Computers, Materials & Continua
DOI:10.32604/cmc.2021.014504
Article

Tech Science Press

Understanding Research Trends in Android Malware Research Using Information Modelling Techniques

Jaiteg Singh[1], Tanya Gera[1], Farman Ali[2], Deepak Thakur[1], Karamjeet Singh[3] and Kyung-sup Kwak[4,*]

[1]Chitkara University Institute of Engineering & Technology, Chitkara University, Punjab, 140401, India
[2]Department of Software, Sejong University, Seoul, 05006, South Korea
[3]Department of Computer Science and Engineering, Thapar University, Patiala, 147001, India
[4]Department of Information & Communication Engineering, Inha University, Incheon, 22212, South Korea
[*]Corresponding Author: Kyung-sup Kwak. Email: kskwak@inha.ac.kr
Received: 24 September 2020; Accepted: 12 October 2020

Abstract: Android has been dominating the smartphone market for more than a decade and has managed to capture 87.8% of the market share. Such popularity of Android has drawn the attention of cybercriminals and malware developers. The malicious applications can steal sensitive information like contacts, read personal messages, record calls, send messages to premium-rate numbers, cause financial loss, gain access to the gallery and can access the user's geographic location. Numerous surveys on Android security have primarily focused on types of malware attack, their propagation, and techniques to mitigate them. To the best of our knowledge, Android malware literature has never been explored using information modelling techniques. Further, promulgation of contemporary research trends in Android malware research has never been done from semantic point of view. This paper intends to identify intellectual core from Android malware literature using Latent Semantic Analysis (LSA). An extensive *corpus* of 843 articles on Android malware and security, published during 2009–2019, were processed using LSA. Subsequently, the truncated singular Value Decomposition (SVD) technique was used for dimensionality reduction. Later, machine learning methods were deployed to effectively segregate prominent topic solutions with minimal bias. Apropos to observed term and document loading matrix values, this five core research areas and twenty research trends were identified. Further, potential future research directions have been detailed to offer a quick reference for information scientists. The study concludes to the fact that Android security is crucial for pervasive Android devices. Static analysis is the most widely investigated core area within Android security research and is expected to remain in trend in near future. Research trends indicate the need for a faster yet effective model to detect Android applications causing obfuscation, financial attacks and stealing user information.

Keywords: Android security; research trends; latent semantic analysis; vulnerabilities; malware; machine learning; clustering

1 Introduction

Android has been dominating the smartphone market for more than a decade and has managed to capture 87.8% of the market share [1]. Affordability, agility and reliability of Android smartphones have encouraged their use in e-commerce, banking, sending emails, using social media and marketing. Such popularity of Android has drawn the attention of cybercriminals and malware developers. An exponential increase in Android malware families has been observed by the research community. These malware families differ in the way they attack, the type of vulnerabilities they exploit and the Android subsystem they target [2]. At times, these attacks may even propagate through application stores like Google play store [3]. The malicious applications can steal sensitive information like contacts, read personal messages, record calls, send messages to premium-rate numbers, cause financial loss, gain access to the gallery and access the user's geographic location. Popularity among users and continuously increasing Android malware attacks has gained the attention of the researchers. Few of the benchmark literature reviews have also been done by the research community [2,4–6].

These surveys have primarily focused on types of malware attack, their propagation, and techniques to mitigate them. Few surveys also highlighted fundamental vulnerabilities of Android platforms. To the best of our knowledge, Android malware literature has never been explored using information modelling techniques. Further, promulgation of contemporary research trends in Android malware has never been done from semantic point of view. This paper intends to identify intellectual core from Android malware literature using Latent Semantic Analysis (LSA). LSA mimics the human brain to filter out semantics from the text as it is mathematically proven to model words, synonyms, and metaphors to elaborate various semantic aspects of qualitative literature [7–11]. LSA is reliably efficient in information retrieval and query optimization [12,13]. Many researchers from different research fields have used LSA to discover the research trends [10,14–17]. LSA identifies the entire of the contexts in which a word could appear and learns to establish a common factor to represent underlying concepts. Keeping into consideration other researches using information modeling techniques; the main contribution of this study is to discover current trends, future research directions, and core research areas pertaining to Android malware. To minimize opinion bias, K-means clustering was used to automatically map the document to its closest possible topic solution.

The rest of the paper is organized as follows: The second section introduces the available materials and methods along with the procedure to deploy LSA. The third section explains experimental results, different topic solutions, research trends, core research areas and their mapping. Section 4 concludes the findings.

2 Materials and Methods

Automated topic modelling techniques require minimal human intervention and can process thousands of articles in one go. However, manual review process does require human intervention at every step and can be biased sometimes [7]. It is very difficult to manually review full length articles in large numbers. A great manual effort is required to draw conclusions like research trends across large literature. At the document level, one of the most useful ways to understand the text is by analyzing its topics. The process of learning, recognizing, and extracting these topics across a collection of documents is called topic modeling. Numerous well-formed algorithms are available to produce research trends and core research areas within research field. In addition to it, numerous machine learning techniques could be used for performing data analysis, visualization and interpretation of results. LSA is the fundamental and most studied techniques in topic modeling. Papadimitriou et al. [18] investigated appropriate conditions for applying LSA. Few of the researchers also performed decision making to analyze trends in blockchain technology using Word2vec-based Latent Semantic Analysis (W2V-LSA) [19]. The experimental results confirmed its usefulness and better topic modelling than tradition bibliometric methods. Initially, research

trend analysis was also performed on doctoral dissertations and master's theses for identifying future research opportunities in the domain of blended learning [20]. Systematic information retrieval using automated and semi-automated approaches in any field of research has itself become a trend. However, recent advancements in text mining, information retrieval and topic modelling has gained attention of research community to forecast research trends [21]. Text mining has also facilitated machine learning algorithms to enhance its capability to mine latent information [22] from user review on websites, newspaper articles and social media information analysis. Other techniques like probabilistic latent semantic analysis use uni-gram format conversion of a word, which generally get fail to capture the specific context in the document [23]. Further, n-gram format leads to decrease in efficiency of model due to wide dimensionality [24]. Owing to these probable limitations of probabilistic latent semantic analysis, LSA has been widely accepted by research community to promulgate trends within research literature.

