ConSecur
[security and consulting]

HOCHSCHULE MITTWEIDA
UNIVERSITY OF APPLIED SCIENCES

MASTERARBEIT

Herr
Dominic Helfer

Korrelation von Zeitstempeln und Pfadangaben von Ausführungsartefakten eines Windows 1x Systems

2022

Fakultät **Angewandte Computer- und Biowissenschaften**

MASTERARBEIT

Korrelation von Zeitstempeln und Pfadangaben von Ausführungsartefakten eines Windows 1x Systems

Autor:
Dominic Helfer

Studiengang:
Cybercrime/Cybersecurity

Seminargruppe:
CY20wC-M

Erstprüfer:
Prof. Ronny Bodach, Hochschule Mittweida

Zweitprüfer:
M. Sc. Felix Rothe, ConSecur GmbH

Einreichung:
Mittweida, 16. September 2022

Faculty of **Applied Computer and Life Science**

MASTER THESIS

Correlation of timestamps and path information of execution artifacts of a Windows 1x system

author:
Dominic Helfer

course of studies:
Cybercrime/Cybersecurity

seminar group:
CY20wC-M

first examiner:
Prof. Ronny Bodach, Hochschule Mittweida

second examiner:
M. Sc. Felix Rothe, ConSecur GmbH

submission:
Mittweida, 16th september 2022

Bibliografische Angaben

Helfer, Dominic: Korrelation von Zeitstempeln und Pfadangaben von Ausführungsartefakten eines Windows 1x Systems 99 Seiten, 16 Abbildungen, Hochschule Mittweida, University of Applied Sciences, Fakultät Angewandte Computer- und Biowissenschaften

Masterarbeit, 2022

Satz: Diese Arbeit wurde mit LaTeX angefertigt.

Referat

Die Sicherheitslage in Deutschland wird für das Berichtsjahr 2021 durch das Bundesamt für Sicherheit in der Informationstechnik (BSI) als angespannt bis kritisch beschrieben. Ein Grund für diese Einschätzung ist, die Zunahme von Ransomware-Angriffen und die daraus resultierenden Schäden. Derartige Angriffe müssen im Rahmen eines Incident Response und Digital Forensic (kurz DFIR) Prozess aufgeklärt, eingedämmt und weitgehend rückgängig gemacht werden. Die immer größer werdenden Mengen an heterogenen Daten und die immer kürzeren Zeitabschnitte, die für eine Analyse zur Verfügung stehen, sind Herausforderungen, mit denen die Incident Responder und die Digitalen Forensiker konfrontiert sind.

Diese Arbeit verfolgt das Ziel, den Aufwand einer forensischen Untersuchung zu verringern, in dem Ausführungsartefakte von Windows 1x Systemen anhand der in ihnen enthaltenen Pfadangaben und Zeitstempel miteinander korreliert werden. Die praktische Anwendbarkeit der in dieser Arbeit gewonnen Erkenntnisse wurde in einem Proof-of-Concept demonstriert. Dessen Umsetzung erfolgte mithilfe von Angular, Flask und Neo4j.

Abstract

The security situation in Germany is described as tense to critical by the Federal Office for Information Security (BSI) for the 2021 reporting year. One reason for this assessment is the increase in ransomware attacks and the resulting damage. Such attacks must be analyzed, contained and recovered as part of an Incident Response and Digital Forensics (DFIR) process. The increasing amount of heterogeneous data and the shorter periods of time available for analysis are challenges that incident responders and digital forensic scientists are confronted with.

The aim of this thesis is to reduce the effort of a forensic investigation by correlating execution artifacts of Windows 1x systems using the path information and timestamps contained in them. The practical applicability of the knowledge gained in this work was demonstrated in a proof-of-concept. It was implemented using Angular, Flask and Neo4j.

I. Inhaltsverzeichnis

II. Abbildungsverzeichnis

III. Tabellenverzeichnis

IV. Abkürzungsverzeichnis

BAM	Background Activity Moderator
CDP	Connected Devices Platform
CLI	Comandline Interface
DAM	Desktop Activity Moderator
DFIR	Digital Forensic and Incidence Response
EinhZeitG	Gesetz über die Einheiten im Messwesen und die Zeitbestimmung (Einheiten- und Zeitgesetz)
ESR	Extended Support Release
GIMP	GNU Image Manipulation Program
GMT	Greenwich Mean Time
GUI	Graphical User Interface
GUI	grafische Benutzeroberfläche (engl. *Graphical User Interface*)
ISO	internationale Organisation für Standardisierung (engl. *International Organization for Standardization*)
KAPE	Kroll Artifact Parser and Extractor
KS-Test	Kolmogorov-Smirnov-Test
MESZ	mitteleuropäische Sommerzeit
MEZ	mitteleuropäische Zeit
NTFS	New Technology File System
NTP	Network Time Protocol
PDO	Physical Device Object
Plaso	Plaso Langar Að Safna Öllu
RTC	Echtzeituhr (engl. *Real Time Clock*)
SI	Système international d'unités (franz. Internationales Einheitensystem)
UNC	Universal Naming Convention
UTC	koordinierte Weltzeit (engl. *Coodinated Universal Time*)
VSN	Volume Serial Number
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language

V. Vorwort

Diese Masterarbeit ist während meines Studiums im Studiengang *Cybercrime/Cybersecurity* an der Hochschule Mittweida entstanden. Sie setzt an meiner Bachelorarbeit an und verfolgt das Ziel, Ausführungsartefakte von Windows 1x Systemen, über die in ihnen enthaltenen Pfadangaben und Zeitstempel, miteinander in Verbindung zu setzen. Ein möglicher Weg dieser Korrelation wurde in einem Proof-of-Concept umgesetzt. Ich wünsche Ihnen viel Spaß beim Lesen dieser Arbeit.

Bevor Sie jedoch mit dem Lesen beginnen, möchte ich mich bei den Unterstützern dieser Arbeit bedanken.

An erster Stelle danke ich meinen beiden Betreuern Prof. Dipl.-Ing. Ronny Bodach und M.Sc. Felix Rothe. Ohne Ihre Unterstützung hätte diese Arbeit nicht umgesetzt werden können. Ich möchte mich auch bei der Consecur GmbH und ihrem gesamten Team, besonders jedoch bei Stephan Ilic, bedanken, welche mir durch die Anstellung als Werksstudent diese Arbeit erst ermöglicht haben.

Für das Motivieren und Korrekturlesen meiner Arbeit möchte ich mich bei meiner Freundin Leoni Richter, meiner Mutter Sylvia Helfer, meinem Vater Tilo Wohs und seiner Freundin Kathleen Stoy, sowie meinem Bruder Sebastian Helfer bedanken.

Zu guter Letzt danke ich meinem Freundeskreisen in Meißen und Mittweida, die immer ein offenes Ohr für meine Anliegen hatten.

1 Einleitung

Das Bundesamt für Sicherheit in der Informationstechnik (BSI) [4] beschreibt in ihrem Bericht *Die Lage der IT-Sicherheit in Deutschland 2021* die Sicherheitslage als angespannt bis kritisch. Grund für diese Einschätzung ist unter anderem eine Zunahme von Ransomware-Angriffen und die daraus resultierenden Schäden. Der Branchenverband der deutschen Informations- und Telekommunikationsbranche bitkom e.V. schätzt für 2022, dass der Wirtschaft durch Cyberangriffe ein Schaden von circa 203 Milliarden Euro entstehen wird. [72] Die Professionalität der Angreifer und der Aufwand, den sie zum Erreichen ihrer Ziele betreiben, stieg in den letzten Jahren kontinuierlich an. Zeitgleich müssen Betreiber von Informationstechnik den Aufwand zum Schutz vor solchen Angriffen ständig erhöhen. Findet trotz aller Bemühungen ein Angriff statt, so muss dieser im Rahmen eines Incidence Response and Digital Forensic (kurz DFIR) Prozesses aufgeklärt, eingedämmt und schlussendlich weitgehend rückgängig gemacht werden. Dies stellt die Incident Responder und Digitalen Forensiker vor eine Herausforderung. So muss in immer kürzer werdenden Zeitabschnitten, ein immer größer werdende Menge an heterogenen Daten analysiert werden. [2][7][16][64]

Diese Arbeit verfolgt das Ziel, den Aufwand einer forensischen Untersuchung zu verringern. Dafür sollen Ausführungsartefakte von Windows 1x Systemen anhand der in ihnen enthaltenen Pfadangaben und Zeitstempel miteinander korreliert werden. Diese Arbeit beantwortet die folgenden zwei Fragen:

- In welchen zeitlichen Kontext entstehen Events in den unterschiedlichen Artefakten, die zu ein und derselben Dateiausführung gehören und kann dieser Kontext zur Korrelation der Events verwendet werden?
- Wie verknüpft Windows die unterschiedlichen Pfadformate und gibt es eine Möglichkeit diese Verknüpfung, mithilfe von nicht volatilen Daten, wiederherzustellen, um diese für eine Korrelation zu verwenden?

Im dritten Teil der Arbeit werden die Erkenntnisse genutzt, um einen Proof-of-Concept zu erstellen, der die Möglichkeiten der Korrelation demonstriert. Dafür wurde eine Webapplikationen mithilfe von Angular, Flask und Neo4j entwickelt.

2 Grundlagen

2.1 Zeit und Zeitstempel

Die Zeit ist eine der physikalischen Grundgrößen und wird zur Beschreibung von Abfolge und Dauer eines Ereignisses verwendet [38]. Die Einheit der Zeit ist nach dem internationalen Einheitensystem (kurz *SI*) die Sekunde. Eine SI-Sekunde ist von der Frequenz des Hyperfrequenzübergangs des Grundzustandes des Ceasium-133-Atoms abgeleitet [5][37]. Wie auch bei den anderen Einheiten des internationalen Einheitensystems, können bei der Sekunde die Präfixe wie Milli, Mikro oder Kilo verwendet werden. Die Präfixe größer als 10^0 werden allerdings im alltäglichen Sprachgebrauch nicht so häufig verwendet. So entspricht eine Kilosekunde (1000 Sekunden) 16 Minuten und 40 Sekunden. Stattdessen werden zum Ausdrücken von größeren Zeitwerten die Einheiten Minute, Sekunde, Tag, Woche, Monat und Jahr verwendet. Dabei gilt:

- 1 Minute = 60 Sekunden
- 1 Stunde = 60 Minuten
- 1 Tag = 24 Stunden
- 1 Woche = 7 Tage
- 1 Jahr = 365 Tage bzw. 366 Tage im Schaltjahr

Ursprünglich wurde die Zeit durch die Beobachtung der Phasen der Himmelskörper gemessen. Die Astronomen erkannt schon früh in der Geschichte der Menschheit, dass ein Jahr circa 365 Tag-Nacht-Zyklen aufweist. Später stellten sie fest, dass die Phasen der Himmelskörper nicht genau der 365 Tage Regel folgten, sondern etwas länger dauerten. Aus diesem Grund wurde eine sogenannte Interkalation in die Kalender integriert, um den Fehler der 365 Tage Regel auszugleichen. Eine Interkalation beschreibt dabei eine Zeiteinheit, meist einen Tag oder Monat, der nach einem Algorithmus in das Jahr eingefügt wird. Der im Jahr 1582 durch Papst Georg XIII eingeführte gregorianische Kalender lösten den bis dahin gültigen julianischen Kalender ab. Diese Kalenderreform wurde vollzogen, da sich die christlichen Feste, die durch die Phasen der Himmelskörper bestimmt werden, nicht mehr mit dem Datum des Kalenders deckten. Das Ziel des gregorianischen Kalenders war eine Verbesserung des Interkalationsalgorithmus, um das Kalenderjahr an die astronomischen Phasen anzupassen. Ein Schaltjahr erhielt demnach Ende Februar einen zusätzlichen Tag, den 29. Februar. Um ein Schaltjahr zu bestimmen, wird der Algorithmus, der in Listing 2.1 abgebildet ist, verwendet. Neben der Interkalation legte der gregorianische Kalender den Anfang eines Jahres auf den 1.Januar. Um den bisherigen Verzug des julianischen Kalenders rückgängig zu machen,

folgte bei der Einführung auf den 02. Oktober der 15. Oktober. [5][37][36][38]

```
1
2  def isLeapYear(year):
3    if ((year % 4) == 0) and (
4      (year % 100 != 0) or (year % 400 == 0)):
5      return True
6    return False
```

Listing 2.1: Interkalationsalgorithmus des gregorianischen Kalenders implementiert als Python3 Funktion.

Mithilfe der Bewegung der Himmelskörper wurde auch die Dauer eines Tages und somit die Uhrzeit ermittelt. Nimmt man an, dass 12:00 Uhr genau dann ist, wenn die Sonne im Zenit steht, so unterscheiden sich die Uhren zwischen zwei Längengraden um 4 Minuten. Um innerhalb definierter Gebiete mit einheitlichen Uhrzeiten arbeiten zu können, wurden Zeitzonen definiert. Seit der Einführung der koordinierten Weltzeit (kurz *UTC*) im Jahr 1972 werden die Uhrzeiten der verschiedenen Zeitzonen von dieser abgeleitet. UTC beschreibt dabei die Zeit am Nullmedian und wird als *UTC+0:00* oder Greenwich Mean Time (kurz *GMT*) bezeichnet. In Deutschland ist im §4 Absatz 1 EinhZeitG festgelegt, dass die Mitteleuropäische Zeit (kurz *MEZ*) gilt. Diese weicht von UTC positiv um eine Stunde ab (*UTC+1:00*). Zwischen dem 01. März und dem 31. Oktober, beginnend und endend immer an einem Sonntag, der vom Bundesministerium für Wirtschaft und Energie bekannt gegeben wird, gilt allerdings die mitteleuropäische Sommerzeit (kurz *MESZ*). Die MESZ weicht von UTC positiv um zwei Stunden ab (*UTC+2:00*). Die Einführung der Sommerzeit soll nach §5 Absatz 1 EinhZeitG einer „[...] besseren Ausnutzung der Tageshelligkeit [...]“ dienen. [77][80]

2.1.1 Zeit- und Datumsangaben

Um die internationale Kommunikation und den Informationsaustausch zu vereinfachen, standardisiert die internationale Organisation für Standardisierung (kurz *ISO*) in der ISO-Norm 8601 das Format für die Angabe von Zeit. Die Norm, die den Titel *Date elements and interchange formats - Information interchange - Representation of dates and times* trägt, definiert, dass die Woche an einem Montag beginnt und die erste Kalenderwoche eines Jahres in der Woche ist, in welcher der erste Donnerstag im Januar liegt. Als Basis für die ISO 8601 wird der gregorianische Kalender verwendet. Die Gültigkeit der Norm wird allerdings auch auf die Zeit vor der Einführung des gregorianischen Kalenders erweitert. Die Angabe von Datum und Uhrzeit erfolgt in absteigender Reihenfolge von der größten zur kleinsten Zeiteinheit. Eine mögliche Zeitangabe nach ISO 8601 ist 2020-12-31T22:55:10+2:00. Diese beschreibt den 31. Dezember 2020 um 22:55 Uhr und 10 Sekunden in der Zeitzone, die UTC zwei Stunden voraus ist. In Deutschland ist die ISO Norm in die DIN 5008 übernommen wurden. Aufgrund der geringen Akzeptanz der Schreibweise nach ISO 8601 wurde im Jahr 2001 festgelegt, dass weiterhin die in Deutschland übliche Schreibweise *Tag.Monat.Jahr* für Datumsangaben zulässig

ist. Seit 2020 soll diese nur noch im innerdeutschen Schriftverkehr verwendet werden. [76]

2.1.2 Zeit in Computersystemen

Zum Messen der Zeit werden gleichmäßig und periodisch ablaufende Prozesse verwendet. In der frühen Geschichte der Zeitmessung wurden dafür die Phasen der Himmelskörper verwendet. Diesem Prinzip folgten ab dem Mittelalter mechanisch angetriebene Apparaturen zur Zeitmessung. Im Jahre 1921 wurde die erste Quarzuhr vorgestellt, welche als periodischen Prozess die Frequenz eines Schwingquarzes verwendete. [79][80]

In Computersystemen werden zwei verschiedene Uhren für die Zeitmessung verwendet. Die Real Time Clock (kurz *RTC*), die auch als Hardwareuhr bezeichnet wird, ist eine Quarzuhr. Damit sie die Zeit auch zählt, wenn der Computer ausgeschaltet ist, wird sie mithilfe einer Batterie betrieben. Beim Start des Systems wird die Hardwareuhr ausgelesen und ihr Wert der zweiten Computeruhr, der sogenannten Software- oder Systemuhr, übergeben. Diese wird vom Betriebssystem verwendet, um zeit-basierte Funktionen, wie das Setzen von Zeitstempeln oder das Anzeigen der aktuellen Uhrzeit, zu realisieren. Während des Betriebs des Systemes kann die Uhr über verschiedene Synchronisationsmethoden vom Abdriften abgehalten werden. Beim Herunterfahren wird der aktuelle Wert der Softwareuhr der Hardwareuhr übergeben, sodass der korrekte Wert während des Starts wieder zur Verfügung steht. Von Betriebssystem zu Betriebssystem unterscheidet sich der verwendete Zeitstandard der Hardwareuhr. Microsoft Windows Betriebssysteme betrachten und stellen die Hardwareuhr auf die in der Zeitzone des Systems gültigen Zeit. Dieses Verhalten lässt sich mithilfe des Registry Keys `HKLM\SYSTEM\CurrentControlSet\Control\TimeZoneInformation\-RealTimeIsUniversal` unterbinden. Das MacOS Betriebssystem betrachtet von vornherein die Hardwareuhr als UTC Zeit. Bei Unix/Linux Systemen unterscheidet sich der verwendete Zeitstandard zwischen den einzelnen Distributionen. [3]

2.1.3 Formate von Zeitstempel in Computersystemen

An verschiedenen Stellen in Computersystemen werden Zeitstempel verarbeitet oder gespeichert. Das Format, in dem die Zeitstempel vorgehalten werden, unterscheidet sich allerdings von dem für den Menschen verständlichen Zeitstempelformaten ISO 8601 oder DIN 5008. Der Grund: Zeitstempel im Format Jahr-Monat-Tag Stunde:Minute:-Sekunde.Sekundenbruchteil lassen sich in einem Computer nicht platzsparend und effektiv speichern und verarbeiten. Im Laufe der Jahre wurden für die verschiedenen Betriebssysteme, Dateisysteme oder Anwendung eine Reihe von Zeitstempelformate entwickelt.

Das im Betriebssystem Microsoft Windows und im New Technology File System (kurz NTFS) am meisten verbreitete Format ist Filetime. Filetime wird in einem ganzzahligen und vorzeichenlosen 64 Bit Wert gespeichert. Dieser gibt die 100 Nanosekunden seit der Einführung des gregorianische Kalender (siehe Abschnitt 2.1), dem 1601-01-01T00:00+0:00, an. In Unix/Linux-Systemen werden die Zeitstempel in einem ganzzahligen und vorzeichenbehafteten 32 Bit Wert gespeichert. Das Epochtime genannte Format speichert die Sekundenabweichung vom 1970-01-01T00:00+0:00. Im Gegensatz zur Filetime besteht bei der Epochtime die Möglichkeit, auch eine Zeitangabe für Werte vor dem Startzeitpunkt anzugeben, da der 32 Bit Wert vorzeichenbehaftet ist. Die Zeitstempelformate in Apple Dateisystemen unterscheiden sich je nach Typ des Dateisystems. In HFS+ werden Zeitstempel in einem ganzzahligen und vorzeichenlosen 32 Bit Wert gespeichert, der die Sekunden seit dem 1940-01-01T00:00+0:00 angibt. Im APFS Dateisystem hingegen werden Zeitstempel in Nanosekunden seit dem 1970-01-01T00:00+0:00 angegeben (ganzzahliger und vorzeichenloser 64 Bit Wert). Alle vorgestellten Zeitstempelformate haben gemeinsam, dass sie ihre Werte in der UTC Zeit angeben. Welches Zeitstempelformat in einer Anwendung verwendet wird, wird durch den Entwickler bestimmt. Dieser kann sich für ein bereits existierendes Format entscheiden oder auch sein eigenes entwickeln. Der Forensiker ist daher bei seiner Untersuchung mit einer Vielzahl von unterschiedlichen Zeitstempelformaten konfrontiert. [43][71]

2.2 Ausführungsartefakte eines Windows 1x Systems

Auf einem Windows 1x System werden bei der Ausführung von Anwendungen verschiedene Artefakte hinterlassen. Diese können in einer forensischen Untersuchung ausgewertet werden und liefern somit wertvolle Hinweise über die ausgeführte Anwendung und ihren Startzeitpunkt. Im folgenden Abschnitt werden Artefakte vorgestellt, die dies ermöglichen. Dabei wird sowohl auf deren Aufbau und ihre eigentliche Funktion innerhalb des Windows Betriebssystems eingegangen.

2.2.1 Get-Process CmdLet

Die Prozessliste enthält die aktuell laufenden Prozesse eines Windows 1x Systems. Sie wird zur Laufzeit im Arbeitsspeicher vorgehalten und kann über die Win32-API abgefragt werden. In der Windows PowerShell ermöglicht das CmdLet `Get-Process` aus dem Modul `Microsoft.PowerShell.Manangement` eine Abfrage der aktuell ausgeführten Prozesse. So lassen sich Informationen, wie der Prozessname, der Startzeitpunkt oder der Pfad zur ausgeführten Executable herausfinden. Informationen über gestoppt Prozesse lassen sich nicht abrufen. [12]

2.2.2 Prefetch Dateien

Das Windows Betriebssystem versucht den Start von Anwendungen mithilfe des Prefetching zu beschleunigen. Dafür werden Dateien und Bibliotheken in den Arbeitsspeicher geladen, bevor diese von einer Anwendung benötigt werden. Um die benötigten Dateien zu ermitteln, zeichnet der Windows Cache Manager beim ersten Start einer Anwendung auf, welche Dateien in den ersten zehn Sekunden angefragt werden. Diese Informationen werden durch *ntkrnlpa.exe* in eine Prefetch-Datei im Verzeichnis `%systemroot%\Prefetch` geschrieben. Ihr Dateiname setzt sich zum einen aus dem Dateinamen der ausgeführten Anwendung, zum anderen aus einem acht Byte langem Hashwert zusammen. Besteht der Dateiname der ausgeführten Anwendung aus mehr als 29 Zeichen, wird er gekürzt. Der Hashwert wird für den vollständigen Pfad zur Anwendung berechnet. Als Pfadformat wird das von der Volume GUID abgelöste `\Device\HardDiskVolume<x>` Format verwendet. Wird eine Anwendung erneut gestartet, erkennt die *ntkrnlpa.exe* anhand des Dateinamens der Prefetch-Datei, ob diese in der Vergangenheit bereits gestartet wurde. Mithilfe der aufgezeichneten Informationen aus der Prefetch Datei stellt sie die benötigten Dateien im Arbeitsspeicher zur Verfügung. Dieselbe Vorgehensweise wird für die Beschleunigung des Bootvorgangs angewendet. In der Windows Registry kann das Verhalten des Prefetchings beeinflusst werden. Die Bedeutung des dafür verantwortlichen Wertes `EnablePrefetcher` im Registry-Schlüssel `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters` ist in Tabelle 2.1 aufgeführt. [35][39][41][69]

Tabelle 2.1: Diese Tabelle zeigt die Bedeutungen des Wertes `EnablePrefetcher` des Registry-Schlüssels `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters`. Diese Tabelle wurde mit Hilfe der Informationen von Microsoft [41] erstellt.

Wert	Bedeutung
0x00	Prefetching ist deaktiviert
0x01	Prefetching des Anwendungsstarts ist aktiviert
0x02	Prefetching des Bootprozesses ist aktiviert
0x03	Prefetching des Anwendungsstarts und des Bootprozesses ist aktiviert

In einer forensischen Untersuchung zählen Prefetch Dateien zu den wertvollsten Informationsquellen, denn sie kommen standardmäßig auf jedem Windows 1x System vor. In ihnen kann ein Forensiker, neben den Zeitstempeln der acht letzten Ausführungen, auch die Pfade zu den Dateien finden, die beim Start der Anwendung geladen werden. Diese Informationen können bei einem Sicherheitsvorfall wichtige Hinweise über die Funktion etwaiger Malware liefern. Des Weiteren lässt sich die Volume Serial Number (kurz *VSN*) und das Erstelldatum der Partition in den Prefetch Dateien finden. Eine Referenz der Dateien einer NTFS-Partition über die MFT Record Number und die MFT Sequence Number ist vorgesehen, scheint aber in Windows 10 und 11 nicht verwendet zu werden. [39][35][69]

In Windows 10 sind erstmals komprimierte Prefetch Dateien zum Einsatz gekommen. Als Kompressionsalgorithmus verwendet Microsoft LZXPRESS Huffman. Der Aufbau einer komprimierten Prefetch Datei ist in der Tabelle 2.2 aufgezeigt.

Tabelle 2.2: Diese Tabelle zeigt den Aufbau einer LZXPRESS Huffman komprimierten Prefetch Datei. Diese Tabelle wurde mithilfe der Informationen von Metz [39] erstellt.

Offset	Länge	Bedeutung
0x00	0x04	Signature `MAM\0x84`
0x04	0x04	Größe der unkomprimierten Prefetch Datei
0x08	0x04	Prüfsumme (Algorithmus ist unbekannt)
0x0C	...	komprimierter Inhalt der Prefetch Datei

Wird diese dekomprimiert, erhält man die eigentliche Prefetch Datei. Da der Aufbau der Prefetch Dateien nicht von Microsoft öffentlich dokumentiert ist, wurde der Aufbau mit Techniken des Reverse Engineering herausgefunden. Trotz einer Beteiligung einer Vielzahl von Forschern konnte der Aufbau bis heute nicht vollständig entschlüsselt werden. Nachfolgend soll der bekannte Aufbau einer dekomprimierten Prefetch Datei eines Windows 1x Systems beschrieben werden. Die Beschreibung ist nicht auf ältere Windows Betriebssysteme anwendbar, da sich der Aufbau der Prefetch Dateien zwischen den Windows Versionen unterscheidet.

Die ersten 92 Bytes der Datei bilden den Datei Header (siehe Tabelle 2.3). In diesem findet man neben der Version der Prefetch Datei, auch deren Größe sowie den Namen der Anwendung und den Prefetch Hashwert.

Tabelle 2.3: Diese Tabelle zeigt den Aufbau des Datei Headers einer Prefetch Datei. Diese Tabelle wurde mithilfe der Informationen von Metz [39] erstellt.

Offset	Länge	Bedeutung
0x00	0x04	Version der Prefetch Datei 17 - wird verwendet in Windows XP und 2003 23 - wird verwendet in Windows Vista und 7 26 - wird verwendet in Windows 8.1 30 - wird verwendet in Windows 10
0x04	0x04	Signatur „SCCA“
0x08	0x04	*unbekannt*
0x0C	0x04	Größe der Prefetch Datei (inklusive Datei Header)
0x10	0x3C	Dateiname der Anwendung (UTF-16 LE)
0x4C	0x04	Prefetch Hashwert
0x50	0x04	evtl. Bootflag (0x01: Anwendung wird während des Bootprozess gestartet)

Dem Datei Header folgt die Datei-Informationsstruktur (engl. *file information*). Deren Aufbau ist abhängig von der Prefetch Version. In Windows 10 kann neben der 216 Byte langen Variante der Prefetch Version 30 auch die 224 Byte lange Variante der Prefetch Version 26 zum Einsatz kommen. In den neueren Windows 10 Builds wird hauptsächlich die 216 Byte lange Datei Informationsstruktur verwendet. In ihr werden die Offsets

und Längen weiterer Strukturen definiert. Des Weiteren enthält sie die Zeitstempel der letzten acht Ausführungen der Anwendungen, sowie die Anzahl der bisherigen Ausführungen. Der Wert für die bisherigen Ausführungen zählt die Anzahl der Ausführungen, seit Erstellung der Prefetch Datei. Der Aufbau der Datei-Informationsstruktur für die Prefetch Version 30 ist in der Tabelle 2.4 dargestellt.

Tabelle 2.4: Diese Tabelle zeigt den Aufbau der Datei Informationsstruktur einer Prefetch Datei in der Version 30. Diese Tabelle wurde mithilfe der Informationen von Metz [39] erstellt.

Offset	**Länge**	**Bedeutung**
0x00	0x04	Offset zu den Datei Metriken (relativ zum Start der Prefetch Datei)
0x04	0x04	Anzahl der Datei Metriken
0x08	0x04	Offset zum Trace Chain Array (relativ zum Start der Prefetch Datei)
0x0C	0x04	Anzahl der Einträge im Trace Chain Array
0x10	0x04	Offset zu den Dateinamen (relativ zum Start der Prefetch Datei)
0x14	0x04	Größe der Dateinamen in Byte
0x18	0x04	Offset zu den Volumeinformationen (relativ zum Start der Datei)
0x1C	0x04	Anzahl der Volumes
0x20	0x04	Größe der Volumeinformationen
0x24	0x08	*unbekannt*
0x2C	0x08 x 0x08 = 0x40	Zeitstempel der letzten acht Ausführung (Filetime oder 0)
0x6C	0x08	*unbekannt*
0x74	0x04	Anzahl der Ausführungen der Anwendung
0x78	0x04	*unbekannt*
0x7C	0x04	*unbekannt*
0x80	0x58	*unbekannt*

Mithilfe der Informationen aus der Datei-Informationsstruktur ist es möglich die Datei-Metriken zu finden. Diese sind als Array aufgebaut und beinhalten Informationen, um die Pfadangaben der zu ladenden Datei zu finden. In den Prefetch Version 26 und 30 wird der Aufbau der Dateimetriken wie in der Version 23 verwendet (siehe Tabelle 2.5). Ein Eintrag im Array ist immer 32 Byte lang. Die in den Dateimetriken vorgesehene NTFS Referenz scheint in Windows 10 und 11 nicht gefüllt zu werden. Die Dateinamen, deren Offset in der Datei Metrik beschrieben wird, sind in UTF-16 Little Endian codiert und enden mit dem End-Of-String Zeichen (0x00 0x00).

Die Volumeninformationen, deren Offset in der Datei-Informationsstruktur angegeben wird, sind in der Prefetch Version 30 immer 96 Byte lang. In ihr wird der Offset zum Gerätepfad der Partition, der Zeitstempel, an dem die Partition erstellt wurde und die Volume Serial Number gespeichert. Des Weiteren werden die Offsets zur Dateireferenz, die als Pointer zu den MFT Records angelegt sind, hier gespeichert. Ebenfalls enthalten ist der Offset zu den Verzeichnissen. Der Aufbau eines Volumeninformationseintrages ist in Tabelle 2.6 abgebildet.

Tabelle 2.5: Diese Tabelle zeigt den Aufbau eines Datei-Metrik Eintrages einer Prefetch Datei in der Version 23. Dieser hat sich in den Versionen 26 und 30 nicht geändert und ist auch für diese Versionen gültig. Diese Tabelle wurde mithilfe der Informationen von Metz [39] erstellt.

Offset	Länge	Bedeutung
0x00	0x04	*unbekannt*
0x04	0x04	*unbekannt*
0x08	0x04	*unbekannt*
0x0C	0x04	Offset zum beschriebenen Dateinamen (relativ zum Start der Dateinamen Struktur)
0x10	0x04	Anzahl der Zeichen des Dateinamens (End-of-String Zeichen wird nicht mitgezählt)
0x14	0x04	*unbekannt*
0x18	0x08	NTFS Dateireferenz (0, wenn nicht vorhanden) 6 Bytes: MFT Record Number 2 Bytes: MFT Sequence Number

Tabelle 2.6: Diese Tabelle zeigt den Aufbau eines Volumeinformation Eintrages einer Prefetch Datei in der Version 30. Die Tabelle wurde mithilfe der Informationen von Metz [39] erstellt.

Offset	Länge	Bedeutung
0x00	0x04	Offset zum Gerätepfad des Volumes / der Partition
0x04	0x04	Anzahl der Zeichen des Gerätepfades des Volumes / der Partition
0x08	0x08	Zeitstempel an dem das Volume / die Partition erstellt wurde
0x10	0x04	Volume Serial Number
0x14	0x04	Offset zur Dateireferenz
0x18	0x04	Größe der Dateireferenz
0x1C	0x04	Offset zu den Verzeichnisnamen
0x20	0x04	Anzahl der Verzeichnisnamen
0x24	0x04	unbekannt
0x28	0x1C	unbekannt
0x44	0x04	unbekannt

Der Gerätepfad des Volumes wird ebenfalls in UTF-16 Little Endian codiert. Er liegt im Format `\<Erstellzeitpunkt des Volumes in Filetime>-<Volume Serial Number>` vor. Die Datei Referenz ist als Array realisiert. In Windows 10 und 11 scheinen allerdings keine Master File Einträge referenziert zu werden. Die Namen der Verzeichnisse sind ebenfalls in einem Array vorgehalten. Zu Beginn jedes Eintrages wird in einem zwei Byte großen Feld die Anzahl der Zeichen des Verzeichnisnamen ohne das End-of-String Zeichen (0x00 0x00) angegeben. Anschließend folgt der eigentliche Verzeichnisname als UTF-16 LE codierte Zeichenkette.

