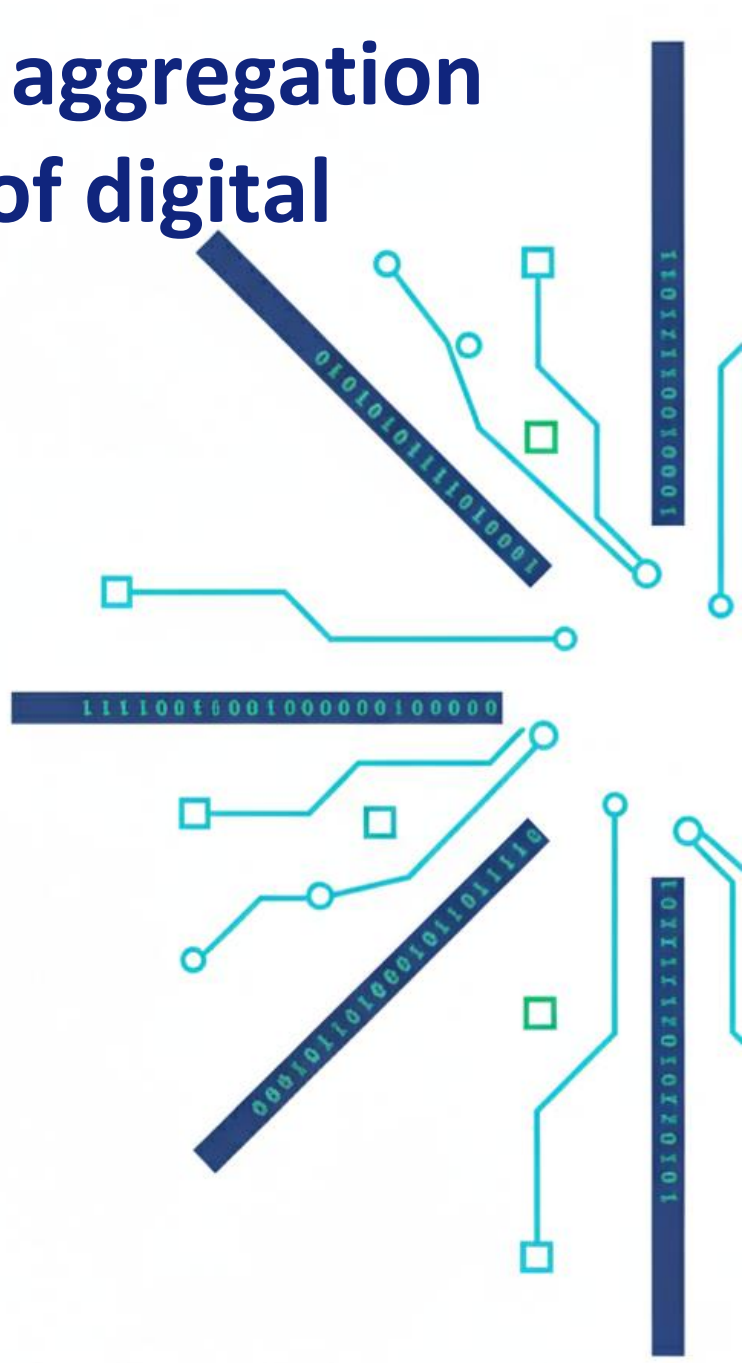
D17 – Model for aggregation and connection of digital evidence



The project Automatization of digital forensics and incident response (ADFIR) funded by the European Union – Next GenerationEU through the Recovery and Resilience Plan of the Slovak Republic under project No. č. 09-I05-03-V02-00079.

CONTENTS

1 Project description

The project **Automatization of digital forensics and incident response** (hereinafter referred to as **"ADFIR"**) is funded by the **European Union – Next GenerationEU through the Recovery and Resilience Plan of the Slovak Republic** under project No. č. 09-I05-03-V02-00079. This project addresses one of the key challenges in cybersecurity and information security – how to process the massive volume of digital evidence generated during cybersecurity incidents or forensic investigations. Currently, this process is highly demanding in terms of human resources and time. Therefore, automation using machine learning methods can significantly **improve the quality of digital forensic analysis** and reduce the time required to perform it. Overall, this enables security teams to respond more effectively to cyber threats. Main benefits of this project are:

- **Accelerated Resolution of Cybersecurity Incidents**. The project ADFIR introduces automated approaches to collecting, processing, and analyzing digital evidence. As a result, security teams can identify the causes of incidents more quickly and adopt effective measures to address them.
- **Reduced Workload for Forensic Analysts**. Routine and time-consuming tasks involved in processing digital evidence will be replaced by automated methods. This will allow analysts to focus on more complex cases and strategic decision-making.
- **Higher Quality and Consistency of Outputs**. The use of unified methodologies and tools ensures that the processed digital evidence will be more accurate, consistent, and easily verifiable. This significantly reduces the risk of errors caused by human factors.
- **Potential Use in Criminal Proceedings.** The project outputs will be developed in compliance with legal requirements and standards, allowing the digital evidence to be accepted as relevant evidence for investigations and court proceedings.

2 Introduction

In the field of digital forensic analysis, the effective processing of large amounts of heterogeneous data presents a major challenge, particularly when reconstructing events and identifying causal links between individual artefacts. Modern approaches therefore increasingly utilise concepts of data aggregation and data linking, which enable the transformation of isolated forensic evidence into a model usable by analysts.

This document focuses on the design and implementation of a model for the aggregation and linking of forensic evidence within the ADFIR project (**Task KPB3.3 – Aggregation and connection (data fusion) of digital evidence**). The aim is to create a model that enables the systematic processing of artefacts, the reduction of the volume of records, their correlation across different sources, and subsequent interpretation within the context of identified system or user behaviour scenarios.

A key part of the work is the process of linking forensic evidence based on common identifiers, temporal relationships and contextual links. This approach enables identified events to be corroborated using multiple artefacts whilst reducing the volume of individual records that the analyst must process. Subsequently, these linked traces are aggregated into defined time windows, thereby creating a clear timeline of events and enabling more effective analysis of the dynamics of monitored processes and changes.

