IL 60680-3338, USA: Clearance Center, Inc.

Prosise, C., and Mandia, K., 2003. Incident Response & Computer Forensics, 2nd edition. New York, Chicago, San Francisco, Lisbon, London, Madrid, Mexico City, Milan, New Delhi, SanJuan, Seoul, Singapore, Sydney, Toronto: McGraw-Hill/Osborne.

Rajnovic, D., 2011. Computer Incident Response and Product Security, 800 East 96th Street Indianapolis, IN 46240 USA: Cisco Press.

Shaaban, A. and Sapronov, K., 2016. Practical Windows Forensics: Leverage the power of digital forensics for Windows systems. Livery Place 35 Livery Street Birmingham B3 2PB, UK: Packt Publishing.

Schweitzer, D., 2003. Incident Response: Computer Forensics Toolkit, 10475 Crosspoint BoulevardIndianapolis, Indiana:Wiley Publishing, Inc.

Solomon, D. and Russinovich, M., 2012. Windows Internals. 6th edition, Part1. Redmond, Washington: Microsoft Press.

Other Sources

Ali, I., & Meghanathan, N. 2011. Virtual machines and networks-installation, performance study, advantages and virtualization options. International Journal of Network Security & Its Applications (IJNSA) 3(1).

Cho, G., and Rogers, M. 2012. Finding Forensic Information on Creating a Folder in $LogFile of NTFS. 1 Gyochon, Youngju, Kyoungbuk, 750-711, Dept. Of Computer Information Warfare, Dongyang Univ: Republic of Korea, 401 N Grant St. W. Lafayette, IN, 47907, Dept. of Computer & Information Technology, Purdue Univ: USA.

Day, R. 2020, The 10 Most Important IT Skills for 2020, [Online], Global Knowledge, https://www.globalknowledge.com/us-en/resources/resource-library/articles/the-10-most- important-it-skills-for-2020/
[Accessed 16 September 2020].

Duranec, A., & Topolcic, D., & Hausknecht, K., & Delija, D. 2019. Investigating file use and knowledge with Windows 10 artifacts. INsig2 d.o.o., Zagreb, Croatia, Zagreb University of Applied Sciences: MIPRO.

Khatri, Y. 2015, Forensic implications of System Resource Usage Monitor (SRUM) data in Windows 8. 163 South Willard Street, P O Box 670, VT 05402-0670, USA: ELSEVIER.

Liu, S. 2020, Desktop OS market share 2013-2019, [Online], The Statistics Portal for Market Data, https://www.statista.com/statistics/218089/global-market-share-of-windows-7/
[Accessed 15 September 2020].

Masjedi, A.A. 2012. A Study on the performance of virtualization programs. Degree of Master of Computer and Information Sciences, Auckland University of Technology.

Naik, S., and Bhovi, I., and Naik, G P., and Naik, P., 2019. Screen Monitoring using Windows Service. International Research Journal of Engineering and Technology (IRJET). Volume: 06 Issue: 05, p. 1.

Pan, Y. 2020, edX, [Online], https://courses.edx.org/courses/course-v1:RITx+CYBER502x+2T2020/course/
[Accessed 27 August 2020].

Shakeel, I. 2015. Introduction to Computer Forensics & Digital Investigation. [Online], https://resources.infosecinstitute.com/topic/introduction-to-computer-forensics-digital-investigation/
[Accessed 2 November 2020].

Taylor, L. 2017. ESE Deep Dive: Part 1: The Anatomy of an ESE database, [Online], Microsoft Tech Community, https://techcommunity.microsoft.com/t5/ask-the-directory-services-team/ese-deep-dive-part-1-the-anatomy-of-an-ese-database/ba-p/400496
[Accessed 14 October 2020].