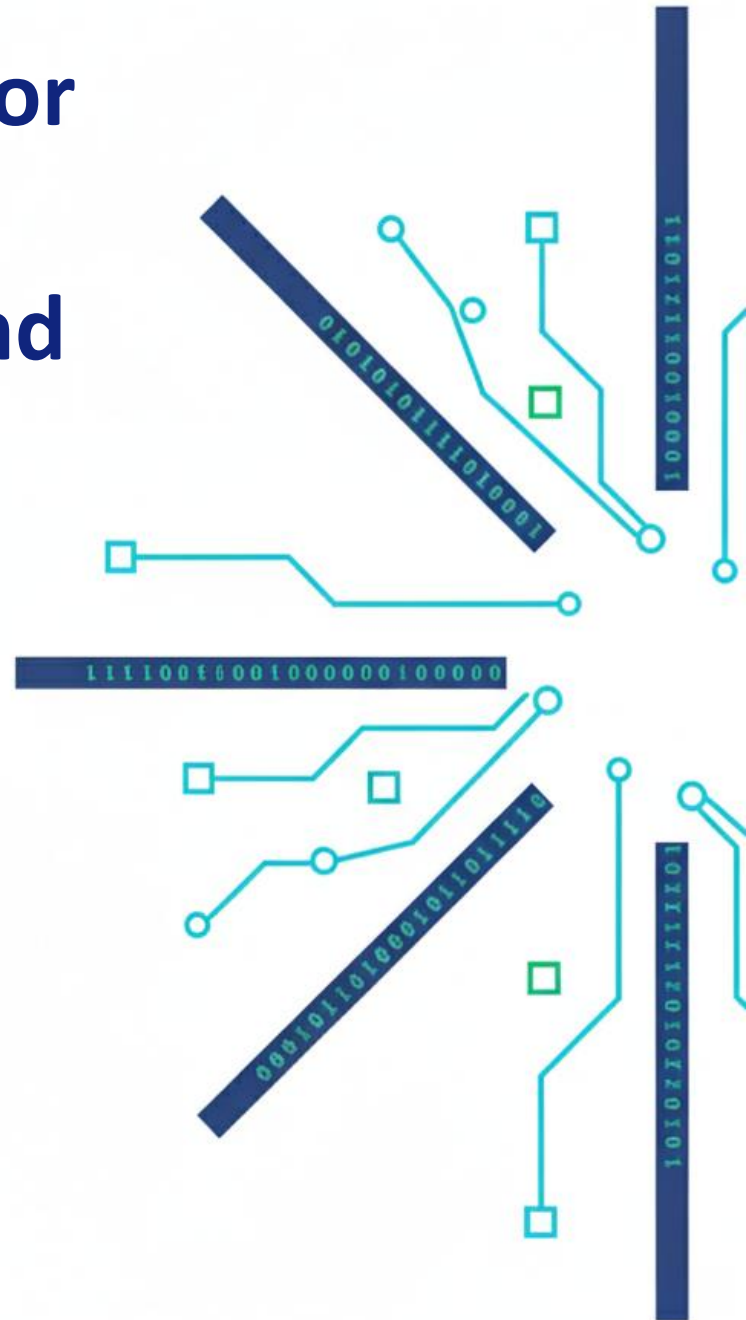


D21 – Method for determination of time frame and attack phases



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. 09I05-03-V02-00079.

Contents

1	Project description.....	2
2	Introduction	3
3	Time series.....	4
4	Method for determining the time frame and phases of an attack.....	5
4.1	<i>Inputs and outputs of the proposed method</i>	5
4.2	<i>Steps of the method</i>	5
4.3	<i>Generating time series from Windows operating system artefacts</i>	6
4.3.1	Selection of window width	7
4.3.2	A missing value is not zero	7
4.4	<i>Selection of relevant time series</i>	8
4.4.1	Coverage	8
4.4.2	Activity	8
4.4.3	Variability	9
4.4.4	Duplicate time series	9
4.5	<i>Detection of anomalies in individual time series</i>	10
4.5.1	Types of anomalies	10
4.5.2	Statistical methods	12
4.5.3	Model-based methods	13
4.5.4	Methods based on time series segments	14
4.5.5	Spectral methods.....	14
4.5.6	Mode change detection	14
4.5.7	Deep learning	15
4.6	<i>Aggregation of anomalies across time series</i>	15
4.6.1	Time tolerance ϵ	16
4.6.2	Voting	16
4.6.3	Threshold θ	16
4.6.4	Merging time windows into phases.....	16
4.7	<i>Determination of the incident time frame</i>	17
4.8	<i>Characterisation and interpretation of incident phases</i>	17
5	Conclusion	20
6	Bibliography	21
7	Appendix	24
7.1	<i>Descriptions of selected time series for EVT_X</i>	24
7.2	<i>Descriptions of selected time series for MFT and journal</i>	29
7.3	<i>Descriptions of selected time series for MFT</i>	34
7.4	<i>Descriptions of selected time series for the register</i>	37
7.5	<i>Descriptions of selected time series for SRUM</i>	37
7.5.1	AppResourceUseInfo.....	38
7.5.2	AppTimelineProvider	42

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. 09I05-03-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 traces. 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 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 unified methodologies and tools ensures that the processed digital traces 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 traces to be accepted as relevant evidence for investigations and court proceedings.

2 Introduction

In this part of the ADFIR project, we will focus on developing a method for the reconstruction and analysis of digital traces through time-series analysis. The strategic objective is to shift forensic analysis from manual log review to (semi-)automated determination of the time frame of the start of an attack and the phases of the attack.

During the first testing phase, it became apparent that data from the PLASO tool [1] (e.g. data from the 'The Stolen Szechuan Sauce' case [2], which we analyse in Output D14 – Synthetic dataset) contained several time frames with missing data. This significantly affects the quality and usability of the time series.

In the next phase, a set of extracted relevant artefacts was selected from various sources, such as EVTX files (Windows event logs), MFTE (Master File Table), SRUM (System Resource Usage Monitor) and the Windows operating system registry, which were used to create time series from selected simulated incidents (The Stolen Szechuan Sauce, Magnet_CTF_2019, Magnet_CTF_2020 and Magnet_CTF_2022 – for further details, see Output D14 – Synthetic dataset).

The generated time series will be used for analysis with the following objectives:

- a) **Determination of 'Time Zero' (the start of the incident)** – the automated generation of time series can enable the identification of the first anomalous activity. Rather than estimating when the breach might have occurred, the system will provide an objective data basis for defining the time window of the attack phase.
- b) **Support for automated anomaly detection using ML** – visual and statistical analysis of time series will serve as the input layer for machine learning (ML). The aim is to select models capable of detecting, in real time or retrospectively, anomalies in the frequency of system function calls, changes to file metadata or application activities. These anomalies represent signals marking the start of complex attacks that evade traditional rule-based detection.
- c) **Improving the effectiveness of Incident Response (IR)** – the method developed can enable incidents to be contextualised automatically. Instead of a list of thousands of event logs, the system generates a condensed timeline ('attack timeline') that links related events from various sources (e.g. process launch → registry entries → network communication). This significantly reduces response times and facilitates the work of forensic analysts.

3 Time series

A time series is commonly defined as a sequence of data values recorded at successive points in time. These values are typically arranged chronologically and often have equal time intervals, although in some contexts irregular intervals are also possible. From a mathematical point of view, time series theory is part of the theory of random processes [3]:

Let $(\Omega, \mathcal{A}, P)$ be a probability space and let the set T be some subset of the set of real numbers $\mathbb{R}$. Let $X(t)$ be, for every $t \in T$, a random variable defined on the same probability space $(\Omega, \mathcal{A}, P)$. The set $X(\cdot) = \{X(t), t \in T\}$ of random variables $X(t)$ is called a random process.

We call the random process $X(\cdot) = \{X(t), t \in T\}$ a continuous-time random process if the set T is an interval on the real line $\mathbb{R}$.

A random process $X(\cdot) = \{X(t), t \in T\}$ is called a discrete-time random process if the set T is a countable subset of $\mathbb{R}$. Discrete-time random processes are also called **time series**.

For simplicity, we shall use simplified notation, where we shall work with numerical values instead of random variables.

A time series can be represented as:

$$\mathbf{X} = \{x_{t_j} \mid j \in \mathbb{N}\},$$

where x_{t_j} is a numerical value recorded at time t_j . As follows from the definition, this also allows for uneven time intervals between individual values. For the condition of equal time intervals, we can write it as:

$$\mathbf{X} = \{x_i \mid i \in \mathbb{N}\},$$

where x_i is the numerical value recorded at time i . Alternatively, even more simply:

$$\mathbf{X} = x_0, \dots, x_t,$$

where x_i represent the values of the time series ordered by time with index $i \in \mathbb{N}$ and $t \in \mathbb{N}$ is the last time index.

Time series are used in many sectors for the collection and analysis of key data. These include, for example, meteorology (collection of measured values such as temperature, humidity, etc.), finance (share prices), energy and public services (electricity and water consumption), cyber security (number of failed login attempts to an SSH server, etc.), transport, healthcare, agriculture, sport and others.

4 Method for determining the timeframe and phases of an attack

This chapter presents the method that is the subject of this deliverable. The method is based on the assumption that an attacker's activity in the Windows operating system is not reflected in a single data source, but leaves a concurrent digital footprint across several relatively independent artefacts, e.g. in event logs, in file system metadata, in the registry, or in system resource usage statistics. Individually, such a digital trace may be inconspicuous and indistinguishable from normal operating system operation or normal user behaviour. However, if the same time interval appears anomalous simultaneously across several independent time series, the likelihood that this is a random fluctuation decreases significantly. The method therefore does not seek the 'largest anomaly', but rather the temporal coincidence of anomalies across sources.

4.1 Inputs and outputs of the proposed method

Input: a set of forensic artefacts obtained from a single system under investigation (EVTX, MFT/\$J, log, SRUM), processed into time series in accordance with Section 4.3, together the time of image acquisition.

