ANDROIDGYNY: Reviewing clustering techniques for Android malware family classification

THALITA SCHARR RODRIGUES PIMENTA, Federal Institute of Parana, Brazil
FABRICIO CESCHIN, Federal University of Parana, Brazil
ANDRE GREGIO, Federal University of Parana, Brazil

Thousands of malicious applications (apps) are daily created, modified with the aid of automation tools, and released on the World Wide Web. Several techniques have been applied over the years to identify whether an APK is malicious or not. The use of these techniques intends to identify unknown malware mainly by calculating the similarity of a sample with previously grouped, already known families of malicious apps. Thus, high rates of accuracy would enable several countermeasures: from further quick detection to the development of vaccines and aid for reverse engineering new variants. However, most of the literature consists of limited experiments—either short-term and offline or based exclusively on well-known malicious apps' families. In this paper, we explore the use of malware phylogeny, a term borrowed from biology, consisting of the genealogical study of the relationship between elements and families. Also, we investigate the literature on clustering techniques applied to mobile malware classification and discuss how researchers have been setting up their experiments.

CCS Concepts: • **Security and privacy** → **Malware and its mitigation**; • **Computing methodologies** → *Cluster analysis.*

Additional Key Words and Phrases: Classification, Mobile Malware, Phylogeny

1 INTRODUCTION

Thousands of variants of malicious programs (a.k.a malware) are created per day due to multiple writing techniques, such as code adaptations and code generations [1]. Therefore, using clustering methods, analysts can group the plethora of similar unlabeled samples and handle them. Moreover, sophisticated obfuscation techniques and new tactics are used to prevent and evade anti-malware [2]. Due to this fact, even with the multitude of academic work and deployed security solutions, there is still urgency for effective and heuristic methods for automatic detection of malicious software and their family classification. For many years, the malware categorization into families has been primarily done manually, through a process that typically requires memorization, library description study, and sample collection research. This manual process is time-consuming and exhaustive [3]. Specifically, Miller et al(2016) [4] reports that a modern company can only afford to label 60 malware samples per day.

Furthermore, malware classification suffers from increasing complexity, mainly because of the number of features required to represent application behavior exhaustively. Consequently, the investigation of millions of records of data and features collected from malicious codes becomes impracticable to be carried out without a high degree of automation.

Authors' addresses: Thalita Scharr Rodrigues Pimenta, thalita.pimenta@ifpr.edu.br, thalita.pimenta@ifpr.edu.br, Federal Institute of Parana, 100 Pedro Koppe Street, Irati, Parana, Brazil, 5584500-039; Fabricio Ceschin, Federal University of Parana, Curitiba, Brazil, fjoceschin@inf.ufpr.br; Andre Gregio, Federal University of Parana, Curitiba, Brazil, gregio@inf.ufpr.br.

2576-5337/2023/3-ART $15.00
https://doi.org/10.1145/3587471

Recently, analysts and researchers have proposed several representation attempts of malicious features and behaviors. For example, examples of the most common data representation methods are binary information, control flow graph(CFG), API-level features and behaviors, and Dalvik bytecode frequency analysis [5]. Through manual or computational analysis, after detecting the malicious behavior of a program, the code can be categorized as a known family variant or as an element of a new family. From this point on, the labeling problem arises. This issue, related to the classification of malware pointed out since the 1990s, refers to the names of malware. A pattern of a malicious file label consists of a hierarchical structure with the type, platform, family, and abbreviation or numbering of the family variant [6]. However, a common practice is creating a name for each analyzed malicious sample. As the appointment is a subjective process, most antivirus companies develop different strategies to name their samples, which results in many specimens of malware referenced with distinguished names [7].

There have been attempts at standards and schemes of malware labeling. However, in practice, the difficulties in analysis and classification remain [8]. Researchers have presented models for label unification, for Android malware, such as Euphony [9] and ANDRE [10].

For commercial antivirus purposes, the detection of malicious traits is more critical than grouping into families and correct labeling. Nevertheless, concerning classification using data mining algorithms, correct labels are fundamental for the training data [11]. Therefore, family variant labels are essential for separating malicious code and also determining the family lineage [12]. Understanding the origin and lineage of polymorphic malware may lead to new techniques for detecting and classifying unknown attacks [13].

Research on malware classification is not recent: it dates from the beginning of the 1990s. The vast majority of papers published in this area today aim to classify samples of malicious code in families. A family contains malware specimens (variants) that implement the same functions or have similar behaviors or structures. Grouping malware samples also aid in the defense, removal, or analysis of malicious software. For example, malware may have alike weaknesses, similar structures, and effects, all of which can be identified by improved detection methods [14]. Thus, clustering techniques and lineage research about relationships between families and variants are significant to security analysts and end-users.

In this paper, we present the types of clustering methods, their disadvantages, strengths, and examples of algorithms in Section 3. Also, we address the main articles available in the field of clustering and phylogenetic analysis of mobile malware, especially the Android platform. We discuss their broadness, advantages, limitations, repeatability, and feasibility of application in the wild.

The remainder of the paper is divided as follows: Section 2 overview concepts about mobile malware types, malware analysis mechanisms, including extraction and selection techniques, and Android malware datasets. Section 3 presents several clustering methods, strengths, and weaknesses. Besides, it includes information about similarity indices, linking criteria, and clustering validation indexes. Section 3.5 briefly introduces phylogeny concepts, especially for computing. On Section 4, we examine the main techniques used in the analyzed studies. Furthermore, we propose an approach to malware clustering. Finally, Section 5 displays the final considerations.

2 BACKGROUND AND RELATED WORK

In previous decades, the primary focus of malware analysis was malicious code targeting desktop operating systems, especially the detection and classification of Windows malware. However, since 2009, the growth of mobile malware has been impressive. Before this year, fewer than 1,000 mobile malware were known [15]. Although there are several surveys regarding mobile malware detection, these articles do not focus on clustering techniques. Besides, a similar article on malware clustering does not address mobile malware and is brief, not examining into concepts of clustering and phylogeny. In Table 1, we show related surveys on malware classification and clustering.

Table 1. Related Surveys of Malware Classification and Clustering

Paper (1st author)	Scope	OS	Classification
Mohini[16]	The authors present a study of the Android security model, application-level security, and security issues of the Android platform.	Android	Binary
Gandotra[17]	This is a review of techniques for analyzing, detecting, and classifying malware.	Windows	Binary
Feizollah[18]	This article is an analysis of feature selection in mobile malware detection.	Android	Binary
Arshad[19]	The review addresses diverse anti-malware techniques, their benefits, and their limitations.	Android	Binary
Riasat[20]	A systematic study based of android malware detection in official and unofficial market.	Android	Binary
Baskaran[21]	In this paper, the authors present an investigation of the evolution of Android malware detection techniques based on machine learning algorithms.	Android	Binary
Yadav[22]	The article addresses the development of detection and existing analysis methods of Android malware.	Android	Binary
Malhotra[23]	The authors reviewed malware detection techniques on the mobile platform.	Android, iOS and Symbian	Binary
Tam[15]	This work is an extensive survey involving Android malware analysis and detection techniques.	iOS, Windows OS, Palm, BlackBerry,Symbian and Android	Binary
Zacharia[24]	The survey is a brief discussion of Android malware detection techniques.	Android	Binary
Yan[25]	The authors compare mobile malware detection methods based on evaluation criteria and measures.	Windows, iOS and Android	Multiclass
Yu[26]	A survey involving malware behavior description, behavior analysis methods, and visualization techniques.	Windows and Android	Binary
Odusami[27]	The authors investigate malware detection techniques for identifying gaps.	Android	Binary
Hamed[28]	This review covered techniques of malware detection by using different systems of cloud-based intrusion detection.	iOS, Windows OS, Palm, BlackBerry,Symbian and Android	Binary
Alswaina[29]	The authors highlight the identified limitations in the literature, challenges, and future research directions regarding the Android malware family.	Android	Multiclass
Kumars[30]	This survey contains strategies for Android Malware Detection focusing on methodology and frameworks to classify, cluster, and extract Android malware features. Android	Binary	
Meijin [31]	This article presents a systematic review of Android malware detection, including information about feature extraction, raw data preprocessing, valid feature subsets selection, and machine learning-based selection models.	Android	Binary

Another recent survey that addresses a crucial issue with Android malware classifiers is that of Pendlebury et al [32]. The authors discuss how to remove experimental bias in malware classification. Also, they implemented TESSERACT, an open source evaluation framework for comparing Android malware classifiers in a realistic setting.
The process of clustering Android malware follows a standard set of steps. In Figure 1, we present a flowchart of the activities performed and methods applied during malware analysis and their assigned families. The following subsections cover methods available in the mobile malware classification and detection literature.

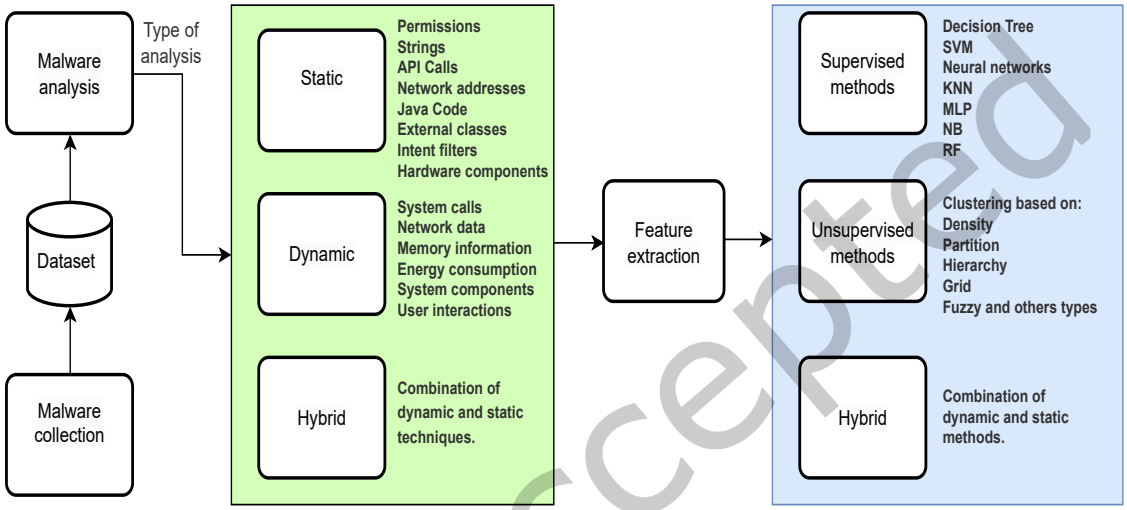

Fig. 1. Mobile Malware Analysis Flowchart

2.1 Mobile Malware: Types and Operating Systems

From the end user's point of view, the information that their device is infected is usually enough to look for ways and software to remove malware. However, for security analysts, and professionals responsible for forensics and incident response, information about families can help damage contention, alert spreading, decision-making, and mitigation tasks.
Qamar et al (2019) have generated a timeline of mobile malware detection between 2000 and 2018 [33]. By analyzing the timeline created, we noted that between 2000 and 2007, most found malware were to Symbian OS, usually worms, trojans, and spyware. As of 2008, despite we found some malware on Ios and Windows, the most significant amount is on Android. Thus, one reason for the growth of malware Android is that this operating system is open, and its app store (Google Play) is not closely monitored for security [34].
Dagon et al (2004) present a taxonomy based on committed security goals: Confidentiality, Integrity, and Availability [35]. In the Confidentiality category are threats of data, such as bluebugging and bluesnarfing. In the second category (integrity) appears Phone hijacking. In the category of Evaluability, the authors mention Protocol-based denial-of-service and battery-draining attacks.
Baskaran et al (2016) cite other taxonomy from the existing and known vulnerabilities. According to the authors, the main types of Mobile threats are [21]:

- **Update Attack:** This type of attack involves a benign application installed on the operating system that downloads malicious third-party updates and installs them on the system. The detection is complex.
- **Information Extraction:** Attacks and codes which objective to steal user data, such as IMEI numbers.
- **Messaging and Automatic Calls:** Codes used to make calls and send messages from attacked and compromised devices.
- **Root Exploits:** mechanisms used to gain administrator privileges, take device control, and make changes in the system.
- **Optimization for search engines:** In this type of attack, the code simulates clicks on specific links to enlarge website traffic, increasing revenue from these pages.
- **Botnets:** Attackers can use compromised device networks for malicious actions, including DDoS attacks and spam delivery.

Chakraborty et al. (2017) reinforce that a security analyst, besides an accurate predictor, needs to understand which features discriminate between different malware families. These findings are valuable in engineering robust signatures for new malware variants detection [36].
According to the literature, there is a window of time for security analysts and end-users to notice new variants. For Android malware analysis, researchers indicate that one can spend about three months to detect new malware [37].
In the malware categorization process, families contain samples which similar attack techniques and structures [38]. According to Malik et al(2017), we can categorize Android malware families by their installation method on devices [39]. Some categories include:

- **Installation Method:** The principal methods to install on devices are repackaging, update attack and drive-by download. Repackaging occurs when malware authors use popular apps to disassemble them and enclose malicious payloads, to then re-assemble. The update attack technique is more complicated to detect. In addition to repackaging, update components download the malicious payload at run time [40].
- **Activation:** Malware use events, such as boot completion, package removing/installation, SMS or call phone activities, and network connectivity tasks to activate.
- **Malicious Payload:** The most common types of malware payload are privilege escalation, remote control, financial, charges, and personal information stealing [40].

Understanding the type of malware, for example, banking trojans, allows you to apply specific solutions to remove the threat.

2.2 Malware Collection and Malware Android Datasets

Regarding mobile malware data acquisition, the most used methods are real-time acquisition, online acquisition, use of datasets, and sandboxes. [33].

Table 2 provides information on some existing malicious APK datasets. In Figure 2, we show information about the most popular datasets for the Android platform in recent years.

Concerning the number of available samples, the smallest datasets are Kharon (60 elements), M0Droid (193 elements), and Contagio (237 elements). The two largest are Androzoo, with over 76.000 distinct families, and Koodous.

The first public Android malware dataset was released in 2011, and one of the most popular is Malgenome. This set includes 1.260 elements of 49 Android malware families collected between August 2010 and October 2011.

Moreover, it is interesting to note that the PRAGuard dataset contains modifications and obfuscation of the specimens present in Malgenome and Contagio.

The Drebin dataset, publicly available in 2014, has 5.560 specimens from 179 families. The collection period was between 2010 and 2012.

Information about the AAGM dataset indicates that 250 adware malware, 150 generic malware, and 1500 benign applications belong to the suite. In addition, the samples correspond to variants of 12 families.

Wei et al. (2017) have created the AMD database, providing information with family information and characteristics, including variant behavior reports [41].

Table 2. Android Malware Datasets. The URLs of M0Droid and AMD Project were not available at the time this essay was in progress.

Name	Collection	Samples	Families	URL
Malgenome	2011-2015	1.260	49	http://www.malgenomeproject.org
Drebin	2010-2012	5.560	179	https://www.sec.cs.tu-bs.de/ danarp/drebin
M0Droid	2011-2013	193	-	http://cyberscientist.org/m0droid-dataset
Koodous	2017	> 100.000	-	https://koodous.com
VirusShare	2016	34.140.004	-	https://virusshare.com
Androzoo	2016	10.147.916	76000	https://androzoo.uni.lu
PRAGuard Dataset	2015	10.479	-	http://pralab.diee.unica.it/en/AndroidPRAGuardDataset
AAGM Dataset	2017	1.900	12	https://www.unb.ca/cic/datasets/android-adware.html
AMD Project	2017	4354	42	http://amd.arguslab.org
Kharon Dataset	2016	60	19	http://kharon.gforge.inria.fr/dataset
Contagio	2011-2013	237	-	http://contagiodump.blogspot.com
Piggybacking	2011-2016	1.136	29	https://github.com/serval-snt-uni-lu/Piggybacking
RmvDroid	2014-2018	9.133	56	https://doi.org/10.5281/zenodo.2593596
AndMal2017	2015-2017	426	42	https://www.unb.ca/cic/datasets/andmal2017.html
AndMal2020	2006-2020	400.000	191	https://www.unb.ca/cic/datasets/andmal2020.html
MalDroid20	2017-2018	17.341	-	https://www.unb.ca/cic/datasets/maldroid-2020.html

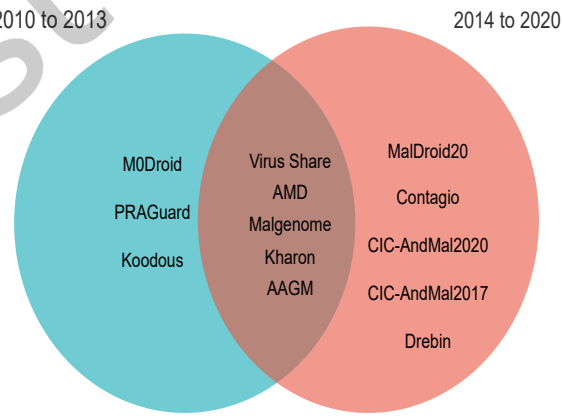

Fig. 2. Datasets and years of collection

2.3 Mobile Android Malware Analysis

The malware analysis techniques aim to investigate samples to determine their functionality, extracting as much information as possible [7]. From the analysis and discovery of its structure, behavior, and weaknesses, "vaccines" can be developed.