The proposed model thus provides a systematic way to move from fragmented data to an integrated view of digital evidence, supporting not only the reconstruction of events but also the identification of behavioural patterns relevant to forensic analysis.

3 Aggregation of forensic artefacts

This chapter focuses on the systematic processing of digital forensic artefacts with the aim of transforming them into a form suitable for analysis. It covers the entire process step by step, from the pre-processing of heterogeneous data, through the identification of behavioural scenarios, to the linking and aggregation of digital evidence. Emphasis is placed on creating a unified representation of events through metadata records, which enable more effective interpretation and correlation of data.

3.1 Pre-processing of artefacts

In the processing of forensic data, multiple sources of artefacts were utilised, differing in their structure, format and level of detail. This data was predominantly represented in tabular form (e.g. CSV outputs from tools), with individual artefacts containing different sets of attributes and differing semantic meanings. Before further processing, it was therefore necessary to carry out a pre-processing phase aimed at unifying the data representation, selecting relevant attributes and removing inconsistencies. We addressed the processing of this data in more detail in deliverable D15 – Model for digital evidence extraction to matrix representation. In this deliverable, we focused on two methods of pre-processing digital forensic artefacts.

Firstly, we used the "Szechuan Sauce" input dataset from the DFIR Madnes portal[1,2] , which was processed using selected tools by Eric Zimmerman [1]. We extracted digital evidence from disk images using the following tools:

- EvtxECmd [2],
- MFTECmd [3],
- RECmd [4],
- PECmd [5],
- AppCompatCacheParser [6],
- JLECmd [7].

The output of the processing was structured CSV files representing individual forensic artefacts. As part of further processing, relevant attributes were extracted from these artefacts, which serve as the basis for subsequent data aggregation.

The "Szechuan Sauce" dataset, together with four other datasets (Output D14 – Synthetic dataset), was also used in the design of aggregation functions for data with missing values. The input data was obtained using the Plaso tool, with the acquired data first undergoing pre-

[1] https://dfirmadness.com/case001/DC01-E01.zip
[2] https://dfirmadness.com/case001/DESKTOP-E01.zip

processing in accordance with Output D15 – Model for digital evidence extraction to matrix representation. The output is a single CSV file for each dataset.

3.2 Identification of behavioural scenarios

As part of the design of the model for aggregating and linking forensic traces, a set of representative behavioural scenarios was defined, capturing typical and forensically relevant activities within the analysed system. These behavioural scenarios represent abstracted higher-level events that can be described using multiple low-level artefacts.

For the purposes of this document, we refer to these abstracted events as meta-records. A **meta-record** represents a logically coherent unit of information created by linking multiple forensic traces based on temporal, identity or contextual relationships. The aim of meta-records is to simplify data interpretation and enable more effective detection and reconstruction of incidents.

The process of defining meta-records involved identifying relevant behavioural scenarios (e.g. authentication, remote access, file manipulation or persistence), to which forensic artefacts capable of indicating the occurrence of a given event were subsequently assigned. For each metadata record, the following sections of this chapter will specify the source artefacts from which the event can be detected and the key attributes extracted from these artefacts. Some of the attributes, such as parameters identifying specific source artefacts for the metadata record, have been added to the individual artefacts.

Based on this methodology, the following metadata records have been defined:

- **Local login**

Attribute	Data type	Source artefact
EventRecordId	int	Event Logs
TimeCreated	datetime	Event Logs
EventId	int	Event Logs
Provider	string	Event Logs
ProcessId	int	Event Logs
ThreadId	int	Event Logs
Computer	string	Event Logs
ChunkNumber	int	Event Logs
MapDescription	string	Event Logs
UserName	string	Event Logs
RemoteHost	string	Event Logs
Target	string	Event Logs
Logon_Type	int	Event Logs
Logon_Id	int	Event Logs

Authentication_Package_Name	string	Event Logs
Logon_Process_Name	string	Event Logs
SourceFile	string	Event Logs
TargetUserSid	string	Event Logs
Created_Std Info	datetime	MFT
Created_File Name	datetime	MFT
File_Name	string	MFT
SI<FN	bool	MFT
Created_On	datetime	UserAccounts SAM
Last_Login_Time	datetime	UserAccounts SAM
User_Id	string	UserAccounts SAM
Groups	string	UserAccounts SAM
LastLoggedOnDisplayName	string	Registers
LastLoggedOnSAMUser	string	Registry
LastLoggedOnUser	string	Registry
LastLoggedOnUserSID	string	Registry
Last_Write_Timestamp	datetime	Registers
Timestamp	datetime	updated
src_EventLogs	bool	added
src_MFT	bool	added
src_UserAccounts_SAM	bool	added
src_HKLM_Software_LogonUI	bool	added
Ts_Event Logs	bool	added
Ts_MFT_Std Info	bool	added
Ts_MFT_File Name	bool	added
Ts_SAM_Created	bool	updated
Ts_SAM_Last Login	bool	added
Ts_Logon UI	bool	added
Payload	string	Event Logs

- **Local failed login**

Attribute	Data type	Source artefact
EventRecordId	int	Event Logs
TimeCreated	datetime	Event Logs
EventId	int	Event Logs
Provider	string	Event Logs
ProcessId	int	Event Logs
ThreadId	int	Event Logs
Computer	string	Event Logs
ChunkNumber	int	Event Logs
MapDescription	string	Event Logs

UserName	string	Event Logs
RemoteHost	string	Event Logs
Target	string	Event Logs
Logon Type	int	Event Logs
FailureReason2	string	Event Logs
SourceFile	string	Event Logs
Payload	json	Event Logs

- **RDP login on the target device**

Attribute	Data type	Source artefact
EventRecordId	Integer	Event Logs
TimeCreated	Datetime	Event Logs
EventId	Integer	Event Logs
Provider	String	Event Logs
Channel	String	Event Logs
ProcessId	Integer	Event Logs
ThreadId	Integer	Event Logs
Computer	String	Event Logs
ChunkNumber	Integer	Event Logs
UserId	String	Event Logs
MapDescription	String	Event Logs
RemoteHost	String	Event Logs
Target	String	Event Logs
Logon Type	Integer	Event Logs
Logon ID	String	Event Logs
Authentication Package Name	String	Event Logs
Logon Process Name	String	Event Logs
Session ID	Integer	Event Logs
SourceFile	String	Event Logs
Payload	String	Event Logs

