Forensic Science International: Digital Investigation 50 (2024) 301797

Contents lists available at ScienceDirect

Forensic Science International: Digital Investigation

journal homepage: www.elsevier.com/locate/fsidi

A Formal Concept Analysis approach to hierarchical description of malware threats

Manuel Ojeda-Hernández, Domingo López-Rodríguez, Ángel Mora

Depto. de Matemática Aplicada, Universidad de Málaga, Andalucía Tech, Málaga, 29071, Spain

ARTICLE INFO

MSC:
03G10
03B70
06A15
68T27
68T30
68U35

Keywords:
Formal Concept Analysis
Hierarchy
Malware classification

ABSTRACT

The problem of intelligent malware detection has become increasingly relevant in the industry, as there has been an explosion in the diversity of threats and attacks that affect not only small users, but also large organisations and governments. One of the problems in this field is the lack of homogenisation or standardisation in the nomenclature used by different antivirus programs for different malware threats. The lack of a clear definition of what a category is and how it relates to individual threats makes it difficult to share data and extract common information from multiple antivirus programs. Therefore, efforts to create a common naming convention and hierarchy for malware are important to improve collaboration and information sharing in this field.

Our approach uses as a tool the methods of Formal Concept Analysis (FCA) to model and attempt to solve this problem. FCA is an algebraic framework able to discover useful knowledge in the form of a concept lattice and implications relating to the detection and diagnosis of suspicious files and threats. The knowledge extracted using this mathematical tool illustrates how formal methods can help prevent new threats and attacks. We will show the results of applying the proposed methodology to the identification of hierarchical relationships between malware.

1. Introduction

The problem of intelligent detection of malware (Kouliaridis and Kambourakis, 2021) has become increasingly relevant in the industry, as there has been a surge in the diversity of threats and attacks that affect not only small users, but also large organisations and governments. The consequences of these attacks can range from the compromise of sensitive information to significant economic losses. *WannaCrypt0r Ransomware* managed to infect 300,000 computers around the globe by requesting and obtaining ransom in Bitcoin (Robb, 2024). In Namanya et al. (2018), the authors claim that malware analysis has evolved due to open source, online and readily available automated malware analysis tools such as VirusTotal's (VirusTotal, 2014), which uses over 70 antivirus engines to analyse uploaded malware samples and provides a free report after analysis. These online tools generate large datasets to be analysed, posing new open problems to be solved, such as the categorisation or clustering of malware by extracting knowledge capable of classifying new attacks.

Therefore, the malware analysis issue has garnered significant attention from researchers and practitioners alike. In short, it is a matter of determining which characteristics present in the malware allow us to detect it at an early stage (Firdaus et al., 2018; Escudero Garcia and DeCastro-Garcia, 2021). In Kellogg et al. (2014) the authors use a hierarchical organization of malware based on big data to manage malware to support effective and efficient malware analysis on large and rapidly evolving sets of malware.

Moreover, intelligent methods are required (Ling et al., 2022) because "malware hides itself through obfuscation techniques and is able to evade existing detection methods". These authors use a hierarchical obfuscation malware detection method based on deep learning to cluster the specific categories of obfuscated malware. Robust and complex methods are claimed to capture semantic and structural knowledge in malware. They propose the MalGraph tool, which represents executables with hierarchical graphs and uses a comprehensive learning framework based on graph neural networks for malware detection.

In fact, most of the methods use neural networks, deep networks, which causes many of them to have limited generalization capacity due to lack of good semantic information (Wang et al., 2021). Explainability and intelligent and formal methods are required.

The problem approached in this work is the lack of homogenisation or standardisation in the nomenclature used by various antivirus software for different malware threats (Maggi et al., 2011; Walker et al.,

E-mail address: manuojeda@uma.es (M. Ojeda-Hernández).

https://doi.org/10.1016/j.fsidi.2024.301797
Received 31 March 2023; Received in revised form 6 June 2024; Accepted 23 June 2024
Available online 4 July 2024

2022). A threat can be given different names depending on which antivirus engine detected it, making it difficult to share data and extract common information from multiple antivirus software.

This absence of standardisation in the naming of malware by different antivirus engines can lead to confusion and makes it difficult to compare and combine data from multiple sources. It can also lead to duplication of efforts, as different engines may be detecting the same threat but giving it different names, making it harder to identify and track. This is why efforts to create a common naming convention and malware hierarchy are important to improve collaboration and information sharing in the field.

The lack of a clear definition of what a *category* is and how it is related to individual threats hinders the process of characterising and understanding *malware threats*. Without a clear definition and common understanding, it is difficult to compare and aggregate data from different sources, making it harder to identify common characteristics and patterns across different threats. Therefore, it is important to establish a common framework and terminology for characterising *malware* categories and their individual members.

Various approaches have been proposed to address this problem, although two alternatives have predominated in recent literature: those based on textual and semantic analysis of the labels or identifiers provided by various antivirus software, and those based on machine learning methods such as clustering or random forests.

In one of the earliest studies on this topic, Maggi et al. (2011) proposed a method for detecting inconsistencies between various antivirus software, instead of determining a standard nomenclature or a mapping between the identifiers used by each program. To achieve this, they modelled the identifier graph of each antivirus software and determined a measure of inconsistency or lack of agreement among the graphs.

The use of natural language processing, regular expressions, and certain heuristic methods is evident in the work of Sebastián and Caballero (2020), which presents a semiautomatic mechanism for creating a threat hierarchy. It should be noted that in these works, the term hierarchy refers to the unification of the nomenclature used by all antivirus systems. It is worth mentioning that, in our proposal, as we will make clear later, we include additional components such as hierarchical relationships and characterisation of malware categories.

Among the methods that use machine learning techniques, we can highlight the approach of Martín et al. (2019), where the authors use clustering on graphs to group malware and antivirus categories, and then employ logistic regression and random forests to predict unknown threats. Salem et al. (2021) propose the use of machine learning techniques to identify threat categories based on VirusTotal reports, using individual identifiers. In Basole and Stamp (2021), the authors use the k-means clustering method to analyse the relationships between various samples of malicious software, allowing the exploration of existing categories (among a predetermined set).

Sitting between the two aforementioned strategies, we find (Liu et al., 2020), where the authors propose hierarchical clustering on the identifiers provided by antivirus software, to which they apply word embedding techniques, which transform the text into a vector in a Euclidean space $\mathbb{R}^n$, so that similar texts, in terms of content or semantics, are mapped to nearby vectors.

In fact, we have not delved into other methods, such as those proposed by Martín et al. (2018) and Nadeem et al. (2021), where the application for malware categorisation is for real-time threat detection through device activity monitoring. Our objective, as we will present shortly, is different and more focused on characterising malware categories based on their activity.

Our approach involves applying formal conceptual analysis (FCA) methods to model and attempt to solve the problem we have just mentioned. In general terms, our goal is to create a standard map or hierarchy of malware threats that allows us to understand their characteristics and properties using formal methods.

The techniques proposed in FCA can also contribute to the development of intelligent systems, particularly in the areas of decision support and expert systems. By applying FCA techniques to large and complex data sets, these systems can provide more accurate and reliable recommendations (Cordero et al., 2020b) or decisions based on the knowledge extracted.