Due to the particularities of mobile malware, analysis requires sophisticated techniques. A critical point of mobile phones is the sensor and event-based system. As with desktop malware analysis, static, dynamic, and hybrid analysis methods are employed [42]:

- **Static analysis:** This analysis type involves disassembling and source code analysis. In this method, it is possible to obtain information without malware running. Analysts perform investigations based on signatures, permissions, and component analysis. As in the traditional static analysis, the major limitation of this scheme refers to the shortage of treatment of special execution conditions and the investigation of the execution paths. Conventional techniques are detection based on the signature, API Calls, commands, dangerous permissions, hardware components, intent, and dex files. Most found static techniques in the literature investigate confidentiality threats. There are gaps related to confidentiality, integrity, and availability threats. [43].
- **Dynamic analysis:** The behavior of malicious code while being executed in a controlled environment is analyzed. When using this type of analysis, we can view information about the device's memory, operating conduct, response time, and typical device performance data [44]. The following techniques are part of static analysis: anomaly-based detection, Tairu analysis, emulation-based detection, out-of-box analysis, in-box analysis, and virtualization. Besides the energy consumption, the analysis process could employ processing and network connection, so it is more suitable to opt for cloud analysis.
- **Hybrid analysis:** This method combines static and dynamic analysis techniques, possibly increasing robustness and code investigation coverage, and enables the analyst to find more vulnerabilities [15].
- **Cloud-based analysis:** Due to the battery and processing device limitations, the analysis process occurs in the cloud, using replicas and emulators of what is running on the mobile phone, returning the outcomes to the mobile device. One benefit is that the malware detection and classification techniques occur on the analysis servers [45].

Feizollah et al (2015) [18], Schmeelk et al(2015) [43], Zacharian et al (2017) [24], Tam et al (2017) [15], Baskaran et al (2016)[21], Arshad et al (2016) [19] and Bakour et al (2019) [46] presented reviews of the types of analysis with more details.

Regarding the feature extraction method, it is noteworthy that the quality directly implies the accuracy of the malware classification algorithm. Consequently, characteristics that represent the structure and behavior of the analyzed variant should be selected [7].

2.4 Attributes, Features Extraction, and Features Selection Techniques

In the study of clustering and phylogenetic analysis of mobile malware, it is essential to emphasize that the resulting quality depends on the extracted features and the analysis type of the malicious code. The following is information about available approaches.

Machine learning approaches need the relevant features extraction of the samples and their families. Features are applied elements or characteristics to classify an application as either malicious or benign code [47]. The features are either categorical or numerical sequences in nature [38].

For mobile malware, the following types of attributes can be listed from [48], [21], and [33]:

- **Static Attributes**: Include sensitive and restricted permissions, Java code, strings, hardware components, APIS calls, native commands, XML elements, application metadata, .dex file opcodes, task intent, and hardware usage analysis.

- **Dynamic Attributes**: system calls, network traffic, SMS, process information battery usage, Dalvik and native memory, CPU consumption, WIFI status, dynamic code execution, file activities, information leakage, system components, user interactions, number of active connections, data packets sent and other data.
- **Other extraction schemes**: Use of metadata (application description, developer ID, and application category), native calls, and hybrid scenarios (a combination of statistical and dynamic characteristics).

Experiments presented in [49] suggest that combining intents and permissions are more efficient for malware detection than just using the application's whitelist.

2.5 Features Extraction and Features Selection

Wang et al (2019) discuss problems related to feature extraction. For example, the main problem is the required time to obtain information from the files. Also, the process can generate millions of features with a value of 0. How to process the vector of values efficiently is also a challenge. Regarding extraction through statistical analysis, the authors cite the difficulty of obtaining data from malicious applications that present advanced obfuscation and encryption techniques. Concerning dynamic analysis, they consider that applications may have runtime protection mechanisms [50].

In a feature extraction process, we can collect thousands of attributes. Thus, the selection feature objective is to clarify the data transference to the classification model. After extracting features, it is necessary to reduce the size of the dataset. Many features may not be valuable for classification. Two popular approaches to this type of selection are attribute-based and subset-based. However, not every feature selection method is suitable for clustering tasks.

Table 3 contains forms of data representation frequently used in malware analysis. Some significant characteristics are:

- **Opcode:** Opcodes or opstrings are abbreviations for operation codes, instruction syllables, and instruction packages. These codes are part of the machine language, which specifies the operations which the device will perform.
- **String extraction:** Analysis of the plain text presents in the application's binary files, including external class names, window strings, and other graphical items [51].
- **Function Call Sequences:** Binaries can be reasonably well identified as ordinary or malicious, looking only at the names of functions and calls that appear in their codes [52].
- **Control flow graphs - CFG:** a program's CFG displays all the possible paths that an executing program can use [53]. The size of the generated graph proportionally increases as far as the computational complexity.
- **Dangerous API calls:** Some techniques detect calls to specific APIs and requests for dangerous permissions. Thus, if the program makes at least one of these calls, the sample is considered unreliable [54].
- **Network and communication data**: Most malware need a network connection to spread and share data from their malicious activities. From network monitoring logs, extracted traces assist in detecting the malware behavior [55].

Table 3. Feature extraction and data representation.

Type	Description
N-grams	Programs are broken into strings, can be bytes of assembly instructions

	lines of code, and so on [56]. Examples: byte code 2-gram, opcode 3-gram
Binary Frequency	The binary vector is dictionary-sized, where: 0 = absence of the term, 1 = presence of the element in the sample code. Example: binary vector of app permissions
Frequency Feature Extraction	In this type of extraction, the vector contains the number of occurrences of dictionary terms, not just the absence and presence of the feature [57]
Frequency Weigthing Factors	Besides calculating the frequency of terms, methods such as TF-IDF use weights to determine the importance of the attribute in the feature set.
Hidden Markov Model	Although not considered merely a form of feature extraction, HMMs are stochastic models that represent states, having satisfactory results in detecting malicious code.

2.6 Features Selection for Clustering

Feature selection differs in supervised and unsupervised methods, mainly due to the class attribute present in supervised methods. Generally, we use sampling techniques to reduce noise and find the most notable features of the set. Yet, for both clustering and classification tasks, feature selection methods can be divided into [58]:

- **Filters:** This method is applied to estimate the performance of features in datasets. The term filter derives from the idea that irrelevant attributes are filtered from the database before the classification algorithm [59].In clustering approaches, all features are evaluated individually by calculating the sample's pairwise similarity. They are generally slower than others feature selection categories, especially with large amounts of data. They are generally slower than others feature selection categories, especially with large numbers of samples and features.
- **Wrappers:** Unlike the filters approach, Wrapper methods are related and operate with clustering algorithms. These approaches evaluate subsets of variables to detect the potential relations among variables [60]. Despite the high computational cost, these methods generally reach more significant clustering performance when compared to filters. [58].
- **Hybrid approaches:** The hybrid feature selection method merges the Filter method with the Wrapper method, combining the computational efficiency of the first and the predictive accuracy of the last approach [61].

- **Embedded:** Embedded approaches usually develop cluster labels through clustering algorithms. Thus, these methods transform unsupervised feature selection into sparse learning-based supervised feature selection with the generated cluster labels [62].

3 CLUSTERING AND PHYLOGENY APPLIED TO MALWARE SAMPLES

Clustering is an unsupervised learning technique used to find unexpected patterns in data. Cluster Analysis is a technique for grouping objects into clusters by one measure of pre-established similarity [63]. Elements in the same group are more similar to each other than objects that are in another defined cluster.
Given the difficulty of analyzing all the possibilities of groups within a large amount of data, many techniques are available to assist cluster formation.
Careful cluster analysis requires methods that provide the following characteristics [64]:

- Scalability to handle large amounts of data and multiple dimensions;
- Capacity to cluster different types of data and objects;
- Ability to define groupings of different sizes and shapes;
- Require minimum knowledge for the correct determination of input parameters;
- Run successfully even in the presence of noise;
- Display consistent results regardless of the order of the presentation of the data.

The clustering task is highly dependent on the parameters, similarity measures, and methods used by the algorithm [65]. Therefore, section 3.1 discusses information on distance and similarity measures.

3.1 Distances and Similarities

In the literature, there are concepts of similarity and distance. While the similarity between two close objects has a value of 1, the distance has a value of 0, and vice versa. Similarity and distance are inverses of each other [38]. Thus, similarity measures are numerical standards to demonstrate how related two objects are. In Table 4, we display prevalent measures in the literature.

Table 4. The main similarity measures found in the literature.

Name	Description	Equations
Manhattan distance	This measure calculates the distance between two points measured along axes at right angles [66].	$MD(i,j) = \sum_{i=1}^{n} \lvert i_x - j_x \rvert$, where i and j = the first and second point, respectively. The i $_x$ and j $_x$ are the components of the points i and j, respectively
Euclidean distance	The Euclidean distance between two points is the length of the line segment connecting them [66].	$ED(x,y) = \sum_{i=1}^{n} \sqrt{v_i - u_i}$, where u_i and v_i are the components of the points i and j, respectively.
Cosine distance	This similarity employs the cosine of the angle between two vectors (v and u) of the feature space from points x and y, respectively [66].	$Cosine(x,y) = 1 - \cos\beta = 1 - \frac{v.u}{\lvert\lvert v\rvert\rvert.\lvert\lvert u\rvert\rvert}$
Jaccard Index	This measure estimates the similarity between sample sets A and B using the intersection size divided by the size of the union of the sample sets [67].	$Jaccard(a,b) = \frac{\lvert a \cap b \rvert}{\lvert a \cup b \rvert}$
Normalized Compression Distance	NCD is a measure based on the application of the compressor to compute the ratio of similarity of two files. [68].	$NCD(p,q) = \frac{C(pq) - min\{C(p).C(q)\}}{max\{C(p).C(q)\}}$
Jaro Winkler	Adaptation of the metric Jaro [69]. This measure employs the number and order of characters familiar to two strings to calculate the similarity [70].	$J = \frac{m}{3a} + \frac{m}{3b} + \frac{m-t}{3m}$, where m is the number of correlations between characters; a and b are the respective sizes of the strings compared; t is the number of transpositions.
Pearson Correlation Coefficient	This coefficient measures statistical relationship (association) between two variables [71].	$PCC = \frac{\sum_i (X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_i (X_i - \overline{X})(Y_i - \overline{Y})}}$

For categorical sequences, we can employ techniques such as Sequence alignment, The Longest Common Subsequence, and N-gram analysis [38].
As mentioned earlier, n-grams are substrings of a long string with length N [72]. Sequence alignment is a method for ordering one sequence above the other to identify the equivalence among similar substrings. The longest Common Subsequence (LCS) algorithm finds the most extended consecutive sequence of common items for all possible prefix combinations of the input strings [73].

According to Borbely (2016), there are still challenges in choosing appropriate data understanding algorithms, the optimal combination techniques, and their parameters for clustering tasks [74].

3.2 Analysis of Clustering Types

Clustering algorithms include various categories and diverse methods, such as the ones listed below.
Partitioning Methods: Subject that there are n objects in the original set, partitioning methods can split the group into k partitions. The best-known algorithms in this category are k-means and k-medoid, which deal with an idea of a center of gravity or a central object. All methods in this group have the same grouping quality and the same problems [75]: you need to know the number of clusters in advance; Clusters with extensive variations in size are difficult to identify, and the method is suitable for concave spherical clusters.

- **Strengths**: Linear scalability. Some partition-based algorithms, such as CLARANS, handle outliers well [76].
- **Weaknesses**: It deals only with numeric data [77]. It is sensitive to the order of recorded data [78].

Hierarchical Methods: A hierarchical grouping produces a differentiated tree, often called a dendrogram. In this tree, the objects are the leaves, and the inner nodes reveal a similar structure to the points. If the grouping hierarchy is formed from the bottom up, each data object is initially a cluster by itself, so the small clusters are merged into larger groups at each cluster level. Ranking until all data objects are in a group. This type of hierarchical method is called agglomerative.
The reverse process is called divisive hierarchical grouping. Several new hierarchical algorithms have appeared in recent years. The main difference between all these hierarchical algorithms is how to measure the similarity between each pair of clusters [75].

- **Strengths**: No initial number of clusters required. Versatility and applicability with different types of attributes [79].
- **Weaknesses**:They are not scalable. Difficulty finding an optimal number of clusters depending on dataset size [80], [81], [36], [82]

Density-based methods: Density-based algorithms are an attempt to address the need for a skillful method to find clusters in arbitrary formats. In these strategies, the idea of groups represents the existence of dense areas of data, separated by regions with low density. Some examples of these algorithms are DBSCAN, OPTICS, and DENCLUE.

- **Strengths**: It is noise resistant and can handle grouping of varying sizes and shapes [83].
- **Weaknesses**: These methods do not handle well when varying density clusters or extensive data. [84] [36]

Grid methods: Grid-based algorithms first quantify the grouping space into a finite number of cells, which forms a grid. The analysts perform all the operations in this grid. Examples of algorithms are Sting, Wavecluster, and Click.

- **Strengths**: These algorithms are suitable for parallel processing and increment updating [85].
- **Weaknesses**: The clustering result is sensitive to the granularity [86].

Fractal-based methods: This clustering method uses data auto-similarity properties. First, the algorithm divides the data into sufficiently large subclusters with high compression. In the second step, the algorithm merges complex subclusters closer to each other. This technique uses a more natural and fully deterministic method to generate the final clusters [87]. An example of the algorithms used in this type of clustering is FD3 [88].

- **Strengths**: Large efficiency and great scalability, dealing with outliers effectively [89].
- **Weaknesses**:the results are firmly sensitive to the parameters [86].

Methods based on Models: In this scheme, the reference algorithms models are used for each grouping to optimize the curve between the analyzed elements through a mathematical model. Thus, these clustering

algorithms are ideal for unknown distributions, such as a mixture of elemental distributions [90]. In this type of clustering, each cluster corresponds to a different distribution; usually, distributions are considered Gaussian. One of the best-known examples is the Expectation-Maximization Algorithm - EM.

- **Strengths**: This clustering category automatically identifies the optimal number of clusters. Also, this type of clustering regards the probability of samples belonging to each group [91].
- **Weaknesses**:Cost and high computational time [92].

Graph-based methods (Graph Theory): The representation of a clustering problem data can use a graph, where a node represents each set element. For modeling the distance, the method employs a specific weight related to the edge that connects two elements. Examples of algorithms are HCS, DTG, CLICK, and CAST [93].

- **Strengths**: These algorithms fit for the data set with random shape and high dimension [86].
- **Weaknesses**: The number of clusters needed to be preset [94].

Methods based on Distribution: This scheme assumes that the data is composed of distributions, such as Gaussian distributions. While the center distance of the distribution increases, the probability of a point belonging to that distribution decreases. DBCLASD [95] is an example of such a clustering method algorithm.

