Hindawi
Complexity
Volume 2021, Article ID 4551067, 28 pages
https://doi.org/10.1155/2021/4551067

Research Article

Identifying Major Research Areas and Minor Research Themes of Android Malware Analysis and Detection Field Using LSA

Deepak Thakur,[1] **Jaiteg Singh**,[1] **Gaurav Dhiman**,[2] **Mohammad Shabaz**,[1,3] **and Tanya Gera**[1]

[1]*Chitkara University Institute of Engineering and Technology, Chitkara University, Punjab, India*
[2]*Government Bikram College of Commerce, Patiala, Punjab, India*
[3]*Arba Minch University, Arba Minch, Ethiopia*

Correspondence should be addressed to Jaiteg Singh; jaiteg.singh@chitkara.edu.in and Mohammad Shabaz; mohammad.shabaz@amu.edu.et

Received 12 August 2021; Revised 25 August 2021; Accepted 28 August 2021; Published 7 September 2021

Academic Editor: Long Wang

Contemporary technologies have ensured the availability of high-quality research data shared over the Internet. This has resulted in a tremendous availability of research literature, which keeps evolving itself. Thus, identification of core research areas and trends in such ever-evolving literature is not only challenging but interesting too. An empirical overview of contemporary machine learning methods, which have the potential to expedite evidence synthesis within research literature, has been explained. This manuscript proposes Simulating Expert comprehension for Analyzing Research trends (SEAR) framework, which can perform subjective and quantitative investigation over enormous literature. TRENDMINER is the use case designed exclusively for the SEAR framework. TRENDMINER uncovered the intellectual structure of a corpus of 444 abstracts of research articles (published during 2010–2019) on Android malware analysis and detection. The study concludes with the identification of three core research areas, twenty-seven research trends. The study also suggests the potential future research directions.

1. Introduction

Data are ubiquitous, whether they are on blogs, social media platforms, discussion forums, reviews, literature, or research studies. Extracting information out of such multidimensional data is not only important but is challenging too. There is a paradigm shift in knowledge transfer among different subareas of the research held. Manual systematic reviews [1] or semiautomated [2–4] are two methods that can be employed for systematic reviews. Manual reviews are more critical and can be biased [5]. The selection of focus area, attribute selections, and interpretation entirely depends on the expertise of the reviewer. Elaborating present trends and forecasting future directions from the existing literature is not only challenging but also time-consuming for systematic manual reviews. In contrast, semiautomated methods are more generic in finding the trends [6]. Deployment of machine learning techniques within semiautomated review methods can facilitate researchers to gain a dynamic review of any literature of choice. This manuscript offers an empirical overview of contemporary machine learning methods, which have the potential to expedite evidence synthesis within research literature using Simulating Expert comprehension for Analyzing Research trends (SEAR) framework. SEAR deploys human-like intelligence to manage knowledge and information effectively. The framework leverages information modeling techniques to simulate how humans read, understand, interpret the meaning of words, and map the semantic relationship in text. The proposed SEAR framework has been deployed as TRENDMINER. As a use case, a corpus pertaining to Android security was used. During the last decade, pieces of malware are propagating at a tremendously high rate using persistent and sophisticated techniques [7]. This situation has led researchers to devise various analyses, detection, and mitigation methods, resulting in building a substantial body of literature. Continuous ongoing research augmentation of the

Android platform and malware has resulted in humongous literature. This research literature has offered numerous research prospects and has promulgated contemporary challenges within the domain. To the best of our knowledge, there is no literature investigating those challenges and research directions using semiautomated machine learning-based methods. Unlike previous works, this study is far beyond any generic study on mobile attack vectors or defense [8–11]. Instead, it oriented around emerging research trends and also suggested future directions using quantitative semiautomatic approaches. With respect to the technique being employed and dataset chosen, this study intends to answer the following research questions as framed by the research community [12]:

RQ1: can the proposed framework uncover leading researchers within a research domain?

RQ2: are those frameworks robust enough to determine the most investigated research areas?

RQ3: would the proposed framework reveal how the focus of topics within each core research area has changed over time?

RQ4: can it unfold the future directions within the research domain of choice?

Numerous topic modeling techniques such as Latent Semantic Analysis (LSA), Latent Dirichlet Allocation (LDA), Probabilistic Latent Semantic Analysis (PLSA), and Correlated Topic Modeling (CTM) are compared and summarized in Table 1. LSA has found to be appropriate for this work as it was successfully deployed by various researchers to analyze research trends in domains such as Volunteered Geographic Information [13], Building Information Modeling [6], Supply Chain Management [14], and OpenStreetMap [5]. Several studies have demonstrated the validity of LSA in constructing a framework that leverages semantic-driven analysis to recognize and infer information from the content. Semantic-driven analysis understands the text structure, words, and the topic discussed in the document [15–29]. LSA is dependably effective in data recovery and question streamlining. It recognizes a whole of the settings where a word could show up and figures out how to set up a typical factor to address basic ideas. Examination in brain science proposes that LSA mirrors the human brain to sift through semantics from the content.

The authors in [30] proposed a method called word2vec-based LSA as a new topic modeling technique to study the trend analysis in blockchain technology. Their proposed methodology was composed of neural network-based word embedding and spherical K-means clustering. They also discussed the downside of traditional methods such as bibliometric and frequency-based analysis. They also compared their results with PLSA. In their findings, PLSA is not successful in capturing the context of the document whereas their proposed methodology was able to capture the context on real data. The authors in [31] reviewed various theoretical aspects of LSA and spatial models. They discussed various characteristics and properties empowering LSA as a suitable topic modeling technique. They also revealed some limitations and misunderstandings related to LSA. They argue that LSA has traveled a lot in providing good results as compared to other models. As a future scope, they mentioned that the fusion of different models tends to produce a coherent ecosystem. The authors in [32] performed text mining using LSA and nonnegative matrix factorization (NNMF). They discussed the strengths of LSA to process the highly sparse term-document matrix with less computation overhead. They discussed the stability of results and clustering performance while deploying LSA in their methodology. They also integrated K-means for cluster formation in their proposed methodology. In [33], the authors utilized LSA as an application to determine the memory reconstruction. LSA was applied to test that sleep reduces the semantic coherence of memory recall. In [34], the authors attempted to deploy kernel matrix estimation using LSA to increase the sharpness of the blurred image. The authors in [35] defined the applicability of LSA in determining problems in aerospace science. The authors in [36] utilized the LSA to extract the features across different knowledge domains such as information systems and operations management. In [37], the authors studied the impact of technology-enhanced learning in higher education. The topics were discovered and analyzed from the corpus related to technology-enhanced learning. The authors in [38] proposed a new taxonomy and future research directions in industry 4.0 using LSA. Various research themes related to the field were discovered and discussed.

Android security is an interesting area to explore. Malware authors tend to plant malicious code matrices inside legitimate applications to unlock their unscrupulous motives. A continuing thread of malware proliferation had let the research community perform various studies related to Android malware detection and analysis techniques. The traditional methods such as bibliometric analysis or frequency-based analysis focus on the quantitative analysis but not on the qualitative analysis [30]. These approaches are highly effort-demanding and time-consuming to perform trend analysis. The authors need to perform a full-text investigation to study the trends in the Android security field [39–41]. These approaches did not reveal the insights of the literature as they consider limited databases with limited time frames. The topic modeling techniques such as Latent Semantic Analysis (LSA) had confirmed their usefulness in determining comprehensive and detailed trend analysis. Studies in [42–45] have witnessed the use of topic modeling to identify research trends to a great extent and shown advantages over traditional methods. Table 1 shows the comparison of LSA with other topic modeling techniques. LSA focuses on revealing the diverse topics that emerged during the given timeline and provides a quantitative and qualitative evaluation. The results produced by LSA help the practitioner to pursue various potential research opportunities. LSA is used on top of this matrix to drastically reduce the vector size and capture latent topics in the corpus, while being able to infer relationships between relevant terms and respective documents, without any loss of context.

The remainder of the paper has been arranged as follows: Section 2 depicts the brief introduction to the SEAR framework. Materials and methods are discussed in Section

Table 1: Comparison of topic modeling techniques.

Technique name	Characteristics	Limitations	Area
Latent Semantic Analysis	Using SVD features, LSA can perform dimensionality reduction of TF-IDF. LSA works on synonyms of words.	Expert help is always required for labeling the topics. The interpretation of loading values sometimes becomes cumbersome.	(i) Spam filtering (ii) Automation in essay grading (iii) Topic identification.
Probabilistic Latent Semantic Analysis	Topics can be easily represented through multinomial random variables. Ability to partially handle polysemy.	Unable to perform document level modeling.	(i) Automation in essay grading (ii) Automation in question recommendation.
Latent Dirichlet Allocation	Provides multinomial distribution across words and Dirichlet distribution over topics. Capable of handling long-length documents.	Cannot predict relations among topics.	(i) Automatic labeling (ii) Emotion topic (iii) Sentiment summarization.
Correlated Topic Model	Uses logistic normal distribution for topic clustering. Produces topic graphs also.	Complex computation is involved in its processing. Too many generic words may lead to inefficiency.	(i) Query classification (ii) Topic identification (iii) Image retrieval.

3. Section 4 discusses the research questions and examines potential future research directions. Section 5 examines the outline of the proposed solution as an implication of future examination while Section 6 discusses the limitation of the investigation. Conclusions and findings are discussed in Section 7. Section 8 discusses the practical implications and future avenues of the research.

2. Proposed SEAR Framework

The proposed SEAR framework operates in the sequence as given in Figure 1.

Step 1: this step involves data gathering methods, creation of repository and XML parser, and conversion of documents to text files.

Step 2: this step involves data preprocessing of the corpus. Stop words and punctuations should be removed from the dataset, and it should be normalized before performing any text mining task.

Step 3: this step implements the TF-IDF and SVD technique, discussed in the further sections.

Step 4: this step involves the identification of core research areas and research trends. It also focuses on the mapping of research trends with the research area.

The SEAR framework utilizes a semantic analysis technique called LSA. It is a well-established algorithm to convert unstructured raw textual data into organized information objects and further analyze these objects to recognize patterns for the revelation of learning [2, 46, 47]. It employs a systematic and comprehensive approach to uncover the research trends in a vast literature dataset [3, 21, 24, 25, 48–52]. This study aims to map the semantic relationship between documents and terms in a large corpus to reveal the varied contextual latent classes using LSA.

The steps in applying LSA to the Android security corpus is identical to previously reported studies [3, 51, 53–57]. The following sections discuss the detailed procedure of this study.

3. The Use Case of SEAR Framework: TRENDMINER

TRENDMINER is the use case of the SEAR framework which takes text documents as an input, as shown in Figures 2 and 3. The dataset of 444 abstracts is considered sufficiently large enough for performing text mining, as explained in [3]. Python 3.7 programming language was used to perform all the experimentation. Table 2 shows the software versions used in our work. The machine used for the experimentation was configured with Intel Core i5 6200U with 2.4 GHz and 8 GB RAM. Once the literature dataset on Android security is successfully uploaded on TRENDMINER, it is further fed to Latent Semantic Analysis (LSA), which is a backbone of TRENDMINER. LSA is a text data mining and natural language processing technique used to retrieve and query a massive corpus of literature [51, 56, 58]. As a scientific and measurable strategy, LSA is utilized to recognize the latent concepts inside the textual data at the semantic level [59–63].

3.1. Step 1: Data Acquisition.

3.1. Step 1: Data Acquisition. This section reveals the keywords, search strategy, and selection criteria used for preparing the large corpus. Reputed databases were used for the collection of research articles on Android security. Inclusion and exclusion criteria were applied to refine the searching results to get relevant research articles. The repository was made to achieve standard uniformity across the research articles.

3.1.1. Task A: Dataset Preparation. The first task was to prepare the literature dataset for TRENDMINER. The approach followed for collecting the literature dataset is primarily focused upon the structure of Android applications, the probable vulnerabilities within existing application development along the methods adopted for malware identification and mitigation. The strategy adopted for searching and selecting literature is defined by 3C's Formula, depicted in Figure 4:

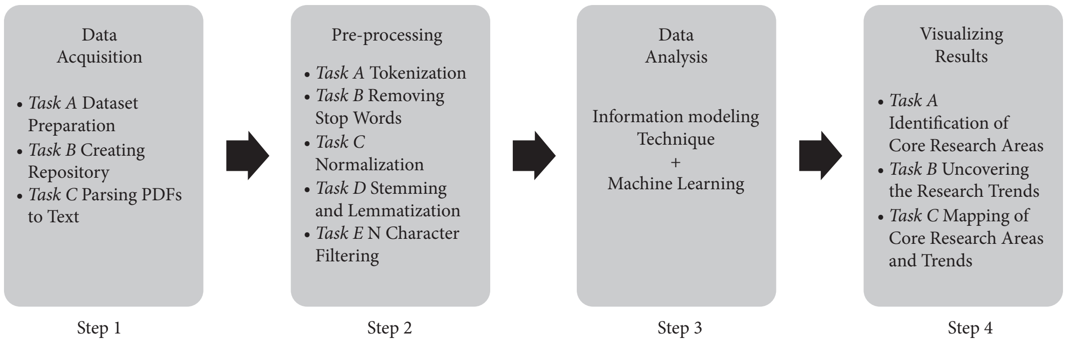

Data Acquisition
• Task A Dataset Preparation
• Task B Creating Repository
• Task C Parsing PDFs to Text
Step 1
Pre-processing
• Task A Tokenization
• Task B Removing Stop Words
• Task C Normalization
• Task D Stemming and Lemmatization
• Task E N Character Filtering
Step 2
Data Analysis
Information modeling Technique + Machine Learning
Step 3
Visualizing Results
• Task A Identification of Core Research Areas
• Task B Uncovering the Research Trends
• Task C Mapping of Core Research Areas and Trends
Step 4

Figure 1: Sequence diagram of SEAR framework.

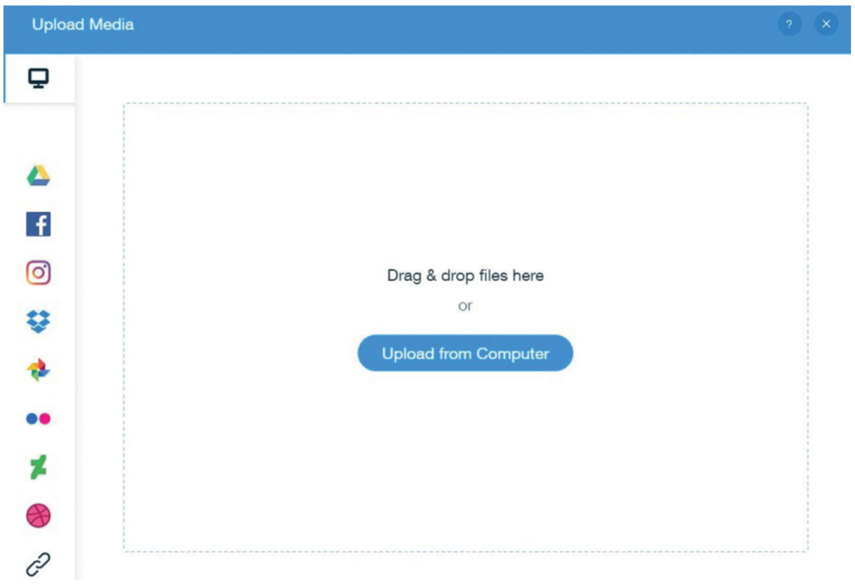
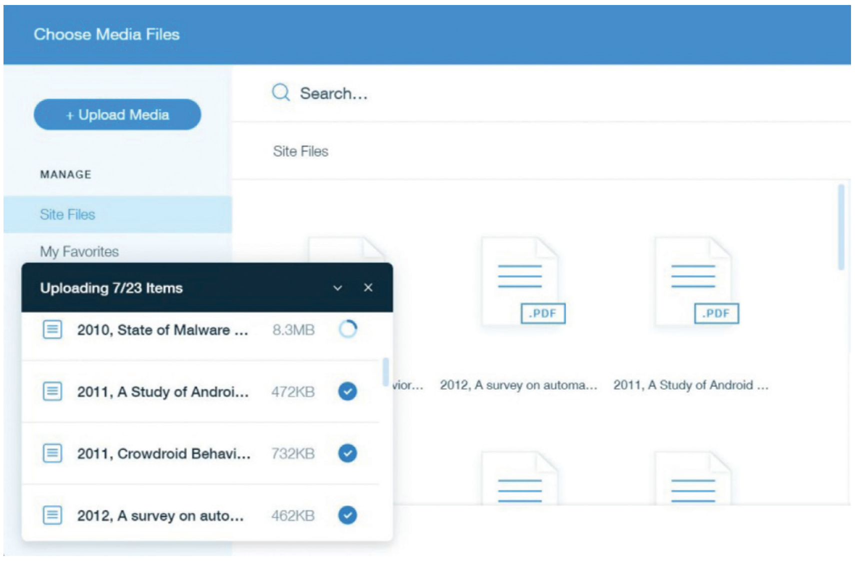


Figure 2: Uploading interface of TRENDMINER.



Uploading 7/23 Items

2010, State of Malware ...	8.3MB	
2011, A Study of Androi...	472KB	✓
2011, Crowdroid Behavi...	732KB	✓
2012, A survey on auto...	462KB	✓

Figure 3: Files getting uploaded on TRENDMINER.