The method's output comprises three components:

- **Incident time frame** – an estimate of the interval $[\hat{t}_0, \hat{t}_1]$, where $\hat{t}_0$ is the estimated start of the incident ('Time Zero') and $\hat{t}_1$ its end, together with an expression of the uncertainty of both boundaries.
- **Set of candidate attack phases** – an ordered sequence of disjoint time windows $F_1 < F_2 < \dots < F_K$, within which concurrent anomalous activity was recorded across multiple time series.
- **Characterisation of phases** – for each phase, a list of the time series that contributed most to it, a confidence score and a proposed interpretation (e.g. assignment to a tactic according to MITRE ATT&CK [4]).

4.2 Steps of the '' method

The method consists of six steps, which are covered in the following sub-chapters:

- 1) **Step 1 – Creating time series from Windows operating system artefacts** – a set of time series is created from the forensic artefacts on a common timeline.

- 2) **Step 2 – Selection of relevant time series** – from the set created, series that do not provide useful information are filtered out, and redundant series are clustered and weighted.
- 3) **Step 3 – Detection of anomalies in individual time series** – several anomaly detection methods are applied independently to each time series; their scores are normalised, thresholded and combined.
- 4) **Step 4 – Aggregation of anomalies across time series** – anomalies are expanded by an ε -neighbourhood, and time windows in which at least a θ -proportion of the time series are anomalous are determined by voting. These windows represent candidate attack phases.
- 5) **Step 5 – Determining the time frame of the incident** – the minimum and maximum values of the set of confirmed anomalous times determine the start and end of the cyber security incident.
- 6) **Step 6 – Characterisation and interpretation of incident phases** – dominant time series are assigned to each phase and their forensic significance is proposed.

4.3 Generating time series from Windows operating system artefacts

Records in artefacts are generated at irregular intervals τ , whenever something occurs in the system. Such a source corresponds to a time series with irregular intervals $\{x_{t_j}\}$. However, most anomaly detection methods, and in particular the comparison of multiple time series, require equal intervals and common time indices. Therefore, the time axis is divided into windows of width Δ , and a single value x_i is calculated for each window i .

Each time series is defined by two properties:

- **a filter** – which records are taken into account (e.g. 'channel = Application'),
- **aggregation** – how the records within a single window are processed (e.g. sum or average is applied).

The value x_i = the result of aggregating all records that have passed through the filter and whose time τ falls within window i .

The result is a time series $X = x_0, \dots, x_t$ with a uniform time interval Δ , exactly as defined in Chapter 3. Let us denote $X^{(m)}$ as the m th time series from the set of time series we have created, so that $x^{(m)}_i$ is the value of the m th time series in time window i . All series have the same indices $0, \dots, t$, so that the value $x^{(m)}_i$ in different series always refers to the same time window.

Aggregations used:

- **count** – the number of records occurring in a given time window (most commonly for EVTX, MFT and operating system registry forensic artefacts),
- **sum** – the sum of numerical values, e.g. the number of bytes transferred within a given time window (typically the SRUM forensic artefact),
- **the number of distinct values** – e.g. how many different EventIDs or users appeared within a time window. Such a sequence indicates when an attacker is ‘expanding their scope’ (reconnaissance, lateral movement),
- **maximum** – the highest value of a single record within the time window, e.g. the largest volume of data read or written by a single application within that time window. Unlike the sum, this aggregation does not depend on the number of records in the time window; therefore, it highlights a single large-scale operation (e.g. bulk reading of files prior to exfiltration) even at a time when there is a lot of minor routine activity taking place in the system, which would otherwise mask such an operation in the sum.

For example, ``ts_{'Channel': 'Application'}.csv`` is the number of records from the Application channel in each window. The file `ts_ [{'SidType': 'LogonId'}, 'BackgroundBytesRead']_value_BackgroundBytes-Read.csv` is the sum of bytes read in the background for LogonId-type records. A list of the selected time series is provided in the appendix.

4.3.1 Choosing the window width

A window that is too short will produce a series full of zeros, from which nothing can be reliably inferred. A window that is too long will obscure brief attacker activity amidst normal traffic. The appropriate width depends on the amount of data collected. The smaller the time window, the more precisely the forensic analyst can determine the timeframe of the attack. On the other hand, this can result in a large number of timeframes when dealing with a large disk image (several months’ worth of data).

We recommend running the method for three window widths (10, 30 and 60 minutes), taking into account the overall timeframe of the analysed data.

4.3.2 A missing value is not zero

Zero ($x_i = 0$) means that the forensic artefact was recording data within time window i , but no event occurred in the system that would have triggered a write operation. A missing value means that we have no information about time window i , as the forensic artefact contains no records for this period, e.g. the period prior to the oldest preserved record following EVTX log

rotation, or a period outside the data retention period in the SRUM database. If these two cases are confused, two errors may occur. Either a false anomaly arises at the boundary of the source's availability, i.e. in the time window in which the forensic artefact's records begin or end, or an actual gap in the log records is overlooked (e.g. following its deletion).

Therefore, for each series, **the coverage $C^{(m)}$** is recorded **for the time series $X^{(m)}$** , i.e. the set of time-window indices for which the source has data: $C^{(m)} = \{i_first, \dots, i_last\}$, where i_first is the window containing the oldest record and i_last the window containing the most recent record of the artefact. Coverage is determined by the extent of the artefact's entire , not by where the time series has non-zero values. For windows within the coverage ($i \in C^{(m)}$), the value x_i is valid even if it is zero; outside the coverage ($i \notin C^{(m)}$), it is marked as missing.

$$C^{(m)} = \{i : i_first \leq i \leq i_last\},$$

where i_first is the index of the window containing the oldest artefact record, and i_last is the index of the time window containing its most recent record. The coverage is determined by the scope of the entire forensic artefact from which the time series was derived (e.g. the entire security log), not by the windows in which the series itself has non-zero values.

We recommend creating a time series **titled 'number of sources with data in time window i'**. A sudden loss of records in an otherwise active channel is in itself a strong indicator (e.g. log deletion, EventID 1102).

4.4 Selection of relevant time series

The selection of time series should be based on clear and repeatable rules, not merely on the judgement of the security analyst. The selection of suitable time series for the chosen Windows operating system forensic artefacts is set out in the appendix to this document.

4.4.1 Coverage

A time series must contain data for at least a certain proportion of the windows 0, ..., t (recommended: at least 30 per cent). Time series with low coverage are not discarded, but in windows where they lack data, they are not counted in the vote.

4.4.2 Activity

A sequence with a few non-zero values x_i does not allow us to estimate what is normal. Each of its non-zero values would automatically appear to be an anomaly.

Very sparse sequences are processed differently: each of their non-zero windows is directly considered **an event of interest**. A typical example is changes in automatic start-up keys (ASEP), where the change itself is already forensically significant.

4.4.3 Variability

A time series whose values remain virtually unchanged carries no information and is therefore discarded. A measure of variability is the coefficient of variation, i.e. the ratio of the standard deviation to the mean of the series' values over its range $C^{(m)}$ (the recommended threshold is given in Table 1).

Another useful indicator is the ratio of the variance to the mean (dispersion index). If this is approximately 1, the values of the time series behave like the counts of independent, randomly occurring events, i.e. they follow a Poisson distribution, for which the variance equals the mean. If the ratio is much greater than 1 (excess variance), events occur in clusters and the time series is 'burst-like'. For such a time series, a negative binomial distribution model (Section 4.5.3) is more appropriate before applying statistical methods [5].

4.4.4 Duplicate time series

A large number of time series can be generated from Windows operating system forensic artefacts, which are very similar and redundant in terms of what they represent. For example, from an SRUM forensic artefact, it is possible to generate various combinations of account type and type of bytes transferred.

Since all time series share the same indices 0, ..., t, they can be compared directly, value by value. Solution for duplicate time series:

1. identify which time series behave almost identically (high correlation, e.g. above 0.8),
2. such time series are grouped together,
3. each group of nearly identical time series has a combined weight of one vote. Either all time series are retained in the group and the vote is distributed evenly amongst them ($w_m = 1/|G|$, where $|G|$ is the number of series in the group), or only one representative is retained, e.g. the series with the greatest coverage, with a weight of $w_m = 1$, and the other time series are discarded.

Following this adjustment, **the effective number of independent time series**, $N_{\text{eff}} = \sum_m w_m$, i.e. the sum of the weights of all series, can be determined. Since each group of nearly identical series has a combined weight of one vote, N_{eff} expresses how many mutually independent votes the voting process actually has at its disposal. In step 4, a threshold is set

relative to N_{eff} , which determines how many independent series must agree for the window to be considered anomalous.

The selection of time series therefore takes place in four steps. First, the coverage, activity and variability of each series are checked. Series that meet the criteria are retained, whilst very sparse series are classified as point indicators. Next, nearly identical time series are grouped together and assigned weights. Finally, the overall influence of any single source is limited. The criteria and their recommended values are summarised in Table 1.

Criterion	What is measured	Recommended value	Purpose
Coverage	Proportion of windows with data	$\geq 30 \%$	Time series below the threshold are not excluded, but are simply flagged
Activity	number of non-zero windows	≥ 100	sufficient data to estimate normal behaviour; time series below the threshold are processed as a point indicator
Variability	measure of variability	> 0.05	exclude almost constant time series
Duplication	correlation	0.9	group nearly identical time series and prevent one source from dominating

Table 1 – Criteria for selecting relevant time series

4.5 Detection of anomalies in individual time series

4.5.1 Types of anomalies

We distinguish between four types of anomalies [6,7], each corresponding to a different attacker behaviour and requires a different class of methods:

- **Point anomaly** – a single value significantly outside the expected range. In a forensic context, this corresponds to a one-off event, such as the launch of a tool, a write to the autostart key, or a one-off data export. Figure 1 illustrates an example of a point anomaly.
- **Contextual anomaly** – a value that is normal in itself but unusual in its context. A typical example is activity at a time when the system is not usually operational (at night, at the weekend). Without modelling daily and weekly seasonality, this type of anomaly cannot be detected. For this reason, it is the most common type in targeted attacks.

- **Collective anomaly (anomalous subsequence)** – individual values are within the norm, but their sequence is not. This corresponds to prolonged activity, such as data exfiltration split into smaller transfers, slow scanning, or file encryption.
- **Change point** – a permanent shift in the mean, variance or periodicity of a time series. This corresponds to the installation of a persistent mechanism, a change in system configuration or the disabling of a security service. The attack phase is precisely the interval between two change points. Figure 1 shows an example of this anomaly.