First, we will use all the knowledge presented in the concept lattice to analyse the categories, as this tool allows us to *browse* the closed sets of *names* or *identifiers* assigned by the different antivirus, detecting those concepts representative of a threat category. This will lead to the creation of a hierarchy of *malware* threats.

On the other hand, the study of such a hierarchy would not be complete without modelling the dependency and hierarchical relationships between threats of the same category and their detection by different antivirus software. With all this information and knowledge available, the *map* of *malware* threats takes a leap forward both qualitatively and quantitatively. Sharing these results will help to improve systems to defend against attacks and increase security in both transactions and the daily use of our devices.

In the field of cybersecurity, the results of this proposal represent a breakthrough in the mapping of *malware* threats and their characterisation. With the new model available to the community and the tools we make available to the community, experts and antivirus companies can use the new knowledge presented to make their systems more efficient, as well as have keys to be prepared for future threats.

The remainder of this work is structured as follows: Section 2 summarises the main notions of FCA. The methodology proposed to extract the knowledge inside the malware datasets is shown in Section 3, where the dataset is described, as well as the whole process of converting the raw data into a format that can be handled by the FCA techniques, and the results obtained for the analysed dataset, and their interpretations. A discussion on the utility of this method and the relevance of the results is presented in Section 4. Some conclusions and future works are highlighted in Section 5.

2. Preliminaries on Formal Concept Analysis

Since its first appearance in the literature, Formal Concept Analysis (FCA) is thought of as a means of translating knowledge into mathematics (Wille, 1982). Moreover, the attribute logic provides the attribute implications of a semantics, apart from the well-known syntax (Ganter and Wille, 1999). FCA is a helpful tool to extract knowledge from a dataset (called formal context). The formal context is defined as a three-tuple $\mathbb{K} = (G, M, I)$ where G is a set of objects, M is a set of attributes, and I is a relation between G and M (called incidence) with the following interpretation: if the pair $(g, m) \in I$ then we say that the object g has the attribute m. The incidence relation is usually represented by a table where the rows are objects and the columns are attributes. When we find a cross in a cell, we have that the object related to the row has the attribute related to the column.

FCA is closely related to Galois connections, which are two maps $\phi : P \to Q$ and $\psi : Q \to P$ between two ordered sets $(P, \leq)$ and $(Q, \leq)$ satisfying:

1) ϕ is antitone, i.e., $p_1 \leq p_2$ then $\phi(p_1) \geq \phi(p_2)$;
2) ψ is antitone, i.e., $q_1 \leq q_2$ then $\psi(q_1) \geq \psi(q_2)$;
3) for all $p \in P$ and $q \in Q$ we have that:

$$p \leq \psi(q) \qquad \Longleftrightarrow \qquad q \leq \phi(p).$$

Given a formal context $\mathbb{K} = (G, M, I)$, we can define a Galois connection between the set of attributes and objects as follows. The first map, $^{\uparrow} : 2^G \to 2^M$ is defined as $A^{\uparrow} = \{m \in M \mid (g, m) \in I \quad \forall g \in A\}$. That is, given a set of objects $A \subseteq G$, $A^{\uparrow}$ is the set of all the attributes shared for all the objects in A. The second map of the Galois connection, denoted by $^{\downarrow}$, is defined as $^{\downarrow} : 2^M \to 2^G$ with $B^{\downarrow} = \{g \in G \mid (g, m) \in I \quad \forall m \in B\}$. In

Table 1
Formal contexts used in Example 1. Left: original; right: clarified formal context.

	x	y	z	t
a	×	×		×
b		×	×	
c	×		×	×
d		×	×	

	$\{x,t\}$	$\{y\}$	$\{z\}$
$\{a\}$	×	×	
$\{b,d\}$		×	×
$\{c\}$	×		×

other words, given a set of attributes $B \subseteq M$, $B^{\downarrow}$ is the set of all objects that have all the attributes in B. The pair $(^{\uparrow}, ^{\downarrow})$ forms a Galois Connection (Wille, 1982; Ganter and Wille, 1999). Therefore, the compositions ${}^{\uparrow}\circ{}^{\downarrow}$ and ${}^{\downarrow}\circ{}^{\uparrow}$ are closure operators. For the sake of the presentation, hereafter, we omit the symbol $\circ$ to denote such a composition; i.e., we write ${}^{\uparrow\downarrow}$ and ${}^{\downarrow\uparrow}$, respectively. A set C is said to be closed under the Galois connection $(^{\uparrow}, ^{\downarrow})$ if $C^{\uparrow\downarrow} = C$ and dually for the closure of attribute sets.

Once the mappings $^{\uparrow}$ and $^{\downarrow}$ have been introduced, we can define the notion of formal concept, which is a pair $(A, B) \subseteq G \times M$, such that $A^{\uparrow} = B$ and $B^{\downarrow} = A$. The subset A is said to be the extent of the formal concept and B is said to be the intent of the formal concept. Given a formal concept (A, B), all the objects in A share all the attributes in B and do not share any other attributes. Moreover, we can define an order relation between formal concepts; given two formal concepts (A, B) and (C, D), we say that $(A, B) \leq (C, D)$ if and only if $A \subseteq C$ (or equivalently, if and only if $D \subseteq B$). Indeed, this order relation defines a structure of complete lattice in the set of formal concepts, where the supremum and infimum are given by:

$$\sup_{j\in J}(A_j, B_j) = \left((\bigcup_{j\in J} A_j)^{\uparrow\downarrow}, \bigcap_{j\in J} B_j\right) \qquad \inf_{j\in J}(A_j, B_j) = \left(\bigcap_{j\in J} A_j, (\bigcup_{j\in J} B_j)^{\downarrow\uparrow}\right)$$

for any family of formal concepts $\{(A_j, B_j) : j \in J\}$. The complete lattice defined by this order is called the Concept Lattice of the formal context $\mathbb{K} = (G, M, I)$ and we denote it by $\underline{\mathbb{B}}(\mathbb{K})$. In addition, it can be proved that every complete lattice L can be seen as a concept lattice of a certain formal context (Davey and Priestley, 2002, Chapters 3 & 7).

Throughout the paper, we will use the term formal concept for pairs $(A, B) \subseteq G \times M$, as well as for subsets of attributes which are closed under the Galois connection $^{\downarrow\uparrow}$, that is, we identify formal concepts with their intents.

In applications, we have to take into account the time and the computation cost of the algorithms. Therefore, we have to apply certain strategies to reduce the size of the formal context without modifying its implicit knowledge.

One of the most used methods is the clarification of the formal context. A formal context (G, M, I) can be clarified when there exist $m, n \in M$ such that $m^{\downarrow} = n^{\downarrow}$, that is, there are two attributes whose columns are the same. Similarly, if two objects $g, f \in G$ satisfy $g^{\uparrow} = f^{\uparrow}$, their rows are identical, and can be clarified. The result of clarification is a formal context (G', M', I') where the new set of objects contains sets of objects and analogously for attributes: following the example above, $\{g, f\}$ is an object in G' and $\{m, n\}$ would be an attribute in M'. This method is plainly reducing duplicity in the formal context. Consider the following example for illustration.

Example 1. Let $G = \{a, b, c, d\}$ and $M = \{x, y, z, t\}$ be sets of objects and attributes, respectively, and let I be the incidence relation given by Table 1 (left).