Table 2: Software specifications.

Library	Version	Implementation in TRENDMINER	Open source
PDFMiner	≥20140328	Used in data acquisition (parsing PDFs to text)	Yes
NLTK	≥3.4	Preprocessing (all tasks)	Yes
Scikit-Learn	≥0.20rc1	Data analysis	Yes

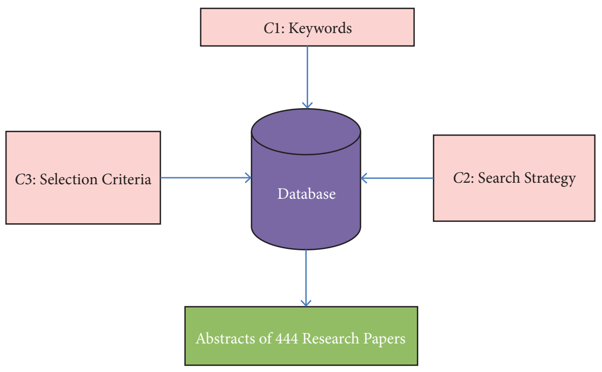


Figure 4: Dataset preparation using 3C's Formula.

(1) Component 1: Keywords. The articles were selected using keywords such as "malware," "vulnerability," "security," "privacy," "monitoring," "application," "smartphone," "android," "virus," "static," "dynamic," "detection," and "data flow."

(2) Component 2: Search Strategy. The TRENDMINER considered reputed research from prominent databases such as IEEE Xplore, ACM Computing Library, Science Direct, Springer, and Google Scholar which was queried to collect high-quality papers on Android malware analysis and detection techniques. Scopus indexed articles from prominent databases were duly included while searching the literature. Figure 5 illustrates the proportion of Scopus indexed articles in our corpus.

(3) Component 3: Selection Criteria. Raw results from the databases mentioned above were refined based on the Android operating system. Papers on operating systems such as Symbian and iOS were discarded.

3.1.2. Task B: Creating a Repository for TRENDMINER. Mendeley, a tool from Elsevier [64], has been used to build the literature database. It provides a systematic way to retrieve the authors, years, and abstracts of all the research papers indexed into its file system and also to export all of them as citations and XML tree structures. Parsing of resultant XML tree structure was one of the significant challenges during this study. A consistent naming convention for the whole literature dataset was necessary. Renaming the articles using particular objects common to all research documents will have a significant impact on their future ease.

A module in TRENDMINER was developed, known as XML Parser. The purposefully generated XML corpus was further parsed to a more structured format, i.e., comma-separated values (CSVs). Figure 6 shows the generic conversion process flow.

The exported files consist of metadata information such as authors, year of publication, and publishers. The following observations were made during the prelim analysis of the corpus. Based on the number of occurrences in the dataset, the top researchers with the most publications on Android security during the period 2010–2019 were calculated and are presented in Figure 7.

Figure 8 shows the top fifteen journals publishing articles related to Android security. Figure 7 interprets that the top authors were Wang, Xiaofeng, and Jiang, Xuxian, with 13 publications, with Zhou, Yajin closely following on 12. The graph obtained was from the analysis performed on the dataset chosen, as described above. Figure 8 identifies Computers and Security (Elsevier) and IEEE, as the top publishers, publishing research in the Android malware and security field. NDSS, Springer, and ACM closely follow them.

3.1.3. Task C: Parsing the PDF Documents to Text. Conversion of pdf to text was subsequently performed to make the dataset input ready, compatible with TRENDMINER. Various tool options for the conversion process are available, namely, PDFMiner, Tika, and Textract. PDFMiner [65] was opted in the experimental study because of the following significant benefits:

(i) PDFMiner can obtain the exact location of text on a page along with information such as fonts or a number of lines.

(ii) It facilitates the conversion of PDF files into other text formats (such as HTML).

(iii) It provides accurate results even under extreme conditions such as parsing large corpus.

3.2. Step 2: Preprocessing the Text Files. After the successful conversion to text files, the next step was to employ preprocessing procedures. The preprocessing module in TRENDMINER helps to gain quality information out of the text by applying appropriate preprocessing techniques. For any text mining algorithm, the preprocessing of the collected dataset is an essential step [66, 67]. This involves the expulsion of names, numbers, abbreviations, slang, acronyms, punctuation, and N characters as recommended in [3].

Preprocessing of corpus involves the execution of the following procedure, developed in Python platform using NLTK package. NLTK is Natural Language Toolkit [68].

3.2.1. Task A (Tokenization). In this step, large chunk of text was tokenized into sentences, then sentences into words.

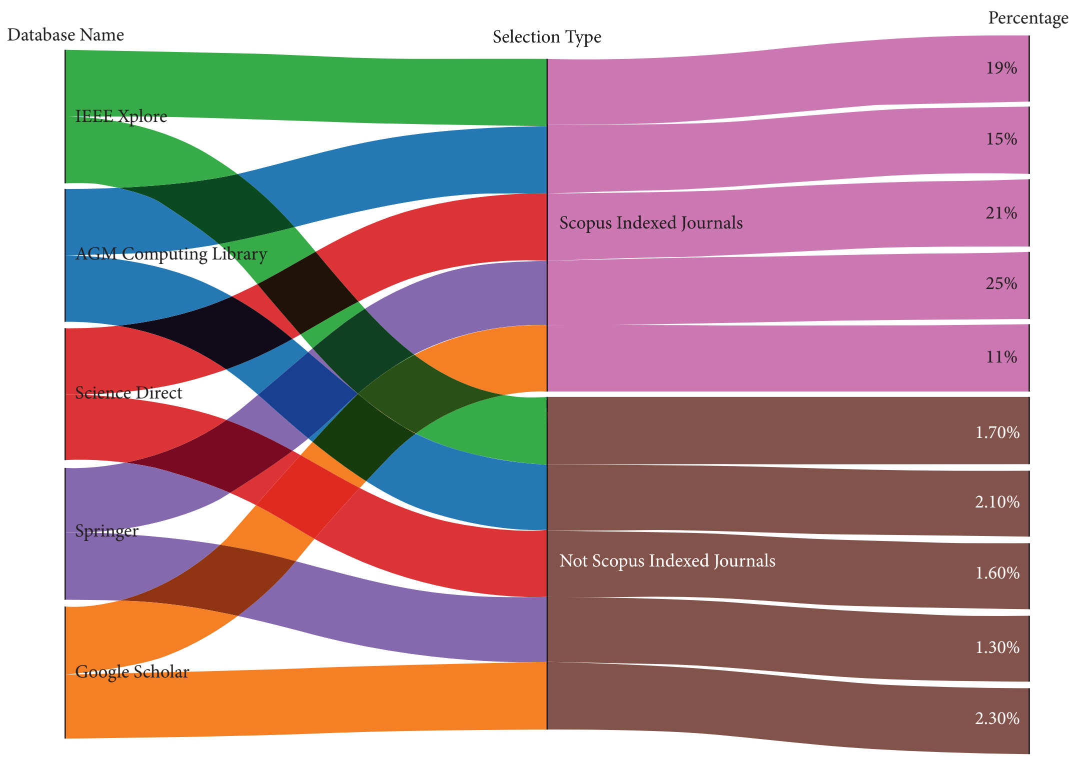


Figure 5: Distribution of Scopus indexed articles among the databases.

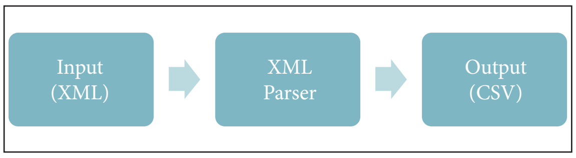


Figure 6: Parsing an XML to CSV.

3.2.2. Task B (Removing Stop Words). Stop words using NLTK's support and common words (sample, benign, learning, malware detection, malware, detection, training, layer, channel, attacker, password, market, call, warning, algorithm, installation, detector, socket, etc.) were removed.

3.2.3. Task C (Normalization). Normalization is applied over the words to introduce uniformity and maintain consistency among the text documents. The task of normalization is composed of several subtasks such as removing punctuation from the text, changing overall content to a similar case either uppercase or lowercase, and converting numbers to words. Normalization helps to keep all words on equivalent balance to allow smooth processing of the textual data.

3.2.4. Task D (Stemming and Lemmatizing). For further processing of documents, the dictionary size has to be reduced and should be populated with unique words. Stemming and lemmatizing are the techniques that are performed over the words to reduce the inflection. The idea is to reduce the words to the common root form. In stemming, base form is known as stem while in the case of lemmatizing, it is known as a lemma. Stems might not be actual or real words, but on the other hand, lemmas are the actual language words. These two techniques help in achieving faster processing of text documents.

3.2.5. Task E (Character Filtering). All words less than length 4 were omitted [3].

It is to be noted that the initial dataset contained 60,184 tokens which represents the length of the vocabulary in the entirety of the corpus. Before the dataset is fed to other computational steps, it has to be nonredundant and free from any kind of noise. After applying appropriate preprocessing procedures as discussed previously, the word list was retained with 1944 tokens. In this study, 444 documents and the resulted wordlist of the 1944 tokens represent columns and rows, respectively. A term frequency is created where each term maps to a count of occurrence in each document. Furthermore, this matrix is transformed into a weighted matrix using the TF-IDF weighting scheme.

3.3. Step 3: Data Analysis Using Information Modeling and Machine Learning Techniques. This work makes use of the information modeling technique to expedite the data analysis process over the corpus. With the conjunction of information modeling and machine learning techniques, human interpretable topics can be extracted from a document corpus. Machine learning approaches enhance the ability of information modeling techniques by allowing researchers to intelligently extract and manage the crucial

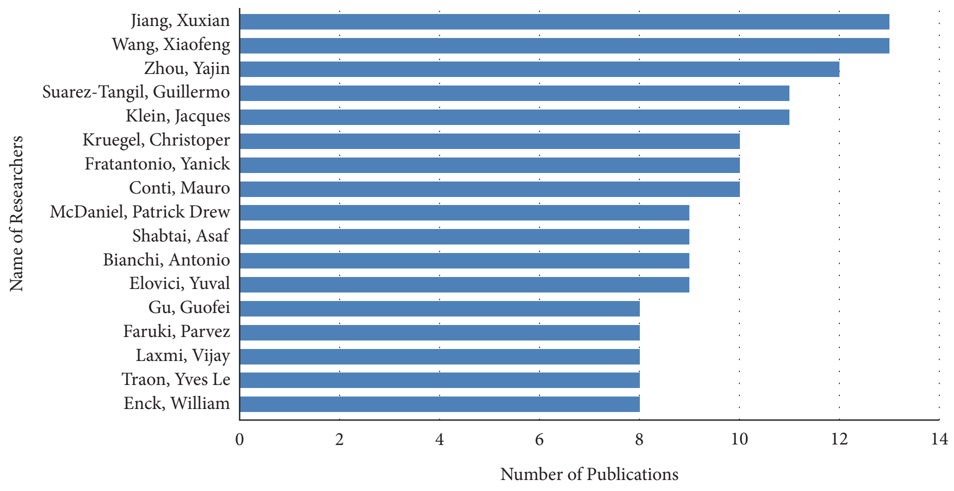


Figure 7: Top researchers in Android security research.

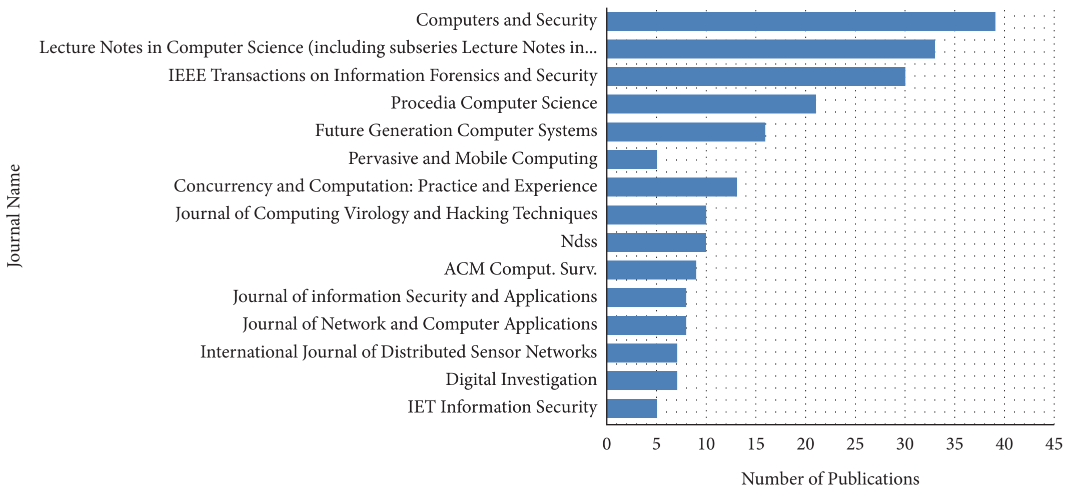


Figure 8: Top journals focusing on Android security field.

information to make smart decisions. Deploying Latent Semantic Analysis (LSA) as the information modeling technique can automatically identify topics and unveil hidden patterns in the vast corpus of data. LSA uses the matrix method called Singular Value Decomposition (SVD) to construct a low-rank approximation from extensive matrix data. SVD is the major strength of the LSA and one of the basic machine learning algorithms. It reduces the dimensions of the data without losing a significant amount of information. The main idea is to apply LSA on a document set and unsupervised machine learning approach on a reduced dimension set to group similar documents according to their topic areas. *K*-means, which is the unsupervised machine learning approach, fitted in the LSA model to uncover the latent structure of the corpus.

3.3.1. Task A: From Documents to Matrices—TF-IDF (Term Frequency Inverse Document Frequency). In this study, a mapping needs to be investigated from the documents to the latent topics that they all relate to. For that, the most important words were to be identified which can later lead to the latent topic discovery. The TRENDMINER leverages the essence of the technique, called Term Frequency Inverse Document Frequency (TF-IDF). There are other weighting methods are available for the analysis. The most common weighting schemes are TF-IDF and log-entropy. As per the study in [3], a potential weakness of log-entropy was discovered and it proved to be biased towards high-frequency terms in the dataset. For instance, log-entropy produces a better result with article titles or documents with a short text. TF-IDF performs better in discovering the patterns in large semantic spaces of larger groups of terms. Motivated by this finding, we utilized the TF-IDF technique as the weighting method in the study.

The Latent Semantic Analysis (LSA) topic model algorithm requires a document-term matrix as the main input. TF-IDF helped in maintaining a document-term matrix that described the frequency of terms that occur in a collection of documents. The documents and words in a matrix correspond to columns and rows, respectively. TF-IDF has widely been into usage for better topic analysis [3, 69, 70]. The resulting document-term matrix of the example stated in the previous example is presented in Table 3.

(1) TF (Term Frequency). It processes the standardized Term Frequency (TF), which is determined as the frequency a term shows up in a report, separated by the complete number of terms in that record, refer to equation (1). TF matrix is shown in Table 4:

$$\mathrm{TF}(t,d) = \frac{\text{number of occurences of term } t \text{ appears in document } d}{\text{total number of terms in the document}}. \tag{1}$$

(2) IDF (Inverse Document Frequency). It estimates how significant a term is. IDF is processed as the logarithm of the quantity of records in the corpus isolated by the quantity of reports where the particular term shows up. Nonetheless, it is realized that specific terms, for example, "is," "of," and "that" or space explicit words, may seem a great deal of times however have little significance. In this way, there is a need to overload the continuous terms while increasing the uncommon ones, by figuring condition 2. IDF grid is introduced in Table 5:

$$\mathrm{IDF}(t,d) = \log\frac{\text{total number of documents}}{\text{number of documents with term } t \text{ in it}}. \tag{2}$$

The below equation (3) presents the TF-IDF scores:

$$w_{t,d} = \mathrm{TF}_{t,d} \times \mathrm{Log}\frac{N}{df_t}. \tag{3}$$

In equation (3), t means the terms, d signifies each record, and N indicates the complete number of reports. Consider Table 6, which addresses the report term lattice with TF-IDF scores for the recently expressed model. A term will have a huge weight when it much of the time happens across the archive yet inconsistently across the corpus. The word malware may show up frequently in an archive, but since it is probable reasonably entirely expected in the remainder of the corpus. To reveal the connection between the words and records and catch the latent themes inside the Android security dataset, dimensionality reduction must be performed, as examined in the following area.

3.3.2. Task B: Learning Latent Relationships between Documents Using SVD (LSA).

Utilizing SVD, two sets of loading matrices were produced as the output of LSA. One is a document-to-topic matrix and the other one is a term-to-topic matrix. The topic solutions are the number of research themes in the literature dataset. High term or document loading in the matrix cell discloses the fact that a specific term or document is more inclined towards a particular topic solution. The researcher can adjust the detail level of a number of topic solutions for identifying research areas and trends. Smaller values of topic solution represent common research core areas, and higher values of topic solution represent principal research trends [51].

Truncated SVD is a framework variable-based math method that breaks down the TF-IDF lattice into a result of three grids: U, Σ, and V. The SVD disintegration is shown in

$$A = U \times \Sigma \times V^T. \tag{4}$$

Here, A addresses the TF-IDF lattice, U addresses the document-to-topic framework portraying relationship between documents attached to different concepts, V addresses the term-to-topic depicting relationship among concepts and terms, and Σ is composed of nonnegative numbers.