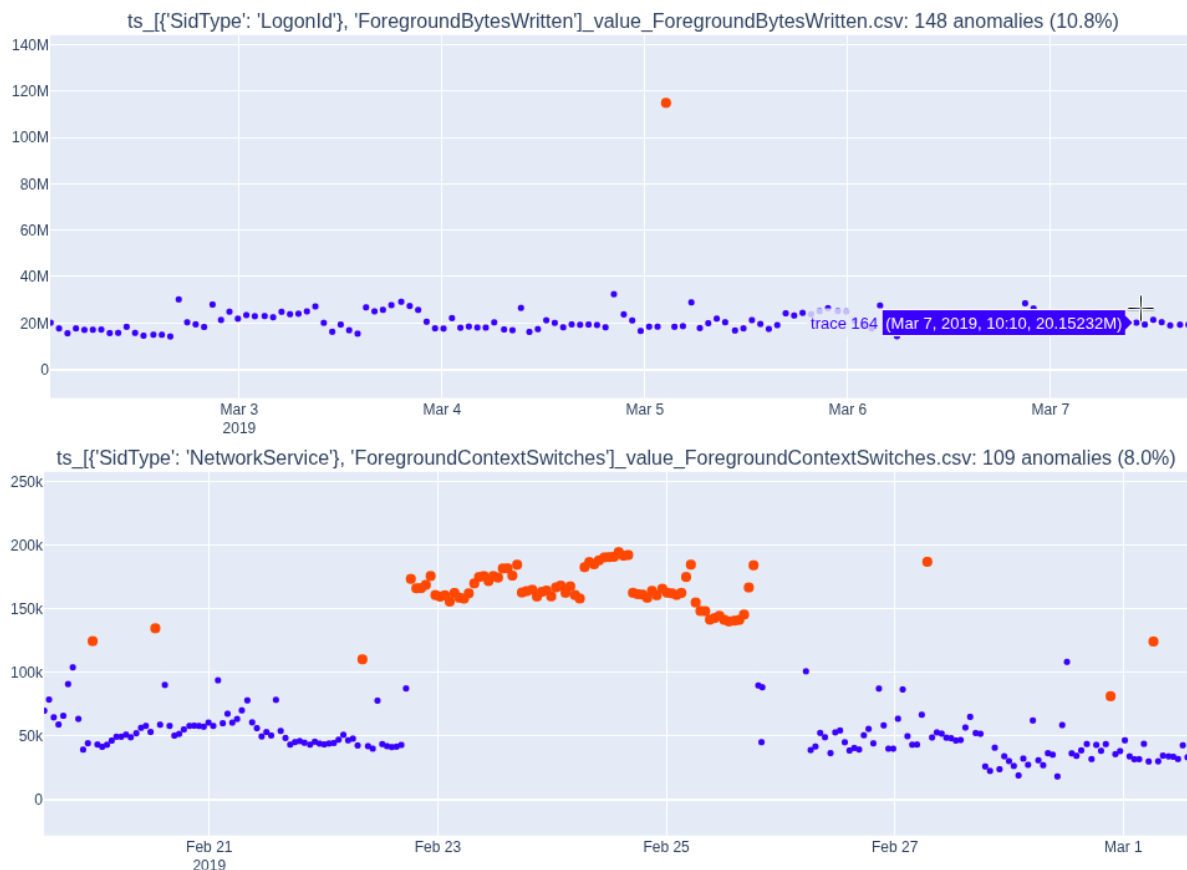


Figure 1 – An example of a point anomaly (top) and an example of a change point (bottom).

We then move on to the implementation of automated methods for anomaly detection. We test several algorithms with varying computational complexity and sensitivity [8], whilst their accuracy, robustness and ability to localise the point of origin of an attack or various suspicious locations are evaluated manually. When carrying out a quantitative evaluation of labelled cyber security incidents, it is necessary to use interval metrics [9], rather than point metrics with a ‘point-adjust’ correction, which overestimates the results [10]. This process should enable the selection of the most appropriate method for distinguishing normal system load from an attacker’s activity.

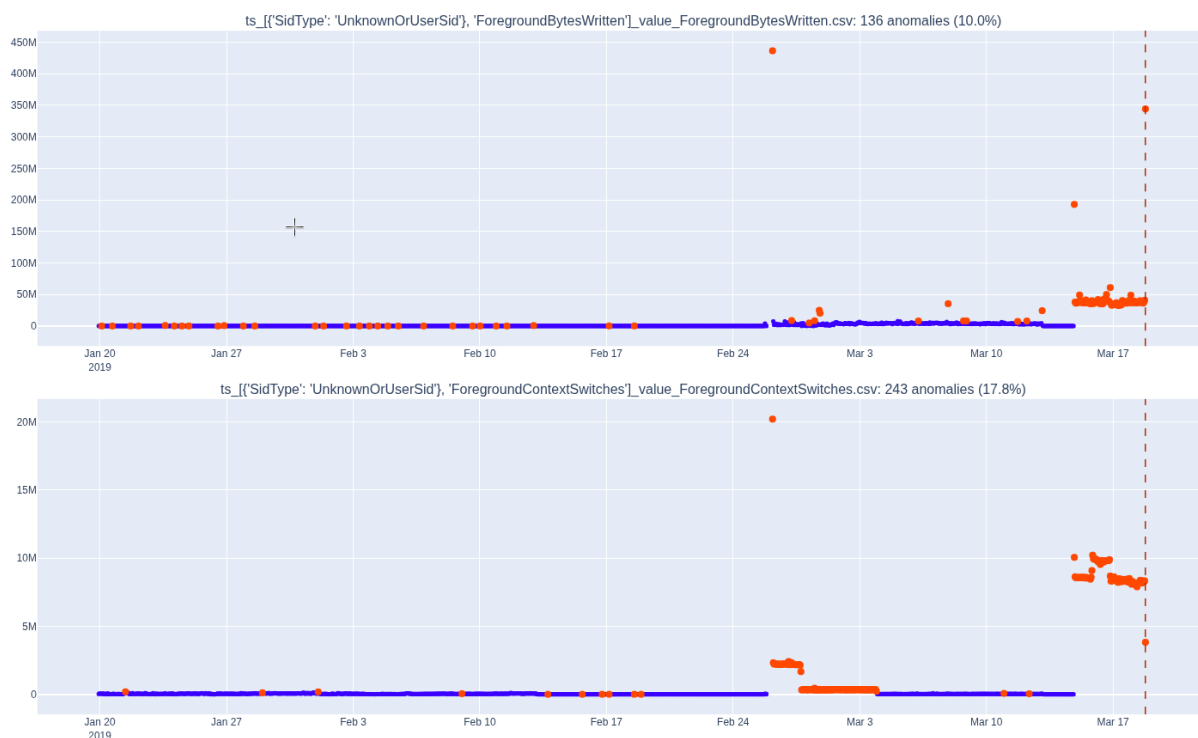


Figure 2: Example of anomaly analysis for the 'Application Resource Usage' table of the SRUM forensic artefact across multiple time series within a single incident.

Another approach was based on analysing the detected anomalies across multiple time series within a single incident simultaneously. This analysis enables us to monitor potential attacker activity from various perspectives (network, disk, system, etc.) and thus better estimate, for example, the starting point of the attack. An example can be seen in Figure 2. We will discuss this analysis in more detail in Step 4.

An important part of the process is the manual comparison of time series and detected anomalies within a single type of time series across various simulated incidents, which can help identify interesting events that warrant general attention.

4.5.2 Statistical methods

Statistical methods are quick and easy to explain, which is important in forensic practice.

The first example of a statistical method is **the robust z-score [11]**. It measures by how many standard deviations (MAD) the value x_i is distant from the median of the series. A value is considered an outlier if the distance exceeds a chosen threshold (typically 3 to 5). If more than half of the values are zero, the typical deviation is zero and the method does not work. Such series are classified as point indicators.

Another example is **the interquartile range (IQR)**. A value is considered an outlier if it lies too far above the upper quartile or below the lower quartile. However, numerical data are skewed (they have a long tail towards the top) and a standard IQR would report many false positives in such cases. Therefore, **a modified box plot [12]** is used, which shifts the boundaries according to the degree of skewness.

Another example of a statistical method is **the moving window**. Each value x_i is compared with the median of the surrounding values. Important: if the comparison interval also includes the value x_i itself, the outlier will influence the comparison. Therefore, x_i should **only** be compared **with the preceding values** x_{i-1} , x_{i-2} , ..., which also corresponds to continuous monitoring.

The most recent examples of statistical methods are **GESD [13]** and **S-H-ESD [14]**. GESD repeatedly identifies the most extreme value, tests it and removes it, thereby gradually determining how many anomalies the series contains. This solves the problem where one anomaly 'hides' another. S-H-ESD does the same, but only after removing the daily and weekly rhythms.

4.5.3 Model-based methods

In this group of anomaly detection methods, the model estimates what values a random variable $X(i)$ should take during normal operation, and **an anomaly is an observed value x_i that deviates significantly from this expectation**.

Forecasting models (SARIMA [15], Holt-Winters [16], Prophet [17]) predict the value at time i based on previous values and also specify the range within which it should fall. A value outside this range is considered an anomaly. The advantage is that they naturally capture both the trend and the rhythm; the disadvantage is the need for a longer time series.

Models for counts (Poisson, negative binomial). For event counts, the most suitable model is one that estimates **the expected number of events in window i** based on the hour of the day, the day of the week and the previous value x_{i-1} . The anomaly score then expresses **the probability that, under normal operating conditions, $X(i)$ would have taken on a value at least as high as the observed x_i** .

Advantages:

- the result is directly a probability (p-value); no further adjustment is required,
- it works correctly even with zeros and small sample sizes,
- for 'burst' data series, it is sufficient to replace the Poisson model with the negative binomial model.

4.5.4 Methods based on sequence segments

In these anomaly detection methods, instead of individual values x_i , **short segments** of the time series are compared, e.g. $x_i, x_{i+1}, \dots, x_{i+5}$. As a result, these methods detect collective anomalies.