2.2.3 Windows Timeline

Am 10. April 2018 führte Microsoft, mit dem Update auf Windows 10 Build 1803, eine neue Funktion namens *„Zeitachse“* (engl. *Timeline*) ein. Diese ist ein Teil des Connected Devices Platform (kurz *CDP*) Services, der für den Austausch von Informationen zwischen verschiedenen Geräten, wie Smartphone, XBox One oder Computern, zuständig ist. Die grafische Benutzeroberfläche (kurz *GUI*) der Windows Timeline zeigt, bei einer aktivierten Synchronisierung mit der Microsoft Cloud, die Aktivitäten der letzten 30 Tage an. Ist die Synchronisierung durch den Nutzer deaktiviert wurden, werden nur die Einträge, die jünger als eine Woche sind, angezeigt. Als Benutzer von Windows erreicht man die grafische Benutzeroberfläche der Zeitachse über die Tastenkombination *Windows-Taste + Tabulator* oder über die Schaltfläche *„Aktive Anwendungen“* in der Taskleiste. Die Abbildung 2.1 zeigt die GUI der Windows 10 Timeline. In der GUI werden allerdings nicht alle aufgezeichneten Aktivitäten aufgeführt. Des Weiteren werden nicht alle auf einem System durchgeführten Aktivitäten durch die Windows Timeline aufgezeichnet. Möchte der Benutzer den Aktivitätsverlauf deaktivieren, so kann er dies in den *Einstellungen* unter *Datenschutz* → *Aktivitätsverlauf* machen. Im selben Einstellungsmenü besteht auch die Möglichkeit, den Aktivitätsverlauf, der in der GUI angezeigt wird, zu löschen. Die Aktivitätseinträge in der SQLite Datenbank, die die Grundlage der GUI bildet, bleiben nach dem Löschen über das Einstellungsmenü erhalten. [23][25][42]

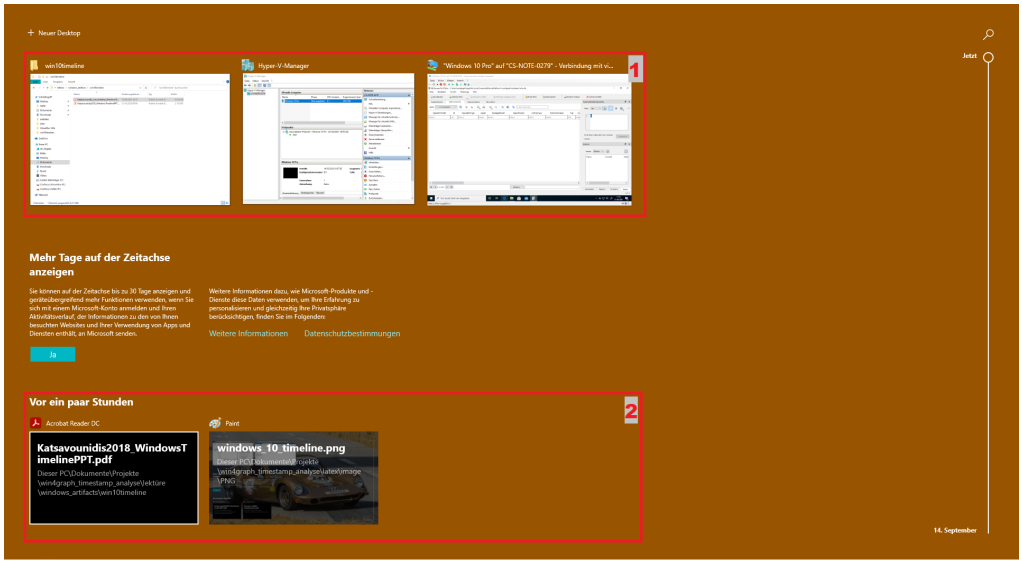


Abbildung 2.1: Diese Abbildung zeigt die Windows Timeline in der grafischen Benutzeroberfläche eines Windows 10 Systems. Die Fläche 1 zeigt die aktiven Anwendungen eines Systems an. In der Fläche 2 werden die in der Vergangenheit liegenden Aktivitäten aufgeführt. Mit einem Mausklick auf die gewünschte Aktivität kann diese erneut geöffnet werden. Abbildung Dominic Helfer

Jeder Benutzer eines Windows 10 Systems besitzt seinen eigenen Aktivitätsverlauf. Dieser ist im Verzeichnis `%LOCALAPPDATA%\ConnectedDevicesPlatform\` angesiedelt. Dieses Verzeichnis enthält eine Reihe von Konfigurationsdateien der Connected

Devices Platform. Die Datei `CDPGlobalSettings.cdp` enthält allgemeine Einstellungen der Connected Devices Platform und ist im JSON-Format angelegt. Ein Auschnitt der Datei und die Bedeutung einiger Einträge ist in Listing 2.2 abgebildet. [23][25]

```
1  {
2     ...
3     "ActivityStoreInfo" : [
4     {
5       "active": true,
6       "activityStoreId": "A1BC9EBC-BC0C-6EF3-5D6D-F3D4DF02E573",
7       "stableUserId": "L.win4graph"
8     }
9     ],
10    "AfcPrivacySettings":  {
11      "ActivityFeed": 0,
12      "CloudSync": 0,
13      "PublishUserActivity": 1,
14      "UploadUserActivity": 1
15    },
16    ...
17 }
```

Listing 2.2: Das Listing zeigt einen Auschnitt aus der Konfigurationsdatei `CDPGlobalSettings.cdp`. Die Zeile 5 zeigt an, ob ein Account aktiv ist und ob die Aktivitäten für diesen Account aufgezeichnet werden. In Zeile 7 ist die BenutzerID des Accounts aufgeführt, für den die Einstellungen gelten. Die ID kann den Aufbau `L.<lokaler Benutzername>` bei einem lokalen Windows 10 Konto, `AAD.xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` bei einem Microsoft Azure Acitive Directory Konto oder eine 16 alphanumerische Zeichen lange Microsoft ID bei einem Microsoft Konto haben. Der Array *ActiveStoreInfo* kann Einträge für mehrere Konten beinhalten. Die Schlüsselwert-Paare des Schlüssels *AfcPrivacySettings*, namens *PublishUserActivity* (Zeile 13) und *UploadUserActivity* (Zeile 14) spiegeln die Checkboxen in den *Einstellungen* unter *Datenschutz –> Aktivitätsverlauf* wieder. Ist eine Checkbox deaktiviert, wird dies in der .cdp Datei als 1, ist sie aktiviert als 0 dargestellt. Der Eintrag in Zeile 13 spiegelt dabei die Checkbox *Meinen Aktivitätsverlauf auf diese Gerät speichern* und der Eintrag in Zeile 14 den der Checkbox *Meinen Aktivitätsverlauf an Microsoft senden* wieder. Die Erläuterungen zu den Bedeutung der Einträge wurden mithilfe von Katasavounidis [23] und Katsavounidis [25] erstellt.

Die Datei `<stableUserID>.cdp` ist ebenfalls eine JSON-Datei und enthält im Eintrag *CNCNotificationLastSynced* einen Zeitstempel, der für das Konto dokumentiert, wann die Timeline zuletzt mit der Microsoft Cloud synchronisiert wurde. Diese Datei existiert für jede, im Eintrag *ActivityStoreInfo* der Datei `CDPGlobalSettings.cdp` aufgeführte, Benutzeridentifikationsnummer. [23][25]

Die eigentliche Timeline wird unter `%LOCALAPPDATA%\ConnectedDevicesPlatform\-<Verzeichnis>\ActivitiesCache.db` gespeichert. Das `<Verzeichnis>` trägt als Na-

men die Benutzeridentifikationsnummer des Kontos, für das ein Aktivitätsverlauf aufgezeichnet wird. Sie wird in einem der drei folgenden Formate angegeben:

- `L.<username>` für ein lokales Windows 10 Konto
- 16 alphanumerische Zeichen (Microsoft ID) für ein Microsoft Konto
- `AAD.xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx` für ein Microsoft Azure Active Directory Konto

Bei der `ActivitiesCache.db` handelt sich um eine SQLite3 Datenbank mit den sieben Tabellen *Activity, ActivityOperation, Activity_PackageId, AppSettings, DataEncryptionKeys, ManualSequence* und *Metadata*. Wird eine neue Aktivität aufgezeichnet, wird zunächst eine Identifikationsnummer, die sogenannte Aktivitätsidentifikationsnummer, geniert und anschließend die Einträge in den Tabellen *Activity* und *Activity_PackageId* generiert. Die *Activity* Tabelle speichert Informationen zu der ausgeführten Aktivität, die Tabelle *Activity_PackageId* beschreibt den Anwendungsnamen sowie die Plattform der Anwendung. Ist die Synchronisation der Aktivitäten mit der Microsoft Cloud aktiviert, werden die Einträge aus der *Activity* Tabelle anschließend als Warteschlange in die *Activity_Operation* Tabelle eingeordnet und nach erfolgreicher Synchronisation wieder gelöscht. Die Tabelle 2.7 beschreibt die bekannten Bedeutungen der Spalten der *Activity* Tabelle. [23][25][32]

In der Tabelle 2.8 ist die Bedeutung der Werte der Spalte *ActivityType* aus der *Activity* Tabelle beschrieben. Der Wert des *ActivityType* beeinflusst, wie die Felder der *Activity* Tabelle gefüllt und interpretiert werden müssen. Je nach Windows 1x Version verändern sich allerdings die Voraussetzungen für eine Befüllung und der ausgefüllte Inhalt der Tabelle.

Nachfolgend soll dargestellt werden, welche Informationen aus der Windows Timeline gewonnen werden können und welche Voraussetzungen bestehen, dass diese aufgezeichnet werden.

Ausführungszeitpunkt einer Anwendung ermitteln

Um den Ausführungszeitpunkt von Anwendung zu ermitteln, werden die Aktivitätseinträge der *Activity* Tabelle betrachtet, die den *ActivityType* 5 besitzen. Wie in Tabelle 2.8 abgebildet, entspricht der Wert 5 dem Öffnen einer Anwendung, Datei oder URL. Die Anwendung, die ausgeführt wurde, kann aus der Spalte *AppId* der *Activity* Tabelle entnommen werden. Alternativ könnte die Aktivitätsidentifikationsnummer des Aktivitätseintrag verwendet werden, um in der *Activity_PackageId* Tabelle die zugehörigen Anwendungseinträge herauszufiltern. Auf den Startzeitpunkt der Anwendung kann mithilfe der Spalte *StartTime* und auf den Zeitpunkt, an dem die Anwendung geschlossen

Tabelle 2.7: Diese Tabelle zeigt den Aufbau der Activity Tabelle der ActivitiesCache.db. Es werden nur die Spalten aufgeführt, deren Bedeutung bekannt ist. Diese Tabelle wurde mithilfe der Informationen von Katsavounidis [25] und Katasavounidis [23] erstellt.

Name der Spalte	Bedeutung
Id	Identifikationsnummer der Aktivität (16 Byte lange GUID)
AppId	Wert ist ein JSON-String, enthält die Plattform und den Namen der Anwendung (teilweise auch den Pfad zur Anwendung)
PackageIdHash	„QuickXor-Hash„ der Anwendung
AppActivityId	kann die URL oder Datei beinhalten, die in einer Anwendung geöffnet wurde
ActivityType	beschreibt die aufgezeichnete Aktivität (siehe Tabelle 2.8)
ActivityStatus	beschreibt den Status eines Eintrages - 1: aktiv - 2: aktualisiert - 3: gelöscht - 4: ignoriert
ParentActivityId	Identifikationsnummer der zum Eintrag übergeordneten Aktivität (16 Byte lange GUID)
Tag	Wenn ActivityType == 11,12 oder 15 wird die Systemoperation präzisiert, welche ausgeführt wurde
Group	Wenn ActivityType == 16, beschreibt der Eintrag ob eine „copy“ oder „paste“ Aktivität stattgefunden hat
LastModifiedTime	Zeitpunkt der letzten Modifikation des Eintrages (Epochtime in UTC)
ExpirationTime	Zeitpunkt an dem der Eintrag seine Gültigkeit verliert (Epochtime in UTC) –> *CreatedTime* + 2592000 Sekunden
Payload	JSON-String, Inhalt ist Abhängig vom Wert der Spalte *ActivityType*
PlatformDeviceId	Identifikationnummer des Systems, kann mit dem Registry-Schlüssel `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\TaskFlow` korreliert werden
StartTime	Startzeitpunkt der Aktivität
EndTime	Endzeitpunkt der Aktivität
LastModifiedOnClient	Zeitpunkt an der Eintrag auf dem aktuellen System das letzte Mal modifiziert wurde
ClipboardPayload	Wenn ActivityType == 10, enthält sie einen JSON-String mit dem Inhalt der Zwischenablage (Base64 codiert)

wurde, mithilfe der Spalte *EndTime* in der *Activity* Tabelle geschlossen werden. Die Zeitstempel liegen im Epochtime-Format vor. Dateien oder URLs, die in der Anwendung geöffnet waren, lassen sich teilweise aus der Spalte *AppActivityId* extrahieren. In der Spalte *Payload* ist dies unter Umständen ebenfalls möglich. Hierbei sind die JSON-Schlüsselwertpaare *description* und *contentUri* von besonderem Interesse, da sie den kompletten Dateipfad oder URL enthalten können. Der Eintrag *displaytext* spiegelt meist den angezeigten Text der Titelleiste und *appdisplayname* den angezeigten Namen der

Anwendung wieder. [23][25][32]

Dauer der Beschäftigung mit einer Anwendung ermitteln

Der *ActivityType* 6 repräsentiert das Betrachten einer Anwendung im Vordergrund durch den Benutzer eines Systems. Die Anwendung, die durch den Aktivitätseintrag beschrieben wird, kann auf dieselbe Weise wie beim *ActivityType* 5 herausgefunden werden. Die Werte der Spalten *StartTime* und *EndTime* haben allerdings einen anderen Kontext als beim *ActivityType* 5. *StartTime* spiegelt wieder, wann ein Nutzer begonnen hat, die Anwendung im Vordergrund zu betrachten. *EndTime* entspricht der Zeit, zu der ein Nutzer die Anwendung nicht mehr im Vordergrund betrachtet hat. In der *Payload* Spalte lässt sich im Schlüssel-Wert Paar *activeDurationSeconds* die Dauer der Beschäftigung mit der Anwendung in Sekunden finden. Der Schlüssel *userTimezone* beschreibt die im System während der Beschäftigung eingestellte Zeitzone.[23][25][32]

Erkennen von Copy-Paste Operationen

Der *ActivityType* 16 dient zum Beschreiben von Copy-Paste Operationen. Die Anwendung, in der diese Art von Operation stattgefunden hat, lässt sich wie bei den bereits vorgestellten Aktivitätstypen herausfinden. In einer Standardkonfiguration von Windows 1x kommt der *ActivityType* 16 allerdings nicht vor. Er wird mit dem Einschalten des

Tabelle 2.8: Diese Tabelle zeigt die Bedeutung der *ActivityType* Werte in der *Activty* Tabelle des ActivitiesCache.db auf. Diese Tabelle wurde mithilfe der Informationen von Katsavounidis [25] und Katasavounidis [23] erstellt.

ActivityType	Bedeutung
2	Benachrichtigung
3	Backup eines Mobilgerätes oder Azure Authentifikation
5	Öffnen einer Anwendung, Datei oder Webseite
6	Betrachten einer Anwendung im Vordergrund
10	Text aus der Zwischenablage
11, 12, 15	Windows Systemoperationen, wie: - Microsoft.Credentials.Vault - Microsoft.Credentials.WiFi - Microsoft.Default - Microsoft.Credentials - Microsoft.Personalization - Microsoft.Language - Microsoft.Acessibility - windows.data.bluelightreduction.settings - windows.data.input.devices.pensyncedsettings - windows.data.bluelightreduction.bluelightreductionstate
16	Copy- und Paste-Operationen

„Zwischenablageverlauf“ in den *Einstellungen* unter *System* → *Zwischenablage* → *Zwischenablageverlauf* aktiviert. Um festzustellen, welche Operation in der Anwendung ausgeführt wurde, kann die *Group* Spalte der *Activity* Tabelle konsultiert werden. Hat eine Copy-Operation stattgefunden, enthält sie das Schlüsselwort *Copy* bei einer Paste-Operation *Paste*. [23][25][34]

Erkennen, auf welchen Gerät eine Aktivität stattgefunden hat

Ist die Synchronisation der Windows 10 Timeline mit der Cloud aktiviert, können in der `ActivitiesCache.db` Aktivitäten von mehren Geräten aufgeführt werden. Diese können in der Tabelle *Activity* anhand der Spalte *PlatformDeviceId* unterschieden werden. Um dieser Identifikationsnummer ein Gerät zuordnen zu können, kann diese ID mit dem Registryschlüssel `NTUSER.DAT\Software\Microsoft\Windows\Windows\-CurrentVersion\TaskFlow\DeviceCache` korreliert werden. Mithilfe dieser Korrelation lässt sich die Marke des Gerätes, die Modellbezeichnung und der Gerätename sowie der Gerätetype herausfinden. Der Gerätetype ist als numerischer Wert codiert und kann mithilfe der Tabelle 2.9 dekodiert werden. [23][25][33]

Tabelle 2.9: Diese Tabelle zeigt die codierten Gerätebezeichnungen im Registryschlüssel `NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\TaskFlow\-DeviceCache` Diese Tabelle wurde mithilfe der Informationen von Katsavounidis [25] und Katasavounidis [23] erstellt.

DeviceType	Bedeutung
0	Windows 10 Dual-Screen Geräte
1	XBox One
6	Apple iPhone
7	Apple iPad
8	Android Gerät
9	Windows 10 Desktop
11	Windows 10 Phone
12	Linux Gerät
13	Windows IoT
14	Surface Hub
15	Windows 10 Laptop
16	Windows 10 Tablet

2.2.4 Windows XML Event Log

In Windows werden Logmeldungen zentral im Windows Event Logging System gesammelt. Im Gegensatz zu Linux Systemen, in denen die Logdateien im Klartext gespeichert werden, verwendet Windows ein binäres Format. Dieses binäre Format ist seit Windows Vista und Windows Server 2008 das *EVTX*-Format, das das seit Windows NT 4.0

verwendete *EVT*-Format ablöste. Der Grund für eine Neuentwicklung lag im hohen Arbeitsspeicherbedarf des *EVT*-Formates. Bei diesem mussten während des Betriebs des Systems die kompletten Logdaten im Arbeitsspeicher vorgehalten werden. Dies änderte sich mit der Einführung von *EVTX*, bei dem nur noch der Datei-Header des Logs sowie der aktuell verwendete Chunk im Arbeitsspeicher vorgehalten werden musste. Eine weitere Neuerung im EVTX-Format war die Verwendung der Extensible Markup Language (kurz *XML*) zur Speicherung der Logmeldungen. Diese ermöglichten, dass die Logdaten anhand des Nachrichteninhaltes durchsucht werden können. Um den Nachteil von XML, dem vielen redundanten Text, aus dem Weg zugehen, entschied sich Microsoft, die Logmeldungen in einem proprietären binären XML-Format zu speichern. [68]

Die Logdateien werden in der Standardkonfiguration im Verzeichnis `%systemroot%\System32\winevt\Logs` gespeichert. Jede in diesem Ordner enthaltene *EVTX*-Datei repräsentiert einen eigenen Log-Kanal, der die für ihn gedachten Logdateien aufnimmt. Die meist verwendeten Log-Kanäle sind `Application.evtx`, `HardwareEvents.evtx`, `Security.evtx`, `Setup.evtx` und `System.evtx`. Der Speicherort der Logdateien dieser Kanäle lässt sich im Registry-Schlüssel `HKLM:\SYSTEM\CurrentControlSet\Services\EventLog\<Log-Kanal>` einsehen und ändern. Die Speicherorte der anderen Kanäle werden unter `HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT\Channels` aufgeführt, wenn diese nicht dem Standardspeicherort entsprechen. [11]

In den *EVTX*-Dateien werden die Events durch eine Event ID kodiert. Erst wenn die Logdaten über die Windows Ereignisanzeige eingesehen werden, werden die Eventbedeutungen mithilfe von DLLs einer Event ID zugeordnet. Die für die Log-Kanäle zuständigen DLLs können in den Registry-Schlüsseln `HKLM:\SYSTEM\CurrentControlSet\Services\EventLog\<Name des Log-Kanals>` und `HKLM:\SOFTWARE\Microsoft\Windows\CurrentVersion\WINEVT` gefunden werden. [11]

Jedes Event enthält, neben den Event-spezifischen Informationen, auch allgemeine Informationen, welche für jedes Element existieren. Diese werden in der XML-Repräsentation der Events in dem *<System>* Element gespeichert. Die Tabelle 2.10 führt alle allgemeinen Informationen auf, die für jedes Event zur Verfügung stehen.

Nachfolgend wird vorgestellt, welche Logeinträge die Ausführung einer Anwendung aufzeichnen, welche Informationen die Logmeldungen enthalten und welche Voraussetzungen erfüllt sein müssen, dass diese Event aufgezeichnet werden.

Event ID 4688 - Ein neuer Prozess wurde erstellt.

In der `Security.evtx` repräsentiert die Event ID 4688 das Ereignis „Ein neuer Prozess wurde erstellt“. In der Standardkonfiguration von Windows werden allerdings nur Anwendungsstarts des Benutzerkontos mit dem Security-Identifier S-1-5-18, welcher nach

Tabelle 2.10: Diese Tabelle zeigt die Einträge der allgemeinen Eventinformationen auf, die für jeden Eventeintrag im Security Logkanal zur Verfügung stehen. Diese Tabelle wurde mit Hilfe der Informationen von Charter [8] erstellt.

Name des Eintrages	Bedeutung
Provider Name Provider Guid	Name des Anbieters des Logeintrages GUID des Anbieters des Logeintrages
EventID	ID des Events (beschreibt die Bedeutung des Events)
Version	Version des Event Eintrages (kann sich zwischen Windows Versionen unterscheiden und kennzeichnet die Revision)
Level	Kategoriersierung des Logeintrages 0 - Information 1 - Kritisch 2 - Fehler 3 - Warnung 4 - Information
Task	
Opcode	numerischer Wert der die Aktivität in der Anwendung beschreibt, die zum Zeitpunkt des auslösen der Logmeldung durchgeführt wurde
Keywords	beschreibt die Kategorie der Logmeldung 0x8020000000000000 - Überwachung erfolgreich 0x8010000000000000 - Überwachung gescheitert
TimeCreated SystemTime	Zeitpunkt an dem das Event eingetreten ist (Zeit in UTC)
EventRecordID	Nummer des Eventeintrages (wird fortlaufend gezählt)
Correlation	ermöglicht die Abbildung das eine Event in einem anderen Event involviert ist
Execution ProcessID Execution ThreadID	Prozessidentifikationsnummer des Prozesses der das Event aufgezeichnet hat Thread der den Prozess aufgezeichnet hat
Channel	Logkanal
Computer	Name des Computers auf dem das Event stattgefunden hat

Microsoft Docs [47] dem Benutzer „Lokales System“ zugeordnet ist, aufgezeichnet. Um alle Anwendungsstarts auf einem System aufzuzeichnen, muss in der *gpedit.msc*, die erst ab der Windows 1x Pro Version zur Verfügung steht, unter *Computerkonfiguration* → *Windows-Einstellungen* → *Sicherheitseinstellungen* → *Erweiterte Überwachungsrichtlinien* → *Systemüberwachungsrichtlinien* → *Detaillierte Überwachung* die Richtlinie

Überwachung der Prozesserstellung aktiviert werden. Diese Überwachung kann sowohl für eine erfolgreiche, als auch für eine fehlgeschlagene Prozesserstellung geschehen. Wurde diese Einstellung getätigt, werden alle Anwendungsstarts für alle Benutzer eines Systems aufgezeichnet und als Event ID 4688 in die `Security.evtx` geschrieben.

Neben den allgemeinen Informationen zu einem Event, die jeder Eventeintrag besitzt, werden die in Tabelle 2.11 aufgeführten Einträge aufgezeichnet. In der Standardkonfiguration wird allerdings der Kommandozeilenaufruf der gestarteten Anwendung nicht mit aufgezeichnet, weshalb der Eintrag *CommandLine* leer bleibt. Soll dies mit aufgezeichnet werden, muss ebenfalls in der *gpedit.msc* unter *Computerkonfiguration* → *Administrative Vorlagen* → *System* → *Prozesserstellung überwachen* die Richtlinie *Befehlszeile in Prozesserstellungsereignisse einschließen* aktiviert werden. Den Zeitpunkt, an dem die Anwendung gestartet wurde, lässt sich aus dem Element *TimeCreated* in den allgemeinen Eventinformationen finden. [45]

Tabelle 2.11: Diese Tabelle zeigt die Bedeutungen der Eventdateneinträge des Event mit der Event ID 4688 im Security Logkanal. Aufgrund der Länge und Redundanz werden nur die Namen der XML Dateneinträge und nicht das komplette XML Element aufgeführt. Diese Tabelle wurde mit Hilfe der Informationen von Microsoft Docs [45] erstellt.

Name des Eintrages	Bedeutung
Beschreibung des Prozesserstellers:	
SubjectUserSid	SID des Benutzers
SubjectUserName	Benutzername
SubjectDomainName	Domain oder Computername des Erstellers
SubjectLogonId	eindeutige ID der Logon-Session (kann mit den Event IDs 4624, 4634 und 4647 einer Logon-Session zugeordnet werden)
Beschreibung des Besitzers des neuen Prozesses	
TargetUserSid	SID des Benutzers
TargetUserName	Benutzername
TargetDomainNAme	Domain oder Computername des Besitzers
TargetLogonId	eindeutige ID der Logon-Session (kann mit den Event IDs 4624, 4634 und 4647 einer Logon-Session zugeordnet werden)
Beschreibung des erstellten Prozesses	
NewProcessId	Prozessidendifikationsnummer des neuen Prozesses (PID)
NewProcessName	Name oder Pfad zur Anwendung
TokenElavationType	
MandantoryLabel	SID der Integritätsbezeichnung des neuen Prozesses
CommandLine	Kommandozeilenaufruf der zur Erstellung verwendet wurde
Beschreibung des Elternprozesses	
ProcesssId	Prozessidenifikationsnummer des Elternprozesses (PPID)
ParentProcessname	Name oder Pfad zur Anwendung

Event ID 4689 - Ein Prozess wurde beendet

Neben der Prozesserstellung kann auch die Beendigung eines Prozesses aufgezeichnet werden. Hierfür ist die Event ID 4689 im Security Log-Kanal vorgesehen. In der Windows Standardkonfiguration werden diese Events allerdings nicht aufgezeichnet. Die Aufzeichnung muss explizit in der *gpedit.msd* unter *Computerkonfiguration* → *Windows-Einstellungen* → *Sicherheitseinstellungen* → *Erweiterte Überwachung* → *Systemüberwachungsrichtlinien* → *Detaillierte Überwachung* aktiviert werden. Die dafür verantwortliche Richtlinie heißt *Prozessbeendung überwachen*. Sie kann, wie auch die Richtlinie der Prozesserstellung, den Erfolgs- und Fehlerfall dokumentieren. Die enthaltenen Eventdateneinträge in einem Event mit der Event ID 4689 sind in Tabelle 2.12 aufgeführt. Der Zeitpunkt, an dem der Prozess beendet wurde, ist in den allgemeinen Informationen des Events im Element *TimeCreated* zu finden. [46]

Tabelle 2.12: Diese Tabelle zeigt die Bedeutungen der Eventdateneinträge des Event mit der Event ID 4689 im Security Logkanal. Aufgrund der Länge und Redundanz werden nur die Namen der XML Dateneinträge und nicht das komplette XML Element aufgeführt. Diese Tabelle wurde mit Hilfe der Informationen von Microsoft Docs [46] erstellt.

Name des Eintrages	Bedeutung
Beschreibung des Subjektes:	
SubjectUserSid	SID des Benutzers
SubjectUserName	Benutzername
SubjectDomainName	Domain oder Computername des Subjektes
SubjectLogonId	eindeutige ID der Logon-Session (kann mit den Event IDs 4624, 4634 und 4647 einer Logon-Session zugeordnet werden)
Beschreibung des beendeten Prozesses	
Status	ExitStatus des Prozesses (wird vom Entwickler der Anwendung festgelegt, allgemeine Konvention: 0 = erfolgreiche Beendung des Prozesses)
ProcessId	Prozessidentifikationsnummer des beendeten Prozesses (PID)
ProcessName	Name oder Pfad des Prozesses

2.2.5 UserAssist

In der Windows Registry zeichnet der UserAssist Registryschlüssel Anwendungen auf, die über den Windows Explorer gestartet wurden [17, S. 211]. Das Ziel dieses Loggingmechanismus ist es, das Betriebssystem interaktiver und benutzerfreundlicher zu gestalten. Der UserAssist Schlüssel wurde mit Windows NT 4.0 eingeführt und ist seither Bestandteil des Betriebssystems. In Windows 10 ist dieses Artefakt immer noch integriert. Der UserAssist Schlüssel ist ein Teil der Registry, genauer gesagt ist es ein

Unterschlüssel im NTUSER.DAT Hive. Da dieser Hive für jeden Benutzer eines Windows 1x Systems zur Verfügung steht, besitzt jeder Benutzer seine eigenen UserAssist Informationen. In ihnen lassen sich die Namen bzw. Pfade der Anwendungen oder die genutzten Verknüpfungen finden, die ausgeführt wurden. Die letzte Ausführung der Anwendung ist mit einem Zeitstempel im Filetime Format dokumentiert. Des Weiteren ist die Anzahl der Ausführungen und die Anzahl und Dauer, die eine Anwendung im Vordergrund betrachtet wurde, dokumentiert. [17][70]

Der NTUSER.DAT Registry Hive wird unter `C:\Users\<Benutzer>\NTUSER.DAT` gespeichert. In diesem befindet sich unter `Software\Microsoft\Windows\CurrentVersion\Explorer\UserAssist` der UserAssist Registryschlüssel. Die Unterschlüssel tragen GUIDs als Namen und sind bis auf die Schlüssel `CEBFF5CD-ACE2-4F4F-9178--9926F41749EA` und `F4E57C4B-2036-45F0-A9AB-443BCFE33D9F` nicht für die Forensik relevant. Jeder Unterschlüssel enthält einen weiteren Unterschlüssel namens `Count`. In diesem werden die eigentlichen UserAssist Informationen als Key-Value Paare abgelegt. Die Werte sind im binär Format gehalten, und deren Bedeutung ist bei beiden GUIDs gleich (siehe Tabelle 2.13). Die Bedeutung der Namen unterscheidet sich allerdings zwischen den GUIDs. Die Informationen im Unterschlüssel `CEBFF5CD-ACE2-4F4F--9178-9926F41749EA` enthalten die ausgeführten Anwendungen, wobei der Pfad einer Anwendung als Name des Wertes angegeben ist. Im Unterschlüssel `F4E57C4B-2036--45F0-A9AB-443BCFE33D9F` werden die zur Ausführung benutzten Verknüpfungen (LNK Dateien) aufgeführt. Der Pfad der LNK ist als Name des Wertes dokumentiert. Die Pfade der Anwendungen oder der LNK Dateien sind ROT-13 codiert und können relativ mithilfe von Verzeichnis-GUIDs angegeben werden. [70]

Tabelle 2.13: Diese Tabelle zeigt den Aufbau eines UserAssist Eintrages. Diese Tabelle wurde mit Hilfe der Informationen von Singh und Singh [70] erstellt.

Offset	Länge	Bedeutung
0x00	0x04	*unbekannt*
0x04	0x04	Anzahl der Ausführungen (weicht von dem Wert in den Prefetch Dateien ab, da die Prefetchdateien nicht benutzerspezifisch sind)
0x08	0x04	Anzahl, wie oft die Anwendung im Vordergrund war
0x0C	0x04	Gesamtzeit in Millisekunden die eine Anwendung im Vordergrund war
0x10	0x2C	*unbekannt*
0x3C	0x08	Zeitpunkt der letzten Ausführung (Filetime in UTC)
0x44	0x04	immer `00 00 00 00`

2.2.6 Windows Background Activity Moderator und Desktop Activity Moderator

Hinweis: Absatz 1 und 2 dieses Abschnittes wurde bereits in der Bachelorarbeit „Untersuchung von forensischen Artefakten eines Windows 10 Systems im Hinblick auf Kor-

relationsmöglichkeiten für eine automatisierte Verknüpfung“ von Helfer, Ilic und Bodach [19] veröffentlicht.

Mit Windows 8 führte Microsoft den Desktop Activity Moderator (kurz *DAM*) ein, um die Akkulaufzeit von Geräten im „Connected-StandBy“ Modus zu erhöhen. Der „Connected-StandBy“ Modus beschreibt den Zustand des Gerätes, bei dem der Bildschirm ausgeschaltet, das Gerät selbst aber eingeschaltet ist und Anwendungen auf ihm laufen (z. B. Voice-Over-IP, Windows Store, etc.). Das Ziel, die Akkulaufzeit zu verlängern, erreicht DAM, indem es die Ausführung von Desktop Prozessen einschränkt, sodass jene Prozesse ausgeführt werden können, die für den „Connected-StandBy“ Modus konstruiert worden sind. [40] Bedingt dadurch, dass der Desktop Activity Moderator nur bei mobilen Geräten, wie Telefonen oder Tabletts, zum Einsatz kommt, lässt sich das Artefakt nur auf diesen Geräten finden [58]. Der Background Activity Moderator (kurz *BAM*) soll ebenfalls die Akkulaufzeit verlängern. Dies wird erreicht, indem der Background Activity Moderator Hintergrundprozesse überwacht und deren Energiebedarf einschränkt [9]. Beide Artefakte können nur bei Geräten aufgefunden werden, die im Batteriebetrieb verwendet worden sind.

Mit beiden Activity Moderatoren ist es für den Forensiker möglich, die Ausführung von Anwendungen in Windows nachzuvollziehen. Beide Artefakte werden in der Registry gespeichert und enthalten Informationen über die SID des Benutzers, der eine Anwendung ausgeführt hat, sowie den Pfad zur ausführbaren Datei und den Zeitstempel der letzten Ausführung. Im Registry Key `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\bam\State\UserSettings` sind die Informationen zum Background Activity Moderator zu finden. Die Informationen des Desktop Activity Moderator werden im Schlüssel `HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\dam\State\UserSettings` gespeichert. [10][58]

Die BAM beziehungsweise DAM Informationen befinden sich in dem jeweiligen Unterschlüssel, der die Security ID des Benutzers als Namen trägt. Der Name eines Eintrag enthält immer den Namen des ausgeführten Pakets oder den Dateipfad der ausgeführten Anwendung. In den binären Daten des Eintrages repräsentieren die ersten acht Byte den Zeitpunkt der letzten Ausführung. Der Zeitstempel liegt im Filetime Format vor und gibt die Zeit in UTC an. [24]

2.3 Zeitstempel in der digitalen Forensik

Die Real Time Clock (kurz *RTC*) eines Computersystems neigt dazu von der physikalischen Zeit (der tatsächlichen Zeit) abzuweichen. Dieses Verhalten wird auch als abdriften bezeichnet und ist abhängig von der Umgebung, in der eine RTC betrieben wird. Die in einem System verwendete Softwareuhr kann allerdings auch von der tatsächlichen Zeit abweichen. Der Verlauf der Abweichung ist dabei theoretisch kontinuierlich. Schatz,

Mohay und Clark [67] fanden experimentell heraus, dass der Verlauf des Abdriftens nicht immer kontinuierlich ist, sondern Anomalien aufweisen kann. Diese Anomalien können auf fehlerhafte Implementierungen der Hardware- und Softwareuhr, Synchronisationsfehler oder der Einflussnahme des Betriebssystems oder installierter Anwendungen auf die Zeit zurückgeführt werden.