This section details the methodology to deploy LSA on Android malware literature. The keywords used to search research articles and adopted inclusion-exclusion criteria for selecting articles are detailed here. A manual search was performed across previously mentioned databases using the following search terms "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" search keywords, with Android as prefix. Prominent research databases like Google Scholar, Mendeley, ACM DL, Hindawi, Taylor and Francis, IEEE, Wiley and Scopus were searched to identify quality literature related to Android malware. The inclusion and exclusion criteria followed for selection of articles is mentioned in Tab. 1. The collected literature was pre-processed and organized using Mendeley reference manager [25]. Mendeley helped to introduce standard formatting to all documents which were indexed according to common objects.

Table 1: Inclusion and exclusion criteria

S. No.	Inclusion criteria	Exclusion criteria
1.	Articles must be published in the time frame 2009-2019.	Articles that were focusing on an operating system other than Android, e.g., BlackBerry, Symbian, iOS, Windows were excluded.
2.	Articles must have a focus on Android security, malware analysis, and malware detection and mitigation techniques.	Articles that did not focus Android security, malware analysis, malware detection, mitigation techniques and Android security threats were excluded

Initially, a total of 1289 abstracts and titles of articles published during 2009–2019 were collected by searching previously mentioned keywords. From amongst the collected documents, 251 duplicate entries were removed. Remaining 1038 articles were accessed and evaluated as per decided inclusion/exclusion criteria. Articles focusing on general malware (39), iOS (45), Symbian operating system (50), windows (61) were excluded. Finally, we were left with 843 articles to be processed using LSA. Owing to the required brevity of manuscript, the method to deploy LSA has been elaborated below with the help of an example:

Assume Sample Doc1 and Sample Doc2 as documents within a given document *corpus*. The required pre-processing and document-term matrix scores for identifying frequency of each term have been detailed in Tabs. 2 and 3.

Sample Doc1: Malware application reads the unique device identifier to track the user's device. Malware applications can misuse user data like his or her phone numbers, contact list, calendar, etc.

Table 2: Pre-processing procedure on sample documents

Steps	Document No.	Result
After Tokenization	Doc1	[‘Malware’, ‘application’, ‘reads’, ‘the’, ‘unique’, ‘device’, ‘identifier’, ‘to’, ‘track’, ‘the’, ‘user’, ‘s’, ‘device’ ‘Malware’, ‘applications’, ‘can’, ‘misuse’, ‘the’, ‘user’, ‘data’, ‘like’, ‘his’, ‘or’, ‘her’, ‘phone’, ‘numbers’, ‘contact’, ‘list’, ‘calendar’, ‘etc.’]
	Doc2	[‘Applications’, ‘can’, ‘track’, ‘down’, ‘the’, ‘exact’, ‘location’, ‘of’, ‘the’, ‘user’, ‘by’, ‘finding’, ‘the’, ‘wifi’, ‘network’, ‘or’, ‘tower’, ‘it’, ‘is’, ‘connected’, ‘to’ “Various’, ‘malware’, ‘applications’, ‘can’, ‘record’, ‘your’, ‘daily’, ‘usage’, ‘data’, ‘and’, ‘send’, ‘it’, ‘to’, ‘servers’, ‘Applications’, ‘can’, ‘access’, ‘the’, ‘message’, ‘logs’, ‘and’, ‘misuse’, ‘them’]
After Normalization	Doc1	Malware application reads the unique device identifier to track the user s device Malware applications can misuse the user data like his or her phone numbers contact list calendar etc.
	Doc2	Applications can track down the exact location of the user by finding the wifi network or tower it is connected to Various malware applications can record your daily usage data and send it to servers Applications can access the message logs and misuse them
After Removing stop words	Doc1	[‘Malware’, ‘application’, ‘reads’, ‘unique’, ‘device’, ‘identifier’, ‘track’, ‘user’, ‘device’, ‘Malware’, ‘applications’, ‘misuse’, ‘user’, ‘data’, ‘like’, ‘phone’, ‘numbers’, ‘contact’, ‘list’, ‘calendar’, ‘etc.’]
	Doc2	[‘Applications’, ‘track’, ‘exact’, ‘location’, ‘user’, ‘finding’, ‘wifi’, ‘network’, ‘tower’, ‘connected’, ‘Various’, ‘malware’, ‘applications’, ‘record’, ‘daily’, ‘usage’, ‘data’, ‘send’, ‘servers’, Applications’, ‘access’, ‘message’, ‘logs’, ‘misuse’]
After Stemming and Lemmatizing	Doc1	[‘malwar’, ‘applic’, ‘read’, ‘uniqu’, ‘devic’, ‘identifi’, ‘track’, ‘user’, ‘devic’, ‘malwar’, ‘applic’, ‘misus’, ‘user’, ‘data’, ‘like’, ‘phone’, ‘number’, ‘contact’, ‘list’, ‘calendar’, ‘etc.’]
	Doc2	[‘applic’, ‘track’, ‘exact’, ‘locat’, ‘user’, ‘find’, ‘wifi’, ‘network’, ‘tower’, ‘connect’, ‘various’, ‘malwar’, ‘applic’, ‘record’, ‘daili’, ‘usag’, ‘data’, ‘send’, ‘server’, ‘applic’, ‘access’, ‘messag’, ‘log’, ‘misus’]
Character Filtering	Doc1	[‘malwar’, ‘applic’, ‘read’, ‘uniqu’, ‘devic’, ‘identifi’, ‘track’, ‘user’, ‘devic’, ‘malwar’, ‘applic’, ‘misus’, ‘user’, ‘data’, ‘like’, ‘phone’, ‘number’, ‘contact’, ‘list’, ‘calendar’]
	Doc2	[‘applic’, ‘track’, ‘exact’, ‘locat’, ‘user’, ‘find’, ‘wifi’, ‘network’, ‘tower’, ‘connect’, various’, ‘malwar’, ‘applic’, ‘record’, ‘daili’, ‘usag’, ‘data’, ‘send’, ‘server, ‘applic’, ‘access’, ‘messag’, ‘misus’]