- **Isolation forest [18]** – searches for segments that can be easily ‘separated’ from the rest. It must be applied to segments, not individual values; otherwise, it will only find what and simpler methods can also detect. Its ‘expected anomaly proportion’ parameter directly determines the number of detections, so it should not be chosen arbitrarily.
- **LOF [19] and kNN [20]** – compare whether a segment is in a ‘sparse’ region relative to its surroundings. Suitable when a time series has multiple regimes (working and non-working hours).
- **Matrix Profile [21]** – for each segment, it finds the most similar other segment in the time series. A segment that **has no similar counterpart** is the most unusual (a so-called ‘discord’). The main parameter is the segment length, which should correspond to the expected attack duration; we recommend trying out several different lengths. It is computationally more demanding, but faster variants exist (STOMP [22], SCRIMP++ [23]).

4.5.5 Spectral methods

Spectral Residual [24] decomposes the series into frequencies, removes the ‘expected’ component and subsequently highlights the windows that do not fit within it. It is fast, requires no training and is suitable for time series with sharp peaks. Its weakness is poorer accuracy at the start and end of the time series (around x_0 and x_t), i.e. precisely where we are looking for the start of an incident. Therefore, the time series is artificially extended at the edges.

4.5.6 Mode change detection

Point-based methods answer the question ‘which value x_i is unusual’. Mode change detection answers the question ‘**from which index did the behaviour of the process change permanently**’, which directly corresponds to the concept of an attack phase.

- **PELT [25], binary segmentation** – divides the sequence $x_0, \dots, x_t$ into segments with similar behaviour. Each additional boundary is ‘penalised’ to prevent the sequence from being unnecessarily fragmented. The output consists directly of the indices of candidate phase boundaries.

- **BOCPD [26]** – continuously monitors how long the process behaves in the same way. If this duration suddenly drops, a change has occurred. The output is the probability of a change in each window.
- **CUSUM [27], Page-Hinkley, ADWIN** – simple running tests that detect even slow shifts in the level, which point-based methods overlook.
- **SAND [28]** – groups similar segments and continuously adapts to changes, thereby combining the search for unusual segments with the monitoring of changes.

Mode changes should be entered into the voting as a separate event type (a boundary, not a segment). If a change occurs in multiple rows within the same or a neighbouring window, this is a very strong indicator of a transition between phases.

4.5.7 Deep learning

Neural networks (LSTM-AE [29], USAD [30], Anomaly Transformer) learn the normal behaviour of a time series and identify anomalies based on how poorly they can ‘reconstruct’ the observed segment.

However, the specialist literature shows that, for individual time series and without labelled data, **they generally do not outperform simple statistical methods**; they are difficult to train and their results are hard to interpret. In a forensic context, interpretability is essential. We therefore recommend them as a supplementary, rather than a primary, approach. Their contribution should subsequently be compared with that of simple methods.

4.6 Aggregation of anomalies across time series

Whilst the previous chapter dealt with each time series in isolation, this step combines the information. The underlying idea is that if the same time interval appears to be anomalous in several mutually independent time series, it is highly likely to be a genuine event in the system, rather than noise from a single source. We refer to such an interval as a candidate attack phase.

The input for the aggregation step consists of the binary indicators $a^{(m)}_i \in \{0, 1\}$ from the previous step 3, where $a^{(m)}_i = 1$ if time window i in series m is anomalous. They are defined only for windows $i \in C^{(m)}$, where $C^{(m)}$ denotes the set of time windows belonging to time series m . Furthermore, the weights w_m from step 2 and the point indicators, for which every non-zero window has $a^{(m)}_i = 1$, are also input. A regime change is recorded as $a^{(m)}_i = 1$ in the window of the detected boundary.

For a more detailed description of aggregation, reference may be made to Output D17 – Model for the aggregation and interconnection of digital traces.

4.6.1 Time tolerance ϵ

The same attacker's activity will not manifest within the same time window across different forensic artefacts. An EVTX record is generated almost immediately, the SRUM forensic artefact stores aggregated data approximately once an hour, and a log entry in \$J or a timestamp change in the MFT may occur with a delay.

Let $A^{(m)} = \{j \in C^{(m)} : a^{(m)}_j = 1\}$ be the set of anomalous time windows of order m . Each anomaly is extended by ϵ_m time windows on both sides (ϵ -neighbourhood):

$$\tilde{a}^{(m)}_i = 1 \text{ if there exists a } j \in A^{(m)} \text{ such that } |i - j| \leq \epsilon_m; \text{ otherwise, } \tilde{a}^{(m)}_i = 0.$$

The parameter $\epsilon_m \in \{0, 1, 2, \dots\}$ is specified as the number of time windows according to the time granularity of the source of order m . For SRUM series, $\epsilon_m = \lceil 60 \text{ min} / \Delta \rceil$ (i.e. 6 when $\Delta = 10 \text{ min}$); for other sources, $\epsilon_m = 1$. If ϵ is too large, neighbouring phases are merged into a single phase.

4.6.2 Voting

For each time window i , the weighted proportion of time series that have flagged it as anomalous is calculated as:

$$V_i = \sum_{m: i \in C^{(m)}} w_m \cdot \tilde{a}^{(m)}_i / \sum_{m: i \in C^{(m)}} w_m, V_i \in [0, 1].$$

The sums include only those series that have data within time window i . A time series outside its coverage area therefore votes neither for nor against.

4.6.3 Threshold θ

A time window i is a candidate if $V_i \geq \theta$, where $\theta \in (0, 1]$. The product $\theta \cdot N_{\text{eff}}$ specifies how many independent time series must agree for the time window to be considered a candidate.

Even independent time series occasionally coincide by chance. For this reason, a p -value is determined for each time window. If the detectors in step 3 provide p -values $p^{(m)}_i$, these are combined using the Fisher method [31] or the weighted Stouffer method [32]:

4.6.4 Merging time windows into phases

Time windows that satisfy $V_i \geq \theta$ and remain significant even after correction are merged into phases. Significant time windows separated by at most g time windows belong to the same

phase. Equivalently, DBSCAN clustering [33] can be applied to window indices with the parameter $\text{eps} = g$.

4.7 Determining the incident time frame

The time frame of a cyber security incident is determined by the minimum and maximum values of the set of anomalous times. This chapter formalises the procedure described above and adds two elements that are essential for forensic applications: a robust variant and the expression of uncertainty.

- **Start of the incident i_Z** = the index of the first window with a **confirmed** anomaly.
- **End of the incident: i_K** = the index of the last window with a **confirmed** anomaly.

The estimate of the start of the incident is $\hat{t}_0 = \tau_0 + i_Z \cdot \Delta$ (the start of the first confirmed time window) and the estimate of the end is $\hat{t}_1 = \tau_0 + (i_K + 1) \cdot \Delta$ (the end of the last confirmed time window), where τ_0 is the start of the timeline (Chapter 4.3).

A confirmed anomaly is one that belongs to one of the significant phases (with a tolerance of ϵ). It would be a mistake to use all raw anomalies: a single false positive at the start of a time series (and in many time series, dozens of such false positives occur) would shift the estimated start time back by weeks.

At the same time, a more robust estimate should also be provided: instead of the very first and last confirmed windows, the first and last 5 per cent of confirmed anomalies are omitted. If the two estimates differ significantly, the timeframe is determined by only a few isolated findings and must be verified manually.

4.8 Characterisation and interpretation of incident phases

The final step of the method converts the time windows into a forensically interpretable output. Without this, the result is merely a list of intervals, which does not tell the analyst what happened within them.

For each phase, we determine **what proportion of its anomalous windows originates from individual time series** (taking their weights into account). By sorting these, we obtain a list of the time series that contribute most to the phase. **A profile of the sources** is also useful – that is, how much of the phase is attributable to EVTX, MFT, the register and SRUM – because the significance of a phase is usually determined precisely by the combination of sources. Table 2 lists typical patterns that can be used as a starting knowledge base.

Dominant time series in the phase	Specific time series in the aggregation	Proposed interpretation	Tactics from the MITRE ATT&CK framework
EVTX Security: an increase in failed logins, followed by a successful login	ts_{'EventId_audit_logon_logoff'} (4625 → 4624); ts_{'EventId_audit_account_logon'} (4768, 4769, 4770/4771); ts_{'Channel': 'Security'}; ts_{'Channel': 'Microsoft-Windows-RemoteDesktopServices-RdpCoreTS-Operational'}; ts_{'Channel': 'Microsoft-Windows-TerminalServices-RemoteConnectionManager-Operational'}	brute-force or credential stuffing attack; remote access via RDP channels	Credential Access, Initial Access
EVTX: increase in process creation, WMI activity OS registry: changes to Services	ts_{'EventId_audit_detailed_tracking'} (4688, 4689); ts_{'ProcessId': N} (new PID series); ts_{'Channel': 'Microsoft-Windows-WMI-Activity-Operational'}; ts_{'Category': 'Services'}; ts_{'Description': 'Services'}; ts_{'HiveType': 'System'} (MC_Output)	launching tools and establishing persistence	Execution, Persistence
ASEP log: changes outside normal hours EVTX: new services, scheduled tasks SRUM: activity under SYSTEM	ts_{'HiveType': 'Software'}; ts_{'HiveType': 'System'} (ASEPs); ts_{'Category': 'Services'}; ts_{'Channel': 'System'} (7045); ts_{'Channel': 'Microsoft-Windows-TaskScheduler-Maintenance'}; ts_{'EventId_audit_privilege_usage'} (4673, 4674); ts_{'SidType': 'LocalSystem'}, 'BackgroundBytesWritten'; ts_{'SidType': 'LocalSystem'} (AppTimelineProvider)	setting up automatic execution with elevated privileges	Persistence, Privilege Escalation
EVTX: security mitigations, AppContainer SRUM: applications outside the normal SidType	ts_{'Channel': 'Microsoft-Windows-Security-Mitigations-KernelMode'}; ts_{'Channel': 'Microsoft-Windows-Security-LessPrivilegeAppContainer-Operational'}; ts_{'Channel': 'Microsoft-Windows-AppModel-Runtime-Admin'}; ts_{'SidType': 'UnknownOrUserSid'}, 'ForegroundContextSwitches']	attempt to exploit a vulnerability, bypass defences	Privilege Escalation, Defence Evasion
EVTX: enumeration of accounts, AD objects, WMI queries	ts_{'Channel': 'Microsoft-Windows-WMI-Activity-Operational'}; ts_{'EventId_audit_ds_access'} (4661, 4662); ts_{'EventId_audit_account_management'} (4720, 4722, 4728, 4732); ts_{'EventId_audit_object_access'} (5140); ts_{'Channel': 'Microsoft-Windows-GroupPolicy-Operational'}	system, account and network audit	Discovery
EVTX: remote logins, RDP sessions, access to admin shares	ts_{'EventId_audit_logon_logoff'} (4624 type 3/10, 4648); ts_{'Channel': 'Microsoft-Windows-TerminalServices-RemoteConnectionManager-Operational'}; ts_{'Channel': 'Microsoft-Windows-RemoteDesktopServices-RdpCoreTS-Operational'}; ts_{'Channel': 'Microsoft-	lateral movement within a computer network	Lateral Movement