Damit unabhängige Uhren vergleichbar sind, bedürfen sie einer Synchronisierung. Dabei reicht es nach Lamport [27] nicht aus, wenn die Uhren eine gleiche Zählgeschwindigkeit haben, um synchron zu laufen. Stattdessen muss die Abweichung zwischen den beiden Uhren aktiv minimiert werden. Schatz, Mohay und Clark [67] demonstrierten, dass Clients in einem Windows 2000 und XP Netzwerk synchron bleiben, wenn diese über ihren Windows 2000 Domaincontroller zeitlich synchronisiert werden. Der Domaincontroller bezog seine Systemzeit von einem NTP-Sever.

Auch bei einer aktiven Zeitsynchronisierung besteht das Problem, dass die Uhr eines Systems manipuliert sein kann. Um zu überprüfen, ob und wie viel eine Uhr von der tatsächlichen Zeit abweicht, stellte Schatz, Mohay und Clark [67] einen Ansatz vor, der die Zeitstempel der Einträge eines Web Proxy mit den Zeitstempel der Einträge im Browserverlauf vergleicht. So ist es möglich, die Abweichung der Systemzeit von der bürgerlichen Zeit zu bestimmen. Willassen [81] präsentierte einen Ansatz, der eine Hypothese über die vergangenen Werte eine Uhr aufstellt. Diese Hypothese kann überprüft werden. Ist diese als wahr anzunehmen, so ist der Wert einer Uhr, den ein Event trägt, theoretisch möglich gewesen und die Uhr ist demzufolge nicht manipuliert oder abgedriftet.

Die Zeitstempel, die in Events dokumentiert werden, entsprechen nicht dem Zeitpunkt, an dem ein Event stattgefunden hat. Dies ist problematisch, da Zeitstempel meistens für die zeitliche Ordnung von Ereignissen innerhalb der Eventrekonstruktion verwendet werden [22]. So ist es möglich, dass ein Ereignis eingetreten ist, in dem sich ein Tatverdächtiger an einem System angemeldet hat, das Ereignis der Anmeldung aber erst dokumentiert wird, wenn der Tatverdächtige keinen Zugriff mehr auf das System hatte. Ihm kann die Anmeldung mithilfe des Events nicht direkt nachgewiesen werden. Um diesem Umstand zu begegnen, zeigten James und Jang [22] einen Ansatz, in dem sie die Zeitdauer bis ein Zeitstempel eines Ereignisses aktualisiert wird, experimentell bestimmten. Die Dauer der Aktualisierung bildeten sie als Normalverteilung ab und konnten so eine Zeitspanne ermitteln, in der ein Ereignis stattgefunden haben könnte.

2.4 Pfadformate in Windows 1x Systemen

Die Betriebssysteme Windows 10 und 11 unterstützt die Verwendung verschiedener Pfadformate für den Zugriff auf Daten. Der Benutzer eines Windows Systems kann zwischen drei verschiedenen Pfadformaten auswählen, um seine Speicherressourcen an-

zusprechen. In diesem Kapitel wird auf diese drei Pfadformate eingegangen. Anschließend wird die Bedeutung des NT Objekt Managers bei der Zuordnung der Pfadformate zu einem Speichermedium erläutert.

2.4.1 Pfadformate für Betriebssystemanwender

Dem Benutzer eines Windows 1x Betriebssystems stehen der DOS-, UNC- und DOS-Gerät-Pfade zur Addressierung seiner Speicherressourcen zur Verfügung. Erster und letzterer dienen dabei zur Adressierung von lokalen Speicherressourcen wie Festplatten, Partitionen oder Volumes. Der UNC-Pfad wird für die Bezeichnung von Netzwerkressourcen verwendet.

DOS-Pfad

Das DOS-Pfadformat ist das Format, das einem Windows 1x Benutzer am häufigsten begegnet. In seiner absoluten Schreibweise besteht es aus einem Laufwerks- oder Volumebuchstaben, Verzeichnisnamen und Dateinamen. Der Laufwerksbuchstabe wird von dem Verzeichnis oder Dateinamen mit einem Doppelpunkt („:“), gefolgt von einem Verzeichnistrennzeichen getrennt. Die Verzeichnisse untereinander oder ein Verzeichnis gegenüber einer Datei werden ebenfalls mit einem Verzeichnistrennzeichen getrennt. In Windows 1x ist sowohl der umgekehrte Schrägstrich (engl. back slash „\“) als auch der Schrägstrich (engl. slash „/“) als Verzeichnistrennzeichen für das DOS-Pfadformat zugelassen. Die Angabe eines relativen Pfades mittels „.“ für das aktuelle Verzeichnis oder „..“ für das übergeordnete Verzeichnis wird durch das DOS-Pfadformat unterstützt. Das Listing 2.3 zeigt eine Pfadangabe im DOS-Pfadformat. [48]

```
C:\Verzeichnis1\Datei1.extension
```

Listing 2.3: Beispiel für einen absoluten DOS-Pfad

UNC-Pfad

Für die Adressierung einer Netzwerkressource wird das UNC-Pfadformat verwendet. Die Abkürzung „UNC“ steht für Universal Naming Convention. Die Angabe des UNC-Pfades erfolgt immer absolut, da eine relative Beschreibung von Netzwerkressourcen nicht eindeutig möglich ist. Wie in Listing 2.5 zu sehen, werden vor die Beschreibung der Netzwerkressource zwei umgekehrte Schrägstriche (engl. back slash„\\“) gestellt. Die Netzwerkressource selbst wird über den Servernamen und den Freigabenamen des Shares beschrieben. Der Servername wird entweder über einen NetBIOS-Computernamen, eine IP-Addresse (IPv4 oder IPv6) oder eine Fully Qualified Domain Name (FQDN) angegeben. Der Freigabename wird durch einen einzelnen umgekehrten Schrägstrich vom Servernamen getrennt. Auf den Freigabenamen folgen die Verzeichnisnamen und

der Dateiname. Diese werden wie bei den DOS-Gerätepfaden durch das Verzeichnistrennzeichen getrennt. Bei UNC-Pfaden ist allerdings nur der umgekehrte Schrägstrich als Verzeichnistrennzeichen zugelassen. [48]

```
\\server.de\Share1\Verzeichnis1\Datei1.extension
```

Listing 2.4: Beispiel für einen UNC-Pfad

DOS-Geräte-Pfade

Das einheitliche Objektmodell von Windows ermöglicht den Zugriff auf alle Ressourcen. Die Objektpfade werden über einen speziellen Ordner im NT Objekt Manager durch die Win32-Schicht dem Anwender zu Verfügung gestellt. Auf den Ordner im NT Objekt Manager und somit auf die Ressourcen kann über das DOS-Geräte-Pfad-Format zugegriffen werden. Am Anfang jedes DOS-Geräte-Pfades steht einer der beiden Gerätebezeichner „\\.\“ oder „\\?\“. Diese kennzeichnen den Pfad als DOS-Geräte Pfad. Nach ihnen folgt eine Bezeichnung der Ressource, auf die zugegriffen werden soll. Im Falle von Speicherressourcen wären dies Festplatten, Partitionen oder Volumes. Der Zugriff auf die Daten auf einer Partition kann zum Beispiel über die Volume-GUID oder ein Laufwerkbuchstabe erfolgen. Das Listing 2.5) zeigt vier Möglichkeiten, auf eine Datei innerhalb einer Partition zuzugreifen. [48]

Im Abschnitt 2.4.2 wird genauer auf das Verzeichnis im NT Objekt Manager eingegangen, dass den Zugriff auf die Ressourcen ermöglicht.

```
\\.\C:\Verzeichnis1\Datei1.extension
\\.\HarddiskVolume1\Verzeichnis\Datei1.extension
\\?\BootPartition\Verzeichnis1\Datei1.extension
\\?\Volume{e35cf23b-8ad6-4c65-894b-b5b6c41e6b6a}\Verzeichnis\
Datei1.extension
```

Listing 2.5: Beispiele für DOS-Geräte-Pfade

2.4.2 NT Objekt Manager

Der NT Kernel von Windows ist für die Verwaltung von logischen oder physischen Ressourcen, wie Dateien, Prozessen oder Festplatten verantwortlich. Um diese Aufgabe erfüllen zu können, implementiert er einen Mechanismus, der es den Anwendungen des Systems ermöglicht, Ressourcen zu adressieren, um mit diesen interagieren zu können, diese mit anderen Anwendungen zu teilen oder diese zu schützen. Die APIs, die den Zugriff auf die Ressourcen ermöglicht, wie der Memory Manager, Process Manager oder I/O Manager, basieren auf dem NT Objekt Manager. Seine Aufgabe ist es, Objektcontainer für die verwalteten Ressourcen breit zustellen, die von den anderen NT Subsystemen mit Inhalt gefüllt werden. Konkret ist der NT Objekt Manager für das Erstellen von neuen Objekttypen, das Erstellen und Löschen von Objekten, sowie für

das Setzen und Abfragen von Attributen der Objekte verantwortlich. Sind die Objekte benannt, existieren sie im sogenannten NT Objekt Manager Namensraum. Dieser verwaltet die benannten Objekte in einem hierarchischen Verzeichnissystem. So werden unter `\Device` benannte Geräte-Objekte (engl. Device Objects) gespeichert, die vom I/O Manager erstellt wurden. Das Unterverzeichnis `\GLOBAL??` (auch als `\??` bezeichnet) enthält Objekte, die durch die Win32 API verwendbar sind. In diesem Verzeichnis können auch die Bezeichnungen der DOS-Geräte-Pfade gefunden werden (siehe Abschnitt 2.4.1). [66]

In den nachfolgenden Abschnitten wird auf die NT Objekt Manager Verzeichnisse `\Device`, `\GLOBAL??` und `\ArcName` detaillierter eingegangen.

\Device Verzeichnis

Das `\Device` Verzeichnis im NT Objekt Manager Namensraum enthält benannte Geräte-Objekte (engl. Device Objects), die von Treibern erstellt wurden [53]. Der Name dieser Geräte-Objekte wird als NT Geräte Name (engl. NT Device Name) bezeichnet. Erstellt ein Treiber, der mit dem Windows Driver Model implementiert wurde, ein physisches Geräte-Objekt (engl. Physical Device Object, kurz PDO), so ist dieses immer benannt. Der Name des physischen Geräte-Objektes wird anhand der Gerätecharakteristik durch das System generiert. Folgt ein Treiber nicht dem Windows Driver Model, muss dieser mindestens ein benanntes Geräte-Objekt im `\Device` Verzeichnis erstellen. [52][59][60]

\GLOBAL?? und \Sessions\0\DosDevices\[Logon Id]

Die Verzeichnisse `\GLOBAL??` und `\Sessions\0\DosDevices\[Logon Id]` enthalten globale beziehungsweise lokale Objekte, die durch die Win32-API verfügbar sind. Die in diesen beiden Verzeichnis gespeicherten Objekte sind symbolische Verknüpfungen, die auf Objekte in anderen Verzeichnissen verweisen. Bei den globalen Objekten handelt es sich um Objekte, die im gesamten System zur Verfügung stehen. Lokale Objekte hingegen stehen nur innerhalb einer Anmeldesitzung eines Benutzers zur Verfügung und werden deshalb unter der Anmelde-Id (Logon-Id) im NT Objekt Manager abgelegt. Ein Beispiel für ein lokales Objekt ist ein Netzlaufwerk, das durch einen Benutzer gemountet worden ist und für die anderen Benutzer nicht zur Verfügung steht. [50][60]

Ein Zugriff auf die Datei `C:\Verzeichnis\Datei.ext` über die Win32-API läuft wie folgt ab. Zunächst wird der DOS-Pfad in einen DOS-Geräte-Pfad umgewandelt `\\.\C:\Verzeichnis\Datei.ext`. Anschließend wird der Pfadprefix durch den I/O Manger in `\??` beziehungsweise `\GLOBAL??` geändert und der Name-Parsing Routine des NT Device Manager übergeben. Diese lokalisiert `C:` als symbolische Verknüpfung im Verzeichnis `\GLOBAL??`, die über weitere symbolische Verknüpfungen in anderen Verzeichnis-

sen auf das Geräte-Objekt `\Device\HarddiskVolume2` zeigt. Anschließend wird das gefundene Geräte-Objekt zusammen mit dem restlichen Pfad an die Device-Objekt-Parse Procedur des I/O Manager zurückgegeben. Diese ist dann für das Auffinden des Dateisystems zuständig. [66][60]

\ArcName

Die Abkürzung ARC steht für Advanced RISC Computing. Advanced RSIC Computing definiert, neben einer standardisierten MISP RISC-basierten Umgebung für Computerhardware und Firmware, eine Namenskonvention für Geräte. In früheren Windows Versionen wurde diese Namenskonvention unter anderem in der BOOT.ini für die Angabe der Bootpartition verwendet. Bei Windows x86/x64 Computern sind zwei ArcName-Formate möglich: mutli(a)disk(b)rdisk(c)partition(d) oder scsi(a)disk(b)rdisk(c)partition-(d). Die multi(a) Variante wird bei IDE, ESDI und SCSI Laufwerken verwendet. [44][60]

Im folgenden wird die Bedeutung der Werte a, b, c und d bei der multi Syntax erläutert:

- multi(a) besitzt immer den Wert 0
- disk(b) besitzt immer den Wert 0, da die multi(0) Syntax in der „BOOT.ini“ das System anweist, das BIOS zum Laden der Systemdateien zu verwenden. Der vom BIOS verwendete Interrupt (INT) 13 benötigt die Informationen von disk(b) nicht.
- rdisk(c) gibt die Nummer der Festplatte am Adapter an
- partition(d) gibt die Nummer der Partition an. Die erste valide Partition erhält die Nummer 1.

In der *scsi* Syntax haben die Werte die folgende Bedeutung:

- scis(a) beschreibt die Nummer des Adapters
- disk(b) beschreibt die SCSI ID der Festplatte
- rdisk(c) beschreibt die SCSI Logical Unit Number (LUN) (meistens 0)
- partition(d) beschreibt die Nummer der Partition. Die erste vailide Partition erhält die Nummer 1

Bei den Objekten in `\ArcName` handelt es sich um symbolische Verknüpfungen. Diese werden mit der Funktion IoAssignArcName aus der ntddk.h von den Treibern der Speichermedien erstellt [51].

3 Bestimmen einer Zeitstempelverteilung der Ausführungsartefakte

In diesem Abschnitt der Arbeit soll die Fragestellung beantwortet werden, ob die Zeitstempel der Ausführungsartefakte über eine statistische Verteilung abgebildet werden können. Ziel ist es, einen Schwellwert zu bestimmen, mit dessen Hilfe die Artefakte zeitlich gruppiert werden können.

3.1 Methode

Die Bestimmung der Zeitstempelverteilung der Ausführungsartefakte erfolgt empirisch. Dafür werden auf zwei Windows 1x Systemen, eins mit Windows 10 und das andere mit Windows 11, ausgewählte Anwendungen gestartet und ihr tatsächlicher Startzeitpunkt dokumentiert. Anschließend erfolgt eine logische Sicherung von forensischen Artefakten in festen Zeitintervallen, aus denen dann die jeweiligen Events extrahiert werden. Mithilfe der Zeitstempel in den Events wird die Abweichung von dem tatsächlichen Startzeitpunkt berechnet. In den folgenden Abschnitt wird diese Methodik detaillierter beschrieben.

3.1.1 Die Windows 1x Testsysteme

Das Windows 10 und 11 System wurden beide mithilfe des Typ 2 Hypervisors VirtualBox [62] virtualisiert. Als Virtualisierungshost kam ein Laptop mit einem Intel Core i7-8665U Prozessor und 16 GB Arbeitsspeicher zum Einsatz. Die Gastsysteme erhielten jeweils 2 Prozessorkerne und 2048 MB Arbeitsspeicher sowie eine virtuelle 50 GB Festplatte. Als Gastsysteme kamen Windows 10 Pro (Build 19043) und Windows 11 Pro (Build 22000) zum Einsatz. Bei der Installation des Betriebssystems wurden die in Tabelle 3.1 zusehenden Einstellungen getätigt.

3.1.2 Maßnahmen der Forensic Readiness

In den Windows 1x Systemen werden in der Standardkonfiguration nicht alle Artefakte erzeugt, die für eine forensische Untersuchung nützlich sind. Aus diesem Grund können, im Vorfeld einer forensischen Untersuchung, Maßnahmen der Forensic Readiness umgesetzt werden. Dieser Begriff beschreibt alle technischen und organisatorischen Maßnahmen, die helfen, eine Unternehmens- oder Organisationsinfrastruktur auf eine forensische Untersuchung vorzubereiten. [65] [73]

Tabelle 3.1: Die Tabelle zeigt alle Einstellungen, die während der Installation von Windows 1x getätigt wurden. Es wurde darauf geachtet, dass Features deaktiviert oder soweit eingeschränkt wurden, dass so wenig wie möglich Daten an Microsoft übertragen werden.

Privatsphäre Einstellungen	**Auswahlmöglichkeit**
Geräte übergreifender Aktivitätsverlauf	nein
Hilfe vom digitalen Assistenten	abgelehnt
Online Spracherkennung	nicht verwenden
Standort verwenden	nein
Gerät suchen	nein
Diagnosedaten senden	einfach
Mithilfe von Diagnosedaten angepasste Erfahrungen erhalten	nein
Apps Werbe-ID verwenden	nein

In dieser Arbeit musste im Zuge der Forensic Readiness die Aufzeichnung von Prozesserstellungsereignissen im EVTX-Log `Security.evtx` (siehe Abschnitt 2.2.4) eingerichtet werden. Um diese zu aktivieren, wurde im Gruppenrichtlinien-Editor `gpedit.msc` unter *Computerkonfiguration* → *Windows-Einstellungen* → *Sicherheitseinstellungen* → *Erweiterte Überwachungsrichtlinienkonfiguration* → *Systemüberwachungsrichtlinien - Lokales Gruppenrichtlinienobjekt* → *Detaillierte Überwachung* die Richtlinie *Prozesserstellung überwachen* für den Erfolgs- und Fehlerfall aktiviert. Die aktivierte Richtlinie produziert bei der Erstellung von Prozessen einen Ereigniseintrag mit der Event ID 4688 in der `Security.evtx`.

Um Einträge in der Windows Timeline beobachten zu können, musste in den *Einstellungen* unter *Datenschutz -> Aktivitätsverlauf* die Punkte *Mein Aktivitäsverlauf auf diesem Gerät speichern* und *Meinen Aktivitätsverlauf an Microsoft senden* aktiviert werden. Für die anderen Artefakte mussten keine Maßnahmen getroffen werden, sie kommen in der Windows 1x Standardkonfiguration vor.

3.1.3 Die gestarteten Anwendungen

Bei den Anwendungen, die in dieser Arbeit gestartet wurden, wird zwischen Kommandozeilen (engl. *Comandline Interface* kurz *CLI*) Anwendungen und Anwendungen mit einer grafischen Benutzeroberfläche (engl. *Graphical User Interface* kurz *GUI*) unterschieden. Im Folgenden werden die verwendeten Anwendungen beider Gruppen vorgestellt.

3.1.4 CLI - Anwendungen

ping.exe

Die CLI-Anwendung `ping.exe` von Microsoft ist in Windows 10 bereits vorinstalliert. Mit ihr ist es möglich, ICMP-Echo-Request Pakete an einen anderen Host zu senden und das ICMP-Echo-Reply Paket zu empfangen. Unter Windows 1x ist die Anwendung unter `C:\Windows\System32\PING.EXE` gespeichert. [56]

PECmd.exe

PECmd.exe ist eine CLI-Anwendung, welche die in einer Prefetch-Datei enthaltenen Informationen extrahiert. Entwickelt wurde sie von Eric Zimmerman. [83] Auf den beiden Windows Systemen ist sie unter `C:\ExperimentPrograms\pecmd\PECmd.exe` installiert worden.

python.exe

Python ist eine Programmiersprache, die in den 1990er Jahren von Guido van Rossum entwickelt wurde. Sie ist für die Betriebssysteme Windows, Mac OS X und Linux verfügbar. Python wurde auf den beiden Systemen unter `C:\ExperimentPrograms\python\python.exe` installiert.

3.1.5 GUI - Anwendungen

Mozilla Firefox

Firefox ist ein freier Webbrowser, der von der Mozilla Foundation entwickelt und unter der GPL-Lizenz zur Verfügung gestellt wird. Die Extended Support Release (kurz ESR) ist aufgrund seiner längeren Update-Unterstützung für den Einsatz in Unternehmen vorgesehen. [78] Die Installation von Mozilla Firefox erfolgt mittels MSI Installer in das Verzeichnis `C:\ExperimentPrograms\firefox_esr\firefox.exe`.

GIMP

GNU Image Mainipulation Program (kurz *GIMP*) ist ein Open Source Programm für die Bearbeitung von Pixelgrafiken. [75] GIMP wurde für alle Benutzer unter `C:\ExperimentPrograms\gimp\bin\gimp-2.10.exe` installiert.

KeePassXC

KeePassXC ist ein Passwort-Manager, der Passwörter in einer Datenbank im KDBX Format verwaltet. [26] In den virtuellen Maschinen wurde KeePassXC als Portable Version unter `C:\ExperimentPrograms\KeePassXC-2.6.6-Win64\KeePassXC.exe` installiert.

RegistryExplorer

Der RegistryExplorer ist ein von Eric Zimmerman entwickeltes Werkzeug für die forensische Untersuchung von Windows Registry Hives. Bei der Executable `RegistryExplorer.exe` handelt es sich um ein Programm mit grafischer Benutzeroberfläche, in der die Registry Schlüssel analysiert werden können. [83] In den Windows 1x Systemen wurde der RegistryExplorer im folgenden Pfad installiert: `C:\ExperimentPrograms\RegistryExplorer\RegistryExplorer.exe`.

3.1.6 Notepad++

Notepad++ ist ein OpenSource Texteditor für Windows. Entwickelt wurde er von Don Ho in der Programmiersprache C++. [20] Auf den beiden Testsystemen wurde Notepad++ mithilfe des MSI-Installers in das Verzeichnis `C:\ExperimentPrograms\notepad++\` installiert.

3.1.7 Betrachtete Artefakte

In dieser Arbeit werden die folgenden forensischen Artefakte betrachtet, die bei der Ausführung einer Anwendung entstehen:

- PowerShell Objekt `System.Diagnostics.Process`
- Prefetch-Dateien
- Windows Timeline (ActivitiesCache.db)
- Windows XML Event Log (Event Id 4688)
- UserAssist Registryschlüssel
- Background Activity Moderator

Das PowerShell Objekt `System.Diagnostics.Process` wird beim Start der Anwendungen durch das, im folgenden Abschnitt beschriebene, PowerShell Skript erzeugt. Der in ihm enthaltene Zeitstempel *StartTime* wird in dieser Arbeit als tatsächlicher Startzeitpunkt gehandelt.

3.1.8 Generierung und Sicherung der forensischen Artefakte

Für die Ausführung der Anwendungen und die Sicherung der forensischen Artefakte wurde ein PowerShell-Skript entwickelt. Es trägt den Namen `Start-Experiment`. Dieses führt im ersten Schritt Anwendungen in einem festen zeitlichen Abstand aus, die in einer Konfigurationsdatei definiert worden sind. Im zweiten Schritt registriert es in der Windows Aufgabenplanung sieben Aufgaben. Diese sorgen für die Sicherung der forensischen Artefakte 5, 15, 30, 40, 55 und 70 Minuten, nach dem die letzte Anwendung gestartet wurde. Nach der Sicherung bei Minute 30 wird das System, ebenfalls über eine geplante Aufgabe, neugestartet, um etwaige Veränderungen in den Artefakten nach einem Neustart beobachten zu können.

Die Sicherung der Artefakte sollte ebenfalls durch ein PowerShell-Skript durchgeführt werden. Da einige Dateien, die für eine forensische Untersuchung wertvoll sind, zur Laufzeit des Betriebssystems durch dieses gesperrt werden, können sie nicht mit herkömmlichen Mitteln kopiert werden. Um diese Einschränkung zu umgehen, wurde der Kroll Artifact Parser and Extractor (kurz KAPE) verwendet. KAPE ermöglicht eine logische Sicherung von Dateien zur Laufzeit eines Betriebssystems. Die Dateien werden anhand von Zielbeschreibung (engl. Target Descriptions), die im YAML-Format definiert werden, gesichert. Bei der Sicherung erkennt KAPE, ob eine Datei durch das Betriebssystem gesperrt ist und sichert diese mithilfe von Zugriffen auf die Rohdaten der Festplatte. [84] Das Listing 3.1 zeigt die in dieser Arbeit verwendete Zielbeschreibung.

```
1
2  Description: TimestampDistribution
3  Author: Dominic Helfer
4  Version: 1.0
5  Id: bbab7d8d-8d8e-4a37-9b89-96adfc13ceb8
6  RecreateDirectories: True
7  Targets:
8    -
9      Name: Prefetch
10     Category: Prefetch
11     Path: Prefetch.tkape
12   -
13     Name: WindowsTimeline
14     Category: FileFolderAccess
15     Path: WindowsTimeline.tkape
16   -
17     Name: RegistryHivesSystem
18     Category: Registry
19     Path: RegistryHivesSystem.tkape
20   -
21     Name: RegistryHivesUser
22     Category: Registry
23     Path: RegistryHivesUser.tkape
24   -
25     Name: EventLogs
```

```
      Category: EventLogs
      Path: EventLogs.tkape
  # Documentation
  # N/A
```

Listing 3.1: KAPE Target Description, die in dieser Arbeit zur Sicherung der forensischen Artefakte verwendet wurde. Die Beschreibung muss im KAPE Verzeichnis unter `Targets\!Local\TimestampDistribution.tkape` gespeichert werden.

Zum Starten des Experiments wird der in Listing 3.2 zu sehende Aufruf des Skriptes `Start-Experiment` verwendet. Der erste Parameter `ExecutableDescription` gibt den Pfad zur Datei an, die die Beschreibung der auszuführenden Anwendungen enthält. Das Listing 3.3 zeigt die in dieser Arbeit verwendete ExecutableDiscription. Jede Anwendung, die ausgeführt werden soll, steht zusammen mit ihren Parametern auf einer Zeile. `Path` beschreibt den Pfad, in dem die Startzeiten der Anwendungen sowie die gesicherten Artefakte abgelegt werden sollen. Die Zeitspanne zwischen dem Starten und Beenden einer Anwendung wird in Sekunden über den Parameter `Delay` angegeben. Der Pfad zur ausführbaren Datei von KAPE muss an `KapePath` übergeben werden, sodass die Aufgaben in der Aufgabenplanung erstellt werden können. Die Definition der zu verwendenden Zielbeschreibung erfolgt mittels `KapeTarget`. Die angegebene Beschreibung muss im Installationspfad von KAPE unter `Targets\` verfügbar sein. Der Schalter `Verbose` gibt detailliertere Informationen über die aktuellen Schritte des Experimentes auf der Kommandozeile aus.

```
Start-Experiment -ExecutableDescription .\ExecutableDescription -
  Path .\Output -Delay 30 -KapePath .\Kape\Kape.exe -KapeTarget
  TimestampDistribution -Verbose
```

Listing 3.2: Aufruf zum Starten des Experiments

```
C:\ExperimentPrograms\firefox_esr\firefox.exe https://consecur.de
C:\ExperimentPrograms\gimp\bin\gimp-2.10.exe
C:\ExperimentPrograms\KeePassXC-2.6.6-Win64\KeePassXC.exe
C:\ExperimentPrograms\notepadpp\notepad++.exe
C:\ExperimentPrograms\pecmd\PECmd.exe
C:\ExperimentPrograms\python\python.exe
C:\ExperimentPrograms\RegistryExplorer\RegistryExplorer.exe
C:\Windows\System32\PING.EXE -n 100 consecur.de
```

Listing 3.3: Konfigurationsdatei der auszuführenden Anwendungen

3.1.9 Parsen der Artefakte

Die Extraktion der Events aus den forensischen Artefakten erfolgte mithilfe des Python-basierten Werkzeuges Plaso. Dieses ermöglicht es, anhand von einzelnen Artefakte-Dateien oder vollständigen Festplattenabbildern, eine Zeitreihe von Events zu erstellen und diese in die unterschiedlichsten Formate zu exportieren. Plaso verwendet einen

zweistufigen Prozess zur Erstellung dieser Zeitreihe. Im ersten Schritt werden die Informationen aus den Artefakten extrahiert und als Events aufbereitet. Das Ergebnis wird in einer SQLite Datenbank, der sogenannten Plaso Storage File, gespeichert. Im zweiten Schritt werden die Events aus der Datenbank in ein gewünschtes Format extrahiert. Der erste Schritt wird durch das Kommandozeilenwerkzeug `log2timeline.py` und der zweite durch `psort.py` vorgenommen. Für die schnellere Erzeugung einer Zeitreihe steht das Werkzeug `psteal.py` zur Verfügung, das beide Schritte in einem Werkzeug vereint. Allerdings stehen bei dieser Variante dem Forensiker nicht alle Optionen der beiden einzelnen Werkzeuge zur Verfügung. [29][30]

In dieser Arbeit wurde `psteal.py` verwendet. Um nicht alle Events aus den gesicherten Artefakten zu extrahieren, wurden nur die Parser `prefetch, sqlite/windows_timeline, winevtx, winreg/userassist` und `winreg/bam` angewendet. Die so extrahierten Events sind anschließend in das JSON Format exportiert worden. Das Listing 3.4 zeigt die verwendeten Parameter von `psteal.py`.

```
1 psteal.py --parsers prefetch,sqlite/windows_timeline,winevtx,
   winreg/bam,winreg/userassist --storage_file storage_file.plaso
   --source ./artifacts/C/ -o json -w output.json
```

Listing 3.4: Aufruf des forensischen Werkzeuges Plaso

3.1.10 Auswertung der Zeitstempel

Für jedes Experiment steht eine CSV-Datei mit den tatsächlichen Ausführungszeitpunkten und sechs JSON-Dateien mit den, durch Plaso extrahierten, Events zur Verfügung. Im ersten Schritt der Auswertung wurden die relevanten Events aus der JSON-Datei mithilfe der Knowledge Base in Tabelle 3.2 extrahiert. Anschließend ist für jede Anwendung die Differenz zwischen dem Zeitstempel des jeweiligen Artefakts und dem tatsächlichen Startzeitpunkt berechnet worden. Im zweiten Schritt erfolgte eine Auswertung der Differenzen und Zeitstempel hinsichtlich Vorhandenseins und zeitlicher Veränderung im Laufe des Experimentes. Die Sicherung, die am meisten Zeitstempel enthielt, wurde ausgewählt und floss in die Berechnung der Zeitstempelverteilung ein.

Tabelle 3.2: Die Tabelle zeigt die Knowledge Base, die zum Filtern der Events verwendet wurde.

Artefakt	Eintrag
Ping:	
Win10 Prefetch Hash	7E94E73E
Win11 Prefetch Hash	4A8A6853
EVTX	C:\Windows\System32\PING.EXE
Win10 BAM	\Device\HarddiskVolume2\Windows\System32\-PING.EXE
	Fortsetzung folgt auf der nächsten Seite

Tabelle 3.2: Fortsetzung der Tabelle

Artefakt	Eintrag
Win11 BAM	\Device\HarddiskVolume3Windows\System32\-PING.EXE
UserAssist	{System32}\PING.EXE
Windows Timeline	{1ac14e77-02e7-4e5d-b744-2eb1ae5198b7}\-ping.exe
PECmd:	
Win10 Prefetch Hash	088A068D
Win11 Prefetch Hash	DB1ADE9A
EVTX	C:\ExperimentPrograms\pecmd\PECmd.exe
Win10 BAM	\Device\HarddiskVolume2\-ExperimentPrograms\pecmd\PECmd.exe
Win11 BAM	\Device\HarddiskVolume3\-ExperimentPrograms\pecmd\PECmd.exe
UserAssist	C:\ExperimentPrograms\pecmd\PECmd.exe
Windows Timeline	C:\ExperimentPrograms\pecmd\PECmd.exe
Python:	
Win10 Prefetch Hash	CA8634E7
Win11 Prefetch Hash	A6955124
EVTX	C:\ExperimentPrograms\python\python.exe
Win10 BAM	\Device\HarddiskVolume2\-ExperimentPrograms\python\python.exe
Win11 BAM	\Device\HarddiskVolume3\-ExperimentPrograms\python\python.exe
UserAssist	C:\ExperimentPrograms\python\python.exe
Windows Timeline	C:\ExperimentPrograms\python\python.exe
Firefox ESR:	
Win10 Prefetch Hash	7D50594E
Win11 Prefetch Hash	20CA901B
EVTX	C:\ExperimentPrograms\firefox\firefox.exe
Win10 BAM	\Device\HarddiskVolume2\-ExperimentPrograms\firefox\firefox.exe
Win11 BAM	\Device\HarddiskVolume3\-ExperimentPrograms\firefox\firefox.exe
UserAssist	39891333BFD11670
Windows Timeline	39891333BFD11670
GIMP:	
Win10 Prefetch Hash	7D58B384
Win11 Prefetch Hash	3D535E19
EVTX	C:\ExperimentPrograms\gimp\gimp-2.10.exe

Fortsetzung folgt auf der nächsten Seite

Tabelle 3.2: Fortsetzung der Tabelle

Artefakt	Eintrag
Win10 BAM	`\Device\HarddiskVolume2\-ExperimentPrograms\gimp\bin\gimp-2.10.exe`
Win11 BAM	`\Device\HarddiskVolume3\-ExperimentPrograms\gimp\bin\gimp-2.10.exe`
UserAssist	`gimp.GimpApplication`
Windows Timeline	`gimp.GimpApplication`
KeePassXC:	
Win10 Prefetch Hash	`6EA7F5AD`
Win11 Prefetch Hash	`A6B7209A`
EVTX	`C:\ExperimentPrograms\-KeePassXC-2.6.6-Win64\KeePassXC.exe`
Win10 BAM	`\Device\HarddiskVolume2\-ExperimentPrograms\KeePassXC-2.6.6-Win64\-KeePassXC.exe`
Win11 BAM	`\Device\HarddiskVolume3\-ExperimentPrograms\KeePassXC-2.6.6-Win64\-KeePassXC.exe`
UserAssist	`C:\ExperimentPrograms\-KeePassXC-2.6.6-Win64\KeePassXC.exe`
Windows Timeline	`C:\ExperimentPrograms\-KeePassXC-2.6.6-Win64\KeePassXC.exe`
RegistryExplorer:	
Win10 Prefetch Hash	`A4ACB5B7`
Win11 Prefetch Hash	`1E5CD6D4`
EVTX	`C:\ExperimentPrograms\RegistryExplorer\-RegistryExplorer.exe`
Win10 BAM	`\Device\HarddiskVolume2\-ExperimentPrograms\RegistryExplorer\-RegistryExplorer.exe`
Win11 BAM	`\Device\HarddiskVolume3\-ExperimentPrograms\RegistryExplorer\-RegistryExplorer.exe`
UserAssist	`C:\ExperimentPrograms\RegistryExplorer\-RegistryExplorer.exe`
Windows Timeline	`C:\ExperimentPrograms\RegistryExplorer\-RegistryExplorer.exe`

3.2 Ergebnisse

In diesem Abschnitt der Arbeit werden die Ergebnisse der Experimente, die für die Bestimmung der Zeitstempelverteilungen durchgeführt wurden, vorgestellt. Dabei wird zunächst auf die vorhandenen Zeitstempel eingegangen. Anschließend wird die Abweichung des Zeitstempels vom tatsächlichen Startzeitpunkt betrachtet. Insgesamt wurden für das Windows 10 sowie das Windows 11 System 20 Experimente durchgeführt. Die Laufzeit der Anwendungen betrug dabei in den ersten zehn Experimenten 30 Sekunden und in den letzten zehn Experimenten 60 Sekunden.

