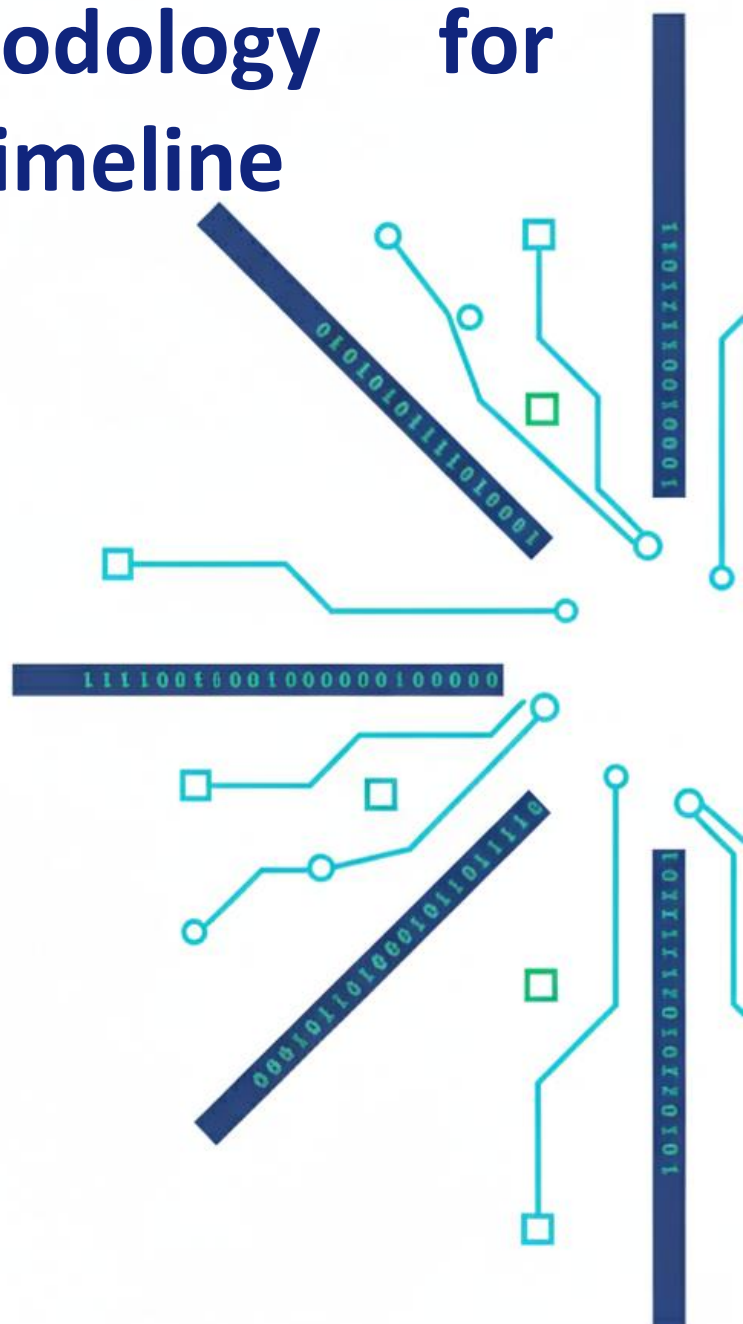


D23 – Methodology for creating a case timeline



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. 09I05-03-V02-00079.

Contents

Project description	2
1 Introduction	3
2 Data processing for the timeline	4
3 Tool for creating and displaying a timeline – ETL Timeline	6
3.1 Dependencies and prerequisites.....	6
3.2 Input data	6
3.3 Running the tool	7
3.4 Data processing in the background of the tool.....	7
3.4.1 Uploading data via UploadModal	8
3.4.2 Parsing and validation	8
3.4.3 Linking events with annotations	8
3.4.4 Transformation to TimelineCase	9
3.4.5 Timeline compression	9
3.4.6 Filtering	9
3.4.7 Building visual layers (for visible events only)	9
3.4.8 Density histogram	10
3.4.9 Drilldown.....	10
3.4.10 Detail panels	11
3.5 Data insertion	11
3.6 Graphical interface and its functionalities.....	12
3.6.1 Timeline graph view.....	12
3.6.2 Enhancing the ETL Timeline with outputs from other analyses	13
3.6.3 Working with text data	15
3.6.4 Filtering events	18
4 Forensic significance.....	20
5 Conclusion.....	21
6 Bibliography.....	22

Project description

The project Automation of Digital Forensic Analysis and Incident Response (hereinafter 'ADFIR') is funded by the European Union – Next Generation EU through the Recovery and Resilience Plan of the Slovak Republic under project number 09I05-03-V02-00079. This project addresses one of the key challenges in the field of cyber security and information security – how to process the vast amounts of digital evidence generated during cyber security incidents or forensic investigations. At present, this process is very demanding in terms of human resources and time. Automation using machine learning methods can therefore significantly improve the quality of digital forensic analysis and reduce the time required to carry it out. Overall, this enables security teams to respond more effectively to cyber threats. The main benefits of this project are:

- **Faster resolution of cyber security incidents.** The ADFIR project introduces automated approaches to the collection, processing and analysis of digital traces. This enables security teams to identify the causes of incidents more quickly and take effective measures to resolve them.
- **Reduced workload for forensic analysts.** Routine and time-consuming tasks associated with processing digital traces 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 standardised methodologies and tools ensures that processed digital traces will be more accurate, consistent and easier to verify. This significantly reduces the risk of errors caused by human factors.
- **Potential use in criminal proceedings.** The project's outputs will be developed in accordance with legal requirements and standards, enabling digital traces to be accepted as relevant evidence in investigations and court proceedings.

