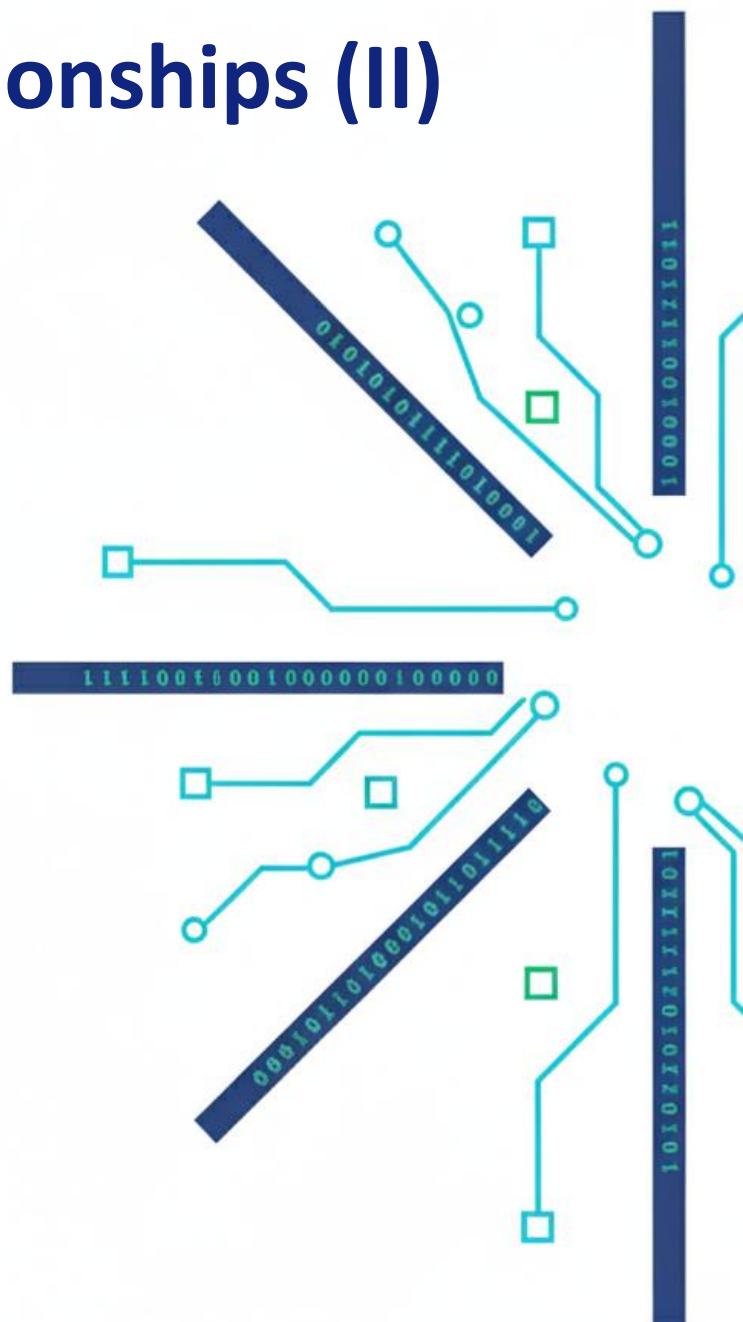


D20 – A method for identifying relationships (II)



The Automation of Digital Forensic Analysis and Incident Response (ADFIR) project 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.

Contents

Project description	2
Introduction.....	3
1 Formal Concept Analysis	4
1.1 Basic concepts	4
1.2 Attribute implications	5
2 A method for identifying significant time windows.....	8
2.1 Input data and their pre-processing	8
2.2 Procedure.....	9
3 Application of the method for identifying significant time windows.....	13
3.1 Data pre-processing	13
3.2 Evaluation of results	14
Conclusion	18
Bibliography	19
Appendices	20

Project description

The project **Automation of Digital Forensic Analysis and Incident Response** (hereinafter referred to as '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 amount 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 evidence 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 for investigations and court proceedings.