Für jedes System werden pro Experiment im Idealfall 80 Zeitstempel erzeugt. Ein Satz besteht immer aus zehn einzelnen Experimenten mit gleicher Ausführungszeit der Anwendungen. Da die Tabelle 3.3 die beiden Experiment Sätze für jedes System zusammenfasst, werden im Idealfall 160 Zeitstempel dokumentiert. Im Anhang A.2 sind die Werte für die einzelnen Ausführungszeiten aufgegliedert dargestellt.

3.2.1 Prefetch

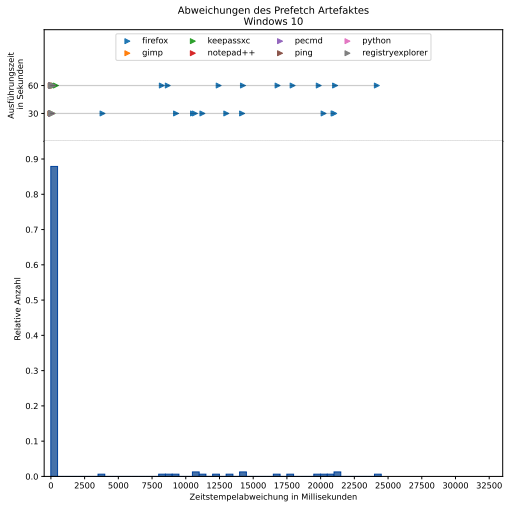
In den zwanzig durchgeführten Experimenten konnte unter Windows 10 dreimal und unter Windows 11 einmal für eine Anwendung kein Zeitstempel aufgefunden werden. Alle Beobachtungen von fehlenden Zeitstempeln traten in den Experimenten mit einer Ausführungszeit von 60 Sekunden auf. Unter Windows 10 fehlten zweimal die Zeitstempel für GIMP und einmal für den Firefox Browser. In Windows 11 trat der einzige Fall für Ping auf. Der Mittelwert sowie die Standardabweichung der Zeitstempelabweichung betrugen im Windows 11 System $2224,84 \pm 6244,06$ Millisekunden und im Windows 10 System $1798,45 \pm 5145,83$ Millisekunden. Die Quantile $0,25$, $0,50$ und $0,75$ wiesen in Windows 11 niedrigere Werte auf als in Windows 10. Der maximale Wert der Zeitstempelabweichung ist beim Windows 10 System mit $24167,29$ Millisekunden fast $8,64$ Sekunden geringer als im Windows 11 System. Wie in Abbildung 3.1 erkennbar, wurden in beiden Systemen die Zeitstempel, die mehr als $2,5$ Sekunden vom tatsächlichen Startzeitpunkt abweichen, vom Firefox Browser erzeugt. Führt man die Daten des Windows 10 und 11 Systems zusammen, so beträgt das $0,75$-Quantil $43,98$ und das $0,90$-Quantil $9990,08$ Millisekunden.

3.2.2 Windows XML Event Log - Event 4688

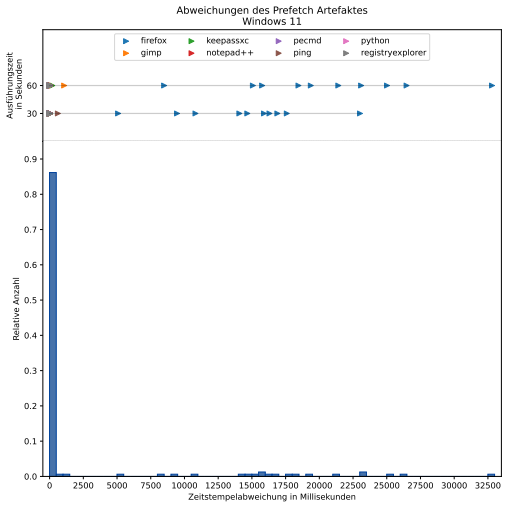
Das Event 4688 des Windows XML Event Log wurde in beiden Windows Systemen für alle Anwendungen in allen Experimenten vorgefunden. Beide Systeme haben mit $0,22 \pm 0,90$ Millisekunden beziehungsweise $0,30 \pm 1,07$ Millisekunden einen geringeren Mittelwert als die Prefetch Dateien. In Windows 10 hatten 75% der Zeitstempel eine Abweichung von $0,13$ Millisekunden oder weniger. In Windows 11 waren es $0,16$

Artefakt	Betriebssystem	$\bar{X} \pm s$			$\tilde{X}_{0,25}$	$\tilde{X}_{0,50}$	$\tilde{X}_{0,75}$	$\tilde{X}_{0,90}$	X_{min}	X_{max}	# Zeitstempel
Prefetch	Windows 10	1798,45	±	5145,83	6,33	19,34	59,03	8910,46	−6,39	24167,29	157
	Windows 11	2224,84	±	6244,06	5,89	12,72	31,47	11467,34	−5,39	32803,51	159
	kombiniert	2021,97	±	5719,73	5,89	15,67	42,98	9990,08	−6,39	32803,51	316
EVTX 4688	Windows 10	0,22	±	0,90	0,07	0,08	0,13	0,19	0,05	8,78	160
	Windows 11	0,30	±	1,07	0,10	0,12	0,16	0,30	0,05	11,71	160
	kombiniert	0,26	±	0,99	0,08	0,11	0,15	0,26	0,05	11,71	320
BAM	Windows 10	42965,10	±	20445,05	30064,37	39497,34	60220,89	65983,80	3367,06	92176.09	159
	Windows 11	40078,06	±	20166,94	30038,88	30648,46	60086,31	62997,09	187,15	72290,02	160
	kombiniert	41517,05	±	20325,56	30042,53	34112,60	60123,45	64986,43	187,15	92176,09	319
Windows Timeline	Windows 10	5874,23	±	9442,12	178,63	2434,31	8516,09	15394,47	−694,50	51898,02	120
	Windows 11	3905,21	±	5115,59	81,71	1987,69	6742,81	11079,67	−851,05	24877,72	133
	kombiniert	4839,14	±	7535,15	110,67	1999,45	7208,60	11997,74	−851,05	51898,02	253

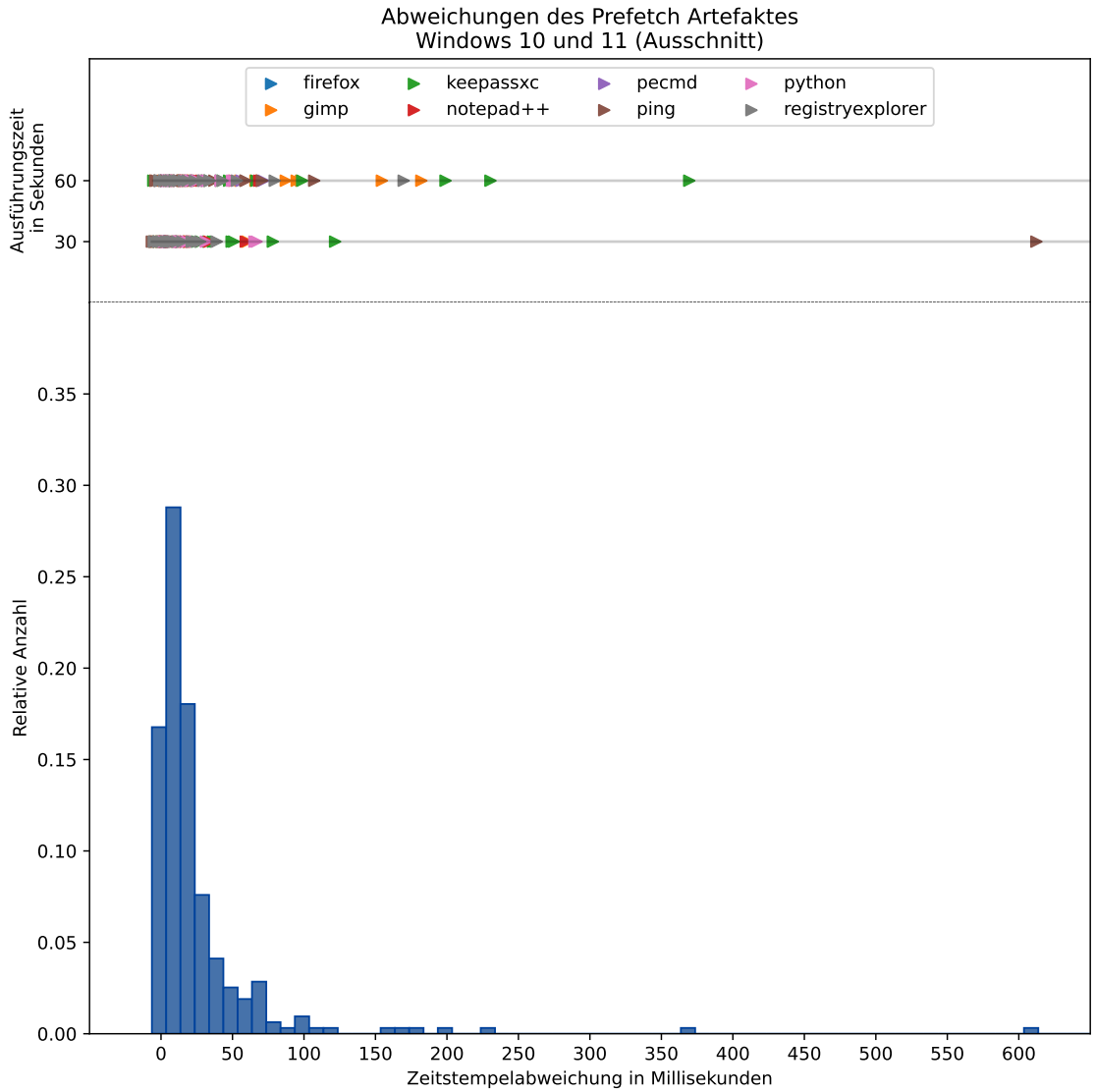
Tabelle 3.3: Die Tabelle zeigt statistische Werte der in den Experimenten gemessenen Zeitstempelverteilungen. Für die Berechnung der Werte wurden die Daten der Experimente mit einer Anwendungsausführungszeit von 30 und 60 Sekunden zusammengeführt. Die statistischen Werte, getrennt nach der Anwendungsausführungszeit, sind im Anhang A.2 abgebildet. Alle Werte wurden auf zwei Stellen nach dem Komma gerundet. Es werden keine Werte für das UserAssist Artefakt aufgeführt, da diese in den Experimenten nicht beobachtet werden konnten.



(a) Histogramm der Zeitstempelabweichungen des Windows 10 Systems mit einer Balkenbreite von 500 Millisekunden.



(b) Histogramm der Zeitstempelabweichungen des Windows 11 Systems mit einer Balkenbreite von 500 Millisekunden.



(c) Ausschnitt aus dem Histogramm der zusammengeführten Zeitstempelabweichungen von Windows 10 und 11 einer Balkenbreite von 10 Millisekunden.

Abbildung 3.1: Histogramme der Zeitstempelabweichungen für das Prefetch Artefakt.

Millisekunden oder weniger. In Windows 10 wiesen nur vier der 160 Zeitstempel eine Abweichung von mehr als einer Millisekunde auf. In Windows 11 waren es 5 von 160 Zeitstempeln. Diese größeren Abweichungen scheinen dabei nicht durch eine spezielle Anwendung verursacht wurden zu sein, wie in Abbildung 3.2a und 3.2b zu sehen ist.

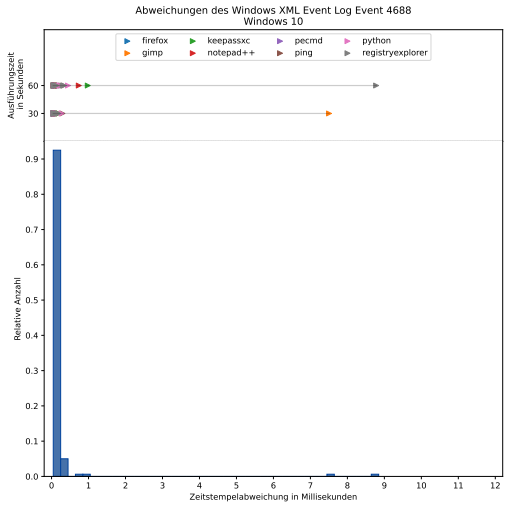
3.2.3 UserAssist

Sowohl in Windows 10 als auch in Windows 11 konnte für keine Anwendung im UserAssist Registryschlüssel ein Zeitstempel gefunden werden. Wurde für eine Anwendung ein Eintrag im UserAssist Registryschlüssel erstellt, so enthielt dieser keinen Zeitstempel, der eine Ausführung dokumentiert. Der `RunCount` betrug bei allen Experimenten 0. Der `FocusCount` hingegen wurde bei einigen Anwendungen aktualisiert.

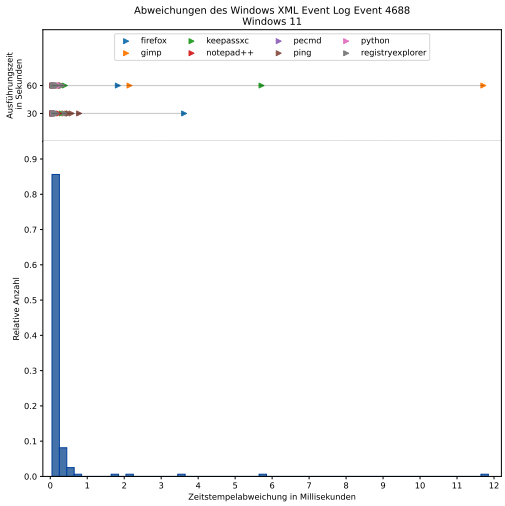
3.2.4 Background Activity Moderator

Die Zeitstempel des Background Activity Moderator sind im Windows 10 System in 17 von 20 Experimenten bereits ab der ersten Sicherung für alle Anwendungen vorhanden. In den anderen drei Experimenten sind sie erst ab der zweiten Sicherung vorhanden. Im Windows 11 System hingegen sind in nur 8 von 20 Experimente alle Zeitstempel in der ersten Sicherung vorhanden. In vier Experimenten bedurfte es einer zweiten Sicherung nach 15 Minuten und in zwei einer dritten Sicherung 30 Minuten, nach dem die letzte Anwendung gestartet wurde. In den verbleibenden sechs Experimenten waren die Zeitstempel erst in der vierten Sicherung, die nach einem Neustart des Systems durchgeführt wurde, vorhanden. Bis auf einen Zeitstempel in einem Experiment (der für den RegistryExplorer) konnten in beiden Systemen allen Anwendungen ein Zeitstempel zugeordnet werden. In beiden Systemen gibt es große Unterschiede in den Zeitstempelabweichungen zwischen den verschiedenen Ausführungszeiten. In fast allen Werten, außer den beiden Extremwerten (X_{min} und X_{max}), gibt es eine nahezu Verdopplung zwischen der Ausführungszeit von 30 und der von 60 Sekunden. So beträgt der Mittelwert, der in der Tabelle A.2 in Anhang A.2 abgebildet ist, bei einer Ausführungszeit von 30 Sekunden unter Windows 10 $29399,55 \pm 10954,69$ Millisekunden (Windows 11: $27205,26 \pm 8514,15$ Millisekunden). Bei einer Ausführungszeit von 60 Sekunden ist der Mittelwert mit $56702,37 \pm 18582,23$ Millisekunden (Windows 11: $52950,85 \pm 20259,69$ Millisekunden) $27,3$ Sekunden (Windows 11: $25,7$ Sekunden) höher. Alle statistischen Werte sind bei Windows 10 größer als bei Windows 11. Insbesondere der Extremwert X_{max} zeigt deutliche Unterschiede. Bei einer Ausführungszeit von 30 Sekunden unterscheidet sich dieser zwischen den beiden Systemen um $25,1$ Sekunden ($65021,02$ Millisekunden zu $39902,23$ Millisekunden), bei einer Ausführungszeit von 60 Sekunden um $19,9$ Sekunden ($92176,09$ Millisekunden zu $72290,02$ Millisekunden).

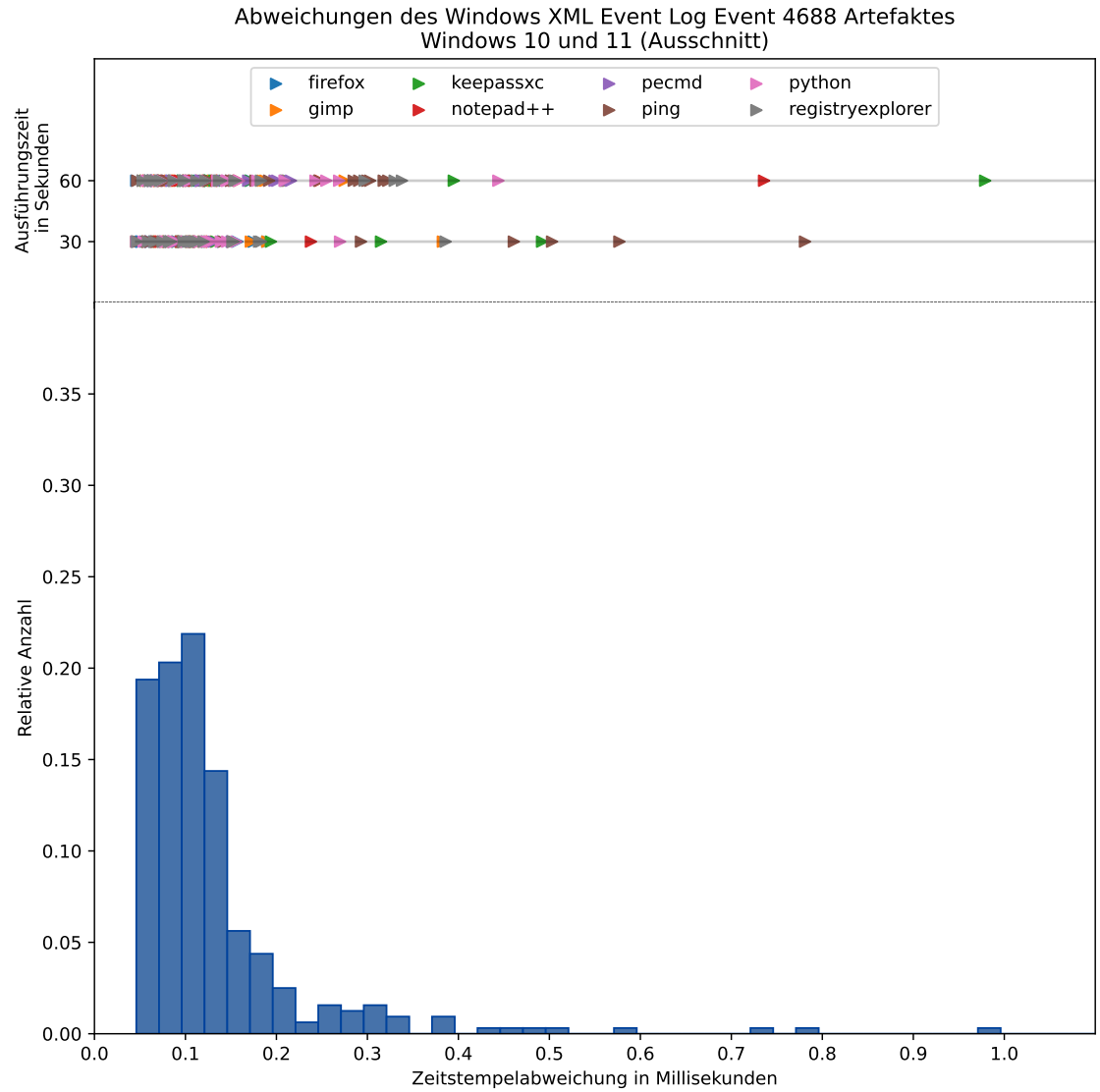
Wie in der Abbildung 3.3c deutlich zu erkennen, gibt es zwei Ausschläge jeweils im Bereich um 30 und 60 Sekunden. Die abgebildeten Zeitreihen oberhalb des Histogramms



(a) Histogramm der Zeitstempelabweichungen des Windows 10 Systems mit einer Balkenbreite von $0,2$ Millisekunden.



(b) Histogramm der Zeitstempelabweichungen des Windows 11 Systems mit einer Balkenbreite von $0,2$ Millisekunden.



(c) Ausschnitt aus dem Histogramm der zusammengeführten Zeitstempelabweichungen von Windows 10 und 11 mit einer Balkenbreite von $0,025$ Millisekunden.

Abbildung 3.2: Histogramme der Zeitstempelabweichungen für das Event 4688 des Windows XML Event Log Artefakt.

zeigen, dass die Ausschläge im Bereich um 30 Sekunden, durch die Anwendungen mit einer Ausführungszeit von 30 Sekunden und die im Bereich um 60 Sekunden, durch die mit einer Ausführungszeit von 60 Sekunden verursacht werden. Die Ausschläge im Bereich von 2500 bis 20000 Millisekunden werden ausschließlich durch die Anwendung PECmd erzeugt. Die beiden Abbildungen 3.3a und 3.3b zeigen, dass im Windows 11 System, die Zeitstempelabweichungen deutlich dichter sind, als im Windows 10 System.

3.2.5 Windows Timeline

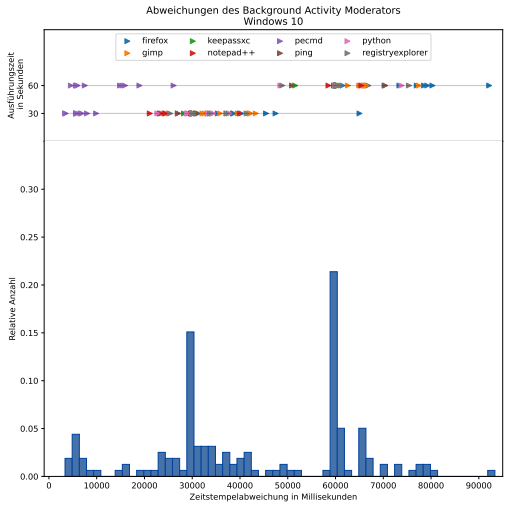
In Windows 10 und 11 fehlen in allen Experimenten die Zeitstempel für die Anwendung RegistryExplorer in der Windows Timeline. Für Notepad++ ist dies in Windows 10 in 19 von 20 Experimenten und in Windows 11 in 7 von 20 Experimenten der Fall. Für den Firefox Browser konnte jeweils in einem Experiment kein Eintrag gefunden werden. Die Mittelwerte in den Experimenten mit 60 Sekunden Ausführungszeit sind in Windows 10 fast doppelt so groß als bei den Experimenten mit 30 Sekunden Ausführungszeit ($7857,93 \pm 12055,97$ Millisekunden zu $3890,53 \pm 5144,75$ Millisekunden). Auch in Windows 11 ist der Mittelwert der Experimente mit 60 Sekunden Ausführungszeit größer ($3089,49 \pm 3802,54$ Millisekunden zu $4733,29 \pm 6088,69$ Millisekunden). Wie in den Zeitreihen der Abbildungen 3.4a und 3.4b zu sehen, sind die Werte bei einer Ausführungszeit von 60 Sekunden nicht so dicht beieinander, verglichen zu denen bei einer Ausführungszeit von 30 Sekunden. Auch ist die maximale Zeitstempelabweichung höher. Als Minimalwert für die Zeitstempelabweichung konnten sowohl in Windows 10 als auch in Windows negative Werte beobachtet werden. Diese lagen aber, je nach Ausführungszeit, zwischen $-617,99$ Millisekunden und $851,05$ Millisekunden.

3.3 Diskussion

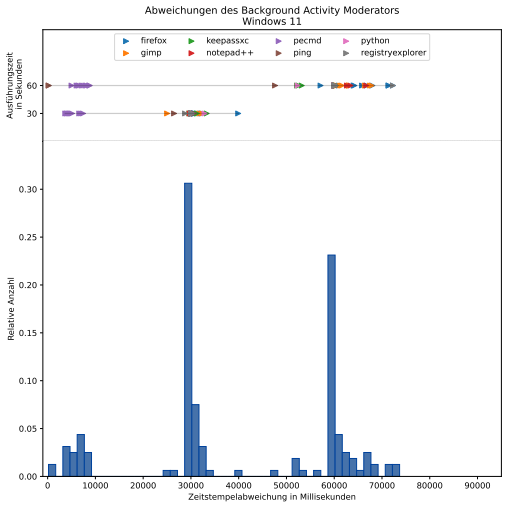
Ziel dieses Teils der Arbeit war es, eine Verteilung der Zeitstempel zu bestimmen, die Rückschlüsse auf die Ausführung von Anwendungen zulassen. Die Erstellung der Verteilung erfolgte durch die Ermittlung des Abstandes eines Zeitstempels zum tatsächlichen Ausführungszeitraum. Im Folgenden werden die Ergebnisse aus Abschnitt 3.2 erläutert und ihre Bedeutungen für eine forensische Untersuchung hervorgehoben.

3.3.1 Zuverlässigkeit der Datensicherung mit KAPE

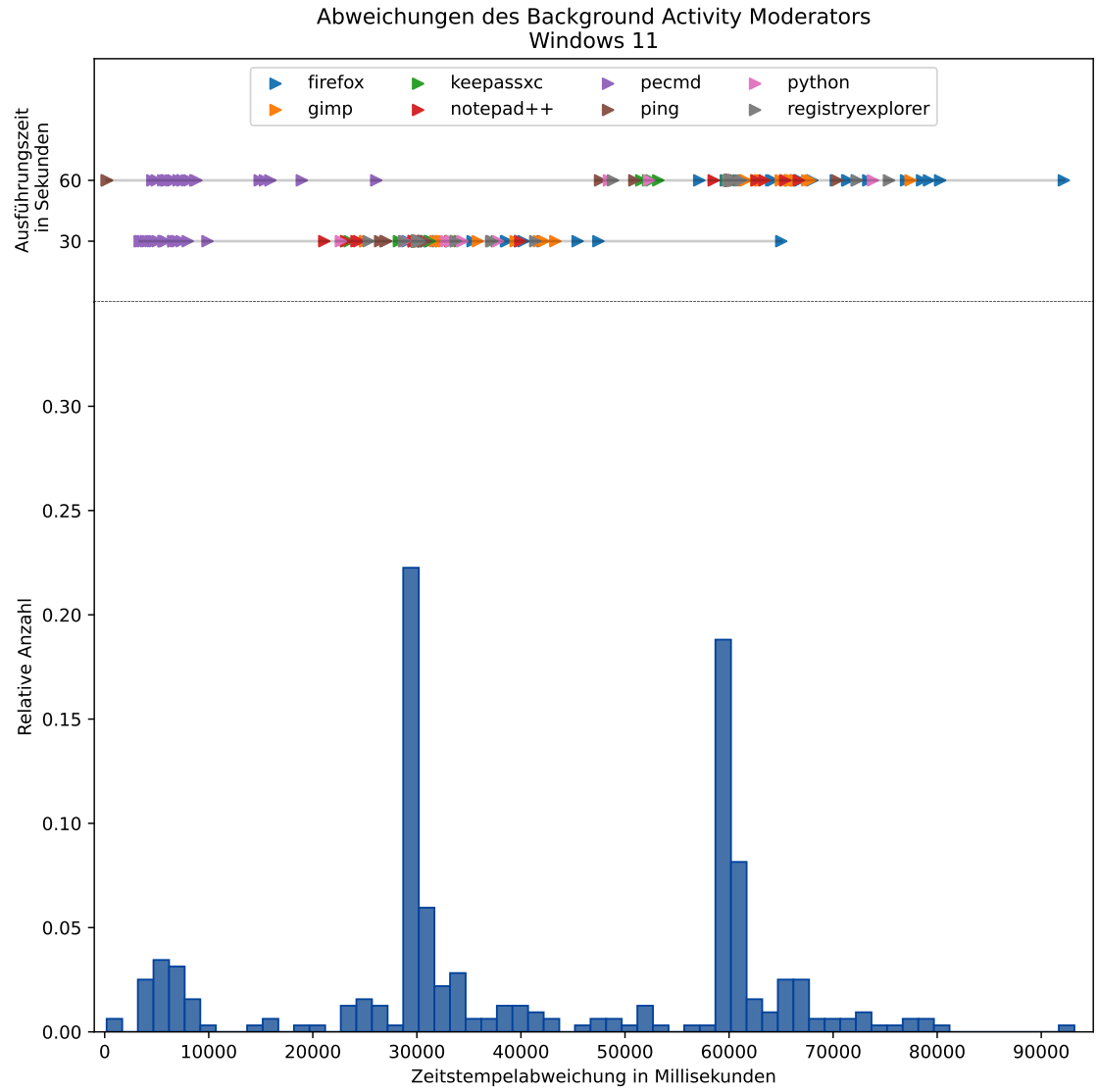
Die Sicherung der Daten wurde durch den Kroll Artifact Parser and Extractor (KAPE) durchgeführt. Dieser wurde über eine geplante Aufgabe durch die Windows Aufgabenplanung (engl. Windows Task Scheduler) gestartet. Für jedes Experiment wurden sechs Sicherungen geplant. Die erste sollte fünf Minuten nach dem letzten Anwendungsstart



(a) Histogramm der Zeitstempelabweichungen des Windows 10 Systems mit einer Balkenbreite von 1500 Millisekunden.

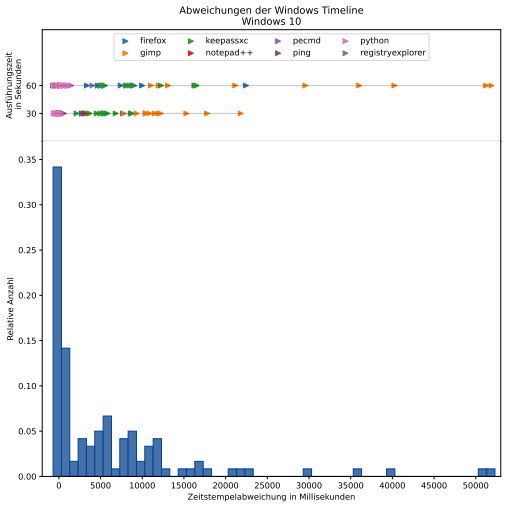


(b) Histogramm der Zeitstempelabweichung des Windows 11 Systems mit einer Balkenbreite von 1500 Millisekunden.

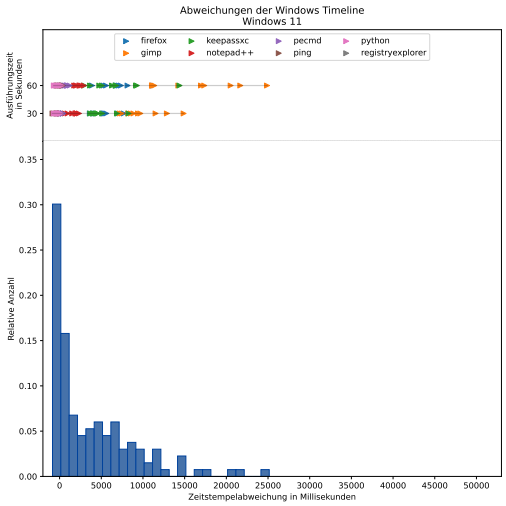


(c) Histogramm der zusammengeführten Zeitstempelabweichungen von Windows 10 und 11 mit einer Balkenbreite von 1500 Millisekunden.