- **RDP failed login on target device**

Attribute	Data type	Source artefact
EventRecordId	Integer	Event Logs
TimeCreated	Datetime	Event Logs
EventId	Integer	Event Logs
Provider	String	Event Logs
ProcessId	Integer	Event Logs
ThreadId	Integer	Event Logs
Computer	String	Event Logs
UserName	String	Event Logs

MapDescription	String	Event Logs
RemoteHost	String	Event Logs
SourceFile	String	Event Logs
Target User	String	Event Logs
Logon Type	Integer	Event Logs
Failure Reason	String	Event Logs
Payload	JSON	Event Logs

- **RDP logout on the target device**

Attribute	Data type	Source artefact
RecordNumber	Integer	Event Logs
TimeCreated	Datetime	Event Logs
EventId	Integer	Event Logs
Provider	String	Event Logs
ProcessId	Integer	Event Logs
ThreadId	Integer	Event Logs
Computer	String	Event Logs
User ID	String	Event Logs
Map Description	String	Event Logs
User Name	String	Event Logs
SourceFile	String	Event Logs
Target	String	Event Logs
Logon Type	Integer	Event Logs
Payload	JSON	Event Logs

- **RDP login on the source device**

Attribute	Data type	Source artefact
EventRecordId	Integer	Event Logs
TimeCreated	Datetime	Event Logs
EventId	Integer	Event Logs
Provider	String	Event Logs
Channel	String	Event Logs
ProcessId	Integer	Event Logs
ThreadId	Integer	Event Logs
Computer	String	Event Logs
ChunkNumber	Integer	Event Logs
UserId	String	Event Logs
MapDescription	String	Event Logs
RemoteHost	String	Event Logs
SourceFile	String	Event Logs
Destination IP	String	Event Logs

Destination Name	String	Event Logs
Payload	JSON	Event Logs

- **RDP logout on the source device**

Attribute	Data type	Source artefact
TimeCreated	Date and time	Event Logs
EventId	Integer	Event Logs
Provider	String	Event Logs
ProcessId	Integer	Event Logs
ThreadId	Integer	Event Logs
Computer	String	Event Logs
User ID	String	Event Logs
MapDescription	String	Event Logs
SourceFile	String	Event Logs
Disconnect Reason	String	Event Logs
Payload	JSON	Event Logs

- **Creating a new service**

Attribute	Data type	Source artefact
Event Record Id	Integer	Event Logs
Time Created	Datetime	Event Logs
Event ID	Integer	Event Logs
Provider	String	Event Logs
Channel	String	Event Logs
Process ID	Integer	Event Logs
Thread ID	Integer	Event Logs
Computer	String	Event Logs
Chunk Number	Integer	Event Logs
User ID	String	Event Logs
Map Description	String	Event Logs
Executable Info	String	Event Logs
Source File	String	Event Logs
Service Name	String	Event Logs
Start Type	String	Event Logs
Account	String	Event Logs
Payload	String	Event Logs

- **Service Start**

Attribute	Data type	Source artefact
Event Record Id	Int	Event Logs
Time Created	Datetime	Event Logs
Event ID	Int	Event Logs
Provider	string	Event Logs
Channel	string	Event Logs
Process ID	int	Event Logs
Thread ID	int	Event Logs
Computer	string	Event Logs
Map Description	string	Event Logs
Executable Info	string	Event Logs
Source File	string	Event Logs
Service Name	string	Event Logs
Status	string	Event Logs

- **Creating and/or running a scheduled task**

Attribute	Data type	Source artefact
Description	String	Registers
Key Path	String	Registers
Hive Path	String	Registers
Last Write Timestamp	Date and Time	Registers
Deleted	Bool	Registers
Time Created	Date and time	Registry, Event Logs
Last start	Date and time	Registers
Last stop	Date and time	Registers
Path	String	Registers
Command	String	Registers
EventRecordId	Integer	Event Logs
EventId	Integer	Event Logs
Provider	String	Event Logs
Channel	String	Event Logs
ProcessId	Integer	Event Logs
ThreadId	Integer	Event Logs
Computer	String	Event Logs
ChunkNumber	Integer	Event Logs
UserId	String	Event Logs
MapDescription	String	Event Logs
UserName	String	Event Logs
Task	String	Event Logs
Instance ID	String	Event Logs

SourceFile	String	Event Logs
Payload	JSON	Event Logs

- **Process launch**

Attribute	Data type	Source artefact
Executable Name	string	Prefetch
Executable Path	string	Prefetch, ShimCache
Hash	string	Prefetch
Size	int	Prefetch
Run Count	int	Prefetch
Last Run	datetime	Prefetch
Previous Run 0	datetime	Prefetch
Previous Run 1	datetime	Prefetch
Previous Run 2	datetime	Prefetch
Previous Run 3	datetime	Prefetch
Previous Run 4	datetime	Prefetch
Previous Run 5	datetime	Prefetch
Previous Run 6	datetime	Prefetch
Volume0Name	string	Prefetch
Volume1Name	string	Prefetch
Directories	string	Prefetch
Files Loaded	string	Prefetch
Last Modified Time UTC	datetime	ShimCache

- **File access**

Attribute	Data type	Source artefact
Entry Number	int	MFT
Sequence Number MFT	int	MFT
Parent Path_MFT	string	MFT
Parent Entry Number	int	MFT
File Name	string	MFT, UsnJ
Extension_MFT	string	MFT
Has Ads	bool	MFT
Is Ads	bool	MFT
File Size	int	MFT
Created_MFT	datetime	MFT
SI_Created_MFT	bool	updated
FN_Created_MFT	bool	updated
Last Access_MFT	datetime	MFT
SI_Last Access_MFT	bool	updated