Observe that the rows of b and d are identical, as well as the columns of x and t. Therefore the clarified context will be $G' = \{\{a\}, \{b, d\}, \{c\}\}$, $M' = \{\{x, t\}, \{y\}, \{z\}\}$ and the incidence relation is given by Table 1 (right).

Even though clarification can drastically reduce the number of objects and attributes of a formal context, the concept lattice remains unchanged, that is, the formal concepts are in a one-to-one relationship with those of the original context. This is a key result in order to apply clarification without losing the information of the original formal context.

3. Detection of malware hierarchies by means of FCA

In this section, we will show the results of applying the tools from formal concept analysis to the identification of hierarchical relationships between malware threats. We focus on the results of applying FCA methods to the dataset provided by VirusTotal.

Having as goal to provide a valuable and collaborative resource to detect and analyse malware and security threats, and although VirusTotal offers paid services, users can scan suspicious files or URLs for free from their browsers. VirusTotal's free service acts as an information aggregator by scanning the uploaded file with more than 70 antivirus programs and additional tools that inspect the file's characteristics to provide a summary of the results. This information is shared with the user and various antivirus programs to improve their performance.

The first stage of Virustotal's procedure entails gathering and storing critical file features, such as nomenclature, dimensions, file format, compression status, permissions, as well as other notable characteristics. Following the execution of a file, a thorough investigation is carried out on its conduct, covering its interaction with the operating system, including the execution of applications and functions, the modification of other files, and the likelihood of activating system shutdown procedures. It is important to note that a file labelled as "suspicious" can perform various activities, some of which conform to standard and non-malicious file behaviour, highlighting the complexity of threat assessment. Furthermore, subsequent to the evaluation by each antivirus solution, VirusTotal generates a detailed report outlining the existence or nonexistence of detected threats. When a threat is detected, the corresponding antivirus software provides a distinctive identifier alongside the report. Additionally, the version of the antivirus software used is thoroughly recorded to aid in tracking differences between subsequent versions.

Regarding data availability, VirusTotal API (VirusTotal, 2024) provides the method for downloading a threat dataset which can be analysed with other techniques. The repository is annually updated and it is built on a collection of JSON files describing the threats, as detailed above.

All results presented here have been obtained using the `fcaR` library (Cordero et al., 2022), developed by our research team. All the R code and functions, as well as the sample data and an instructions manual to replicate all these results, are stored in the repository (Ojeda-Hernández et al., 2024). To present the results of this section with the appropriate readability, we will use a reduced version of the data set where we only consider the following antivirus: Avast-Mobile, Avira, ESET-NOD32, Kaspersky, McAfee, and Microsoft.

3.1. Data preprocessing

We have downloaded the Android threat dataset to be analysed using our proposal. This dataset consists of 183 JSON files containing information such as the noted in the previous section. However, for our objectives, we will only need the information regarding the detected identifiers given by the 66 different antivirus systems. Thus, let us begin with an example of the relevant parts of a JSON file in Listing 1. We will use this example throughout this section to show the process performed on the data.

In this example, av1, av2 and av3 stand for the names of three antivirus systems. If no malware is detected by an antivirus, then, the "detected" field has the `false` value, otherwise it is `true` and in the "result" field, the name or identifier of the detected malware is stored. In this case, av1 and av3 did not detect malware in the corresponding file, while av2 detected the `tag1` malware.

The first step is to transform the JSON input into a table where: (a) every file is represented in a row; (b) antivirus systems are represented

```
{
  "av1": {
    "detected": [false],
    "version": ["xxxxxxx"],
    "result": {},
    "update": ["yyyyyy"]
  },
  "av2": {
    "detected": [true],
    "version": ["xxxxxxx"],
    "result": ["tag1"],
    "update": ["yyyyyy"]
  },
  "av3": {
    "detected": [false],
    "version": ["xxxxxxx"],
    "result": {},
    "update": ["yyyyyy"]
  }
}
```

Listing 1: Example of anonymized JSON from VirusTotal.

Table 2
Data matrix constructed from the JSON file in Listing 1.

	av1	av2	av3	···
file00001	NA	tag1	NA	···
⋮	⋮	⋮	⋮	⋮

Table 3
Example of scaling matrix used in the preprocessing of the dataset.

	⟨av, ∅⟩	⟨av, tag1⟩	⟨av, tag2⟩	⟨av, tag3⟩	…
NA	×				
tag1		×			
tag2			×		
tag3				×	
⋮					

in columns; (c) the entry (i, j) in the table is the name of the malware detected by antivirus j for file in row i, or *NA* if antivirus j has not detected any malware for file i. Using the above JSON excerpt, the table may look like the one in Table 2.

Now, let us build a formal context from the above table. Note that the values in each column are nominal, therefore, a transformation of each column is necessary to obtain a binary data table. The process by which we perform this transformation is called nominal scaling, and the *mapping* that encodes the nominal values into binary values is called nominal scaling. It is important to note that in other areas of Machine Learning, this process is commonly referred to as *one-hot encoding*.

For instance, assume that an antivirus software known as "av" encodes malware based on the identifiers "tag1", "tag2", etc. In this case, the scale required to encode values defined by "av" is given in Table 3.

Therefore we need to create as many binary columns as possible identifiers are given by the antivirus, plus an extra column (⟨av, ∅⟩) to indicate that the antivirus has not detected any malware. In Table 4, there is an example of the result of this procedure on simulated data.

This procedure is applied attribute-wise, i.e. encoding the identifiers given by all the antivirus systems.

Specifically, for our problem at hand, the dataset provided by VirusTotal consists of 183 files, each of which presents the results generated by 66 different antivirus programs. After the corresponding processing, a formal context $\mathbb{K} = (G, M, I)$ is obtained with $|G| = 183$ files and $|M| = 491$ attributes, with I being the relationship between the file and the detections of the different antivirus. It should be noted that if we presented the results of all the antivirus, we would have a total of more than 1500 attributes, which would reduce the readability of this section.

3.2. Detection of equivalent identifiers

The first step is to apply *clarification* to the formal context to detect equivalent attributes. We say that two attributes ⟨av1, tag1⟩ and ⟨av2, tag2⟩ are equivalent, and we will denote this fact by "⟨av1, tag1⟩≡⟨av2, tag2⟩", if the set of files (objects in our formal context) that have the ⟨av1, tag1⟩ attribute coincides with the set of files having ⟨av2, tag2⟩. In terms of the concept-forming operators:

$$\langle \text{av1, tag1} \rangle \equiv \langle \text{av2, tag2} \rangle \quad \text{if and only if}$$
$$\{\langle \text{av1, tag1} \rangle\}^{\downarrow} = \{\langle \text{av2, tag2} \rangle\}^{\downarrow}$$

From a practical standpoint, based on the information presented within the formal context, it can be concluded that both identification markers provided by separate antivirus programs refer to the same malware. This is due to the fact that these markers are conceptually identical and cannot be differentiated.

In our case, we have also clarified the objects because it reduces the computational cost for calculating concepts, and it does not alter the lattice structure: the lattice of the clarified context is isomorphic to that of the original context. The clarified context, $\mathbb{K}' = (G', M', I')$, where I' represents the restriction of the previous relation I to the final objects and attributes, has a size of 147 objects × 194 attributes. It is remarkable that the context is very sparse, only 2.06% of the entries are non-zero.