Suppose d is the number of records, t is the number of terms in the documents, and k is considered as the hyperparameter demonstrating the quantity of points to be separated from the corpus. A_k is the low-rank estimate of matrix A and can be delivered utilizing shortened SVD as continues in

$$A_k = U_k \times \Sigma_k \times V_k^T, \tag{5}$$

where U_k is the document-to-topic matrix ($d \times k$), V_k is a term-to-topic matrix ($t \times k$), and Σ_k is the topic-to-topic matrix ($k \times k$). Table 6 shows the changed term frequencies subsequent to applying TF-IDF. SVD procedure must be applied to the TF-IDF matrix introduced in Table 6.

Tables 7 and 8 contain the factor loading values that are arbitrarily positive and negative. The set of terms and documents need to be mapped with the latent topics. To interpret the meaning of the loading values, the technique known as varimax rotation was applied on terms and document loading matrices. The varimax rotation helps to uncover the best correlation of terms with the latent topics. The rotation magnifies the association of terms and documents to the latent topics. Furthermore, a threshold value needs to be selected to discover the significant terms as discussed in [3, 5]. Empirical probability distribution was utilized to select the threshold values for different factor solutions. The loading values are transformed into a vector and sorted in descending order, thereby defining the threshold as retaining $1/n$ of the loadings, where n is the factor solution as explained in [5, 6]. For each factor solution, loading values are grouped by considering their absolute values to unveil latent topics. As an application of LSA followed by an unsupervised machine learning approach, discussed further, it will help to identify topic solutions.

TRENDMINER is used to identify the core research areas and significant research trends in Android security, and an optimal value for k topic solutions has to be determined. Choosing an optimal value for k is always a challenge; because the more the number of dimensions k chosen, the more will be the risk of induction of noise in the data [58, 71]. However, at the same time, selecting a smaller value of k will lead to losing important semantics. It is a good practice to include a bigger k, as an approach to deduce more trends or classify many trends into a single category [72]. A k-iterative process has been applied to uncover the core research areas and their subclassification of related trends. SVD provides the matrix of singular values that are defined as the square root of the eigenvalues. These values provide

Table 3: Document-term matrix describing frequency of terms.

Terms	Doc1	Doc2	Doc3	Doc4	Doc5
Access	0	0	0	0	1
Applic	1	1	1	1	1
Calendar	0	1	0	0	0
Connect	0	0	1	0	0
Contact	0	1	0	0	0
Daili	0	0	0	1	0
Data	0	1	0	1	0
Devic	2	0	0	0	0
Exact	0	0	1	0	0
Find	0	0	1	0	0
Identifi	1	0	0	0	0
Like	0	1	0	0	0
List	0	1	0	0	0
Locat	0	0	1	0	0
Malwar	1	1	0	1	0
Messag	0	0	0	0	1
Misus	0	1	0	0	1
Network	0	0	1	0	0
Number	0	1	0	0	0
Phone	0	1	0	0	0
Read	1	0	0	0	0
Record	0	0	0	1	0
Send	0	0	0	1	0
Server	0	0	0	1	0
Tower	0	0	1	0	0
Track	1	0	1	0	0
Uniqu	1	0	0	0	0
Usag	0	0	0	1	0
User	1	1	1	0	0
Various	0	0	0	1	0
Wifi	0	0	1	0	0

Table 4: Term frequency scores for each document.

Documents	Term frequency scores
Doc1	{"Malwar": 0.1111111111111111, "applic": 0.1111111111111111, "read": 0.1111111111111111, "uniqu": 0.1111111111111111, "devic": 0.2222222222222222, "identifi": 0.1111111111111111, "track": 0.1111111111111111, "user": 0.1111111111111111}
Doc2	{"Malwar": 0.09090909090909091, "applic": 0.09090909090909091, "misus": 0.09090909090909091, "user": 0.09090909090909091, "data": 0.09090909090909091, "like": 0.09090909090909091, "phone": 0.09090909090909091, "number": 0.09090909090909091, "contact": 0.09090909090909091, "list": 0.09090909090909091, "calendar": 0.09090909090909091}
Doc3	{"Applic": 0.1, "track": 0.1, "exact": 0.1, "locat": 0.1, "user": 0.1, "find": 0.1, "wifi": 0.1, "network": 0.1, "tower": 0.1, "connect": 0.1}
Doc4	{"Various": 0.1111111111111111, "malwar": 0.1111111111111111, "applic": 0.1111111111111111, "record": 0.1111111111111111, "daili": 0.1111111111111111, "usag": 0.1111111111111111, "data": 0.1111111111111111, "send": 0.1111111111111111, "server": 0.1111111111111111}
Doc5	{"Applic": 0.25, "access": 0.25, "messag": 0.25, "misus": 0.25}

the concept strength and are arranged in descending order. The k singular values are selected using a scree plot as depicted in Figure 9. As illustrated in the study [24], a high level of topics must be chosen using an empirical approach that involves multiple trials of LSA. The number of factors in individual trials ranged from 2 to 10. After reviewing high-loading terms/documents for each factor solution, experts decided to set three as core high-level research areas. It should be noted that it also depends upon the semantic space chosen for the experimentation.

Furthermore, based on the expert opinions and scree plot analysis [14, 73], dimensionalities of 27 topics were found to be significant elbow point detected through an iterative log-likelihood ratio test on eigenvalues [74]. The optimal number of twenty-seven topic solutions can be considered optimal for depicting the research trends in Android security of a large corpus; in addition, three topic solutions were considered to describe the core research areas. Topic clustering, topic labeling, and detailed analysis have been discussed in further sections.

Table 5: Inverse document frequency score for each term.

Terms	IDF score
Access	2.098612
Applic	1.000000
Calendar	2.098612
Connect	2.098612
Contact	2.098612
Daili	2.098612
Data	1.693147
Devic	2.098612
Exact	2.098612
Find	2.098612
Identifi	2.098612
Like	2.098612
List	2.098612
Locat	2.098612
Malwar	1.405465
Messag	2.098612
Misus	1.693147
Network	2.098612
Number	2.098612
Phone	2.098612
Read	2.098612
Record	2.098612
Send	2.098612
Server	2.098612
Tower	2.098612
Track	1.693147
Uniqu	2.098612
Usag	2.098612
User	1.405465
Various	2.098612
Wifi	2.098612

3.3.3. Task C: Topic Clustering. As stated in [3], clustering and factor analysis are the two analytic steps that are involved in post-LSA procedures. The authors discussed the main considerations that would let practitioners/decision-makers/researchers deploy these analytic steps as per their requirements. They focused on the fact that LSA has been used for clustering and factor analysis purposes. Based on the semantic space created in this study, the domain experts decided to pursue the clustering technique. The clustering approach was implemented through the *K*-means algorithm. Machine learning can be employed on top of results obtained after the application of Latent Semantic Analysis to significantly reduce the manual effort by a domain expert in determining the document to its closest topic. *K*-means is an unsupervised machine learning technique generally used when there are no labels of the data points and it learns them based on their relative positions in a vector space. The centroid feature weights may be used to identify the nature of the cluster while defining the groups, which may be used to label new data [75, 76]. *K*-means is easy to implement and can process extremely large samples [77]. Usually, the inputs into *K*-means are passed through a dimensionality reduction algorithm. LSA and *K*-means are applied in a linear combination for the interpretation of the results to find similar documents and their associations with the terms contained in the textual corpus [78–80], which is done to recommend research papers corresponding to a particular topic label. The interpretation of the results obtained is domain-specific. For instance, if data points were research articles on Android security in extensive literature, *K*-means will segregate the entire documents into *k* subgroups. The research trends in the domain of Android security which are a part of each subgroup or cluster have some common features, which are used for further analysis. The number of clusters was chosen to be three, with the selection done iteratively. It is to be noted that the choice of too few clusters may not reveal the actual underlying relationships, while too many clusters may account for noise, which would not be useful for any further analysis on the outputs obtained. The output, in the form of a multidimensional array, is composed of titles for all documents labeled with the respective cluster numbers. Taking the dot product of the components obtained from LSA with the cluster centroids, the results obtained are sorted to show only the top topics corresponding to each cluster, which require sensible topic labeling as discussed in the next section.

3.3.4. Task D: Topic Labeling. The term-to-topic and document-to-topic matrices consist of significant values to uncover topics. Each cell in both matrices represents the loading values which were later sorted in descending order. The results obtained from previous steps of TRENDMINER become the input for successful topic labeling. High-loading terms and documents were examined together and sensible labels were given against three and twenty-seven topic solutions, as shown in Figures 10 and 11. We have implemented the Delphi method [81] to perform the topic labeling process. The graphical representation of the Delphi method is also shown in Figure 12. Topic labeling is a collective intelligence task that involves the most reliable opinions of a group of experts. The Delphi method is an iterative method that worked under controlled monitoring and feedback mechanisms to build robust consensus.

3.4. Step 4: Results and Findings. As a result, three topic solutions present the major core research areas, as shown in Figures 13 and 14 along with the sensible topic label. Each topic solution is denoted as *Tm.n* where *m* denotes topic solution whereas *n* denotes an *n*th factor of the *m* topic solution. For instance, T27.3 illustrates the third factor of twenty-seven topic solutions. The graphical representations plotted for all point arrangements likewise give count about the publication distribution for every topic solution during three unique periods inside 2010–2019, as shown in Figures 13 and 15. The distribution check related with every subject arrangement addresses the significance of the comparing research region inside that theme arrangement. Furthermore, to reveal the examination patterns and future scope in the field of Android security, 27 point arrangements were found as depicted in Figures 15(a) and 15(b). The semantic relationship between 27 theme arrangements and three core research areas assists with recognizing research patterns inside each center exploration area of Android security, as depicted in Figures 10 and 11.

Table 6: Transformed term frequencies after TF-IDF generation.

Terms	Doc1	Doc2	Doc3	Doc4	Doc5
Access	0.000000	0.000000	0.000000	0.000000	0.589463
Applic	0.160859	0.164157	0.165134	0.176043	0.280882
Calendar	0.000000	0.344502	0.000000	0.000000	0.000000
Connect	0.000000	0.000000	0.346553	0.000000	0.000000
Contact	0.000000	0.344502	0.000000	0.000000	0.000000
Daili	0.000000	0.000000	0.000000	0.369447	0.000000
Data	0.000000	0.277942	0.000000	0.298067	0.000000
Devic	0.675160	0.000000	0.000000	0.000000	0.000000
Exact	0.000000	0.000000	0.346553	0.000000	0.000000
Find	0.000000	0.000000	0.346553	0.000000	0.000000
Identifi	0.337580	0.000000	0.000000	0.000000	0.000000
Like	0.000000	0.344502	0.000000	0.000000	0.000000
List	0.000000	0.344502	0.000000	0.000000	0.000000
Locat	0.000000	0.000000	0.346553	0.000000	0.000000
Malwar	0.226081	0.230717	0.000000	0.247423	0.000000
Messag	0.000000	0.000000	0.000000	0.000000	0.589463
Misus	0.000000	0.277942	0.000000	0.000000	0.475575
Network	0.000000	0.000000	0.346553	0.000000	0.000000
Number	0.000000	0.344502	0.000000	0.000000	0.000000
Phone	0.000000	0.344502	0.000000	0.000000	0.000000
Read	0.337580	0.000000	0.000000	0.000000	0.000000
Record	0.000000	0.000000	0.000000	0.369447	0.000000
Send	0.000000	0.000000	0.000000	0.369447	0.000000
Server	0.000000	0.000000	0.000000	0.369447	0.000000
Tower	0.000000	0.000000	0.346553	0.000000	0.000000
Track	0.272357	0.000000	0.279596	0.000000	0.000000
Uniqu	0.337580	0.000000	0.000000	0.000000	0.000000
Usag	0.000000	0.000000	0.000000	0.369447	0.000000
User	0.226081	0.230717	0.232090	0.000000	0.000000
Various	0.000000	0.000000	0.000000	0.369447	0.000000
Wifi	0.000000	0.000000	0.346553	0.000000	0.000000

Table 7: Term-loading with five latent topics.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Access	0.198118	−0.244194	−0.399978	0.007002	−0.317144
Applic	0.348381	−0.034393	−0.080514	−0.037085	−0.120670
Calendar	0.166381	−0.098917	−0.004802	0.024460	0.299730
Connect	0.107482	0.219059	−0.075849	−0.241415	0.008454
Contact	0.166381	−0.098917	−0.004802	0.024460	0.299730
Daily	0.128417	−0.108420	0.265857	−0.127129	−0.161347
Data	0.237841	−0.167279	0.210618	−0.082833	0.111647
Locat	0.107482	0.219059	−0.075849	−0.241415	0.008454
Malwar	0.284974	−0.031505	0.205508	0.104866	0.037136
Messag	0.198118	−0.244194	−0.399978	0.007002	−0.317144
Misus	0.294076	−0.276820	−0.326574	0.025384	−0.014050
Network	0.107482	0.219059	−0.075849	−0.241415	0.008454
Number	0.166381	−0.098917	−0.004802	0.024460	0.299730
Phone	0.166381	−0.098917	−0.004802	0.024460	0.299730
Read	0.130719	0.160295	0.045805	0.259253	−0.082933
Record	0.128417	−0.108420	0.265857	−0.127129	−0.161347

3.4.1. Task A: Identification of Core Research Areas in Android Security. Core research areas shown in Figure 13 were discovered as three topic solutions that focused on "Application Structure Analysis" (T3.1), "Static Level Monitoring" (T3.2), and "Automatic Malware Analysis" (T3.3). The word cloud for three topic solutions is shown in Figure 14. These articles emphasized imperative techniques to analyze, detect, and assess Android malware.

The outcomes showed that various high-stacking distributions joined to one exploration region, i.e., "Static Level Monitoring" (T3.2) in the three theme arrangements. Static investigation is the most used examination strategy for

Table 8: Document loading with five latent topics.

	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Doc1	0.542964	0.677204	0.434886	0.487392	0.471276
Doc2	0.491949	−0.297480	0.654891	−0.304044	−0.429196
Doc3	0.129873	−0.013342	−0.209489	0.688773	−0.649469
Doc4	0.640698	0.592341	−0.581166	−0.287077	0.919112
Doc5	−0.189251	0.670234	0.187921	−0.336431	−0.414465

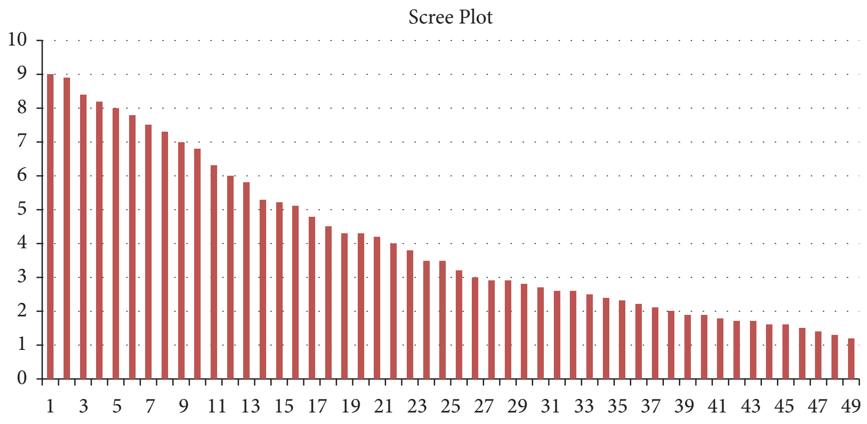


Figure 9: Scree plot.