Table 3: Document-term matrix describing frequency of terms

Term	Doc1	Doc2	Term	Doc1	Doc2	Term	Doc1	Doc2
applic	2	2	exact	0	1	misus	1	0
calendar	1	0	find	0	1	network	0	1
connect	0	1	identifi	1	0	number	1	0
contact	1	0	like	1	0	phone	1	0
daili	0	1	list	1	0	read	1	0
data	1	1	locat	0	1	record	0	1
devic	2	0	malwar	2	1	send	0	1
server	0	1	uniqu	1	0	various	0	1
tower	0	1	usag	0	1	wifi	0	1
track	1	1	user	2	1			

Sample Doc2: Applications can track down the exact location of the user by finding the WiFi network or tower it is connected to. Various malware applications can record your daily usage data and send it to servers

As the task is to mine the relevant terms that provide useful or quality information about the document, the document-term matrix has to be replaced by Term-Frequency/Inverse Document Frequency (TF-IDF) weights for further processing of the matrix. TF-IDF weight is a measure to interpret the importance of a term to a document in a collection of the large *corpus*. TF-IDF works on the fact that relevant words are not necessarily frequent words. The TF-IDF weight is build-up of two terms that need to be calculated beforehand:

(i) TF (Term Frequency): It provides the frequency of a term (TF) with normalized value, which can be computed as the division of the number of occurrences of a term in a document and the total number of terms in that document, refer Eq. (1). Sample TF scores in context with an example taken are shown in Tab. 4.

Table 4: Term frequency scores for each sample document

Documents	Term frequency scores
Doc1	{'malwar': 0.1, 'applic': 0.1, 'read': 0.05, 'uniqu': 0.05, 'devic': 0.1, 'identifi': 0.05, 'track': 0.05, 'user': 0.1, 'misus': 0.05, 'data': 0.05, 'like': 0.05, 'phone': 0.05, 'number': 0.05, 'contact': 0.05, 'list': 0.05, 'calendar': 0.05}
Doc2	{'applic': 0.105, 'track': 0.053, 'exact': 0.053, 'locat': 0.053, 'user': 0.053, 'find': 0.053, 'wifi': 0.053, 'network': 0.053, 'tower': 0.053, 'connect': 0.053, 'various': 0.053, 'malwar': 0.053, 'record': 0.053, 'daili': 0.053, 'usag': 0.053, 'data': 0.053, 'send': 0.053, 'server': 0.053 }

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

(ii) IDF (Inverse Document Frequency): IDF indicates the importance of each term within the document. IDF is calculated as a log of the division of count of documents in the *corpus* divided by the count of documents where the specific term appears. However, it is quite obvious that

some terms, like "is", "the", "of", and "that" or certain domain-specific words, may appear repeatedly but may not have much importance. Thus, arises a need to weigh down the importance of most occurred terms while scaling up the rare ones. Therefore, IDF scores are important and can be computed as given in Eq. (2). IDF scores for sample documents are presented in Tab. 5.

Table 5: Inverse document frequency score for each term

Terms	IDF score	Terms	IDF score	Terms	IDF score	Terms	IDF score
applic	1.000000	list	1.405465	tower	1.405465	find	1.405465
calendar	1.405465	locat	1.405465	track	1.000000	identifi	1.405465
connect	1.405465	malwar	1.000000	uniqu	1.405465	like	1.405465
contact	1.405465	misus	1.405465	usag	1.405465	record	1.405465
daili	1.405465	network	1.405465	user	1.000000	send	1.405465
data	1.000000	number	1.405465	various	1.405465		
devic	1.405465	phone	1.405465	wifi	1.405465		
exact	1.405465	read	1.405465	server	1.405465		

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

Hence, after the calculation of TF and IDF scores, the final document-term matrix with TF-IDF scores is calculated with the following Eq. (3).