	Windows-Winlogon-Operational'}; ts_{'EventId_audit_object_access'} (5140, 5145)		
• MFT/USN: bulk file creation in temporary and user directories	ts_{'UpdateReasons': 'FileCreate'}; 'FileCreate-Close'; 'DataExtend-FileCreate'; 'DataExtend-FileCreate-Close'; ts_{'FileAttributes': 'Archive-Temporary'}; 'Hidden-Archive-Temporary'; ts_Created0x10_{'SiFlags': 'Archive'}	Transfer of tools to the system, data preparation	Command and Control, Collection
• SRUM: sharp increase in bytes read in the foreground/background • MFT: bulk access • EVTX: access to shared resources	ts_['{Appld': N}, 'ForegroundBytesRead']; 'BackgroundBytesRead'; ts_['{SidType': 'LgonId'}, 'BackgroundBytesRead']; ts_['{SidType': 'UnknownOrUserSid'}, 'BackgroundBytesRead']; ts_{'Appld': N} (AppTimelineProvider); ts_LastAccess0x10_{'SiFlags': 'Archive'}; ts_{'UpdateReasons': 'DataExtend-Close'} (archive growth); ts_{'EventId_audit_object_access'} (5140, 5145)	data collection, transmission of data outside the system	Collection, Exfiltration
• MFT/USN: bulk modification, overwriting, renaming and deletion of files • SRUM: high write activity and context switches	ts_{'UpdateReasons': 'DataOverwrite'}; 'DataOverwrite-Close'; 'DataOverwrite- DataExtend-DataTruncation'; 'DataTruncation'; 'RenameNewName'; 'RenameOldName'; 'FileDelete-Close'; ts_LastModified0x10_{'SiFlags': 'Archive'}; ts_['{Appld': N}, 'ForegroundBytesWritten']; 'ForegroundContextSwitches'; ts_{'Channel': 'Microsoft-Windows-Storage- Storport-Operational'}	data encryption or corruption (ransomware)	Impact
• EVTX: loss of records, log deletion, audit policy change, decrease in Defender activity	ts_{'EventId_audit_non_audit'} (1100, 1102, 1104); ts_{'EventId_audit_policy_change'} (4715, 4719); ts_{'Channel': 'Microsoft-Windows- Windows Defender-Operational'} (drop / 5001, 5010); ts_{'Channel': 'Security'} (gap in the series); ts_{'UpdateReasons': 'SecurityChange'}; 'FileDelete-Close' (deletion of tools)	covering digital traces, disabling security protection	Defence Evasion

Table 2 – Patterns for interpreting attack phases

5 Conclusion

This output presents a method for the (semi-)automated determination of the timeframe and phases of a cyber security incident in the Windows operating system using time series analysis. The method is based on the assumption that an attacker's activity leaves a concurrent digital trail across several relatively independent artefacts, particularly in event logs (EVTX), file system metadata (MFT, \$J), the registry and the SRUM database. Rather than searching for a single, most prominent anomaly, it therefore focuses on the temporal correlation of anomalies across sources.

The proposed procedure consists of six steps: creating time series with equal time intervals from irregular artefact records, selecting relevant series and suppressing the influence of duplicate series, detecting anomalies individually in each series, voting across series with a time tolerance, determining the start and end of an incident, and interpreting individual phases, including their assignment to MITRE ATT&CK tactics. Emphasis is placed on features important for forensic use, in particular controlling the false positive rate, verifying that the match of anomalies is not coincidental, expressing the uncertainty of the estimate, and ensuring the traceability of each phase back to the original records.

The output of the method is a condensed incident timeline, which allows the analyst to focus on a few suspicious time periods rather than thousands of records. The results of the method should be understood as a hypothesis intended for verification by conventional forensic analysis, not as a standalone conclusion. The method has known limitations, notably sensitivity to the manipulation of timestamps, a limited history of artefacts, and slow attacks that are more difficult to detect.

6 Bibliography

- [1] log2timeline. (n.d.). *Plaso (log2timeline)* [Computer software]. GitHub. <https://github.com/log2timeline/plaso>
- [2] DFIR Madness. (2020). *The stolen Szechuan sauce* [Data set]. <https://dfirmadness.com/the-stolen-szechuan-sauce/>
- [3] Brockwell, P. J., & Davis, R. A. (2016). *Introduction to time series and forecasting* (3rd ed.). Springer. <https://doi.org/10.1007/978-3-319-29854-2>
- [4] Strom, B. E., Applebaum, A., Miller, D. P., Nickels, K. C., Pennington, A. G., & Thomas, C. B. (2018). *MITRE ATT&CK: Design and philosophy* (Technical report). The MITRE Corporation.
- [5] Anscombe, F. J. (1948). The transformation of Poisson, binomial and negative-binomial data. *Biometrika*, 35(3/4), 246–254. <https://doi.org/10.2307/2332343>
- [6] Chandola, V., Banerjee, A., & Kumar, V. (2009). Anomaly detection: A survey. *ACM Computing Surveys*, 41(3), Article 15. <https://doi.org/10.1145/1541880.1541882>
- [7] Blázquez-García, A., Conde, A., Mori, U., & Lozano, J. A. (2021). A review on outlier/anomaly detection in time series data. *ACM Computing Surveys*, 54(3), Article 56. <https://doi.org/10.1145/3444690>
- [8] Schmidl, S., Wenig, P., & Papenbrock, T. (2022). Anomaly detection in time series: A comprehensive evaluation. *Proceedings of the VLDB Endowment*, 15(9), 1779–1797. <https://doi.org/10.14778/3538598.3538602>
- [9] Tatbul, N., Lee, T. J., Zdonik, S., Alam, M., & Gottschlich, J. (2018). Precision and recall for time series. In *Advances in Neural Information Processing Systems 31* (pp. 1920–1930). Curran Associates.
- [10] Kim, S., Choi, K., Choi, H.-S., Lee, B., & Yoon, S. (2022). Towards a rigorous evaluation of time-series anomaly detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(7), 7194–7201. <https://doi.org/10.1609/aaai.v36i7.20680>
- [11] Iglewicz, B., & Hoaglin, D. C. (1993). *How to detect and handle outliers*. ASQC Quality Press.
- [12] Hubert, M., & Vandervieren, E. (2008). An adjusted boxplot for skewed distributions. *Computational Statistics & Data Analysis*, 52(12), 5186–5201. <https://doi.org/10.1016/j.csda.2007.11.008>
- [13] Rosner, B. (1983). Percentage points for a generalised ESD many-outlier procedure. *Technometrics*, 25(2), 165–172. <https://doi.org/10.1080/00401706.1983.10487848>
- [14] Hochenbaum, J., Vallis, O. S., & Kejariwal, A. (2017). *Automatic anomaly detection in the cloud via statistical learning* (arXiv:1704.07706). arXiv. <https://doi.org/10.48550/arXiv.1704.07706>
- [15] Kumari, S., & Muthulakshmi, P. (2024). SARIMA model: an efficient machine learning technique for weather forecasting. *Procedia Computer Science*, 235, 656–670.

- [16] Winters, P. R. (1960). Forecasting sales by exponentially weighted moving averages. *Management Science*, 6(3), 324–342. <https://doi.org/10.1287/mnsc.6.3.324>
- [17] Rafferty, G. (2023). *Forecasting Time Series Data with Prophet*. Packt Publishing.
- [18] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining* (pp. 413–422). IEEE. <https://doi.org/10.1109/ICDM.2008.17>
- [19] Breunig, M. M., Kriegel, H.-P., Ng, R. T., & Sander, J. (2000). LOF: Identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data* (pp. 93–104). ACM. <https://doi.org/10.1145/342009.335388>
- [20] Martínez, F., Frías, M. P., Pérez, M. D., & Rivera, A. J. (2019). A methodology for applying k-nearest neighbour to time series forecasting. *Artificial Intelligence Review*, 52(3), 2019–2037.
- [21] Yeh, C.-C. M., Zhu, Y., Ulanova, L., Begum, N., Ding, Y., Dau, H. A., Silva, D. F., Mueen, A., & Keogh, E. (2016). Matrix profile I: All-pairs similarity joins for time series: A unifying view that includes motifs, discords and shapelets. In *2016 IEEE 16th International Conference on Data Mining* (pp. 1317–1322). IEEE. <https://doi.org/10.1109/ICDM.2016.0179>
- [22] Zhu, Y., Zimmerman, Z., Senobari, N. S., Yeh, C.-C. M., Funning, G., Mueen, A., Brisk, P., & Keogh, E. (2016). Matrix Profile II: Exploiting a novel algorithm and GPUs to break the one hundred million barrier for time series motifs and joins. In *2016 IEEE 16th International Conference on Data Mining* (pp. 739–748). IEEE. <https://doi.org/10.1109/ICDM.2016.0085>
- [23] Zhu, Y., Yeh, C.-C. M., Zimmerman, Z., Kamgar, K., & Keogh, E. (2018). Matrix Profile XI: SCRIMP++: Time series motif discovery at interactive speeds. In *2018 IEEE International Conference on Data Mining* (pp. 837–846). IEEE. <https://doi.org/10.1109/ICDM.2018.00099>
- [24] Ren, H., Xu, B., Wang, Y., Yi, C., Huang, C., Kou, X., Xing, T., Yang, M., Tong, J., & Zhang, Q. (2019). Time-series anomaly detection service at Microsoft. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 3009–3017). ACM. <https://doi.org/10.1145/3292500.3330680>
- [25] Killick, R., Fearnhead, P., & Eckley, I. A. (2012). Optimal detection of changepoints with a linear computational cost. *Journal of the American Statistical Association*, 107(500), 1590–1598. <https://doi.org/10.1080/01621459.2012.737745>
- [26] Adams, R. P., & MacKay, D. J. C. (2007). *Bayesian online changepoint detection* (arXiv:0710.3742). arXiv. <https://doi.org/10.48550/arXiv.0710.3742>
- [27] Page, E. S. (1954). Continuous inspection schemes. *Biometrika*, 41(1/2), 100–115. <https://doi.org/10.2307/2333009>