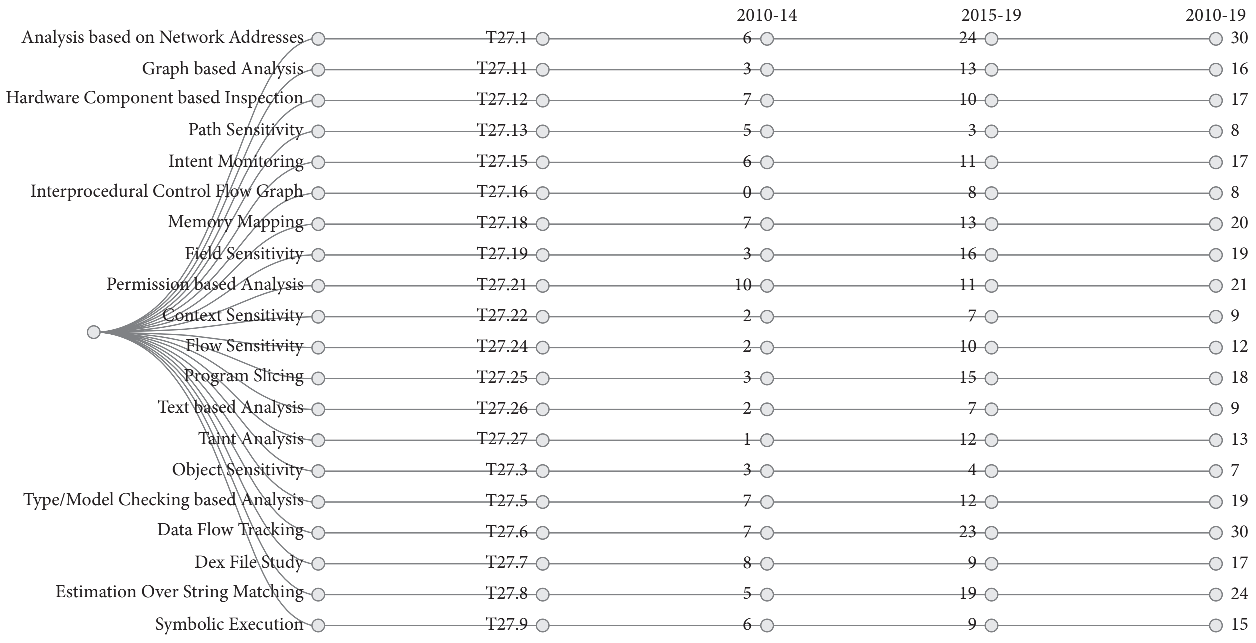

2010-14
2015-19
2010-19
Analysis based on Network Addresses T27.1 6 24 30
Graph based Analysis T27.11 3 13 16
Hardware Component based Inspection T27.12 7 10 17
Path Sensitivity T27.13 5 3 8
Intent Monitoring T27.15 6 11 17
Interprocedural Control Flow Graph T27.16 0 8 8
Memory Mapping T27.18 7 13 20
Field Sensitivity T27.19 3 16 19
Permission based Analysis T27.21 10 11 21
Context Sensitivity T27.22 2 7 9
Flow Sensitivity T27.24 2 10 12
Program Slicing T27.25 3 15 18
Text based Analysis T27.26 2 7 9
Taint Analysis T27.27 1 12 13
Object Sensitivity T27.3 3 4 7
Type/Model Checking based Analysis T27.5 7 12 19
Data Flow Tracking T27.6 7 23 30
Dex File Study T27.7 8 9 17
Estimation Over String Matching T27.8 5 19 24
Symbolic Execution T27.9 6 9 15

Figure 10: Mapping of core research area 3.2 and trends.

malware investigation; thus, it is obvious that “Static Level Monitoring” (T3.2) stayed in the moving exploration region over time 2010–2019. Results likewise showed that “Automatic Malware Analysis” (T3.3) additionally turned into a moving exploration region during the year 2015–2019. However, “Application Structure Analysis” (T3.1) had less impact on the set of papers collected in this study.

In the corpus, approaches dependent on Static Level Monitoring (T3.2) are the most well-known techniques (about 74%) utilized by the scientists to catch the security dangers in an Android system. Automatic Malware Analysis (T3.3) is around 20%, and Application Structure Analysis (T3.1) is 6%. First static analysis technique was introduced in 2009 [82] and in the year 2010, and dynamic analysis

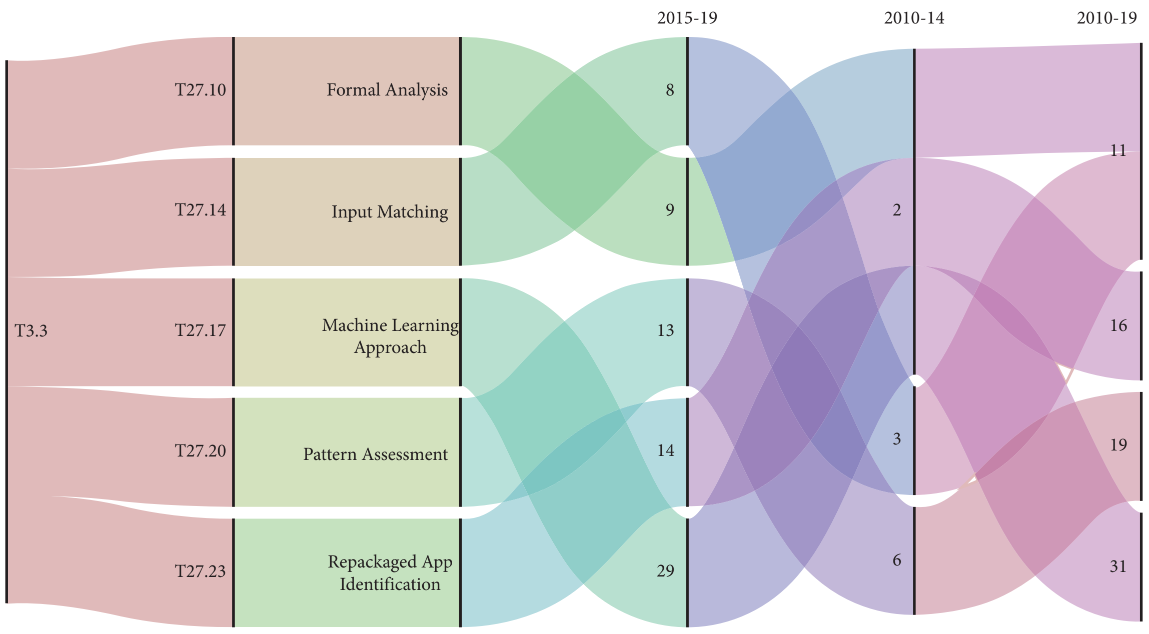


Figure 11: Mapping of core research area 3.3 and trends.

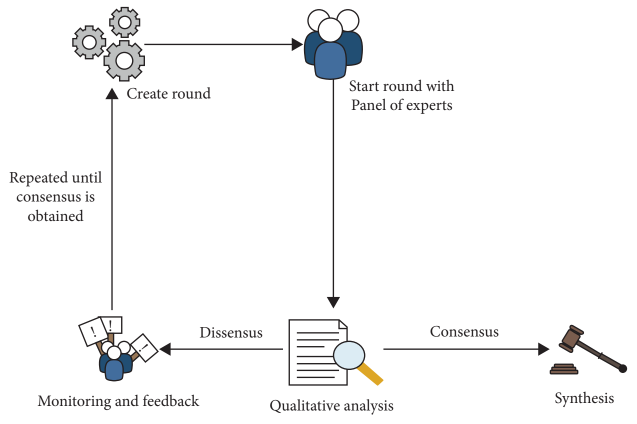


Figure 12: Working flow of the Delphi method.

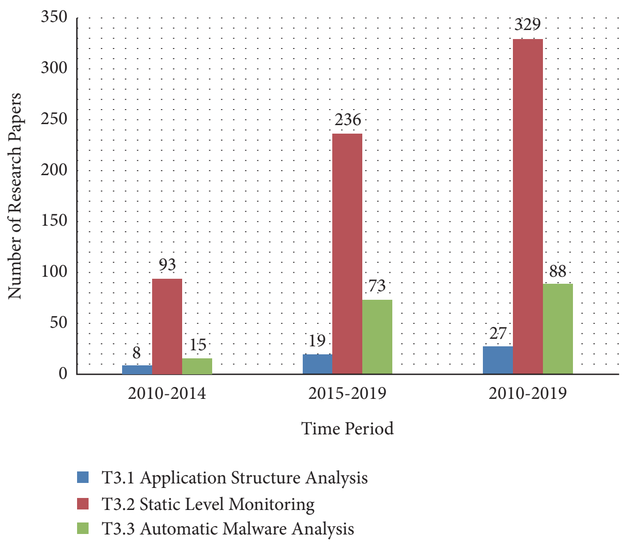


Figure 13: Publication count for three-factor solution during three different time periods.

technique was first explored by the researchers [83, 84]. The former investigated the data flows in applications that violate the security policies stored in an application's configurations. The latter identified the data leakage from sensitive sources of an application. Notwithstanding static and dynamic methodologies, there exist a couple of hybrid approaches that take advantage of the upsides of investigation such as static and dynamic. These techniques typically first apply static investigation to identify potential security threats in an Android system and after that perform dynamic procedures to enhance their accuracy by dispensing with the false alerts. For instance, in [85], the authors first used the static investigation to distinguish possibly vulnerable applications.

3.4.2. Task B: Identification of Android Security Research Trends and Task C: Core Research Areas and Trend Mapping. The TRENDMINER uncovered 27 subject core research trends as displayed in Figures 15(a) and 15(b). Figures 10 and 11 show the relationship of core areas with the research themes. The relationship is performed dependent on similarity scores. Documents were clustered into a lesser number of topic solutions as a start, while the higher value was chosen later. The points comparing to the last were to some degree identified with the previous and were checked utilizing similitude scores. The likeness scores were determined because of string coordinating, with the string similitudes indicating the closeness of the low and high upsides of theme arrangements. This was done to verify the understanding that the result while choosing a lower value of topic solutions would correspond somewhat to having chosen a comparatively higher value. The likeness scores present a reasonable connection between the core areas and their connected

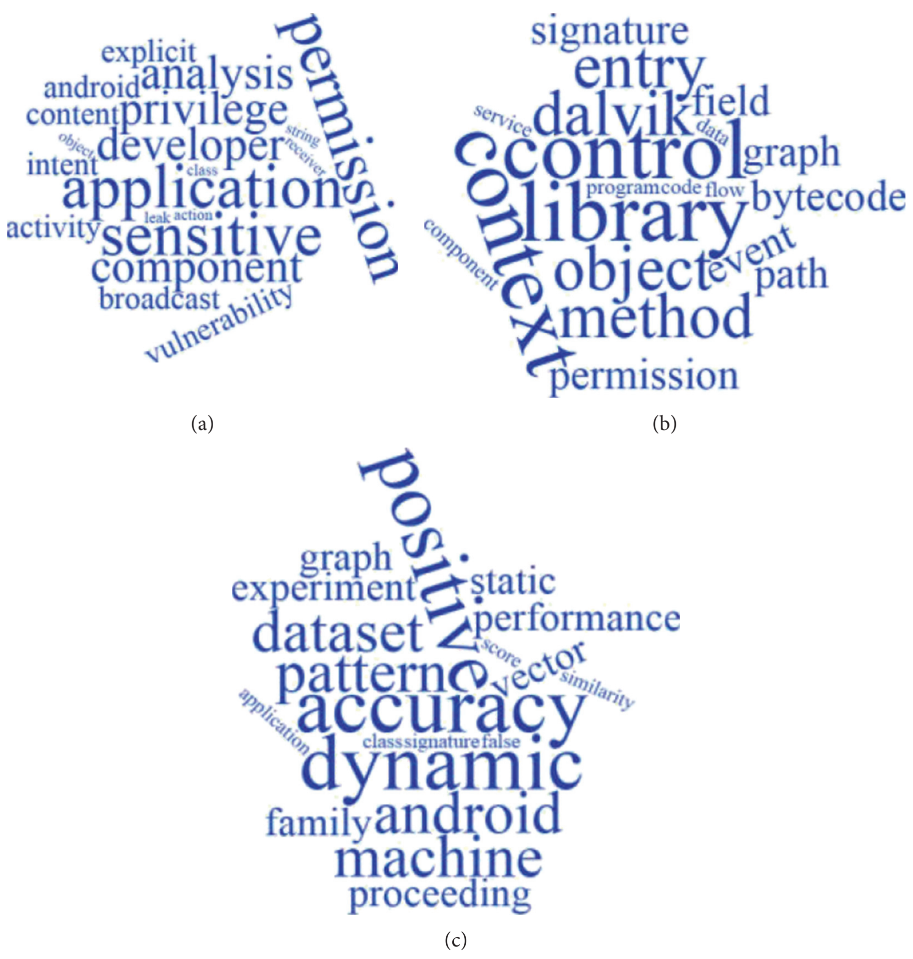


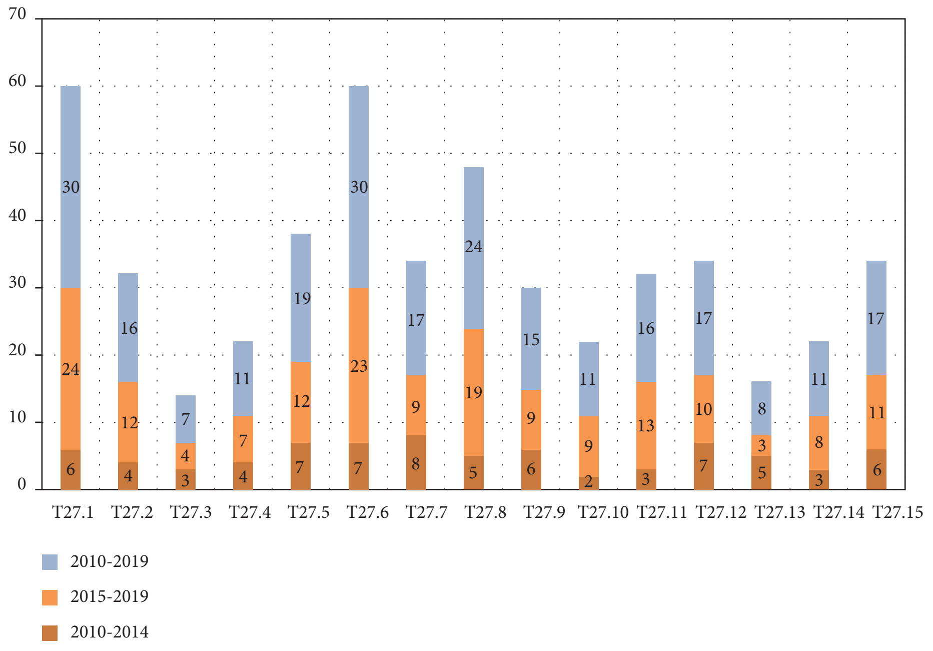
Figure 14: Word clouds generated by TRENDMINER for three topic solution (a)–(c). (a) Word cloud of topic solution 3.1. (b) Word cloud of topic solution 3.2. (c) Word cloud of topic solution 3.3.

patterns, which likewise approves the techniques created to show their semantic association.

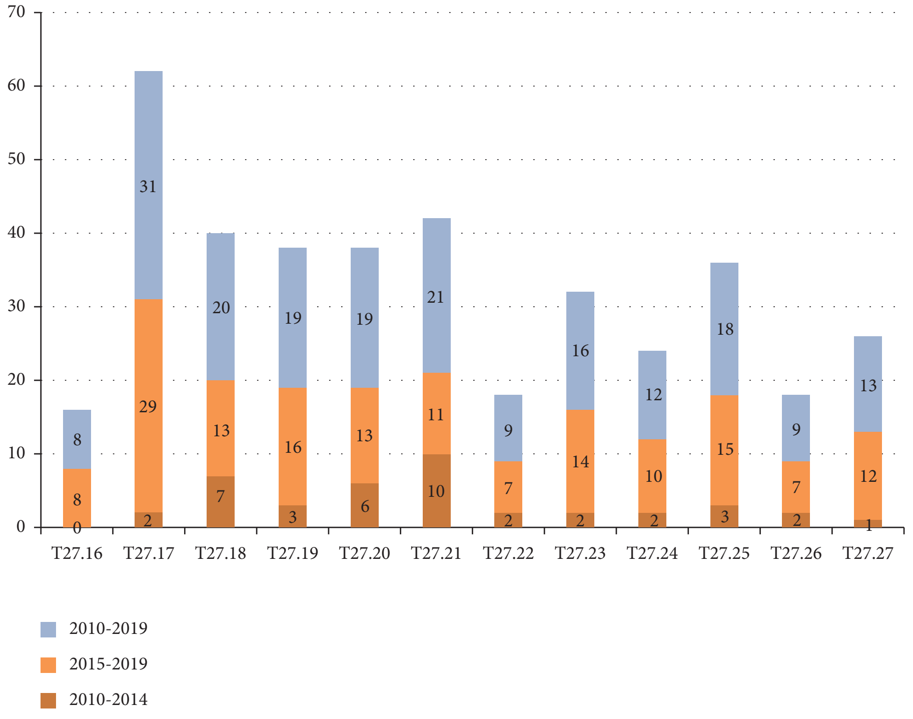
(1) Application Structure Analysis (T3.1). The trends Metadata-Based Study (T27.4) and App Level Features (T27.2) revealed the utilization of metadata. This pattern was found in the system named WHYPER [86], the researchers get to the permissions mentioned by applications' developers and utilized natural language processing (NLP) algorithms to search for sentences in application description that legitimizes the requirement for the mentioned permissions. Similarly, in another work, the study on metadata was accelerated by accounting additional information such as a number of application's screenshots, price, category, title, developer ID, website, and promotional videos. Furthermore, the analysis of application metadata was performed using machine learning algorithms. The trend Application Level features (27.2) unfolds the usage of CPU and memory usage to track malicious applications. In the project named MADAM, running processes, CPU utilization, memory state, Wi-Fi, and Bluetooth of the device were considered to train the k-nearest neighbor algorithm for effective detection [87].

(2) Static Level Monitoring (T3.2). It is the most investigated research area. Figure 10 demonstrates that out of twenty-seven topic solutions, twenty research trends such as Intent Monitoring (T27.15), Type and Model Checking-Based Analysis (T27.5), Memory Mapping (T27.18), Symbolic Execution (T27.9), Interprocedural Control Flow Graph (T27.16), Analysis Based on Network Addresses (T27.1), Program Slicing (T27.25), Context Sensitivity (T27.22), Text-Based Analysis (T27.26), Field Sensitivity (T27.19), Graph-Based Analysis (T27.11), Permission-Based Analysis (T27.21), Data Flow Tracking (T27.6), Dex File Study (T27.7), Object Sensitivity (T27.3), Flow Sensitivity (T27.24), Taint Analysis (T27.27), Hardware Component-Based Inspection (T27.12), Estimation over String Matching (T27.8), and Path Sensitivity (T27.13) mapped to T3.2.