- **Strengths**: Moderately high scalability and supported by the well-developed statistical science [86]
- **Weaknesses**: Relatively high time complexity and a strong influence of many parameters [86].

Methods based on Neural Networks: Competitive neural networks, including RNA, can be models for clustering. They are commonly called self-organizing networks and are typically methods without supervision. Two famous algorithms are SOM and Adaptive Resonant Theory (ART) [96]. The Kohonen map (SOM) uses the concept of neighborhood. This type of network learns to recognize sections of neighborhoods in the input space and the topology of the input vectors on which training takes place [97]. Typical applications of the ART method include radar signal recognition and image processing.

- **Strengths**: Easy interpretation ability to handle large amounts of data and complex data [98].
- **Weaknesses**: A good dataset is critical; Determining which factors are relevant to the problem can be a complicated task [99].

Methods based on Evolutionary Computing: Evolutionary algorithms, such as genetic algorithms, can be applied in cluster analysis. In this clustering method, the algorithms evaluated characteristics that best represent the elements, and the operators for selection, mutation, and crossing of [100]. Examples of these algorithms are the genetic KM [101] and the genetically guided algorithm [102].

- **Strengths**: Experiments have shown compact and well-separated clusters [103] .
- **Weaknesses**: The computational effort still is an issue [104].

Fuzzy-based methods: This type of clustering allows an individual to partially belong to more than one group, with varying degrees of the neighborhood. Cluster boundary elements are not required to belong entirely to a cluster. However, degrees of association between 0 and 1 are assigned, indicating their partial association [105]. Generally, these models are standard in pattern recognition. The Fuzzy c-means and Fuzzy c-shells algorithms are examples of fuzzy clustering.

- **Strengths**: This method has the advantage of robustness for ambiguity and maintains more information than any rigid clustering method [106].
- **Weaknesses**: High sensitivity to noise [107].

Kernel based methods: Kernel Algorithms feature space to allow nonlinear separation in the input. Kernel-based clustering applies this procedure and works the implicitly grouped by a Kernel method, which performs a mapping not appropriate linear input data for a high-quality feature space substituting the internal

product between the nonlinear variables with an appropriately determined kernel [108]. K-Means Kernel and SVC - Support Vector Clustering are examples of this type of clustering.

- **Strengths**: Theses methods are able to identify the non-linear structures [109].
- **Weaknesses**: Time complexity is large. Complex Algorithms in nature [109].

We emphasize that the clustering task is highly dependent on the parameters, similarity measures, and methods used by the algorithm [65].

3.3 Linkage measure

Hierarchical algorithms are clustering techniques that create a hierarchy of relationships between objects. They work in two modes: the agglomerative, which constructs clusters from isolated elements.
In the divider approach, the process starts with a broad set, which is broken into parts until it reaches isolated elements [110]. The principal aspect of the hierarchical algorithms is the selection of pairs based on the linkage function.
The linkage measure uses similarity or distance indices between the elements of the set. In this process, similar elements are grouped into a single cluster, reducing the total number of remaining clusters. One of the essential uses of the agglomerative method is to identify the point at which two groups of elements are too distinct to form a homogeneous cluster. With increasing distance values between samples, it is ideal to stop the clustering process avoiding the resulting groups from being too dissimilar [111].
Some of the linkage methods are the following:

- **Maximum linkage or Complete Linkage:** This method, also known as a Complete Linkage or Nearest Neighbor, has two clusters with the closest maximum distance merged. This process repeats until there is only a single cluster left. Complete linkage is also called farthest neighbor linkage [112].
- **Minimum linkage or Single Linkage:** This measure involves estimating the distance between clusters using the elements with the greatest distance between them. For this reason, the single linkage is sometimes called nearest neighbor linkage [111].
- **Average linkage:**Also known as UPGMA or Unweighted Pair Group Method with Arithmetic Mean. This measure uses the average pair-wise proximity between their member individuals of all different groups to merge clusters [113]. The UPGMA method is relative to its weighted variant, the WPGMA method [111].
- **Centroid Method:** Also known as UPGMC: Unweighted Pair Group Method with Centroid Averaging. In this type, the distance between the centroids of the groups defines the proximity between them [114]. UPGMC employs only Euclidean distances [115].
- **Ward Method:** For Ward's linkage, two clusters are merged by calculating the total sum of square derivations from the mean of a cluster [112].
- **Median Method:** In this form, the distance between two clusters is the median distance between an element in one group and an observation in the other cluster [112].

After the decision on the most appropriate method and its application, we generate the clusters. Besides, the interpretation of the generated clusters is a complex step, which aims to discover meanings related to the area of the analyzed data sets.

3.4 Clustering Validation Techniques

Clustering validation estimates the quality of clustering results. This validation is one of the essential issues to the success of clustering applications [116], [117]. There are Internal Criteria and External Criteria for validation. External Criteria evaluate the outcome regarding a prespecified structure, while internal criteria estimate the result with respect to information intrinsic to the data alone [118]. Table 5 presents the most well-known indexes of internal and external validation of clustering.

Table 5. Cluster Validation Indexes: The first four indices are for internal validation. The other three (purity, entropy, and F-measure) are for external validation.

Name	Description	Equations
Calinski-Harabasz index	The measure of Calinski–Harabasz is one of the most usual approaches for estimation clustering algorithms quality. A higher value indicates a better division into clusters [119] [120], [10]	$H = \frac{trace(S_b)}{trace(S_w)} . \frac{n_p - 1}{n_p - k}$, where S(b) = the between-cluster scatter matrix, S(w)= the internal scatter matrix, (p)= the number of clustered samples, and k = the number of clusters
Davies-Bouldin Index	This index, proposed in 1979, aims to identify compact and well-separated groups. A lower value indicates a better division into clusters.[118],[119] [121], [119]	$DB = \frac{1}{c} . \sum_{i=1}^{c} Max_{\mathrm{i}} \neq j\{\frac{d(X_{\mathrm{i}}) + d(X_{\mathrm{j}})}{d(c_{\mathrm{i}}, c_{\mathrm{j}})}\}$,where c = number of clusters, i, j = cluster labels d(X i), d(X j) = all the samples in clusters, i and j = their respective cluster centroids, d(c i, c j) = distance between these centroid [118].
Dunn Index	This index produces some lacks, which is why diverse adaptations and generalizations have appeared in the literature [122]. A high value indicates a valid clustering solution [118],[5], [121]	$D = min_{1 \leq i \leq c}\{min\{\frac{d(c_i, c_j)}{max_{1 \leq k \leq c}(d(X_k))}\}\}$, where d(c i),c j) = the intercluster distance between cluster, X i and X j d(X k) = the intracluster distance of cluster (X k), c = the number of cluster of dataset [118]
Silhouette Index	One of the most common indices for internal validation. The samples have been "well-clustered" when the result is close to 1 [123],[120], [124], [5], [125], [126]	$s(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)}\}$, where a(i) is the average distance between the i sample and all of the samples included in X_j; 'max' is the maximum operator, and b(i) is the minimum average distance between the i sample and all of the samples clustered in $X_{\mathrm{k}}(k = 1, ..., c; k \neq j)$ [123].
Entropy	A high value of purity indicates optimal clustering. How the entropy is a negative measure, low entropy values mean more reliable clustering [127], [128], [129], [130]	$E(D_{\mathrm{i}}) = -\sum_{i=1}^{k} Pr_{\mathrm{i}}(c_{\mathrm{j}}) log_2 Pr_{\mathrm{i}}(c_{\mathrm{j}})$ where $E_{\mathrm{total}}(D) = \sum_{i=1}^{k} \frac{\lvert Di \rvert}{\lvert D \rvert} . E(D_{\mathrm{i}})$ and Pr(c) = the proportion of data points of c_j in cluster i or D_i [131] .
Purity	Purity is similar to entropy. However, using this index, a high value of purity indicates optimal clustering [127],[129], [130], [132]	$p(c_{\mathrm{j}}) = \frac{1}{\lvert C_{\mathrm{j}} \rvert} max(\lvert c_{\mathrm{j}} \rvert_{\mathrm{class\ =\ i}})$, where k=number of clusters, $\lvert C_{\mathrm{j}} \rvert$ = the size of cluster c_j, $\lvert C_{\mathrm{j}} \rvert_{\mathrm{class\ =i}}$ = the number of points of class i assigned to the cluster j. The total purity for the clusters is a weighted sum of the individual purities [132].

F-measure	A high F- Measure value indicates better clustering quality. [118], [133], [9], [130],[134]	$F(i,j) = \frac{2Recall(i,j)Precision(i,j)}{Precision(i,j)+Recall(i,j)}$ where $Recall(i,j) = \frac{nij}{ni}, Precision(i,j) = \frac{nij}{nj}$ and n_{ij} = the number of elements of class i that are in cluster j, n_i,n_j = the number of elements in cluster i and j, respectively. [118]

3.5 Phylogeny

Phylogeny is the genealogical history of a group of organisms and a hypothetical representation of ancestral/descendant relationships [135].

Binary trees (with or without root), phylogenetic networks, or graphs are indicated to represent the relationship between species [136]. For rooted trees, the root describes the common ancestor of all species represented in the tree. When an ancestor is not specified or assumed, a rootless tree is applicable for representing relationships between groups, regardless of the common ancestor. The nodes in phylogeny denote the unit of taxonomy or species.

Phylogeny differs from taxonomy while the first one uses a group with shared characteristics as base, whereas it takes on a common ancestor and seeks to derive relationships among its descendants [137].

Computational phylogenetics is the application of computer algorithms, methods, and programs for phylogenetic analysis. The phylogenetics of biology inspires malware phylogeny research. We can adapt methods commonly used in biology to construct, evaluate, and compare phylogenetic trees in malicious programs research [56].

Even if there are apparent differences between malicious software and species, by considering software products as species and source code as genes, software evolution can be investigated in the same way as biological evolution [138].

The most common tree-building methods apply distance, parsimony, maximum likelihood, and Bayesian approaches. For example, one of the most common methods of building phylogenetic trees is the MP - Maximum Parsimony Method [139]. This method assumes that the most accurate tree is the one that requires the least changes to produce the data contained in the alignment [140]. Another popular method is maximum likelihood estimation.- MLE is used to estimate the parameters of a statistical model.

The MP and MLE methods are character-based, i.e, they depend on the individual values of each alignment sequence. There are other methods known as distance-based methods, which, when estimating phylogeny consider an entire taxonomy sequence, often using a distance matrix. The simplest distance-based method is the Unweighted Pair Group Method (UPGMA).In this method, the sequence pair (or sequence group) to be grouped first is the one with the shortest distance between all sequence pairs (or groups).

Both character-based and distance-based methods have advantages and disadvantages. The UPGMA method, for example, is easy to implement but is not widely used for phylogenetic tree creation because it does not take into account different rates of evolutionary change between sequences represented in the [141] tree. Character-based methods are more faithful in the evolutionary representation of phylogeny but are more computationally expensive to implement.

One of the recurring problems in phylogenetic analysis is the size of the generated tree, especially in sets with thousands of specimens. This issue is not easily solved by just using the zoom and panning tools [142]. Besides, there is no clear separation of hierarchical structure lines and labels by any available technique.

Although tree-based models are the main ones in the literature, they are not suitable for representing cross-linking events, which can occur in malware generation [143]. Thus, phylogenetic networks emerged as an

alternative way to model evolution. These networks are graphs in which each labeled sheet represents an instance, and nodes can have more than one parent.

4 EVALUATION OF ANDROID MALWARE CLUSTERING/PHYLOGENY APPROACHES

We reviewed several studies that involved clustering Android malware into families or Android malware phylogeny. In this section, we describe our methodology and our main findings.

4.1 Methodology

The activities carried out can be divided into:

- **Search:** To specify the papers for evaluation, we searched in electronic databases, such as *Google Scholar* and *Scopus*. The filter keywords were: *malware families* AND *phylogeny* AND *classification*, with data from 2010, totaling 541 initial results.
- **Exclusion criteria:** As exclusion criteria, we decided to evaluate only works with unrestricted access. After reading the abstracts, as a second criterion, we excluded studies with a limited objective of detecting *malware*. From these specifications, a set of 82 articles resulted. We examined the following information: the study objectives, number of samples informed by the authors, datasets mentioned, distance/similarity measure, clustering method used, use of supervised and unsupervised methods together, clustering results, and whether the authors applied metrics for clustering results validation.
- **Summarization:** By observing Table 1, the first three columns refer to the type of analysis performed. The Hybrid option indicates that the work used concomitant static and dynamic analysis. The next set of columns refers to the type of features used. The last two groups of columns are the use of supervised and unsupervised methods for clustering malware into families. Due to the great diversity of elements in the literature, such as distance measures, analysis tools, types of representation, and datasets, the papers analyzed were divided into seven tables, by year, for better visualization by the reader.

Other relevant information is available in the charts of Figure 3 to Figure 7 and condensed below. Table 6 presents information on articles dated between 2011 and 2015.

Table 6. Analyzed papers from 2011 to 2015 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	2011	2012	2013			2014					2015					
	Type/Ref	[144]	[134]	[145]	[146]	[130]	[147]	[92]	[148]	[48]	[149]	[150]	[82]	[151]	[80]	[152]	Total:
Analysis	*Static*		✓	✓	✓	✓	✓		✓		✓	✓		✓		✓	10
	Dynamic	✓						✓		✓			✓		✓		5
	Hybrid																0
Features	*Permissions*			✓		✓								✓			3
	API Calls										✓						1
	Opcodes																0
	System Calls																0
	Network Data	✓								✓					✓		3
	Code				✓				✓			✓	✓			✓	5
	Others		✓	✓			✓	✓									4
S.M	*K-means*			✓		✓		✓		✓				✓			5
	HAC						✓		✓				✓				3
	EM					✓		✓			✓						3
	Fuzzy	✓															1
	Birch														✓		1
	DBSCAN																0
	Proposed Model				✓											✓	2
	Phylogeny																0
	Others						✓	✓		✓		✓					4
U.M	*Not mentioned*	✓			✓	✓	✓	✓		✓	✓	✓	✓		✓	✓	11
	SVM																0
	KNN		✓						✓								2
	Decision Tree			✓													1
	RF			✓													1
	NB		✓											✓			2
	Logistic																0
	Neuro Fuzzy																0
	Others																0

In the period of the articles in Table 6, there was a 200% increase in malware collected between 2012 and 2013, according to a Mcafee report [153]. With the large volume of new samples for investigation, despite the disadvantages of static analysis, we observe that this type of analysis still prevails as the most applied. Analysts prefer it mainly due to the time limitation, computational power restriction, and the number of tools available for reverse engineering and APK decompression.

Regarding the features used, we highlight the code analysis, permissions, and linking network data connected with dynamic analysis. In this period, the analyzed studies using supervised methods with clustering algorithms were still succinct.

Table 7 presents information on articles dating from 2016.

Table 7. Analyzed papers from 2016 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	**2016**								
	Type/Ref	[154]	[155]	[156]	[157]	[158]	[159]	[160]	[161]	Total:
Analysis	*Static*		✓				✓		✓	3
	Dynamic	✓		✓	✓	✓		✓		5
	Hybrid									0
Features	*Permissions*						✓			1
	API Calls			✓						1
	Opcodes		✓							1
	System Calls					✓		✓		2
	Network Data				✓					1
	Code			✓						1
	Others	✓			✓				✓	3
S.M	*K-means*						✓			1
	HAC			✓	✓	✓		✓	✓	5
	EM									0
	Fuzzy									0
	Birch	✓								1
	DBSCAN									0
	Proposed Model									0
	Phylogeny		✓							1
	Others					✓				1
U.M	*Not mentioned*	✓	✓	✓	✓			✓	✓	6
	SVM									0
	KNN									0
	Decision Tree						✓			1
	RF									0
	NB									0
	Logistic									0
	Neuro Fuzzy									0
	Others					✓				1

In 2016, we noticed an increase in the use of HAC, both with static and dynamic analysis. Between 2016 and 2017, there was an increase in the application of dynamic analysis techniques, as we can visualize in Table 7. A fact that possibly contributed to this increase was that some tools popularly used for dynamic analysis of Android malware were published during this period, such as Droidmate [162] and Droidbot [163]. Another important fact for the period was the release of Google's multilayer architecture in an Android security review in 2017. However, as most devices were still running outdated system versions, application exploits continued to spread [164].