- [28] Boniol, P., Paparrizos, J., Palpanas, T., & Franklin, M. J. (2021). SAND: Streaming subsequence anomaly detection. *Proceedings of the VLDB Endowment*, 14(10), 1717–1729. <https://doi.org/10.14778/3467861.3467863>
- [29] Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). *LSTM-based encoder-decoder for multi-sensor anomaly detection* (arXiv:1607.00148). arXiv. <https://doi.org/10.48550/arXiv.1607.00148>
- [30] Audibert, J., Michiardi, P., Guyard, F., Marti, S., & Zuluaga, M. A. (2020). USAD: Unsupervised anomaly detection on multivariate time series. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (pp. 3395–3404). ACM. <https://doi.org/10.1145/3394486.3403392>
- [31] Fisher, R. A. (1925). *Statistical methods for research workers*. Oliver and Boyd
- [32] Stouffer, S. A., Suchman, E. A., DeVinney, L. C., Star, S. A., & Williams, R. M., Jr. (1949). *The American soldier: Adjustment during army life* (Vol. 1). Princeton University Press.
- [33] Hahsler, M., Piekenbrock, M., & Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software*, 91, 1–30.

7 Appendix

7.1 Descriptions of selected time series for EVTX

- Channel – Application

The **Application** channel contains events generated by user applications and selected system components via the Windows Event Log service. It records information about application execution, failures, alerts and other operational events.

- Channel - Microsoft-Windows-AppModel-Runtime/Admin

The **Microsoft-Windows-AppModel-Runtime/Admin** channel logs administrative events relating to the Windows AppModel Runtime environment, which manages the lifecycle of Universal Windows Platform (UWP) applications. It contains information about the initialisation, launch, termination and error states of these applications.

- Channel - Microsoft-Windows-AppReadiness/Admin

The **Microsoft-Windows-AppReadiness/Admin** channel contains events generated by the App Readiness service whilst preparing applications following the installation of a user profile or an operating system update. The recorded events document the progress of application initialisation prior to their first use.

- Channel - Microsoft-Windows-AppXDeployment/Operational

The **Microsoft-Windows-AppXDeployment/Operational** channel logs events related to the deployment of applications distributed in AppX or MSIX format. It contains information about the installation, updating, registration and removal of application packages.

- Channel - Microsoft-Windows-AppXDeploymentServer/Operational

The **Microsoft-Windows-AppXDeploymentServer** channel contains events generated by the service responsible for processing requests for the deployment of AppX packages. It logs the individual stages of the installation process, including the identification of any errors that occur.

- Channel - Microsoft-Windows-AppxPackaging/Operational

The **Microsoft-Windows-AppxPackaging/Operational** channel logs events related to the processing of application packages in the AppX format. It contains information on

the verification of package integrity, signatures and metadata during the installation process.

- Channel - Microsoft-Windows-Diagnosis-DPS/Operational

The **Microsoft-Windows-Diagnosis-DPS/Operational** channel contains events generated by the Diagnostic Policy Service. It logs diagnostic operations, the detection of system issues and automated remediation actions that have been carried out.

- Channel - Microsoft-Windows-GroupPolicy/Operational

The **Microsoft-Windows-GroupPolicy/Operational** channel logs the processing of Group Policy Objects (GPOs). It contains information on the loading, application and any errors that occur during the processing of security and configuration policies.

- Channel - Microsoft-Windows-PushNotification-Platform/Operational

The **Microsoft-Windows-PushNotification-Platform/Operational** channel contains events relating to the system notification infrastructure. It logs the receipt, processing and delivery of push notifications to user applications.

- Channel - Microsoft-Windows-RemoteDesktopServices-RdpCoreTS/Operational

The **Microsoft-Windows-RemoteDesktopServices-RdpCoreTS/Operational** channel logs events generated by Remote Desktop Services (RDS). It contains information about the creation, authentication, management and termination of remote user sessions.

- Channel - Microsoft-Windows-Security-LessPrivilegeAppContainer/Operational

The **Microsoft-Windows-Security-LessPrivilegeAppContainer/Operational** channel contains events relating to mechanisms for restricting user privileges when processes are running. It logs events associated with the application of the Principle of Least Privilege.

- Channel - Microsoft-Windows-Security-Mitigations/KernelMode

The **Microsoft-Windows-Security-Mitigations** channel logs events relating to security mechanisms implemented to mitigate the consequences of software vulnerabilities being exploited. It contains information on the activation of protective mechanisms such as DEP, ASLR or Control Flow Guard.

- Channel - Microsoft-Windows-SettingSync/Debug

The **Microsoft-Windows-SettingSync/Debug** channel contains diagnostic events relating to the synchronisation of user settings between devices. It logs the

synchronisation process and any errors that occur during the transfer of configuration data.

- Channel - Microsoft-Windows-Shell-Core/Operational

The **Microsoft-Windows-Shell-Core/Operational** channel contains events generated by the Windows Shell component, which provides the operating system's graphical user interface. It records events relating to the initialisation of the working environment, window management and user interaction with the system.

- Channel - Microsoft-Windows-StateRepository/Operational

The **Microsoft-Windows-StateRepository/Operational** channel contains events from the State Repository database, which stores the state of modern Windows applications. It logs operations performed on the database and changes to the state of registered applications.

- Channel - Microsoft-Windows-Storage-Storport-Operational

The **Microsoft-Windows-Storage-Storport/Operational** channel records events generated by the Storport driver, which handles communication with storage devices. It contains information about input/output operations, storage status and disk subsystem errors.

- Channel - Microsoft-Windows-Store-Operational

The **Microsoft-Windows-Store/Operational** channel contains events generated by the Microsoft Store application. It logs the downloading, installation, updating and management of applications distributed via the Microsoft Store.

- Channel - Microsoft-Windows-TaskScheduler-Maintenance

The **Microsoft-Windows-TaskScheduler/Maintenance** channel logs events generated by the Task Scheduler whilst performing system maintenance tasks. It contains information about the scheduled start, progress and outcome of individual tasks.

- Channel - Microsoft-Windows-TerminalServices-RemoteConnectionManager-Operational

The **Microsoft-Windows-TerminalServices** channel contains events relating to the management of terminal sessions. It logs the creation, management and termination of user sessions via Terminal Services.

- Channel - Microsoft-Windows-UniversalTelemetryClient-Operational

The Microsoft-Windows-UniversalTelemetryClient channel records events generated by the Windows operating system telemetry client. It contains information about the collection of diagnostic data and its processing prior to transmission.

- Channel - Microsoft-Windows-WMI-Activity-Operational

The Microsoft-Windows-WMI-Activity/Operational channel contains events generated by the Windows Management Instrumentation (WMI) service. It logs the execution of WMI queries, object management and the activity of administrative scripts using WMI.

- Channel - Microsoft-Windows-Windows Defender-Operational

The **Microsoft-Windows-Windows Defender/Operational** channel records events generated by the Microsoft Defender antivirus solution. It contains information on the detection of malicious code, scans performed and security measures applied.

- Channel - Microsoft-Windows-WindowsUpdateClient-Operational

The **Microsoft-Windows-WindowsUpdateClient** channel contains events relating to the operation of the Windows Update service. It records the search for, download, installation and outcome of operating system updates.

- Channel - Microsoft-Windows-Winlogon-Operational

The **Microsoft-Windows-Winlogon/Operational** channel logs events generated by the Winlogon process during the logon, logoff and lockout of user sessions. It contains information on user authentication and the management of logon sessions.

- Channel - Security

The **Security** channel contains security events generated by the Windows operating system's auditing mechanism. It logs user authentication, access to objects, changes to permissions and other events relevant to security.

- Channel – System

The **System** channel contains events generated by the operating system kernel, device drivers and system services. It records system start-up and shutdown, hardware status, driver loading and other system events.

- EventId_audit_account_logon – 4768, 4769, 4770

The **Audit Account Logon** category contains events relating to the verification of user account identities during the authentication process. It records both successful and unsuccessful attempts to verify local or domain accounts.

- EventId_audit_account_management - 4720, 4722

The **Audit Account Management** category logs operations relating to the management of user accounts, groups and security identifiers. It contains events relating to the creation, deletion, modification or change of permissions for user objects.

- EventId_audit_detailed_tracking - 4688, 4689

The **Audit Detailed Tracking** category contains detailed events describing the execution of processes and other system operations. It provides detailed information on the creation and termination of processes, as well as other operating system activities.

- EventId_audit_ds_access - 4661, 4662, 4928

The **'Directory Service Access' audit** category logs access to Active Directory objects. It contains events relating to the reading, modification or management of directory objects in a domain environment.

- EventId_audit_logon_logoff – 4624, 4625, 4626

The **Audit Logon/Logoff** category records both successful and unsuccessful logons, logoffs and other events related to user sessions. It provides the information needed to reconstruct user activity on the system.

- EventId_audit_non_audit - 1100, 1101, 1102

The **Audit Non-Audit** category contains events relating to the management of the operating system's audit mechanism itself. It includes, for example, the starting or stopping of the audit service and the manipulation of security logs.

- EventId_audit_object_access – 5145, 5140, 5142

The **Audit Object Access** category records access by users or processes to objects protected by the Windows security mechanism. It includes events relating to access to files, registry keys, shared resources and other system objects.

- EventId_audit_policy_change - 4715, 4719, 4817

The **Audit Policy Change** category contains events documenting changes to the operating system's security policies and audit settings. It enables the identification of tampering with the system's security configuration.

- EventId_audit_privilege_usage - 4673, 4674

The **Audit Privilege Usage** category records the use of special system privileges by processes or users. It provides information on operations requiring elevated security privileges.