In the topic solution Permission-Based Analysis (T27.21), authorizations played indispensable component for examination of vindictive applications, as most actions require explicit assents remembering the ultimate objective to be accomplished [88]. Permissions are declared in the manifest file and therefore, easy to obtain. Numerous systems, developed in studies [86, 89, 90], use static examination to evaluate the risks of the Android consent system and individual applications.



(a)



(b)

Figure 15: Twenty-seven factor solution during three different time periods (a) and (b).

Another significant research trend emerged as Analysis Based on Network Addresses (T27.1), focused on network addresses. Malware authors make use of network addresses to build communication with command and control (C&C) worker to send the client's classified information. Analysts discovered IP addresses as one of the key static components for investigation [91–93].

Another examination pattern that arose in this space is the Dex record study (T27.7), which played a vital role in understanding the dex files, which are usually cumbersome to interpret by humans. To recognize malevolent code sections, scientists first decompile the dex code into more possible organizations such as gathering, Smali, Dalvik bytecode, source code, container, Jimple, or Java bytecode [94]. This trend can be further relate to numerous articles and tools deployed by researchers for successful translation such as dexdump [95], Pegasus [96], ded [97], SAAF [98], PScout [89], AppSealer [99], ded/DARE [100], dedexer [90], dex2jar [101], and FlowDroid [102].

The core research area discovered interesting research trends such as Data Flow Tracking (27.6), Interprocedural Control Flow Graph (27.16), and Graph-Based Analysis (27.11). All emerged trends relate to an interesting and pivotal branch in the field of static security mechanisms to identify commandeering vulnerabilities in the Android ecosystem. Data Flow Tracking (T27.6), which deals with tracing out the flow of sensitive information from the device to outside entities at the time application execution [103–107], came out as important and consistent topic. Information stream examination and control stream investigation help in understanding the hazardous usefulness such as protection spillage and communication administrations abuse [95, 108, 109] by tracking the flow of information across different points of execution.

Bytecode control-flow graph investigation recognizes all possible ways that an application can take while it is executed. These deduced trends helped in fostering advance investigation, by creating control flow bytecode graph (CFG) for intraprocedural analysis or between procedural investigation (crossing across various strategies). Creators in [110] formalized the Dalvik bytecode to play out the control stream investigation-based semantic marks to recognize malware applications. The studies [89, 95, 96, 102, 104, 108, 111] leverage the trends Data Flow Tracking (27.6), Interprocedural Control Flow Graph (27.16), and Graph-Based Analysis (27.11).

The trend of Intent Monitoring (T27.15) relates to the concept that intents declared in the application's manifest file are capable enough to leak the data to C&C servers. Intents are the objects which are used to move from one activity to another by making use of widgets in an Android application. Starting an activity, starting a service, and delivering a broadcast are the three fundamental use cases of intents, helps in establishing the communication between components in several ways. This trend was found in popular studies [91, 112]. The former employed numerous machine learning algorithms such as K-means, k-nearest neighbor, and naïve Bayes to analyze the intents, permissions, components, and APIs that were extracted from the manifest file. The latter employed support vector machines to detect malware and achieve a detection rate of 94%. Another trend Hardware Component-Based Inspection (T27.12) reflects the analysis of hardware components listed in an application for static investigation. Researchers in [91] made use of the components declared in the manifest file for analysis. This can be compelling as malicious applications with a specific end-goal demand all the hardware, e.g., camera, GPS, and microphone.

Estimation over String Matching (T27.8) is found as another significant trend in this area, which uncovered the analysis over various strings available in an Android application. Work done by researchers in [113] expressed that it is one of the broadly utilized strategies for recognizing the malware through analyzing the strings, accessible in the Android files. Scientists utilized the Vector Space Model (VSM) [114] and addressed the strings as vectors in a multidimensional space. Besides, scientists utilized distance estimates such as Manhattan distance, Euclidean distance, and Cosine similarity to learn irregularity of the data. The researchers assessed the outcomes over 666 samples of Android applications and accomplished 83.51% accuracy in their tests.

(3) Automatic Malware Analysis (T3.3). Figure 11 demonstrates research trends under T3.3. This core research explored the research trends Pattern Assessment (T27.20), Input Matching (T27.14), Repackaged App Identification (T27.23), Formal Analysis (T27.10), and Machine Learning Approach (T27.17) which were related to automation in identifying Android malware. To gather a predefined set of application features, researchers focus first to analyze application statically or dynamically. Furthermore, build a detection model capable of distinguishing malware and benign applications based on the training dataset. The trend proved as well explored and promising as researchers used numerous combination of different features such as API call sequences, permission request, package information, hardware components, application categories, and network activity to build detection models, as reported in studies [91, 115–118]. Another exploration pattern that arose was Repackaged App Identification (T27.23). Many articles such as [119] related to this trend were published in recent years. DroidMoss [88], Droidsim [120], DNADroid [121], ViewDroid [122], ResDroid [123], and AnDarwin [124] have witnessed to tame the problem of repackaging.

The trend Pattern Assessment (T27.20) uncovered the fact that an attacker can deduce sensitive information of the user by accessing the behavioral pattern of shared resources. The impact of this trend has been seen in a variety of articles [125–129] where side channel communication was compromised to infer confidential input patterns such as PIN, password, or screen taps.

4. Discussion and Potential Future Directions

This section determines that the results obtained from TRENDMINER can be used to answer the research questions stated in Section 1.

4.1. RQ1: Can the Proposed Framework Uncover Leading Researchers within a Research Domain? Figures 7 and 8 present the top journals and leading researchers in the Android security field. Some of the top journal lists include Computer and Security, IEEE Transaction on Information Forensic and Security, Future Generation Computer System, Journal of Information Security and Applications, and Journal of Networks and Computer Applications. Suarez Tangil has a major contribution in the research community who has framed a variety of antimalware techniques such as Alterdroid [130], Dendroid [131], and Droidsieve [132]. A fully automated malware identification mechanism with an appreciable accuracy of 82.93% has been framed by Wang et al. [133]. Enck et al., who proposed a project named Taintdroid [83], are top leading researchers in this field. He had developed an effective model for tracking sensitive information leakage in third-party applications. On top of this, many other dynamic analysis tools such as Andrubis [134] and Droidbox [135] were deployed. He was first to perform on-device malware assessment in which authors defined a set of rules to identify dangerous permissions granted before installing the application, by the security service known as Kirin [136]. To detect kernel-level attacks, Yan and Yin presented a project named Droidscope [137], which is a unique method of dynamic analysis by keeping its process out of emulator and was able to achieve promising results. Faruki et al. [138] proposed a methodology called Androsimilar which produces marks by extricating measurably powerful components, to identify noxious Android applications. Proposed strategy was powerful against code jumbling and repackaging methods that will in general engender concealed variations of known malware by avoiding AV signatures.

4.2. RQ2: Are Those Frameworks Robust Enough to Determine the Most Investigated Research Areas? The consequences of the examination showed that Static Level Monitoring (T3.2) had been end up being the most generally researched point in Android malware investigation and location. The strategies utilized under Static Level Monitoring (T3.2) analyses the code without running the application on an Android emulator or gadget. The upside of static investigation is that the expense of calculation is low, less dreary, and low asset use. Figure 16 reveals that most of the trends tend to fall in the topic solution Static Level Monitoring (T3.2). Out of the 20 research trends, eleven such trends have shown a significant rise in time frame 2 (2015–2019) than time frame 1 (2010–2014). The rate of change varies from 0.82% to 4.01%. Nine such trends showed a downfall in time frame 2. Examination under this work uncovered that studies identified with static level observing significantly center around network addresses, information stream, control stream, string coordinating, consents, dex documents, setting, and purposes.

Static level monitoring emerged as an important technique to accomplish various security concerns such as detecting private data leaks, detecting component hijacking or intent injection, building frameworks for intercomponent vulnerabilities and content provider-based vulnerabilities, dangerous permissions used by malicious applications, energy consumption concerns by Android applications, comparing Android applications for clone detection, automatic testing by generating test cases, and checking the correctness of the Android application through code verification. On further investigation, it was found that there are various tools available for static monitoring, such as Soot, Dex2jar, Dexdump, Dedexer, Ded, Dare, and WALA. Soot is the most adopted support tool for static monitoring, and Jimple is the widely used intermediate representation (IR) format for the further analysis of Android applications. The trend line in Figure 16 illustrates that specific research trends orient towards sensitivities. Sensitivities maximize the precision and recall of static monitoring. The research trends Field Sensitivity (T27.19), Context Sensitivity (T27.22), and Flow Sensitivity (T27.24) are primarily taken into account by the Android research community. Other research trends, such as Path Sensitivity (T27.13) and Object Sensitivity (T27.3), have not gained much attention from the researchers. The trend line also revealed that the trend Taint Analysis (T27.27) widely used in data tracking emerged as the most applied technique in static monitoring.

4.3. RQ3: Would the Proposed Framework Reveal How the Focus of Topics within Each Core Research Area Has Changed over Time? In this study, two time frames 2010–2014 and 2015–2019 were used to maintain valid interpretations and comparisons among the topics. Table 9 shows the focus of major topics within each core research area changed over time. It depicts the paradigm shift from the time window from 2010–2014 to 2015–2019. The study made the following observations:

(a) Machine learning approaches are demonstrated to be compelling among other serious methodologies in the location of Android malware. These methodologies are all around investigated and promising during the period 2015–2019.

(b) Detection of piggybacked applications utilized delicate chart investigation/information followed by use of AI calculations, during the period 2015–2019.

(c) Permissions have discovered quite possibly the most utilized static elements to identify Android malware application. The trend was popular during 2010–2019. Some specific permissions are declared in the manifest file to activate certain events in an Android ecosystem.

(d) Static analysis is largely performed by the researchers to address security and privacy issues, due to its ease of implementation. However, static analysis is vulnerable to stealthy techniques such as encryption and native code, resulting in a downfall in the usage of pure static solutions. Nevertheless, it is still popular in the research community.

(e) Taint analysis is a widely applied technique in publications. It is the kind of information flow analysis in which objects are tainted and tracked using data flow analysis.

(f) During 2015–2019, one research trend emerged as "Analysis Based on Network Addresses" (T27.1), focused on network addresses. Malware authors make use of network addresses to build communication with the command and control (C&C) server to send the user's personal confidential data. Researchers found network addresses as one of the key static features for analysis.

(g) The trend "Text-Based Analysis" (T27.26) relies on extracting the critical phrases and keywords, for example, sensitive APIs and permission for the

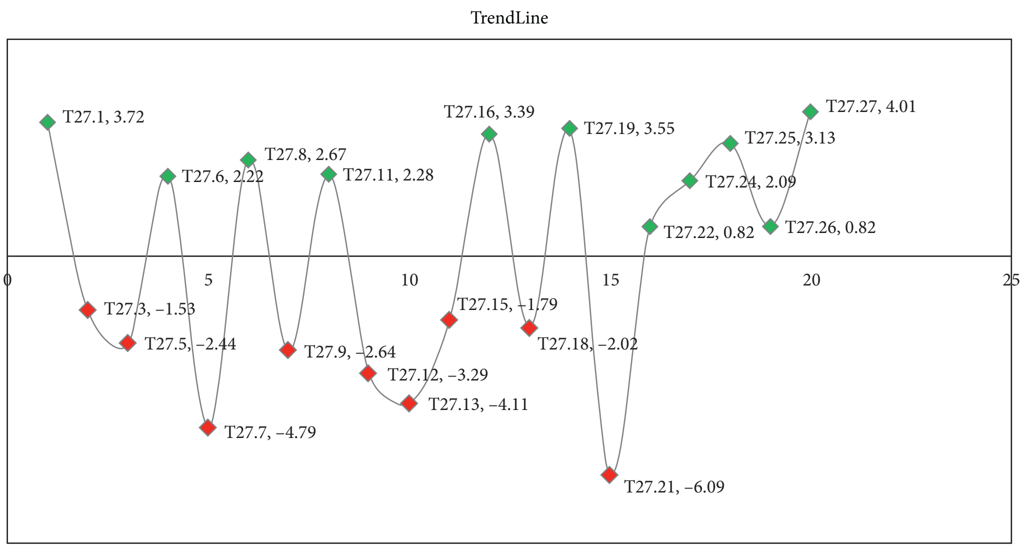


Figure 16: Impact of research trends during time frame 2015–2019.

Table 9: Focus of topics changed over time from 2010 to 2019.

Topic no.	Label	2010–2014	Impact in time frame 1 (%)	2015–2019	Impact in time frame 2 (%)	+/−	
T27.1	Analysis Based on Network Addresses	6	6.45	24	10.17	+3.72	▲
T27.3	Object Sensitivity	3	3.23	4	1.69	−1.53	▼
T27.5	Type and Model Checking-Based Analysis	7	7.53	12	5.08	−2.44	▼
T27.6	Data Flow Tracking	7	7.53	23	9.75	+2.22	▲
T27.7	Dex File Study	8	8.60	9	3.81	−4.79	▼
T27.8	Estimation over String Matching	5	5.38	19	8.05	+2.67	▲
T27.9	Symbolic Execution	6	6.45	9	3.81	−2.64	▼
T27.11	Graph-Based Analysis	3	3.23	13	5.51	+2.28	▲
T27.12	Hardware Component-Based Inspection	7	7.53	10	4.24	−3.29	▼
T27.13	Path Sensitivity	5	5.38	3	1.27	−4.11	▼
T27.15	Intent Monitoring	6	6.45	11	4.66	−1.79	▼
T27.16	Interprocedural Control Flow Graph	0	0.00	8	3.39	+3.39	▲
T27.18	Memory Mapping	7	7.53	13	5.51	−2.02	▼
T27.19	Field Sensitivity	3	3.23	16	6.78	+3.55	▲
T27.21	Permission-Based Analysis	10	10.75	11	4.66	−6.09	▼
T27.22	Context Sensitivity	2	2.15	7	2.97	+0.82	▲
T27.24	Flow Sensitivity	2	2.15	10	4.24	+2.09	▲
T27.25	Program Slicing	3	3.23	15	6.36	+3.13	▲
T27.26	Text-Based Analysis	2	2.15	7	2.97	+0.82	▲
T27.27	Taint Analysis	1	1.08	12	5.08	+4.01	▲
T27.10	Formal Analysis	2	13.33	9	12.33	−1.00	▼
T27.14	Input Matching	3	20	8	10.96	−9.04	▼
T27.17	Machine Learning Approach	2	13.33	29	39.73	+26.39	▲
T27.20	Pattern Assessment	6	40	13	17.81	−22.19	▼
T27.23	Repackaged App Identification	2	13.33	14	19.18	+5.84	▲
T27.2	App Level Features	4	50	12	63.16	13.16	▲
T27.4	Metadata-Based Study	4	50	7	36.84	−13.16	▼

analysis. This trend became popular during the time frame of 2015–2019.

(h) The trend "Symbolic Execution" (T27.9) showed a downfall in the time frame 2015–2019. It deals with generating all possible program inputs to explore all conditional branches inside the path. This process could be time-consuming and hence became less popular among the research communities during 2015–2019.

(i) Another research trend that emerged was "Repackaged App Identification" (T27.23). Repackaging is one of the popular techniques being employed by malware authors to generate fraudulent repackaged applications. Many articles related to this trend were published during the time frame 2015–2019.

(j) The trend "Metadata-Based Study" (T27.2) uncovered the utilization of metadata to identify and dissect Android malware applications. Metadata involves required authorizations, depiction, form, last refreshed, rating, number of establishments, and engineer data. This pattern encounters a defeat during 2015–2019.

(k) Table 9 revealed that the trend "Program Slicing" (T27.25) had gained momentum during 2015–2019. The trend "Program Slicing" (T27.25) specifies the technique by focusing on selected aspects of semantics for simplifying the programs. Slicing avoids those parts of the program that may not have caused the malicious behavior, instead focus attention on only those parts of programs that may contain malicious behavior. This technique tends to reduce the set of program behavior and hence became trending during 2015–2019.

(l) The trend "Field Sensitivity" (27.19) appears to be the most considered among all the sensitivities, depicted in Table 9. It may be due to the reason since Android apps are written in Java, an Object-Oriented language where object fields are pervasively used to hold data. Research trends such as "Context Sensitivity" (T27.22) and "Flow Sensitivity" (T27.24) are also largely taken into account. The least considered sensitivity is "Path Sensitivity" (T27.13) and Object Sensitivity (T27.3); probably, it is because of the scalability issues that it raises.

(m) The trends "Type and Model Checking-Based Analysis" (T27.5) showed a sudden fall during 2015–2019. When an Android application is developed for some task, it is common to define a certain set of properties that the application must satisfy. Model checking helps to ensure that the given system has met given specification or correctness properties. Type checking ensures that the given program is type-safe by keeping the possibility of type errors (e.g., applying integer operations on float numbers) to a minimum.

(n) Another research trend that emerged was the "Dex File Study" (T27.7), which played a vital role in understanding the dex files was popular during the time frame 2010–2014. Dex code is usually cumbersome to interpret by humans and therefore shows a downfall during 2015–2019.