Introduction

A security analyst often works with large volumes of data in which security-relevant activity is scattered amongst normal activity. An important task in cyber security is therefore to identify those parts of the data which, by their characteristics, differ from normal activity and may indicate the occurrence of a security incident.

One approach suitable for analysing relationships within data is formal conceptual analysis. It enables the identification of groups of objects with shared properties and, using attribute implications, the expression of dependencies between these properties. Attribute implications thus capture combinations of attributes whose occurrence conditions the presence of other attributes.

The aim is to propose a method based on formal conceptual analysis that will enable the identification of time windows characterised by recurring combinations of attributes specific to a given time period. The output of the method consists of rules linking the attribute characteristics of objects to specific time windows. The segments identified in this way may indicate the occurrence of a security incident or other non-standard activity and alert the analyst to parts of the data that are suitable for further analysis.

1 Formal Concept Analysis

Formal Concept Analysis (FCA)[1] is a mathematical framework based on set theory, which is used to discover concepts in binary data and organise them hierarchically. The input data can be represented as a binary table, in which rows correspond to objects, columns to attributes, and a value of 1 indicates that the object in question possesses the relevant attribute.

1.1 Basic concepts

Formally, let B and A be non-empty sets of objects and attributes, and let $R \subseteq B \times A$ be a binary relation. Then the triple (B, A, R) is called a **formal context** – an example is given at **Figure1**.

	i	ii	iii
a	×	×	×
b		×	
c			×

Figure1 – Formal context

To find the common attributes of a subset of objects and the common objects of a subset of attributes, the operators $\uparrow: \mathcal{P}(B) \rightarrow \mathcal{P}(A)$ and $\downarrow: \mathcal{P}(A) \rightarrow \mathcal{P}(B)$, are used, where $\mathcal{P}(B)$ and $\mathcal{P}(A)$ denote the power sets of the sets B and A respectively.

For the subset of objects $X \subseteq B$, we define

$$X^\uparrow = \{a \in A \mid \forall b \in X: (b, a) \in R\}.$$

Similarly, for the subset of attributes $Y \subseteq A$, the following holds

$$Y^\downarrow = \{b \in B \mid \forall a \in Y: (b, a) \in R\}.$$

The operators $\uparrow$ and $\downarrow$ form a **Galois connection**. By composing them, $X^{\uparrow\downarrow}$ and $Y^{\downarrow\uparrow}$, we obtain closure operators.

A pair (X, Y) , for which $X^\uparrow = Y$ and $Y^\downarrow = X$ hold simultaneously is called a **formal concept**. The object part of the concept X is called the extent, and the attribute part of the concept Y is called the intent.

A formal concept therefore expresses a complete pair – a closed subset of objects together with all the attributes that this subset has in common. In a tabular representation, we can think of a formal concept as the largest rectangle filled with units. This means that we cannot extend it by adding another row or column without it containing a zero.

We denote the set of all formal concepts of the formal context (B, A, R) as $\mathcal{B}(B, A, R)$.

Formal concepts can be ordered using the relation $(X_1, Y_1) \leq (X_2, Y_2)$ precisely when $X_1 \subseteq X_2$, or $Y_2 \subseteq Y_1$.

We call a partially ordered set of formal concepts $(\mathcal{B}(B, A, R), \leq)$ a **concept bundle**. A concept bundle can be visualised using a line diagram (**Figure2**).

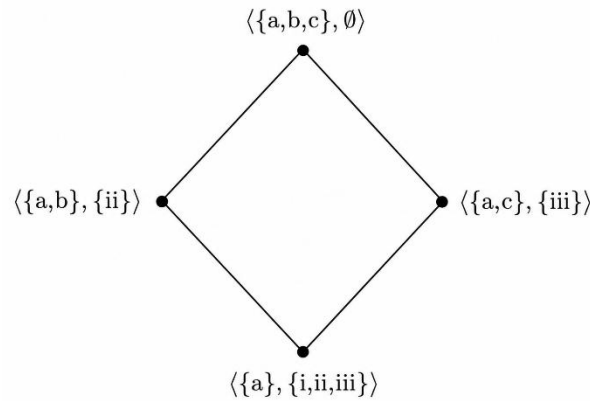


Figure2 – Concept bundle

According to the fundamental theorem on concept sets [1], this set is complete; that is, for every subset of formal concepts within it, there exists both an infimum and a supremum.