Table 8 presents information about the studied articles from 2017.

Table 8. Analyzed papers from 2017 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	**2017**														
	Type/Ref	[165]	[166]	[36]	[167]	[168]	[169]	[9]	[170]	[171]	[172]	[173]	[174]	[175]	[176]	Total:
Analysis	*Static*	✓	✓	✓		✓	✓	✓	✓	✓	✓	✓	✓		✓	12
	Dynamic				✓									✓		2
	Hybrid															0
Features	*Permissions*	✓	✓	✓					✓			✓	✓		✓	7
	API Calls										✓		✓		✓	3
	Opcodes															0
	System Calls				✓		✓									2
	Network Data													✓		1
	Code										✓	✓				2
	Others					✓		✓		✓			✓			4
S.M	*K-means*					✓			✓			✓	✓	✓		5
	HAC									✓					✓	2
	EM	✓										✓				2
	Fuzzy												✓			1
	Birch															0
	DBSCAN	✓									✓					2
	Proposed Model		✓	✓			✓	✓								4
	Phylogeny				✓											1
	Others											✓				1
U.M	*Not mentioned*		✓	✓	✓			✓		✓	✓		✓	✓		8
	SVM											✓			✓	2
	KNN														✓	1
	Decision Tree											✓			✓	2
	RF					✓						✓				2
	NB															0
	Logistic															0
	Neuro Fuzzy	✓														0
	Others	✓					✓		✓			✓				4

The data from the analyzed articles from 2017 demonstrate the tendency of analyzing application permissions strongly connected to static analysis. By viewing Table 8, the preference for K-means has remained constant.

Data from the articles from 2018 are presented in Table 9.

Table 9. Analyzed papers from 2018 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	2018													
	Type/Ref	[177]	[178]	[179]	[180]	[181]	[182]	[183]	[184]	[185]	[186]	[84]	[187]	[188]	Total:
Analysis	*Static*				✓			✓			✓	✓	✓	✓	6
	Dynamic	✓		✓		✓									3
	Hybrid		✓				✓		✓	✓					4
Features	*Permissions*							✓					✓		2
	API Calls		✓	✓	✓				✓						4
	Opcodes	✓								✓		✓			3
	System Calls										✓				1
	Network Data					✓	✓								2
	Code							✓							1
	Others								✓	✓				✓	3
S.M	*K-means*					✓	✓	✓			✓	✓		✓	6
	HAC											✓			1
	EM														0
	Fuzzy	✓				✓									2
	Birch											✓			1
	DBSCAN		✓									✓			2
	Proposed Model			✓					✓		✓	✓	✓		5
	Phylogeny	✓			✓					✓					3
	Others											✓			1
U.M	*Not mentioned*	✓	✓	✓		✓	✓	✓	✓	✓					8
	SVM				✓							✓			2
	KNN				✓									✓	2
	Decision Tree				✓							✓			2
	RF				✓							✓			2
	NB											✓	✓		2
	Logistic											✓			1
	Neuro Fuzzy														0
	Others										✓				1

In 2018, static analysis use continued to predominate, similarly permissions as the main feature. By viewing Table 8, we can notice a growth in the number of articles combining supervised and unsupervised algorithms. The diversity of clustering algorithms has also expanded, including DBSCAN, Fuzzy, and EM. Also, API calls are features used in all three types of malware analysis.

Table 10 presents data on the articles dating from 2019.

Table 10. Analyzed papers from 2019 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	**2019**													
	Type/Ref	[189]	[190]	[191]	[192]	[193]	[194]	[195]	[196]	[197]	[198]	[199]	[10]	[126]	Total:
Analysis	*Static*	✓	✓			✓			✓			✓	✓	✓	7
	Dynamic			✓				✓			✓				3
	Hybrid				✓		✓			✓					3
Features	*Permissions*				✓		✓	✓				✓			4
	API Calls		✓		✓		✓								3
	Opcodes				✓	✓								✓	3
	System Calls			✓											1
	Network Data														0
	Code				✓								✓		2
	Others	✓							✓	✓	✓				4
S.M	*K-means*	✓		✓	✓			✓	✓						5
	HAC											✓			1
	EM														0
	Fuzzy														0
	Birch													✓	1
	DBSCAN														0
	Proposed Model		✓			✓	✓				✓		✓		5
	Phylogeny									✓					1
	Others	✓			✓		✓				✓	✓		✓	6
U.M	*Not mentioned*		✓	✓		✓				✓		✓			5
	SVM				✓			✓	✓				✓	✓	5
	KNN	✓			✓				✓				✓		4
	Decision Tree	✓											✓		2
	RF	✓											✓		2
	NB				✓				✓				✓		3
	Logistic	✓													1
	Neuro Fuzzy														0
	Others	✓			✓		✓				✓		✓		5

We observed that Android's new malware development decreased in 2018 [200]. From the second half of 2019 onwards, the development of new Android malware samples increased consistently. In this context, dynamic analysis is interesting for studying malicious code structures and developing vaccines. Research with dynamic and hybrid analysis increased in this period (Table 10.) Articles were labeled as Hybrid Analysis when using static and dynamic techniques together.

Table 11 presents data on the studied articles dating from 2020, while Table 12 presents data on the studies dated between 2021 and 2022.

Table 11. Analyzed Papers from 2020 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	**2020**						
	Type/Ref	[201]	[202]	[203]	[204]	[205]	[206]	Total:
Analysis	Static	✓	✓			✓	✓	4
	Dynamic			✓	✓			2
	Hybrid							0
Features	*Permissions*	✓	✓					2
	API Calls	✓	✓				✓	3
	Opcodes					✓		1
	System Calls			✓	✓			2
	Network Data	✓	✓					2
	Code							0
	Others	✓	✓					2
S.M	*K-means*	✓					✓	2
	HAC							0
	EM							0
	Fuzzy							0
	Birch							0
	DBSCAN							0
	Proposed Model		✓			✓		2
	Phylogeny			✓	✓			2
	Others							0
U.M	*Not mentioned*			✓	✓			2
	SVM						✓	1
	KNN							0
	Decision Tree		✓					1
	RF	✓						1
	NB							0
	Logistic							0
	Neuro Fuzzy							0
	Others	✓				✓		2

Just by viewing the data from the analyzed articles from 2020, it can be complex to notice a significant trend. Constant changes in the malware scenario consequently impact this research area. In 2018 and 2019, there was an outbreak of Android ransomware [207]. Despite the number of mobile attacks by this type of malware decreasing in 2020, we had the beginning of the COVID-19 pandemic. With this atypical moment, several malicious applications using this disease as a theme were the focus of cybercriminals [208].

Table 12. Analyzed papers from 2021 and 2022 with information about analysis type, employed features, and use of supervised and unsupervised methods

	Year	**2021**										**2022**		
	Type/Ref	[209]	[210]	[211]	[212]	[213]	[214]	[215]	[216]	[217]	[218]	[219]	[220]	Total:
Analysis	*Static*	✓	✓	✓		✓	✓	✓	✓	✓	✓			9
	Dynamic											✓	✓	2
	Hybrid				✓									1
Features	*Permissions*	✓			✓	✓		✓		✓				5
	API Calls	✓		✓	✓			✓		✓				5
	Opcodes				✓		✓		✓		✓			4
	System Calls													0
	Network Data												✓	1
	Code		✓	✓				✓						3
	Others	✓			✓		✓	✓	✓			✓	✓	7
S.M	*K-means*	✓			✓						✓	✓	✓	5
	HAC				✓						✓			2
	EM													0
	Fuzzy				✓	✓								2
	Birch				✓						✓			2
	DBSCAN	✓	✓	✓		✓			✓		✓			6
	Proposed Model	✓						✓						2
	Phylogeny									✓				1
	Others	✓	✓		✓		✓			✓	✓			5
U.M	*Not mentioned*				✓									1
	SVM						✓	✓			✓		✓	4
	KNN							✓			✓		✓	3
	Decision Tree						✓				✓	✓		3
	RF	✓					✓	✓		✓	✓	✓	✓	7
	NB						✓	✓					✓	3
	Logistic						✓	✓						2
	Neuro Fuzzy													0
	Others	✓	✓	✓		✓	✓	✓	✓		✓		✓	9

As of 2018, we notice a consistent increase in the use of API calls for classifying Android malware. Studies indicate that trojans have larger sets of API calls compared to the other malware categories [221]. Also, according to malware reports, Trojans remain of the biggest threats to consumers in 2021, with them accounting for 90% of all pandemic-related malware [222].

4.2 Findings

As the quantity of new variants and families of Android malware has increased in recent years, research on clustering and phylogeny has been present in journals and anti-virus companies. As can be observed in Figure 3, more articles were found between the years 2016 and 2019.

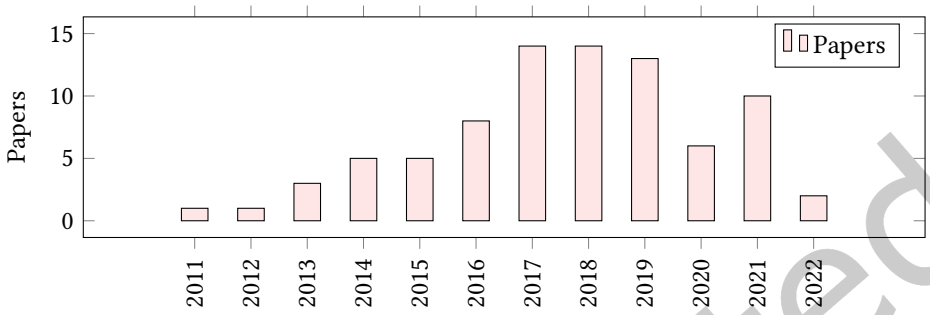

Fig. 3. Analysed papers per year

The results of the study analysis are summarized below:

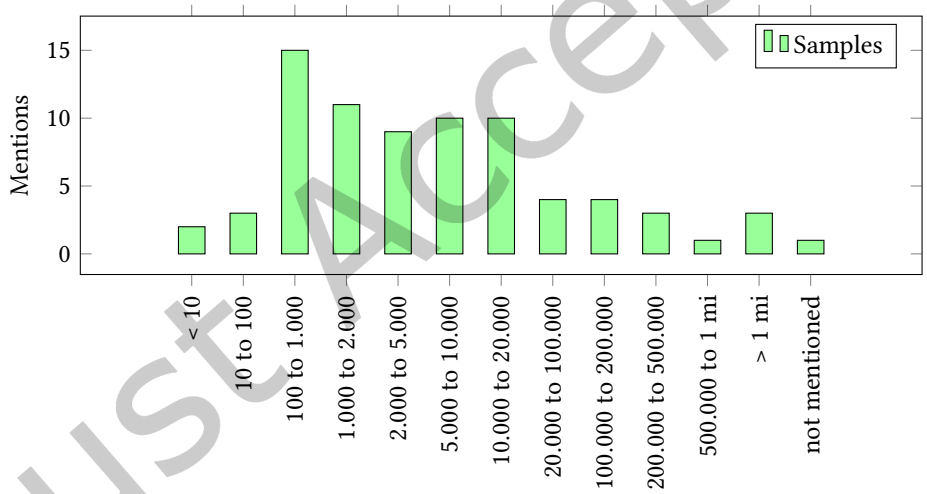

Fig. 4. Number of samples mentioned by the authors

Datasets: About datasets, there are several sources available free of charge for research, as available in Table 2. However, although they offered vectors ready for analysis, they rarely present detailed information about the sample set, such as categorizing the samples by type of malware and family. Also, the number of samples available is not yet representative and is not updated continuously [15]. Several studies used more than one dataset for experiments or created their sample set. The most cited datasets were Malgenome and Drebin.

Number of samples: Regarding the number of malware specimens used in experiments, [223] states that it is complex to determine the actual amount of samples used to train and test machine learning models. Thus, we emphasize that the quantities presented here were informed by the authors of the papers analyzed. By observing