- EventId_audit_system – 5024, 5025, 5027, 5028

The **Audit System** category contains security events relating to the operation of the operating system itself. It records events relating to system startup, changes to the security state and the activity of security services.

- EventId_audit_uncategorised - 4960, 4961, 4962

The **Audit Uncategorised** category contains security events that do not fall into any of the standard audit categories. It represents a supplementary group of events useful for detailed forensic analysis.

- ProcessId: 0
- ProcessId: 1016
- ProcessId: 1152
- ProcessId: 1924
- ProcessId: 2592
- Process ID: 2716
- Process ID: 2748
- Process ID: 3156
- Process ID: 316
- Process ID: 332
- Process ID: 4332
- Process ID: 4504
- Process ID: 4696
- Process Id: 4
- Process ID: 5112
- Process ID: 552
- Process ID: 5908
- Process Id: 612

7.2 Descriptions of selected time series for MFT and journal

- FileAttributes: 'Archive'

The **'Archive'** attribute indicates a file intended for backup or archiving. This flag is set by default when a file is created or modified and can be utilised by backup software.

- FileAttributes - Archive-Compressed'

The **'Archive-Compressed'** attribute indicates a file that is compressed using the NTFS mechanism and is also marked for archiving. This entry allows compressed files whose content has been created or modified to be identified.

- FileAttributes - 'Archive-NotContentIndexed'

The **'Archive-NotContentIndexed'** attribute indicates a file designated for archiving that is excluded from Windows Search indexing. It is used on files whose content is not to be included in full-text search.

- FileAttributes - Archive-Temporary'

The **'Archive-Temporary'** attribute identifies a temporary file intended for short-term data storage. Such files are typically created by applications during data processing.

- FileAttributes - Directory'

The **'Directory'** attribute identifies a file system object as a directory. Entries with this attribute represent containers intended for organising files and other directories.

- FileAttributes: 'xml-ph-0000@deepl.internal'

The **Hidden-Archive** attribute denotes a hidden file intended for archiving. Such files are not normally displayed to the user, whilst remaining part of the backup process.

- FileAttributes - Hidden-Archive-Temporary'

The **'Hidden-Archive-Temporary'** attribute identifies a hidden temporary file intended for archiving. This type of file is most commonly created by the system or by applications during temporary data processing.

- FileAttributes - Hidden-System-Archive'

The **Hidden-System-Archive** attribute identifies a system file that is hidden from the user and also marked for archiving. Such files are essential for the operating system to function correctly.

- FileAttributes - Normal'

The **'Normal'** attribute denotes a standard file with no special attributes. It represents the most common type of object stored in the NTFS file system.

- FileAttributes – 'NotContentIndexed'

The '**NotContentIndexed**' attribute indicates a file or folder whose contents are not included in Windows Search indexing. Its use reduces the load on the indexing service or protects sensitive data from being indexed.

- FileAttributes - 'System-Archive-NotContentIndexed'.csv.csv

The **System-Archive-NotContentIndexed** attribute identifies a system file intended for archiving, which is also excluded from content indexing. These are predominantly system objects used by the operating system.

- UpdateReasons –16777472'

The value 16777472 represents a numerical bit mask of the reasons for updating a record in the USN Journal or MFT. It combines several change flags, which must be interpreted according to the NTFS flag definitions.

- UpdateReasons': '2164261120'

The value 2164261120 represents a combination of several bit flags determining the reason for updating a file system record. Its meaning depends on the specific combination of set flags defined by the NTFS implementation.

- UpdateReasons': 'BasicInfoChange'

The 'BasicInfoChange' reason indicates a change to the basic metadata of a file or directory. This may include modifications to the object's timestamps, attributes or ownership.

- UpdateReasons - BasicInfoChange-Close'

This record indicates a change to the basic metadata, followed by the closure of the file. It indicates the completion of an operation that alters the basic properties of the object.

- UpdateReasons - Close'

The 'Close' reason indicates the closure of an open file descriptor by an application or a system process. It does not indicate a change to the file's contents, but rather the completion of work on the object.

- UpdateReasons - DataExtend'

The 'DataExtend' reason indicates an increase in the file size due to the writing of new data. This typically occurs when saving or adding to the file's contents.

- UpdateReasons – DataExtend-Close'

Indicates an extension of the file's contents, followed by its closure. It indicates the successful completion of the write operation.

- UpdateReasons - DataExtend-DataTruncation'

The record contains a combination of file extension and subsequent truncation during a single operation. This behaviour occurs, for example, when overwriting or optimising file content.

- UpdateReasons - DataExtend-DataTruncation-Close'

Indicates a combined file extension and truncation operation concluded by closing the object. It represents the completion of a complex operation on the file's contents.

- UpdateReasons - DataExtend-FileCreate'

This reason indicates the creation of a new file, during which data was written. This is the typical process of creating a new file with content.

- UpdateReasons - DataExtend-FileCreate-BasicInfoChange'

The record indicates the creation of a file, the writing of data and the subsequent modification of its basic metadata. It represents the standard process of creating a file by an application.

- UpdateReasons - DataExtend-FileCreate-Close'

Indicates the creation of a file, the writing of its contents, and the completion of work on the file by closing it. This represents the complete lifecycle of a file creation operation.

- UpdateReasons - DataOverwrite'

The 'DataOverwrite' reason indicates the overwriting of existing data without changing the file size. The file's contents are altered, whilst the space allocated to it remains unchanged.

- UpdateReasons - DataOverwrite-Close'

Indicates the overwriting of the file's contents, followed by the closure of the object. It signals the completion of the data modification operation.

- UpdateReasons - DataOverwrite-DataExtend-DataTruncation'

The entry represents a combination of data overwrite, file extension and file truncation during a single operation. This is a complex modification of the file's contents carried out by an application or the system.

- UpdateReasons - DataOverwrite-DataExtend-FileCreate'

Indicates the creation of a file involving the writing of both new and overwritten data. This sequence is typical when creating or copying files.

- UpdateReasons - DataTruncation'

The 'DataTruncation' reason indicates a reduction in file size through the removal of part of the file. This occurs, for example, when truncating logs or editing documents.

- UpdateReasons - FileCreate'

The 'FileCreate' reason indicates the creation of a new file entry in the NTFS file system. It marks the start of a new object's lifecycle.

- UpdateReasons - FileCreate-Close'

Indicates the creation of a new file, followed by its closure. It indicates that the operation to create the object has been successfully completed.

- UpdateReasons - FileDelete-Close'

The FileDelete-Close reason indicates the deletion of a file, confirmed by the closure of the object. The record indicates the completion of the file deletion operation.

- UpdateReasons - FileDelete-TransactedChange-Close'

The entry indicates the deletion of a file carried out within the NTFS transactional mechanism, followed by the closure of the object. Such operations are less common and utilise the file system's transactional interface.

- UpdateReasons - 'HardLinkChange-Close'

This reason indicates a change to a hard link of an existing file, resulting in the closure of the object. It indicates the creation or removal of a link to existing data.

- UpdateReasons - HardLinkChange-TransactedChange'

This entry represents a hard link change performed via an NTFS transaction. It enables the identification of transactional operations on file system links.

- UpdateReasons - IndexableChange-BasicInfoChange'

This reason indicates a change to a file's indexing settings, together with a change to its basic metadata. Such operations occur, for example, when file attributes are changed.

- UpdateReasons - RenameNewName'

The 'RenameNewName' reason indicates the creation of a new file or directory name during a renaming operation. It allows the target name of the object to be identified after the operation has been performed.

- UpdateReasons - RenameNewName-BasicInfoChange'

This record indicates the renaming of an object associated with a change to its basic metadata. It indicates a simultaneous change to both the name and the attributes of the object.

- UpdateReasons - RenameNewName-BasicInfoChange-Close'

Indicates the renaming of an object, the modification of its metadata, and the completion of the operation by closing the object. It represents the complete renaming process.

- UpdateReasons - RenameNewName-Close'

This reason indicates the successful completion of the operation to rename a file or directory. The record is created after the object is closed by the application.

- UpdateReasons - RenameOldName'

The 'RenameOldName' reason records the original name of the object prior to the renaming operation. It enables the original identity of the file or directory to be reconstructed.

- UpdateReasons - RenameOldName-BasicInfoChange'

The record contains information about the object's original name and the current change to its basic metadata. It is useful for reconstructing the course of the renaming operation.

- UpdateReasons - SecurityChange'

The 'SecurityChange' reason indicates a change to the security descriptor of a file or directory. It includes changes to the owner, access rights or audit rules.

- UpdateReasons - SecurityChange-Close'}.csv

The record indicates the completion of an operation to change an object's security attributes by closing the file. It confirms the successful completion of the security descriptor modification.

7.3 Descriptions of selected time series for MFT

The **Created0x10** time series is based on the object's creation timestamp stored in the **\$STANDARD_INFORMATION (0x10)** attribute of the MFT record. It records the creation time of a file or folder, together with the relevant object attributes. The **LastAccess0x10** time series is based on the timestamp of the last access to the object, stored in the **\$STANDARD_INFORMATION (0x10)** attribute. It provides information about the last recorded opening or use of a file or folder. The **LastModified0x10** time series is based on the timestamp of the last change to the object's content, stored in the **\$STANDARD_INFORMATION (0x10)** attribute. It allows the time of the last modification to the data stored in the file to be identified. The **LastRecordChange0x10** time series is based on the timestamp of the last change to the MFT record stored in the **\$STANDARD_INFORMATION (0x10)** attribute. It records changes to the object's metadata, such as attributes, permissions or the file name. The **Archive** flag indicates an object intended for archiving or backup. It is set when a file is created or modified and is used primarily by backup applications. The **Archive-NotContentIndexed** flag indicates an object intended for archiving which is also excluded from content indexing by the Windows Search service. It is used for files whose content is not to be included in the search index. The **Archive-RecallOnOpen** attribute identifies an object intended for archiving, the contents of which can be retrieved from remote or hierarchical storage when the object is opened. It is found primarily in Hierarchical Storage Management mechanisms or cloud solutions such as OneDrive Files On-Demand. The **Hidden-System-NotContentIndexed** flag indicates a system object that is hidden from the user and also excluded from content indexing. Such files form part of the operating system or its internal mechanisms.