However, binary attribute values are not always sufficient to capture an object's property, as they express only its presence or absence. The fuzzy extension of formal concept analysis[3] therefore allows one to work with values from the interval $[0,1]$, whereby this value is usually interpreted as the degree to which an object belongs to the attribute. The proposed method does not utilise fuzzy formal conceptual analysis directly; operations on the formal context are applied exclusively to binary attributes. However, when pre-processing the timestamp, we drew inspiration from the representation of values in the unit interval. The timestamp is first converted to a numerical epoch time value, then normalised to the interval $[0,1]$, and finally discretised into a finite set of values representing individual time windows. These values do not represent degrees of membership in the sense of fuzzy formal conceptual analysis, but rather the relative temporal position of objects.

1.2 Attribute implications

Whilst a concept bundle represents a data structure through hierarchically organised formal concepts, attribute implications[1][2] allow dependencies between attributes contained within this structure to be expressed in the form of logical rules.

Formally, let A be a non-empty set of attributes and let $Z, W \subseteq A$. An **attribute implication** over the set A is a formal expression

$$Z \Rightarrow W,$$

which states that every object that has all the attributes from the set Z also has all the attributes from the set W . The left-hand side of the rule is called **the antecedent** and the right-hand side **the consequent**.

The implication $Z \Rightarrow W$ holds on the set of attributes $Y \subseteq A$ if

$$Z \subseteq Y \Rightarrow W \subseteq Y.$$

Equivalently, the implication $Z \Rightarrow W$ is valid on Y if $Z \not\subseteq Y$ or $W \subseteq Y$.

The above definition describes the validity of an implication with respect to a single set of attributes. In a formal context, however, we work with multiple objects, each of which has its own set of attributes. We shall therefore generalise the validity of an implication to a system of sets.

Let $\mathcal{Y} \subseteq \mathcal{P}(A)$ be a system of subsets of a set of attributes. An attribute implication $Z \Rightarrow W$ is valid on $\mathcal{Y}$ if it is valid on every set $Y \in \mathcal{Y}$, that is, if for every $Y \in \mathcal{Y}$ the following holds

$$Z \subseteq Y \Rightarrow W \subseteq Y.$$

Let (B, A, R) be a formal context. Each object $b \in B$ is associated with a set of attributes

$$\{b\}^\uparrow = \{a \in A \mid (b, a) \in R\}.$$

From the perspective of attribute implications, the formal context (B, A, R) can be represented by a system of sets

$$\mathcal{Y} = \{\{b\}^\uparrow \mid b \in B\},$$

which contains the sets of attributes associated with individual objects.

An attribute implication $Z \Rightarrow W$ is valid in the formal context (B, A, R) if it is valid on every set $\{b\}^\uparrow$. Thus, for every object $b \in B$, the following holds

$$Z \subseteq \{b\}^\uparrow \Rightarrow W \subseteq \{b\}^\uparrow.$$

The support of **the antecedent** Z is the number of objects that contain all the attributes from Z :

$$\text{supp}(Z) = |Z^\downarrow|.$$

The support of **the attribute implication** $Z \Rightarrow W$ is the number of objects that contain all attributes from $Z \cup W$:

$$\text{supp}(Z \Rightarrow W) = |(Z \cup W)^\downarrow|.$$

A rule $Z_1 \Rightarrow W$ is called **an a-maximal**[4] if there is no rule $Z_2 \Rightarrow W$ for which the following holds

$$Z_1 \subset Z_2$$

and, at the same time,

$$\text{supp}(Z_1 \Rightarrow W) = \text{supp}(Z_2 \Rightarrow W).$$

The antecedent Z_1 cannot therefore be extended by a further attribute without reducing the set of objects that support the rule.