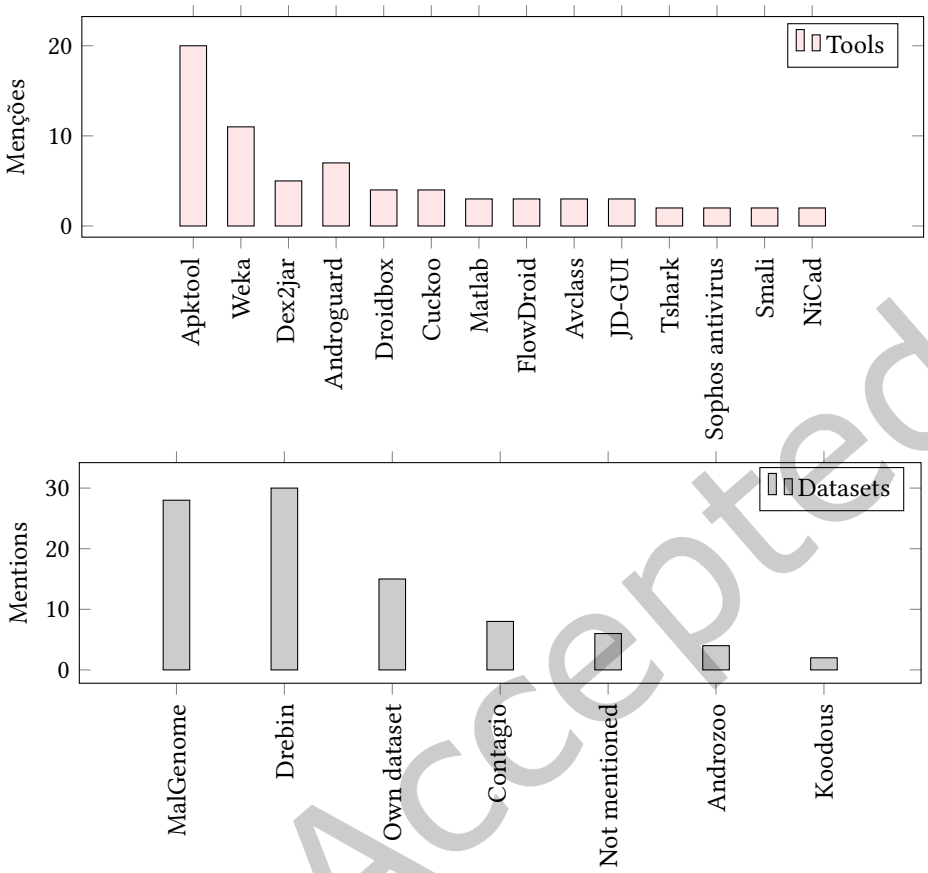

Fig. 5. The most used tools and malware datasets

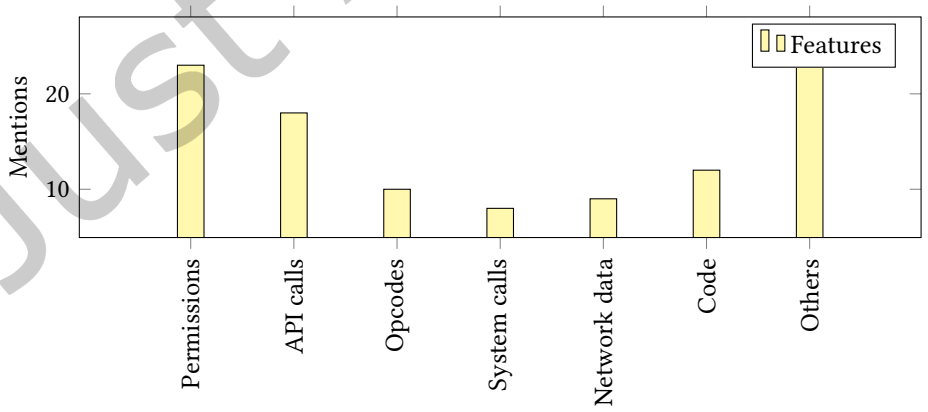

Fig. 6. The most used features

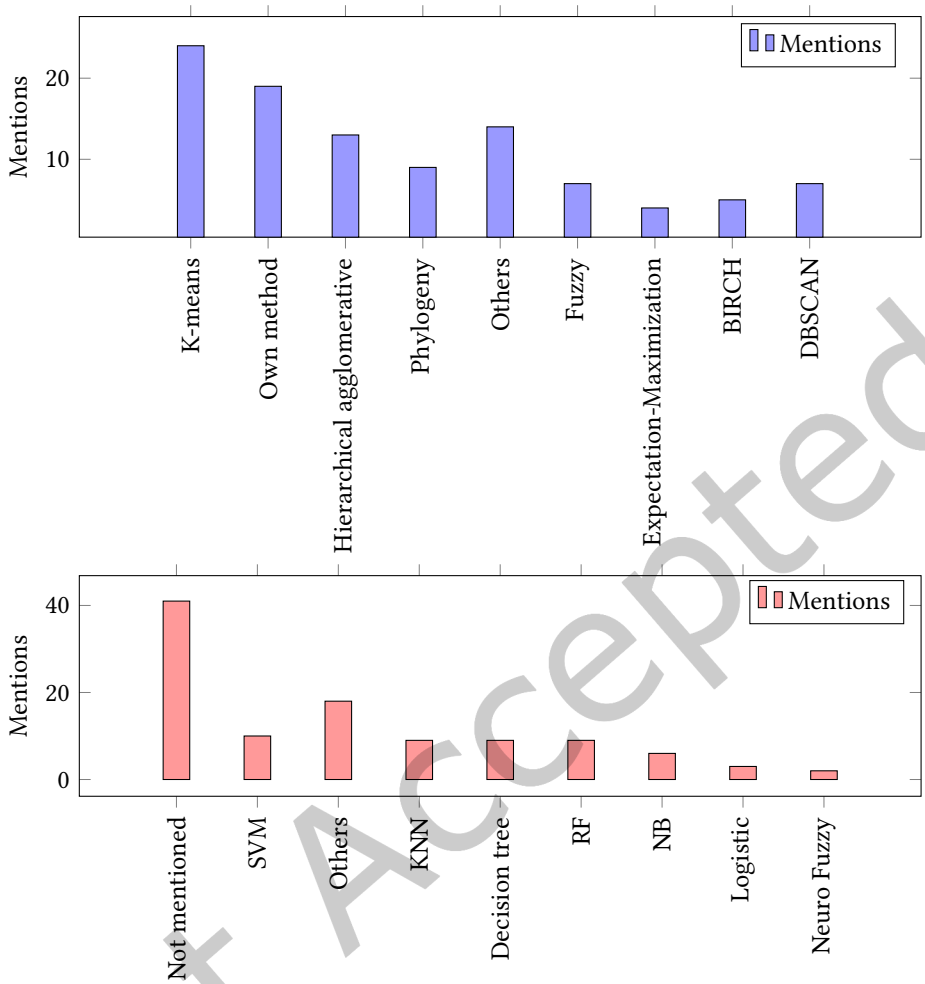

Fig. 7. The most unsupervised and supervised learning methods

Figure 4 it is possible to see that there are several ranges of sample amounts. For instance, 15 studies performed experiments between 100 and 1000 samples, 11 studies between 1000 and 2000, ten studies between 5.000 and 10.000, and ten studies between 10.000 and 20.000 copies. Only three studies analyzed more than 1 million specimens.

Type of analysis and use of tools: Concerning malware analysis, 49 studies performed static methods, 19 used dynamic analysis, and 8 applied hybrid analysis. In the case of phylogeny studies, only, 60 % used dynamic analysis. Less than 50 % of the analyzed Android malware phylogeny studies presented information and discussions about the relationships between the analyzed families. The tools most cited by the authors are in Figure 2, accentuating Apktool and Weka.

Features and representation: As can be seen in Figure 6, the most adopted features were permissions and API calls. In descending order of use, the methods include binary vectors, XML files, graphs, strings, n-grams, ARR files, Markov chains diagrams, grayscale images, and XES format.

Distance and similarity measures: When analyzing similar papers, we noticed that several measures of similarity and dissimilarity were applied. Besides, the number of studies that did not mention the used measure was also significant (15 studies out of 82). The most used measures were Euclidean distance and Jaccard Index.

Most applied unsupervised methods : As previously mentioned, clustering methods have been improved in recent years. However, more than ten analyzed studies employed the K-means algorithm as the primary clustering method. Also, some papers apply clustering with other Artificial Intelligence techniques, as can be seen in Figure 7. Among these techniques, the use of Random Forest and SVM stands out.

Clustering Results Validation: About the studies analyzed that cited clustering results validation indexes, 33% cited Accuracy, 17% cited F-Measure, and 16% cited True Positive and False Positive ratios. Purity, Calinski Harabaz score, ROC Curve, and other indexes validation were cited too.

There was a lack of normalization and standards in the presentation and validation of results, especially in phylogeny studies that did not use popular clustering methods.

4.3 Limitations and Challenges of Computational Phylogeny

Although phylogenetic methods that estimate maximum likelihood are popular in biology, in the malware domain, there are several limitations. For example, given a multiple alignment sequence and a specific phylogenetic tree, the likelihood for each character in the alignment must be calculated. With n sequences and states, the complexity The computational value of the calculation will be O (s n-2) [224]. This value is expensive when considering data such as DNA where there are only 4 states (A, G, T, C), but for malware where assembly instructions are states, the number of s is in the hundreds [225].

Another challenge presented in the research of Walenstein et al (2007) [226] is evolutionary changes in code: How do you know how members of a malware family are related and how different families are related through code sharing?

Other known obstacles in establishing the malware lineage are the lack of sufficient field data (e.g., malware samples and contextual information about its real-world impact), lack of metadata about the process of collecting existing data sets, and the difficulty of developing tools and methods for rigorous data analysis [13].

Moreover, additional challenges related to malware phylogeny that are still unresolved are splitting, merging, and alternative layouts, and viewing, exploring, and comparing trees [142].

Another limitation related to malware analysis and phylogeny is the compression of copies made of virtual machine memory during dynamic analysis procedures. James Fowler (2016) has been getting significant results with malware samples, intending to run the VirusShare project, which would require a total of 9 Petabytes of RAM on one virtual machine to be dynamically analyzed [227].

Dumitras et al (2011) pointed out two significant challenges in determining lineage and malware provenance. The first is in grouping samples into families. Current approaches use features such as signatures or call graphs to identify specimens from the same family. The inability of current tools to unzip obfuscated codes or generate dynamic call graphs makes it difficult to expose the connection between independent variables. The other challenge pointed out by the authors involves assembling the phylogenetic tree. The lack of relevant information from malware writers, such as code development processes and dissemination strategies, prevents analysts from establishing a fundamental truth for the validation of the results of the generated lineage [13].

Walenstein and Lakhotia (2012) address the foremost contemporary implications for building (evolutionary or not) relationships of malware. According to the authors, two significant current issues in capturing malware

developments are lack of conceptual clarity and lack of formalism appropriate and necessary to model relationships [228].

Other issues addressed, for example, are the descent of multiple lineages, the sharing of multi-variant functionality updates, and differences in code generation and compilation mechanisms.

Pfeffer et al (2012) indicated an alternative to the use of other forms of analysis to extract malware characteristics, such as genetics and linguistic ones. The continuous and recent research in the area, due to the constant increase in the number of malware and the evolution of the techniques used by its creators, researchers believe that there are several limitations and challenges to be researched. As demonstrated by some authors and starting from the initial proposal of Phylogeny suggests the application of the latest methods from other areas that have not been studied computationally in the scope of malware, such as Medicine and Biology, for the classification and lineage of malicious programs [229].

4.4 Stages of Malware Clustering: a proposal

Based on the analysis of the most prevalent clustering techniques in the malware literature, we propose the flow of tasks in Figure 8, to guide malicious code clustering.

- **Stage 0 - Establish Clustering Purpose**
  We perform analysis and clustering of mobile malware samples for different purposes. For example, hardware device manufacturers may be interested in investigating large amounts of mobile malicious code samples to identify potential threats to physical products. A network security analyst can analyze malware such as Android ransomware to respond to incidents. Other purposes include anti-malware solutions development, malicious software lineage research, and malware hunting.
  Different combinations of analysis type, clustering algorithms, and distance measures depend on the clustering purpose, dataset size, computational power, and available time.
- **Stage 1 - Samples Analysis and Attribute Extraction**
  According to the extent of the collection of elements, as a base of *.apks* files, the computational power, and the time available, the analyst experiments and defines the type of analysis. After defining and performing the analysis of the samples, we extract the attributes and define the representation types of the data.
- **Stage 2 - Data Normalization**
  With the set of features extracted, we perform the following tasks to ensure the quality of the clustering result.
  Data Normalization: The nature of the attributes (discrete, binary, continuous) and the variable scale (nominal, ordinal) affect the structure of the clusters. Thus, tasks such as standardization or data normalization can benefit the quality of the results. Standardization tends to minimize dispersion problems, normalize features according to a specific criterion, and eliminate noise and redundancy to improve the result accuracy. Some available methods include Z Score, Maximum Amplitude, and Range 0 to 1.
  Feature Selection: We have previously highlighted that feature selection can improve the quality of results using supervised and unsupervised methods. We also mentioned (Section 2.6) that not all feature selection methods apply to clustering. The choice of selection type, using filters, wrappers, or another approach, also depends on the purpose of the analysis, the computational power, and the time available for this pre-clustering step. Filters are not dependent on algorithm clustering, despite wrappers generally providing more satisfactory results. One of the recommended alternatives is to combine both techniques when possible. In Babaagba and Adesanya's (2019) malware analysis experiments [230], the accuracy increased from 54.3 to 74.4% using feature selections with the EM clustering algorithm.
  Outlier Identification: Outliers are objects considered dissimilar and inconsistent with the rest of the set. These anomalies can affect the result negatively or be just what we are sorting [231]. In malware analysis,

outliers can bring important information, such as identifying mislabeled samples. Generally, in clustering, we assume that most of the elements are in denser clusters. Thus, we investigate outliers data that do not belong to any group or small clusters. Strategies for dealing with anomalies include inconsistent elements exclusion and separate analysis of the anomalous set.

- **Stage 3 - Distance Measure Choice** Determining the best distance measure is not a trivial task, depending directly on the analyzed data. Several studies and surveys on distance and similarity measures are available in the literature. Thus, the question remains: how should I decide?
  The default option is usually the Euclidean distance, although this measurement is not ideal in numerous cases.
  Generally, when the analyst comprehends the sample data, he tries out several measurements and visually verifies which ones delivered greater consistency with the pattern of the dataset elements. For instance, operating with features in the form of strings, can we verify if with the Hamming distance, Jaccard Index, and Levenshtein (edit) Distance, using samples from the same family, the result indicated high similarity between the samples? And what happens to samples from very different families using these same metrics?
  Although there are dozens/hundreds of equations and similarity measures, the table presents recommendations based on the analyzed papers and the data type.

Type	**Measures and indexes**
Strings	Hamming Distance, Jaccard Index, Levenshtein (Edit) Distance, NCD, Cosine
Documents	Dice distance, Cosine, Jaccard Index
Binary Vector	Jaccard Index, Hamming Distance, Euclidean Distance
Graph	Maximal Common Subgraphs, Edit Distance
Images	Euclidean Distance, Manhattan, VCAD, distance of color histograms
Hidden Markov Models	Kullback-Leibler divergence

Table 13. Distance Measure and features types

- **Stage 4 - Clustering Method Selection and Results Validation**

As mentioned earlier, each type of clustering has advantages and disadvantages. The selection of the method depends on the nature of the data, the dimensionality of the features, the computational cost, and the time available for processing. For example, hierarchies are advantageous when you know that your data have hierarchical relationships. They do not deal well with outliers, not being suitable for extensive datasets. Also, it is often difficult to define the optimal number of clusters by visually analyzing the dendrogram. The Expectation Maximization method is not indicated to find significantly short clusters, such as when the dataset has families with only one or very few samples.

Partitions methods, such as K-means, are fast and highly scalable. However, they depend on the initial number of clusters. They are suitable for datasets with spherical shapes. As the number of features increases, the scalability of the algorithm decreases. It is also sensitive to outliers, so we indicate data normalization. Density-based Algorithms are great for data with high dimensionality and with different forms of sets. Also, they are a great choice for noisy datasets, because this type handles outliers well. However, they often suffer from overfitting.

In Table 14 we present examples of this clustering proposal.

Purpose	Analysis Type	Distance Metric	Clustering method
Investigate siblings families	Hybrid - Java API Call	Jaccard index	UPGMA
Detect malware	Dynamic - CFG	Edit distance	K-means and K-medoid
Analysis of families malware reports	Static - TFIDF	NCD	DBSCAN
Verify dangerous frequent permissions	Static - Permissions	Euclidean distance	Self-Organizing Map
Identify potential banking malware behaviors	Dynamic - network data	the Normalized Euclidean distance	Fuzzy C-Means

Table 14. Examples of clustering malware purposes.

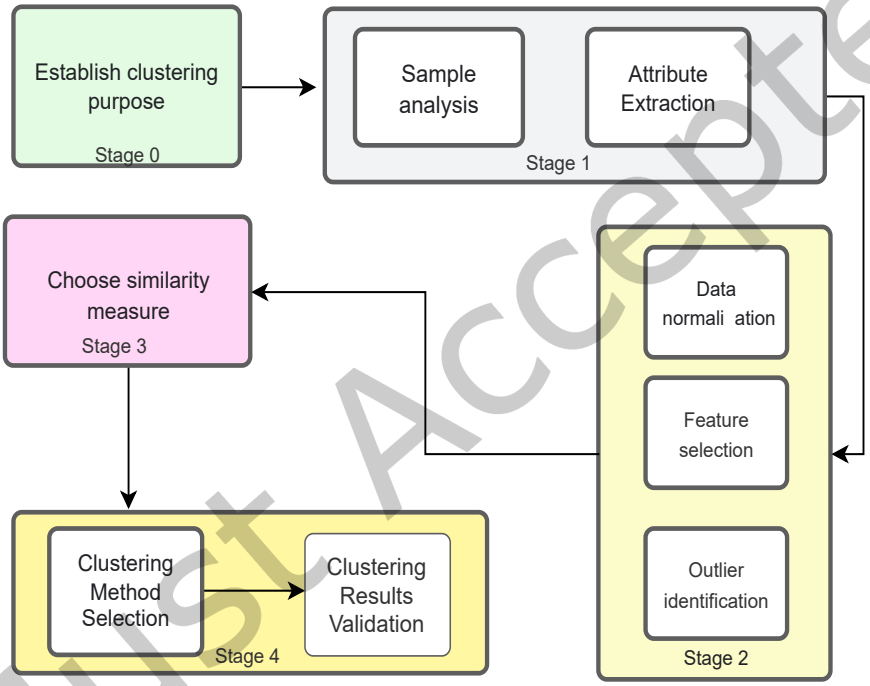

Fig. 8. Basic Approach to Clustering Malware.

5 FINAL CONSIDERATIONS

In addition to the current cybersecurity challenges, new technologies bring different scenarios, including solutions to existing situations and possible unexplored problems. One example is the technology of Digital Twins, which we can apply to simulations and contingency plans for several types of cyberattacks, which may involve zero-day malware. In this way, there are still several possibilities for acquiring knowledge about attacks, the mode of operating malicious codes, and other applications. Therefore, the clustering algorithms for malware analysis are still necessary and efficient for several purposes, especially when there are so much data to investigate.

The continued application of clustering methods in distinguishing between benign and malicious apps reiterates that satisfactory results are still possible. However, our investigations reiterate that for clustering to present better results, procedures include data normalization, feature selection through filters and wrappers, and the choice of appropriate similarity measures and clustering algorithms. Furthermore, we present a short tutorial for malware clustering with four distinct steps. Our findings show that most of the challenges and limitations of phylogeny are related to the lack of labeled datasets with samples separated into families, problems in labeling and the determination of the dates of variants, and the insufficiency of standardization of results quality analysis. As indications for future research, we suggested evaluating the use of Big Data approaches, and the usage of more recent clustering and machine learning algorithms.