FN_Last Access_MFT	bool	updated
Last Modified_MFT	datetime	MFT
SI_Last Modified_MFT	bool	updated
FN_Last Modified_MFT	bool	added
Last Record Change_MFT	datetime	MFT
SI_Last Record Change_MFT	bool	updated
FN_Last Record Change_MFT	bool	added
Parent Path_J	string	UsnJ
Sequence Number J	int	UsnJ
Extension_J	string	UsnJ
Update Sequence Number	int	UsnJ
Update Reasons	string	UsnJ
File Attributes	string	UsnJ
Update Timestamp	datetime	UsnJ
App Id_Jump L	string	Jump List
Creation Time_Jump L	datetime	Jump List
Last Modified_Jump L	datetime	Jump List
Interaction Count_Jump L	int	Jump List
Target Created_Jump L	datetime	Jump List
Target Modified_Jump L	datetime	Jump List
Target Accessed_Jump L	datetime	Jump List
Path_Jump L	string	Jump List
File Size_Jump L	int	Jump List
Machine ID	string	Jump List

- **Delete file**

Attribute	Data type	Source artefact
Delete Time	datetime	UsnJournal
File Name	string	UsnJournal
Extension	string	UsnJournal
Entry Number	int	UsnJournal
Sequence Number	int	UsnJournal
Parent Entry Number	int	UsnJournal
Update Reasons	string	UsnJournal

- **File download**

Attribute	Data type	Source artefact
Entry Number	int	MFT
Sequence Number	int	MFT
Parent Path	string	MFT

Parent Entry Number	int	MFT
In Use	bool	MFT
File Name	string	MFT
Extension	string	MFT
Has Ads	bool	MFT
File Size	int	MFT
Created0x10	datetime	MFT
Created0x30	datetime	MFT
LastAccess0x10	datetime	MFT
LastAccess0x30	datetime	MFT
Copied	bool	MFT
SI<FN	bool	MFT

- **Changing file time attributes (timestamping)**

Attribute	Data type	Source artefact
Entry Number	int	MFT
Sequence Number	int	MFT
Parent Path	string	MFT
Parent Entry Number	int	MFT
In Use	bool	MFT
File Name	string	MFT
Extension	string	MFT
File Size	int	MFT
Created0x10	datetime	MFT
Created0x30	datetime	MFT
LastModified0x10	datetime	MFT
LastModified0x30	datetime	MFT
LastRecordChange0x10	datetime	MFT
LastRecordChange0x30	datetime	MFT
LastAccess0x10	datetime	MFT
LastAccess0x30	datetime	MFT

- Executable file creation

Attribute	Data type	Source artefact
Entry Number	int	MFT, UsnJ
Parent Path	string	MFT, UsnJ
Parent Entry Number	int	MFT, UsnJ
Sequence Number	int	MFT
File Name	string	MFT, UsnJ
Extension	string	MFT, UsnJ

Has Ads	bool	MFT
File Size	int	MFT
Created 0x10	datetime	MFT
Created0x30	datetime	MFT
Copied	bool	MFT
SI<FN	bool	MFT
Usn Create Time	datetime	UsnJ

- **Archive creation**

Attribute	Data type	Source artefact
Entry Number	int	MFT, UsnJ
Parent Path	string	MFT, UsnJ
Parent Entry Number	int	MFT, UsnJ
Sequence Number	int	MFT
File Name	string	MFT, UsnJ
Extension	string	MFT, UsnJ
Has Ads	bool	MFT
File Size	int	MFT
Created0x10	datetime	MFT
Created0x30	datetime	MFT
Copied	bool	MFT
SI<FN	bool	MFT
Usn Create Time	datetime	UsnJ

Metarecords defined in this way form a basic layer over aggregated data and enable the transition from isolated artefacts to interpretable events that are directly usable in forensic analysis and the detection of cyber security incidents.

3.3 Linking digital evidence

Once the metadata records have been created, a situation arises in which a single real-world event is represented by multiple metadata records, each derived from a different artefact or data source. Although these metadata records describe the same activity, they differ in their level of detail, the accuracy of their timestamps, and the available attributes. Such redundancy

is a natural consequence of digital forensic analysis based on heterogeneous data, but at the same time creates a need for further processing. Several attributes of the metadata, for different artefacts, remain undefined.

For this reason, **a data fusion** process was applied, the aim of which is to merge multiple partial representations of the same event into a single consistent and comprehensive entity. The result of this process is a unified representation of the event, which integrates information from various sources whilst eliminating duplicate records.

Data fusion is carried out on the basis of defined correlation criteria, which include in particular:

- the temporal proximity of events within a defined time window,
- matching identifiers (e.g. user, SID, device name, IP address),
- contextual links (e.g. same process, session ID, logon ID),
- event type (match in the metadata record category).

The process of merging meta-records consists of identifying candidate meta-records that likely represent the same event and subsequently merging them into a single resulting meta-record. During the merging process, the following occurs:

- selection of the most accurate or reliable attribute values,
- the completion of missing data from other metadata records,
- preservation of information regarding the origin of the data.

The result is an enriched metadata record that provides a more comprehensive view of the analysed event whilst reducing data redundancy. This approach enables more efficient analysis, more accurate reconstruction of the sequence of events, and better interpretation of system or user behaviour. The linking of digital evidence thus represents a key step between the extraction of individual forensic traces and their final aggregation into temporal or scenario-based units, thereby significantly improving the quality and usability of the resulting data model.

3.4 Aggregation of digital evidence

The aggregation of metadata represents an advanced stage of processing in which metadata that has already been created and, where applicable, merged, is organised into larger analytical units. The aim of this phase is to reduce the fragmentation of recorded events, highlight their interrelationships, and create a clearer picture of the course of the analysed activities. In digital forensic analysis, this approach is a natural continuation of artefact processing, as it allows one to move from individual events to coherent sequences or behavioural scenarios.

The first method of aggregation is grouping metadata records according to a selected time window. With this approach, events that occurred within the same or a closely related time interval are combined into a single aggregated unit. Groups created in this way allow us to track local clusters of activity, analyse their temporal sequence, and create clear time intervals that can represent, for example, a specific user session, a phase of an attack, or a separate security incident.