An attribute implication $Z \Rightarrow W$ is called trivial if $W \subseteq Z$.

The formal context from **Figure 1** contains 6 non-trivial attribute implications:

$$\begin{aligned} \{i\} &\Rightarrow \{ii\}, \\ \{i\} &\Rightarrow \{iii\}, \\ \{i\} &\Rightarrow \{ii, iii\}, \\ \{i, ii\} &\Rightarrow \{iii\}, \\ \{i, iii\} &\Rightarrow \{ii\}, \\ \{ii, iii\} &\Rightarrow \{i\}. \end{aligned}$$

The last four attribute implications are A-maximal.

2 A method for identifying significant time windows

The aim of the method is to identify time windows characterised by recurring combinations of attributes that do not occur in other parts of the dataset. Windows defined in this way may indicate the occurrence of a security incident or other non-standard activity and alert security analysts to sections of data suitable for further analysis.

The output of the method is a set of attribute implications, in which the left-hand side of the rule consists of a combination of binary attributes and the right-hand side consists of a time window. For a set of objects $E \subseteq B$ that belong to the same time window τ , the resulting rule takes the form

$$E^\uparrow \Rightarrow \text{time_window}(\tau),$$

where

$$E^\uparrow = \{a \in A \mid \forall b \in E: (b, a) \in R\}$$

represents the set of all binary attributes common to the objects from E and $\text{time_window}(\tau)$ is a predicate stating that an object belongs to the time window τ . Verbally, we can interpret the rule as follows: if an object contains all attributes from $E^\uparrow$, then it belongs to the time window τ .

The rule is a-maximal with respect to the given set of objects E , because its antecedent contains all the attributes that are common to every object from E . No further attribute can be added to it without changing the set of objects satisfying the rule.

The parameter *minsupp* specifies the minimum number of objects on which a given attribute pattern must occur within a single time window:

$$|E| \geq \text{minsupp}.$$

The '*minsupp* = 1' value is generally not suitable, as it allows rules to be created based on a single object. Such rules can only describe individual or random characteristics of a specific record, not a recurring pattern characteristic of a given time period. A higher minimum support ensures that the identified characteristics are confirmed by repeated occurrence across multiple objects.

The method thus assigns to each identified time window closed combinations of attributes that occur repeatedly within it and do not occur in the same combination outside it. The result is time windows together with attribute characteristics specific to them.

2.1 Input data and their pre-processing

The proposed method assumes an input dataset containing binary attributes and a single attribute representing the timestamp of each object. The binary attributes take the values 1

and 0, , representing the presence or absence of the relevant property, respectively. The timestamp is determined by the date and time at which the event was recorded.

Timestamps must be converted to values from the unit interval [0,1], which represents the standard range of values used in fuzzy formal concept analysis.

The timestamp is first converted to a numerical representation of the epoch time, which expresses the number of seconds elapsed since a fixed reference point in time. For the object b_i , let us denote this value as e_i . High epoch time values are subsequently normalised to a unit interval. The normalised time value of the object b_i is determined by the relationship

$$t_i = \frac{e_i - e_{\min}}{e_{\max} - e_{\min}},$$

where $e_{\min}$ and $e_{\max}$ represent the minimum and maximum epoch time values in the dataset. The value $t_i = 0$ corresponds to the start of the observed period and the value $t_i = 1$ to its end.

However, precise timestamps are generally unique to individual objects, whilst the aim of the method is to analyse activity within broader time intervals. For this reason, the normalised time interval is divided into q equally wide time windows. The width of a single window is

$$\Delta = \frac{1}{q}.$$

Each object is assigned to the relevant window according to its normalised time value, with the boundary value of 1 belonging to the last window. The object is then assigned a discrete time value corresponding to the start of that time window.

Objects with similar timestamps are thus grouped into the same time window. Discretisation allows for the analysis of recurring combinations of attributes within common time intervals, rather than assessing each timestamp individually.