REFERENCES

[1] A. Cani, M. Gaudesi, E. Sanchez, G. Squillero, and A. Tonda. 2014. Towards automated malware creation: code generation and code integration. In *Proceedings of the 23rd International Conference on World Wide Web*. ACM, 157–160.

[2] S. K. Sahay, A. Sharma, and H. Rathore. 2020. Evolution of Malware and Its Detection Techniques. In *In Information and Communication Technology for Sustainable Development*. Springer, 139–150.

[3] Y. Ye, T. Li, K. Huang, Q. Jiang, and Y. Chen. 2010. Hierarchical associative classifier (HAC) for malware detection from the large and imbalanced gray list. *Journal of Intelligent Information Systems* (2010).

[4] B. Miller, A. Kantchelian, M. Tschantz, S. Afroz, R. Bachwani, R. Faizullabhoy, L. Huang, V. Shankar, T. Wu, G. Yiu, A. D. Joseph, and J. D. Tygar. 2016. Reviewer Integration and Performance Measurement for Malware Detection. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 122–141.

[5] A. Martín, H. D. Menéndez, and D. Camacho. 2017. MOCDroid: multi-objective evolutionary classifier for Android malware detection. *Soft Computing, 21(24), 7405-7415* (2017).

[6] S. Chakraborty, J.W. Stokes, L. Xiao, D. Zhoud, M. Marinescu, and A. Thomas. 2017. Hierarchical learning for automated malware classification. *MILCOM 2017-2017 IEEE Military Communications* (2017).

[7] D. M. Caldas. 2016. *Análise e extração de características estruturais e comportamentais para perfis de malware*. Master's thesis. Mestrado em Engenharia Elétrica - Universidade de Brasília, Brasília - DF.

[8] V. Bontchev. 2004. Anti-virus spamming and the virus-naming mess: Part 2. *Virus Bulletin* (2004).

[9] M. Hurier, G Suarez-Tangil, S. K. Dash, Bissyandé, Traon T. F., J. Y. L., Klein, and L. Cavallaro. 2017. Euphony: Harmonious unification of cacophonous anti-virus vendor labels for Android malware. In *Proceedings of the 14th International Conference on Mining Software Repositories*. IEEE.

[10] Y. Zhang, Y. Sui, S. Pan, Z. Zheng, B. Ning, I. Tsang, and W. Zhou. 2019. Familial Clustering For Weakly-labeled Android Malware Using Hybrid Representation Learning. In *IEEE Transactions on Information Forensics and Security*. IEEE.

[11] A. Kantchelian, M.C. Tschantz, S. Afroz, B. Miller, V. Shankar, R. Bachwani, and J.D. Tygar. 2015. Better malware ground truth: Techniques for weighting anti-virus vendor labels. In *Proceedings of the 8th ACM Workshop on Artificial Intelligence and Security*. ACM, 45–56.

[12] M. Sebastián, R. Rivera, P. Kotzias, and J. Caballero. 2016. Towards automated malware creation: code generation and code integration. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 230–253.

[13] T. Dumitras and I. Neamtiu. 2011. A Story of Provenance and Lineage for Malware.. In *CSET*. ACM.

[14] J. D. Seideman, B. Khan, and A. C. Vargas. 2014. Identifying malware genera using the Jensen-Shannon distance between system call traces. *Malicious and Unwanted Software: The Americas (MALWARE), 2014 9th International Conference* (2014).

[15] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro. 2017. The evolution of android malware and android analysis techniques. In *ACM Computing Survey*. ACM.

[16] T. Mohini, S. A. Kumar, and G. Nitesh. 2013. Review on Android and smartphone security. *Research Journal of Computer and Information Technology Science* (2013).

[17] E. Gandotra, D. Bansal, and S. Sofat. 2014. Malware analysis and classification: A survey. *Journal of Information Security* (2014).

[18] A. Feizollah, N. B. Anuar, R. Salleh, and A. W. A Wahab. 2015. A review on feature selection in mobile malware detection. *Digital investigation* (2015).

[19] S. Arshad, M. A. Shah, A. Khan, and M. Ahmed. 2016. Android malware detection & protection: a survey. *International Journal of Advanced Computer Science and Applications* (2016).

[20] R. Riasat, M. Sakeena, W. A. N. G. Chong, A. H. Sadiq, and Y. J Wang. 2016. A Survey on Android Malware Detection Techniques. In *DEStech Transactions on Computer Science and Engineering*. wcne.

[21] Balaji Baskaran and Anca L. Ralescu. 2016. A Study of Android Malware Detection Techniques and Machine Learning. In *MAICS*.

[22] N. Yadav, A. Sharma, and A. Doegar. 2016. A Survey on Android Malware Detection. *International Journal of New Technology and Research* (2016).
[23] A. Malhotra and K. Bajaj. 2016. A survey on various malware detection techniques on mobile platform. *Int J Comput Appl* (2016).
[24] R. Zachariah, K. Akash, M. S. Yousef, and A. M Chacko. 2017. Android malware detection a survey.. In *2017 IEEE international conference on circuits and systems (ICCS)*. IEEE, 238–244.
[25] P. Yan and Z. Yan. 2018. A survey on dynamic mobile malware detection. *Software Quality Journal* (2018).
[26] B. Yu, Y. Fang, Q. Yang, Y. Tang, and L. Liu. 2018. A survey of malware behavior description and analysis. *Frontiers of Information Technology & Electronic Engineering* (2018).
[27] M. Odusami, O. Abayomi-Alli, S. Misra, O. Shobayo, R. Damasevicius, and R. Maskeliunas. 2018. Android Malware Detection: A Survey. In *International Conference on Applied Informatics*. Springer, 255–266.
[28] Y. S. I. Hamed, S. N. A. AbdulKader, and M. S. M. Mostafa. 2019. Mobile Malware Detection: A Survey. *International Journal of Computer Science and Information Security (IJCSIS)* (2019).
[29] Fahad Alswaina and Khaled Elleithy. 2020. Android malware family classification and analysis: Current status and future directions. *Electronics* 9, 6 (2020), 942.
[30] Rajesh Kumars, Mamoun Alazab, and WenYong Wang. 2021. A survey of intelligent techniques for Android malware detection. *Malware Analysis Using Artificial Intelligence and Deep Learning* (2021), 121–162.
[31] Li Meijin, Fang Zhiyang, Wang Junfeng, Cheng Luyu, Zeng Qi, Yang Tao, Wu Yinwei, and Geng Jiaxuan. 2022. A Systematic Overview of Android Malware Detection. *Applied Artificial Intelligence* 36, 1 (2022), 2007327.
[32] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX Security Symposium (USENIX Security 19)*. 729–746.
[33] A. Qamar, A. Karim, and V. Chang. 2019. Mobile malware attacks: Review, taxonomy & future directions. *Future Generation Computer Systems* (2019).
[34] A. Feizollah, N. B. Anuar, R. Salleh, F. Amalina, and S. Shamshirband. 2013. A study of machine learning classifiers for anomaly-based mobile botnet detection. *Malaysian Journal of Computer Science* (2013).
[35] D. Dagon, T. Martin, and T. Starner. 2004. Mobile phones as computing devices: The viruses are coming! *IEEE Pervasive Computing* (2004).
[36] T. Chakraborty, F. Pierazzi, and V. S. Subrahmanian. 2017. Ec2: Ensemble clustering and classification for predicting android malware families. *IEEE Transactions on Dependable and Secure Computing*. (2017).
[37] A. Apvrille and T. Strazzere. 2012. Reducing the window of opportunity for Android malware Gotta catch'em all. *Journal in Computer Virology, 8(1-2), 61-71* (2012).
[38] Azqa Nadeem. 2018. *Clustering Malware's Network Behavior using Simple Sequential Features*. Master's thesis. University of Technology, Faculty of Electrical Engineering, Mathematics and Computer Science.
[39] Sapna Malik. 2017. *Malware Detection in Android Phones*. Anchor Academic Publishing.
[40] Y. Zhou and X. Jiang. 2012. Dissecting android malware: Characterization and evolution. In *Proceedings of the 2012 IEEE Symposium on Security and Privacy*. IEEE, 95 – 109.
[41] F. Wei, Y. Li, S. Roy, X. Ou, and W. Zhou. 2017. Deep ground truth analysis of current android malware. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*. Springer, 252–276.
[42] Abdelmonim Naway and Yuancheng Li. 2019. Android Malware Detection Using Autoencoder. *CoRR* abs/1901.07315 (2019). arXiv:1901.07315 http://arxiv.org/abs/1901.07315
[43] Suzanna Schmeelk, Junfeng Yang, and Alfred Aho. 2015. Android malware static analysis techniques. In *Proceedings of the 10th annual cyber and information security research conference*. 1–8.
[44] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb. 2019. A Survey on Android Malware Detection Techniques Using Machine Learning Algorithms. In *2019 Sixth International Conference on Software Defined Systems (SDS)*. IEEE, 110–117.
[45] Sonal Mohite and PR Sonar. 2014. A survey on mobile malware: A war without end. *International Journal of Computer Science and Business Informatics* 9, 1 (2014), 23–35.
[46] K. Bakour, H.M. Ünver, and R. Ghanem. 2019. The Android malware detection systems between hope and reality. *SN Applied Sciences* (2019).
[47] A. Firdaus, N.B. Anuar, A. Karim, and M. F. Ab Razak. 2018. Discovering optimal features using static analysis and a genetic search based method for Android malware detectio. *Frontiers of Information Technology & Electronic Engineering* (2018).
[48] A. Feizollah, N. B. Anuar, R. Salleh, and F Amalina. 2014. Comparative study of k-means and mini batch k-means clustering algorithms in android malware detection using network traffic analysis. In *2014 International Symposium on Biometrics and Security Technologies (ISBAST)*. IEEE, 193–197.
[49] G. Shrivastava and P. Kumar. 2019. Intent and permission modeling for privacy leakage detection in android. *Energy Systems* (2019).
[50] W. Wang, M. Zhao, Z. Gao, G. Xu, H. Xian, Y. Li, and X. Zhang. 2019. Constructing features for detecting android malicious applications: issues, taxonomy and directions. *IEEE Access* (2019).

[51] Richard Killam, Paul Cook, and Natalia Stakhanova. 2016. Android malware classification through analysis of string literals. In *Workshop Programme*. 27.
[52] A. D. Schmidt, R. Bye, H. G. Schmidt, J. Clausen, O. Kiraz, K. A. Yuksel, S. A Camtepe, and S. Albayrak. 2009. Static Analysis of Executables for Collaborative Malware Detection on Android. *2009 IEEE International Conference on Communications* (2009).
[53] S. Alam, I. Traore, and I. Sogukpinar. 2015. Annotated control flow graph for metamorphic malware detection. *The Computer Journal,* (2015).
[54] A. Skovoroda and D. Gamayunov. 2015. Review of the mobile malware detection approaches. *2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing* (2015).
[55] R. Perdisci, W.Lee, and N. Feamster. 2010. Behavioral Clustering of HTTP-Based Malware and Signature Generation Using Malicious Network Traces. In *NSDI,Vol. 10*. 14.
[56] M. E. Karim, A. Walenstein, A. Lakhotia, and L. Parida. 2005. Malware phylogeny generation using permutations of code. *Journal in Computer Virology* (2005).
[57] Mohammad Imran, Muhammad Tanvir Afzal, and Muhammad Abdul Qadir. 2017. A comparison of feature extraction techniques for malware analysis. *Turkish Journal of Electrical Engineering & Computer Sciences* 25, 2 (2017), 1173–1183.
[58] Emrah Hancer, Bing Xue, and Mengjie Zhang. 2020. A survey on feature selection approaches for clustering. *Artificial Intelligence Review* 53, 6 (2020), 4519–4545.
[59] Avrim L Blum and Pat Langley. 1997. Selection of relevant features and examples in machine learning. *Artificial intelligence* 97, 1-2 (1997), 245–271.
[60] Tu Minh Phuong, Zhen Lin, and Russ B Altman. 2005. Choosing SNPs using feature selection. In *2005 IEEE Computational Systems Bioinformatics Conference (CSB'05)*. IEEE, 301–309.
[61] Huan Liu and Hiroshi Motoda. 2007. *Computational methods of feature selection*. CRC Press.
[62] Suhang Wang, Jiliang Tang, and Huan Liu. 2015. Embedded unsupervised feature selection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.
[63] Z. Huang. 1997. A Fast Clustering Algorithm to Cluster Very Large Categorical Data Sets in Data Mining. *SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery (SIGMOD-DMKD'97)* (1997).
[64] J. Han, J. Pei, and M. Kamber. 2011. *Data mining: concepts and techniques*. Elsevier.
[65] Jean Metz. 2006. *Análise e extração de características estruturais e comportamentais para perfis de malware*. Master's thesis. Mestra em Ciências de Computação e Matemática Computacional - USP., São Carlos - SP.
[66] B. Sanz, I. Santos, X. Ugarte-Pedrero, C. Laorden, J. Nieves, and P.G Bringas. 2014. Anomaly detection using string analysis for android malware detection. In *International Joint Conference SOCO'13-CISIS'13-ICEUTE'13*. Springer, 469–478.
[67] S. Pandit and S. Gupta. 2011. A comparative study on distance measuring approaches for clustering. *International Journal of Research in Computer Science, 2(1)* (2011).
[68] M. Cebrián, M. Alfonseca, and A.Ortega. 2007. The normalized compression distance is resistant to noise. *IEEE Transactions on Information Theory* (2007).
[69] M. A. Jaro. 1995. Probabilistic linkage of large public health data files. *Statistics in medicine* (1995).
[70] W. E. Winkler. 1999. The state of record linkage and current research problems. *Statistics of Income Division - Internal Revenue Service Publication* (1999).
[71] V. Katos. 2007. Network intrusion detection: Evaluating cluster, discriminant, and logit analysis. *Information Sciences 177(15)* (2007).
[72] H. Pareek, P. Eswari, N. S. C. Babu, and C. Bangalore. 2013. Entropy and n-gram analysis of malicious PDF documents. *International Journal of Engineering, 2(2)* (2013).
[73] A. A. E. Elhadi, M. A. Maarof, and B. I. Barry. 2013. Improving the detection of malware behaviour using simplified data dependent api call graph. *International Journal of Security and Its Applications* (2013).
[74] R.S. Borbely. 2016. On normalized compression distance and large malware. *Journal of Computer Virology and Hacking Techniques* (2016).
[75] O. R. Zaïane, A. Foss, C. H. Lee, and W. Wang. 2002. On data clustering analysis: Scalability, constraints, and validation. *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2002).
[76] J Swarndeep Saket and Sharnil Pandya. 2016. An overview of partitioning algorithms in clustering techniques. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)* 5, 6 (2016), 1943–1946.
[77] Lior Rokach and Oded Maimon. 2005. Clustering methods. In *Data mining and knowledge discovery handbook*. Springer, 321–352.
[78] Sungjune Park, Nallan C Suresh, and Bong-Keun Jeong. 2008. Sequence-based clustering for Web usage mining: A new experimental framework and ANN-enhanced K-means algorithm. *Data & Knowledge Engineering* 65, 3 (2008), 512–543.
[79] Osama Abu Abbas. 2008. Comparisons between data clustering algorithms. *International Arab Journal of Information Technology (IAJIT)* 5, 3 (2008).
[80] M. Aresu, D. Ariu, M. Ahmadi, D. Maiorca, and Giacinto. 2015. Clustering android malware families by http traffic. In *10th International Conference on Malicious and Unwanted Software (MALWARE)*. IEEE, 128–135.