A total of 84 multiple attribute equivalences have been found. As an example, in Table 5 we only show the first equivalences found. We can see that some equivalences may be more or less straightforward, for instance, ⟨Avast-Mobile, fakeplayer-i⟩ ≡ ⟨Microsoft, fakeplayer!rfn⟩, and may be derived by using techniques from natural language processing, but the identification of others, such as ⟨McAfee, tripoli⟩≡⟨Avira, banker.faar.gen⟩, need a more careful and sophisticated approach than a simple visual inspection or string matching. In other words, this method allows us to identify and detect matching tags given by different vendors, which can be of great help in generating more complete reports, as well as in the study of consistency between different analysis approaches.

For the sake of clarity, we will briefly explain the meaning of one of the equivalences of the list shown in Table 5:

$$\langle \text{McAfee, !0edb4e42346f} \rangle \equiv \langle \text{ESET-NOD32, addisplay.tapcore.b} \rangle$$
$$\equiv \langle \text{Avira, andr.addisplay.amaa.gen} \rangle$$

Any file flagged by McAfee antivirus on VirusTotal as containing the malware named "!0edb4e42346f" has also been identified by ESET-NOD32 as hosting the malware "addisplay.tapcore.b" and by Avira as

Table 4
Result of scaling a simulated dataset. Left: original dataset; right: the scaled (binary) formal context.

	av
file1	tag1
file2	tag2
file3	NA
file4	tag1
file5	tag3

scales to

	⟨av, ∅⟩	⟨av, tag1⟩	⟨av, tag2⟩	⟨av, tag3⟩
file1		×		
file2			×	
file3	×			
file4		×		
file5				×

Table 5
Equivalences found between identifiers used by different antivirus software. Some identifiers have been simplified to improve clarity.

⟨McAfee, adwind-fdwb.jar!be68bbb422d0⟩	≡ ⟨Avira, .fgxw.gen⟩ ≡ ⟨Microsoft, vigorf.a⟩
⟨McAfee, tripoli⟩	≡ ⟨Avira, banker.faar.gen⟩
⟨McAfee, !02f9c9bb715b⟩	≡ ⟨Microsoft, multiverze⟩
⟨McAfee, !03eaa0113ccf⟩	≡ ⟨ESET-NOD32, spy.androrat.ak⟩ ≡ ⟨Kaspersky, heur:backdoor.os.climap.a⟩ ≡ ⟨Avast-Mobile, androrat-k⟩ ≡ ⟨Avira, androrat.b.gen⟩ ≡ ⟨Microsoft, androrat.a!mtb⟩
⟨McAfee, !041a4a561b51⟩	≡ ⟨ESET-NOD32, spy.smsspy.it⟩ ≡ ⟨Kaspersky, heur:-spy.os.smforw.ff⟩ ≡ ⟨Avast-Mobile, smforw-ut⟩ ≡ ⟨Avira, styricka.d.gen⟩ ≡ ⟨Microsoft, smforw.f⟩
⟨McAfee, !05906bbf77a9⟩	≡ ⟨Avira, andr.agent.fjub.gen⟩
⟨McAfee, !0b40c7b97ac3⟩	≡ ⟨Avast-Mobile, smsreg-cxq⟩
⟨McAfee, !0edb4e42346f⟩	≡ ⟨ESET-NOD32, addisplay.tapcore.b⟩ ≡ ⟨Avira, andr.addisplay.amaa.gen⟩
⟨McAfee, !1014e9dfb449⟩	≡ ⟨ESET-NOD32, sms.agent.ddy⟩ ≡ ⟨Kaspersky, heur:-sms.os.agent.acl⟩ ≡ ⟨Avast-Mobile, fakeplayer-i⟩ ≡ ⟨Microsoft, fakeplayer!rfn⟩
⟨McAfee, !10a9bb90535e⟩	≡ ⟨ESET-NOD32, addisplay.dowgin.bx⟩ ≡ ⟨Kaspersky, not-a-virus:heur:.os.dowgin.i⟩ ≡ ⟨Avira, andr.dowgin.bs.gen⟩

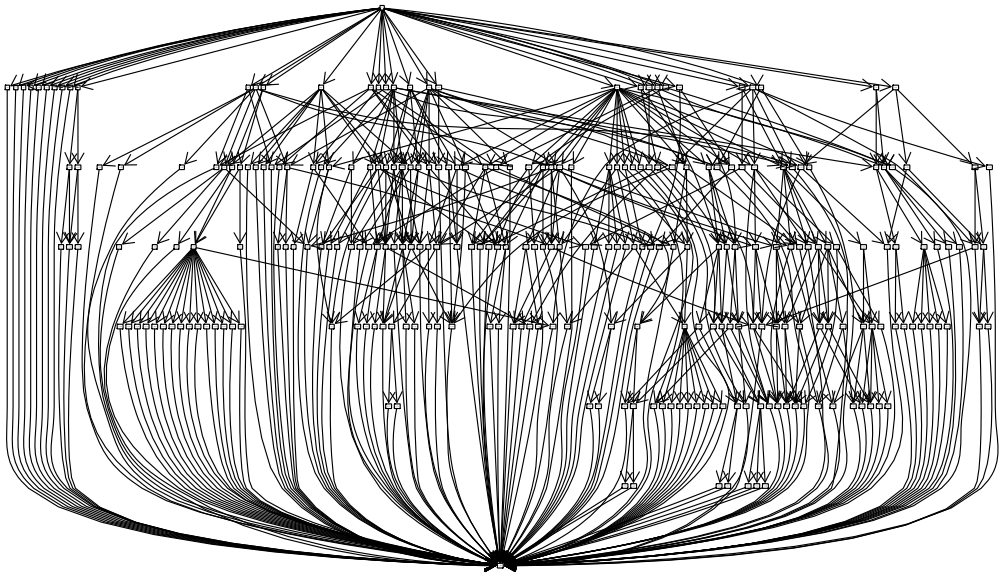

Fig. 1. Concept lattice corresponding to the formal context used in this work.

carrying "andr.addisplay.amaa.gen". The converse also holds true, indicating a mutual agreement between the different antivirus programs on this particular malware. Note that these identifiers are specific to each antivirus, and the *translation* between them is only achievable through contextual clarification.

3.3. Hierarchies from the concept lattice

From the constructed context, we can compute the concept lattice using the NextClosure algorithm (Ganter, 2010). The result is a lattice of 265 concepts. To make it easier to visualise the complex structure of the lattice, in Fig. 1 we will only show the structure of the lattice without the identifiers and antivirus names.