1 Introduction

A timeline is one of the fundamental tools of digital forensic analysis for visualising and interpreting events recorded in the operating system. Windows generates a vast amount of logs across multiple sources — Event Log, prefetch files, the registry, scheduled tasks and other artefacts — with each of these sources contributing its own layer of information about the activity of the system and its users. Taken in isolation, these records are difficult to interpret; it is only when they are merged into a single timeline that correlations and patterns in system behaviour become apparent.

When working with such a volume of data, linearly scrolling through the records is practically unfeasible. A visual representation over time allows the analyst to immediately identify clusters of events, unusual lulls in activity or sudden changes in the intensity of logs — phenomena that would go unnoticed in the raw data.

Furthermore, the ETL Timeline tool for viewing and working with the timeline can highlight various anomalies identified by other analytical tools developed as part of this project. Anomalies identified by these tools are visualised directly within the context of the timeline, giving the analyst an integrated view of the incident without the need to switch between multiple environments. Such an anomaly may indicate an unauthorised login, the launch of an unexpected process, a change to critical system files, or another event requiring further investigation.

This methodology describes how to compile, interpret and document such timelines within the platform, including the definition of relevant data sources, criteria for identifying anomalies, and requirements for recording analytical conclusions.

Forensic investigations often require the reconstruction of a timeline of events and activities relating to digital devices or users. Such timelines can provide valuable information on various criminal activities, including malware, brute-force attacks or an attacker's activities after gaining control of a system. The process of timeline analysis is complex and time-consuming, particularly when dealing with large volumes of digital data from multiple sources. Traditional methods of timeline analysis rely on manual analysis, which can be subjective and prone to error, and may result in key information being overlooked [1].

Event reconstruction involves reconstructing past events through the analysis of digital artefacts, enabling investigators to determine system behaviour and draw well-founded conclusions about what happened. Whilst traditional forensic science draws on a clearly defined framework that summarises the discipline, event reconstruction in digital forensic analysis is often examined in fragments, with attention focused on tasks such as creating a super-timeline, detecting tampering or identifying environmental specifics. As a result, research has centred on these narrow aspects, whilst broader challenges have remained under-explored or have been overlooked. The lack of a unified perspective has also led to the emergence of a multitude of inconsistent terms, which hinders a comprehensive discussion of event reconstruction or the search for relevant research; for example, some studies use the

term ‘artefact’, whilst others refer to ‘observable aspects’. Terms such as ‘events’, ‘user actions’, ‘interactions’ or ‘clicks’ are used inconsistently in the specialist literature [2].

2 Data processing for the timeline

The data obtained from the Windows operating system (disk bit images) passes through two parallel analytical workflows, the outputs of which are subsequently merged in the ETL Timeline tool. The entire processing diagram is shown in **Figure 1**.

The first workflow utilises Eric Zimmerman (EZ) tools [3] — a suite of specialised tools for parsing Windows operating system artefacts. The output consists of several CSV tables with different structures. These are consolidated using the ETL tool into a standardised format ready for visualisation.

Based on predefined scenarios, the ETL tool creates so-called meta-records – standardised records of events such as the creation of a process, the creation of a scheduled task, the downloading of a file, and so on. It then aggregates these meta-records. A more detailed description of how the ETL tool works, from pre-processing to visualisation, is provided in the output ADFIR-D17-KPB3-03- Model for aggregation and connection of digital evidence in Chapter 3 – Aggregation of forensic artefacts.

The second workflow utilises the Plaso tool, which aggregates events from various sources into a single super-timeline. Four types of analytical outputs are subsequently identified from this data — Clusters, Anomalies, Attack Stages and FCA — which are described in more detail in the KPB4 outputs (Clusters and Anomalies in the output D19-KPB4-01 Method for digital evidence identification, Stages in output D21-KPB4-03 Method for determination of time frame and attack phases, and FCA in output D20-KPB4-02.01 Method for identification of relations).

All exports from ETL and analysis outputs from Plaso are ultimately displayed in the ETL Timeline, which provides an integrated view of the case with visual highlighting of identified anomalies and events.