$$w_{t,d} = TF_{t,d} * log \frac{N}{df_t} \tag{3}$$

where t denotes the terms; d denotes each document; N denotes the total number of documents. Consider Tab. 6 above, it represents the document-term matrix with TF-IDF scores for previously stated example. A term will have more TF-IDF value when its occurrences across the document are more but less across the *corpus*. Let's take an example of a domain-specific word i.e., "malware" which was very common in entire of the *corpus*, but may appear often in a document hence, it will not have a high TF-IDF score. However, the word "permissions" may appear frequently in a document, and appears less in the rest of the *corpus*, it will have a higher TF-IDF score. As shown in the example given above, a large *corpus* of the research papers on Android security and malware resulted in high dimensional TF-IDF matrix. High dimensional matrix is generally expected to be redundant and noisy. Therefore, to uncover the relationship among the words and documents and to capture the latent topics within the *corpus*, dimensionality reduction was required. Term Frequency-Inverse Document Frequency (TF-IDF) matrix (as produced in the previous step) was fed to truncated Singular Vector Decomposition (SVD). SVD transforms the data from high dimensional vector space to a low dimensional vector space keeping the originality of the data sustained. Using SVD, the LSA would produce two matrices, one for the terms loading values and other for the documents loading values. High loading terms and high loading documents help to interpret their association with topic solutions. Each topic solution signifies the research theme across the *corpus*. Researchers can modify selection by varying the selections of the number of factor solutions. The most common research areas can be found easily by selecting a lower-level factor aggregation. A higher-level factor aggregation signifies principal or core research themes.

Truncated SVD is a matrix algebra technique which decomposes TF-IDF matrix into a product of three matrices—U, ∑ and V. The SVD decomposition over matrix A is represented as follows in Eq. (4):

Table 6: Transformed term frequencies after TF-IDF generation

Terms	Doc1	Doc2	Terms	Doc1	Doc2	Terms	Doc1	Doc2
applic	0.309883	0.344626	**like**	0.217765	0.000000	**send**	0.000000	0.242180
calendar	0.217765	0.000000	**list**	0.217765	0.000000	**server**	0.000000	0.242180
connect	0.000000	0.242180	**locat**	0.000000	0.242180	**tower**	0.000000	0.242180
contact	0.217765	0.000000	**malwar**	0.309883	0.172313	**track**	0.154942	0.172313
daili	0.000000	0.242180	**misus**	0.217765	0.000000	**uniqu**	0.217765	0.000000
data	0.154942	0.172313	**network**	0.000000	0.242180	**usag**	0.000000	0.242180
devic	0.435530	0.000000	**number**	0.217765	0.000000	**user**	0.309883	0.172313
exact	0.000000	0.242180	**phone**	0.217765	0.000000	**various**	0.000000	0.242180
find	0.000000	0.242180	**read**	0.217765	0.000000	**wifi**	0.000000	0.242180
identifi	0.217765	0.000000	**record**	0.000000	0.242180			

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

Here, A represents the TF-IDF matrix, U represents documents-to-concepts matrix describing associations between concepts and terms, V represents terms-to-concepts matrix describing associations between documents rooted to various concepts and ∑ represents a diagonal matrix with non-negative real numbers, arranged in descending order. Moreover, these diagonal values represent the relative strength of each concept. A maximum number of concepts (also known as topics) cannot be more than the total number of documents rather its value needs to be adjusted to develop a latent semantic representation of the original matrix. Let's assume d is the total number of documents, t is the total number of terms in all the documents and k is considered as the hyperparameter indicating the number of topics to be extracted from the textual data. A_k is the low-rank approximation of matrix A and can be produced using truncated SVD as follows in Eq. (5):

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

where U_k is the document-to-topic matrix (t × k), V_k is the term-to-topic matrix (d × k), and $\sum_k$ is the topic-to-topic matrix (k × k). To decrease the importance of frequent terms and increase the rare terms in documents, the TF-IDF technique was used and Tab. 6 shows the term frequencies being transformed after applying TF-IDF. To illustrate the text mining done by LSA, consider the same textual data example, SVD operation has to be applied on the TF-IDF matrix. As discussed, each of the k reduced dimensions corresponds to a latent concept which helps to discriminate the documents. To obtain the most significant dimensions, the optimal value for k needs to be adjusted. But for the above example, the value of k is taken to be two. The document loading-matrix and term-loading matrix are shown in Tabs. 7 and 8. As an application of LSA followed by clustering approach will help to identify topic solutions.

Table 7: Term-loading with five latent topics

Terms	Topic 1	Topic 2	Terms	Topic 1	Topic 2	Terms	Topic 1	Topic 2
Applic	0.411164	−0.028694	**like**	0.136800	0.179853	**send**	0.152138	−0.200017
Calendar	0.136800	0.179853	**list**	0.136800	0.179853	**server**	0.152138	−0.200017
Connect	0.152138	−0.200017	**locat**	0.152138	−0.200017	**tower**	0.152138	−0.200017
Contact	0.136800	0.179853	**malwar**	0.302917	0.113620	**track**	0.205582	−0.014347
Daily	0.152138	−0.200017	**misus**	0.136800	0.179853	**uniqu**	0.136800	0.179853
Data	0.205582	−0.014347	**network**	0.152138	−0.200017	**usag**	0.152138	−0.200017
Devic	0.273601	0.359705	**number**	0.136800	0.179853	**user**	0.302917	0.113620
Exact	0.152138	−0.200017	**phone**	0.136800	0.179853	**various**	0.152138	−0.200017
Find	0.152138	−0.200017	**read**	0.136800	0.179853	**wifi**	0.152138	−0.200017
Identify	0.136800	0.179853	**record**	0.152138	−0.200017			

Table 8: Document-loading with two latent topics

	Topic 1	Topic 2
Doc1	0.795922	0.605399
Doc2	0.795922	−0.605399

3 Experimental Results and Findings

Using procedure detailed in Section 2 and weighting scheme as given in Eq. (3), a *n**843 term-document weighted matrix was created for *t* term in *d* document for all *n* documents within *corpus*. This *corpus* would be used for identification of prominent topic solutions within Android malware literature. Initially the dataset of 843 documents had 10076 tokens. After pre-processing, the count was reduced to 1122 tokens. 843 sparse vectors were created with 1122 tokens. The collection of 843 articles on Android malware was then transformed to a singular row of vectors wherein rows of the matrix used as 1122 terms of 843 columns, each vector representing an article on Android malware. This process would provide a unique numeric value to the collection of words as discussed in Tab. 6 for sample documents and in Tab. 10 for actual 843 documents (as proof of concept). This list of words is further used for the creation of matrix containing weights. As mentioned previously, large *corpus* of 843 documents on Android malware resulted into high dimensional TF-IDF matrix which was likely to be noisy and redundant across its many dimensions. Therefore, to uncover the latent structure within words and documents and to identify latent topics within *corpus*, dimensionality reduction step was performed as detailed in subsequent sections below.