Let us explain in detail and with an example what a concept is in this scenario. Recall that a formal concept is a pair (A, B) where A is a subset of the objects and B is a subset of the attributes, so that $A^{\uparrow} = B$ and $B^{\downarrow} = A$, i.e., the attributes common to the objects in A are those in B, and the objects in A are the only ones that contain all the attributes in B. In our case, A will be a set of JSON files (the objects in our formal context) and B will be a set consisting of identifiers of the type ⟨av, tag⟩ (attributes). For example:

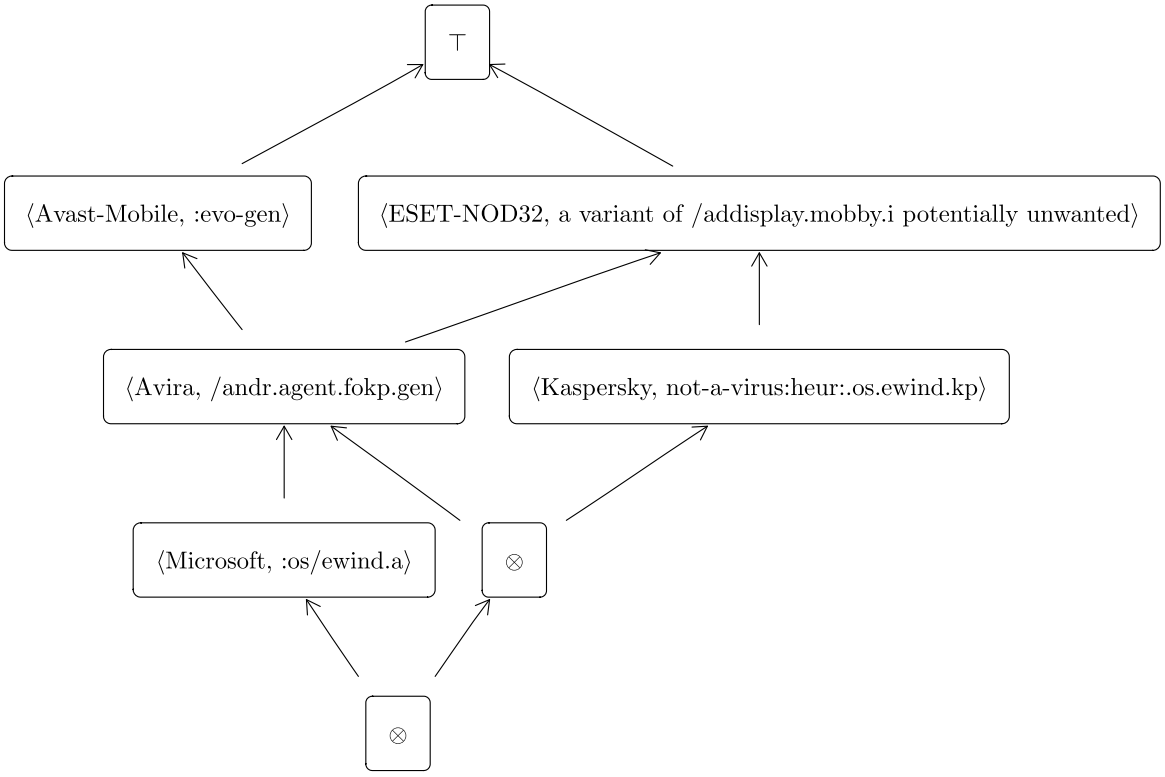

Fig. 2. Sublattice considered for the purpose of understanding the hierarchical organisation of malware categories.

$$(A, B) \quad \text{with} \quad \begin{cases} A = & \{\text{cd581c0251f2b1e7559c4b5830.json}, \\ & \text{fb1bd5c6486f120268c0803901.json}, \\ & \text{44ed7c95e37adfa1e90cc55847.json}\} \\ B = & \{\langle \text{Kaspersky, not-a-virus:heur:.os.adlo.b} \rangle, \\ & \langle \text{Avast-Mobile, :evo-gen} \rangle, \\ & \langle \text{Avira, /agent.mer.gen} \rangle, \\ & \langle \text{Microsoft, :script/wacatac.b!ml} \rangle\} \end{cases}$$

is a formal concept of $\mathbb{K} = (G, M, I)$. The names of the files have been shortened for the sake of readability. The fact that this is a concept means that the identifiers expressed in B are the only ones repeated among the reports of these files. There is no other malware identifier in common among these files, i.e., no other ⟨av, tag⟩ is shared by these files. And conversely, the only three files that have been assigned all the identifiers given in B are those in A. No file other than those in A has all the identifiers in B. This fact permits the identification of concepts with classes or categories of malware, defining the common properties or attributes across multiple files.

In any lattice, there is implicit a *generalisation - specialisation* relationship that can be read vertically: concepts closer to the top ($\top$) are more general than those closer to the bottom ($\bot$). In particular, the subconcepts of a given concept are more specific than the latter. This can be understood as if a concept C is a subconcept of concepts $A_1, \ldots, A_n$, then the extent of C includes the intersection of the extents of all the A_i.

In order to enhance readability, we will abuse the notation and language and identify a concept or category (A, B) with its intent B or with its extent $A = B^{\downarrow}$, since they are uniquely determined. Therefore, when stating that category B_0 is the intersection of categories B_1 and B_2, we formally mean that $B_0^{\downarrow} = B_1^{\downarrow} \cap B_2^{\downarrow}$.

Let's demonstrate how to extract information on the ordering relation between concepts and apply it to the hierarchical relationships between categories and malware identifiers. To explain this process, we will use a sublattice (shown in Fig. 2) of the whole concept lattice as an example. In that diagram, we represent only the reduced version of the lattice, that is, only *attribute concepts* are shown with a text label. The *attribute concept* corresponding to an attribute ⟨av, tag⟩ $\in M$ is the first concept in the lattice where this attribute appears, starting from the top of the lattice, specifically $\left(\{\langle \text{av, tag} \rangle\}^{\downarrow}, \{\langle \text{av, tag} \rangle\}^{\downarrow\uparrow}\right)$. We only show the name of the attribute that *generates* each concept. Note that, although only one label is presented in each concept, the idea of this plot is to show a simplified view of the lattice. Actually every concept's intent contains all the attributes of its superconcepts. The symbol ⊗ is a placeholder for concepts that are not attribute concepts. For instance, the rightmost ⊗ concept has as intent:
⟨Avast-Mobile, :evo-gen⟩, ⟨Avira, /andr.agent.fokp.gen⟩, ⟨ESET-NOD32, a variant of /addisplay.mobby.i potentially unwanted⟩, ⟨Kaspersky, not-a-virus:heur:.os.ewind.kp⟩

Suppose we are given an identifier from an antivirus program ⟨av, tag⟩. We aim to ascertain which identifiers it relates to and the nature of their relationship. From an FCA perspective, we can characterise an attribute ⟨av, tag⟩ $\in M$ by identifying its *attribute concept*, so we will focus our description on studying the location of attribute concepts inside the lattice. Thus, we compute the attribute concept C corresponding to ⟨av, tag⟩. Then, there are several possibilities:

1. If C is a coatom, it can be considered a generic category, it is not a subcategory of another category: its only superconcept is the top concept in the lattice. Therefore we can say that ⟨av, tag⟩ is a generic malware identifier.
   In our example, ⟨Avast-Mobile, :evo-gen⟩ and ⟨ESET-NOD32, a variant of /addisplay.mobby.i potentially unwanted⟩ represent the concepts which are the coatoms in the sublattice, therefore they are generic identifiers representing generic categories of malware.
2. If C is not a coatom:
   (a) If C is a $\wedge$-irreducible element in the lattice, that is, if C has only one upper neighbour (A, B), then we can say that the category C is a proper subcategory of (A, B), since $\{\langle \text{av, tag} \rangle\}^{\downarrow} \subsetneq A$. From the point of view of the identifiers given by anti-virus software, this means that ⟨av, tag⟩ is strictly contained in the

intersection of all the identifiers in B (recall the identification between a concept and its intent mentioned above).