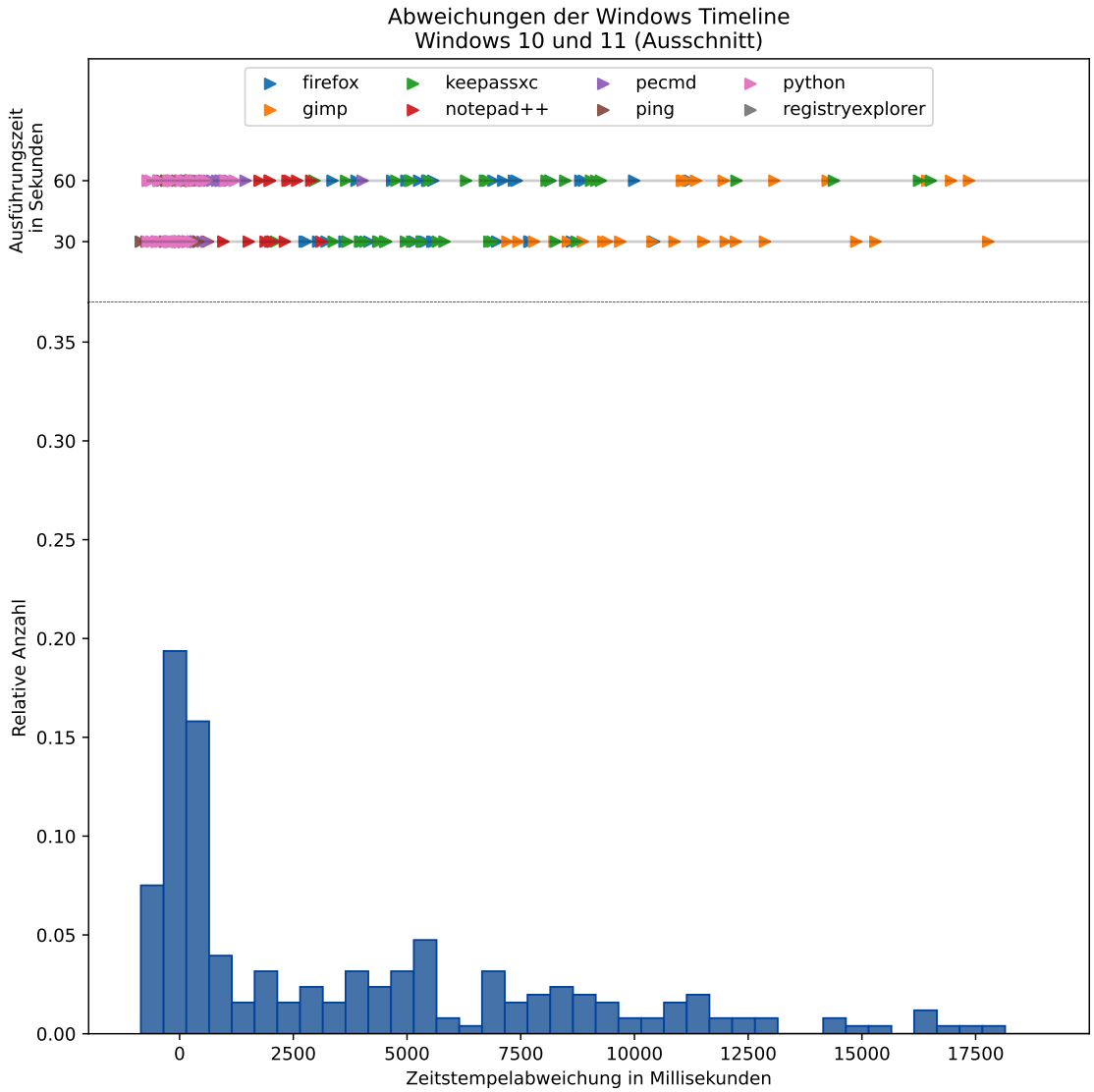
Abbildung 3.3: Histogramme der Zeitstempelabweichungen des Background Activity Moderator.



(a) Histogramm der Zeitstempelabweichungen des Windows 10 Systems mit einer Balkenbreite von 1000 Millisekunden.



(b) Histogramm der Zeitstempelabweichungen des Windows 11 Systems mit einer Balkenbreite von 1000 Millisekunden.



(c) Ausschnitt aus dem Histogramm der zusammengeführten Zeitstempelabweichungen von Windows 10 und 11 mit einer Balkenbreite von 500 Millisekunden.

Abbildung 3.4: Histogramme der Zeitstempelabweichungen der Windows Timeline.

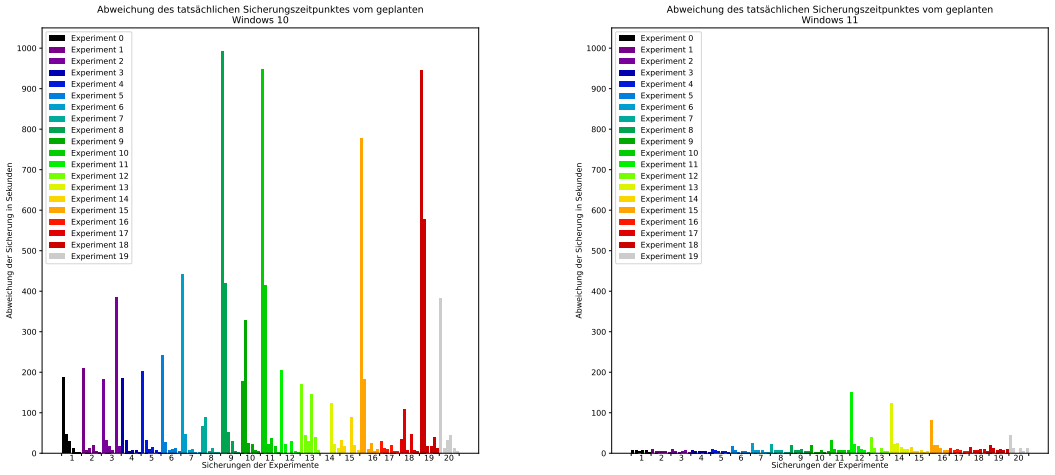
durchgeführt werden. Die zweite Sicherung wurde nach 15 und die dritte nach 30 Minuten durchgeführt. Anschließend wurde das System heruntergefahren und nach weiteren fünf Minuten die vierte Sicherung gestartet. Die fünfte Sicherung fand 15 und die sechste 30 Minuten nach dem Neustart des Systems statt. Mithilfe des Consolen-Logs von KAPE kann die Differenz zwischen dem geplanten und dem tatsächlichen Zeitpunkt der Sicherung bestimmt werden. Sie ist in der Form eines Balkendiagramms in der Abbildung 3.5a für Windows 10 und in 3.5b für Windows 11 abgebildet.

Die Abweichungen des tatsächlichen vom geplanten Sicherungszeitpunkt sind in Windows 10 um ein Vielfaches größer als unter Windows 11. So haben in Windows 10 in fünf Experimenten die ersten Sicherungen eine Abweichung von mehr als 400 Sekunden ($6,67$ Minuten). Eine Abweichung mehr als zehn Minuten findet man für die erste Datensicherung in den Experimenten 9 ($16,52$ Minuten), 11 ($15,81$ Minuten), 19 ($15,76$ Minuten) und 16 ($12,96$ Minuten). Die niedrigste Abweichung für die erste Sicherung weißen die Experimente 17 (48 Sekunden), 15 (54 Sekunden) und 18 (56 Sekunden) auf. Für die Experimente 12, 14 und 15 konnte für die dritte Sicherung kein Consolen-Log von KAPE gefunden werden, was darauf hindeutet, dass das System neugestartet wurde, bevor KAPE die Sicherung abschließen konnte. Im Experimente 14 fehlte das Console-Log zusätzlich für die ersten beiden Sicherungen.

Die größten Abweichungen für die erste Sicherung in Windows 11 weißen die Experimente 12 ($2,50$ Minuten), 14 ($2,07$ Minuten) und 16 ($1,38$ Minuten) auf. Die ersten Sicherungen in den anderen Experimenten haben eine Abweichung zwischen $8,26$ und $43,27$ Sekunden. Wie auch bei Windows 10 haben die Sicherungen zwei bis sechs eine geringere Differenz zwischen dem geplanten und dem tatsächlichen Sicherungszeitpunkt. Im Experiment 13 und 20 konnte für die dritte Sicherung in Windows 11 kein Console-Log gefunden werden. Diese Sicherungen scheinen auch durch den Neustart des System unterbrochen wurden zu sein.

3.3.2 Vorhandensein von Zeitstempeln

In keinem der durchgeführten Experimente konnten Zeitstempel im UserAssist Registryschlüssel gefunden werden. Dieser zeichnet nach Singh und Singh [70] nur dann den Start einer Anwendung auf, wenn diese durch den Windows Explorer (`explorer.exe`) gestartet wird. Da die Ausführung der Anwendungen in den Experimenten durch ein PowerShell Skript vorgenommen wurde, werden die Anwendungen nicht durch den Windows Explorer, sondern durch die PowerShell (`powershell.exe`) gestartet. Eine Datierung der Anwendungsstarts erfolgt demzufolge nicht im UserAssist Registryschlüssel. Für einige Anwendungen wurden trotzdem Einträge im UserAssist Registryschlüssel vorgefunden. Diese enthielten allerdings nur einen Wert `FocusCount`, der die Anzahl der Fokussierungen durch den Benutzer zählt. Anhand dessen lässt sich feststellen, dass eine Anwendung ausgeführt wurde, allerdings ist ein Rückschluss auf den Ausfüh-



(a) Abweichung des tatsächlichen Sicherungszeitpunktes zum Geplanten in Sekunden (Windows 10)

(b) Abweichung des tatsächlichen Sicherungszeitpunktes zum Geplanten in Sekunden (Windows 11)

Abbildung 3.5: Abbildung der Abweichung des tatsächlichen Sicherungszeitpunktes zum Geplanten in Sekunden.

rungszeitpunkt nicht möglich.

Nachdem das Logging der Prozessstarts in den Gruppenrichtlinien aktiviert wurde, werden in der Security.evtx alle Startzeitpunkte der Anwendungen im EVTX Event 4688 dokumentiert. Mithilfe dieser Events war es in allen Experimenten zuverlässig möglich, die Startzeitpunkte nachzuvollziehen. Als einziges betrachtetes forensisches Artefakt ermöglicht es auch, die Eltern-Anwendungen sowie den Kommandozeilenaufruf zu ermitteln. Letzteres muss allerdings in der Gruppenrichtlinie aktiviert werden.

Mit dem Background Activity Moderator war es in Windows 10 in einem Experiment nicht möglich den Startzeitpunkt für eine Anwendung nachzuvollziehen. Dafür konnte der Background Activity Moderator in 17 von 20 Experimenten in Windows 10 bereits in der ersten Sicherung gefunden werden. In Windows 11 war dies nur in 8 von 20 Experimenten der Fall. Da die erste Sicherung in Windows 10 im besten Fall $0,48$ Minuten und im schlechtesten Fall $16,52$ Minuten vom geplanten Startzeitpunkt, der sich fünf Minuten nach dem Start der letzten Anwendung befand, abweichen, kann für den Background Activity Moderator unter Windows 10 keine Aussage getroffen werden, wann das Artefakt entsteht. Auch unter Windows 11, wo die Abweichung der Sicherungenzeitpunkte deutlich geringer ist, lässt sich keine allgemeingültige Aussage zum Entstehungszeitpunkt formulieren. Da unter Windows 11 in der Mehrzahl der Experimente der Background Activity Moderator erst in späteren Sicherungen beziehungsweise spätestens nach dem Neustart des Systems vollständig war, ist anzunehmen, dass er nicht sofort aktualisiert wird. Eine Sicherung, die direkt nach dem Start einer Anwendung durchgeführt wird, kann somit unter Umständen den Start der Anwendung nicht enthalten.

Mithilfe der Prefetch-Dateien konnten in Windows 10 nur drei von 160 und in Windows 11 nur einer von 160 erwarteten Zeitstempeln nicht gefunden werden. In Windows 10 war dies zweimal für GIMP und einmal für Firefox der Fall. In Windows 11 hingegen fehlte der eine Zeitstempel für Ping.

Die Ausführungszeitpunkte konnten, sowohl in Windows 10 als auch in Windows 11, für den RegistryExplorer nicht mithilfe der Windows Timeline nachvollzogen werden. Für Notepad++ war dies unter Windows 10 in nur einem Experiment und in Windows 11 immerhin in zwölf Experimente möglich. Für Firefox wurde der Start unter Windows 10 und 11 jeweils nur in einem Experiment nicht dokumentiert. Der Zeitpunkt, an dem die Sicherung stattfindet, hatte dabei keinen Einfluss auf die Verfügbarkeit der Zeitstempel. Auch nach längerer Wartezeit und einem Systemneustart sind keine neuen Einträge hinzugekommen. Die Dauer, die ein Eintrag in der ActivitesCache.db gespeichert bleibt, wurde in dieser Arbeit nicht betrachtet. Andere Arbeiten, wie die von Marziale [32], zeigen, dass die Einträge bis zu 30 Tage lang vorgehalten werden. Voraussetzung dafür ist, dass der Benutzer dies in den Einstellungen aktiviert hat.

3.3.3 Verteilung der Zeitstempel

Für die Zeitstempelabweichungen sollte eine stochastische Wahrscheinlichkeitsverteilung bestimmt werden, mit der diese modelliert werden kann. Aus den in den Experimenten ermittelten Abweichungen wurden für jedes Artefakt eine Normalverteilung $X \sim N(\mu, \sigma)$ bestimmt, wobei μ den Mittelwert und σ die Standardabweichung beschreibt. Mithilfe des Kolomogrov-Smirnov-Test (kurz KS-Test) wurde anschließend überprüft, ob die Daten durch die aufgestellte Normalverteilung beschrieben werden.

Die Nullhypothese H_0 des KS-Test beschreibt den Umstand, dass die Daten durch eine Normalverteilung $X \sim N(\mu, \sigma)$ beschrieben werden. Als Signifikanzniveau wurde $\alpha = 0.05$ gewählt. In der Tabelle 3.4 sind die verwendeten Normalverteilungen sowie die Ergebnisse des Kolomogorov-Smirnov-Tests für jedes Artefakt abgebildet. Aus dieser geht hervor, dass keine der Normalverteilungen die Daten des jeweiligen Artefaktes beschreibt.

Die im Kapitel 3.2 abgebildeten Histogramme haben bereits gezeigt, dass die Zeitstempelabweichungen nicht einer Normalverteilung ähneln. Eine mögliche Ursache könnte sein, dass die Zeitstempelabweichungen in den Artefakten nicht als zufallsbasierte Ereignisse modelliert werden können, da sie deterministisch erzeugt werden. Für diese Ursache gibt es in den Daten einige Anzeichen, die im Folgenden erläutert werden.

Artefakt	Betriebssystem	Normalverteilung $X \sim N(\mu, \sigma)$	p-Wert	H_0 wird ...
Prefetch	Windows 10	$X \sim N(1798,45;5145,83)$	$1,23 * 10^{-38}$	abgelehnt
	Windows 11	$X \sim N(2224,84;6244,06)$	$1,51 * 10^{-35}$	abgelehnt
	kombiniert	$X \sim N(2013,00;5719,73)$	$2,59 * 10^{-70}$	abgelehnt
EVTX 4688	Windows 10	$X \sim N(0,22;0,90)$	$4,56 * 10^{-27}$	abgelehnt
	Windows 11	$X \sim N(0,30;1,07)$	$1,56 * 10^{-24}$	abgelehnt
	kombiniert	$X \sim N(0,26;0,99)$	$2,13 * 10^{-50}$	abgelehnt
BAM	Windows 10	$X \sim N(42965,10;20445,05)$	$3,55 * 10^{-6}$	abgelehnt
	Windows 11	$X \sim N(40078,05;20166,03)$	$1,17 * 10^{-7}$	abgelehnt
	kombiniert	$X \sim N(41517,05;20325,56)$	$3,98 * 10^{-13}$	abgelehnt
Windows Timeline	Windows 10	$X \sim N(5874,23;9442,12)$	$9,70 * 10^{-7}$	abgelehnt
	Windows 11	$X \sim N(3905,21;5115,59)$	$4,38 * 10^{-4}$	abgelehnt
	kombiniert	$X \sim N(4839,13;7535,15)$	$7,37 * 10^{-12}$	abgelehnt

Tabelle 3.4: Modellierung der Zeitstempelabweichungen mittels Normalverteilung und Ergebnisse des Kolmogorov-Smirnov-Tests.

Anzeichen beim Prefetch Artefakt

Sowohl in Windows 10 als auch in Windows 11 weisen über 85 Prozent der Prefetch Zeitstempel eine Abweichung von bis zu 600 Millisekunden auf. Die verbleibenden Zeitstempel haben eine größere Abweichung und stammen alle vom Firefox Browser. Diese entstehen durch den mehrfachen Start der `firefox.exe` Executable während der Laufzeit des Browser. Dieses wird verwendet, um Browser-Tabs und Hintergrundaufgaben in eigene Prozesse auszulagern. Da bei jedem Start erneut ein Zeitstempel in die Prefetch-Datei geschrieben wird und diese nur bis zu acht Zeitstempel speichern kann, fällt der tatsächliche Ausführungszeitpunkt aus der Prefetch-Datei heraus und die Abweichung zu diesem wird größer. In den Abbildungen 3.1a und 3.1b ist zu erkennen, dass die Firefox Zeitstempel bei einer längeren Laufzeit tendenziell eine größere Abweichung aufweisen, als bei einer Laufzeit von 30 Sekunden.

Anzeichen beim EVTX Artefakt

Die Zeitstempelabweichungen des Windows XML Event Log Event 4688 scheinen unabhängig von der ausgeführten Anwendung und der Ausführungszeit zu sein. Es ist auch das Artefakt mit der geringsten Abweichung und der geringsten Streuung.

Anzeichen beim Background Activity Moderator Artefakt

Im Histogramm, das die Zeitstempelabweichungen des Background Activity Moderators zeigt (siehe Abbildung 3.3), sind zwei markante Ausschläge bei 30 und 60 Sekunden zu sehen. Der Ausschlag bei 30 Sekunden stammt von Anwendungen mit einer Laufzeit von 30 Sekunden und der bei 60 Sekunden von Anwendungen mit einer Laufzeit von 60 Sekunden. Aus den Daten der Experimente lässt sich dadurch ableiten, dass der Background Activity Moderator den Zeitstempel der Beendigung einer Anwendung enthält. Ob dies auch für Anwendungen gilt, die länger als der 60 Sekunden ausgeführt werden und ob der Zeitstempel bereits während einer Ausführung aktualisiert wird, geht aus den Daten nicht hervor. Für den letzteren Punkt gibt es Anzeichen, da in zwei Experimenten für eine Anwendung der Zeitstempel zwischen zwei Sicherungen aktualisiert wurde.

In Windows 11 konnten in den meisten Experimenten erst nach einem Neustart beziehungsweise in einer späteren Sicherung ein Zeitstempel im Background Activity Moderator gefunden werden. Dies lässt sich mit der Funktionsweise der Registry erklären. Zur Laufzeit des Systems werden die Registry Hives im Arbeitsspeicher vorgehalten und alle Änderungen in einen Transaktionslog geschrieben. Dieser Transaktionslog wird bei einem Neustart oder nach einer bestimmten Zeit in den Registry Hive auf der Festplatte geschrieben. Da der Plaso Parser `winreg/bam` den Transaktionslog nicht auswerten kann, können die Informationen erst dann extrahiert werden, wenn diese bereits in den

Registry Hive übernommen worden sind. In Windows 10 konnte diese Verhalten nicht beobachtet werden. Da, wie in Abschnitt 3.3.1 beschrieben, die Sicherungzeitpunkte nicht wie geplant eingehalten wurden, ist davon auszugehen, dass der Background Activity Moderator im SYSTEM Hive ebenfalls nicht sofort aktualisiert wird.

Anzeichen beim Windows Timeline Artefakt

Die Windows Timeline ist das einzige Artefakt, in dem gehäuft negative Zeitstempelabweichungen beobachtet werden konnten. Auch ist es das Artefakt, in dem mit Abstand die wenigsten Ausführungen nachvollzogen werden konnten. Für den ersten Umstand gibt es eine technische Erklärung. In den anderen Artefakten wird der Ausführungszeitpunkt im Filetime Format mit einer maximalen Auflösung von 100-Nanosekunden angegeben. In der Windows Timeline wird hingegen das Epoch-Format mit einer maximalen Ausflösung von einer Sekunde verwendet. Die Unterschiede in den Auflösungen haben zur Folge, dass der Epoch-Zeitstempel bis zu $\pm 999,9999$ Millisekunden vom Filetime Zeitstempel abweichen kann. Dies erklärt zum einen die negativen Abweichungen, zum anderen aber auch die Konzentration der Zeitstempel im Interval von -1000 bis 1000 Millisekunden.

3.3.4 Reihenfolge der Artefakte

Aus den Daten der Experimente ergibt sich folgende Reihenfolge der Zeitstempelabweichungen. Das Event 4688 weist den geringsten Abstand zum eigentlichen Startzeitpunkt auf. Anschließend folgen die Zeitstempel in den Prefetch-Dateien. Die Einträge in der Windows Timeline weichen um einige Sekunden von dem eigentlichen Ausführungszeitpunkt ab. Die größten Differenzen zum Startzeitpunkt konnten beim Background Activity Moderator beobachtet werden.

3.3.5 Einsatzmöglichkeit der Zeitstempelverteilung

In ihrer Arbeit bestimmten James und Jang [22] für die Erzeugung von Einträgen in der Browser Chronik die Zeitstempelabweichungen und modellierten diese als Normalverteilung. Mithilfe dieser Normalverteilung bestimmten sie einen Schwellwert θ, den sie anschließend für die Gruppierung von gleichartigen Ereignissen verwendeten. Unter der Verwendung der Zeitstempel des zuverlässigsten (t_1) und des unzuverlässigsten Artefaktes (t_2), das heißt dem Artefakt mit der kleinsten und dem mit der größten Zeitstempelabweichung, bestimmten sie ein Intervall, in dem weitere gleichartige Ereignisse zu finden sein müssten. Für die Bestimmung des Schwellwertes θ nutzten sie den Wert 2σ aus der Normalverteilung.

$$[t_2 - \theta; t1] \tag{3.1}$$

James und Jang [22] hoben in ihrer Arbeit hervor, dass diese Methode der Gruppierung an ihre Grenzen stößt, wenn mehrere gleichartige Events in einem kurzen Zeitraum ähnliche Artefakte produzieren. Da keins der vier Ausführungsartefakte über eine Normalverteilung abgebildet werden konnte, wird im dritten Teil dieser Arbeit der Schwellwert θ mithilfe des 0,75-Quantil ($\tilde{X}_{0,75}$) bestimmt.

4 Korrelation der Windows Pfadformate

Die Bezeichnung von Dateien in forensischen Artefakten eines Windows 1x Systems ist nicht einheitlich. So kann das Artefakt 1 eine Datei als `\\?\Volume{edeb6740-2b25--4263-89b2-c4b363291f19}\File1.txt` und Artefakt 2 `C:\File1.txt` bezeichnen. Für die Überprüfung der Hypothese, dass beide Pfadangaben dieselbe Datei beschreiben, müssen die symbolischen Verknüpfungen des NT Objekt Managers ausgewertet werden. Diese liegen allerdings nur vor, wenn die Maschine angeschaltet ist. Wurde diese bereits heruntergefahren, kann der Namensraum des NT Objekt Manager nicht mehr konsolidiert werden. Dieser Abschnitt verfolgt das Ziel, die verfügbaren symbolischen Verknüpfungen für Partitionen, Festplatten und CD-Rom zu kartieren und deren Aufbau zu beschreiben. Ziel ist es, diese Informationen aus nicht volatilen Artefakten wiederherstellen zu können.

4.1 Methode

Für die Kartierung des NT Objekt Manager Namensraum wird dieser zunächst gesichert und anschließend in eine Graphstruktur überführt. Für die Rekonstruktion des relevanten Teils des NT Objekt Manager Namensraums werden Strukturen auf der gesicherten Festplatte gesucht, die dafür verwendet werden können. Die folgenden Abschnitte beschreiben die Methodik detaillierter.

4.1.1 Aufsetzten der Testsysteme

Im ersten Schritt der Arbeit werden drei `Windows 10 Pro` (Build 19043) Systeme in `Virtual Box` (6.1.32) [62] aufgesetzt. Allen Maschinen wurde ein Prozessorkern, 2048 MB Arbeitsspeicher und eine 50 GB Festplatte zugewiesen. Auf der ersten Maschine wurde Windows 10 auf einer Festplatte mit GUID Partition Table (kurz GPT) installiert. Bei der zweiten und dritten Maschine wurde Windows auf einer Master Boot Record (kurz MBR) formatierten Festplatte installiert. Im Unterschied zur zweiten Maschine wurde bei der dritten keine leere Festplatte verwendet. Stattdessen wurden vorher zwei Partitionen mit einer Größe von 1000 MB und dem MBR Partitionstype `0x06` angelegt.

4.1.2 Sicherung des NT Objekt Manager Namensraum

Die Sicherung und Auswertung des NT Objekt Manager Namensraum erfolgte im zweiten Schritt. Die Sicherung des Namensraumes in das CSV-Format erfolgt mit der Hilfe des PowerShell Skriptes `Export-NtObjectManager`, dass das Modul `NtObjectManager` [14] in der Version 1.1.32 verwendet. Der gesicherte NT Objekt Manager Namens-

raum wurde in Python mithilfe des Pakets `pandas` (1.3.1) [74] geladen und anschließend mittels `networkx` (2.6.2) [15] in eine Graphstruktur überführt. Diese Graphstruktur ermöglicht eine Nachverfolgung von symbolischen Verknüpfungen. Die einzelnen Graphen wurden nach Geräteobjekten durchsucht, die „disk“ oder „volume“ im Namen enthalten. Für diese wurden, mithilfe des Algorithmus „Single-Target-Shorted-Paths“, alle symbolischen Verknüpfungen ermittelt, die auf dieses Geräteobjekt verweisen.

4.1.3 Sicherung der Festplatte

Die Sicherung der Festplatte erfolgt im Expert Forensic Wittness (kurz EWF) Disk Image Format mittel des `FTK Imagers`[1] (4.5.0.3). Um auch die Partitionstabelle betrachten zu können, wird die vollständige Festplatte gesichert.

4.1.4 Auswertung der Festplatte

Eine Auswertung der, auf der Festplatte, gespeicherten Informationen erfolgt mit den folgenden Werkzeugen:

- `HxD` (2.5.0.0) [21] - Betrachten von binär Dateien
- `The Sleuth Kit` (4.10.2) [6] - Betrachten des Aufbaus der Festplatte und extrahieren von Dateien
- `MbrGptParser.py` (0.2.3) [18] - Analyse der Partitionstabelle
- `Registry Explorer` (1.6.0) [85] - Auswertung der Registry Hives

4.2 Ergebnisse

Die symbolischen Verknüpfungen, die im NT Objekt Manager Namensraum erstellt werden, hängen von dem Gerät ab, auf das sie verweisen. So besitzt das Gerät, das eine CDRom repräsentiert andere symbolische Verknüpfungen als ein Gerät das eine Partition auf einer Festplatte repräsentiert. Des Weiteren hängen die verfügbaren Verknüpfungen auch von der Funktion des Gerätes ab. So hat eine Partition, auf der das Windows Betriebssystem gespeichert ist, zusätzliche symbolische Verknüpfungen, die sie als solche kennzeichnen. Nachfolgend werden die beobachteten Strukturen der symbolischen Verknüpfungen vorgestellt. Für den variablen Teil in einer symbolischen Verknüpfung werden folgende Bezeichnungen gewählt:

[x] Nummer der Festplatte oder des CdROM Geräts

[y] Nummer der Partition auf einer Festplatte

[z] Nummer eines Volumes im System

Bei allen drei Bezeichnern handelt es sich um numerische Werte. Wie diese bestimmt werden können, wird in Abschnitt 4.3 erläutert.

4.2.1 CdRom Gerät

Auf das CdRom Gerät verweisen drei symbolische Verknüpfungen, aus dem Verzeichnis `\GLOBAL??`. Die linke symbolische Verknüpfung in Abbildung 4.1 gibt den Laufwerkbuchstaben an, an dem die CdRom eingehängt wurde. In der Mitte wird die Volume-GUID mit dem Gerät verknüpft und im rechten wird eine für Anwendung nutzbare symbolische Verknüpfung `CdRom0` erstellt.

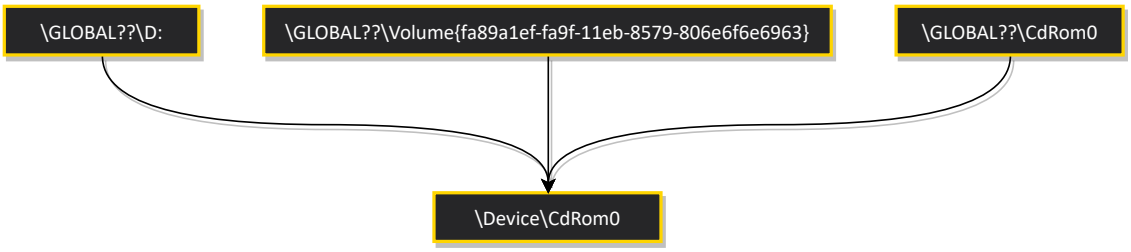


Abbildung 4.1: Beispiel für symbolische Verknüpfungen auf das Gerät `\Device\CdRom0`

4.2.2 Festplatten

Insgesamt vier symbolische Verknüpfungen verweisen auf die Festplatte, die als `DR[x]` im Verzeichnis `\Device\Harddisk[x]` bezeichnet wird. Zwei der symbolischen Verknüpfungen liegen im Verzeichnis `\GLOBAL??`. Die erste verknüpft dabei die Disk-GUID mit dem Gerät, dass die Festplatte repräsentiert. Die zweite symbolische Verknüpfung hat die Form `PhysicalDrive[x]`.

Die Verknüpfung `multi(0)disk(0)rdisk([x])` im Verzeichnis `\ArcName` verweist in allen drei virtuellen Maschinen auf die symbolische Verknüpfung `Partition0` im Verzeichnis `\Device\Harddisk[x]`, die wiederum auf das Gerät verweist.

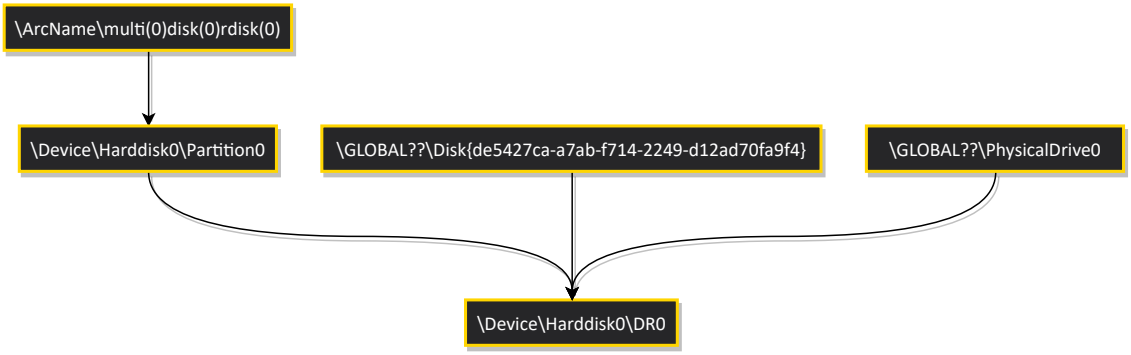


Abbildung 4.2: Beispiel für symbolische Verknüpfungen auf das Gerät `\Device\ Harddisk0\ DR0`

4.2.3 Partitionen

Die vorhanden symbolischen Verknüpfungen unterscheiden sich je nach Funktion und Eigenschaft der Partition. Das vom Treiber erzeugte und benannte Gerät einer Partition befindet sich immer im Verzeichnis `\Device`. Die Benennung folgt dem Schema `HarddiskVolume[z]`. Der Wert von `z` startet bei 1 und wird für jede im System registrierte Partition um eins erhöht. Der Wert `z` hängt davon ab, wie viele Partitionen bereits vom entsprechenden Treiber am System registriert wurden [60].

Im folgenden werden die folgenden Funktionen und Zustände von Partitionen betrachtet:

- Partition mit und ohne Laufwerksbuchstabe
- Bootpartition
- Windows Partition
- Windows Recovery Partition (nur GPT)
- Microsoft Reserved Partition (nur GPT)

Partition mit und ohne Laufwerksbuchstaben

Partitionen mit und ohne Laufwerksbuchstaben werden im NT Objekt Manager unabhängig vom verwendeten Partitionsschema gleich abgebildet. Die Abbildung 4.3 zeigt, wie eine Partition mit Laufwerksbuchstaben im NT Objekt Manager durch symbolische Verknüpfungen dargestellt wird. Ist einer Partition kein Laufwerksbuchstabe zugewiesen, fehlt im NT Objekt Manager die symbolische Verknüpfung des Laufwerksbuchstaben (in Abbildung 4.3 `\GLOBAL??\F:`).

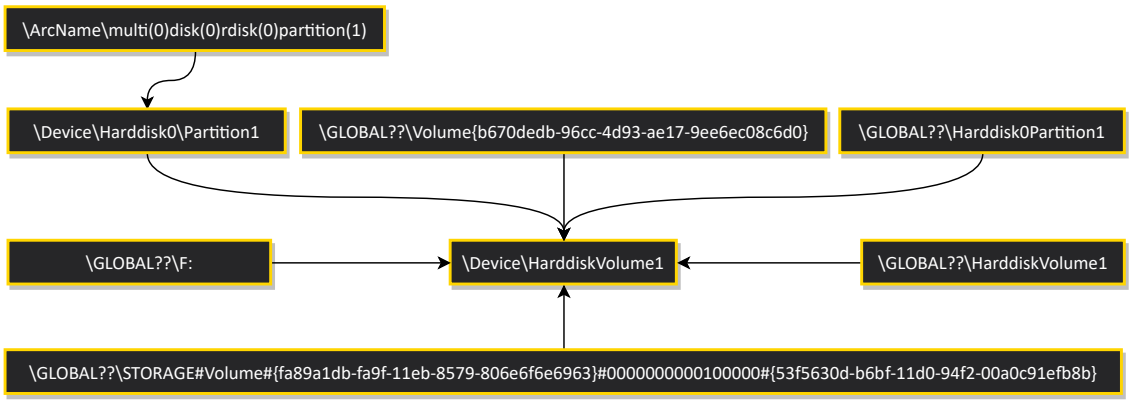


Abbildung 4.3: Beispiel für symbolische Verknüpfungen auf das Gerät `\Device\ HarddiskVolume1`, dass eine Partition repräsentiert der eine Laufwerksbuchstabe zugewiesen wurde

Wie auch bei der Festplatte (siehe Abschnitt 4.2.2), verweist die Verknüpfung im `\ArcName` Verzeichnis auf eine Verknüpfung `\Device\Harddisk[x]\Partition[y]`. Allerdings entspricht der Wert `y` diesmal der Partitionsnummer, die mit eins beginnend

die Partitionen einer Festplatte zählt. Der Name der symbolischen Verknüpfung im `\ArcName` Verzeichnis wird bei einer Partition allerdings um die Angabe der Partitionsnummer (`partition([z])`) ergänzt. Weitere Verknüpfungen, die auf das Gerät `\Device\ HarddiskVolume[z]` verweisen, liegen im `\GLOBAL??` Verzeichnis. Sie haben die Form
`Volume{[Volume GUID]}`), `Harddisk[x]Partition[y]`, `HarddiskVolume[z]` und `STORAGE#Volume#{[Disk Id]}#[Byte Offset]#{[Geräteklassen GUID]}`.