[81] L. Weichselbaum, M Neugschwandtner, M. Lindorfer, Y Fratantonio, V. van der Veen, and C. Platzer. 2014. *A practical test for univariate and multivariate normality.* Technical Report. Vienna University of Technology, Tech. Rep.
[82] D. Korczynski. 2015. *ClusTheDroid: clustering android malware.* Technical Report. Royal Holloway, Univ. London, London, UK, Tech. Rep.
[83] Stan Salvador and Philip Chan. 2004. Determining the number of clusters/segments in hierarchical clustering/segmentation algorithms. In *16th IEEE international conference on tools with artificial intelligence*. IEEE, 576–584.
[84] H. Rathore, S. K. Sahay, P. Chaturvedi, and M Sewak. 2018. Android Malicious Application Classification Using Clustering. In *International Conference on Intelligent Systems Design and Applications*. Springer, 659–667.
[85] Chao Deng, Jinwei Song, Ruizhi Sun, Saihua Cai, and Yinxue Shi. 2018. GRIDEN: An effective grid-based and density-based spatial clustering algorithm to support parallel computing. *Pattern Recognition Letters* 109 (2018), 81–88.
[86] D. Xu and Y. Tian. 2015. A comprehensive survey of clustering algorithms. In *Annals of Data Science*. Springer, 165–193.
[87] S. Garcia and M. Noe. 2011. *Fractal dimension for clustering and unsupervised and supervised feature selection.* Cardiff University.
[88] J. Sarraille and P. DiFalco. Fd3. ([n. d.]). http://tori.postech.ac.kr/softwares
[89] Madjid Khalilian and Norwati Mustapha. 2010. Data Stream Clustering: Challenges and Issues. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, Vol. 1.
[90] C. Fraley and A. E. Raftery. 2002. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American statistical Association* (2002).
[91] Brad Boehmke and Brandon Greenwell. 2019. *Hands-on machine learning with R.* Chapman and Hall/CRC.
[92] A. El Attar, R. Khatoun, and M. Lemercier. 2014. A Gaussian mixture model for dynamic detection of abnormal behavior in smartphone applications. In *2014 global information infrastructure and networking symposium (GIIS)*. IEEE, 1–6.
[93] R. Xu and I. I. Donald Wunsch. 2005. Survey of Clustering Algorithms. In *IEEE TRANSACTIONS ON NEURAL NETWORKS*. IEEE.
[94] Zhao Kang, Liangjian Wen, Wenyu Chen, and Zenglin Xu. 2019. Low-rank kernel learning for graph-based clustering. *Knowledge-Based Systems* 163 (2019), 510–517.
[95] X. Xu, M. Ester, H. P. Kriegel, and J. Sander. 1998. A distribution-based clustering algorithm for mining in large spatial databases. In *Proceedings 14th International Conference on Data Engineering*. IEEE.
[96] G. A. Carpenter, S. Grossberg, and D. B. Rosen. 1991. Fuzzy ART: Fast stable learning and categorization of analog patterns by an adaptive resonance system. *Neural networks* (1991).
[97] C. Haykin. 2007. *Redes neurais: princípios e prática.* Bookman Editora.
[98] K-L Du. 2010. Clustering: A neural network approach. *Neural networks* 23, 1 (2010), 89–107.
[99] D. Barrera, H. G. Kayacik, P. C. Van Oorschot, and A. Somayaji. 2010. A methodology for empirical analysis of permission-based security models and its application to android. In *Proceedings of the 17th ACM conference on Computer and communications security*. ACM, 73–84.
[100] D. Corne, J. Handl, and J Knowles. 2010. *Encyclopedia of Machine Learning*. Springer US".
[101] K. Krishna and N. Murty. 1999. Genetic K-means algorithms. In *Transactions on Systems Man And Cybernetics-Part B: Cybernetics*. IEEE.
[102] L. O. Hall, I. B. Ozyurt, and J. C. Bezdek. 1999. Clustering with a genetically optimized approach. In *IEEE Transactions on Evolutionary computation*. IEEE.
[103] P. Nerurkar, A. Shirke M. Chandane, and S. Bhirud. 2018. A novel heuristic for evolutionary clustering.. In *Procedia Computer Science*. ScienceDirect, 780–789.
[104] ZM Nopiah, MI Khairir, Shahrum Abdullah, MN Baharin, and A Arifin. 2010. Time complexity analysis of the genetic algorithm clustering method. In *Proceedings of the 9th WSEAS international conference on signal processing, robotics and automation, ISPRA*, Vol. 10. 171–176.
[105] D. J. Bora, D. Gupta, and A. Kumar. 2014. A comparative study between fuzzy clustering algorithm and hard clustering algorithm. *International Journal of Computer Trends and Technology (IJCTT)* (2014).
[106] Salar Askari. 2021. Fuzzy C-Means clustering algorithm for data with unequal cluster sizes and contaminated with noise and outliers: Review and development. *Expert Systems with Applications* 165 (2021), 113856.
[107] Anjana Gosain and Sonika Dahiya. 2016. Performance analysis of various fuzzy clustering algorithms: a review. *Procedia Computer Science* 79 (2016), 100–111.
[108] J. Shawe-Taylor and N. Cristianini. 2004. *Kernel methods for pattern analysis.* Cambridge University Press.
[109] Radha Chitta, Rong Jin, Timothy C Havens, and Anil K Jain. 2011. Approximate kernel k-means: Solution to large scale kernel clustering. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. 895–903.
[110] R. Linden. 2009. Técnicas de agrupamento. *Revista de Sistemas de Informação da FSMA* (2009).
[111] O. Yim and K. T. Ramdeen. 2015. Hierarchical cluster analysis: comparison of three linkage measures and application to psychological data. *The quantitative methods for psychology, 11(1)* (2015).
[112] R. Sánchez, A. Herrero, and E. Corchado. 2017. Clustering extension of MOVICAB-IDS to distinguish intrusions in flow-based data. *Logic Journal of the IGPL, 25(1)* (2017).

[113] A. Fernández and S. Gómez. 2008. Solving Non-Uniqueness in Agglomerative Hierarchical Clustering Using Multidendrograms. *Journal of Classification, 25(1)* (2008).
[114] J. Bien and R. Tibshirani. 2011. Hierarchical clustering with prototypes via minimax linkages. *J. Amer. Statist. Assoc.* (2011).
[115] Q. He. 1999. A review of clustering algorithms as applied in IR. *Graduate School of Library and İnformation Science University of İllinois at Urbana-Compaign* (1999).
[116] Y. Liu, Z. Li, H. Xiong, X X. Gao, and J. Wu. 2010. Understanding of internal clustering validation measures.. In *2010 IEEE International Conference on Data Mining*. IEEE, 911–916).
[117] U. Maulik and S. Bandyopadhyay. 2002. Performance evaluation of some clustering algorithms and validity indices. *IEEE PAMI* (2002).
[118] E. Rendón, I. Abundez, A. Arizmendi, and E. M. Quiroz. 2011. Internal versus external cluster validation indexes. *International Journal of computers and communications, 5(1)* (2011).
[119] R. A. Ognev, E. V. Zhukovskii, and D. P. Zegzhda. 2019. Clustering of Malicious Executable Files Based on the Sequence Analysis of System Calls. *Automatic Control and Computer Sciences, 53(8)* (2019).
[120] G. Pitolli, L. Aniello, G. Laurenza, L. Querzoni, and R. Baldoni. 2017. Malware family identification with BIRCH clustering. In *2017 International Carnahan Conference on Security Technology (ICCST)*. IEEE, 1–6.
[121] B. K. Mishra, N. R. Nayak, and A. K. Rath. 2016. Assessment of basic clustering techniques using teaching-learning-based optimisation. *International Journal of Knowledge Engineering and Soft Data Paradigms, 5(2)* (2016).
[122] J.C. Bezdek and N. R. Pal. 1998. Some new indexes of cluster validity.. In *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*. IEEE, 301–315).
[123] N. Bolshakova and F. Azuaje. 2003. Cluster validation techniques for genome expression data. *Signal processing, 83(4)* (2003).
[124] S. Luan, X. Kong, B. Wang, Y. Guo, and X. You. 2012. Silhouette coefficient based approach on cell-phone classification for unknown source images. In *2012 IEEE International Conference on Communications (ICC)*. IEEE.
[125] C. Giannella and E.Bloedorn. 2015. Spectral malware behavior clustering. In *In 2015 IEEE International Conference on Intelligence and Security Informatics (ISI)*. IEEE.
[126] Yaocheng Zhang, Wei Ren, Tianqing Zhu, and Yi Ren. 2019. SaaS: A situational awareness and analysis system for massive android malware detection. *Future Generation Computer Systems* 95 (2019), 548–559.
[127] S. C. Sripada and M. S. Rao. 2011. Comparison of purity and entropy of k-means clustering and fuzzy c means clustering. *Indian journal of computer science and engineering, 2(3)* (2011).
[128] M. Bat-Erdene, H. Park, H. Li, H. Lee, and M. S Choi. 2017. Entropy analysis to classify unknown packing algorithms for malware detection. *International Journal of Information Security* (2017).
[129] J. Stiborek, T. Pevný, and M. Rehák. 2018. Probabilistic analysis of dynamic malware traces. *Computers & Security* (2018).
[130] A. A. A. Samra, K. Yim, and O. A. Ghanem. 2013. Analysis of clustering technique in android malware detection. In *2013 Seventh International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing*. IEEE, 729–733.
[131] E. Marin, A. Diab, and P. Shakarian. 2016. Product offerings in malicious hacker markets. In *2016 IEEE conference on intelligence and security informatics (ISI)*. IEEE, 187–189.
[132] J. Yearwood, D. Webb, L.Ma, P. Vamplew, B. Ofoghi, and A. Kelarev. 2009. Applying clustering and ensemble clustering approaches to phishing profiling. In *Proceedings of the Eighth Australasian Data Mining Conference-Volume 101*. ACM, 25–34.
[133] L. Onwuzurike, E. Mariconti, P. Andriotis, E. D. Cristofaro, G. Ross, and G Stringhini. 2019. MaMaDroid: Detecting android malware by building markov chains of behavioral models. In *ACM Transactions on Privacy and Security (TOPS)*. ACM, 14.
[134] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu. 2012. Droidmat: Android malware detection through manifest and api calls tracing. In *2012 Seventh Asia Joint Conference on Information Security*. IEEE, 62–69.
[135] W. Hennig. 1966. *Phylogenetic Systematics*. University of Illinois Press.
[136] J. Liu, Y. Wang, P. Dai XIE, and Y. J. Wang. 2017. Inferring phylogenetic network of malware families based on splits graph. *IEICE TRANSACTIONS on Information and Systems* (2017).
[137] W. M. Khoo and P.Lió. 2011. Unity in diversity: Phylogenetic-inspired techniques for reverse engineering and detection of malware familie. *2011 First SysSec Workshop* (2011).
[138] M. Hashimoto and A. Mori. 2008. Diff/TS: A tool for fine-grained structural change analysis.. In *Virus Bulletin Conference*. IEEE, 279–288.
[139] Y. Ye, T. Li, K. Huang, Q. Jiang, and Y. Chen. 1983. Parsimony in systematics: biological and statistical issues. *Annual review of ecology and systematics* (1983).
[140] J. Kim and T. Warnow. 1999. Tutorial on phylogenetic tree estimation. *Intelligent Systems for Molecular Biology* (1999).
[141] B. Haubold and T. Wiehe. 2006. Introduction to Computational Biology. *Basel, Switzerland: Birkhauser* (2006).
[142] A. Walenstein, M. Hayes, and A. Lakhotia. 2007. Phylogenetic comparisons of malware. In *Virus Bulletin Conference*. 41.
[143] J. Liu, Y. Wang, and Y. Wang. 2016. Inferring phylogenetic networks of malware families from api sequences.. In *2016 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*. IEEE.

[144] Ali Feizollah, Nor Badrul Anuar, and Rosli Salleh. 2018. Evaluation of network traffic analysis using fuzzy C-means clustering algorithm in mobile malware detection. *Advanced Science Letters* 24, 2 (2018), 929–932.
[145] Z. Aung and W. Zaw. 2013. Permission-based android malware detection. *International Journal of Scientific & Technology Research* (2013).
[146] J. Crussell, C. Gibler, and H Chen. 2013. Andarwin: Scalable detection of semantically similar android applications. In *European Symposium on Research in Computer Security*. Springer, 182–199.
[147] Y. Shao, X. Luo, C. Qian, P. Zhu, and L. Zhang. 2014. Towards a scalable resource-driven approach for detecting repackaged android applications.. In *Proceedings of the 30th Annual Computer Security Applications Conference*. ACM, 56–65.
[148] Guillermo Suarez-Tangil, Juan E Tapiador, Pedro Peris-Lopez, and Jorge Blasco. 2014. Dendroid: A text mining approach to analyzing and classifying code structures in android malware families. *Expert Systems with Applications* 41, 4 (2014), 1104–1117.
[149] L. Deshotels, V. Notani, and A. Lakhotia. 2014. Droidlegacy: Automated familial classification of android malware. In *Proceedings of ACM SIGPLAN on program protection and reverse engineering workshop 2014*. ACM, 3.
[150] P.Faruki, V.Laxmi, A.Bharmal, M.S.Gaur, and V.Ganmoor. 2015. AndroSimilar: Robust signature for detecting variants of Android malware. *Journal of Information Security and Applications* (2015).
[151] S. B. Almin and M. Chatterjee. 2015. A novel approach to detect android malware. *JProcedia Computer Science v45* (2015).
[152] J. Chen, M. H. Alalfi, T. R. Dean, and Y. Zou. 2015. Detecting android malware using clone detection. *Journal of Computer Science and Technology* (2015).
[153] Benjamin Cruz, D Gupta, A Kapoor, L Haifei, D McLean, F Moreno, et al. 2021. Pandemic Fears and Mobile Banking Are Popular Malware Targets. *McAfee Inc., Santa Clara, CA. Available: https://www.mcafee.com/content/dam/global/infographics/McAfeeMobileThreatReport2021.pdf* (2021).
[154] D. A. K Dutta. 2016. Detection of Malware and Malicious Executables Using E-Birch Algorithm. *Journal of Advanced Computer Science and Applications* (2016).
[155] G. Canfora, F. Mercaldo, A. Pirozzi, and C. A Visaggio. 2016. How I Met Your Mother? – An Empirical Study about Android Malware Phylogenesis. In *In Proceedings of the 13th International Joint Conference on e-Business and Telecommunications*. SCITEPRESS-Science and Technology Publications, Lda, 310–317.
[156] S. W. Hsiao, Y. S. Sun, and M. CL Chen. 2016. Behavior grouping of Android malware family. In *2016 IEEE International Conference on Communications (ICC)*. IEEE, 1–6.
[157] J. W. Jang, J. Yun, A. Mohaisen, J. Woo, and H. K. Kim. 2016. Detecting and classifying method based on similarity matching of Android malware behavior with profile. *SpringerPlus* (2016).
[158] Riccardo Sartea, Mila Dalla Preda, Alessandro Farinelli, Roberto Giacobazzi, and Isabella Mastroeni. 2016. Active Android malware analysis: an approach based on stochastic games. In *Proceedings of the 6th Workshop on Software Security, Protection, and Reverse Engineering*. 1–10.
[159] S. Verma and S.K Muttoo. 2016. An android malware detection framework-based on permissions and intents. *Defence Science Journal* (2016).
[160] M. L. Bernardi, M. Cimitile, and F. Mercaldo. 2016. Process mining meets malware evolution: a study of the behavior of malicious code. In *2016 Fourth International Symposium on Computing and Networking*. IEEE, 616–622.
[161] M. Asquith. 2016. Extremely scalable storage and clustering of malware metadata. *Journal of Computer Virology and Hacking Techniques* (2016).
[162] Konrad Jamrozik and Andreas Zeller. 2016. Droidmate: a robust and extensible test generator for android. In *Proceedings of the International Conference on Mobile Software Engineering and Systems*. 293–294.
[163] Yuanchun Li, Ziyue Yang, Yao Guo, and Xiangqun Chen. 2017. Droidbot: a lightweight ui-guided test input generator for android. In *2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*. IEEE, 23–26.
[164] Ashawa Moses and Sarah Morris. 2021. Analysis of mobile malware: a systematic review of evolution and infection strategies. *Journal of Information Security and Cybercrimes Research* 4, 2 (2021), 103–131.
[165] A. Altaher. 2017. An improved Android malware detection scheme based on an evolving hybrid neuro-fuzzy classifier (EHNFC) and permission-based features. *Neural Computing and Applications, 28(12), 4147-4157* (2017).
[166] Gürol Canbek, Nazife Baykal, and Seref Sagiroglu. 2017. Clustering and visualization of mobile application permissions for end users and malware analysts. In *2017 5th International Symposium on Digital Forensic and Security (ISDFS)*. IEEE, 1–10.
[167] A. Cimitile, F. Mercaldo, F. Martinelli, V. Nardone, A. Santone, and G. Vaglini. 2017. Model checking for mobile android malware evolution. In *Proceedings of the 5th International FME Workshop on Formal Methods in Software Engineering*. IEEE, 24–30.
[168] I. R. A. Hamid, N. S. Khalid, N. A. Abdullah, N. H. Ab Rahman, and C. C. Wen. 2017. Android malware classification using K-means clustering algorithm. In *IOP Conference Series: Materials Science and Engineering (Vol. 226, No. 1, p. 012105)*. IOP Publishing.
[169] M. Hassen and P. K. Chan. 2017. Scalable function call graph-based malware classification. In *Proceedings of the Seventh ACM on Conference on Data and Application Security and Privacy (pp. 239-248)*. ACM, 239–248.