The size of the time window can be adjusted to suit the nature of the data being analysed and the required granularity of the results. Smaller time windows (e.g. 30 seconds) allow for a finer capture of immediately consecutive events, whilst larger windows (e.g. 24 hours) support the identification of broader activities consisting of several interrelated steps. The aggregation may also include selected parameters, such as event type, user identity, hostname, process, file path or session, thereby reducing the risk of temporally close but factually unrelated events being incorrectly merged into a single entity. With this type of aggregation, we have the option of selecting several parameters, or even all of them, except for time.

The second method of aggregation involves grouping metadata records according to a selected common parameter, i.e. without a primary focus on a fixed time window. In this case , scenario sets are created representing activities linked to a specific entity, such as a user account, file, process, service, IP address or device. This method of aggregation allows us to track the behaviour of a specific object across multiple events and better identify its significance within the analysed sequence of events.

An example would be the aggregation of all metadata records linked to a specific user, creating a comprehensive overview of their logins, running processes, file accesses or remote sessions. Similarly, it is possible to create a scenario set for a specific file, within which events relating to its creation, modification, access, download, archiving or deletion will be linked. The result is an analytical structure that provides not only a temporal view of events, but also an entity-oriented view of their progression and interrelationships.

Both aggregation approaches complement each other and enable the analysis of data from two different yet complementary perspectives. Temporal aggregation supports the reconstruction of events in chronological form, whilst entity-based aggregation helps to build scenario-based or object-oriented views of system and user behaviour. These two approaches can, of course, be combined, allowing us to focus not only on selected entities but also on specific time periods in which events occurred. When combined, these approaches enhance the interpretative value of the resulting model and provide a suitable basis for subsequent forensic analysis, incident detection and the reconstruction of complex activities within the system.

For the aggregation of metadata records, we opted for a temporal approach, aggregating individual metadata records separately within each behavioural scenario, including the use of parameters. We employ high-level aggregation (referred to as ‘Aggregation 1’ in the following tables), which combines all events of the same type (e.g. all local logins) within the required

time window, as well as second-level aggregation (referred to as “Aggregation 2” in the following tables), which takes selected parameters into account (e.g. user SID).

For the individual scenarios, we selected the following attributes for aggregation:

- **Local login**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
Timestamp	Local logon from "Computer" to user "Target"	Target	X local log-ons to user "Target"	X	
Timestamp	First log-on by user "UserName"	x		X	
Timestamp	The last user to log on was "UserName"	x		x	
Created_On	Creation of user "UserName"	All	X users created.		

- **RDP login on the target device**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
TimeCreated	RDP login from "RemoteHost" to "Target".	All	X RDP logins.	RemoteHost	X RDP logins from "Remote Host".
TimeCreated		All	X RDP logins.	RemoteHost	X RDP logins from "Remote Host".

- **RDP logins on the source device**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
TimeCreated	RDP connection from "Computer" to "Destination Name".	All	X RDP connections.	Destination Name	X RDP connections to "Destination Name"

TimeCreated	RDP connection from "Computer" to "Destination IP".	All	X RDP connections.	Destination Name	X RDP connections to "Destination Name"

- **RDP logout on the source device**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
TimeCreated	RDP logoff initiated by computer "Computer".	All	X RDP logoffs initiated.	x	

- **Creation of a new service**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
TimeCreated	The "Service Name" service with "Start type" was installed by "User Id".	All	X services were installed	User Id	X services were installed by the "User Id" user

- **Starting the service**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
TimeCreated	The "Service Name" service has been launched.	All	X services launched	x	

- **Create and/or run a scheduled task**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
TimeCreated	The scheduled task "Task" was created by "User Id" to run under	All	X scheduled tasks created.	User Id	X scheduled tasks created by user "User Id"

	the "User Name" user.				
TimeCreated	Scheduled task "Task" associated with user "User Id" was created.	All	X scheduled tasks created.	User Id	X scheduled tasks associated with "User Id" created
TimeCreated	Scheduled task "Task" started under user "User Name".	All	X scheduled tasks started.	User Name	X scheduled tasks started under the user "User Name"
TimeCreated	The scheduled task "Task" was triggered by the logon of user "User Name".	All	X scheduled tasks triggered by logon	User Name	X scheduled tasks triggered by the logon of "User Name"

- **Process start**

Time attribute	**Event**	**Aggregation 1**		**Aggregation 2**	
		Aggregation attribute	Event	Aggregation attribute	Event
Last Run	"ExecutableName" process was executed from "ExecutablePath"	All	X processes were executed	ExecutablePath	X processes were executed from the "ExecutablePath" path.

- **File access**

Time attribute	**Event**	**Aggregation 1**		**Aggregation 2**	
		Aggregation attribute	Event	Aggregation attribute	Event
Last Access_MFT	Access to the "FileName" file.	All	X files accessed	x	
Target Accessed_Jump L	Access to the "Path_Jump L" file.	All	X files accessed	x	

- **Deleting a file**

Time attribute	**Event**	**Aggregation 1**		**Aggregation 2**	
		Aggregation attribute	Event	Aggregation attribute	Event

Delete Time	Deletion of the "FileName" file.	All	X files were deleted	x	

- **File download**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
Created0x10	The file "FileName" was downloaded.	All	X files were downloaded	Extension	X files with the "Extension" extension were downloaded.

- **Changing file timestamps (timestomping)**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
LastModified0x10	Suspected timestomping of "FileName".	All	Suspected timestomping of X files.	Extension	Suspected timestomping of X "Extension" files.

- **Executable file creation**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
Created0x10	The executable file "FileName" was created in "Parent Path".	All	X executables were created	Parent Path	X executables were created in the "ParentPath" folder

- **Archive creation**

Time attribute	Event	Aggregation 1		Aggregation 2	
		Aggregation attribute	Event	Aggregation attribute	Event
Created0x10	The archive "FileName" was created in the path "ParentPath".	All	X archives were created	Parent Path	X archives were created in the "ParentPath" folder

3.5 Visualisation of aggregated digital evidence

A visualisation tool (Figure 1) was created based on the processed metadata records, designed to clearly display the aggregated results and facilitate their further analysis. Only the most important parameters of individual metadata records were selected for display, in order to maintain a clear overview whilst capturing the essence of individual events. The tool supports the adjustment of the aggregation window size, sorting of records by time, and filtering mechanisms, which allows the view of the data to be tailored to the requirements of the current analytical task.