Windows Partition

Ist auf einer Partition das Windows 10 Betriebssystem installiert, werden, zusätzlich zu den bereits genannten symbolische Verknüpfungen, die in Abbildung 4.4 abgebildeten Verknüpfungen gefunden. Im `\Device` Verzeichnis existieren die zusätzlichen Verknüpfungen `BootDevice`, `BootPartition` und `OSDataDevice`. Die im Wurzelverzeichnis angesiedelte Verknüpfung `OSDataRoot` zeigt auf `OSDataDevice`, die ihrerseits auf die Partition im `\ArcName` Verzeichnis verweist. Im `\GLOBAL??` Verzeichnis befindet sich eine einzige zusätzliche symbolische Verknüpfung mit dem Namen *BootPartition*.

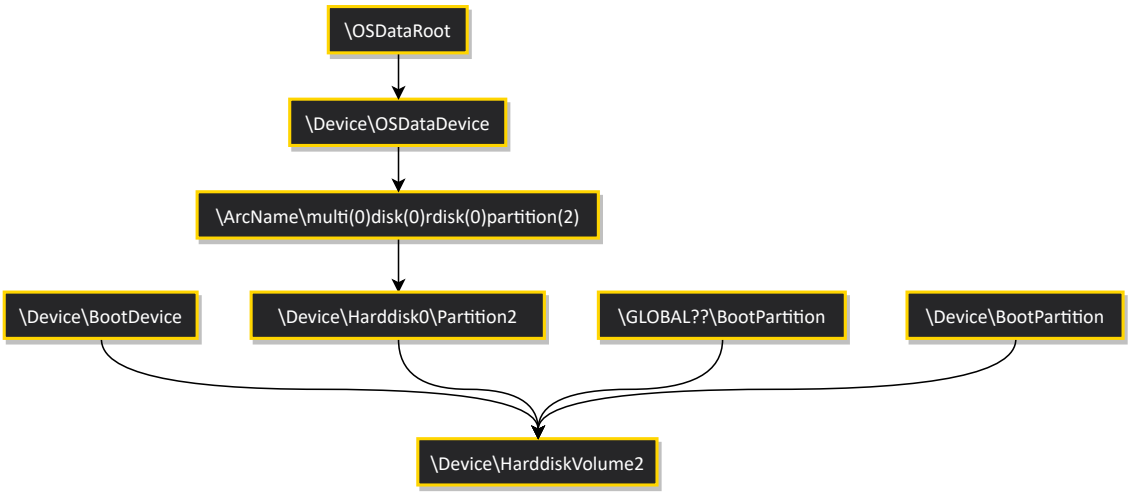


Abbildung 4.4: Beispiel für symbolische Verknüpfungen auf das Gerät `\Device\ HarddiskVolume2`, dass eine Partition repräsentiert, auf der das Windows 10 Betriebssystem installiert wurde. In der Abbildung werden nur die zusätzlichen Verknüpfungen gegenüber der Abbildung 4.3 dargestellt.

Bootpartition

Die Partition, auf der sich der Bootloader befindet, erzeugen im NT Objekt Manager ähnliche Verknüpfungen, wie eine Partition ohne Laufwerksbuchstaben (siehe Abschnitt 4.2.3) Zusätzlich dazu verweisen die Verknüpfungen `\GLOBAL??\SystemPartition` und `\Device\SystemPartition` auf das Gerät, dass die Partition repräsentiert.

Windows Recovery Partition

Auf das Gerät, dass die Windows Recovery Partition repräsentiert, verweisen dieselben Verknüpfungen, wie bei einer Partition, der kein Laufwerksbuchstabe zugeordnet wurde.

Microsoft Reserved Partition

Die Microsoft Reserved Partition tritt nur bei GPT formatierten Festplatten auf. Die beobachtbaren symbolischen Verknüpfungen waren dieselben wie bei einer Festplatte ohne Laufwerksbuchstaben, mit dem Unterschied, dass die Verknüpfung der Form `\GLOBAL??\Volume{Volume GUID}` fehlt.

4.3 Diskussion

In diesem Abschnitt wird der inhaltliche Aufbau der symbolischen Verknüpfungen beschrieben. Anschließend wird erläutert, wie die symbolischen Verknüpfungen mithilfe auf der Festplatte befindlichen Datenstrukturen rekonstruiert werden können.

4.3.1 CdRom Gerät

Auf ein CdRom Gerät verweisen insgesamt drei symbolische Verknüpfungen (siehe Abschnitt 4.2.1).

CdRom[x]

Das Geräteobjekt `\Device\CdRom[x]` und die symbolische Verknüpfung `\GLOBAL??\CdRom[x]` besitzen beide einen variablen Wert `x`. Dieser wird für jedes CdRom Gerät im System um eins inkrementiert, beginnend bei Null. Die Reihenfolge, in der die Geräte betrachtet werden, richtet sich nach dem SCSI Port, an dem sie am System eingebunden sind. Dabei gilt: Das Gerät, dass an dem niedrigsten SCSI Port hängt, wird als erstes vom System erkannt.

Beispiel An einem System hängen zwei CdRom Geräte, wobei das erste den SCSI Port 3 und das zweite den SCSI Port 5 benutzt. Dem Geräteobjekt an SCSI Port 3 wird 0 für den Wert `x` zugewiesen. Das Gerät an SCSI Port 5 erhält den Wert 1.

Der SCSI Port an dem ein Gerät hängt, lässt sich über den Wert `LocationInformation` im Registryschlüssel `SYSTEM:\ControlSet[ControlSet]\Enum\SCSI\CdRom&Ven_[Vendor]&Prod_[Produkt]\[Instanz Id]` ermitteln. Bei dem Wert handelt es sich

um eine Zeichenkette mit dem Aufbau `BusNumber [SCSI Port], Target Id [Target Id], LUN [LUN]`.

\GLOBAL??\[Laufwerksbuchstabe]

Jedes CdRom Laufwerk wird ein Laufwerksbuchstabe zugeordnet, den der Anwender nutzen kann, um auf das Laufwerk zuzugreifen. Für die Zuweisung der Laufwerksbuchstaben ist der Mount Manager verantwortlich. Er erstellt im NT Objekt Manager symbolische Verknüpfungen, die die Form `\GLOBAL??\Volume{[Volume GUID]}` haben. Des Weiteren verwaltet er in der Registry unter `SYSTEM\MountedDevices` eine Datenbank, die den Laufwerksbuchstaben eine Volume GUID zuordnet. Die Datenbank wird für die persistente Speicherung der Zuordnung benötigt, da die symbolischen Verknüpfungen im NT Objekt Manager beim Herunterfahren des Systems verloren gehen. [54]

Mithilfe dieser Datenbank ist eine Rekonstruktion der Zuordnung des Laufwerksbuchstabens zu einem CdRom Gerät möglich. Der Laufwerksbuchstaben wird aus dem Wertenamen im Registryschlüssel `SYSTEM\ MountedDevices` bestimmt. Er besitzt die Form `\DosDevices\[Laufwerksbuchstabe]:`. Der Wert hat die Form `\??\SCSI#CdRom&Ven_[Vendor]&Prod_[Produkt]#[Instanz Id]#[Geräteklassen GUID]`. Er kann einem CdRom Gerät über den Registryschlüssel `SYSTEM:\ControlSet[ControlSet]\ Enum\SCSI\CdRom&Ven_[Vendor]&Prod_[Produkt]\[Instanz Id]` eindeutig zugeordnet werden.

\GLOBAL??\Volume {[Volume GUID]}

Die Volume GUID wird, wie der Laufwerksbuchstabe, durch den Mount Manager einem Gerät zugeordnet. Eine Zuordnung zu einem CdRom Gerät im Registryschlüssel `SYSTEM:\ControlSet[ControlSet]\Enum\SCSI\CdRom&Ven_[Vendor]&Prod_[Produkt]\[Instanz Id]` erfolgt, ähnlich wie beim Laufwerksbuchstaben im Abschnitt 4.3.1, über den Wertenamen im Schlüssel `SYSTEM\MountedDevices`. Dieser Name hat die Form `{[Volume GUID]}`.

4.3.2 Festplatten

Auf das NT Objekt `\Device\Harddisk[x]\DR[x]` verweisen drei symbolische Verknüpfung direkt und eine indirekt.

Harddisk[x]

Das NT Objekt `\Device\Harddisk[x]\DR[x]` und die symbolische Verknüpfung der Form `\Device\Harddisk[x]\Partition0` besitzen beide den variablen Wert `x`. Die-

ser wird, bei Null beginnend, für jede im System existierende Festplatte um eins erhöht. Die Reihenfolge, in der die Festplatten am System erkannt werden, hängt vom verwendeten SCSI Port ab. Wie bei den CdRom Geräten (siehe Abschnitt 4.3.1) wird die Festplatte am niedrigeren SCSI Port vor einer Festplatte an einem höheren Port erkannt. Für die Ermittlung des Wertes `x` zählen dabei nur am System befindliche Festplatten und keine anderen SCSI Geräte.

\GLOBAL??\PhysicalDrive[x]

Die symbolische Verknüpfung `PhysicalDrive[x]` im Verzeichnis `\GLOBAL??` des NT Objekt Manager Namensraum ermöglicht den Zugriff auf die Festplatte für Anwendungen. Der variable Wert `x` wird, wie im Abschnitt 4.3.2 beschrieben, ermittelt.

\ArcName\multi(0)disk(0)rdisk([x])

Der variable Wert `x` in `\ArcName\multi(0)disk(0)rdisk([x])` wird, wie in den Abschnitten 4.3.2 und 4.3.2 bereits dargestellt, ermittelt. Die symbolische Verknüpfung verweist allerdings nicht direkt auf das NT Objekt, das die Festplatte repräsentiert. Stattdessen verweist es auf eine symbolische Verknüpfung der Form `\Partition0` im Verzeichnis `\Device\Harddisk[x]` des NT Objekt Manager Namensraum.

\GLOBAL??\Disk{[Disk GUID]}

Die Wiederherstellung der symbolischen Verknüpfung `\GLOBAL??\Disk{[Disk GUID]}` ist derzeit nicht möglich, da nicht bekannt ist, wie die `Disk GUID` durch Windows vergeben wird. In der GUID Partitionstabelle existiert zwar ein 16 Byte langer Eintrag, mit dem Namen Disk GUID. Diese GUID stimmt allerdings nicht mit der Disk GUID in der symbolischen Verknüpfung überein.

4.3.3 Partitionen

Auf ein NT Objekt, das eine Partition repräsentiert, verweisen je nach Eigenschaft der Partition unterschiedliche symbolische Verknüpfungen.

Harddisk[x]Partition[y]

Die beiden symbolischen Verknüpfungen `\Device\Harddisk[x]Partition[y]` und `\GLOBAL??\Harddisk[x]Partition[y]` besitzen die beiden variablen Werte `x` und `y`. Der Wert `x` gibt die Nummer der Festplatte an, auf der sich die beschriebene Partition

befindet. Dieser Wert startet bei null und wird, wie im Abschnitt 4.3.2 beschrieben, ermittelt. Die Nummer der Partition auf der Festplatte wird durch den Wert `y` angegeben. Die erste Partition auf einer Festplatte erhält den Wert eins. Die Reihenfolge, in der die Partitionen gezählt werden, entspricht der Reihenfolge, wie sie in der Partitionstabelle aufgeführt werden.

\ArcName\multi(0)disk(0)rdisk([x])partition([y])

Die Werte `x` und `y` in der symbolischen Verknüpfungen `\ArcName\multi(0)disk(0)rdisk([x])partition([y])` werden, wie bereits in Abschnitt 4.3.3 beschrieben, ermittel. Die Verknüpfung im `ArcName` Verzeichnis, die eine Partition bezeichnet, verweist allerdings nicht wie die meisten Verknüpfungen auf `\Device\HarddiskVolume[z]`. Stattdessen zeigt sie auf eine Verknüpfung der Form `\Device\Harddisk[x]Partition[y]`.

HarddiskVolume[z]

Die Gerätebezeichnung `\Device\HarddiskVolume[z]` beschreibt ein Volume und wird vom Treiber im NT Objekt Manager angelegt. Der Wert `z` ist numerisch und startet bei eins. Er wird für jedes neu erstellte Volume-Gerät um eins inkrementiert. [60] Der Wert `z` in der symbolischen Verknüpfung `\GLOBAL??\HarddiskVolume[z]` besitzt dieselbe Bedeutung.

Der numerische Wert `z` hängt von der Anzahl der bereits registrierten Festplatten und Partitionen ab. Die Reihenfolge der Festplatten orientiert sich an den SCSI Ports, an denen die Festplatten hängen (siehe Abschnitt 4.3.2). Die Partitionen werden in ihrer Reihenfolge betrachtet, wie sie in der Partitionstabelle einer Festplatte aufgeführt werden.

Beispiel Ein System besitzt zwei Festplatten mit jeweils zwei Partitionen. Die erste Festplatte wurde am SCSI Port 0 und die zweite am Port 4 angeschlossen. Den Partitionen werden die folgenden Werte `z` zugewiesen:

- Festplatte 0 Partition 1 $\Rightarrow$ `z = 1`
- Festplatte 0 Partition 2 $\Rightarrow$ `z = 2`
- Festplatte 1 Partition 1 $\Rightarrow$ `z = 3`
- Festplatte 1 Partition 2 $\Rightarrow$ `z = 4`

\GLOBAL??\[Laufwerksbuchstabe]

Jeder Partition, die in einem Windows System eingehängt wird und auf die der Anwender zugreifen kann, wird ein Laufwerksbuchstabe zugeordnet. Die Zuweisung des

Laufwerksbuchstaben erfolgt durch den Mount Manager. Er erstellt im NT Objekt Manager eine symbolische Verknüpfung mit der Form `\GLOBAL??\Volume[Volume GUID]`. Dieser ordnet er einen Laufwerksbuchstaben zu. Damit diese Zuordnung auch nach einem Neustart des Systems bestehen bleibt, pflegt der Mount Manager in der Registry eine Datenbank (`SYSTEM\MountedDevices`). [54]

Mithilfe dieser Datenbank kann ein Laufwerksbuchstabe einer `Volume GUID` zugeordnet werden. Der Laufwerksbuchstabe ist als Wertenamen unter `SYSTEM\MountedDevices` in der Form `\DosDevices\[Laufwerksbuchstabe]:` gespeichert. Der binäre Wert repräsentiert die `Volume GUID`. Die folgenden zwei Codierungen der `Volume GUID` sind möglich.

DMIO:ID Format Ist der binäre Wert 24 Bytes lang, so handelt es sich um eine `Volume GUID` im `DMIO:ID` Format. In diesem entsprechen die ersten 8 Bytes der Signatur `DMIO:ID`. Die verbleibenden 16 Bytes repräsentieren die `Volume GUID`. [61]

Disk Signature - Offset Format Das zweite Format weist eine Länge von 12 Bytes auf. Die ersten 4 Bytes entsprechen der Disk Signature der Festplatte. Diese wird nur bei MBR formatierten Festplatten eingesetzt. Die verbleibenden 8 Bytes geben den Offset der Partition in Bytes an. Aus diesen beiden Informationen lässt sich, wie in Abschnitt 4.3.3 ausgeführt, die Volume GUID rekonstruieren. [61]

\GLOBAL??\Volume{[Volume GUID]}

Bei der Volume GUID handelt es sich um einen 16 Byte großen Identifikator für Volumes. Die 16 Byte werden in fünf Blöcke geteilt. Der erste Block weist eine Größe von vier Byte auf. Er wird gefolgt von drei, zwei Byte großen, Blöcken. Die übrigen sechs Byte repräsentieren den letzten Block. Die Bedeutung der 16 Byte einer Volume GUID hängt von dem Partitionsschema der Festplatte ab, auf dem eine Partition erstellt wurde. Im NT Objekt Manager Namensraum verweist die symbolische Verknüpfung `\GLOBAL??\Volume{[Volume GUID]}` auf das Geräteobjekt einer Partition.

Volume GUID bei MBR formatierten Festplatten Bei MBR formatierten Festplatten entspricht der erste GUID-Block der „MBR Disk Signature“. Bei dieser handelt es sich um einen vier Byte großen Wert, der am Offset 0x1B8 des Master Boot Records gefunden werden kann. Die folgenden drei Blöcke der Volume GUID haben den Wert „0000“. Der letzte Block repräsentiert bei MBR formatierten Festplatten den Offset in Bytes, an dem eine Partition startet. Um diesen Offset zu erhalten, werden die sechs Byte als Little-Endian interpretiert und um den Wert 16 nach bitweise nach links verschoben.

Beispiel: Die Abbildung 4.5 zeigt einen Auschnitt des Master Boot Records. In Gelb wurde die „MBR Disk Signature“ markiert. Diese wird im Little-Endian Format betrachtet und repräsentiert die ersten vier Bytes der Volume GUID. In Schwarz

wurde der Offset der Partition eingefärbt. Dieser wird ebenfalls als Little-Endian interpretiert. Da die Angabe mithilfe des Logical Block Addressing erfolgt, muss bei der Umrechnung zum Byte Offset der Wert mit `0x200 (512)` multipliziert werden. Dies entspricht in diesem Beispiel `0x800 * 0x200 = 0x100000`. Anschließend muss dieser Wert um `0x10 (16)` nach rechts verschoben werden und wieder in das Little Endian Format überführt werden. Die drei zwei Byte großen Blöcke der GUID werden mit Nullen gefüllt. Die Volume GUID für dieses Beispiel lautet: `{6eed82d3-0000-0000-0000-100000000000}`.

```
000001b0: 65 6d 00 00 00 63 7b 9a d3 82 ed 6e 00 00 80 20  em...c{....n...
000001c0: 21 00 07 7f 39 06 00 08 00 00 00 90 01 00 00 7f  !...9...........
000001d0: 3a 06 07 fe ff ff 00 98 01 00 7c f3 2d 06 00 fe  :.........|.-...
000001e0: ff ff 27 fe ff ff 00 90 2f 06 00 60 10 00 00 00  ..'...../..`....
000001f0: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 aa  ..............U.
```

Abbildung 4.5: Partitionstabelle des Master Boot Records. Die “MBR Drive Signature“ wurde gelb markiert. In Schwarz eingefärbt wurde der Start der Partition (Logical Block Addressing, kurz LBA).

Volume GUID bei GPT formatierten Festplatten Bei Partitionen, die auf einer GPT formatierten Festplatte existieren, entspricht die Volume GUID der eindeutigen Partitions-GUID (engl. Unique Partition GUID), wie sie in der Partitionstabelle angegeben wird. Liegt eine GUID im binären Format vor, so werden die ersten drei Blöcke (8 + 4 + 4 Bytes) als Little-Endian ausgewertet. Die letzten zwei Blöcke (4 + 12 Bytes) müssen im Big Endian Format betrachtet werden. Bei GPT formatierten Festplatten weist der Mount Manager, mit Ausnahme der Microsoft Reserved Partition (`Partitionstyp: {e3c9e316-0b5c-4db8-817d-f92df00215ae}`), jeder Partition eine Volume GUID zu [55].

Beispiel Die Abbildung 4.6 zeigt einen GPT Partitionseintrag, bei dem, mit gelber Farbe, die eindeutige Partitions-GUID markiert wurde. Die Volume GUID für diese Partition lautet: `{b670dedb-96cc-4d93-ae17-9ee6ec08c6d0}`.

```
00000400: 28 73 2a c1 1f f8 d2 11 ba 4b 00 a0 c9 3e c9 3b  (s*......K...>.;
00000410: db de 70 b6 cc 96 93 4d ae 17 9e e6 ec 08 c6 d0  ..p....M........
00000420: 00 08 00 00 00 00 00 00 ff 27 03 00 00 00 00 00  .........'......
00000430: 00 00 00 00 00 00 00 80 45 00 46 00 49 00 20 00  ........E.F.I. .
00000440: 73 00 79 00 73 00 74 00 65 00 6d 00 20 00 70 00  s.y.s.t.e.m. .p.
00000450: 61 00 72 00 74 00 69 00 74 00 69 00 6f 00 6e 00  a.r.t.i.t.i.o.n.
00000460: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000470: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

Abbildung 4.6: Partitionseintrag in der GUID Partition Table. Mit Gelb wurde die eindeutige Partitions-GUID (engl. Unique Partition GUID) markiert.

\GLOBAL??\STORAGE#Volume#{[Disk Id]}#[Byte Offset]#{[Geräteklassen GUID]}

Die symbolische Verknüpfung `\GLOBAL??\STORAGE#Volume#{[Disk Id]}#[Byte Offset]#{[Geräteklassen GUID]}` besteht aus den drei Bestandteilen `Disk Id`, `Byte Offset` und `Geräteklassen GUID`.

Bei der `Disk Id` handelt es sich um eine 16 Bytes lange GUID, die die Festplatte eindeutig identifiziert. Dabei entspricht die `Disk Id` nicht der `Disk GUID` aus der symbolischen Verknüpfung `\GLOBAL??\Disk{[Disk GUID]}` oder aus der GUID Partition Table. Mithilfe der beiden Registryschlüssel `SCSI` und `STORAGE` unter `SYSTEM\ControlSet[ControlSet]\Enum` sowie dem Byte Offset kann eine Zuordnung zu einer Partition und somit zu einer Festplatte erfolgen.

Im Unterschlüssel `STORAGE\Volume` werden alle am System registrierte Partitionen (Volumes) im Format `{[Disk ID]}#[Byte Offset]` aufgeführt. Die Small Computer System Interface (kurz SCSI) Geräte, zu denen auch Festplatten zählen, werden unter `SCSI` aufgeführt. Jede Festplatten-Instanz besitzt einen Unterschlüssel mit dem Namen `Device Parameters`, der die `Disk Id` enthält. Des Weiteren existiert in dem Unterschlüssel ein binärer Wert `PartitionTableCache`, der die Partitionstabelle der Festplatte enthält und somit eine Zuordnung zu der `Volume GUID` einer Partition ermöglicht. Der Aufbau des Wertes `PartitionTableCache` ist im Anhang A.1 beschrieben.

Die dritte Komponente der symbolischen Verknüpfung ist die `Gerätekklassen GUID`. In dieser Arbeit konnten zwei GUIDs beobachtet werden. `53f5630d-b6bf-11d0-94f2-00a0c91efb8b` steht für die Geräteklasse `GUID_DEVINTERFACE_VOLUME` [49]. Sie kommt bei allen Partitionen, mit Ausnahme der `Microsoft Reserved Partition`, vor. Diese erhielt als einzige die GUID `7f108a28-9833-4b3b-b780-2c6b5fa5c062`.

Windows Partition

Die Windows Partition enthält das eigentliche Betriebssystem und ist dem Anwender in der Standardkonfiguration als Laufwerk `C:` bekannt. Im NT Objekt Manager verweisen auf die Windows Partition direkt und indirekt fünf zusätzliche symbolische Verknüpfungen.

Die Verknüpfung `\OSDataRoot` verweist auf `\Device\OSDataDevice`, die wiederum auf `\ArcName\multi(0)disk(0)rdisk([x])partition([y])` zeigt. Die drei Verknüpfungen `\Device\BootDevice`, `\Device\BootPartition` und `\GLOBAL??\BootPartition` zeigen direkt auf das Geräteobjekt.

Die Bestimmung der Windows Partition erfolgt über den Wert `SystemBootDevice` im Registryschlüssel `SYSTEM\ControlSet[ControlSet]\Control`. Dieser Wert muss mit

den symbolischen Verknüpfungen im \ArcName Verzeichnis abgeglichen werden.

Bootpartition

Die Bootpartition enthält den Bootloader, der den eigentlichen Windows Kernel startet. Eine Bootpartition wird im NT Objekt Manager Namensraum durch die beiden zusätzlichen symbolischen Verknüpfungen \Device\SystemPartition und \GLOBAL??\SystemPartition gekennzeichnet. Die Bestimmung der Bootpartition kann mithilfe der symbolischen Verknüpfung im \ArcName Verzeichnis und dem Wert FirmwareBootDevice im Registryschlüssel SYSTEM\ControlSet[Controlset]\Control erfolgen. Beide Angaben liegen im Format multi(0)disk(0)rdisk([x])partition([y]) vor.

4.3.4 PartitionTableCache-Parser.py - Proof-of-Concept

Die in den vorherigen Abschnitten genannten Möglichkeiten zur Rekonstruktion von bestimmten Objektnamen des NT Objekt Manager Namensraum, wurden im Proof-of-Concept PartitionTableCache-Parser.py implementiert. NTRebuild.py ist ein in Python 3 geschriebenes Werkzeug, dass den SYSTEM Registry Hive für die Wiederherstellung der NT Objekte von Festplatten und Partition verwendet. Für diese NT Objekte konnte der Name (FullName), der Objekttyp (TypeName) und das Ziel der symbolischen Verknüpfung (SymbolicLinkTarget) wiederhergestellt werden. Dies gelang in der Arbeit für alle NT Objekte der drei virtuellen Machine. Eine Ausnahme bildet die symbolische Verknüpfung \GLOBAL??\Disk{[DISK GUID]} für Festplatten.

4.3.5 Verwendung in einer forensischen Untersuchung

Diese Arbeit versetzt den Forensiker in die Lage, Objekte des NT Objekt Managers für Festplatten und Partitionen aus dem SYSTEM Hive wiederherzustellen, die ansonsten nur zur Laufzeit im volatilen Speicher vorhanden sind. Da der NT Objekt Manager, insbesondere das Verzeichnis GLOBAL??, eine zentrale Rolle bei der Interaktion zwischen Anwendungen und Geräten spielt, ist dieser für eine forensische Untersuchung von besonderem Nutzen. Die folgenden Anwendungsmöglichkeiten sollen kurz erläutert werden:

- Zuordnung von Pfadangaben zu einer Partition auf einer Festplatte
- Korrelation von verschiedenen Pfadformaten

Mithilfe der wiederhergestellten Informationen des NT Objekt Managers und des Registryschlüssels SYSTEM\ControlSet[ControlSet]\ENUM kann eine Pfadangabe einem konkreten Speichermedium zugeordnet werden. So lässt sich auch der Hersteller und die Produktbezeichnung des jeweiligen Speichermediums bestimmen.

Eine weitere Anwendungsmöglichkeit ist die Korrelation von verschiedenen Pfadformaten. So können die in den Artefakten gefundenen Pfadformate, ähnlich dem im Abschnitt 2.4.2 beschriebenen Vorgehen des NT Subsystems, einem Geräteobjekt zugeordnet werden. Somit kann festgestellt werden, ob die in den Artefakten dokumentierten Pfade synonym füreinander sind.

In dieser Arbeit konnte die Bedeutung und Herkunft der Disk GUID in der symbolischen Verknüpfung `\GLOBAL??\Disk{[DISK GUID]}` nicht geklärt werden. Zukünftige Arbeiten sollten sich mit den Erscheinungsformen nicht globaler NT Objekte wie USB-Speichermedien und Netzlaufwerken und deren Wiederherstellung beschäftigen. Diese könnten sich bei der Aufklärung von USB Drop oder Innentäter-Angriffen als Hilfreich erweisen. Des Weiteren sollte geklärt werden, ob es eine Möglichkeit gibt, weitere Eigenschaften der NT Objekte wie Charakteristik (Characteristics), Eigentümer (Owner), Gruppe (Group), Zugriffsrechte (MaximumGrantedAccess) oder Gerätetyp (DeviceType) aus einem Offline System zu extrahieren.

5 Proof-of-Concept

Im dritten Teil dieser Arbeit sollen die Erkenntnisse aus den Abschnitten 3 und 4 praktisch angewendet werden. Dafür wird ein forensisches Werkzeug umgesetzt, dass eine Untersuchung von, mit Plaso aus forensischen Artefakten extrahierte, Events ermöglicht. In der Detailansicht eines Events sollen ähnliche Events, anhand des Pfades einer ausführbaren Datei beziehungsweise anhand des Pfades und des Zeitstempels, gefunden und dem Forensiker vorgeschlagen werden. Der zweite Mechanismus zielt dabei darauf ab, alle Events, die aufgrund derselben Ausführung einer Anwendung entstanden sind, zu finden. Für die Suche nach ähnlichen Events anhand des Pfades der ausgeführten Datei wurde sich auf die Events aus den folgenden forensischen Artefakten beschränkt:

- Windows XML Event Log (Event-Id 4688)
- Prefetch
- Windows Timeline
- UserAssist
- Background Activity Moderator

Die Korrelation von ähnlichen Events anhand des Pfades und des Zeitstempels wird, mit Ausnahme dem UserAssit Registryschlüssel, für alle Artefakte umgesetzt. Für diesen ist eine Umsetzung derzeit nicht möglich, da er in den Experimenten in Abschnitt 3 nicht aufgefunden wurde und somit keine Daten über seine Zeitstempelverteilung vorliegen.

5.1 Implementierung

Die Implementierung des Proof-of-Concepts besteht aus drei Teile (siehe Abbildung 5.1). Das Frontend stellt die grafische Oberfläche zur Verfügung, über die Events gefiltert und betrachtet werden können. Außerdem stellt sie dem Forensiker eine Möglichkeit bereit, neue Daten hochzuladen. Das Backend beantwortet die Anfragen, die über das Frontend gestellt werden. Es ist dafür zuständig, Daten aus der Neo4J Graphdatenbank abzufragen und diese für das Frontend aufzubereiten. Das Einfügen neuer Daten in die Graphdatenbank sowie die Korrelation von Events anhand der Pfadangaben und der Ausführungsartefakte geschieht ebenfalls im Backend. Die Neo4J Datenbank speichert die Events in Form von Knoten und Kanten. In den folgenden Abschnitten werden die Komponenten des Proof-of-Concepts vorgestellt.

Abbildung 5.1: Komponenten des Proof-of-Concepts

5.1.1 Plaso (ehemals Log2Timeline)

Plaso Langar Að Safna Öllu (kurz Plaso) ist ein in Python entwickeltes Werkzeug, was die Erstellung von Zeitreihen aus forensischen Artefakten ermöglicht. Mittlerweile hat sich Plaso zu einem Framework entwickelt, welches durch Parser- und Analyse-Plugins erweitert werden kann. [29][30]

Plaso erstellt eine Zeitreihe in zwei Arbeitsschritten. Zunächst wird das Werkzeug *log2-timeline.py* verwendet, um die in den forensischen Artefakten enthaltenen Events in eine Plaso Storage File zu schreiben. Derzeit handelt es sich bei der Plaso Storage File um eine SQLite Datenbank. Zukünftig soll dafür ein Attribute Container Storage verwendet werden, der eine Verwendung von anderen Backends außer SQLite für die Storage File zulässt [28].

Im zweiten Arbeitsschritt exportiert *psort.py* die Events aus der Plaso Storage File in ein gewünschtes Format. So können die Events beispielsweise in das CSV, XLSX oder JSON Format überführt und weiter analysiert werden. Beide Arbeitsschritte können direkt nacheinander durch das Tool *psteal.py* durchgeführt werden, wobei der Forensiker auf einige Filterfunktionen verzichten muss.

In diesem Proof-of-Concept werden die durch Plaso extrahierten Events als JSON exportiert und in die Neo4j Datenbank importiert. Für die Erstellung der Plaso JSON aus einem EWF Image kann der Befehl in Listing 5.1 verwendet werden. Die Plaso Version *20211229* ist mit der Implementierung des PoCs getestet wurden.