(o) In the research trend "Permission-Based Analysis" (T27.21), permissions are declared in the manifest file and, therefore, easy to obtain, and that could be the reason for its popularity among researchers during 2010–2014. However, examining only permissions is not useful in detecting malicious applications. Therefore, this trend experiences a downfall during 2015–2019.

(p) Interesting research trends such as "Data Flow Tracking" (27.6), "Interprocedural Control Flow Graph" (27.16), and "Graph-Based Analysis" (27.11) are the data structures for the analysis. Data flow analysis and control flow analysis help in understanding unsafe functionality such as privacy spillage and telephony services misuse by tracking the flow of information across different points of execution. The advantage is that bytecode control-flow graph investigation recognizes all possible ways that an application can take while it is executed and hence popular during 2015–2019.

(q) The trend of "Intent Monitoring" (T27.15) relates to the concept that intents declared in the application's manifest file are capable enough to leak the data to C&C servers. Intents are the objects which are used to move from one activity to another by making use of widgets in an Android application. Starting an activity, starting a service, and delivering a broadcast are the three fundamental use cases of intents, which helps in establishing the communication between components in several ways. It was more popular in the time frame 2010–2014 than the time frame 2015–2019.

(r) Another trend, "Hardware Component-Based Inspection" (T27.12), reflects the analysis of hardware components listed in an application for static investigation. It can be compelling as malicious applications with a specific end-goal demand all the hardware, e.g., camera, GPS, and microphone. This trend gradually decreases in the time frame of 2015–2019.

(s) Another significant trend, "Estimation over String Matching" (T27.8), uncovered the analysis over various strings available in an Android application. Its impact is slightly more during 2015–2019.

(t) The trend "Application Level Features" (27.4) unfolds the usage of CPU and memory usage to track malicious applications. It remains trending during 2015–2019.

4.4. RQ4: Can It Unfold the Future Directions within the Research Domain of Choice? Many obstacles are set forth by Android malware, which needs to be carefully addressed after being thoroughly observed. Based on the results of

TRENDMINER, with no doubt, it is evident that Android security has gotten a ton of consideration in recently published literature. Perhaps, it is mainly due to the ubiquity of Android as a famous operating system in the community. Significant patterns are observed in the previous decade, as reflected by the aftereffects of this writing survey. Hence, based on the results of the TRENDMINER, a few recommendations are made that are discussed as follows:

(a) Mapping of API usage with permissions to achieve more fine-grained results: API calls are used to communicate and transfer sensitive information over the network. Malware families such as Fakeinst, Opfake, and Smsreg make use of API calls such as sendSMS() and readSMS(), which implies that collected information may be sent by SMS. There is an urgent need to deeply analyze the API calling patterns and what permissions these APIs demand [139].

(b) Complications in static analysis: static analysis techniques are incapable when applications are made using camouflage techniques [39, 139–143]. Static analysis also leads to a large number of false positives [7, 144].

(c) Evolution of intelligent malware: applications tend to use techniques such as rooting, antidebugging, code obfuscation, and kernel-level features to dodge the detection process [145, 146]. Despite this, most of the approaches still implement emulators. Limited efforts are made to curtail remote triggering. It enhances the stealthiness of malware by allowing malware authors to trigger and execute malware whenever they want [147].

(d) Development of nonintuitive features for robust malware analysis and detection: static and dynamic features need to be explored to the next level to characterize the behavior of an application [146] better. Attackers repackage the legitimate app to insert the malicious snippet and distribute it via stores [88].

(e) Need of automation in malware classification: development of semisupervised approaches to detect the malicious applications [146, 148] and faster detection and classification of malware families is required [141]. Also, the features and characteristics of a family that can be used to classify malware to a particular family have been less discussed among the research communities [7].

(f) Hindering the effectiveness of dynamic analysis: computation time and resource constraints are the major reasons for the hindered performance of dynamic analysis [7, 39, 140, 143]. To ensure that an application had triggered all its malicious behavior (all execution paths traversed) during dynamic analysis is a matter of concern [141, 142, 144].

(g) Limited availability of datasets: limited availability of ransomware datasets and lack of understanding of smart tactics limits the efficacy of detection mechanisms [149]. Generally, researchers download the samples from VirusTotal [150].

(h) Low-precision prediction mechanisms: the biggest challenge being faced by researchers is the high rate of false alarms in predicting ransomware. Most of the present techniques produce a large amount of false positive and false negative alarms, which affects the accuracy of detection mechanisms. There is a need for a cutting edge methodology to produce fewer false alarms [149].

The study uncovered that methodologies of examining malware incorporate static examination and dynamic investigation or perhaps a blend of both. The static examination essentially centers around dismantling the code, trailed by manual examination to look for the pernicious examples in the code. On the other hand, dynamic investigation executes the code in the virtual platform and breaks down its execution follow to notice the noxious conduct of an application. The static examination helps follow unique and full execution ways; subsequently, it gives total code inclusion; however, at last it experiences code obscurity. The application must be decoded first to perform static investigation. The issues of obstinate intricacy ruin the examination. Dynamic examination is more productive and need not bother with the executable to be unloaded or unscrambled. The dubious application is checked in a controlled arrangements. This cycle is time and asset devouring. It additionally raises adaptability issues. Besides, some malevolent conduct may be unseen on the grounds that the environment does not fulfill the setting off conditions. Besides, malware creators utilize mechanization innovation to produce a colossal measure of new malware variations, accordingly representing a major test to malware experts. The current situation with the-workmanship requests the combination of existing crude strategies with valuable methods to accomplish a powerful arrangement. The yield of TRENDMINER proposes that strengthening strategies ought to be utilized to supplement the arrangement of quickly developing Android malware families. Beneficial methods can end up being viable in deciding strange current vindictive conduct or security weaknesses. In view of the assortment of information got by this investigation, a plan for designing a cutting edge environment has been imagined for the characterization of Android malware families, as examined in the next section.

5. Towards Engineering a Visualization-Based Solution

Malware is developing quickly which is a result of the ability of malware creators to change little pieces of the first source code to produce new malware variations. A malware variation can be imagined as a grayscale image. A picture can catch even little changes. Thus, in the current work, a perception structure is proposed to decrease the impact of obscurity by changing the malware's noninstinctive components into unique finger impression images followed by the arrangement of Android malware families. The proposed methodology which is known as the SWAYAM (Stop WAY for Android Malware) system is shown in Figure 17.

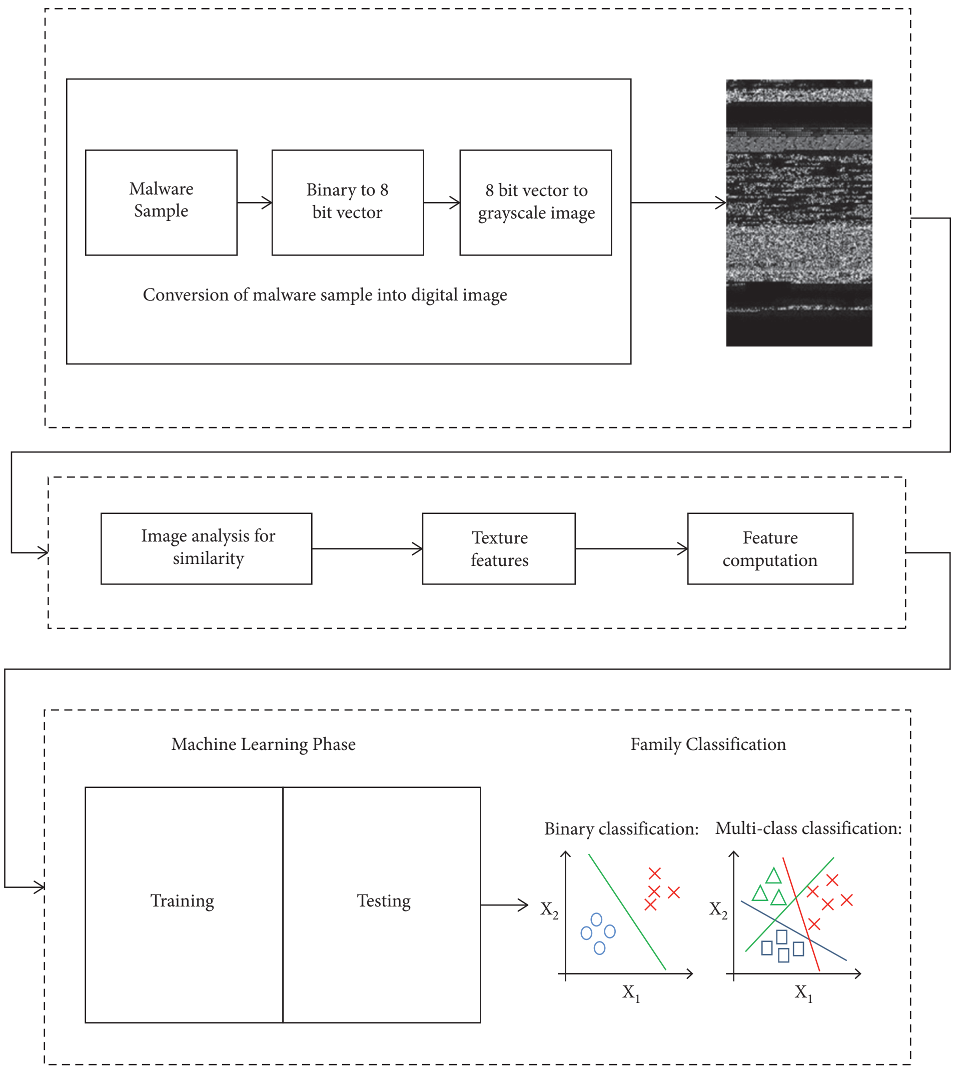


Figure 17: The proposed SWAYAM (Stop WAY for Android Malware) system.

5.1. Module I. This module deals with converting the malware samples into digital images. The malware binaries are first converted into 8-bit vectors and then converted into grayscale images. The overall structure of grayscale images is composed of various sections. Each section has a fixed width, but height is varied according to the file size. In a nutshell, malware samples tend to be represented as images and there is a strong propensity that malware variants from the same family form similar and visual implications [151]. On the other hand, malware samples from different families show dissimilar structural and visual implications.

5.2. Module II. Once the images are converted into digital images, the next step is to extract the features out of the images. Features play a vital role in classifying malware samples to a particular family. Various image descriptors such as Global Image deScripTors (GIST), Gray Level Co-occurrence Matrix-based (GLCM), and Local Binary Pattern (LBP) are available to extract the features from the images and thus formed a feature vector. Texture patterns, intensity, color patterns, and frequencies in images constitute the image features of the samples. Euclidean distance or standard deviation can be used to measure the distance in feature space [152].

5.3. Module III. Further machine learning algorithms or neural networks are employed over feature vectors to identify the family of a sample. For instance, in the KNN approach, a sample is classified to family f1 if it has *k*-nearest neighbors belonging to family f1. It is to be noted that many solutions leveraging machine learning and big data techniques are appearing to develop malware detection models [153–155]. Computer vision techniques have been becoming popular among the research communities to detect and classify malware applications [156, 157].

6. Limitation of the Study

This study encountered a few issues that may have arisen during the collection of the literature dataset on Android security. It depends upon certain factors, for example, the type of queries and sources used while preparing the literature dataset. To discover the appropriate publications, the articles were selected using "malware" OR "vulnerability" OR "security" OR "privacy" OR "monitoring" OR "application" OR "smartphone" OR "android" OR "virus" OR "static" OR "dynamic" OR "detection" OR "data flow" as search keywords. The prominent databases which were leftover during the automated search were also browsed to get the influenced publication in the area. Relevant papers were filtered using inclusion and exclusion criteria on the search results to limit the purpose of the current study. Nonetheless, it may be possible that a few significant publications may have been left during the process.

TRENDMINER is backed by the goodness of the Latent Semantic Analysis (LSA) technique. LSA being an unsupervised way of uncovering synonyms improves the vector space model. However, the number of topic solutions cannot be decided statistically. To alleviate this situation, the value for an optimal number of topic solutions was decided after having intensive discussions with an expert. Ultimately, this work deduced that the process of topic labeling was purely based on human judgment, which may lead to subjective bias as well.

There might be impediments identified with the speculation of the outcomes. A stepwise procedure was followed to infer the core research areas and research trends. The procedure included literature collection, preprocessing of the dataset, generation of TF-IDF matrix, truncated SVD, and topic labeling. Every step in the algorithm tends to influence the results. For instance, the outcomes will be influenced if the dataset used in this study is modified to a composition of only titles or full-length articles.

Having done LSA representation of some documents, a new document cannot be just added to this collection. A new document, hence, can only be added incrementally. It fails to capture the elements of the new documents added. Hence, the performance of LSA degrades on the addition of new documents, allowing recomputation.

7. Conclusion

One of the key inspirations of the work was that the conventional manual literature reviews are often not ready to exploit huge literature because of human obstructions in time and insight. Hence, this study proposed another literature review method to deal with this challenge. This study unveiled a framework called the SEAR framework, which can perform subjective and quantitative investigation over enormous literature. It is an adaptable and versatile framework to draw information-driven investigation and conceptualize the advancement of inclining research measurements in any field of literature. The SEAR framework utilizes the linear combination of information modeling technique, i.e., LSA followed by the K-means clustering algorithm, which enables connections and groupings to be recognized that are usually missed by manual techniques constituting human interpretations. Machine learning techniques have reduced the manual effort to a great extent in determining the document to its closest topic.

TRENDMINER is designed as the use case of the SEAR framework. To exhibit the utility and use of TRENDMINER, a wide body of literature on the Android security field was utilized as the contextual investigation. The framework takes the contribution of 444 abstracts of research articles distributed during the period 2010–2019. This study identifies three core research areas and twenty-seven research trends as outcomes. Results demonstrated that specific research patterns have stayed reliable over the examined time frame. Taxonomy and future research directions in the field of Android security have been provided in this study. Time trend plots for each factor solution have been discussed. Some research trends have developed while a couple has likewise declined. TRENDMINER amplifies the utility and commitment by proposing potential future research directions in developmental research to mitigate human predispositions. This study also stresses answering the research questions framed with respect to the technique being employed and the dataset chosen. This paper additionally exhibited general suggestions to help new researchers to comprehend the idea of Android security research and assess their regions of interest for their latent capacity research alongside the related research pattern.

This examination additionally sets up an objective and observational establishment for future directions about the structure and analytical decomposition of Android security research. The particular research area and trends uncovered in this work can engage future research dimensions, which can be utilized by the research scientists and industry. Furthermore, researchers can pick at least one research area and make another investigation with the equivalent or another approach. Nonetheless, other factual factor investigation strategies can apply to this exploration. For future work, the researchers can apply a similar technique to a different comparable dataset to see the proclivity and decent variety of core research areas and trends inside related articles. To increase the application areas of this research, the SEAR framework can be enhanced by building a dynamic query system on the same or different corpus by applying deep learning models.

8. Practical Implications and Future Research Directions

This manuscript exhibits a panoramic view of the Android security field. The study has certain interesting practical implications. First, the research areas and trends uncovered in this work can engage future research dimensions, which can be utilized by the new research scientists and industry. The analysis obtained from the study can assist them to understand the diversity and depth of the Android security field. Second, the academic universities can enhance their teaching content and students' motivation by revising the curriculum to focus more on research activities related to the Android security field.

Third, perspectives drawn from the research will help the editors of the esteemed journals to plan the special sessions on Android malware research topics such as static analysis of Android applications, security and privacy for IoT and multimedia devices, application-focused threats, new frontiers in Android malware analysis and detection, cryptojacking, component-based Android malware analysis, deep learning for Android malware classification, deep learning for digital forensics, and cybersecurity. There are avenues for future research which are discussed as follows.

8.1. Ranking Permissions for Android Malware Analysis and Detection. Using too many features for Android malware analysis and detection is a cumbersome task. Permissions as a special feature of the Android ecosystem are present in the manifest.xml file of the Android file structure. Permissions are needed to perform the application-sensitive operations. They are embedded in the manifest.xml file in the form of text. They play a vital role in detecting the suspicious application running on an Android device. Some permissions which malware authors use to exploit the sensitive information from the device are *access_coarse_location*, *access_ ne_location*, *access_network_state*, *access_wi _state*, *battery_stats*, *answer_phone_calls*, *bind_carrier_messaging_service*, *read_contacts*, *read_call_log*, *read_phone_state*, *read_external_storage*, *read_sms*, *record_audio*, *request_install_packages*, *read_calendar*, *bluetooth_privileged*, *read_history_bookmarks*, and many more. The most important permissions in the malicious dataset can be identified using a technique called Term Frequency Inverse Document Frequency (TF-IDF) which can later lead to the discovery of malicious applications. It would help in maintaining an application-permission matrix that would describe the frequency of permissions that occur in the collection of malicious applications. TF-IDF assigns the permission value to each permission and calculates the sensitive value of each application by utilizing its weighing formula as discussed in this work. Furthermore, machine learning algorithms may be deployed to perform the detection or classification of Android malware applications.