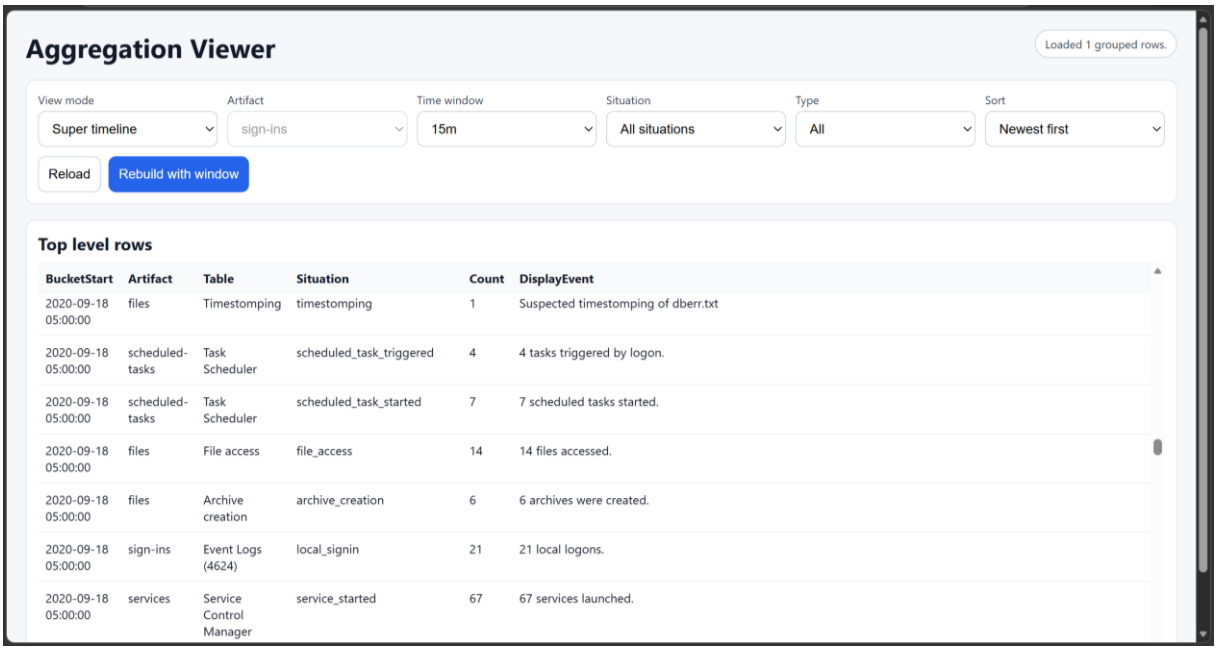


Figure 1 – Aggregation visualisation tool

The display operates at the level of aggregated meta-records; when a specific aggregated record is displayed, it is also possible to view the individual meta-records included in the resulting aggregation. The solution also includes a link to the original records in the artefacts, making it possible to trace the source of the information and verify the data from which the resulting event was derived (Figure 2 and Figure 3). This approach combines a high degree of clarity with the requirements for traceability and transparency in forensic data processing.