3.1 Rank Lowering Using Singular Vector Decomposition

The weighted matrix TF-IDF obtained after preprocessing steps was provided to the SVD to further perform rank lowering. The SVD model $X = U \sum V\text{^}t$ is used to perform matrix X factorization into variables [13,26]. The following terminology is used in Eqs. (6) and (7).

U: Initial rotation

∑: Scaling

V: Final rotation

$$XX^t = \left(U\sum V^t\right)\left(U\sum V^t\right)^t = \left(U\sum V^t\right)\left(V^{tt}\sum{}^tU^t\right) = U\sum V^tV\sum{}^tU^t = U\sum\sum{}^tU^t \quad (6)$$

$$X^tX = \left(U\sum V^t\right)^t\left(U\sum V^t\right) = \left(V^t\sum{}^tU^t\right)\left(U\sum V^t\right) = V\sum{}^tU^tU\sum V^t = V\sum{}^t\sum V^t \quad (7)$$

The mathematical expression XX^t and X^tX provides term-loading and document-loading respectively. $\sum\sum$^t represents the weights of the topics in descending order. The maximum number of topics generated was equal to the number of documents in the *corpus*. For extracting a few topics (k), the topmost k singular values were taken from the matrix $\sum\sum^t$ [27,28]. Text is represented as a matrix of form $X = U\sum V^t$ such that each row stands for unique word and each column represents unique document. Each cell represents the number of occurrences of the word with which it appears in a document. Apply preliminary transformation wherein weights have been assigned describing word importance in particular document w.r.t all other documents. The dimension reduction step had structured the matrices in such a way that words that did not appear originally in some contexts now do appear, at least fractionally. Afterwards, apply SVD which decomposed the original matrix into the product of three other matrices. Term loading XX^t matrix represents terms loaded for a particular topic solution. Each cell contains term weight for a particular topic giving more weightage to that topic solution as per the specified threshold value. Document loading matrix X^tX represents documents loaded for a particular topic solution. Each cell contains document weight for a particular topic giving more weightage to that topic in terms of number of documents loaded for that topic as per the specified threshold value.

3.2 Selecting Optimal Topic Solutions

Optimal topic loadings come from dimensionality reduction. It offers a detailed analysis of obtaining k optimal terms or values from the term matrix produced by it. Selecting an optimal topic value has been difficult because of its requirement to understand and requiring several procedures to obtain favorable value [26]. The five prominent topic loadings for 843 *corpus* of documents is shown in Tab. 9. It is suitable enough to identify trends in Android malware research.

The results in Fig. 1 showed that numerous high loading publications converged to one research area, i.e., “Static Level Monitoring” (T5.2) in the five topic solutions. Static analysis is the most utilized analysis technique for malware analysis; hence it is unsurprising that “Static Level Monitoring” (T5.2) remained to trend research area throughout the year 2009-2019. Results also showed that “Automatic Malware Analysis” (T5.3) and “Hybrid level monitoring” (T5.4) also became a trending research area during the year 2014–2019. “Dynamic level monitoring” (T5.5) also had been a dominant research area in Android malware research.

In the *corpus*, the most common approaches used by researchers to capture security threats in Android ecosystem are based on static level monitoring (39%). Dynamic Level Monitoring (26%), Hybrid Level Monitoring (21%), Automatic Malware Analysis (about 11%), and Application Structure Analysis (3%) were also found to have considerable share in Android malware literature. It is to be noted that from the year 2009, many static technique to detect android malware were proposed [29] and the techniques based on dynamic analysis were first explored by the researchers in 2010 [30].

The ten-topic solution as shown in Fig. 2 uncovered ten trending areas, viz. “Machine Learning Approach” (T10.9), “Dynamic Code Loading” (T10.2), “Dalvik Byte Code Analysis” (T10.6), “Feature-Based Analysis” (T10.10), and “Repackaged App Identification” (T10.3) which were dominant research areas for the time period 2014–2019. However, in Fig. 3 the twenty-topic solution uncovered various trending areas, viz. “Permission-Based Analysis” (T20.5), “Data Flow Tracking” (T20.7), “Context

Monitoring" (T20.16). "Permission-Based Analysis" (T20.5) was a dominant research area throughout the period 2009–2019. However, "Kernel Level Check" (T20.6), "Data Flow Tracking" (T20.7) and other supplementary techniques like "Machine Learning Approach" (T20.12), "Formal Analysis" (T20.20) became trending research areas during the year 2014–2019.