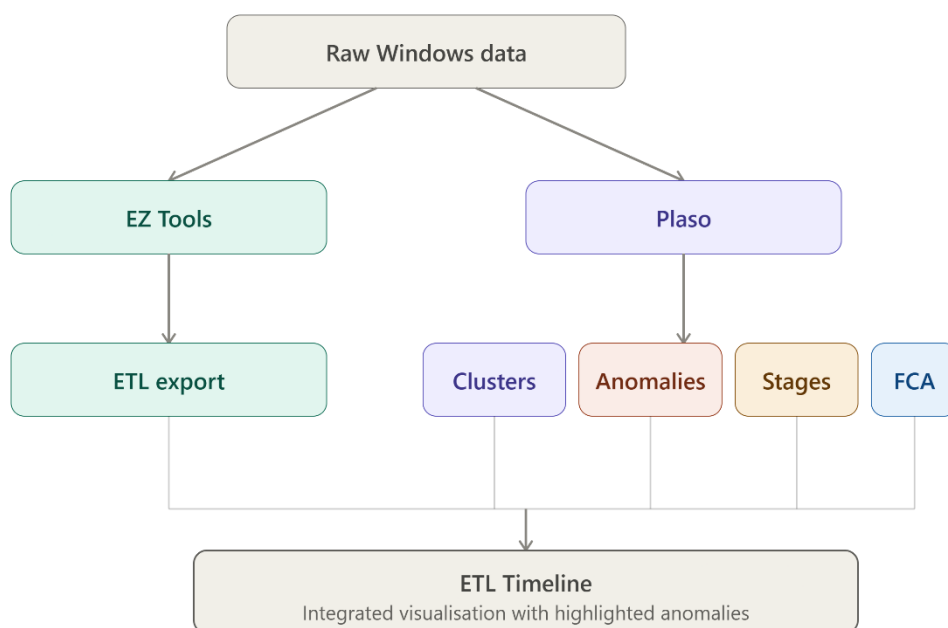


Figure 1 – Data processing diagram

3 Tool for creating and displaying a timeline – ETL Timeline

The ETL Timeline tool is used to display processed data from the Windows operating system in the form of a timeline of aggregated records, together with supplementary information on anomalies, clusters of similar data, and so on. It provides analysts with a clear way of visualising large volumes of data so that they are not overwhelmed by the sheer volume of information. At the same time, the supplementary information highlights potentially important or interesting time periods.

3.1 Dependencies and prerequisites

The front-end part of the application is built using **React** technology [4] , which is used to develop a modern user interface using components.

npm (Node Package Manager)[5] is used to manage packages and dependencies. It helps to install the libraries and tools required to develop, build and run the front-end application.

Before using the ETL Timeline, it is first necessary to run the ETL tool, which is used to obtain the required input data. A detailed procedure for processing the data into a suitable format is described in output D17-KPB3-03 ‘Model for the aggregation and linking of digital traces’.

3.2 Input data

The input must consist of data exported from the ETL tool. This is an exported file in CSV format, which forms the basic timeline. It contains a single timestamp, a description of the event that occurred, and information on multi-level aggregation.

As further inputs that enable the display of additional information – particularly data similarity – the following optional data in JSON format, which are outputs from analytical tools, can be added to ETL Timeline:

- *Clusters* – this is the output from a hierarchical clustering method. The method creates groups (clusters) of time windows that are similar to one another. This similarity may represent naturally similar activities, but may also indicate suspicious activity. Further information can be found in D19-KPB4-01 Method for digital evidence identification.
- *Attack Stages* – outputs from time series. Further information can be found in D21-KPB4-03 Method for determination of time frame and attack phases.
- *Anomalies* – anomalies are time windows that are assessed as being the most different from other time windows. Further information can be found in D19-KPB4-01 Method for digital evidence identification.
- *FCA* – Formal Concept Analysis (FCA) is a method for identifying groups of similarity based on common attributes across the largest possible set of rows. Further information can be found in D20-KPB4-02.01 Method for identification of relations.

Standardised format for optional data:

```
{
  "<object_name>": {
    "id": <id>
    "metadata": {
      "<metadata1>": <attribute1>,
      ...
    }
    "ts_objects": [
      {
        "ts_start": <time_start>,
        "ts_end": <time_end>
      },
      ...
    ]
  }
}
```

3.3 Running the tool

The ETL Timeline tool is launched in the following two steps:

1. Launching the application:

```
npm i
```

```
npm run dev # alternatively: npm run build && npm run preview
```

2. Opening the application in a browser:

The console will display the relevant link in the format `http://localhost:<port>`, for example `http://localhost:5173`.

3.4 Data processing in the background of the tool

ETL Timeline uses TypeScript. Data processing takes place in the following steps:

- 1) data loading,
- 2) data parsing and validation,
- 3) linking events with annotations,
- 4) transformation to TimelineCase,
- 5) timeline compression,
- 6) filtering,
- 7) building visual layers,
- 8) creation of a density histogram,
- 9) drill-down,
- 10) building detail panels.

3.4.1 Uploading data via UploadModal

The user can upload 5 types of data, as listed in **Table 1**. A specific parser is used to process each type. The result is a processed object.

Data	Format	Required	Parser → processed object
Timeline	CSV	yes	utils/csv.ts → utils/events.ts
Clusters	JSON	no	utils/temporalAnnotations.ts → utils/clusters.ts
Attack Stages	JSON	no	utils/temporalAnnotations.ts → utils/stages.ts
Anomalies	JSON	no	utils/temporalAnnotations.ts → utils/anomalies.ts
FCA factors	JSON	no	utils/temporalAnnotations.ts → utils/fca.ts

Table 1 – Data types suitable for the Timeline ETL tool

3.4.2 Parsing and validation

Depending on the input data format (CSV or JSON), the data is processed (parsed) and validated.