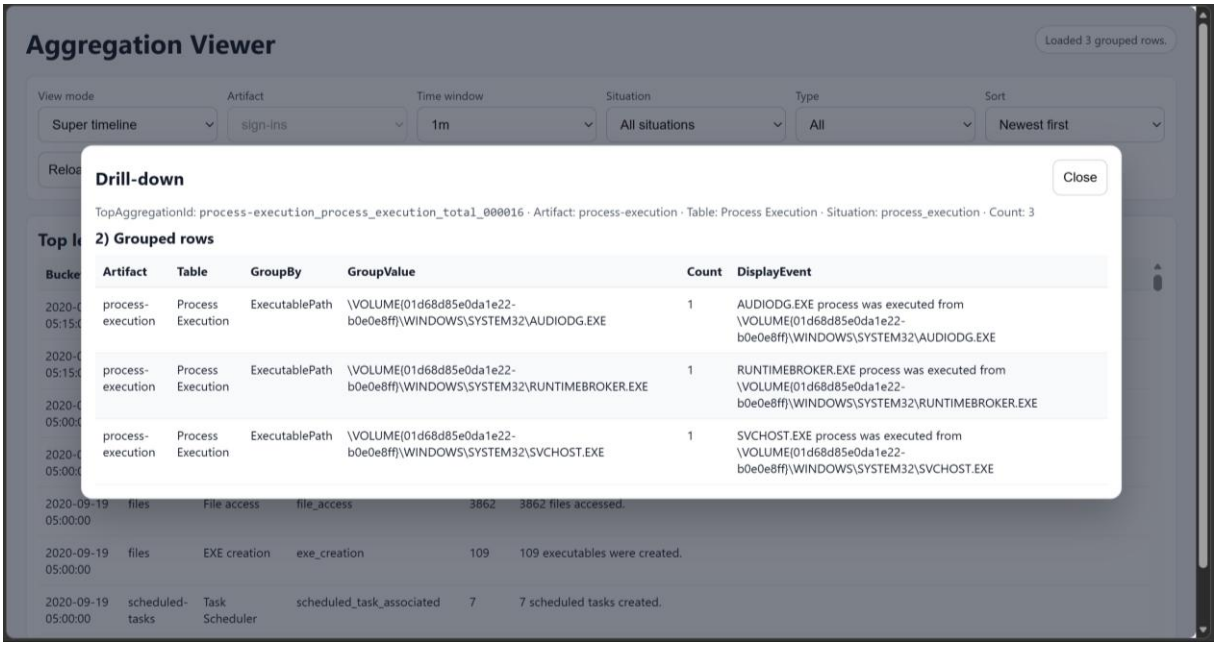


Figure 2 – Linking aggregated data to original records in artefacts – example 1

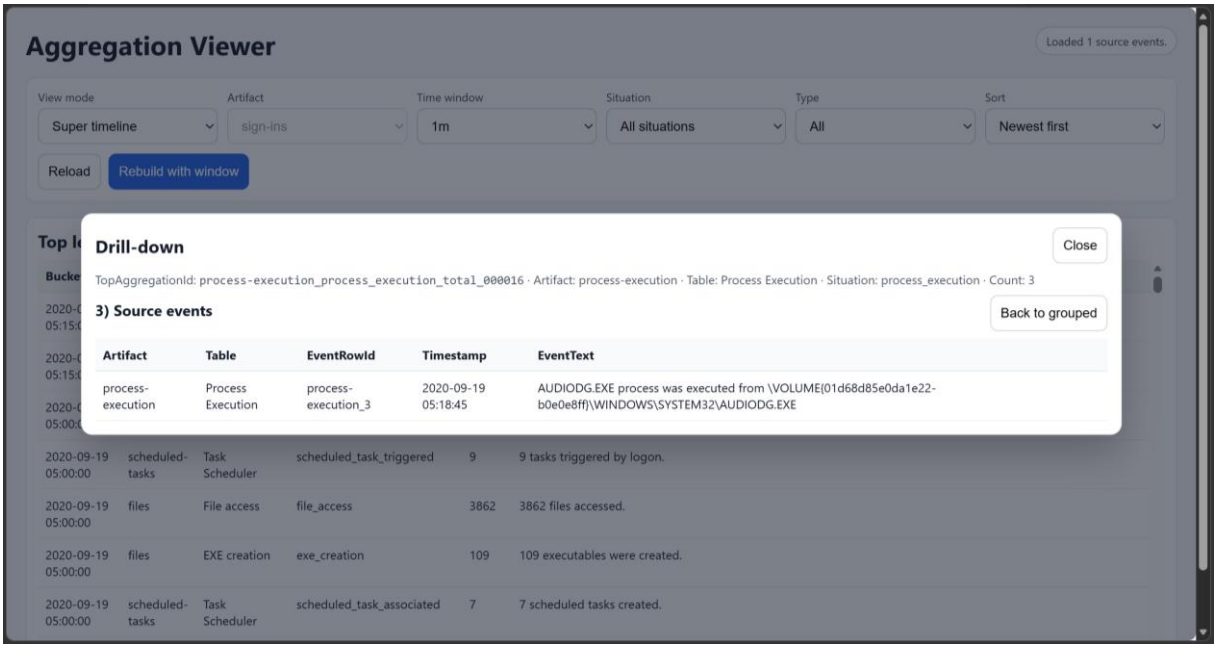


Figure 3 – Linking aggregated data to original records in artefacts – example 2

4 Aggregation functions with regard to unknown values (NaN)

As part of our investigation into methodologies for processing data with incomplete data (i.e. with missing values NaN), we also focused on the design and formal specification of **aggregation functions** capable of explicitly reflecting **the presence of such unknown data**. From a mathematical perspective, we understand these functions as mappings $A\colon \mathrm{N}_0^3 \to [0,1]$ that transform discrete frequencies (abundances) of state occurrences into a continuous measure of relevance or information density. Traditional aggregation operators such as minimum, maximum, sum or arithmetic mean are designed to work exclusively with available numerical values. They do not take into account information about the number or proportion of missing values, which can lead to an incomplete or distorted interpretation of the results. In many data structures, however, it is precisely the number of missing values that represents important information (e.g. in sensor networks, a value of NaN may indicate a node failure, which is a qualitatively different state from a measured zero). For this reason, it is advisable to define new aggregation functions or extend classical ones in a way that allows their results to be modified depending on the incompleteness of the data. Aggregation functions designed in this way can significantly contribute to a more accurate interpretation of the data.

The aggregation functions we propose operate on a triplet of input parameters that summarise the basic characteristics of binary data: the number of zeros ($\#0$), the number of ones ($\#1$) and the number of missing values ($\#\mathrm{NaN}$). An important prerequisite is always the relationship

$$\#0 + \#1 + \#\mathrm{NaN} = N,$$

where$N$ is a natural number. This relationship implies that such data can be represented as a point in a discrete simplex in three-dimensional space. Thus, a reduction in the number of0 must be reflected in an increase in the number of1 orNaN values, such that their sum remains unchanged. The same applies to the other parameters. Depending on the nature of the data being processed and the significance attributed to individual values, we distinguish between the three types of situations listed below. These situations correspond to different semantic interpretations of uncertainty in the data. Each represents a distinct interpretative framework and therefore requires a specifically defined aggregation function. The resulting aggregation value is determined by the required properties that the function must satisfy – in particular, the way in which it must respond to the presence of missing data, to the dominance of one of the binary values, or to the need to maintain a certain monotonicity or robustness against incomplete data. In each of the cases under consideration, we consider aggregation functions whose output is a value from the interval$[0,1]$, where the boundary values are included. This interval represents a natural range for interpreting aggregation, as it ensures compatibility with a probabilistic interpretation, normalisation across different datasets (comparison of aggregated values) and computational stability, which is important from a technical perspective.

4.1 The values 0 and 1 are equally significant

In a situation where the values 0 and 1 are considered equally significant, it is necessary for the aggregation to be symmetric with respect to these two categories. Symmetry (commutativity) means that the aggregation does not distinguish between these values; both belong to the same set of valid binary values. Formally, we require that $A(\#0, \#1, \#\mathrm{NaN}) = A(\#1, \#0, \#\mathrm{NaN})$ holds. The aggregation function should therefore respond only to their joint count relative to the number of unknown values NaN. In this setting, it naturally follows that:

- the maximum aggregation is achieved when all data are known (i.e. consist exclusively of 0 and 1),
- the minimum occurs when the data consists of unknown values (only NaN), because the aggregation has no information available.