In Fig. 2, we can see that the attribute concept of

$\langle$*Kaspersky, not-a-virus:heur:.os.ewind.kp*$\rangle$

is a $\wedge$-irreducible element as it has only one upper neighbour, $\langle$*ESET-NOD32, a variant of /addisplay.mobby.i potentially unwanted*$\rangle$*. This means that the identifier* $\langle$*Kaspersky, not-a-virus:heur:.os.ewind.kp*$\rangle$ *(its associated concept) is a proper subcategory of the latter. Every file that has been flagged by Kaspersky with "not-a-virus:heur:.os.ewind.kp" has also been identified as "a variant of /addisplay.mobby.i potentially unwanted" by ESET-NOD32, but there are files with this latter identifier that do not possess the former.*

The same situation happens with the attribute concept for

$\langle$*Microsoft, :os/ewind.a*$\rangle$*,*

which is a proper subcategory of $\langle$*Avira, /andr.agent.fokp.gen*$\rangle$.

(b) If C is not $\wedge$-irreducible (C has more than one upper neighbour), let $\{(A_i, B_i)\}$ be the set of the $\wedge$-irreducible elements such that $C = \bigwedge_i (A_i, B_i)$ (by Birkhoff's representation theorem (Birkhoff, 1940)). Then, $\{\langle \text{av, tag}\rangle\}^{\downarrow} = \bigcap_i A_i$. Regarding the identifier $\langle$av, tag$\rangle$, we can then say that it corresponds exactly to the intersection of all the identifiers in $\bigcup_i B_i$.

For our example, we can check that there is an attribute concept which is not $\wedge$-irreducible in the lattice: the one corresponding to

$\langle$*Avira, /andr.agent.fokp.gen*$\rangle$.

Here, $\langle$*ESET-NOD32, a variant of /addisplay.mobby.i potentially unwanted*$\rangle$ *and* $\langle$*Avast-Mobile, :evo-gen*$\rangle$ *are the attributes generating its parent concepts, which are $\wedge$-irreducible. Then, we can deduce that*

$\langle$*Avira, /andr.agent.fokp.gen*$\rangle^{\downarrow} =$
$\langle$*Avast-Mobile, :evo-gen*$\rangle^{\downarrow} \cap$
$\cap\langle$*ESET-NOD32, ... /addisplay.mobby.i potentially unwanted*$\rangle^{\downarrow}$

Note that this is what we have previously termed as

> *"the category* $\langle$*Avira, /andr.agent.fokp.gen*$\rangle$ *coincides with the intersection of categories* $\langle$*Avast-Mobile, :evo-gen*$\rangle$ *and* $\langle$*ESET-NOD32, a variant of /addisplay.mobby.i potentially unwanted*$\rangle$*".*

In the whole concept lattice, these situations can be identified similarly, establishing relationships of inclusion and characterisation of many malware categories. In particular, there are 39 sets of malware that can be characterised as the intersection of other malware categories and 123 categories that are properly contained in the intersection of others. In Tables 6 and 7, we show some of the relationships that can be found when exploring the concept lattice.

As mentioned above, this hierarchical view between concepts, which represent co-occurring sets of tags provided by different vendors, allows us to determine very precisely the relationship between the nomenclatures used by the different actors. It is therefore important to analyse the factors that can influence the quality of an analysis, such as the level of granularity or specificity of the identifying tags, or the coverage of the tags: a high granularity allows us to identify the threat in a file precisely, while a high coverage in the database implies that most of the threats will be captured by the analysis. These two measures of a detection engine's ability to analyse can be defined from the concept lattice, where both the generalisation and the specialisation of the different nomenclatures used can be determined.

4. Analysis of results and discussion

In this section we will outline the results obtained through the application of FCA to the dataset and will point out the similarities and differences that arise from this analysis compared to other approaches.

This paper demonstrates that our field of study can be applied to address real-world issues, such as the analysis of malware threats. Furthermore, it reveals to malware analysts that there are other tools, which are found in the theoretical literature, that can be used to tackle the problems they are facing. But this work does not end in a mere experimentation of the methodology on a given dataset: we must emphasise that FCA, and the knowledge extracted by its means, is able to provide actionable insights for developers of antivirus software or aggregators like Google with VirusTotal in the following way:

- Providing enhanced detection strategies: patterns and relationships in malware data are revealed with our methodology and those can contribute to the development of more effective detection strategies; insights gained from FCA could assist developers in refining algo-

Table 6
A sample of the characterisations found in the concept lattice.

	is equal to the intersection of:
⟨McAfee, !5720b5e8f3bd⟩	⟨ESET-NOD32, .gzb⟩ ⟨Kaspersky, not-a-virus:heur:.os.adlo.a⟩ ⟨Avast-Mobile, evo-gen⟩ ⟨Avira, agent.mer.gen⟩ ⟨Microsoft, script/sabsik.fl.b!ml⟩
⟨Avast-Mobile, spymax-d spy⟩	⟨Avira, ∅⟩ ⟨ESET-NOD32, spy.agent.bat⟩ ⟨Kaspersky, heur:-spy.os.spynote.an⟩
⟨Avira, agent.fjnr.gen⟩	⟨McAfee, ∅⟩ ⟨ESET-NOD32, downloader.agent.jn⟩ ⟨Kaspersky, heur:-downloader.os.agent.jy⟩ ⟨Avast-Mobile, metasploit-q ⟩ ⟨Microsoft, hacktool:os/mesploit.a⟩
⟨Microsoft, hacktool:os/mesploit.a⟩	⟨ESET-NOD32, downloader.agent.jn⟩ ⟨Kaspersky, heur:-downloader.os.agent.jy⟩ ⟨Avast-Mobile, metasploit-q ⟩
⟨ESET-NOD32, jsmshider.o⟩	⟨Microsoft, script/wacatac.b!ml⟩ ⟨Kaspersky, heur:.os.agent.aw⟩ ⟨Avira, smssend.h.gen⟩
⟨Avira, dropper.fjvv.gen⟩	⟨Kaspersky, heur:-dropper.os.hqwar.bk⟩ ⟨Avast-Mobile, evo-gen⟩ ⟨Microsoft, script/wacatac.b!ml⟩

Table 7
A sample of the inclusion relationships found in the concept lattice.