Table 9: Core research areas identified through LSA

Topic No.	Topic label	Loading terms
T5.1	Application Structure Analysis	component, intent, receiver, broadcast, leak, vulnerability, string, class, object, activity, privilege, developer, sensitive, analysis, content, explicit, application, action, android, permission
T5.2	Static Level Monitoring	signature, bytecode, graph, context, dalvik, flow, permission, component, control, library, program, service, method, object, entry, event, field, code, data, path
T5.3	Automatic Malware Analysis	machine, accuracy, dataset, class, performance, positive, family, false, application, similarity, proceeding, experiment, signature, pattern, dynamic, android, vector, static, score, graph
T5.4	Hybrid Level Monitoring	dynamic, analysis, static, cloud, taint, application, instruction, execution, component, sensitive, bytecode, android, library, program, native, object, string, dalvik, class, event
T5.5	Dynamic Level Monitoring	kernel, privilege, escalation, policy, control, enforcement, security, exploit, memory, vulnerability, library, native, context, component, Linux, mechanism, access, resource, sandbox, virtual

Table 10: Transformed term frequencies after TF-IDF generation for 843 documents

Terms	Doc1	Doc2	Doc3	Doc4	Doc5	…..	Doc843
kernel	0.000000	0.000000	0.346366	0.000000	0.589463	**…..**	0.000000
dynamic	0.176043	0.160859	0.170893	0.165134	0.280882	**…..**	0.164157
machine	0.000000	0.000000	0.000000	0.000000	0.000000	**…..**	0.344502
accuracy	0.000000	0.000000	0.000000	0.346553	0.000000	**…..**	0.000000
graph	0.000000	0.000000	0.000000	0.000000	0.000000	**…..**	0.344502
…..	**…..**	**…..**	**…..**	**…..**	**…..**	**…..**	**…..**

3.3 Android Security Research Trends

Considering the twenty-topic solutions, the prominent research trends in android security research are; the distribution of articles clearly interprets, "Machine Learning Approach" (T20.12), "Data Flow Tracking" (T20.7), "Context Monitoring" (T20.16), "Kernel Level Check" (T20.6), emerged as among highly explored topic solutions. This was consistent with the five, ten topic solutions.

The research trend, "Machine Learning Approach" (T20.12) is one of the highly explored topics over the last few years in which android applications are analyzed statically as well as dynamically to collect some set

of features and then make the system learn it for making a decision about unknown sample of malicious applications. Machine learning methods were used in [31–37]. Another significant trend that emerged was "Kernel Level Check" (T20.6) which focuses on the techniques designed to make the kernel level attacks more difficult to execute [4].

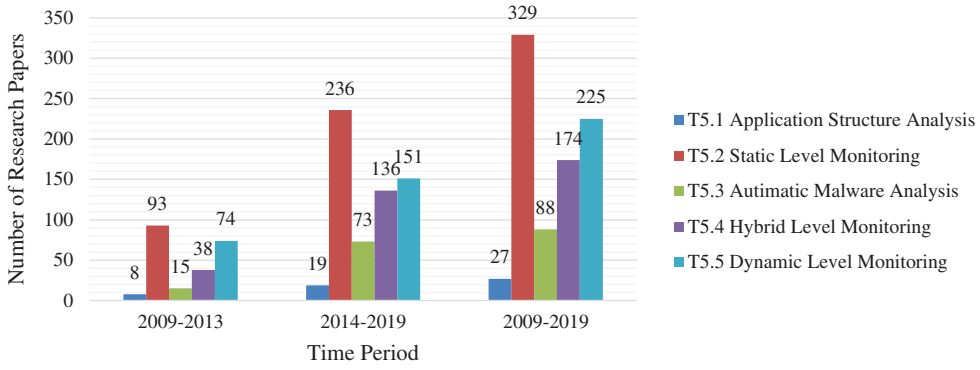
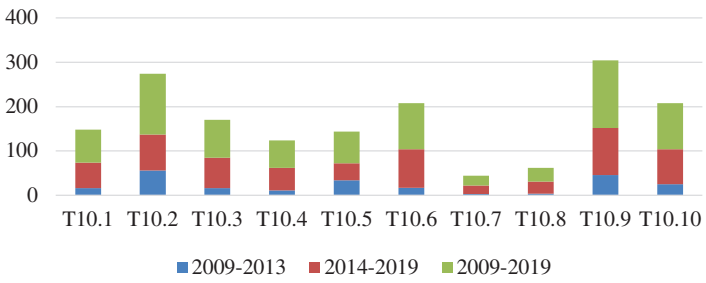
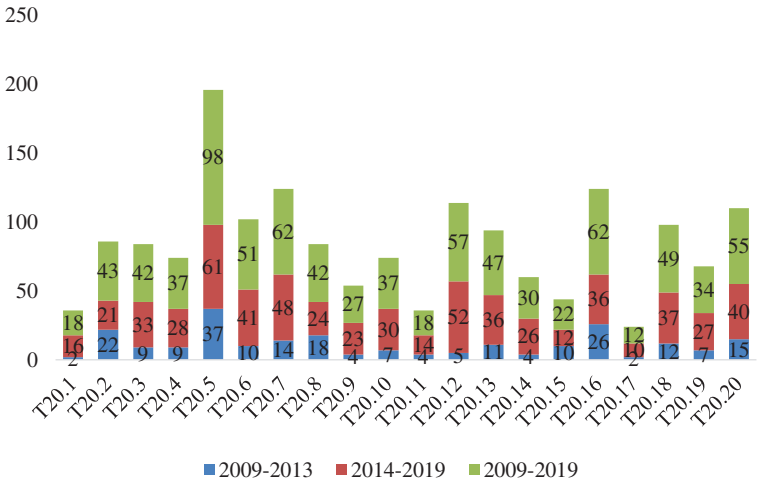


Figure 1: Publication count for five-factor solution during three different time periods



Topic No	Topic Label
T10.1	Emulator Based Analysis
T10.2	Dynamic Code Loading
T10.3	Repackaged App Identification
T10.4	Context Monitoring
T10.5	Component Based Study
T10.6	Dalvik Byte Code Analysis
T10.7	Permission Based Analysis
T10.8	Classification Based on App Behaviour
T10.9	Machine Learning Approach
T10.10	Feature Based Analysis