8.2. Crowdsourced User Reviews at Application Stores. The suspicious application can also be identified by evaluating the user reviews at the application stores. The feedbacks of the users are vital as they tend to write reviews about the particular application based on their real-time usage and experience. The security firms cannot ignore the reviews whether they are positive or negative. The user reviews are expressed for various purposes such as functionality, UI (user interface)/design, battery consumption report, and other security issues of an application. Furthermore, the security issues in the application are broadly classified into four categories: malware code injected into the application for monetary benefits, spamming, information leakage, and use of overprivileged permissions in the application. Latent Semantic Analysis can be applied to crowdsourced user reviews to discover security-related issues of the application. At the initial step, relevant reviews can be filtered out from the noisy crowdsourced reviews by applying the preprocessing techniques as employed in this manuscript. The relevant terms in the reviews may be then mapped with Android API documentation to form the clusters based on the components addressed in the review.

Assume the user review for the cricket game application, "Whenever I open this CRC League application, it automatically clicks my photograph and also deducts one dollar from my account. I also received the message that says Thank you for subscribing to IOIO service." After reading this review, one undoubtedly thinks that this is a malicious application. There may be hundreds of reviews related to this context. The data-driven analysis here can understand the text structure, words, and the topic discussed in the review. This review reflects that this application accesses camera, sends the SMS, and deducts the amount from the user account. One may think that a cricket game can never be made for performing these types of sensitive operations. This scenario only depicts the security issue of an application. Therefore, the semantics of the review can be discovered to flag these applications as suspicious using LSA.

8.3. Preserving the Proprietary Rights of the Android Developers. Repackaging is an open issue in the Android malware detection and analysis field. Using this technique, malware authors first download the legitimate application from the application stores and then extract all files and folders of the application. After the extraction process, they inject the malicious code or segment into the application and upload the same on other application stores. They also entice users to download that malicious application by performing social engineering activities. Innocent users not aware of this fact get trapped and download the malicious version of the legitimate application. In this way, the malware penetrates the phone and their device gets compromised. Repackaging thus opens the other dimensions for the malware authors to generate malicious clone or plagiarized versions of the legitimate applications. In a nutshell, the proprietary rights of developers are widely exploited and abused among malware authors to create clone Android malware variants of legitimate applications. Furthermore, they also deploy the evasion technique to dodge the detection process. In this scenario, LSA can be used to infer the semantics from the corpus of source code files. The degree of similarity can be measured by comparing the code segments of the source code files.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this article.

Acknowledgments

This research work was self-funded.

References

[1] A. White and K. Schmidt, "Systematic literature reviews," *Complementary Therapies in Medicine*, vol. 13, no. 1, pp. 54–60, 2005.

[2] D. Delen and M. D. Crossland, "Seeding the survey and analysis of research literature with text mining," *Expert Systems with Applications*, vol. 34, no. 3, pp. 1707–1720, 2008.

[3] N. Evangelopoulos, X. Zhang, and V. R. Prybutok, "Latent semantic analysis: five methodological recommendations," *European Journal of Information Systems*, vol. 21, no. 1, pp. 70–86, 2012.

[4] S. Lee, J. Song, and Y. Kim, "An empirical comparison of four text mining methods," *Journal of Computer Information Systems*, vol. 51, no. 1, pp. 1–10, 2010.

[5] S. Sehra, J. Singh, and H. Rai, "Using latent semantic analysis to identify research trends in openstreetmap," *ISPRS International Journal of Geo-Information*, vol. 6, no. 7, p. 195, 2017.

[6] M. Yalcinkaya and V. Singh, "Patterns and trends in building information modeling (BIM) research: a latent semantic analysis," *Automation in Construction*, vol. 59, pp. 68–80, 2015.

[7] N. Xie, X. Wang, W. Wang, and J. Liu, "Fingerprinting android malware families," *Frontiers of Computer Science*, vol. 13, no. 3, pp. 637–646, 2019.

[8] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C. Wolf, "Mobile security catching up? Revealing the nuts and bolts of the security of mobile devices," in *Proceedings of the 2011 IEEE Symposium on Security and Privacy*, IEEE, Oakland, CA, USA, 2011.

[9] W. Enck, "Defending users against smartphone apps: techniques and future directions," in *Proceedings of the 2011 International Conference on Information Systems Security*, 2011.

[10] P. Faruki, A. Bharmal, V. Laxmi et al., "Android security: a survey of issues, malware penetration, and defenses," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 2, pp. 998–1022, 2014.

[11] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and A. Ribagorda, "Evolution, detection and analysis of malware for smart devices," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 961–987, 2013.

[12] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering—a systematic literature review," *Information and Software Technology*, vol. 51, no. 1, pp. 7–15, 2009.

[13] L. See, P. Mooney, G. Foody et al., "Crowdsourcing, citizen science or volunteered geographic information? The current state of crowdsourced geographic information," *ISPRS International Journal of Geo-Information*, vol. 5, no. 5, p. 55, 2016.

[14] A. Kundu, V. Jain, S. Kumar, and C. Chandra, "A journey from normative to behavioral operations in supply chain management: a review using latent semantic analysis," *Expert Systems with Applications*, vol. 42, no. 2, pp. 796–809, 2015.

[15] E. Altszyler, S. Ribeiro, M. Sigman, and D. Fernández Slezak, "The interpretation of dream meaning: resolving ambiguity using latent semantic analysis in a small corpus of text," *Consciousness and Cognition*, vol. 56, pp. 178–187, 2017.

[16] A. Balahur, R. Mihalcea, and A. Montoyo, "Computational approaches to subjectivity and sentiment analysis: present and envisaged methods and applications," *Computer Speech & Language*, vol. 28, no. 1, pp. 1–6, 2014.

[17] J. N. De Boer, A. E. Voppel, M. J. H. Begemann, H. G. Schnack, F. Wijnen, and I. E. C. Sommer, "Clinical use of semantic space models in psychiatry and neurology: a systematic review and meta-analysis," *Neuroscience & Biobehavioral Reviews*, vol. 93, pp. 85–92, 2018.

[18] O. B. Driss, S. Mellouli, and Z. Trabelsi, "From citizens to government policy-makers: social media data analysis," *Government Information Quarterly*, vol. 36, pp. 560–570, 2019.

[19] H. Elghazel, A. Aussem, O. Gharroudi, and W. Saadaoui, "Ensemble multi-label text categorization based on rotation forest and latent semantic indexing," *Expert Systems with Applications*, vol. 57, pp. 1–11, 2016.

[20] G. Gao, Y.-S. Liu, P. Lin, M. Wang, M. Gu, and J.-H. Yong, "Bimtag: concept-based automatic semantic annotation of online BIM product resources," *Advanced Engineering Informatics*, vol. 31, pp. 48–61, 2017.

[21] J. Guan, A. S. Manikas, and L. H. Boyd, "The international journal of production research at 55: a content-driven review and analysis," *International Journal of Production Research*, vol. 57, no. 15-16, pp. 4654–4666, 2019.

[22] P. D. Hutchison, R. J. Daigle, and B. George, "Application of latent semantic analysis in AIS academic research," *International Journal of Accounting Information Systems*, vol. 31, pp. 83–96, 2018.

[23] H. Kim, H. Lee, and J. Seo, "A reliable FAQ retrieval system using a query log classification technique based on latent semantic analysis," *Information Processing & Management*, vol. 43, no. 2, pp. 420–430, 2007.

[24] S. S. Kulkarni, U. M. Apte, and N. E. Evangelopoulos, "The use of latent semantic analysis in operations management research," *Decision Sciences*, vol. 45, no. 5, pp. 971–994, 2014.

[25] X. Lin, Y. Li, and X. Wang, "Social commerce research: definition, research themes and the trends," *International Journal of Information Management*, vol. 37, no. 3, pp. 190–201, 2017.

[26] O. Müller, T. Schmiedel, E. Gorbacheva, and J. Vom Brocke, "Towards a typology of business process management professionals: identifying patterns of competences through latent semantic analysis," *Enterprise Information Systems*, vol. 10, no. 1, pp. 50–80, 2016.

[27] G. Pilato and E. D'Avanzo, "Data-driven social mood analysis through the conceptualization of emotional fingerprints," *Procedia Computer Science*, vol. 123, pp. 360–365, 2018.

[28] Y. Tonta and H. R. Darvish, "Diffusion of latent semantic analysis as a research tool: a social network analysis approach," *Journal of Informetrics*, vol. 4, no. 2, pp. 166–174, 2010.

[29] C.-P. Wei, C. C. Yang, and C.-M. Lin, "A latent semantic indexing-based approach to multilingual document clustering," *Decision Support Systems*, vol. 45, no. 3, pp. 606–620, 2008.

[30] S. Kim, H. Park, and J. Lee, "Word2vec-based latent semantic analysis (W2V- LSA) for topic modeling: a study on blockchain technology trend analysis," *Expert Systems with Applications*, vol. 152, no. 113, p. 401, 2020.

[31] G. Jorge-Botana, R. Olmos, and J. M. Luzón, "Bridging the theoretical gap between semantic representation models without the pressure of a ranking: some lessons learnt from LSA," *Cognitive Processing*, vol. 21, no. 1, pp. 1–21, 2020.

[32] A. Hassani, A. Iranmanesh, and N. Mansouri, "Text mining using nonnegative matrix factorization and latent semantic analysis," *Neural Computing and Applications*, Springer, Berlin, Germany, 2021.

[33] X. Ren and M. N. Coutanche, "Sleep reduces the semantic coherence of memory recall: an application of latent semantic analysis to investigate memory reconstruction," *Psychonomic Bulletin & Review*, vol. 28, pp. 1336–1343, 2021.

[34] S. Gowthami and R. Harikumar, "Conventional neural network for blind image blur correction using latent semantics," *Soft Computing*, vol. 24, pp. 15223–15237, 2020.

[35] F. Sastre, A. Velazquez, L. S. de Leon, J. Montanes, and J. Rodrigo, "Method to solve redundant inverse problems based on a latent semantic analysis approach. application to an aerojet engine," *Aerospace Science and Technology*, vol. 102, Article ID 105854, 2020.

[36] N. Evangelopoulos and S. Y. Amirkiaee, "Extracting LSA topics as features for text classifiers across different knowledge domains," *Quality & Quantity*, vol. 54, no. 1, pp. 249–261, 2020.

[37] C. Shen and J. Ho, "Technology-enhanced learning in higher education: a bibliometric analysis with latent semantic approach," *Computers in Human Behavior*, vol. 104, Article ID 106177, 2020.

[38] A. A. Wagire, A. Rathore, and R. Jain, "Analysis and synthesis of industry 4.0 research landscape," *Journal of Manufacturing Technology Management*, vol. 31, 2019.

[39] M. F. A. Razak, N. B. Anuar, R. Salleh, and A. Firdaus, "The rise of "malware": bibliometric analysis of malware study," *Journal of Network and Computer Applications*, vol. 75, pp. 58–76, 2016.

[40] K. Liu, S. Xu, G. Xu, M. Zhang, D. Sun, and H. Liu, "A review of android malware detection approaches based on machine learning," *IEEE Access*, vol. 8, pp. 124579–124607, 2020.

[41] S. R. T. Mat, M. F. Ab Razak, M. N. M. Kahar, J. M. Arif, S. Mohamad, and A. Firdaus, "Towards a systematic description of the field using bibliometric analysis: malware evolution," *Scientometrics*, vol. 126, no. 3, pp. 2013–2055, 2021.

[42] A. Amado, P. Cortez, P. Rita, and S. Moro, "Research trends on big data in marketing: a text mining and topic modeling based literature analysis," *European Research on Management and Business Economics*, vol. 24, no. 1, pp. 1–7, 2018.

[43] S. Kavvadias, G. Drosatos, and E. Kaldoudi, "Supporting topic modeling and trends analysis in biomedical literature," *Journal of Biomedical Informatics*, vol. 110, Article ID 103574, 2020.

[44] M. Mustak, J. Salminen, L. Plé, and J. Wirtz, "Artificial intelligence in marketing: topic modeling, scientometric analysis, and research agenda," *Journal of Business Research*, vol. 124, pp. 389–404, 2021.

[45] J. Rumbut, H. Fang, and H. Wang, "Topic modeling for systematic review of visual analytics in incomplete longitudinal behavioral trial data," *Smart Health*, vol. 18, Article ID 100142, 2020.

[46] J. An, K. Kim, L. Mortara, and S. Lee, "Deriving technology intelligence from patents: preposition-based semantic analysis," *Journal of Informetrics*, vol. 12, no. 1, pp. 217–236, 2018.

[47] J. L. Hurtado, A. Agarwal, and X. Zhu, "Topic discovery and future trend forecasting for texts," *Journal of Big Data*, vol. 3, no. 1, p. 7, 2016.

[48] N. E. Evangelopoulos, "Latent semantic analysis," *Wiley Interdisciplinary Reviews: Cognitive Science*, vol. 4, no. 6, pp. 683–692, 2013.

[49] K. R. Larsen and D. E. Monarchi, "A mathematical approach to categorization and labeling of qualitative data: the latent categorization method," *Sociological Methodology*, vol. 34, no. 1, pp. 349–392, 2004.

[50] J. F. López-Quintero, J. M. Cueva Lovelle, R. González Crespo, and V. García-Díaz, "A personal knowledge management metamodel based on semantic analysis and social information," *Soft Computing*, vol. 22, no. 6, pp. 1845–1854, 2018.

[51] A. Sidorova, N. Evangelopoulos, J. S. Valacich, and T. Ramakrishnan, "Uncovering the intellectual core of the information systems discipline," *MIS Quarterly*, vol. 32, no. 3, pp. 467–482, 2008.

[52] H. Tao, J. Li, T. Luo, and C. Wang, "Research on topics trends based on weighted k-means," in *Proceedings of the 2017 7th IEEE International Conference on Electronics Information and Emergency Communication (ICEIEC)*, 2017.

[53] J. Chen, W. Wei, C. Guo, L. Tang, and L. Sun, "Textual analysis and visualization of research trends in data mining for electronic health records," *Health Policy and Technology*, vol. 6, no. 4, pp. 389–400, 2017.

[54] S. Goyal, M. Ahuja, and J. Guan, "Information systems research themes: a seventeen-year data-driven temporal analysis," *Communications of the Association for Information Systems*, vol. 43, no. 1, p. 23, 2018.

[55] H. Jiang, M. Qiang, and P. Lin, "A topic modeling based bibliometric exploration of hydropower research," *Renewable and Sustainable Energy Reviews*, vol. 57, pp. 226–237, 2016.

[56] M. Kamber and J. Pei, *Data Mining*, Morgan Kaufmann, Burlington, MA, USA, 2006.

[57] T. Young, D. Hazarika, S. Poria, and E. Cambria, "Recent trends in deep learning based natural language processing," *IEEE Computational Intelligence Magazine*, vol. 13, no. 3, pp. 55–75, 2018.

[58] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, no. 6, pp. 391–407, 1990.

[59] M. M. Hossain, V. Prybutok, and N. Evangelopoulos, "Causal latent semantic analysis (CLSA): an illustration," *International Business Research*, vol. 4, no. 2, p. 38, 2011.

[60] P. Kherwa and P. Bansal, "Latent semantic analysis: an approach to understand semantic of text," in *Proceedings of the 2017 International Conference on Current Trends in Computer, Electrical, Electronics and Communication (CTCEEC)*, 2017.

[61] C. S. Kim, S. J. Choi, and K. Y. Kwahk, "Investigation of research trends in information systems domain using topic modeling and time series regression analysis," *Journal of Digital Contents Society*, vol. 18, no. 6, pp. 1143–1150, 2017.

[62] S. K. Sehra, Y. S. Brar, N. Kaur, and S. S. Sehra, "Research patterns and trends in software effort estimation," *Information and Software Technology*, vol. 91, pp. 1–21, 2017.

[63] L. Sun and Y. Yin, "Discovering themes and trends in transportation research using topic modeling," *Transportation Research Part C: Emerging Technologies*, vol. 77, pp. 49–66, 2017.

[64] Elsevier, "Mendeley desktop," 2020, https://www.mendeley.com/download-desktop/.

[65] Y. Shinyama, *PDF Miner*, 2004.

[66] R. Feldman and J. Sanger, *The Text Mining Handbook: Advanced Approaches in Analyzing Unstructured Data*, Cambridge University Press, Cambridge, UK, 2007.

[67] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[68] Python, "Natural language toolkit (Nltk)," 2020, https://www.nltk.org/.

[69] J. H. Paik, "A novel TF-IDF weighting scheme for effective ranking," in *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2013.

[70] G. Salton and C. Buckley, "Term-weighting approaches in automatic text retrieval," *Information Processing & Management*, vol. 24, no. 5, pp. 513–523, 1988.

[71] S. T. Dumais, "Latent semantic analysis," *Annual Review of Information Science and Technology*, vol. 38, no. 1, pp. 188–230, 2004.

[72] R. B. Bradford, "An empirical study of required dimensionality for large-scale latent semantic indexing applications," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, 2008.

[73] A. A. Wagire, A. P. S. Rathore, and R. Jain, "Exploration of pillars of industry 4.0 using latent semantic analysis technique," in *Intelligent Manufacturing and Energy Sustainability*, pp. 711–719, Springer, Berlin, Germany, 2020.

[74] M. Zhu and A. Ghodsi, "Automatic dimensionality selection from the scree plot via the use of profile likelihood," *Computational Statistics & Data Analysis*, vol. 51, no. 2, pp. 918–930, 2006.

[75] R. Jingbiao and Y. Shaohong, "Research and improvement of clustering algorithm in data mining," in *Proceedings of the 2010 2nd International Conference on Signal Processing Systems*, vol. 1, 2010.