```
1 psteal.py --storage_file storage_file.plaso --source ./ewf_image.
    e01 -o json -w plaso_json.json
```

Listing 5.1: Aufruf des forensischen Werkzeuges Plaso zum Erzeugen einer JSON Datei, die die extrahierten Events enthält.

5.1.2 Angular Frontend

Bei Angular handelt es sich um ein von Google entwickeltes Frontend-Webapplikationsframework, was die Erstellung von Single-Page-Anwendungen ermöglicht. Bei Single-Page Anwendung wird zu Beginn einer Sitzung die komplette Webseite vom Webser-

ver heruntergeladen und zur Laufzeit mit aktuellen Daten aus dem Backend versorgt, die dann zur Anzeige gebracht werden. [31, S. VII-X] Entwickelt wird in der Programmiersprache TypeScript, die auf JavaScript basiert [31, S. 27-30]. Das Frontend des Proof-of-Concepts wurde in Angular Version 14.0.0 implementiert.

Im Proof-of-Concept stellt das Angular Frontend eine Oberfläche bereit, in welcher der Forensiker die extrahierten Events hochladen, betrachten und durchsuchen kann. Die Hauptkomponente ist die, in Abbildung 5.2c zu sehende, Übersichtstabelle über die Events mit den Filter Optionen. Mithilfe dieser kann sich der Forensiker einen Überblick über die aus den Artefakten extrahierten Events verschaffen und mit den Filteroptionen interessante Events finden. Zur Zeit stehen ihm folgende Filter Optionen zur Verfügung:

Filter by message Das *Message* Feld des Events muss ein bestimmtes Schlüsselwort enthalten.

Filter since timestamp Zeigt alle Events nach einem bestimmten Zeitstempel an.

Filter due to timestamp Zeigt alle Events bis zu einem bestimmen Zeitstempel an.

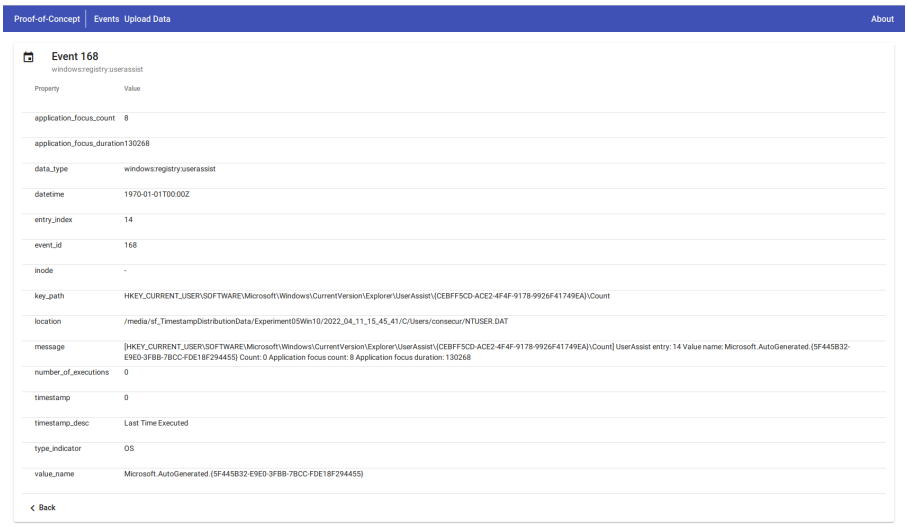
Filter by Parsers Das Event muss von einem der ausgewählten Parser extrahiert wurden sein.

Filter by Event Data Types Das Event muss einen der ausgewählten Event-Datentypen aufweisen.

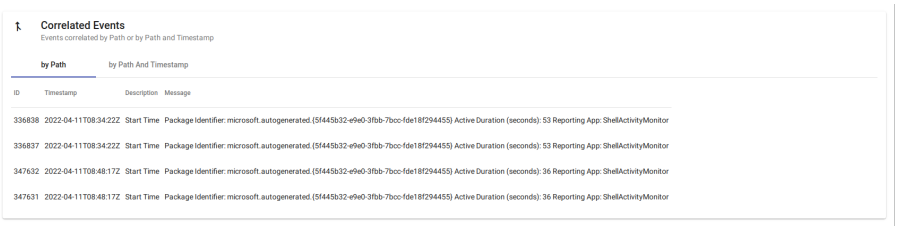
Werden mehrere Filteroptionen gleichzeitig gewählt, so werden diese durch ein logisches *UND* miteinander verknüpft. Auf die Konstruktion der Cypher-Abfrage wird im Abschnitt 5.1.3 eingegangen.

Klickt der Forensiker auf ein Event in der Übersichtstabelle, so gelangt er zu dessen Detailseite (siehe Abbildung 5.2a). Auf dieser werden alle Eigenschaften des Events präsentiert. Über die Schaltfläche „Back“ gelangt der Forensiker zurück zur Übersichtsseite. Unterhalb der Event-Eigenschaften gibt es eine weitere Tabelle, die ähnliche Events auflistet (siehe Abbildung 5.2b). Diese werden anhand der Pfadinformationen beziehungsweise der Pfadinformationen und dem Ausführungszeitpunkt bestimmt. Dieses Feature funktioniert derzeit nur für ausgewählte Event-Datentypen. Auf die Funktionsweise wird detailliert im Abschnitt 5.1.3 eingegangen.

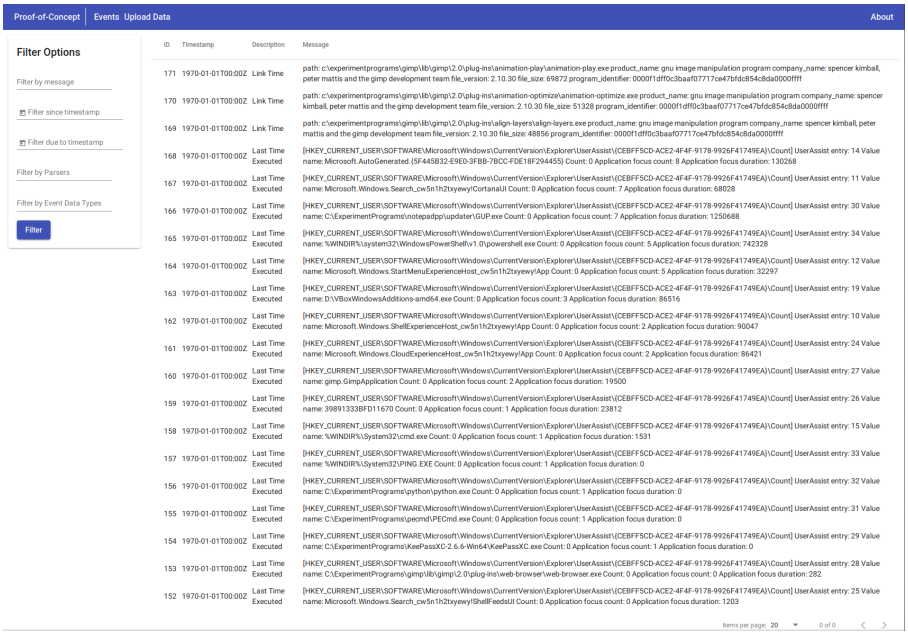
Eine weitere Komponente ermöglicht es, neue Daten hochzuladen. Nachdem der Forensiker die Plaso Events als JSON sowie den SYSTEM Registry Hive des Systems hochgeladen hat, kann er über einen Button den Import Prozess starten. Sind die Daten fertig importiert, wird er zurück zur Übersichtsseite geleitet und kann die neuen Events analysieren.



(a) Detailansicht eines Events



(b) Anhand der Pfadangabe oder der Pfadangabe und Ausführungszeit vorgeschlagenen Events



(c) Übersichtstabelle mit Filter Optionen

Abbildung 5.2: Überblick über die wichtigsten Komponenten des Frontends.

5.1.3 Flask Backend

Bei Flask handelt es sich um Webframework, das es dem Entwickler ermöglicht, dynamische Webseiten in Python zu entwickeln. Da es nur mit einer Grundfunktionalität ausgeliefert wird, spricht man bei Flask auch von einem leichtgewichtigen Webframework. Funktionen können aber über externe Python Pakete hinzugefügt werden. [63] Für die Kommunikation mit dem Webserver verwendet Flask das Web Server Gateway Interface (kurz WSGI). Dieses spezifiziert die Kommunikation zwischen dem Webserver und einer Webanwendung. [82] In dieser Arbeit wird Flask in der Version 2.1.2 verwendet.

Das Backend hat die Aufgabe, Anfragen des Frontends entgegenzunehmen und diese in Cypher-Abfragen umzuwandeln, die dann in der Neo4j Datenbank ausgeführt werden. Die Ergebnisse der Datenbankenabfrage müssen aufbereitet werden, sodass das Frontend die Informationen darstellen kann. Im Folgenden sollen die API-Endpunkte und ihre Funktion vorgestellt werden.

/api/events

Diese Route gibt standardmäßig alle Events als JSON Response zurück, die in der Datenbank vorhanden sind. Die Anzahl der zurückgegeben Events kann über die Request Arguments *page_index* und *page_size* beeinflusst werden. Diese implementieren einen Paging Mechanismus, der in der Übersichtstabelle des Frontends für das Vor- und Zurückblättern verwendet wird. *page_index* gibt dabei die Nummer der Tabellenseite an und *page_size* die Anzahl der Events pro Tabellenseite. Das Request Argument *descending* spezifiziert, ob die Einträge in der Tabelle absteigend nach Datum sortiert werden soll. Standardmäßig werden die Einträge mit dem ältesten Zeitstempel zuerst sortiert.

Die Argumente *due_date* und *after_date* ermöglichen es, Events basierten auf ihrem Zeitstempel zu filtern. Beide Argumente erwarten als Wert eine Zeichenkette, die einen ISO8601 formatierten Zeitstempel repräsentiert. Mittels *message_contains* können Events gefunden werden, die eine bestimmte Zeichenkette im Message-Feld enthalten. Die beiden Argumente *event_data_type* und *parser* können mehrfach spezifiziert werden. Sie suchen Events anhand ihrem Event-Datentyp beziehungsweise dem Parser, der ein Event extrahiert hat. Werden die beiden Argumente mehrfach angegeben, so werden dies über ein logisches ODER verknüpft. Der API-Aufruf in Listing 5.2 findet alle Events, die bis zum 19. August 2022 um 12:00 Uhr stattgefunden haben, die Zeichenkette „explorer.exe“ enthalten und entweder durch den „prefetch“ oder dem „winevtx“ Parser erzeugt wurden. Von den gefundenen Events wird nur die zweite Tabellenseite ausgegeben. Diese hat bis zu zehn Einträge. Die Cypher-Query, die durch den API-Aufruf in Listing 5.2 ausgeführt wird, ist in Listing 5.3 abgebildet.

```
/api/events?page_index=2&page_size=10&due_date=2022-08-19T12
 :00:00.00000+00:00&message_contains=explorer.exe&parser=prefetch
 &parser=winevtx
```

Listing 5.2: Beispiel für einen API-Aufruf der die Filterparameter verwendet.

```
MATCH (timestamp:Timestamp)<-[:occured_at]-(event:Event)-[:
 parsed_from]->(path:Path)
WHERE timestamp.datetime <= datetime("2022-08-19T12
 :00:00.00000+00:00")
AND ToLower(event.message) CONTAINS ToLower("explorer.exe")
WITH timestamp, event, path
MATCH (parser:Parser)<-[:parsed_by]-(event)
WHERE parser.name IN ["prefetch", "winevtx"]
RETURN timestamp, event, path
ORDER BY timestamp.timestamp
SKIP 20 LIMIT 10
```

Listing 5.3: Cypher-Query die durch den API-Aufruf in Listing 5.2 ausgeführt wird.

/api/events/total

Die Gesamtanzahl der Events beziehungsweise die Anzahl der Events, auf die ein bestimmter Filter zutrifft, kann über diese Route ermittel werden. Sie akzeptiert mit Ausnahme von *page_index* und *page_size* dieselben Request Arguments wie die Route */api/events*.

/api/events/<int:event_id>

Über diese Route kann ein einzelnes Event anhand der Event-ID zurückgegeben werden. Bei der Event-ID handelt es sich um eine numerische Zeichenkette, die für jedes Event einmalig während des Import-Vorgangs vergeben wird. Anhand dieser kann ein Event eindeutig in der Datenbank identifiziert werden. Die Cypher-Query, die beispielhaft beim Aufruf der URL */api/events/3* ausgeführt wird, ist in Listing 5.4 abgebildet.

```
MATCH (timestamp:Timestamp)<-[:occured_at]-(event:Event {event_id
 : 3})-[:parsed_from]->(path:Path)
RETURN timestamp, event, path
```

Listing 5.4: Cypher-Query die beim Aufruf der URL */api/events/3* ausgeführt wird.

/api/parsers und /api/event_data_types

Diese Routen führen die Cypher-Query in Listing 5.5 beziehungsweise 5.6 aus und geben alle Parser oder Event-Datentypen zurück, die in der Neo4j Datenbank enthalten sind. Diese Information wird für die automatische Vervollständigung der Filter-Optionen im Angular Frontend benötigt.

```
MATCH (parser:Parser)
RETURN parser.name as name
```

Listing 5.5: Cypher-Query die beim Aufruf der URL */api/parsers* ausgeführt wird.

```
MATCH (event:Event)
RETURN DISTINCT event.data_type as data_type
```

Listing 5.6: Cyper-Query die beim Aufruf der URL */api/event_data_types* ausgeführt wird.

/api/upload/<string:uri_file_type>

Für das Hochladen von neuen Events kann der API-Endpunkt */api/upload/<string:uri_file_type* verwendet werden. Da der Proof-of-Concept zwei unterschiedliche Dateien für seine Arbeit benötigt, wird zur Unterscheidung der hochgeladenen Dateien der URL-Parameter *uri_file_type* verwendet. Beim Hochladen der JSON Datei mit den Events, die durch Plaso generiert wurden, muss die Zeichenkette *plaso_json* angegeben werden. Wird der SYSTEM Registry Hive zum Server transferiert, muss *system_hive* als *uri_file_type* spezifiziert werden.

/api/insert_into_neo4j und /api/insert_into_neo4j/state

Nachdem die neuen Daten über die URL */api/upload/<string:uri_file_type>* hochgeladen wurden, kann der Import in die Neo4j Datenbank über die URL */api/insert_into_neo4j* getriggert werden. Der Status des Imports kann über */api/insert_into_neo4j/state* abgerufen werden. Der Importvorgang erfordert, aufgrund von Limitierungen in der Community Edition von Neo4j, ein Stoppen des Datenbankenservers und das vollständige Löschen der Datenbank. Anschließend werden die Events aus der Plaso JSON Datei, mithilfe des in dieser Arbeit entwickelten Python Modules *ConvertPlasoJson2Neo4jCsv*, in CSV Dateien umgewandelt. Diese CSV Dateien sind so strukturiert, dass sie durch das Neo4j Werkzeug *neo4j-admin import* in die Graphdatenbank importiert werden. Das Schema, in das die Daten in die Datenbank importiert werden, ist in Abschnitt 5.1.4 abgebildet. Nachdem der Import abgeschlossen ist, wird der Neo4j Server neugestartet.

/api/events/<int:event_id>/correlate_by_path

Diese Route findet zu einem gegebenen Event, dass die Ausführung einer Datei beschreibt, andere Events, die ebenfalls die Ausführung dieser Datei beschreiben. Derzeit funktioniert diese Funktion nur, wenn das gegebene Event einen der folgenden Event-Datentypen aufweist:

- windows:prefetch:execution
- windows:timeline:user_engaged

- windows:registry:userassist
- windows:evtx:record (und *event_identifier* gleich *4688*)
- windows:registry:bam

Eine weitere Voraussetzung ist, dass der SYSTEM Registry Hive des zu untersuchenden Systems hochgeladen wurde. Mithilfe des SYSTEM Registry Hives und der Erkenntnisse über den NT Objekt Manager und den PartitionTableCache (siehe Abschnitt 4), können sämtliche Pfadformate der Artefakte ineinander überführt werden. Die Bereitstellung der Informationen aus dem PartitionTableCache geschieht über das, in dieser Arbeit entwickelte, Python Modul *PartitionTableCache-Parser*. In den folgenden Unterabschnitten wird die Vorgehensweise bei der Suche nach weiteren Events beschrieben.

Generelle Vorgehensweise Alle unterstützten Event-Datentypen, mit Ausnahme des Datentyps *windows:prefetch:execution*, enthalten den Pfad der ausgeführten Datei. Diese liegen allerdings in unterschiedlichen Formaten vor, sodass diese zunächst normalisiert werden müssen, bevor sie weiterverwendet werden können. So können Events mit dem Datentyp *windows:registry:userassist* Umgebungsvariablen in ihrem Pfad enthalten. Bei Umgebungsvariablen handelt es sich um Zeichenketten, die durch Windows nachgeschlagen und ersetzt werden. Sie können unter anderem Teile eines Pfades beschreiben. Events mit dem Datentyp *windows:timeline:user_engaged* enthalten hingegen sogenannte Verzeichnis-GUIDS. Ähnlich wie bei Umgebungsvariablen, entspricht eine Verzeichnis-GUID einem bestimmten Teil eines Pfades. Der erste Schritt ist daher, die Verzeichnis-GUIDs und Umgebungsvariablen, die in einem Pfad enthalten sind, mithilfe einer Ersetzungstabelle einem konkreten Teilpfad zuzuordnen. Anschließend erfolgt eine Konvertierung des Pfades in das gewünschte Pfadformat. Dafür werden die Informationen des PartitionTableCache verwendet. Die so konvertierten Pfade können dann mithilfe einer Cypher-Query in der Datenbank abgefragt werden. Um *windows:prefetch:execution* Events zu finden, wird der Pfad zunächst in einen Gerätepfad umgewandelt und von diesem anschließend der SCCA2008-Hashwert gebildet. Das Listing 5.7 zeigt beispielhaft eine Cypher-Query für die Suche nach Events, welche die Ausführung der Datei *C:\ Program Files\ Mozilla\ Firefox.exe* beschreiben.

```
1 MATCH (timestamp:Timestamp)<-[:occured_at]-(event:Event)-[:
   parsed_from]->(path:Path)
2 WHERE NOT event.event_id = 12
3 AND ( event.prefetch_hash = 1711366097
4 OR toUpper(event.package_identifier) CONTAINS toUpper("{905e63b6-
   c1bf-494e-b29c-65b732d3d21a}\\Mozilla\\FIREFOX.EXE")
5 OR toUpper(event.value_name) CONTAINS toUpper("%PROGRAMFILES%\\
   Mozilla\\Firefox.exe")
6 OR ( event.event_identifier = 4688
7 AND toUpper(event.strings[5]) CONTAINS toUpper("C:\\\\Program
   Files\\\\Mozilla\\\\Firefox.exe"))
```

```
8   OR toUpper(event.binary_path) CONTAINS toUpper("C:\\Program Files
     \\Mozilla\\Firefox.exe"))
9   RETURN timestamp, event, path
10  ORDER BY timestamp.timestamp
```

Listing 5.7: language

Sonderfall windows:prefetch:execution Das Prefetch Artefakt stellt eine Besonderheit dar, da es den vollständigen Pfad zu einer ausführbaren Datei nicht enthält. Stattdessen wird nur der Name der ausführbaren Datei und ein SCCA2008-Hashwert des Pfades gespeichert. Über die Bestimmung des Gerätepfades der Form *\Device\HarddiskVolume[z]*, aus den Pfaden in den anderen Artefakten, kann der SCCA2008-Hashwert bestimmt werden. Sollen allerdings andere Events anhand des Event-Datentyp *windows:prefetch:execution* gefunden werden, so funktioniert dieser Ansatz nicht. Events mit dem Datentyp *windows:prefetch:execution* können zwar über den SCCA2008-Hashwert gefunden werden, da aber der vollständige Pfad zur ausführbaren Datei, außer über einen Bruteforce-Ansatz, nicht ermittelt werden kann, wird geschaut, ob der Name der ausführbaren Datei in den Pfaden der anderen Artefakte vorkommt. Existiert beispielsweise für die ausführbare Datei *FIREFOX.EXE* eine *windows:prefetch:execution* Event, das den SCCA2008-Hashwert 0x66015FD1 (als dezimaler Wert 1711366097) und die Event-ID 12 aufweist, so kann die Cypher-Query in Listing 5.8 weitere Events finden, die die Ausführung der Datei beschreiben.

```
1   MATCH (timestamp:Timestamp)<-[:occured_at]-(event:Event)-[:
     parsed_from]->(path:Path)
2   WHERE NOT event.event_id = 12
3   AND ( event.prefetch_hash = 1711366097
4   OR toUpper(event.package_identifier) CONTAINS toUpper("FIREFOX.
     EXE")
5   OR toUpper(event.value_name) CONTAINS toUpper("FIREFOX.EXE")
6   OR ( event.event_identifier = 4688
7   AND toUpper(event.strings[5]) CONTAINS toUpper("FIREFOX.EXE"))
8   OR toUpper(event.binary_path) CONTAINS toUpper("FIREFOX.EXE"))
9   RETURN timestamp, event, path
10  ORDER BY timestamp.timestamp
```

Listing 5.8: Beispiel einer Cypher-Query zur Suche von ähnlichen Events für ein Event mit dem Datentyp *windows:prefetch:execution*

/api/events/<int:event_id>/correlate_by_timestamp

Die URL */api/events/<int:event_id>/correlate_by_timestamp* erweitert die Suche nach ähnlichen Events um die Zeitstempel, die in den Events enthalten sind. Ziel dieser Route ist es, Events zu finden, die aus ein und derselben Dateiausführung resultieren. In Windows 1x Systemen kann der Start einer Anwendung anhand von mehreren Arte-

fakten nachvollzogen werden (siehe Abschnitt 2.2). Dabei kann es sein, dass eine Ausführung in mehreren Artefakten zeitgleich dokumentiert wird. Da, wie in Abschnitt 3 gezeigt, die Zeitstempel in den einzelnen Artefakten sich deutlich unterscheiden, soll dieser API-Endpunkt dem Forensiker eine Möglichkeit geben, Events einer einzelnen Dateiausführung einzugrenzen. Dafür werden neben den Dateipfaden in den Events, auch die dokumentierten Zeitstempel mit einbezogen. Derzeit ist dieser Ansatz für die folgenden Event-Datentypen möglich:

- windows:prefetch:execution
- windows:timeline:user_engaged
- windows:evtx:record
- windows:registry:bam

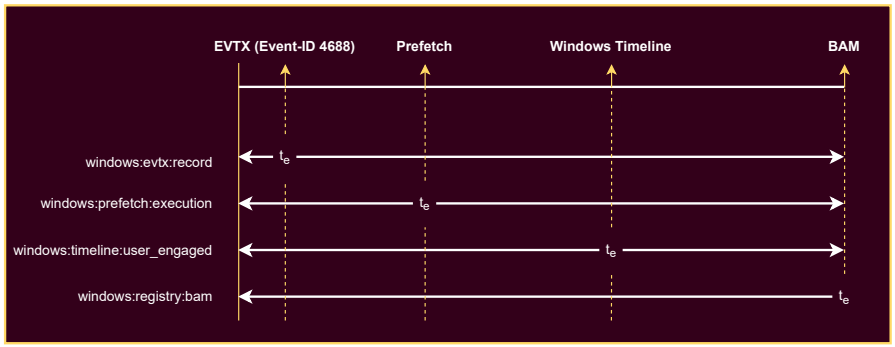


Abbildung 5.3: Skizze der Intervallberechnung

Um die Events einer Dateiausführung zu gruppieren, wird ein Suchfenster verwendet. Der Algorithmus, der zur Bestimmung des Suchintervalls genutzt wird, wurde von James und Jang [22] inspiriert. Wie in Abbildung 5.3 zu sehen, hängt die Bestimmung des Suchintervalls vom Event-Datentyp des aktuell betrachteten Events ab.

Für alle forensischen Artefakte, die in Abschnitt 3 untersucht wurden und für die eine Abweichung vom tatsächlichen Startzeitpunkt bestimmt werden konnte, wird das 0.75-Quantil betrachtet. Alle 0.75-Quantile werden in einer Liste aufsteigend sortiert. Die Reihenfolge der 0.75-Quantile der forensischen Artefakte entspricht der Folgenden:

1. Windows XML Event 4688: $Q_{0.75} = 0,15$m s
2. Prefetch: $Q_{0.75} = 42,98$ ms
3. Windows Timeline: $Q_{0.75} = 7208,60$ ms
4. Background Activity Moderator: $Q_{0.75} = 60123,45$ ms

Mithilfe des Zeitstempels des aktuell betrachteten Events und dessen Event-Datentyp, der die Art des Artefaktes angibt, kann das Suchintervall bestimmt werden. Für die unter Grenze wird das 0,75-Quantil des aktuell betrachteten Events verwendet. Diese wird von dessen Zeitstempel t_e abgezogen. Für ein EVTX 4688 Event würde sich die

untere Grenze wie folgt berechnen.

$$untere \quad Grenze = t_e - Q_{0.75}(EVTX4688)$$

Die obere Grenze des Intervalls berechnet sich ähnlich. Hierfür wird ebenfalls die Summe aus allen 0,75-Quantile, die kleiner oder gleich dem 0,75-Quantils des aktuell betrachteten Events sind, gebildet. Anschließend wird diese Summe vom 0,75-Quantil des Background Activity Moderator abgezogen. Das Ergebnis der Differenz wird mit dem Zeitstempel des aktuell betrachteten Events addiert. Eine Ausnahme bildet das Event mit dem Event-Datentyp windows:registry:bam. Dessen obere Grenze ist t_e. Für das Prefetch Event berechnet sich die obere Grenze wie folgt:

$$obere \quad Grenze = t_e + (Q_{0.75}(BAM) - (Q_{0.75}(Prefetch) + Q_{0.75}(EVTX)))$$

Mit dem berechneten Intervall kann die Cypher-Query, die die Pfadkorrelation implementiert, erweitert werden. Für ein windows:prefetch:execution Event, dass am 26. August 2022 um 12:00 Uhr GMT (als Epoch-Zeitstempel 1661515200000000) stattgefunden hat, ist die Cypher-Query in Listing 5.9 abgebildet. Die Tabelle A.3 in Anhang A.3 listet die oberen und unteren Grenzen des Intervalls für jedes Artefakt auf.

```
1  MATCH (timestamp:Timestamp)<-[:occured_at]-(event:Event)-[:
     parsed_from]->(path:Path)
2  WHERE NOT event.event_id = 12
3  AND ( event.prefetch_hash = 1711366097
4  OR toUpper(event.package_identifier) CONTAINS toUpper("FIREFOX.
     EXE")
5  OR toUpper(event.value_name) CONTAINS toUpper("FIREFOX.EXE")
6  OR ( event.event_identifier = 4688
7  AND toUpper(event.strings[5]) CONTAINS toUpper("FIREFOX.EXE"))
8  OR toUpper(event.binary_path) CONTAINS toUpper("FIREFOX.EXE"))
9  AND timestamp.timestamp >= 1661515199956870
10 AND timestamp.timestamp <= 1661515260080320
11 RETURN timestamp, event, path
12 ORDER BY timestamp.timestamp
```

Listing 5.9: Beispiel einer Cypher-Query zur Suche von ähnlichen Events für ein Event mit dem Datentyp *windows:prefetch:execution*

5.1.4 Neo4j Graphdatenbank

Bei der Neo4j Graphdatenbank handelt es sich um eine native Graphdatenbank, die das Konzept des Labled Property Graph Modells verwendet. Die Interaktion mit der Datenbank erfolgt mithilfe der eigens entwickelten Abfragesprache Cypher. [13] Neo4j steht in der selbst gehosteten Variante in zwei Editionen zur Verfügung. Die Enterprise

Edition weist eine Vielzahl von zusätzlichen Features, wie die Verwendung von mehreren Datenbanken auf einer Server-Instanz oder ein fein granuläres Rechtesystem, auf. Allerdings steht diese, im Gegensatz zu der Community Edition, nicht unter einer Open Source GPLv3 Lizenz. Für die Umsetzung des Proof-of-Concepts wurde die Neo4j Community Edition in der Version 4.4 verwendet.

Die Abbildung 5.4 zeigt die Struktur, in der die, durch Plaso, extrahierten Events als Knoten und Kanten abgebildet werden. Die Struktur ist so angelegt, dass alle Knoten, mit Ausnahme der *Event*-Knoten, einmalig in der Datenbank vorkommen. Eine Suche nach einem Event, das mit einem bestimmten Hashwertes eines Pfades in Verbindung steht, lässt sich durch eine ressourcenschonende Kantentraversierung finden. Im Gegensatz zur Bachelorarbeit „Untersuchung von forensischen Artefakten eines Windows 10 Systems im Hinblick auf Korrelationsmöglichkeiten für eine automatisierte Verknüpfung“ [19] wurde beim Entwurf dieser Graphstruktur auf die Verwendung von Kanteneigenschaften verzichtet. Alle Eigenschaften des Events sind in den Knoten oder in den Beziehungen der Knoten untereinander abgebildet. Für den Import der Events in die Graphdatenbank wurde, wie in Abschnitt 5.1.3 bereits erwähnt, das von Neo4j bereitgestellte Werkzeug *neo4j-admin import* verwendet. Dieses importiert CSV Dateien mit einem speziellen Aufbau in Neo4j. Der Aufbau der CSV Dateien beschreibt dabei die Struktur, in der die Daten in den Graphen importiert werden. [57]

5.2 Evaluation

Um die Korrelation von ähnlichen Events evaluieren zu können, wurden die in den Experimenten (siehe Abschnitt 3) generierten Daten verwendet. Für das erste und zehnte Experiment der beiden Windows 1x Systeme wurde jeweils die von Plaso generierte JSON Datei und der SYSTEM Hive der letzten Datensicherung aus den Experimenten in den Proof-of-Concept geladen. Die Zeitstempel der Artefakte, die für die Berechnung der Zeitstempelverteilung verwendet wurden, wurden genutzt, um mithilfe der Cypher-Query in Listing 5.10, die Event-Ids zu den Zeitstempeln zu bestimmen. Für jede Anwendung aus den Experimenten wurde überprüft, ob die Routen */api/events/<int:event_id>/correlate_by_path* und */api/events/<int:event_id>/correlate_by_timestamp* zu einer gegeben Event-Id die erwarteten Events finden.