- CSV parser* (utils/csv.ts:3) – a manual parser supporting arrays in quotation marks; it maps headers to keys in lower case.
- JSON parser* (utils/temporalAnnotations.ts:12-38) – accepts a JSON array of objects or an object with a `ts\_objects` field. Each object contains time windows (ts_start/ts_end), metadata, and an ID. To display data more efficiently on the timeline graph, adjacent windows are merged (clusterWindowMergeGapMs).

3.4.3 Linking events with annotations

If clusters are specified as one of the inputs, individual clusters are assigned to events whilst the timeline is being prepared. The following function is used for this:

`applyClusterAssignments` (utils/clusters.ts:14) – assigns a cluster to each event depending on whether the event's timestamp falls within one of the windows of that cluster. This is the only cross-link between datasets – other annotations (anomalies, FCA, stages) are evaluated only during rendering, not during import.

3.4.4 Transformation to TimelineCase

The next step in processing is the transformation of the loaded events into the internal data structure of the TimelineCase object, which represents the basic unit used for further processing and visualisation of the timeline.

`buildCasesFromRows` (utils/events.ts:35) – creates an array of `SourceEvent` objects from the CSV rows, sorts them chronologically and wraps them into a `TimelineCase` object. This object therefore contains all the source events.

3.4.5 Timeline compression

The timeline compression process involves dividing the time range into segments and creating a scaling function that maps timestamps to their corresponding positions on the X-axis. This is handled by the following two functions:

`buildCompressedTimelineSegments` (utils/compressedTimeline.ts:4) – creates segments from `caseTimeRange`.

`buildCompressedTimelineScale` (utils/compressedTimeline.ts:89) – creates a scaling function that maps a timestamp to a specific pixel on the X-axis. It supports zooming via `zoomRange` (offset range).

3.4.6 Filtering

Once events have been loaded, user-selected filters are applied to them to determine which events will be included in further processing and display. The `filteredEvents` function is used for this purpose.

`filteredEvents` (App.tsx:157-168) – applies 5 layers of filters (source, eventType, situation, group, signal). Each filter is a `Record<string, boolean>`.

3.4.7 Building visual layers (for visible events only)

Each type of visualisation overlaid on the timeline (clusters, anomalies, FCA) is built in two levels:

Level 1 – constructed for the entire case (from `visibleCaseEvents`, which are events in `timelineSegments`):

- `selectedCaseGroups = buildEventGroups(visibleCaseEvents, clusterImport.clusters, timelineSegments)`
– used for the filter list and detail panel

- `selectedAnomalyGroups` = `buildAnomalyGroups(visibleCaseEvents, anomalyImport.anomalies, timelineSegments)` – used for the filter list and detail panel
- `selectedFcaFactorGroups` = `buildFcaFactorGroups(visibleCaseEvents, fcaImport.factors, timelineSegments)` – used for the filter list and detail panel

Level 2 – constructed for the currently visible zoom (from `graphEvents`, which are events in `timelineScale.viewSegments`):

- `graphClusterGroups`, `graphAnomalyGroups`, `graphFcaFactorGroups` – used to render the graph
- `graphGroups` = `packGraphGroups([...graphClusterGroups, ...graphAnomalyGroups, ...graphFcaFactorGroups])`
- `packGraphGroups` (`utils/events.ts:298`) assigns a row number in the layout to each visible group. These groups are then rendered as coloured bands (lanes) at the top of the graph.

A special mechanism is provided by so-called ‘*Stage markers*’: these are created directly in `App.tsx:256–280` as `TimelineStageMarker[]`, and do not pass through the `build*` functions or `packGraphGroups`. They are discrete vertical indicators on the timeline, not continuous lanes.

3.4.8 Density histogram

The density histogram provides an overview of the distribution of events on the timeline. It is created by dividing the time range into intervals, within which the number of events is counted, whilst the size of the intervals is dynamically adjusted to the current display range.

`buildDensityBins` (`utils/chart.ts:117`) – divides the visible time range into bins (“containers”), and counts the events in each bin. The bin width is dynamic (natural density bin width – rounded to ‘nice’ values such as 100 ms, 250 ms, 500 ms, 1 s, 5 s, etc.). The result is plotted as an area chart.

3.4.9 Drilldown

To display a detailed view of the events, a hierarchical tree structure is created, which allows the aggregated data to be progressively expanded down to the level of individual source events.

`buildDrilldownRows` (`utils/events.ts:302`) – creates a hierarchical tree from `matchingEvents`: top-level aggregation → grouped aggregation (optional) → source events. Virtualisation (only visible rows = `virtualDrilldown`).