[76] M. Srinivas and C. K. Mohan, "Efficient clustering approach using incremental and hierarchical clustering methods," in *Proceedings of the 2010 International Joint Conference on Neural Networks (IJCNN)*, 2010.

[77] S. Singh and A. Yadav, "Study of k-means and enhanced k-means clustering algorithm," *International Journal of Advanced Research in Computer Science*, vol. 4, no. 10, 2013.

[78] I. S. Dhillon and D. S. Modha, "Concept decompositions for large sparse text data using clustering," *Machine Learning*, vol. 42, no. 1-2, pp. 143–175, 2001.

[79] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[80] A. Rangrej, S. Kulkarni, and A. V. Tendulkar, "Comparative study of clustering techniques for short text documents," in *Proceedings of the 20th International Conference Companion on World Wide Web*, 2011.

[81] H. A. Linstone and M. Turoff, *The Delphi Method*, Addison-Wesley, Boston, MA, USA, 1975.

[82] A. P. Fuchs, A. Chaudhuri, and J. S. Foster, "SCanDroid: automated security certification of android application," Technical report, University of Maryland, College Park, MD, USA, 2009.

[83] W. Enck, P. Gilbert, S. Han et al., "Taintdroid: an information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems (TOCS)*, vol. 32, no. 2, p. 5, 2014.

[84] P. Faruki, V. Kumar, B. Ammar, M. S. Gaur, V. Laxmi, and M. Conti, "Platform neutral sandbox for analyzing malware and resource hogger apps," in *Proceedings of the 2014 International Conference on Security and Privacy in Communication Networks*, Springer, Beijing, China, 2014.

[85] D. Sounthiraraj, J. Sahs, G. Greenwood, Z. Lin, and L. Khan, "SMV-hunter: large scale, automated detection of SSL/TLS man-in-the-middle vulnerabilities in android apps," in *Proceedings of the 21st Annual Network and Distributed System Security Symposium*, San Diego, CA, USA, 2014.

[86] R. Pandita, X. Xiao, W. Yang, W. Enck, and T. Xie, "WHYPER: towards automating risk assessment of mobile applications," in *Proceedings of the 22nd USENIX Security Symposium*, Washington, DC, USA, 2013.

[87] G. Dini, F. Martinelli, A. Saracino, and D. Sgandurra, "MADAM: a multi-level anomaly detector for android malware," in *Proceedings of the 2012 International Conference on Mathematical Methods, Models, and Architectures for Computer Network Security*, 2012.

[88] W. Zhou, Y. Zhou, X. Jiang, and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *Proceedings of the 2nd ACM Conference on Data and Application Security and Privacy*, 2012.

[89] K. W. Y. Au, Y. F. Zhou, Z. Huang, and D. Lie, "PScout: analyzing the android permission specification," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, ACM, Raleigh, NC, USA, 2012.

[90] A. P. Felt, E. Chin, S. Hanna, D. Song, and D. Wagner, "Android permissions demystified," in *Proceedings of the 18th ACM Conference on Computer and Communications Security*, 2011.

[91] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "DREBIN: effective and explainable detection of android malware in your pocket," in *Proceedings of the 2014 Network and Distributed System Security Symposium*, vol. 14, San Diego, CA, USA, 2014.

[92] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac, and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2671–2701, 2019.

[93] Z. Luoshi, N. Yan, W. Xiao, W. Zhaoguo, and X. Yibo, "A3: automatic analysis of android malware," in *Proceedings of the 1st International Workshop on Cloud Computing and Information Security*, University of Western Australia, Perth, Australia, 2013.

[94] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh, and L. Cavallaro, "The evolution of android malware and android analysis techniques," *ACM Computing Surveys (CSUR)*, vol. 49, no. 4, p. 76, 2017.

[95] J. Kim, Y. Yoon, K. Yi, J. Shin, and S. Center, "Scandal: static analyzer for detecting privacy leaks in android applications," *MoST*, vol. 12, no. 110, p. 1, 2012.

[96] K. Z. Chen, N. M. Johnson, V. D'Silva et al., "Contextual policy enforcement in android applications with permission event graphs," in *Proceedings of the 2013 Network & Distributed System Security Symposium*, San Diego, CA, USA, 2013.

[97] A. Desnos and G. Gueguen, "Android: from reversing to decompilation," in *Proceedings of the 2011 Black Hat*, pp. 77–101, Abu Dhabi, UAE, 2011.

[98] J. Hoffmann, M. Ussath, T. Holz, and M. Spreitzenbarth, "Slicing droids: program slicing for smali code," in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, Coimbra, Portugal, 2013.

[99] M. Zhang and H. Yin, "Appsealer: automatic generation of vulnerability-specific patches for preventing component hijacking attacks in android applications," in *Proceedings of the 2014 NDSS*, San Diego, CA, USA, 2014.

[100] W. Enck, D. Octeau, P. D. McDaniel, and S. Chaudhuri, "A study of android application security," in *Proceedings of the*

2011 USENIX Security Symposium, vol. 2, San Francisco, CA, USA, 2011.

[101] C. Gibler, J. Crussell, J. Erickson, and H. Chen, "AndroidLeaks: automatically detecting potential privacy leaks in android applications on a large scale," in *Proceedings of the 2012 International Conference on Trust and Trustworthy Computing*, 2012.

[102] S. Arzt, S. Rasthofer, C. Fritz et al., "FlowDroid: precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps," in *ACM Sigplan Notices*, vol. 49, pp. 259–269, ACM, New York, NY, USA, 2014.

[103] Y. Feng, S. Anand, I. Dillig, and A. Aiken, "Apposcopy: semantics-based detection of android malware through static analysis," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014.

[104] L. Li, A. Bartel, T. F. Bissyandé et al., "ICCTA: detecting inter-component privacy leaks in android apps," in *Proceedings of the 37th International Conference on Software Engineering*, 2015.

[105] M. Sun, T. Wei, and J. Lui, "TaintART: a practical multi-level information-flow tracking system for android runtime," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, ACM, Vienna, Austria, 2016.

[106] F. Wei, S. Roy, X. Ou, and Robby, "Amandroid: a precise and general inter- component data flow analysis framework for security vetting of android apps," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, ACM, Scottsdale, AZ, USA, 2014.

[107] S. Wu, P. Wang, X. Li, and Y. Zhang, "Effective detection of android malware based on the usage of data flow apis and machine learning," *Information and Software Technology*, vol. 75, pp. 17–25, 2016.

[108] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: scalable and accurate zero-day android malware detection," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ACM, Windermere, UK, 2012.

[109] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang, "Hey, you, get off of my market: detecting malicious apps in official and alternative android markets," in *Proceedings of the 2012 NDSS*, vol. 25, San Diego, CA, USA, 2012.

[110] H. S. Karlsen, E. R. Wognsen, M. C. Olesen, and R. R. Hansen, "Study, formalisa-tion, and analysis of dalvik bytecode," in *Proceedings of the 2012 7th Workshop on Bytecode Semantics, Verification, Analysis and Transformation*, Tallinn, Estonia, 2012.

[111] P. Faruki, V. Laxmi, M. S. Gaur, and P. Vinod, "Mining control flow graph as API call-grams to detect portable executable malware," in *Proceedings of the 5th International Conference on Security of Information and Networks*, ACM, Jaipur, India, 2012.

[112] D. J. Wu, C. H. Mao, T. E. Wei, H. M. Lee, and K. P. Wu, "Droidmat: android malware detection through manifest and API calls tracing," in *Proceedings of the 2012 7th Asia Joint Conference on Information Security*, 2012.

[113] B. Sanz, I. Santos, X. Ugarte-Pedrero, C. Laorden, J. Nieves, and P. G. Bringas, "Anomaly detection using string analysis for android malware detection," in *Proceedings of the 2014 International Joint Conference SOCO13-CISIS13-ICEUTE13*, 2014.

[114] R. Baeza-Yates, B. Ribeiro-Neto, and B. d. A. N. Ribeiro, *Modern Information Retrieval*, Vol. 463, ACM Press, New York, NY, USA, 1999.

[115] Y. Aafer, W. Du, and H. Yin, "DroidAPIMiner: mining api-level features for robust malware detection in android," in *Proceedings of the 2013 International Conference on Security and Privacy in Communication Systems*, 2013.

[116] V. Avdiienko, K. Kuznetsov, A. Gorla et al., "Mining apps for abnormal usage of sensitive data," in *Proceedings of the 37th International Conference on Software Engineering*, 2015.

[117] J. Dave, S. Saharan, P. Faruki, V. Laxmi, and M. S. Gaur, "Secure random encryption for deduplicated storage," in *Proceedings of the 2017 International Conference on Information Systems Security*, 2017.

[118] H. Peng, C. Gates, B. Sarma et al., "Using probabilistic generative models for ranking risks of android apps," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, 2012.

[119] Y. Zhou and X. Jiang, "Dissecting android malware: characterization and evolution," in *Proceedings of the 2012 IEEE Symposium on Security and Privacy*, 2012.

[120] X. Sun, Y. Zhongyang, Z. Xin, B. Mao, and L. Xie, "Detecting code reuse in android applications using component-based control flow graph," in *Proceedings of the 2014 IFIP International Information Security Conference*, 2014.

[121] J. Crussell, C. Gibler, and H. Chen, "Attack of the clones: detecting cloned applications on android markets," in *Proceedings of the 2012 European Symposium on Research in Computer Security*, 2012.

[122] F. Zhang, H. Huang, S. Zhu, D. Wu, and P. Liu, "ViewDroid: towards obfuscation-resilient mobile application repackaging detection," in *Proceedings of the 2014 ACM Conference on Security and Privacy in Wireless & Mobile Networks*, ACM, Oxford, UK, 2014.

[123] Y. Shao, X. Luo, C. Qian, P. Zhu, and L. Zhang, "Towards a scalable resource-driven approach for detecting repackaged android applications," in *Proceedings of the 30th Annual Computer Security Applications Conference*, 2014.

[124] J. Crussell, C. Gibler, and H. Chen, "AnDarwin: scalable detection of semantically similar android applications," in *Proceedings of the European Symposium on Research in Computer Security*, 2013.

[125] A. Al-Haiqi, M. Ismail, and R. Nordin, "On the best sensor for keystrokes inference attack on android," *Procedia Technology*, vol. 8, pp. 947–953, 2013.

[126] L. Deshotels, "Inaudible sound as a covert channel in mobile devices," in *Proceedings of the 8th USENIX Workshop on Offensive Technologies*, San Diego, CA, USA, 2014.

[127] E. Miluzzo, A. Varshavsky, S. Balakrishnan, and R. R. Choudhury, "Tapprints: your finger taps have fingerprints," in *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, ACM, Low Wood Bay Lake District, UK, 2012.

[128] R. Schlegel, A. Kapadia, and X. Wang, "Soundcomber: a stealthy and context-aware sound Trojan for smartphones," in *Proceedings of the 2011 NDSS*, vol. 11, San Diego, CA, USA, 2011.

[129] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng, "Stealthy video capturer: a new video-based spyware in 3G smartphones," in *Proceedings of the 2nd ACM Conference on Wireless Network Security*, ACM, Zurich, Switzerland, 2009.

[130] G. Suarez-Tangil, J. E. Tapiador, F. Lombardi, and R. Di Pietro, "Alterdroid: differential fault analysis of obfuscated

smartphone malware," *IEEE Transactions on Mobile Computing*, vol. 15, no. 4, pp. 789–802, 2015.

[131] G. Suarez-Tangil, J. E. Tapiador, P. Peris-Lopez, and J. Blasco, "Dendroid: a text mining approach to analyzing and classifying code structures in android malware families," *Expert Systems with Applications*, vol. 41, no. 4, pp. 1104–1117, 2014.

[132] G. Suarez-Tangil, S. K. Dash, M. Ahmadi, J. Kinder, G. Giacinto, and L. Cavallaro, "DroidSieve: fast and accurate classification of obfuscated android malware," in *Proceedings of the 7th ACM on Conference on Data and Application Security and Privacy*, ACM, Scottsdale, AZ, USA, 2017.

[133] W. Wang, Y. Li, X. Wang, J. Liu, and X. Zhang, "Detecting android malicious apps and categorizing benign apps with ensemble of classifiers," *Future Generation Computer Systems*, vol. 78, pp. 987–994, 2018.

[134] M. Lindorfer, M. Neugschwandtner, L. Weichselbaum, Y. Fratantonio, V. Van Der Veen, and C. Platzer, "Andrubis 1,000,000 apps later: a view on current android malware behaviors," in *Proceedings of the 2014 3rd International Workshop on Building Analysis Datasets and Gathering Experience Returns for Security (BADGERS)*, 2014.

[135] A. Desnos and P. Lantz, "Droidbox: an android application sandbox for dynamic analysis," Technical report, Lund University, Lund, Sweden, 2011.

[136] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, 2009.

[137] L. K. Yan and H. Yin, "Droidscope: seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis," in *Proceedings of the 21st USENIX Security Symposium*, Bellevue, WA, USA, 2012.

[138] P. Faruki, V. Laxmi, A. Bharmal, M. S. Gaur, and V. Ganmoor, "Androsimilar: robust signature for detecting variants of android malware," *Journal of Information Security and Applications*, vol. 22, pp. 66–80, 2015.

[139] A. T. Kabakus and I. A. Dogru, "An in-depth analysis of android malware using hybrid techniques," *Digital Investigation*, vol. 24, pp. 25–33, 2018.

[140] J. Fu, J. Xue, Y. Wang, Z. Liu, and C. Shan, "Malware visualization for fine- grained classification," *IEEE Access*, vol. 6, pp. 14510–14523, 2018.

[141] S. Ni, Q. Qian, and R. Zhang, "Malware identification using visualization images and deep learning," *Computers & Security*, vol. 77, pp. 871–885, 2018.

[142] S. Sun, X. Fu, H. Ruan, X. Du, B. Luo, and M. Guizani, "Real-time behavior analysis and identification for android application," *IEEE Access*, vol. 6, pp. 38041–38051, 2018.

[143] J. Yan, Y. Qi, and Q. Rao, "LSTM-based hierarchical denoising network for android malware detection," *Security and Communication Networks*, vol. 2018, Article ID 5249190, 18 pages, 2018.

[144] J. Li, L. Sun, Q. Yan, Z. Li, W. Srisa-an, and H. Ye, "Significant permission identification for machine-learning-based android malware detection," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 7, pp. 3216–3225, 2018.

[145] P. Faruki, H. Fereidooni, V. Laxmi, M. Conti, and M. Gaur, "Android code protection via obfuscation techniques: past, present and future directions," 2016, https://arxiv.org/abs/1611.10231.

[146] W. Wang, Z. Gao, M. Zhao, Y. Li, J. Liu, and X. Zhang, "Droidensemble: detecting android malicious applications with ensemble of string and structural static features," *IEEE Access*, vol. 6, pp. 31798–31807, 2018.

[147] S. Hyun, J. Cho, G. Cho, and H. Kim, "Design and analysis of push notification- based malware on android," *Security and Communication Networks*, vol. 2018, Article ID 8510256, 12 pages, 2018.

[148] Y. Liu, K. Guo, X. Huang, Z. Zhou, and Y. Zhang, "Detecting android malwares with high-efficient hybrid analyzing methods," *Mobile Information Systems*, vol. 2018, Article ID 1649703, 12 pages, 2018.

[149] Symantec, "Internet security threat report 2018," 2018, https://www.symantec.com/content/dam/symantec/docs/reports/istr-23-2018-en.pdf.

[150] B. A. S. Al-rimy, M. A. Maarof, and S. Z. M. Shaid, "Ransomware threat success factors, taxonomy, and countermeasures: a survey and research directions," *Computers & Security*, vol. 74, pp. 144–166, 2018.

[151] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: visualization and automatic classification," in *Proceedings of the 8th Inter- National Symposium on Visualization for Cyber Security*, ACM, Pittsburgh, PA, USA, 2011.

[152] L. Nataraj, D. Kirat, B. Manjunath, and G. Vigna, "SARVAM: search and retrieval of malware," in *Proceedings of the Annual Computer Security Conference (ACSAC) Workshop on Next Generation Malware Attacks and Defense (NGMAD)*, University of Western Australia, Los Angeles, CA, USA, 2013.

[153] K. Bakour and H. M. Ünver, "VisDroid: android malware classification based on local and global image features, bag of visual words and machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 8, pp. 3133–3153, 2021.

[154] A. Mahindru and A. L. Sangal, "MLDroid-framework for android malware detection using machine learning techniques," *Neural Computing and Applications*, vol. 33, no. 10, pp. 5183–5240, 2021.

[155] Y. Zhao, L. Li, H. Wang et al., "On the impact of sample duplication in machine-learning-based android malware detection," *ACM Transactions on Software Engineering and Methodology*, vol. 30, no. 3, pp. 1–38, 2021.

[156] J. Singh, D. Thakur, F. Ali, T. Gera, and K. S. Kwak, "Deep feature extraction and classification of android malware images," *Sensors*, vol. 20, no. 24, p. 7013, 2020.

[157] D. Vasan, M. Alazab, S. Wassan, H. Naeem, B. Safaei, and Q. Zheng, "IMCFN: image-based malware classification using fine-tuned convolutional neural network architecture," *Computer Networks*, vol. 171, Article ID 107138, 2020.