2.2 Procedure

Step 1 – Generating candidate attribute sets

From the set of binary attributes A , non-empty candidate sets of attributes are generated

$$C \subseteq A,$$

, the size of which is limited by the parameter $k_{\max}$, i.e.

$$1 \leq |C| \leq k_{\max}.$$

The set of all candidates under consideration is

$$C_{k_{\max}} = \{C \subseteq A \mid 1 \leq |C| \leq k_{\max}\}.$$

Each candidate C represents a potential combination of attributes common to a certain group of objects.

Step 2 – Calculating the extent of a candidate

For each candidate $C \in \mathcal{C}_{k_{max}}$, the set of objects that have all the attributes from C is determined using the operator $\downarrow$:

$$C^\downarrow = \{b \in B \mid \forall a \in C: (b, a) \in R\}.$$

The set $C^\downarrow$ is the extent of the candidate C .

A candidate's support is defined as the number of objects in its extent:

$$supp(C) = |C^\downarrow|.$$

Only candidates meeting the minimum support specified by the value $minsupp$ are retained for further processing:

$$supp(C) \geq minsupp.$$

The set of retained candidates is

$$\mathcal{C}^* = \{C \in \mathcal{C}_{k_{max}} \mid |C^\downarrow| \geq minsupp\}.$$

Step 3 – Grouping candidates by the same extent

Different candidate sets may identify the same group of objects. An equivalence is therefore defined for candidates $C_1, C_2 \in \mathcal{C}^*$

$$C_1 \sim C_2 \iff C_1^\downarrow = C_2^\downarrow.$$

A class of candidates is created for each unique extent $E \subseteq B$

$$\mathcal{E}(E) = \{C \in \mathcal{C}^* \mid C^\downarrow = E\}.$$

Each unique extent E is processed only once in the subsequent steps. Candidates belonging to the same class $\mathcal{E}(E)$ describe the same group of objects, even though they contain different combinations of attributes.

Step 4 – Calculating the intent

For each unique extent $E \subseteq B$, its intent – the set of attributes common to all objects from E – is calculated using the operator $\uparrow$:

$$E^\uparrow = \{a \in A \mid \forall b \in E: (b, a) \in R\}.$$

The set $E^\uparrow$ represents the complete common attribute description of the objects belonging to the extent E . Since E was formed as an extent of a candidate set of attributes, the following holds

$$E^{\uparrow\downarrow} = E.$$

The set $E^\uparrow$ is therefore a closed set of attributes. It contains all attributes that can be included in the description without altering the relevant set of objects. In this sense, it represents the maximal attribute description with respect to the extent E .

Step 5 – Verification of the temporal homogeneity of the extent

For each extent E , the set of time windows for its objects is determined:

$$T(E) = \{\tau(b) \mid b \in E\}.$$

An extent E is temporally homogeneous if all its objects belong to the same time window:

$$|T(E)| = 1.$$

If an extent contains objects from multiple time windows, no time rule is created for it.

Step 6 – Creating a closed time rule

For each temporally homogeneous extent E whose objects belong to the time window τ , a rule is created

$$E^\uparrow \Rightarrow \text{time_window}(\tau).$$

The antecedent $E^\uparrow$ contains all attributes common to the objects in the extent E and represents the maximum attribute description with respect to this group of objects.

The rule's support is

$$\text{supp}(E^\uparrow \Rightarrow \text{time_window}(\tau)) = |E|.$$

The resulting rule describes a complete subset of attributes common to at least minsupp objects belonging to the same time window. Furthermore, no object outside this time window satisfies the entire left-hand side of the rule.

Step 7 – Converting the time window back to the original time interval

For each identified time window τ , the corresponding interval from the dataset's original time axis is determined. If a time window in the normalised interval $[0,1]$ has the form $[\tau, \tau + \Delta)$, its start and end are calculated as

$$e_{\text{start}} = e_{\text{min}} + \tau(e_{\text{max}} - e_{\text{min}})$$

and

$$e_{\text{end}} = e_{\text{min}} + (\tau + \Delta)(e_{\text{max}} - e_{\text{min}}),$$

where e_{min} and e_{max} represent the smallest and largest epoch time values in the dataset, and Δ is the width of the time window. For the last time window, the upper bound of the interval is set to e_{max} . The obtained epoch time values are converted back to the original date and time representation.