3.4.10 Detail panels

Upon selecting an event, group or marker, a panel displaying detailed information about the selected item is shown. The type of panel displayed depends on the nature of the selection and provides either detailed information about a single event or an overview of corresponding events within the hierarchical structure. There are two types of panels available:

- DetailsPane – contains metadata, FCA factors and raw JSON data.
- MatchingDetailsPane – drill-down tree, time windows, zoom-to-selection option

3.5 Data insertion

Data can be uploaded by clicking the *'Upload data'* button. You can then enter the required input data by clicking on the individual fields (see **Figure 2**).

LOCAL IMPORT

Upload data ×

Timeline 126704 events

Choose timeline CSV

etl-timeline-export.csv

Clusters	Optional	Stages	Optional	Anomalies	Optional	FCA	Optional
Choose clusters JSON		Choose stages JSON		Choose anomalies JSON		Choose FCA JSON	
No clusters file selected		No stages file selected		No anomalies file selected		No FCA file selected	

1 case imported.

Figure 2 – Screenshot showing how to enter input data

3.6 Graphical interface and its functionalities

The tool for demonstrating the timeline functionality (see **Figure 3**) consists of the following parts:

- timeline graph,
- filters,
- text data,
- event details.

The core component is the timeline graph itself, which graphically displays the volume of data over time along with basic information such as the number of events displayed.

The filters on the left side of the window allow you to select the required event types from both the main and optional data.

Text information about events is displayed at the bottom of the window and clicking on an event displays more detailed information on the right side of the window.

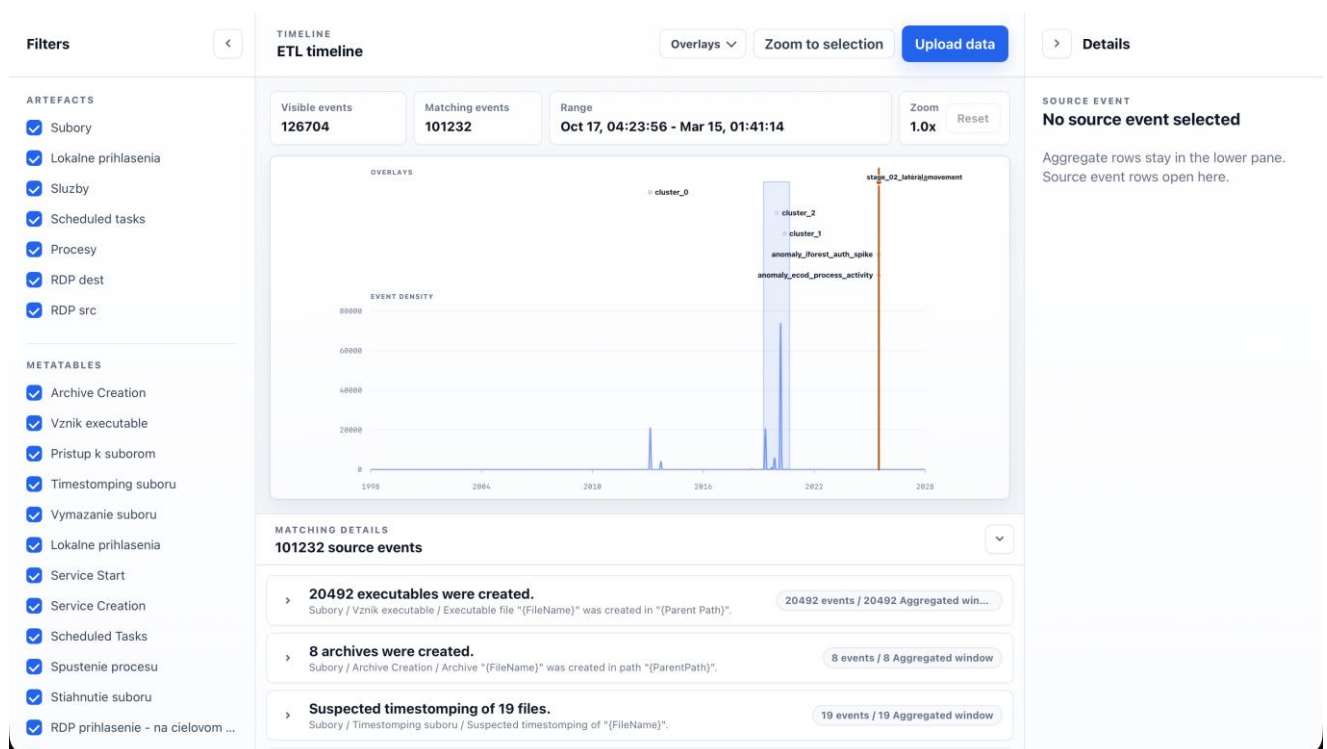


Figure 3 – ETL Timeline graphical interface

3.6.1 Timeline graph view

The timeline graph is displayed automatically once the required data has been entered. It shows the number of events at a given time. On the timeline, you can select a specific data

range for which you wish to view specific details (in **Figure 4**, the selected range is highlighted in blue).