	is contained in the intersection of:
⟨McAfee, !0c5c3d793761⟩	⟨ESET-NOD32, .hvn⟩ ⟨Kaspersky, not-a-virus:heur:.os.adlo.b⟩ ⟨Avast-Mobile, evo-gen⟩ ⟨Avira, agent.mer.gen⟩ ⟨Microsoft, script/sabsik.fl.b!ml⟩
⟨ESET-NOD32, .iyt⟩	⟨Avast-Mobile, ∅⟩ ⟨Microsoft, spy:os/bray.a!mtb⟩ ⟨Kaspersky, heur:-banker.os.bray.pac⟩ ⟨Avira, dropper.fjkv.gen⟩
⟨Kaspersky, not-a-virus:uds:.os.ewind.kp⟩	⟨McAfee, !919a1900c529⟩ ⟨ESET-NOD32, addisplay.mobby.i⟩ ⟨Avast-Mobile, evo-gen⟩ ⟨Avira, andr.agent.fokp.gen⟩
⟨Avast-Mobile, shedun-v⟩	⟨ESET-NOD32, dropper.shedun.v⟩ ⟨Kaspersky, heur:-dropper.os.wapnor.a⟩ ⟨Microsoft, dropper:os/shedun.a!mtb⟩
⟨Avira, agent.hutg⟩	⟨ESET-NOD32, dropper.shedun.v⟩ ⟨Kaspersky, heur:-dropper.os.wapnor.a⟩ ⟨Microsoft, dropper:os/shedun.a!mtb⟩
⟨Microsoft, os/ewind.a⟩	⟨ESET-NOD32, addisplay.mobby.i⟩ ⟨Avast-Mobile, evo-gen⟩ ⟨Avira, andr.agent.fokp.gen⟩

rithms and heuristics for identifying and classifying new malware threats.
- Optimising antivirus: this type of analysis may uncover nuances in how different antivirus programs respond to specific types of malware, helping developers optimise their software for improved performance.
- Standardisation in the nomenclature: highlighting discrepancies in how different engines classify and label malware; promoting industry standardisation; refining the malware detection and filtering mechanisms.

Therefore, we have shown that FCA is an effective tool applicable to malware analysis, a novel application in the literature, as far as our knowledge extends. In particular, the results of this proposal suggest further research and collaboration with cybersecurity industry, translating to them intelligent ways to address malware identification.

Several works have explored the challenge of malware analysis through label comparison. However, a lack of consensus on the nomenclature for different computer viruses results in each antivirus employing its own unique nomenclature. To address this issue, our analysis avoided using different tags and instead focused on the various responses generated by antivirus software when opening corrupted files. By adopting this approach, we were able to identify similarities and differences among different antiviruses and their associated labels, even uncovering equivalences such as, for example, $\langle \text{McAfee}, \text{adwind-fdwb.jar!be68bbb422d0} \rangle \equiv \langle \text{Avira}, \text{.fgxw.gen} \rangle$.

Specifically, through this analysis, we gain insight into the generality or particularity of each label, which is readily discernible through the specialization-generalization hierarchy presented by the concept lattice.

Here we can outline for example $\langle \text{Microsoft}, \text{os/ewind.a} \rangle$ being strictly contained in the intersection of $\langle \text{ESET-NOD32}, \text{addisplay.mobby.i} \rangle$, $\langle \text{Avast-Mobile}, \text{evo-gen} \rangle$ and $\langle \text{Avira}, \text{andr.agent.fokp.gen} \rangle$, that is, the denomination for a threat labelled with the identifier $\langle \text{Microsoft}, \text{os/ewind.a} \rangle$ only rises if all the other labels are also assigned. Since the inclusion is strict, even if all the other labels are detected it might be the case that the tag $\langle \text{Microsoft}, \text{os/ewind.a} \rangle$ is not present. Consequently, this label is highly specific, surpassing the specificity of those mentioned earlier.

The primary limitation of this study is that it is not exhaustive. Just like any other analysis using FCA, the inclusion of new information expands the initial formal context, potentially discovering more current knowledge through the acquired data. This new knowledge that we would obtain may be different in the sense of giving a finer and more precise knowledge of the problem. The data used in this study were extracted from VirusTotal via its API as an initial step toward addressing the challenge of malware categorisation. Based on the outcomes of this preliminary analysis, we anticipate ongoing collaboration to obtain larger datasets and formulate more specific inquiries. This iterative process aims to further develop the theory for resolution. For example, while this study focuses on the assignments of different labels and antiviruses, it omits considerations of permissions or activities associated with distinct malwares. Consequently, there is ample room for improvement and enrichment of the analysis in future research. Since one single antivirus can contain several distinct labels, this analysis cannot discern whether a certain company provides a better service than another. This problem is considerably harder and could be tackled in the future.

Unfortunately, starting from the same dataset there are no metrics that can evaluate the effectiveness and reliability of this process to compare it with other approaches in the literature. However, any other method applied on the table would give a subset of the knowledge we got by applying FCA due to the concept lattice having the highest granularity in the co-occurrence of malware labels. Thus, it would be the "ground truth" to compare with on a philosophical level.

5. Conclusions and future work

We have presented how Formal Concept Analysis (FCA) is able to extract useful knowledge to solve relevant issues in the cybersecurity area, specifically to make an intelligent malware analysis based on an algebraic and logic framework. The problem was revealed in a masterclass to our computer science students in ETSI Informática at the University of Málaga. A real dataset generated by VirusTotal with threats and the results of the different antivirus was provided.

The methods shown in this paper have been a first approach to solving the posed problem. This shows how even with the most direct application of FCA we obtained knowledge that was unknown by experts in the field. This is due to the novel approach of studying the calls of the distinct antiviruses instead of comparing the labels each antivirus gives to each threat. Since this is intended to be a continued collaboration, when more specific problems are proposed we could develop the theory further.

The results obtained using FCA have shown that creating a hierarchy of malware threats is an effective approach to the problem of intelligent malware detection. The proposed methodology has successfully applied FCA to the identification of hierarchical relationships between malware. The approach has involved modelling the dependency and hierarchical relationships between threats of the same or different categories and their detection by different antivirus software, using formal methods. More precisely, the concept lattice allows us to deduce the specialisation-generalisation hierarchy in categories. It paves the way for the use of formal reasoning methods (Mora et al., 2012; Cordero et al., 2012) to get a deeper insight into this problem.

The hierarchy of malware threats obtained from our study provides a significant contribution to the field. It is a valuable resource that can be used by various stakeholders, including researchers, antivirus software developers, and security experts. By sharing this information, we can promote collaboration and standardisation in the field of malware detection, which will ultimately lead to better protection against new and emerging threats.

In conclusion, our study demonstrates that the use of FCA as a tool for creating a map or hierarchy of malware threats is a viable approach to the problem of intelligent malware detection. It provides an effective method for understanding the characteristics and properties of malware, which can lead to better protection against emerging threats. Future work can involve extending this approach to include more comprehensive datasets and integrating additional sources of information to create a more comprehensive map of malware threats. Also, the future possibilities of retrieving more complex knowledge using more advanced FCA methods seem promising. A recommender system based on FCA (Cordero et al., 2020a) could be useful for malware detection. The results of this work will be of great value in the ongoing fight against cyber threats.

CRediT authorship contribution statement

Manuel Ojeda-Hernández: Writing – review & editing, Writing – original draft, Visualization, Validation, Project administration, Funding acquisition, Formal analysis, Conceptualization. **Domingo López-Rodríguez:** Writing – review & editing, Writing – original draft, Validation, Software, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Ángel Mora:** Writing – review & editing, Writing – original draft, Validation, Supervision, Investigation, Formal analysis, Data curation, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

This work has been partially funded by the predoctoral contract FPU19/01467 (MCIU), the "VALID" project with reference PID2022-140630NB-I00 (MCIN/ AEI/ 10.13039/ 501100011033) and the research project with reference PID2021-127870OB-I00 (MCIU/AEI/ ERDF, EU).