Based on these requirements, the following aggregation function is a suitable choice:

$$A(\#0, \#1, \#\mathrm{NaN}) = \frac{\#0 + \#1}{\#0 + \#1 + \#\mathrm{NaN}}$$

This function expresses the proportion of known binary values relative to the total number of values (including unknowns). It represents a measure of the 'completeness' of the data set, or, to put it another way, it expresses the probability that a randomly selected element from the set is not missing. By its construction, the aggregation function also satisfies the property of monotonicity (while maintaining a constant N):

- for a fixed number of unknownsNaN , the aggregation is constant with respect to the number of0 and1 , as it does not distinguish between them (it depends only on their sum),
- conversely, the aggregation function is decreasing with respect to the increasing number ofNaN : the more unknown data there is, the smaller the proportion of valid information.

4.2 A value of 1 is more significant than 0

In this case, it is important to assume that the occurrence of the value 1 has greater informational value than the occurrence of the value 0. For this reason, we assign a weight of 1 to the unit, whilst assigning a weight of$w \in (0,1)$ to zero, which reflects its relatively lower significance. It is natural to require that an increasing number of unknown values NaN leads to a systematic decrease in the aggregated value, since the available information represents an increasingly smaller proportion of the total number of known values. Based on this, the aggregation function should satisfy that

- the maximum aggregation value is attained when the number of values 1 is maximal, and thus the number of 0 and NaN is zero,
- the minimum value of the aggregation is reached when all values are unknown, i.e. NaN,
- if all values are 0, the aggregation takes on the value$w$, which reflects the chosen weight for the value 0.

Based on the above, we propose the following aggregation function:

$$A(\#0, \#1, \#\mathrm{NaN}) = \frac{\#1 + w \cdot \#0}{\#0 + \#1 + \#\mathrm{NaN}}, \quad w \in (0,1)$$

The aggregation function defined in this way respects the asymmetry of significance between the values 1 and 0 , whilst at the same time responding sensitively to the presence of missing data and preserving the interpretability of the result within the normalised interval $[0,1]$. More precisely, parameterisation using$w$ allows for fine-tuning the model's sensitivity to the presence of 'negative' (0) versus 'positive' (1) results. At the same time, the monotonicity properties are satisfied (for a fixed$N$), meaning that the proposed aggregation function

- is increasing with respect to the increasing number of 1 given a fixed number of 0 or NaN, since units have the highest weight,
- is increasing with respect to the increasing number of 0 given a fixed number of 1, because zeros contribute to the numerator with a positive weight,
- it is decreasing with respect to the increasing number of 0 given a fixed number of NaN, since an increase in the values of0 simultaneously reduces the number of values of 1,
- is decreasing given the increasing number of NaN, since an increase in unknown values reduces the number of 0 and1.

4.3 The values 0 and NaN are equally significant

In the event that the value1 is the sole carrier of relevant information and the values 0 and NaN have the same significance in terms of interpretation (symmetry), it is necessary to use the aggregation function to capture exclusively the relative proportion of values 1 in the entire set of values. This situation is typical for sparse data, where we are only interested in the occurrence of a phenomenon and its absence is indistinguishable from its non-measurement. In this context

- the maximum value of the aggregation function should be achieved when the number of values 1 is as large as possible,
- the minimum value of the aggregation should occur when no value 1 appears in the data – whether it consists of all 0 , all NaN , or any combination thereof, which reflects the symmetrical relationship between the values 0 and NaN .

The proposed aggregation function takes the form:

$$A(\#0, \#1, \#\mathrm{NaN}) = \frac{\#1}{\#0 + \#1 + \#\mathrm{NaN}}$$

From a mathematical point of view, this is a limit case of the previous function for$w \to 0^+$. In this model, there is a total loss of discriminatory power between 'known zero' and 'unknown value', which simplifies the calculation in binary classifiers with a high degree of uncertainty. Thus, aggregation defined in this way provides a simple and interpretatively stable method of evaluating data in situations where only the presence of the value 1 is relevant, and all other values represent an equivalent form of irrelevance or the absence of positive information. Such an aggregation function also satisfies the required properties of monotonicity (for a fixed N):

- it is increasing with respect to an increasing number of 1, since each additional unit increases the proportion of relevant information,
- remains constant as the number of 0 and NaN increases, given a fixed number of 1, since these two values are symmetric and therefore swapping them does not alter the resulting ratio of 1.

4.4 Summary and comparison

All three of the functions mentioned can be written in a generalised form:

$$A(\#0, \#1, \#\mathrm{NaN}) = \frac{\alpha \cdot \#1 + \beta \cdot \#0 + \gamma \cdot \#\mathrm{NaN}}{\#0 + \#1 + \#\mathrm{NaN}}$$

where, for individual cases, we choose coefficients (α, β, γ) from the set$\{0, w, 1\}$. This unified view enables the implementation of a single modular function in data science programming environments, where the aggregation behaviour changes only by adjusting the weighting parameters depending on the problem's semantics.

5 Bibliography

[1] Zimmerman, E. (n.d.). *Eric Zimmerman's Tools*. Available at https://ericzimmerman.github.io/#!index.md (accessed 15 April 2026).

[2] Zimmerman, E. (n.d.). *EvtxECmd*. GitHub repository. Available at https://github.com/EricZimmerman/evtx (accessed 15 April 2026).

[3] Zimmerman, E. (n.d.). *MFTECmd*. GitHub repository. Available at https://github.com/EricZimmerman/MFTECmd (accessed 15 April 2026).

[4] Zimmerman, E. (n.d.). *RECmd*. GitHub repository. Available at https://github.com/EricZimmerman/RECmd (accessed 15 April 2026).

[5] Zimmerman, E. (n.d.). *PECmd*. GitHub repository. Available at https://github.com/EricZimmerman/PECmd (accessed 15 April 2026).

[6] Zimmerman, E. (n.d.). AppCompatCacheParser. GitHub repository. Available at https://github.com/ericzimmerman/appcompatcacheparser (accessed 15 April 2026).

[7] Zimmerman, E. (n.d.). *JLECmd*. GitHub repository. Available at https://github.com/EricZimmerman/JLECmd (accessed 15 April 2026).