[170] Z. Kasiran, N. Awang, and F. N Rusli. 2017. Permission Based in Android Malware Classification. *DEStech Transactions on Engineering and Technology Research* (2017).
[171] Y. Li, J. Jang, X. Hu, and X Ou. 2017. Android malware clustering through malicious payload mining. In *International Symposium on Research in Attacks, Intrusions, and Defenses*. Springer, 192–214.
[172] G. Meng. 2017. *A Semantic-based Analysis of Android Malware for Detection, Generation, and Trend Analysis*. Ph.D. Dissertation. QuNanyang Technological University.
[173] N. Milosevic, A. Dehghantanha, and K. K. R. Choo. 2017. Machine learning aided Android malware classification. *Computers & Electrical Engineering* (2017).
[174] E. Y. Pavlenko, A. V. Yarmak, and D. A. Moskvin. 2017. Application of clustering methods for analyzing the security of Android applications. *Automatic Control and Computer Sciences* (2017).
[175] Shanshan Wang, Zhenxiang Chen, Xiaomei Li, Lin Wang, Ke Ji, and Chuan Zhao. 2017. Android malware clustering analysis on network-level behavior. In *International Conference on Intelligent Computing*. Springer, 796–807.
[176] W. Yang, D. Kong, T. Xie, and C. A. Gunter. 2017. Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM, 288–302.
[177] G. Acampora, M. L. Bernardi, M. Cimitile, G. Tortora, and A. Vitiello. 2018. A Fuzzy Clustering-based Approach to study Malware Phylogeny. In *2018 IEEE International Conference on Fuzzy Systems (FUZZ-IEEE)*. IEEE, 1–8.
[178] Andrea Atzeni, Fernando Díaz, Andrea Marcelli, Antonio Sánchez, Giovanni Squillero, and Alberto Tonda. 2018. Countering android malware: A scalable semi-supervised approach for family-signature generation. *IEEE Access* 6 (2018), 59540–59556.
[179] S.C. Chang, Y.S. Sun, W. L. Chuang, M. C. Chen, B. Sun, and T. Takahashi. 2018. ANTSdroid: Using RasMMA Algorithm to Generate Malware Behavior Characteristics of Android Malware Family. In *2018 IEEE 23rd Pacific Rim International Symposium on Dependable Computing (PRDC)*. IEEE.
[180] M. Fan, J. Liu, X. Luo, K. Chen, Z. Tian, Q. Zheng, and T. Liu. 2018. Android malware familial classification and representative sample selection via frequent subgraph analysis. In *IEEE Transactions on Information Forensics and Securit*. IEEE, 1890–1905.
[181] A. Feizollah, N. B. Anuar, and R Salleh. 2018. Evaluation of network traffic analysis using fuzzy C-means clustering algorithm in mobile malware detection. *Advanced Science Letters* (2018).
[182] G. He, B. Xu, and H. Zhu. 2018. AppFA: A novel approach to detect malicious android applications on the network. *Security and Communication Networks* (2018).
[183] Soussi Ilham, Ghadi Abderrahim, and Boudhir Anouar Abdelhakim. 2018. Clustering Android applications using k-means algorithm using permissions. In *The Proceedings of the Third International Conference on Smart City Applications*. Springer, 678–690.
[184] H. M. Kim, H. M. Song, J. W. Seo, and H. K Kim. 2018. Andro-Simnet: Android Malware Family Classification using Social Network Analysis. In *2018 16th Annual Conference on Privacy, Security and Trust (PST)*. IEEE, 1–8.
[185] F. Martinelli, F. Mercaldo, and A. Saracino. 2018. POSTER: A Framework for Phylogenetic Analysis in Mobile Environment. In *Proceedings of the 2018 on Asia Conference on Computer and Communications Security*. ACM, 825–827.
[186] Annamalai Narayanan, Charlie Soh, Lihui Chen, Yang Liu, and Lipo Wang. 2018. apk2vec: Semi-supervised multi-view representation learning for profiling android applications. In *2018 IEEE International Conference on Data Mining (ICDM)*. IEEE, 357–366.
[187] F. Shang, Y.Li, D. Xiaolin, and D. He. 2018. Android malware detection method based on naive Bayes and permission correlation algorithm. *Cluster Computing* (2018).
[188] Zhi Xiong, Ting Guo, Qinkun Zhang, Yu Cheng, and Kai Xu. 2018. Android malware detection methods based on the combination of clustering and classification. In *International Conference on Network and System Security*. Springer, 411–422.
[189] Ming Fan, Xiapu Luo, Jun Liu, Chunyin Nong, Qinghua Zheng, and Ting Liu. 2019. Ctdroid: leveraging a corpus of technical blogs for android malware analysis. *IEEE Transactions on Reliability* 69, 1 (2019), 124–138.
[190] Ming Fan, Xiapu Luo, Jun Liu, Meng Wang, Chunyin Nong, Qinghua Zheng, and Ting Liu. 2019. Graph embedding based familial analysis of android malware using unsupervised learning. In *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)*. IEEE, 771–782.
[191] K. Ghosh and J. Mills. 2019. Automated Construction of Malware Families. In *International Conference on Security, Privacy and Anonymity in Computation, Communication and Storage*. Springer, 465–474.
[192] R. Kumar, X. Zhang, W. Wang, R. U. Khan, J. Kumar, and A. Sharif. 2019. A multimodal malware detection technique for Android IoT devices using various features. *IEEE Access 7* (2019).
[193] S. Lee, W. Jung, S. Kim, and E. T. Kim. 2019. Android Malware Similarity Clustering using Method based Opcode Sequence and Jaccard Index. In *2019 International Conference on Information and Communication Technology Convergence (ICTC)*. IEEE, 178–183.
[194] M. Khoda, T. Imam, J. Kamruzzaman, I. Gondal, and A. Rahman. 2019. Selective Adversarial Learning for Mobile Malware. In *Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. IEEE, 272–279.
[195] S. Lou, S. Cheng, J. Huang, and F. Jiang. 2019. TFDroid: Android Malware Detection by Topics and Sensitive Data Flows Using Machine Learning Techniques. In *2019 IEEE 2nd International Conference on Information and Computer Technologies (ICICT)*. IEEE, 30–36.

[196] H. Naeem, B. Guo, F. Ullah, and M. R. Naeem. 2019. A Cross-Platform Malware Variant Classification based on Image Representation. *KSII Transactions on Internet & Information Systems* (2019).
[197] M. M. Saudi, S. Sukardi, A. S. M Syafiq, A. Ahmad, and M.A Husainiamer. 2019. Mobile Malware Classification for Cyber Physical System (CPS) based on Phylogenetics. *International Journal of Engineering and Advanced Technology (IJEAT)* (2019).
[198] R. Vega Vega, H. Quintián, C. Cambra, N. Basurto, Á. Herrero, and J. L. Calvo-Rolle. 2019. Delving into Android Malware Families with a Novel Neural Projection Method. *Complexity* (2019).
[199] Lun-Pin Yuan, Wenjun Hu, Ting Yu, Peng Liu, and Sencun Zhu. 2019. Towards large-scale hunting for Android negative-day malware. In *22nd International Symposium on Research in Attacks, Intrusions and Defenses ({RAID} 2019)*. 533–545.
[200] AV-TEST Institute. 2020. *Security Report 2019/2020*. Technical Report. AV-TEST Institute, Germany.
[201] G. Andresini, A. Appice, and D. Malerba. 2020. Dealing with Class Imbalance in Android Malware Detection by Cascading Clustering and Classification. *Complex Pattern Mining* (2020).
[202] Annalisa Appice, Giuseppina Andresini, and Donato Malerba. 2020. Clustering-aided multi-view classification: a case study on android malware detection. *Journal of intelligent information systems* 55, 1 (2020), 1–26.
[203] Pasquale Ardimento, Mario Luca Bernardi, and Marta Cimitile. 2020. Malware Phylogeny Analysis using Data-Aware Declarative Process Mining. In *2020 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*. IEEE, 1–8.
[204] M. G. Cimino, N. De Francesco, F. Mercaldo, A. Santone, and G. Vaglini. 2020. Model checking for malicious family detection and phylogenetic analysis in mobile environment. *Computers & Security* (2020).
[205] ElMouatez Billah Karbab, Mourad Debbabi, Abdelouahid Derhab, and Djedjiga Mouheb. 2020. Android Malware Clustering using Community Detection on Android Packages Similarity Network. *CoRR* abs/2005.06075 (2020). arXiv:2005.06075 https://arxiv.org/abs/2005.06075
[206] Jiayun Xu, Yingjiu Li, Robert Deng, and Ke Xu. 2020. SDAC: A Slow-Aging Solution for Android Malware Detection Using Semantic Distance Based API Clustering. *IEEE Transactions on Dependable and Secure Computing* (2020).
[207] Shweta Sharma, Rakesh Kumar, and C Rama Krishna. 2020. RansomAnalysis: The evolution and investigation of Android ransomware. In *Proceedings of International Conference on IoT Inclusive Life (ICIIL 2019), NITTTR Chandigarh, India*. Springer, 33–41.
[208] Liu Wang, Ren He, Haoyu Wang, Pengcheng Xia, Yuanchun Li, Lei Wu, Yajin Zhou, Xiapu Luo, Yulei Sui, Yao Guo, et al. 2021. Beyond the virus: a first look at coronavirus-themed Android malware. *Empirical Software Engineering* 26, 4 (2021), 82.
[209] Matin Katebi, Afshin Rezakhani, and Saba Joudaki. 2021. ADCAS: Adversarial Deep Clustering of Android Streams. *Computers & Electrical Engineering* 95 (2021), 107443.
[210] ElMouatez Billah Karbab and Mourad Debbabi. 2021. Resilient and Adaptive Framework for Large Scale Android Malware Fingerprinting using Deep Learning and NLP Techniques. *CoRR* abs/2105.13491 (2021). arXiv:2105.13491 https://arxiv.org/abs/2105.13491
[211] Qian Li, Qingyuan Hu, Yong Qi, Saiyu Qi, Xinxing Liu, and Pengfei Gao. 2021. Semi-supervised two-phase familial analysis of Android malware with normalized graph embedding. *Knowledge-Based Systems* 218 (2021), 106802.
[212] Zhen Liu, Ruoyu Wang, Nathalie Japkowicz, Deyu Tang, Wenbin Zhang, and Jie Zhao. 2021. Research on unsupervised feature learning for Android malware detection based on restricted Boltzmann machines. *Future Generation Computer Systems* 120 (2021), 91–108.
[213] Altyeb Altaher Taha and Sharaf Jameel Malebary. 2021. Hybrid classification of Android malware based on fuzzy clustering and the gradient boosting machine. *Neural Computing and Applications* 33, 12 (2021), 6721–6732.
[214] Jiezhong Xiao, Qian Han, and Yumeng Gao. 2021. Hybrid Classification and Clustering Algorithm on Recent Android Malware Detection. In *2021 5th International Conference on Computer Science and Artificial Intelligence*. 249–255.
[215] Liang Zhao, Jiayang Wang, Ye Chen, Fan Wu, Yuan'an Liu, et al. 2021. Famdroid: learning-based android malware family classification using static analysis. *arXiv preprint arXiv:2101.03965* (2021).
[216] ElMouatez Billah Karbab and Mourad Debbabi. 2021. PetaDroid: Adaptive Android Malware Detection Using Deep Learning. In *Detection of Intrusions and Malware, and Vulnerability Assessment: 18th International Conference, DIMVA 2021, Virtual Event, July 14–16, 2021, Proceedings*. 319–340.
[217] Kazuya Nomura, Daiki Chiba, Mitsuaki Akiyama, and Masato Uchida. 2021. Auto-creation of Android Malware Family Tree. In *ICC 2021-IEEE International Conference on Communications*. IEEE, 1–6.
[218] Hemant Rathore, Sanjay K Sahay, Shivin Thukral, and Mohit Sewak. 2021. Detection of malicious android applications: Classical machine learning vs. deep neural network integrated with clustering. In *Broadband Communications, Networks, and Systems: 11th EAI International Conference, BROADNETS 2020, Qingdao, China, December 11–12, 2020, Proceedings 11*. Springer, 109–128.
[219] Nazifa Mosharrat, Iqbal H Sarker, Md Musfique Anwar, Muhammad Nazrul Islam, Paul Watters, and Mohammad Hammoudeh. 2022. Automatic Malware Categorization Based on K-Means Clustering Technique. In *Proceedings of the International Conference on Big Data, IoT, and Machine Learning: BIM 2021*. Springer, 653–664.
[220] Sanjeev Kumar, B Janet, and Subramanian Neelakantan. 2022. Identification of malware families using stacking of textural features and machine learning. *Expert Systems with Applications* 208 (2022), 118073.
[221] Omid Mirzaei, Guillermo Suarez-Tangil, Jose M de Fuentes, Juan Tapiador, and Gianluca Stringhini. 2019. Andrensemble: Leveraging api ensembles to characterize android malware families. In *Proceedings of the 2019 ACM Asia conference on computer and communications*

security. 307–314.
[222] Carlos Castillo, Raj Samani, et al. 2014. Mcafee labs threats report. *McAfee Inc., Santa Clara, CA. Available: http://www. mcafee. com/us/resources/reports/rp-quarterlythreat-q1-2014. pdf* (2014).
[223] Giovanni Apruzzese, Pavel Laskov, and Aliya Tastemirova. 2022. SoK: The impact of unlabelled data in cyberthreat detection. In *2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P)*. IEEE, 20–42.
[224] D. H. Huson, R. Rupp, and C. Scornavacca. 2010. *Phylogenetic networks: concepts, algorithms and applications*. Cambridge University Press.
[225] B. H. Anderson. 2014. *Integrating Multiple Data Views for Improved Malware Analysis*. Ph.D. Dissertation. The University of New Mexico.
[226] A. Walenstein and A. Lakhotia. 2007. The software similarity problem in malware analysis. In *Dagstuhl Seminar Proceedings. Schloss Dagstuhl-Leibniz-Zentrum für Informatik*.
[227] J. E. Fowler. 2016. Delta Encoding of Virtual-Machine Memory in the Dynamic Analysis of Malware. In *Data Compression Conference (DCC)*. IEEE, 592–592.
[228] A. Walenstein and A. Lakhotia. 2012. A transformation-based model of malware derivation. In *7th International Conference on Malicious and Unwanted Software*. IEEE, 17–25.
[229] A. Pfeffer, C. Call, J. Chamberlain, L. Kellogg, J. Ouellette, T. Patten, and R Hall. 2012. Malware analysis and attribution using genetic information. In *2012 7th International Conference on Malicious and Unwanted Software*. IEEE, 39–45.
[230] Kehinde Oluwatoyin Babaagba and Samuel Olumide Adesanya. 2019. A study on the effect of feature selection on malware analysis using machine learning. In *Proceedings of the 2019 8th international conference on educational and information technology*. 51–55.
[231] Maitreyee Dutta et al. 2012. Performance analysis of clustering methods for outlier detection. In *2012 Second International Conference on Advanced Computing & Communication Technologies*. IEEE, 89–95.