This reverse conversion allows identified rules to be assigned to a specific period on the original timeline. It is then possible to search for objects that match individual rules and analyse them alongside the original forensic data. A security analyst can thus verify what events or activities occurred during a given time period and whether they are related to a security incident.

The identified time windows can be further analysed by repeatedly applying the method. Objects belonging to the selected window are isolated, their timestamps are re-normalised with respect to the time range of that window, and they are subsequently divided into smaller windows. Repeated application of the method allows for more precise localisation of attribute patterns and the identification of shorter time periods in which a specific activity occurs.

3 Application of the method for identifying significant time windows

To apply the method, we used a dataset from the ‘Magnet CTF 2019’ dataset, which was aggregated using a 900-second time window. Detailed pre-processing of the dataset is described in output KPB2-02. Individual rows of the dataset represent clusters of events occurring within this interval. The complete output from applying the method is provided in the appendix Magnet_CTF_2019_rules_900s.xlsx, where each row contains the rule ID, rule support, its length and the rule itself.

3.1 Data pre-processing

The dataset contains 5,830 objects and 223 attributes. In the first step, non-binary attributes were removed, with the exception of the `time_cluster` attribute, which contains the date and time of the relevant object. Of the original attributes, `version`, `container_id`, `id` and `cluster_length` were identified as non-binary and removed. Following this adjustment, 218 binary attributes and one temporal attribute remained.

The timestamp was subsequently converted to a numerical representation of epoch time and normalised to the interval [0,1]. The normalised interval was divided into 20 equally wide time windows, each with a width of

$$\Delta = \frac{1}{20} = 0,05.$$

Each object was assigned a discrete value representing the start of the relevant time window, based on its normalised time value. The numbers of objects in each window are listed in **Table1**.

Start of time window	Number of objects
0	35
0.05	24
0.1	29
0.15	27
0.2	28
0.25	11
0.3	23
0.35	31
0.4	54
0.45	23
0.5	10
0.55	36
0.6	16
0.65	583

0.7	602
0.75	615
0.8	856
0.85	901
0.9	993
0.95	933

Table1 – Number of objects in time windows

In the next step, constant attributes were identified amongst the binary attributes, which took the value 0 for all objects. Such attributes do not allow for differentiation between objects and have no impact on the resulting rules. Their removal therefore preserves the information content relevant to the proposed method whilst reducing the dimension of the input data. A total of 56 constant zero attributes were removed. Constant attributes with a value of 1 did not occur in the dataset.

The resulting dataset used in applying the method contains 5,830 objects, 162 binary attributes and 1 discretised time attribute. The time attribute does not serve as part of the candidate sets on the left-hand side of the rules, but determines the time window specified on their right-hand side.

3.2 Evaluation of results

The method was applied to the dataset using the following parameters: $k_{\max} = 3$ and $\text{minsupp} = 3$.

The output of the method application consists of rules for **time windows of 0.9 and 0.95**. The identified time windows correspond approximately to the following intervals from the original dataset: from 25 February 2019 at 08:00 to 9 March 2019 at 02:30, and from 9 March 2019 at 02:30 to 20 March 2019 at 21:15.

For the 0.9 time window, 44 temporal rules were identified, whilst for the 0.95 window, as many as 710 rules were identified. The results therefore suggest that, particularly in the latter time window, there is a large number of recurring attribute patterns which do not occur in the same form in any other time segment of the dataset. This window is therefore the most distinctive from the perspective of the proposed method and may indicate specific or non-standard time-concentrated activity.

Table2 presents a comparison of selected time windows and their rules.

Start of time window	Number of objects	Number of rules	Maximum rule support	Maximum number of attributes in a rule
0.9	993	44	9	47

0.95	933	710	379	53
------	-----	-----	-----	----

Table2 – Selected time windows and their rules

The absence of rules in the other time windows does not mean that no significant activity took place during them. However, given the selected parameters, no sufficiently supported combination of attributes was found that occurred exclusively in one of these windows.