All the following time series represent the filtering of the relevant timestamp according to the value of the **SiFlags** attribute:

- **Created0x10 + Archive** – contains the creation times of objects marked with the *Archive* attribute.
- **Created0x10 + Archive-NotContentIndexed** – contains the creation times of objects designated for archiving that are not indexed.
- **Created0x10 + Archive-RecallOnOpen** – contains the creation times of objects using the Recall-on-Open mechanism.
- **Created0x10 + Hidden-System-NotContentIndexed** – contains the creation times of hidden system objects without content indexing.
- **LastAccess0x10 + Archive** – contains the times of the last access to objects with the *Archive* attribute.
- **LastAccess0x10 + Archive-NotContentIndexed** – contains the times of last access to non-indexed archived objects.
- **LastAccess0x10 + Archive-RecallOnOpen** – contains the times of last access to objects using Recall-on-Open.

- **LastAccess0x10 + Hidden-System-NotContentIndexed** – contains the last access times for hidden system objects.
- **LastModified0x10 + Archive** – contains the times of the last modification of objects marked with the *Archive* attribute.
- **LastModified0x10 + Archive-NotContentIndexed** – contains the times of the last modification of archived objects that are not indexed.
- **LastModified0x10 + Archive-RecallOnOpen** – contains the times of the last modification of objects using Recall-on-Open.
- **LastModified0x10 + Hidden-System-NotContentIndexed** – contains the last modification times of hidden system objects.
- **LastRecordChange0x10 + Archive** – contains the times of the last metadata changes for objects marked with the *Archive* attribute.
- **LastRecordChange0x10 + Archive-NotContentIndexed** – contains the times of the last change to the metadata of archived, non-indexed objects.
- **LastRecordChange0x10 + Archive-RecallOnOpen** – contains the times of the last change to the metadata of objects using Recall-on-Open.
- **LastRecordChange0x10 + Hidden-System-NotContentIndexed** – contains the times of the last metadata changes for hidden system objects.

mtfeMFT :

- ts_Created0x10_{'SiFlags': 'Archive'}
- ts_Created0x10_{'SiFlags': 'Archive-NotContentIndexed'}
- ts_Created0x10_{'SiFlags': 'Archive-RecallOnOpen'}
- ts_Created0x10_{'SiFlags': 'Hidden-System-NotContentIndexed'}
- ts_LastAccess0x10_{'SiFlags': 'Archive'}
- ts_LastAccess0x10_{'SiFlags': 'Archive-NotContentIndexed'}
- ts_LastAccess0x10_{'SiFlags': 'Archive-RecallOnOpen'}
- ts_LastAccess0x10_{'SiFlags': 'Hidden-System-NotContentIndexed'}
- ts_LastModified0x10_{'SiFlags': 'Archive'}
- ts_LastModified0x10_{'SiFlags': 'Archive-NotContentIndexed'}
- ts_LastModified0x10_{'SiFlags': 'Archive-RecallOnOpen'}
- ts_LastModified0x10_{'SiFlags': 'Hidden-System-NotContentIndexed'}
- ts_LastRecordChange0x10_{'SiFlags': 'Archive'}
- ts_LastRecordChange0x10_{'SiFlags': 'Archive-NotContentIndexed'}
- ts_LastRecordChange0x10_{'SiFlags': 'Archive-RecallOnOpen'}
- ts_LastRecordChange0x10_{'SiFlags': 'Hidden-System-NotContentIndexed'}.csv

7.4 Descriptions of selected time series for the register

registry_MC_Output :

- Category': 'Services'

The 'Services' category contains registry entries relating to system services and drivers registered by the Windows operating system. These entries define how a service is started, its type, dependencies and other parameters required for its operation.

- Description': 'Services'

The 'Description: Services' value identifies artefacts whose significance relates to the configuration and management of operating system services. From a forensic perspective, '' enables the monitoring of the existence, changes and startup behaviour of both legitimate and potentially malicious services.

- HiveType': 'System'

'HiveType: System' denotes data originating from the SYSTEM registry, which contains the configuration of hardware, drivers, services and selected system parameters. This is one of the most important registry hive files for reconstructing the configuration and behaviour of the operating system.

registry_ASEPs_Output

- HiveType': 'Software'

HiveType: Software refers to entries retrieved from the **SOFTWARE** registry, which contains the configuration of the operating system, installed applications and automatic start-up mechanisms. When analysing ASEPs, it enables the identification of applications or components that start automatically upon user log-in or during system start-up.

- HiveType: 'System'.csv

HiveType: System refers to entries originating from the **SYSTEM** registry, which contains the configuration of system services, drivers and mechanisms launched at operating system start-up. When analysing ASEPs, it provides information on components that ensure automatic launch even before the user logs in and enables the identification of changes affecting system initialisation.

7.5 Descriptions of selected time series for SRUM

7.5.1 AppResourceUseInfo

SRUM (System Resource Usage Monitor) artefacts store historical information on the utilisation of system resources by individual applications and security identities. Time series generated from the AppResourceUseInfo table describe the disk and processor activity of foreground and background applications.

AppId represents the internal identifier of the application in the SRUM database. Individual values, such as 252, 255, 383, 409, 410, 411, 413, 422, 423 or 451, merely distinguish specific applications; the number itself, without further mapping, does not specify the name of the programme. The **BackgroundBytesRead** metric represents the amount of data read by the application whilst running in the background. Elevated values may indicate automatic file loading, caching, database operations () or other activities performed without direct user interaction. The **BackgroundBytesWritten** metric represents the amount of data written by the application whilst running in the background. An increase may be related to logging, caching, updates, database activity or other automatic operations. The **BackgroundContextSwitches** metric records the number of processor context switches whilst the application is running in the background. A higher number indicates more intensive scheduling of its threads by the operating system, which may not necessarily be due to user activity. The **ForegroundBytesRead** metric represents the amount of data read by the application whilst it is actively running in the foreground. Increased values may be related to the loading of documents, databases or other data whilst the user is working with the application. The **ForegroundBytesWritten** metric represents the amount of data written by the application whilst it is running in the foreground. An increase may indicate the saving or modification of data whilst the user is actively working with the application. The **ForegroundContextSwitches** metric indicates the number of processor context switches whilst the application is running in the foreground. Higher values, combined with foreground read and write operations, may indicate intensive application activity in the user context. **SidType** denotes the type of security identity under which the activity was recorded. Values such as Administrator, LocalSystem, LocalService, NetworkService, LogonId or UnknownOrUserSid allow for the distinction between system, service and user activity.

- ts_['AppId': 252], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 252], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 252], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 252], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 252], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 252], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 255], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv

- ts_['AppId': 255], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 255], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 255], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 255], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 255], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 383], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 383], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 383], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 383], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 383], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 383], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 409], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 409], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 409], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 409], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 409], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 409], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 410], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 410], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 410], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 410], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 410], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 410], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 411], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 411], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 411], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 411], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 411], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 411], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv

- ts_['AppId': 413], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 413], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 413], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 413], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 413], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 413], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 422], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 422], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 422], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 422], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 422], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 422], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 423], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 423], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 423], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 423], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 423], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 423], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['AppId': 451], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['AppId': 451], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['AppId': 451], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['AppId': 451], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['AppId': 451], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['AppId': 451], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['SidType': 'Administrator'], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['SidType': 'Administrator'], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['SidType': 'Administrator'], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv

- ts_['SidType': 'Administrator'],
'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['SidType': 'Administrator'],
'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['SidType': 'Administrator'],
'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['SidType': 'LocalService'],
'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['SidType': 'LocalService'],
'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['SidType': 'LocalService'],
'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['SidType': 'LocalService'],
'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['SidType': 'LocalService'],
'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['SidType': 'LocalService'],
'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['SidType': 'LocalSystem'],
'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['SidType': 'LocalSystem'],
'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['SidType': 'LocalSystem'],
'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['SidType': 'LocalSystem'],
'ForegroundddBytesRead']_value_ForegroundBytesRead.csv
- ts_['SidType': 'LocalSystem'],
'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['SidType': 'LocalSystem'],
'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['SidType': 'LogonId'], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['SidType': 'LogonId'],
'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['SidType': 'LogonId'],
'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['SidType': 'LogonId'], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['SidType': 'LogonId'],
'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv

- ts_['SidType': 'LogonId'], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['SidType': 'NetworkService'], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['SidType': 'NetworkService'], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['SidType': 'NetworkService'], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['SidType': 'NetworkService'], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['SidType': 'NetworkService'], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['SidType': 'NetworkService'], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv
- ts_['SidType': 'UnknownOrUserSid'], 'BackgroundBytesRead']_value_BackgroundBytesRead.csv
- ts_['SidType': 'UnknownOrUserSid'], 'BackgroundBytesWritten']_value_BackgroundBytesWritten.csv
- ts_['SidType': 'UnknownOrUserSid'], 'BackgroundContextSwitches']_value_BackgroundContextSwitches.csv
- ts_['SidType': 'UnknownOrUserSid'], 'ForegroundBytesRead']_value_ForegroundBytesRead.csv
- ts_['SidType': 'UnknownOrUserSid'], 'ForegroundBytesWritten']_value_ForegroundBytesWritten.csv
- ts_['SidType': 'UnknownOrUserSid'], 'ForegroundContextSwitches']_value_ForegroundContextSwitches.csv

7.5.2 AppTimelineProvider

AppTimelineProvider is a component of the SRUM artefact that records the temporal activity of applications in the Windows operating system. It allows you to track when a particular application was active and the user or system context associated with its activity.

The **AppId** value represents the application's internal identifier in the SRUM database, with individual numeric values distinguishing between different applications or system components. **SidType** specifies the type of security identity, such as a user, LocalSystem, LocalService or NetworkService, whilst **UserName** allows the activity to be assigned to a specific user or system account.

- Appld': 111
- Appld': 17
- Appld': 2243
- Appld': 2249
- Appld': 2276
- Appld': 2286
- Appld': 2288
- Appld': 3472
- Appld': 3
- Appld': 99
- SidType': 'Administrator'
- SidType': 'LocalService'
- SidType': 'LocalSystem'
- SidType': 'NetworkService'
- SidType': 'UnknownOrUserSid'
- UserName': 'Administrator'
- UserName': 'LocalService'
- UserName': 'NetworkService'
- UserName': 'SelmaBouvier'
- UserName': 'systemprofile'