Figure 2: Ten factor solution during three different time periods



Topic No	Label
T20.1	Obfuscated Code Analysis
T20.2	App Level Analysis
T20.3	Hybrid Analysis
T20.4	Input Matching
T20.5	Permission Based Analysis
T20.6	Kernel Level Check
T20.7	Data Flow Tracking
T20.8	Classification Based on App Behaviour
T20.9	Dynamic Code Loading
T20.10	Emulator Based Analysis
T20.11	Dex File Study
T20.12	Machine Learning Approach
T20.13	Flow Monitoring
T20.14	Component Based Study
T20.15	Syntactic and Semantic Pattern
T20.16	Context Monitoring
T20.17	Feature Based Analysis
T20.18	Dalvik Byte Code Analysis
T20.19	Repackaged App Identification
T20.20	Formal Analysis

Figure 3: Twenty factor solution during three different time periods

3.4 Mapping of Core Research Areas and Research Trends

Tab. 11 shows the mapping of research trends with five core research areas. The mapping is done on the basis of similarity scores. The topics corresponding to the latter were somewhat related to the former and were verified using similarity scores. The similarity scores were calculated as a result of the low and high loading values of topic solutions. The similarity scores present a clear relationship between the core research area and their related trends which also validates the procedures developed to show their semantic connection. The detailed discussion is presented in the following sections.

Table 11: Mapping of core research areas and trends

Topic No.	Five topic labels	Topic No.	Topic label
T5.1	Application Structure Analysis	T20.2	App Level Analysis
T5.2	Static Level Monitoring	T20.5	Permission Based Analysis
		T20.7	Data Flow Tracking
		T20.11	Dex File Study
		T20.14	Component Based Study
		T20.15	Syntactic and Semantic Pattern
		T20.16	Context Monitoring
		T20.17	Feature Based Analysis
T5.3	Automatic Malware Analysis	T20.4	Input Matching
		T20.12	Machine Learning Approach
		T20.19	Repackaged App Identification
		T20.20	Formal Analysis
T5.4	Hybrid Level Monitoring	T20.1	Obfuscated Code Analysis
		T20.3	Hybrid Analysis
		T20.9	Dynamic Code Loading
		T20.10	Emulator Based Analysis
		T20.13	Flow Monitoring
		T20.18	Dalvik Byte Code Analysis
T5.5	Dynamic Level Monitoring	T20.6	Kernel Level Check
		T20.8	Classification Based on App Behaviour

3.5 Mapping of Core Research Areas and Trends

This section details the mapping of top five topic labels with top twenty topic solutions. Initial mapping of Topic (T5.1) "Application Structure Analysis" has a clear overlap with "App Level Features" (20.2). It uncovered the use of metadata and features of an Android application to detect and analyze Android malware. Metadata can be characterized as the displayed information which is available before downloading and installing the Android application, e.g., required permissions, description, version, last updated, rating, number of installations, developer information. This trend was seen in the project named WHYPER [38]. Among all five core research areas, results revealed that (T5.2) "Static Level Monitoring" has been investigated the most. Tab. 11 demonstrates that out of twenty topic solutions,

seven research trends converged to core research area T5.2. In "Permission-Based Analysis" (T20.5), permissions played a vital role for investigation of malicious content, as most activities (e.g., a collection of APIs) require specific consents keeping in mind the end goal to be achieved [3]. Another research trend that emerged in this area is the "Dex file study" (T20.11). Though dex files are difficult to humans to understand, yet it is prudent to study dex files for mitigating malware attacks. To identify malicious code segments, researchers first decompile the dex code into more conceivable formats such as assembly, smali, dalvik bytecode, source code, jar, jimple or java bytecode [39]. Automatic Malware Analysis (T5.3), primarily explored "Input Matching" (T20.4), "Repackaged App Identification" (T20.19), "Formal Analysis" (T20.20), and "Machine Learning Approach" (T20.12). All such techniques were related to automated identification of Android malware. To achieve the effectiveness and scalability of Android malware detection, the trend "Machine Learning Approach" (T20.12) came into light during the time period 2014-2019. Another research trend that emerged was "Repackaged App Identification" (T20.19). Repackaging is one of the popular techniques being used by malware authors to generate fraudulent repackaged applications [40]. The year 2012 was a high time when malware started evolving and a need of hybrid techniques for malware monitoring was felt (T5.4). The rampant utilization of techniques such as dynamic code loading, native code, java reflection, and code coverage by malware community was limiting the significance of static and dynamic detection methods. Researchers were in need of a robust solution to analyze and mitigate the impact of malware. The trends like "Dynamic Code Loading" (T20.9), "Dalvik ByteCode Analysis" (T20.18), "Obfuscated Code Analysis" (T20.1) which emerged during 2014–2019 played the vital role in introducing/establishing hybrid approaches to combat against the evolving malware. The past hybrid projects such as "AppIntent [41], SmartDroid [42], IntelliDroid [43], Harvester [44], A5 [45] witness the curtailment of smart tactics and had produced precise detection results. Dynamic Level Monitoring (T5.5) is also a prominent trend that signifies the requirement of runtime analysis of Android applications. The trend "Kernel Level Check" (T20.6) uncovered the use of system calls for effective malware detection. Android Linux kernel has more than 250 system calls in common [46].