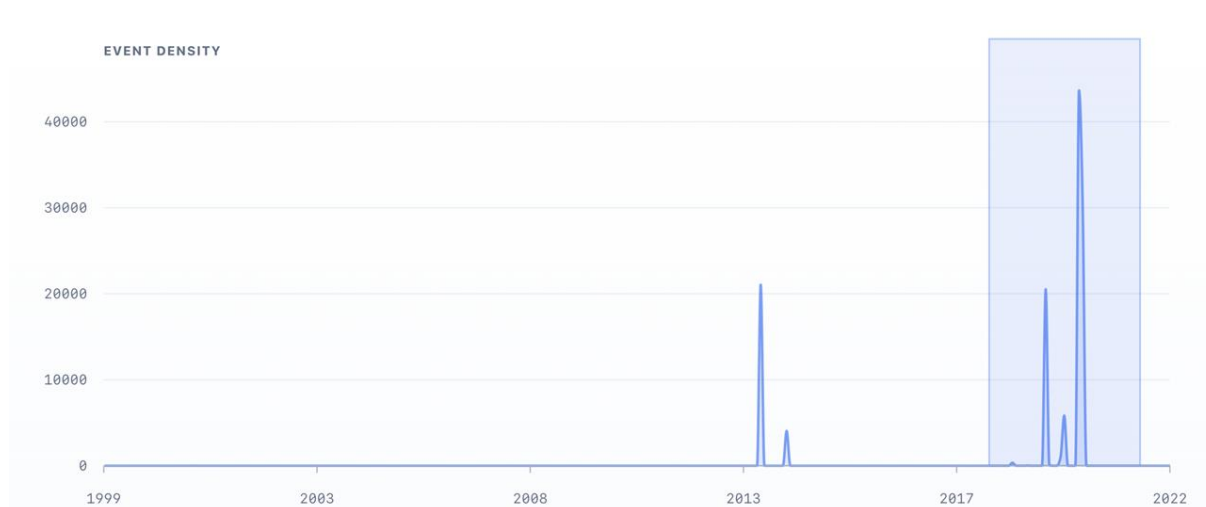


Figure 4 – Example of a timeline with a selected time period

A zoom function has also been implemented – this allows the timeline to be zoomed in so that the required time period is displayed in greater detail, enabling a more detailed examination. **Figure 5** shows an example of a zoomed-in timeline for the years 2017–2020. This is particularly useful for timelines covering a long period.

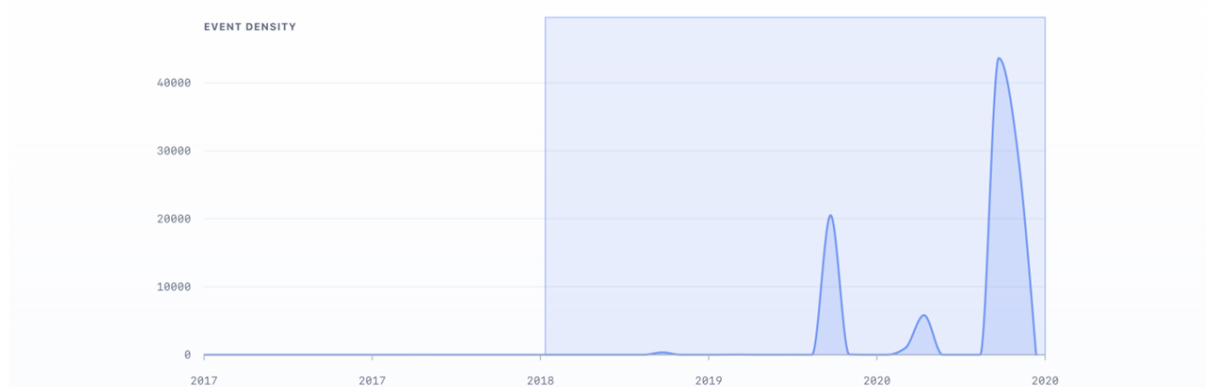


Figure 5 – Example of a timeline zoomed in on the period 2017–2020

To deselect a selection, simply click on the graph again; to reset the zoom level, use the **Reset** button.

3.6.2 Enhancing the ETL Timeline with outputs from other analyses

The enrichment feature allows you to display the outputs of other analytical tools directly above the timeline. This makes it easier to compare the results of individual analyses and to more quickly estimate the time window during which suspicious activity took place.

The basic prerequisite for displaying supplementary data is to upload the relevant data via the **‘Upload data’** interface. The individual artefacts are then automatically processed and prepared for display.

All supported analysis outputs (with the exception of **FCA**) are automatically displayed as overlapping layers (hereinafter referred to simply as ‘overlays’) on top of the timeline graph. The user can change their visibility at any time via the **‘Overlays’** interface, where individual overlays can be turned on or off as required. This allows only relevant information to be displayed without unnecessarily cluttering the graph.

Clusters, anomalies and FCA factors are displayed as circles. Clusters are shown in blue, anomalies in red and factors in green. Attack stages are depicted as orange vertical lines. An example of the display of clusters and stages can be found at **Figure 6**.



Figure 6 – Timeline enhanced with clusters and stages

All displayed layers are interactive. Clicking on a specific element displays more detailed information about that output, such as its identifier, timestamps or other attributes depending on the artefact type.