Support determines how many objects satisfy a given rule. Rules with higher support describe recurring activity among a larger group of objects. Conversely, rules with lower support capture narrower, but potentially more security-relevant, combinations of attributes.

The left-hand side of a rule contains the complete subset of attributes common to the objects that satisfy the rule. A large number of attributes does not automatically indicate a more significant rule. A rule often contains general technical attributes that occur together in a given type of forensic record. These attributes are found in a large number of rules and may represent a common technical basis for the objects rather than a characteristic of a specific incident. When interpreting the rules, therefore, the attributes that supplement this general basis are of greater interest; for example, attributes relating to scripts, suspicious domains, executable files, logins or changes to permissions.

When the rules are ranked by support, the most widely supported rules are concentrated within the 0.95 confidence interval. A closer examination reveals that these are not entirely distinct rules, but predominantly a single common underlying pattern supplemented by additional attributes. This common basis consists of file and NTFS artefacts from the user's home directory and the AppData folder, which are linked to the Chrome cache, Windows events and a suspicious domain. In the other rules, this basic pattern has been supplemented primarily with attributes relating to HTTPS communication and browser history, restored records, permission changes, logon events, Prefetch records and repeated programme launches.

The top 10 rules with the highest support contain 18 common attributes:

- has_macb_accessed,
- has_macb_modified,
- has_modified_flag,
- has_sha256,
- is_accessed,
- is_file,
- is_file_entry,
- is_filename_attr,
- is_from_users_dir,
- is_in_appdata,
- is_ntfs,
- is_ntfs_event,
- is_standard_info,
- is_suspicious_domain,

- mask_Chrome_Cache,
- mask_File_stat,
- mask_NTFS_file_stat,
- mask_WinEVTX.

In

the

Table '3', only the attributes by which the left-hand side of the rule differs from the common base are listed for these rules.

Rule number	Support	Additional attributes compared to the common base
45	379	-
46	173	has_transition_type, is_https, mask_Chrome_History
47	144	is_log_recovered
48	113	is_privilege_change
49	99	has_transition_type, is_https, is_log_recovered, mask_Chrome_History
50	65	is_login_event
51	61	mask_WinPrefetch
52	59	is_run_often, mask_WinPrefetch
53	56	mask_Registry_Key
54	55	has_transition_type, is_https, is_privilege_change, mask_Chrome_History

Table '3' – Selected rules and their additional attributes

Three potentially interesting rules were selected from those with the lowest support, which contain attributes related to security-relevant activity. An overview of these is provided in **Table4**.

Rule number	Start of time window	Number of objects	Number of attributes	Attributes
22	0.9	3	45	IsDynamicLibrary, IsInAppData, IsUserHive, MissingMInMACB, has_macb_accessed, has_macb_modified, has_modified_flag, has_sha256, is_accessed, is_executable, is_file, is_file_entry, is_filename_attr, is_from_system32, is_from_users_dir, is_in_appdata, is_in_system32, is_in_temp_or_spool, is_in_winsxs, is_link_file, is_log_recovered, is_login_event, is_ntfs, is_policy_change, is_privilege_change, is_rare_app, is_script_file, is_standard_info, is_suspicious_domain, is_system_path, is_unallocated, is_user_profile_shortcut, is_xml_file, key_new_registry_branch, key_suspicious_value, macb_is_only_access_birth, mask_Chrome_Cache, mask_File_entry_shell_item, mask_File_stat, mask_MSIE_WebCache_container_record, mask_NTFS_file_stat, mask_Registry_Key, mask_WinEVTX, mask_WinPrefetch, mask_Windows_Shortcut
726	0.95	3	28	accessed_local_file, accessed_sensitive_file, contains_secret_path, has_macb_accessed, has_macb_modified, has_modified_flag, has_sha256, has_transition_type, is_accessed, is_file, is_file_entry, is_filename_attr, is_from_users_dir, is_https,