```
1 MATCH (event:Event)-[:occured_at]->(timestamp:Timestamp)
2 WHERE timestamp.timestamp = 1650369538074412
3 RETURN event.event_id, event.message
```

Listing 5.10: Cypher-Query zum Bestimmen der Event-Ids anhand der bekannten Zeitstempel.

Die Korrelation der Events anhand des Pfades erfolgte, mit Ausnahme für Windows Timeline Events der Anwendungen Firefox und GIMP, zuverlässig. Diese konnten, aufgrund eines anderen Pfadformates, den anderen Events nicht zugeordnet werden. Die

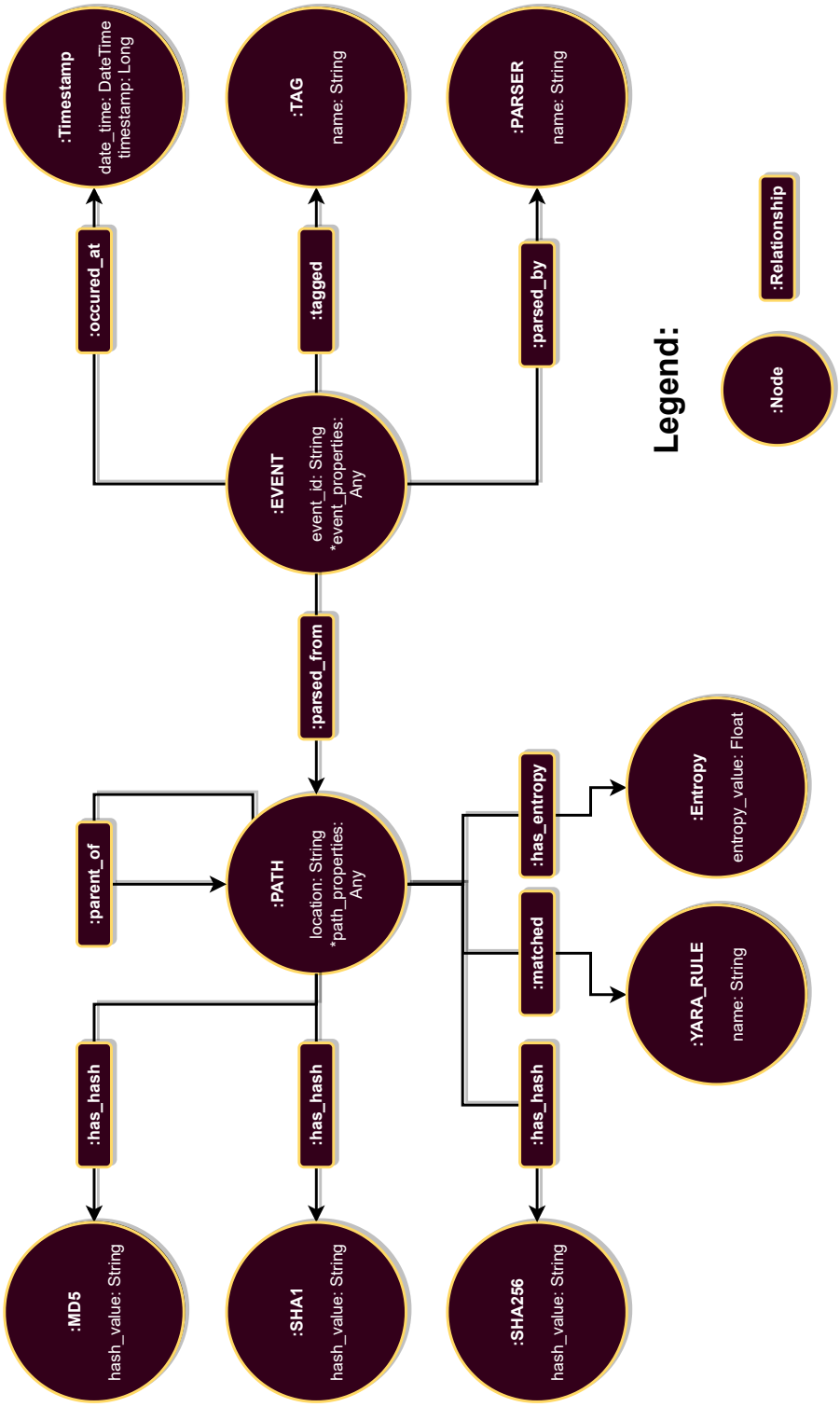


Abbildung 5.4: Die Abbildung zeigt die Graphstruktur, die in der Neo4J Graphdatenbank verwendet wird. Das Werkzeug *ConvertPlasoJson2Neo4j.py* wandelt eine JSON Datei, die durch Plaso erzeugt wurde, in mehrere CSV Dateien um. Diese werden mithilfe des Neo4j Werkzeugs *neo4j-admin* eingelesen und erzeugen die in der Abbildung gezeigte Graphstruktur.

Zuordnung von Events anhand ihres Pfades und Zeitstempels erfolgte für die folgenden

Konstellationen von Event-Datentypen zuverlässig:

- *windows:registry:bam* zu *windows:evtx:record*, *windows:prefetch:execution* und *windows:timeline:user_engaged*
- *windows:timeline:user_engaged* zu *windows:evtx:record*, *windows:prefetch:execution* und *windows:registry:bam*
- *windows:evtx:record* zu *windows:registry:bam*

In 4 von 32 Fällen funktionierte eine Zuordnung von *windows:evtx:record* zu *windows:-prefetch:execution* nicht. Umgekehrt war dies in 8 von 32 getesteten Events der Fall. Die Zuordnung von *windows:evtx:record* beziehungsweise *windows:prefetch:execution* zu *windows:timeline:user_engaged* wiesen mit 14 von 32 und 13 von 32 fehlenden Events, die höchste Fehlerquote auf. Davon können allerdings acht fehlende Zuordnungen auf das sich unterscheidende Pfadformat der Anwendungen Firefox und GIMP in der Windows Timeline zurückgeführt werden. Eine Überprüfung der Korrelation von Events, die aus dem UserAssist Registryschlüssel extrahiert wurden, konnte aufgrund des Fehlens in den Testdaten nicht durchgeführt werden.

Der Ansatz der Korrelation von Ausführungsevents anhand des Pfades der ausführbaren Datei hat sich als sehr zuverlässig herausgestellt. Bezieht man zusätzlich den Zeitstempel der Artefakte mit ein, um Events von ein und dem selben Startzeitpunkt zu finden, so kommt es zu fehlenden Zuordnungen. Eine mögliche Ursache könnte die, in Abschnitt 3 beschriebene, breite Streuung der Zeitstempel sein. Diese wird durch verschiedene Parameter des Systems beeinflusst und zum jetzigen Zeitpunkt durch den Algorithmus, der das Suchfenster berechnet, nicht beachtet. Auch wenn derzeit einige Events nicht zugeordnet werden, so bietet der Proof-of-Concept für den Forensiker einen Mehrwert. Ihm werden Ausführungsevents vorgeschlagen, die die Ausführung derselben ausführbaren Datei beschreiben. Der im Backend verwendete Algortihmus zur Normalisierung von Pfadangaben kann zukünftig auch in anderen forensischen Werkzeugen eingesetzt werden, um die Suche nach Dateien und Events zu vereinfachen.

6 Auswertung und Ausblick

Diese Arbeit beschäftigte sich mit der Korrelation von Pfadangaben und Zeitstempel der Ausführungsartefakte von Windows 1x Systemen. Das Ziel von ihr war es, die in der Bachelorarbeit „Untersuchung von forensischen Artefakten eines Windows 10 Systems im Hinblick auf Korrelationsmöglichkeiten für eine automatisierte Verknüpfung“ [19] aufgekommen Fragen hinsichtlich einer Verknüpfung von Ereignissen über Pfadangaben und Zeitstempel zu klären. So stellten Helfer, Ilic und Bodach [19] fest, dass es unter anderem einer Möglichkeit bedurfte, die in den Artefakten vorhandene Pfadangaben zu normalisieren. Des Weiteren hoben sie hervor, dass die in den Artefakten enthaltenen Events immer nur in einem zeitlichen Kontext miteinander korreliert werden können.

Die folgende zwei Fragen sollten daher in der Arbeit beantwortet werden:

- In welchen zeitlichen Kontext entstehen Events in den unterschiedlichen Artefakten, die zu ein und derselben Dateiausführung gehören und kann dieser Kontext zur Korrelation der Events verwendet werden?
- Wie verknüpft Windows die unterschiedlichen Pfadformate und gibt es eine Möglichkeit diese Verknüpfung mithilfe von nicht volatilen Daten wiederherzustellen, um diese für eine Korrelation zu verwenden?

Besonderes Augenmerk wurde in der gesamten Arbeit auf die Ausführungsartefakte von Windows 1x Systemen, namentlich Windows XML Event Log Event 4688, Prefetch-Datei, UserAssist Registryschlüssel, Windows Timeline und Background Activity Moderator, gelegt. Ihr Aufbau und forensische Aussagekraft sowie der aktuelle Stand der Forschung wurden zu Beginn dieser Arbeit beschrieben.

Die Arbeit gliederte sich in drei Bestandteile, wobei die ersten beiden die Grundlagen für den letzten Teil lieferten. Im ersten Teil wurde empirisch die Abweichung der Zeitstempel in den Ausführungsartefakten vom tatsächlichen Startzeitpunkt einer Anwendung bestimmt. Mithilfe dieser Abstände konnte die Reihenfolge bestimmt werden, in der die Artefakte erzeugt werden. Die Modellierung der Zeitstempelabweichungen, über eine stochastische Verteilung, wie sie James und Jang [22] für Browserverlaufsdaten generiert haben, war nicht möglich, da die gemessenen Zeitstempelabweichungen keiner Zufallsverteilung unterlagen. Dies ist ein Hinweis, dass die Abweichungen von deterministischen Faktoren abhängen. So wird bei Prefetch-Dateien die Abweichung vom tatsächlichen Startzeitpunkt größer, wenn eine ausführbare Datei wiederholt gestartet wird, um weitere Prozesse zu erzeugen. Dieses Verhalten konnte bei Firefox beobachtet werden, dass die firefox.exe genutzt hat, um zur Laufzeit weitere Prozesse zu erzeugen. Der Background Activity Moderator wies auffällige Häufungen um die Werte auf, die im Bereich der Ausführungsdauer lagen. Ob die Abweichung tatsächlich von der Ausfüh-

rungsdauer abhängt oder ob mit einer längeren Laufzeit als 60 Sekunden eine andere Abweichung eintritt, konnte nicht geklärt werden. Die Abweichungen der Zeitstempel in der Windows Timeline konzentrierten sich, aufgrund der Verwendung des Epoch-Zeitstempels mit einer Auflösung im Sekundenbereich, im Interval von -1000 bis 1000 Millisekunden.

In einer zukünftigen Arbeit sollte die empirische Bestimmung der Zeitstempelabweichungen fortgeführt werden. Diese bedarf eine Berücksichtigung von weiteren Parametern während der Durchführung der Experimente. So muss der Einfluss der Anwendung auf die Zeitstempelabweichung näher betrachtet werden, da es in den Daten dieser Arbeit Hinweise gibt, dass leichtgewichtige Anwendungen wie Ping oder PECmd in einigen Artefakten geringere Abweichungen erzeugen. Weitere Parameter die beleuchtet werden müssen, sind die Systemauslastung und die Hardwarekonfiguration, die in einem direkten Verhältnis zueinander stehen. Die Ausführungsdauer einer Anwendung scheint, wie bereits beschrieben, einen Einfluss auf die Zeitstempelabweichung zu haben, weshalb dieser Aspekt ebenfalls mit berücksichtigt werden muss.

Die Suche nach einer Möglichkeit, verschiedene Pfadformate miteinander zu korrelieren, wurde im zweiten Teil der Arbeit durchgeführt. Im Grundlagenteil wurde recherchiert, wie der Mechanismus der Korrelation von Pfadformaten innerhalb des Windows Betriebssystems funktioniert. Eine zentrale Rolle spielte dabei der NT Objekt Manager. Die in ihm enthaltenen NT Objekte und symbolischen Verknüpfungen wurden im zweiten Teil der Arbeit ausgewertet und anschließend deren Aufbau bestimmt. Mithilfe des PartitionTableCache Wertes im Enum Registryschlüssel gelang eine Rekonstruktion der meisten, für eine Korrelation der Pfadformate notwendigen, NT Objekte und deren symbolischen Verknüpfungen. Damit die Auswertung des PartitionTableCache und die Rekonstruktion nicht händisch durchgeführt werden muss, wurde das Python-Modul PartitionTableCache-Parser entwickelt. Dieses kann sowohl über die Kommandozeile als eigenständiges Werkzeug oder programmatisch als Python Bibliothek verwendet werden. Eine Fortführung der Arbeit sollte klären, ob es die Möglichkeit gibt die Objekte des NT Objekt Managers auch aus einer Arbeitsspeichersicherung oder Strukturen wie den Hyberfile.sys oder der Pagefile.sys wiederherzustellen. Des Weiteren könnte die Frage beantwortet werden, wie die NT Objekte für entfernbare Speichermedien oder Netzwerk Speicher wiederhergestellt werden können. Dies ist insofern wichtig, da diese in einer Vielzahl von Angriffsvektoren eine zentrale Rolle spielen.

Im dritten Teil der Arbeit wurden die Erkenntnisse aus den beiden vorherigen Teilen verwendet, um eine Korrelation der in den Artefakten enthaltenen Events über die Pfadangabe und den Zeitstempel zu demonstrieren. Für die Demonstration wurde ein Proof-of-Concept entwickelt, der durch Plaso extrahierte Events in eine Neo4j Datenbank überführt und anschließend in einer Weboberfläche ausgibt. Die Korrelation der Events erfolgt durch die Normalisierung des Pfades und der Berechnung eines Zeitstempelintervalls im Backend sowie einer anschließenden Abfrage in der Neo4j Graphdatenbank.

Angezeigt werden die korrelierten Events in der Detailansicht des Frontends. Der Forensiker erhält so die Möglichkeit, ähnliche Events, wie das aktuell Betrachtete, zu finden. In der Evaluation des Proof-of-Concepts konnte gezeigt werden, dass die Korrelation von Events nur anhand des Pfades sehr gut funktioniert. Ausnahme bildeten Anwendungen, die in den Artefakten nicht über ihren Pfad, sondern über ein anderes Bezeichnungsschema referenziert wurden. Die Korrelation von Events anhand ihres Pfades und ihres Zeitstempels funktionierte in den meisten Fällen gut. In den Fällen, in denen Events entgegen der Erwartung nicht zugeordnet werden konnten, konnte der im Backend verwendet Algorithmus zum Berechnen des Zeitstempelintervalls nicht mit Ausreißern der Zeitstempelabweichungen umgehen. Aus diesem Grund muss der Algorithmus in zukünftigen Versionen des Proof-of-Concepts angepasst werden. Ebenfalls muss untersucht werden, ob durch das Suchintervall Events fälschlicherweise miteinander korreliert werden, obwohl diese nicht aus derselben Ausführung resultieren. James und Jang [22] hoben in ihrer Arbeit hervor, dass dies vorkommen kann, falls die Zeitpunkte der tatsächlichen Events zu nah liegen, um sich unterscheiden zu können. Eine Anpassung und Erweiterung des Proof-of-Concepts sollte die Verbesserung des Datenuploads, erweiterte Filtermöglichkeiten und die Möglichkeit zur Annotation von Events durch den Forensiker enthalten.

Anhang A: Anhang

A.1 PartitionTableCache

Bei dem `PartitionTableCache` handelt es sich um einen gleichnamigen binären Wert im Registryschlüssel `SYSTEM\ControlSet[ControlSet]\Enum\SCSI\Disk&Ven_[Vendor]&Prod_[Produkt]\[Instanz Id]\Device Parameters\Partmgr`. In diesem wird der Inhalt der Partitionstabelle der Festplatte gespeichert. Das Format, in dem die Informationen im `PartitionTableCache` abgelegt werden, wird in Tabelle A.1 beschrieben.

Tabelle A.1: Aufbau des binären Wertes `PartitionTableCache` im Registryschlüssel `SYSTEM\ControlSet[ControlSet]\Enum\SCSI\Disk&Ven_[Vendor]&Prod_[Produkt]\[Instanz Id]\Device Parameters\Partmgr`

Offset	**Länge**	**Beschreibung**
PartitionTableCache Header		
0x00	0x04	0x00 ⇒ MBR, 0x01 ⇒ GPT
0x04	0x04	Anzahl der Partitionen (bei MBR immer 0x04)
	Wenn Offset 0x00 ⇒ 0x00 (MBR)	
0x08	0x04	MBR Drive Signature
0x0c	0x04	Unbekannt
0x10	0x20	Unbekannt (0x00 Padding)
0x30	0x90*(Anzahl Partitionen)	Partitionseinträge
	Wenn Offset 0x00 ⇒ 0x01 (GPT)	
0x08	0x10	Disk GUID
0x18	0x08	Unbekannt (0x4400)
0x20	0x08	Allokierbarer Speicherplatz in Bytes (Gesamter Speicherplatz - 2*(GPT Größe))
0x28	0x08	Unbekannt (0x80) evtl. maximale Anzahl Partitionseinträge Einträge
0x30	0x90*(Anzahl Partitionen)	Partitionseinträge
PartitionTableCache Partitionseintrag		
0x00	0x08	0x00 ⇒ MBR, 0x01 ⇒ GPT
0x08	0x08	Offset der Partition in Byte
0x10	0x08	Größe der Partition in Byte
0x18	0x04	Unbekannt (0xFFFFFFFF)
0x1c	0x04	Unbekannt (0x00)
	Wenn Offset 0x00 ⇒ 0x00 (MBR)	
0x20	0x01	Partitionstyp
0x21	0x01	Boot Flag

Fortsetzung auf der nächsten Seite

Tabelle A.1 – *Fortsetzung der vorherigen Seite*

Offset	Länge	Beschreibung
0x22	0x02	Unbekannt evtl. 0x00 ⇒ nicht benutzte Partition, 0x01 ⇒ benutzte Partition
0x24	0x04	Offset der Partition in LBA
0x28	0x10	Volume GUID (siehe Abschnitt 4.3.3)
0x38	0x58	Unbekannt (0x00 Padding)
	Wenn Offset 0x00 ⇒ 0x01 (GPT)	
0x20	0x10	Partition Type GUID
0x30	0x10	eindeutige Partitions-GUID/ Volume GUID (siehe Abschnitt 4.3.3)
0x40	0x08	Attribute
0x48	0x48	Partitionsname (UTF-16)

A.2 Zusätzliche statistische Werte aus den Experimenten

Die Tabelle A.2 enthält die statistischen Werte der Experimente aus Abschnitt 3 und führt diese getrennt nach Windows System auf.

A.3 Proof-of-Concept - Intervall Grenzen der Artefakte

Die Tabelle A.3 zeigt die Werte, die für die Berechnung des Suchfensters im Proof-of-Concept verwendet werden.

A.4 Inhalt der beiliegenden DVD

Dieser Arbeit liegt in der gedruckten Fassung eine DVD bei. Diese enthält:

- die Masterarbeit als PDF,
- das PowerShell Module *TimestampDistribution*, dass zum Erzeugen der Zeitstempel im Abschnitt 3 verwendet wurde,
- das Python-Module *PartitionTableCache-Parser* in der Version 1.0.0,
- das Python-Module *MbrGptParser* in der Version 0.2.3 und
- den Proof-of-Concept als Docker Version.

Artefakt	Betriebssystem	Ausführungszeit	$\bar{X} \pm s$			$\tilde{X}_{0,25}$	$\tilde{X}_{0,50}$	$\tilde{X}_{0,75}$	$\tilde{X}_{0,90}$	X_{min}	X_{max}	# Zeitstempel
Prefetch	Windows 10	30s	1699,75	±	4879,05	3,94	11,37	25,85	9405,01	−6,38	21011,40	80
		60s	1901,00	±	5439,40	15,33	32,17	70,22	8399,54	−4,03	24167,29	77
	Windows 11	30s	1814,31	±	5062,60	3,98	9,26	22,88	9583,43	−5,08	23013,27	80
		60s	2640,57	±	7257,40	8,51	16,59	35,36	15209,72	−5,39	32803,51	79
EVTX 4688	Windows 10	30s	0,19	±	0,83	0,07	0,08	0,11	0,15	0,05	7,50	80
		60s	0,24	±	0,98	0,07	0,09	0,15	0,25	0,05	8,78	80
	Windows 11	30s	0,19	±	0,41	0,10	0,11	0,14	0,32	0,05	3,62	80
		60s	0,41	±	1,45	0,10	0,13	0,18	0,30	0,05	11,71	80
BAM	Windows 10	30s	29399,55	±	10954,69	27037,69	30146,25	34366,51	40374,06	3367,06	65021,02	80
		60s	56702,37	±	18582,23	60049,78	60220,12	65188,55	73469,28	4587,12	92176,09	79
	Windows 11	30s	27205,26	±	8514,15	30023,32	30051,44	30131,25	31252,72	3699,29	39902,23	80
		60s	52950,85	±	20259,69	60030,79	60088,14	61034,56	66718,32	187,15	72290,02	80
Windows Timeline	Windows 10	30s	3890,53	±	5144,75	6,70	1356,05	6071,87	10946,89	−617,99	21823,62	60
		60s	7857,93	±	12055,97	257,57	3693,88	9282,26	21284,86	−694,50	51898,02	60
	Windows 11	30s	3089,49	±	3802.54	20,62	1892,79	4254,04	8378,67	−851,05	14877,71	67
		60s	4733,29	±	6088,69	243,19	2180,72	7062,32	12801,72	−695,59	24877,72	66

Tabelle A.2: Die Tabelle zeigt statistische Werte der in den Experimenten gemessenen Zeitstempelverteilungen. Alle Werte wurden auf zwei Stellen nach dem Komma gerundet. Es werden keine Werte für das UserAssist Artefakt aufgeführt, da diese in den Experimenten nicht beobachtet werden konnten.

Artefakt	untere Grenze	obere Grenze
EVTX 4688	$t_e - 150$	$t_e + 60123300$
Prefetch	$t_e - 42980$	$t_e + 60080320$
Windows Timeline	$t_e - 7208600$	$t_e + 52871720$
BAM	$t_e - 60123450$	t_e

Tabelle A.3: Die Tabelle zeigt die Werte, die zur Berechnung des Suchintervalls im Proof-of-Concept verwendet werden. Alle Werte wurde in Mikrosekunden angegeben.

Literatur

[1] AccessData Group, Inc. *FTK Imager*. Version 4.5.0.3. 8. Okt. 2020. URL: https://accessdata.com/product-download/ftk-imager-version-4-5 (besucht am 15. 02. 2022).

[2] F. Amato, G. Cozzolino, A. Mazzeo und N. Mazzocca. „Correlation of Digital Evidences in Forensic Investigation through Semantic Technologies“. In: *2017 31st International Conference on Advanced Information Networking and Applications Workshops (WAINA)*. 2017, S. 668–673. DOI: 10.1109/WAINA.2017.4.

[3] ArchLinux Wiki. *System Time*. URL: https://wiki.archlinux.org/index.php/System_time (besucht am 07. 07. 2022).

[4] Bundesamt für Sicherheit in der Informationstechnik (BSI). *Die Lage der IT-Sicherheit in Deutschland 2021*. Forschungsber. 1. Sep. 2021. URL: https://www.bsi.bund.de/DE/Service-Navi/Publikationen/Lagebericht/lagebericht_node.html (besucht am 07. 09. 2022).

[5] Physikalisch-Technische Bundesanstalt. *Neue Definition im Internationalen Einheitensystem (SI)*. URL: https://www.ptb.de/cms/fileadmin/internet/forschung_entwicklung/countdown_new_si/Lesezeichen_zum_neuen_SI.pdf (besucht am 18. 07. 2022).

[6] Brian Carrier. *The Sleuth Kit*. Version 4.10.2. 4. Jan. 2020. URL: https://www.sleuthkit.org (besucht am 07. 07. 2022).

[7] Yoan Chabot, Aurélie Bertaux, Christophe Nicolle und Tahar Kechadi. „An ontology-based approach for the reconstruction and analysis of digital incidents timelines“. In: *Digital Investigation* 15 (Dez. 2015), S. 83–100. DOI: 10.1016/j.diin.2015.07.005.

[8] Brandon Charter. „EVTX and Windows Event Logging“. In: *Sans Insitute Information Security Reading Room* (13. Nov. 2008).

[9] Chris Cortes. *Battery awareness and background activity*. 1. Aug. 2016. URL: https://blogs.windows.com/windowsdeveloper/2016/08/01/battery-awareness-and-background-activity/ (besucht am 01. 07. 2022).

[10] Richard Davis. *SANS DFIR Cheat Sheet*. URL: https://www.13cubed.com/downloads/dfir_cheat_sheet.pdf (besucht am 30. 06. 2022).

[11] Quang Do, Ben Martini, Jonathan Looi, Yu Wang und Kim-Kwang Choo. „Windows Event Forensic Process“. In: *Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications*. Springer International Publishing, 2014, S. 87–100. DOI: 10.1007/978-3-662-44952-3_7.

[12] Microsoft Docs. *Get-Process*. URL: https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.management/get-process?view=powershell-7.1 (besucht am 30. 06. 2022).

[13] Stefan Edlich, Achim Friedland, Jens Hampe, Benjamin Brauer und Markus Brückner. *NoSQL - Einstieg in die Welt nichtrelationaler Web 2.0 Datenbanken*. 2., aktualisierte und erweiterte Auflage. Carl Hanser Verlag München, 2011. Kap. 6. Graphdatenbanken, S. 207–328. ISBN: 978-3-446-42855-3.

[14] Google LCC. *sandbox-attacksurface-analysis-tools*. 18. Aug. 2021. URL: https://github.com/googleprojectzero/sandbox-attacksurface-analysis-tools (besucht am 30. 06. 2022).

[15] Aric A. Hagberg, Daniel A. Schult und Pieter J. Swart. „Exploring Network Structure, Dynamics, and Function using NetworkX“. In: *Proceedings of the 7th Python in Science Conference*. Hrsg. von Gaël Varoquaux, Travis Vaught und Jarrod Millman. Pasadena, CA USA, 2008, S. 11–15. URL: https://networkx.org/ (besucht am 07. 07. 2022).

[16] Gavin Hales. „Visualisation of device datasets to assist digital forensic investigation“. In: *2017 International Conference On Cyber Situational Awareness, Data Analytics And Assessment (Cyber SA)*. IEEE, Juni 2017. DOI: 10.1109/cybersa.2017.8073402.

[17] Nihad A. Hassan. *Digital Forensics Basics*. Apress, 2019. DOI: 10.1007/978-1-4842-3838-7.

[18] Dominic Helfer. *MbrGptParser.py*. Version 0.2.3. URL: https://gitlab.consecur.de/consecur-cdc/dfir/mbr-gpt-parser (besucht am 07. 07. 2022).

[19] Dominic Helfer, Stephan Ilic und Prof. Ronny Bodach. „Untersuchung von forensischen Artefakten eines Windows 10 Systems im Hinblick auf Korrelationsmöglichkeiten für eine automatisierte Verknüpfung“. Bachelor. Hochschule Mittweida, 17. Aug. 2020.

[20] Don Ho. *Notepad++ 8.3.2 (Declare variables, not war)*. Version 8.3.2. 27. Feb. 2022. URL: https://notepad-plus-plus.org/downloads/v8.3.2/ (besucht am 03. 05. 2022).

[21] Michael Hörz. *HxD Hex Editor*. Version 2.5.0.0. 2021. URL: https://mh-nexus.de/en/hxd/ (besucht am 15. 06. 2022).

[22] Joshua I. James und Yunsik Jang. „Update Thresholds of More Accurate Time Stamp for Event Reconstruction“. In: *Journal of the Korean Society for Internet, Broadcasting and Communication* (30. Apr. 2017). DOI: 10.7236/JIIBC.2017.17.2.7.

[23] Costas Katasavounidis. *Windows 10 Timeline - Windows 10 (v1803+) Activities-Cahce.db parsers (SQlite, PowerShell, .EXE)*. URL: https://kacos2000.github.io/WindowsTimeline/ (besucht am 18. 07. 2022).

[24] Costa Katsavounidis. *An Alternativ to Prefetch -> BAM*. 26. Feb. 2018. URL: https://www.linkedin.com/pulse/alternative-prefetch-bam-costas-katsavounidis/ (besucht am 28. 06. 2022).

[25] Costas Katsavounidis. *Windows 10 1803 Timeline (powerpoint)*. Nov. 2018.

[26] KeePassXC Team. *KeePassXC - Cross-Platform Password Manager*. Software. Version 2.6.6. 12. Juni 2021. URL: https://github.com/keepassxreboot/keepassxc/releases/tag/2.6.6 (besucht am 11. 06. 2022).

[27] Leslie Lamport. „Time, clocks, and the ordering of events in a distributed system“. In: *Communications of the ACM* 21.7 (Juli 1978), S. 558–565. DOI: 10.1145/359545.359563.

[28] Log2Timeline. *ACStore*. 6. Aug. 2022. URL: https://github.com/log2timeline/acstore (besucht am 23. 08. 2022).

[29] Log2Timeline. *Plaso (Plaso Langar Að Safna Öllu) - super timeline all the things*. Version 20211229. 29. Dez. 2021. URL: https://github.com/log2timeline/plaso (besucht am 28. 06. 2022).

[30] Log2Timeline. *Welcome to the Plaso documentation*. Version 20211229. 29. Dez. 2021. URL: https://plaso.readthedocs.io/en/latest/index.html (besucht am 22. 04. 2022).

[31] Ferdinand Malcher, Johannes Hoppe und Danny Koppenhagen. *Angular*. Dpunkt.Verlag GmbH, 15. Okt. 2020. ISBN: 3864907799. URL: https://www.ebook.de/de/product/39324310/ferdinand_malcher_johannes_hoppe_danny_koppenhagen_angular.html.

[32] Vico Marziale. *Exploring the Windows Activity Timeline, Part 1: The High Points*. 20. Juli 2020. URL: https://www.cellebrite.com/en/blog/exploring-the-windows-activity-timeline-part-1-the-high-points/ (besucht am 28. 06. 2022).

[33] Vico Marziale. *Exploring the Windows Activity Timeline, Part 2: Syncing Across Devices*. 30. Juli 2020. URL: https://www.cellebrite.com/en/blog/exploring-the-windows-activity-timeline-part-2-syncing-across-devices/ (besucht am 28. 06. 2022).

[34] Vico Marziale. *Exploring the Windows Activity Timeline, Part 3: The Value of Clipboard Content*. 5. Aug. 2020. URL: https://www.cellebrite.com/en/blog/exploring-the-windows-activity-timeline-part-3-the-value-of-clipboard-content/ (besucht am 28. 06. 2022).

[35] Jamie McQuaid. *Forensic Analysis of Prefetch files in Windows*. Aug. 2014. URL: https://www.magnetforensics.com/blog/forensic-analysis-of-prefetch-files-in-windows/ (besucht am 07. 07. 2022).

[36] Merriam-Webster Dictionary. *Gregorian calendar (noun) - Definition of Gregorian calendar*. URL: https://www.merriam-webster.com/dictionary/Gregorian%20calendar (besucht am 18. 07. 2022).

[37] Merriam-Webster Dictionary. *second (noun (2)) - Definition of second (Entry 4 of 5)*. URL: https://www.merriam-webster.com/dictionary/second (besucht am 18. 07. 2022).

[38] Merriam-Webster Dictionary. *time (noun) - Definition of time (Entry 1 of 3)*. URL: https://www.merriam-webster.com/dictionary/time (besucht am 18. 07. 2022).

[39] Joachim Metz. *Windows Prefetch File (PF) format*. 2020. URL: https://github.com/libyal/libscca/blob/master/documentation/Windows%20Prefetch%20File%20(PF)%20format.asciidoc (besucht am 29. 06. 2022).

[40] Microsoft. *Desktop Activity Moderator*. 7. Nov. 2018. URL: https://docs.microsoft.com/en-us/windows/compatibility/desktop-activity-moderator (besucht am 01. 07. 2022).

[41] Microsoft. *Disabling Prefetch*. URL: https://docs.microsoft.com/en-us/previous-versions/windows/embedded/ms940847(v=winembedded.5)?redirectedfrom=MSDN (besucht am 07. 07. 2022).

[42] Microsoft. *Hilfe zur Chronik*. 21. Mai 2020. URL: https://support.microsoft.com/de-de/help/4230676/windows-10-get-help-with-timeline (besucht am 07. 07. 2022).

[43] Microsoft. *Microsoft Documentation - FILETIME*. URL: https://docs.microsoft.com/de-de/office/client-developer/outlook/mapi/filetime (besucht am 18. 07. 2022).

[44] Microsoft Corpoeration. *Q102873: BOOT.INI and ARC Path Naming Conventions and Usage*. 28. Feb. 2002. URL: https://jeffpar.github.io/kbarchive/kb/102/Q102873/ (besucht am 07. 07. 2022).

[45] Microsoft Docs. *4688(S): A new process has been created.* 19. Apr. 2017. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4688 (besucht am 07. 07. 2022).

[46] Microsoft Docs. *4689(S): A process has exited.* 19. Apr. 2017. URL: https://docs.microsoft.com/en-us/windows/security/threat-protection/auditing/event-4689 (besucht am 07. 07. 2022).

[47] Microsoft Docs. *Bekannte Sicherheits-IDs in Windows Betriebssystemen*. 8. Sep. 2020. URL: https://docs.microsoft.com/de-de/troubleshoot/windows-server/identity/security-identifiers-in-windows (besucht am 28. 06. 2022).

[48] Microsoft Docs. *Formate von Dateipfaden*. 6. Juni 2019. URL: https://docs.microsoft.com/de-de/dotnet/standard/io/file-path-formats (besucht am 22. 06. 2022).

[49] Microsoft Docs. *GUID_DEVINTERFACE_VOLUME*. 17. Okt. 2018. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/install/guid-devinterface-volume (besucht am 07. 07. 2022).

[50] Microsoft Docs. *Introduction to MS-DOS Device Names*. 16. Juni 2017. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/introduction-to-ms-dos-device-names (besucht am 07. 07. 2022).

[51] Microsoft Docs. *IoAssignArcName macro (ntddk.h)*. 30. Apr. 2021. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/ddi/ntddk/nf-ntddk-ioassignarcname (besucht am 07. 07. 2022).

[52] Microsoft Docs. *NT Device Names*. 20. Apr. 2017. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/nt-device-names (besucht am 07. 07. 2022).

[53] Microsoft Docs. *Object Directories*. 16. Juni 2017. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/kernel/object-directories (besucht am 07. 07. 2022).

[54] Microsoft Docs. *Supporting Mount Manager Requests in a Storage Class Driver*. 20. Apr. 2017. URL: https://docs.microsoft.com/en-us/windows-hardware/drivers/storage/supporting-mount-manager-requests-in-a-storage-class-driver (besucht am 07. 07. 2022).

[55] Microsoft Docs. *UEFI-/GPT-basierte Festplattenpartitionen*. 13. Mai 2021. URL: https://docs.microsoft.com/de-de/windows-hardware/manufacture/desktop/configure-uefigpt-based-hard-drive-partitions (besucht am 07. 07. 2022).

[56] Microsoft Support. *How to use Ping.exe to check <our Microsoft Broadband Network Adapter*. 20. Aug. 2020. URL: https://support.microsoft.com/en-us/help/814155/how-to-use-ping-exe-to-check-your-microsoft-broadband-network-adapter (besucht am 07. 07. 2022).

[57] Neo4j Docs. *Operations Manual - Neo4j Admin import*. Version 4.4. 2022. URL: https://neo4j.com/docs/operations-manual/4.4/tutorial/neo4j-admin-import/ (besucht am 23. 08. 2022).

[58] Sara Newcomer. *Analyzing Programm Execution Windows Artifacts*. 10. Feb. 2020. URL: https://www.blackbagtech.com/blog/analyzing-program-execution-windows-artifacts/ (besucht am 07. 07. 2022).

[59] Walter Oney. *Programming the Microsoft Windows Driver Model*. In: Redmond, Wash: Microsoft Press, 1999. Kap. 1 - Introduction. ISBN: 0735605882.

[60] Walter Oney. *Programming the Microsoft Windows Driver Model*. In: Redmond, Wash: Microsoft Press, 1999. Kap. 2 - Basic Structure of a WDM Driver. ISBN: 0735605882.

[61] opentext. *Windows Drive Letter Assignments*. URL: https://security.opentext.com/appDetails/Windows-Drive-Letter-Assignments (besucht am 07. 07. 2022).

[62] Oracle. *Oracle Virtual Box 6.1*. Version 6.1.32r149290. 2021. URL: https://www.virtualbox.org (besucht am 07. 07. 2022).

[63] Pallets Projects. *Flask*. 5. Aug. 2022. URL: https://flask.palletsprojects.com/en/2.2.x/ (besucht am 24. 08. 2022).

[64] Sriram Raghavan. „Digital forensic research: current state of the art“. In: *CSI Transactions on ICT* 1.1 (Nov. 2012), S. 91–114. DOI: doi.org/10.1007/s40012-012-0008-7.

[65] Robert Rowlingson. „A ten step process for forensic readiness“. In: *International Journal of Digital Evidence* 2.3 (2004), S. 1–28.

[66] Mark Russinovich. „Inside NT's Object Manager“. In: *ITProToday* (30. Sep. 1997). URL: https://www.itprotoday.com/compute-engines/inside-nts-object-manager (besucht am 07. 07. 2022).

[67] Bradley Schatz, George Mohay und Andrew Clark. „A correlation method for establishing provenance of timestamps in digital evidence“. In: *Digital Investigation* 3 (Sep. 2006), S. 98–107. DOI: 10.1016/j.diin.2006.06.009.

[68] Andreas Schuster. „Introducing the Microsoft Vista event log file format“. In: *Digital Investigation* 4 (Sep. 2007), S. 65–72. DOI: 10.1016/j.diin.2007.06.015.

[69] N. Shashidhar und D. Novak. „Digital Forensic Analysis on Prefetch Files Exploring the Forensic Potential of Prefetch Files in the Windows Platform“. In: 2015.

[70] Bhupendra Singh und Upasna Singh. „Program Execution Analysis using UserAssist Key in Modern Windows“. In: *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS - Science und Technology Publications, 2017. DOI: 10.5220/0006416704200429.

[71] Jong Hwa Song, Se Ho Kim, Song Yi Hwang, Seung gyu Kim und Sung Jin Lee. „A study on the APFS timestamps in MACOS“. In: *International journal of engineering and technology* 7 (2018), S. 133.

[72] Andreas Streim und Simran Mann. *203 Milliarden Euro Schaden pro Jahr durch Angriffe auf deutsche Unternehmen*. 31. Aug. 2022. URL: https://www.bitkom.org/Presse/Presseinformation/Wirtschaftsschutz-2022 (besucht am 07. 09. 2022).

[73] John Tan. „Forensic readiness“. In: *Cambridge, MA:@ Stake* (2001), S. 1–23.

[74] The pandas development team. *pandas-dev/pandas: Pandas*. Version 1.3.1. Feb. 2020. DOI: 10.5281/zenodo.3509134. URL: https://doi.org/10.5281/zenodo.5136416 (besucht am 07. 07. 2022).

[75] The GIMP Team. *GIMP - GNU Image Manipulation Program*. URL: https://www.gimp.org/ (besucht am 30. 06. 2022).

[76] Wikipedia. *ISO 8601*. 10. Mai 2020. URL: https://de.wikipedia.org/wiki/ISO_8601 (besucht am 18. 07. 2022).

[77] Wikipedia. *Koodinierte Weltzeit*. URL: https://de.wikipedia.org/wiki/Koordinierte_Weltzeit (besucht am 18. 07. 2022).

[78] Wikipedia. *Mozilla Firefox*. 5. Okt. 2020. URL: https://de.wikipedia.org/wiki/Mozilla_Firefox (besucht am 01. 07. 2022).

[79] Wikipedia. *Uhr*. URL: https://de.wikipedia.org/wiki/Uhr (besucht am 18. 07. 2022).

[80] Wikipedia. *Zeit*. URL: https://de.wikipedia.org/wiki/Zeit (besucht am 18. 07. 2022).

[81] Svein Willassen. „Hypothesis-Based Investigation of Digital Timestamps“. In: *Advances in Digital Forensics IV*. Hrsg. von Indrajit Ray und Sujeet Shenoi. Boston, MA: Springer US, 2008, S. 75–86. ISBN: 978-0-387-84927-0.

[82] WSGI.org. *What is WSGI?* 19. Aug. 2011. URL: https://wsgi.readthedocs.io/en/latest/what.html (besucht am 24. 08. 2022).

[83] Eric Zimmerman. *Eric Zimmerman's Tools*. github.io. URL: https://ericzimmerman.github.io/#!index.md (besucht am 29. 06. 2022).

[84] Eric Zimmerman. *Kape Documentation - What is KAPE*. Version 1.2.0.0. 10. März 2022. URL: https://ericzimmerman.github.io/KapeDocs/#!index.md (besucht am 22. 04. 2022).

[85] Eric Zimmerman. *Registry Explorer/RECmd*. Version 1.6.0.0. 2021. URL: https://ericzimmerman.github.io/ (besucht am 07. 07. 2022).

Nutzungs- und Verwertungsrecht

Ich übertrage zusätzliche Nutzungs- und Verwertungsrechte für die vorliegende Arbeit und allen damit in Zusammenhang stehenden Daten auf Grundlage der *Creative Commons License „CC0“* an alle genannten Betreuer der Arbeit.

Mittweida, 16. September 2022

Eidesstattliche Versicherung

Hiermit versichere ich an Eides statt, dass ich meine Arbeit selbstständig verfasst, keine anderen als die angegebenen Quellen und Hilfsmittel benutzt und die Arbeit noch nicht anderweitig für Prüfungszwecke vorgelegt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Mittweida, 16. September 2022