For outputs from **Cluster**, **Anomalies** and **FCA**, this interaction also offers extended functionality. Upon selecting a specific element, the time range of the graph covered by that output is automatically highlighted; see **Figure 7**. The user can thus immediately see the interval to which the result relates, which facilitates analysis and comparison with other events displayed on the timeline.

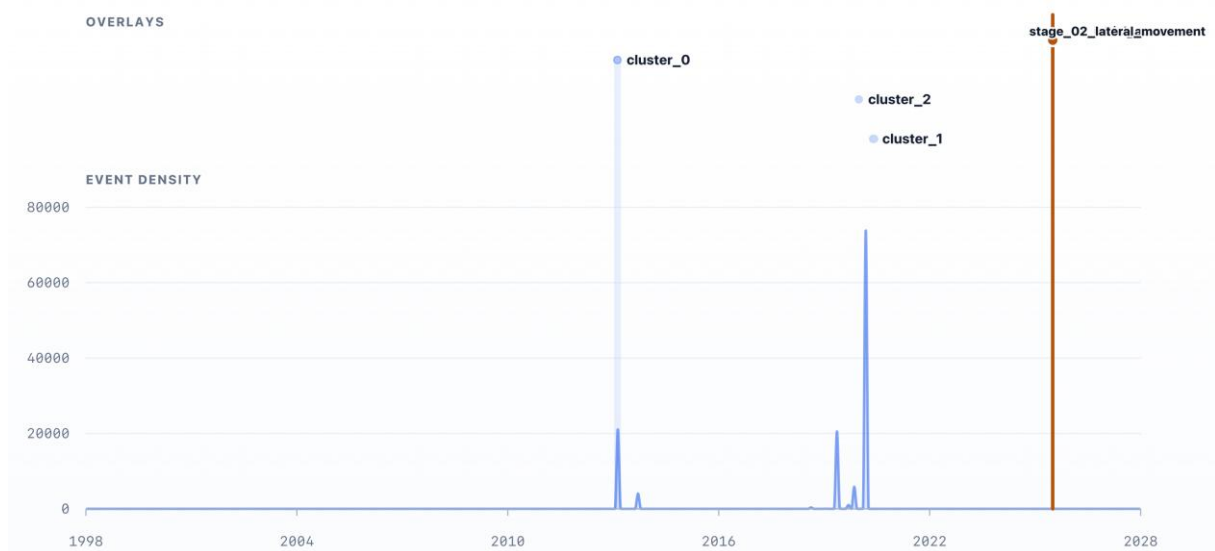


Figure 7 – Timeline with the section for 'cluster_0' highlighted

3.6.3 Working with text data

The bottom panel labelled '**Matching Details**', allows you to work with timeline events. It displays events only from the selected time interval (if no interval is selected, the panel remains empty, even if there are records available). The panel mimics the way events are handled in the ETL tool.

In the top right-hand corner, the user is shown information on the number of events and aggregated windows. In **Figure 8**, there are 35 events and 35 aggregated windows – meaning that every event is aggregated. Clicking on an aggregated record expands the individual events below it. However, individual events that have not been aggregated may also be displayed directly.

MATCHING DETAILS			
126704 source events			
35 executables were created in .\Windows\SysWOW64 folder. ParentPath: .\Windows\SysWOW64			35 events / 35 Aggregated window
23:35:16	Executable file mfc40.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	
23:35:16	Executable file mfc40u.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	
23:35:33	Executable file ctl3d32.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	
23:35:33	Executable file msvcrt20.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	
23:35:33	Executable file crt.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	
23:36:39	Executable file aspnet_counters.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	
23:36:40	Executable file msvcp120_clr0400.dll was created in .\Windows\SysWOW64. Subory / Vznik executable	Aggregated window	

Figure 8 – Display of data from the selected time period with details of the aggregated record

Clicking on a non-aggregated record displays its details in the panel on the right-hand side; see **Figure 9**. In addition to the timestamp, the details include information on the data type, the user, and the template used to create the event. If the event belongs to a cluster (provided one has been added to the tool), the corresponding cluster will also be displayed.

If input data from the FCA is available, a **‘Find all similar’** button will appear in the record details. When activated, the system will search for and highlight all factor windows in which the record appears. This functionality allows you to quickly identify all occurrences of the record within the FCA analysis results and track their interrelationships.

SOURCE EVENT	
Find all similar	1 factor
Executable file mfc40.dll was created in .\Windows\SysWOW64.	
ID	files_exe_created_total_000001:files_exe_created_by_parentpath_000212:files_21504:000007
TIME	Aug 21, 23:35:16
METATABLE	Vznik executable
SITUATION	Executable file "{FileName}" was created in "{Parent Path}".
TYPE	Aggregated window
CLUSTER	Aug 21 overnight activity
HOST / USER	Unknown host / Unknown user
ARTEFACT	Subory
Executable file mfc40.dll was created in .\Windows\SysWOW64.	

Figure 9 – Details of the source event

At the bottom of the details view, detailed information is also displayed about the factor (see **Figure 10**) to which the selected record belongs. Information is available on the number of attributes that make up the factor, as well as the number of other records that fall under that factor.