References

Basole, Samanvitha, Stamp, Mark, 2021. Cluster Analysis of Malware Family Relationships. Springer International Publishing, Cham. ISBN 978-3-030-62582-5, pp. 361–379.

Birkhoff, Garrett, 1940. Lattice Theory, vol. 25. American Mathematical Soc.

Cordero, Pablo, Enciso, Manuel, Mora, Ángel, Ojeda-Aciego, Manuel, 2012. Computing minimal generators from implications: a logic-guided approach. In: Proceedings of the Ninth International Conference on Concept Lattices and Their Applications, pp. 187–198.

Cordero, Pablo, Enciso, Manuel, López-Rodríguez, Domingo, Mora, Ángel, 2020a. A conversational recommender system for diagnosis using fuzzy rules. Expert Syst. Appl. 154, 113449. https://doi.org/10.1016/j.eswa.2020.113449.

Cordero, Pablo, Enciso, Manuel, Mora, Ángel, Ojeda-Aciego, Manuel, Rossi, Carlos, 2020b. A formal concept analysis approach to cooperative conversational recommendation. Int. J. Comput. Intell. Syst. 13 (1). https://doi.org/10.2991/ijcis.d.200806.001.

Cordero, Pablo, Enciso, Manuel, López Rodríguez, Domingo, Mora, Ángel, 2022. fcaR, formal concept analysis with R. R J. (ISSN 2073-4859) 14, 341–361. https://doi.org/10.32614/RJ-2022-014.

Davey, B.A., Priestley, H.A., 2002. Introduction to Lattices and Order, second edition. Cambridge University Press, Cambridge.

Escudero Garcia, David, DeCastro-Garcia, Noemi, 2021. Optimal feature configuration for dynamic malware detection. Comput. Secur. 105, 102250. https://doi.org/10.1016/j.cose.2021.102250.

Firdaus, Ahmad, Anuar, Nor Badrul, Karim, Ahmad, Faizal, Mohd, Razak, Ab, 2018. Discovering optimal features using static analysis and a genetic search based method for Android malware detection. Front. Inf. Technol. Electron. Eng. 19 (6), 712–736. https://doi.org/10.1631/FITEE.1601491.

Ganter, Bernhard, 2010. Two basic algorithms in concept analysis. In: Formal Concept Analysis: 8th International Conference, ICFCA 2010. Agadir, Morocco, March 15-18, 2010. Proceedings 8. Springer, pp. 312–340.

Ganter, Bernhard, Wille, Rudolf, 1999. Formal Concept Analysis - Mathematical Foundations. Springer.

Kellogg, Lee, Ruttenberg, Brian, O'Connor, Alison, Howard, Michael, Pfeffer, Avi, 2014. Hierarchical management of large-scale malware data. In: Proceedings of the 2014 IEEE International Conference on Big Data (Big Data), pp. 666–674.

Kouliaridis, Vasileios, Kambourakis, Georgios, 2021. A comprehensive survey on machine learning techniques for Android malware detection. Information (ISSN 2078-2489) 12 (5). https://doi.org/10.3390/info12050185.

Ling, Xiang, Wu, Lingfei, Deng, Wei, Qu, Zhenqing, Zhang, Jiangyu, Zhang, Sheng, Ma, Tengfei, Wang, Bin, Wu, Chunming, Ji, Shouling, 2022. MalGraph: hierarchical graph neural networks for robust windows malware detection. In: Proceedings - IEEE INFOCOM, pp. 1998–2007.

Liu, Yannan, Lai, Yabin, Wei, Kaizhi, Gu, Liang, Yan, Zhengzheng, 2020. Nlabel: an accurate familial clustering framework for large-scale weakly-labeled malware. In: 2020 IEEE 19th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom), pp. 210–216.

Maggi, Federico, Bellini, Andrea, Salvaneschi, Guido, Zanero, Stefano, 2011. Finding non-trivial malware naming inconsistencies. In: International Conference on Information Systems Security. Springer, pp. 144–159.

Martín, Ignacio, Hernández, José Alberto, de los Santos, Sergio, 2018. SignatureMiner: a fast anti-virus signature intelligence tool. In: 2018 IEEE Conference on Communications and Network Security (CNS), pp. 1–2.

Martín, Ignacio, Hernández, José Alberto, de los Santos, Sergio, 2019. Machine-learning based analysis and classification of Android malware signatures. Future Gener. Comput. Syst. (ISSN 0167-739X) 97, 295–305. https://doi.org/10.1016/j.future.2019.03.006.

Mora, Angel, Cordero, Pablo, Enciso, Manuel, Fortes, Inmaculada, Aguilera, Gabriel, 2012. Closure via functional dependence simplification. Int. J. Comput. Math. 89 (4), 510–526. https://doi.org/10.1080/00207160.2011.644275.

Nadeem, Azqa, Hammerschmidt, Christian, Gañán, Carlos H., Verwer, Sicco, 2021. Beyond Labeling: Using Clustering to Build Network Behavioral Profiles of Malware Families. Springer International Publishing, Cham. ISBN 978-3-030-62582-5, pp. 381–409.

Namanya, Anitta Patience, Cullen, Andrea, Awan, Irfan U., Disso, Jules Pagna, 2018. The world of malware: an overview. In: Proceedings of the IEEE 6th International Conference on Future Internet of Things and Cloud, FiCloud, pp. 420–427.

Ojeda-Hernández, Manuel, López-Rodríguez, Domingo, Mora, Ángel, 2024. Github repository. https://github.com/Malaga-FCA-group/VirusTotal-Malware-Hierarchy. (Accessed 6 June 2024).

Robb, Brenda, 2024. The state of ransomware in 2023. https://www.blackfog.com/the-state-of-ransomware-in-2023/. (Accessed 6 June 2024).

Salem, Aleieldin, Banescu, Sebastian, Pretschner, Alexander, 2021. Maat: automatically analyzing VirusTotal for accurate labeling and effective malware detection. ACM Trans. Priv. Secur. (ISSN 2471-2566) 24 (4). https://doi.org/10.1145/3465361.

Sebastián, Silvia, Caballero, Juan, 2020. AVclass2: massive malware tag extraction from AV labels. In: Annual Computer Security Applications Conference, ACSAC '20. New York, NY, USA. ISBN 9781450388580. Association for Computing Machinery, pp. 42–53.

VirusTotal, 2014. Virustotal-free online virus, malware and URL scanner. https://www.virustotal.com/en. (Accessed 6 June 2024).

VirusTotal, 2024. VirusTotal API v3 overview. https://docs.virustotal.com/reference/overview. (Accessed 6 June 2024).

Walker, Aaron, Shukla, Raj Mani, Das, Tapadhir, Sengupta, Shamik, 2022. Ohana means family: malware family classification using extreme learning machines. In: 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC), pp. 534–542.

Wang, Shuai, Zhao, Yuran, Liu, Gongshen, Su, Bo, 2021. A hierarchical graph-based neural network for malware classification. In: Mantoro, Teddy, Lee, Minho, Ayu, Media Anugerah, Wong, Kok Wai, Hidayanto, Achmad Nizar (Eds.), Neural Information Processing. Springer International Publishing, pp. 621–633.

Wille, Rudolf, 1982. Restructuring lattice theory: an approach based on hierarchies of concepts. In: Ordered Sets. Springer, pp. 445–470.