				is_in_appdata, is_log_recovered, is_modified_after_access, is_ntfs, is_ntfs_event, is_standard_info, is_suspicious_domain, is_suspicious_keyword, mask_Chrome_Cache, mask_Chrome_History, mask_File_stat, mask_MSIE_WebCache_container_record, mask_NTFS_file_stat, mask_WinEVTX
735	0.95	3	27	has_macb_accessed, has_macb_modified, has_modified_flag, has_sha256, is_accessed, is_admin_activity, is_error_event, is_failed_login, is_file, is_file_entry, is_filename_attr, is_frequent_execution, is_in_appdata, is_log_recovered, is_login_event, is_ntfs, is_policy_change, is_privilege_change, is_run Often, is_standard_info, is_system_process, is_xml_file, mask_File_stat, mask_NTFS_file_stat, mask_Registry_Key, mask_WinEVTX, mask_WinPrefetch

Table4 – Selected potentially security-relevant rules

Rule 22 contains the attributes *key_suspicious_value*, *is_script_file*, *is_suspicious_domain*, *is_rare_app*, *is_system_path*, *is_in_temp_or_spool*, *is_unallocated*, *is_privilege_change* and registry entries. The *key_suspicious_value* attribute directly identifies a potentially suspicious registry value. When combined with a script, a suspicious domain, and system or temporary paths, this may indicate a configuration related to the execution of a non-standard application.

Rule 726 contains the attributes *accessed_local_file*, *accessed_sensitive_file*, *contains_secret_path*, *is_https*, *is_suspicious_domain*, *is_suspicious_keyword*, *mask_Chrome_History*, *mask_Chrome_Cache* and *is_modified_after_access*. Such a combination may indicate the handling of sensitive data in connection with web activity.

Rule 735 contains the attributes *is_failed_login*, *is_login_event*, *is_admin_activity*, *is_privilege_change*, *is_frequent_execution*, *is_run Often*, *is_system_process* and *is_error_event*. This combination may indicate failed login attempts in connection with administrative activity. This could be a common user error, but it could also indicate attempts to gain unauthorised access.

Conclusion

The proposed method enables the identification of significant time windows in forensic data that may indicate the occurrence of non-standard or security-relevant activity. The proposed procedure, based on formal conceptual analysis methods, utilises recurring combinations of attributes that are specific to selected parts of the dataset. The output consists of temporal rules linking a closed attribute description of a group of objects to a specific time window. Thus, in addition to localising the time interval, the method also provides its characteristics, which can be further utilised in forensic analysis.

The proposed method does not generate all valid attribute implications with a temporal attribute in the consequent. For each identified temporally homogeneous extent, it creates only one rule with a maximal antecedent, thereby reducing redundancy in the resulting rules.

The output also depends on the constraint ' $k_{\max}$ '. The value of the parameter ' $k_{\max}$ ' represents a trade-off between the scope of the search and computational complexity. Higher values may lead to the identification of additional time-specific extents; however, the number of candidate sets examined increases sharply as the number of attributes rises. In the dataset in question, even a value of $k_{\max} = 3$ was sufficient to identify time windows containing specific attribute patterns. The results thus suggest that even a low value of $k_{\max}$ may be sufficient to locate relevant time intervals.

Bibliography

- [1] Ganter, B., Wille, R. (2024). *Formal Concept Analysis: Mathematical Foundations*. Springer.
<https://doi.org/10.1007/978-3-031-63422-2>
- [2] Carpineto, C., Romano, G. (2004). *Concept Data Analysis: Theory and Applications*. John Wiley & Sons.
<https://doi.org/10.1002/0470011297>
- [3] Bělohlávek, R. (2002). *Fuzzy Relational Systems: Foundations and Principles*. Kluwer Academic/Plenum Publishers.
<https://doi.org/10.1007/978-1-4615-0633-1>
- [4] Bayardo, R. J., Agrawal, R. (1999). *Mining the Most Interesting Rules*. ACM.
<https://doi.org/10.1145/312129.312219>

Appendices

ADFIR-D20-KPB4-02-Magnet_CFT_2019_rules_900s.xlsx