Figure 10 – Example of factor details

3.6.4 Filtering events

ETL Timeline allows you to enable or disable filters at various levels based on your needs. Records can be selected by type – relating to files, processes, RDP logins, etc. A further level allows the selection of specific meta-records, e.g. archive creation or service startup; see **Figure 11**.

ARTEFACTS

- ☒ Subory
- ☒ Lokálne prihlásenia
- ☒ Služby
- ☒ Scheduled tasks
- ☒ Procesy
- ☒ RDP dest
- ☒ RDP src

METATABLES

- ☒ Archive Creation
- ☒ Vznik executable
- ☒ Prístup k suborom
- ☒ Timestamping suboru
- ☒ Vymazanie suboru
- ☒ Lokálne prihlásenia
- ☒ Service Start
- ☒ Service Creation
- ☒ Scheduled Tasks
- ☒ Spustenie procesu

Figure 11 – Example of filters

4 Forensic significance

The main purpose of the ETL Timeline tool is to consolidate diverse sources of forensic data into a single, clear overview. Windows generates activity logs across many independent sources – Event Log, prefetch files, the registry, scheduled tasks and other artefacts – each of which has its own structure, format and level of granularity. On its own, each of these sources provides only a partial view of what is happening on the system, and manually reviewing and comparing them is time-consuming and prone to errors. ETL Timeline consolidates these heterogeneous sources into a single, unified timeline, allowing analysts to work in one place rather than switching between multiple tools, tables and formats.

The key benefit lies not only in the merging of the data itself, but also in its organisation into meta-records (e.g. process creation, scheduled task creation, file download, etc.) and its aggregation into time windows. Instead of dealing with tens or hundreds of individuals, unprocessed records, the analyst works with a significantly smaller number of meaningful events, which directly reduces the volume of data that needs to be reviewed. From a forensic perspective, this reduction is crucial, as the risk of a relevant event being overlooked increases with large volumes of data. Aggregation significantly mitigates this risk without compromising the ability to drill down to the original source record if necessary.

Another significant benefit is the graphical representation of the timeline, which allows the analyst to identify clusters of activity, outages or sudden changes in the intensity of entries at a glance – phenomena that would remain hidden in the raw data. The tool further enhances this view by visualising the outputs of other analytical methods directly above the timeline – clusters, anomalies, attack stages and factors from formal conceptual analysis (FCA). As a result, the analyst does not have to search ‘blindly’ for suspicious sections across the entire dataset, but is actively guided to them: anomalies highlight time windows that differ most from normal activity and may indicate, for example, an unauthorised login or the launch of an unexpected process, clusters reveal groups of similar events (both natural and potentially suspicious), and FCA factors enable the ‘Find all similar’ function to quickly locate all records sharing common attributes. This allows the analyst to focus specifically on those parts of the data that genuinely warrant closer examination, rather than having to review the entire volume of records uniformly.

By combining these features – the consolidation of various sources into a single location, aggregation that reduces the volume of data to be reviewed, and the highlighting of anomalies, clusters and other outputs directly within the context of the timeline – ETL Timeline provides the analyst with an integrated and efficient view of the case, which speeds up analysis whilst reducing the risk of overlooking relevant events.

5 Conclusion

The ETL Timeline tool integrates the outputs of two independent analytical workflows into a single environment for joint interpretation. ETL Timeline links standardised meta-records created from EZ Tools outputs with analytical results derived from data processed by the Plaso tool. As a result, events extracted from forensic artefacts are displayed in a shared temporal context alongside analysis results, such as identified clusters, anomalies, attack phases and relationships between events. Such an integrated view eliminates the need to manually link results from multiple tools and enables the analyst to more effectively identify temporal and logical relationships between individual events, which significantly supports the digital forensic analysis process.

Opportunities for further development lie primarily in expanding the range of supported data sources and metadata types, and in integrating additional analytical methods, the outputs of which could be displayed on a shared timeline. Extending the tool to include real-time data processing and more advanced filtering options would also enhance its usefulness in resolving complex security incidents, including in real-world investigative environments.

6 Bibliography

- [1] Hudan Studiawan, F. B. (2025). Towards a standardised methodology and dataset for evaluating LLM-based digital forensic timeline analysis. *Forensic Science International: Digital Investigation*, Volume 54, Supplement. doi:<https://doi.org/10.1016/j.fsidi.2025.30198>
- [2] Frank Breiting, H. S. (2025). SoK: Timeline-based event reconstruction for digital forensics: Terminology, methodology, and current challenges. *Forensic Science International: Digital Investigation*, Volume 53, Supplement. doi:<https://doi.org/10.1016/j.fsidi.2025.301932>
- [3] ZIMMERMAN, Eric. *Eric Zimmerman Tools* [online]. [accessed 4 August 2026]. Available at: <https://ericzimmerman.github.io/>
- [4] React [online]. [accessed 4 August 2026]. Available at: <https://react.dev/>
- [5] npm, Inc., "About npm" [online]. [accessed 4 August 2026]. Available at: <https://docs.npmjs.com/about-npm>