4 Discussion

The results of this study revealed that "Static Level Monitoring" (T5.2) had been proved to be the most widely investigated topic in Android malware research. The studies related to static analysis majorly focus on network addresses, data flow tracking, control flow graphs, string matching, permissions, dex files, context, and intents. Studies also focused on behavioral and structural analysis to extend its coverage for advanced malware applications. Kernel-level analysis, API call monitoring, taints were its major highlights. Researchers identified that the results of the combined effects of structural and behavioral features produce richer and robust analysis. Numerous studies support hybrid techniques to detect destructive payloads. Though automation in malware analysis and analysis with supplementary techniques were comparatively less in number yet are effective enough to produce promising results.

To maintain the effective interpretations and comparisons among the topics; change of focus over two time-frames 2009–2013 and 2014–2019 were observed. It depicts the paradigm shift from time window 2009–2013 to 2014–2019. Machine learning approaches were found to be effective among other competitive approaches to detect Android malware. These approaches are well explored and promising during the time 2014–2019. The trend kernel-level check greatly influenced the research community during 2014–2019. Applications were inspected at the kernel level in order to understand their real-time behavior. Detection of piggybacked application through sensitive graph analysis/data tracking followed by usage of machine learning algorithms was widely studied during 2014–2019. Application permissions have remained the topmost static features to detect Android malware. It has been widely investigated during 2009–2019, as it poses as the first barrier to the malware authors. To activate certain events in an

Android ecosystem, some specific permission should be declared in the manifest file. Due to the ability of malicious applications to hide their actual behavior through the user interface, it became cumbersome to analyze all possible paths or inputs, while performing sandbox analysis. During 2014–2019, smart interaction solutions came into light which focused on generating activity, and function call graph using static analysis and exploring paths using dynamic analysis.

5 Conclusion

Reviewing literature manually may result in biased and incomplete inferences. This work systematically analyses a large *corpus* of 843 research articles on Android security using an information modeling technique. The major outcomes of this study are the analytical interpretation of five core research areas and twenty substantial research themes. The results suggest that Static Level Monitoring is the most widely investigated topic in Android malware analysis and detection. Behavioral analysis in addition to structural is a must to extend the coverage for advanced malware and vulnerable applications. Kernel-level analysis, API call monitoring, taints are also important indicators of malware behavior. Research trends indicate that results of the combined effects of structural and behavioral features produce richer and robust analysis.

This investigation also identifies new future dimensions for researchers. The results of this study will help others to choose their areas of interest for their potential research along with the associated research trend. The most impacting factor of this work lies in that researchers can apply the same methodology in any other research fields by little or almost no changes.

Funding Statement: This work was supported by National Research Foundation of Korea-Grant funded by Korean Government (Ministry of Science & ICT)-NRF-2020R1A2B5B02002478 through Dr. Kyung-sup Kwak.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] H. Anwa, D. Pfah and S. N. Srirama, "Evaluating the impact of code smell refactoring on the energy consumption of android applications," in *Proc. 45th Euromicro Conf. on Software Engineering and Advanced Applications*, Chalkidiki, Greece, pp. 82–86, 2019.

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

[3] W. Zhou, Y. Zhou, X. Jiang and P. Ning, "Detecting repackaged smartphone applications in third-party android marketplaces," in *Proc. 2nd ACM Conf. on Data and Application Security and Privacy*, San Antonio, USA, pp. 317–326, 2012.

[4] M. Xu, C. Song, Y. Ji, M. W. Shih, K. Lu *et al.,* "Toward engineering a secure android ecosystem: A survey of existing techniques," *ACM Computing Surveys*, vol. 49, no. 2, pp. 1–47, 2016.

[5] 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.

[6] B. Rashidi and C. Fung, "A survey of android security threats and defenses," *Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications*, vol. 6, no. 3, pp. 3–35, 2015.

[7] 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.

[8] 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.

[9] L. See, P. Mooney, G. Foody, L. Bastin, A. Comber *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, pp. 55, 2016.

[10] 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, pp. 195, 2017.

[11] M. B. W. Wolfe and S. R. Goldman, "Use of latent semantic analysis for predicting psychological phenomena: Two issues and proposed solutions," *Behavior Research Methods, Instruments, & Computers*, vol. 35, no. 1, pp. 22–31, 2003.

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

[13] 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.

[14] V. Gandhi and J. Singh, "An automated review of body sensor networks research patterns and trends," *Journal of Industrial Information and Integration*, vol. 18, 100132, 2020.

[15] J. N. De Boer, A. E. Voppel, M. J. H. Begemann, H. G. Schnack, F. Wijnen *et al.,* "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.

[16] 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.

[17] 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.

[18] C. H. Papadimitriou, P. Raghavan, H. Tamaki and S. Vempala, "Latent semantic indexing: A probabilistic analysis," *Journal of Computer and System Sciences*, vol. 61, no. 2, pp. 217–235, 2000.

[19] 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, 113401, 2020.

[20] J. S. Drysdale, C. R. Graham, K. J. Spring and L. R. Halverson, "An analysis of research trends in dissertations and theses studying blended learning," *Internet and Higher Education*, vol. 17, pp. 90–100, 2012.

[21] H. J. Kang, C. Kim and K. Kang, "Analysis of the trends in biochemical research using latent Dirichlet allocation (LDA)," *Processes*, vol. 7, no. 6, pp. 379, 2019.

[22] Y. M. Kim and D. Delen, "Medical informatics research trend analysis: A text mining approach," *Health Informatics Journal*, vol. 24, no. 4, pp. 432–452, 2018.

[23] Y. Lu and C. Zhai, "Opinion integration through semi-supervised topic modeling," in *Proc. 17th Int. Conf. on World Wide Web*, Beijing, China, pp. 121–130, 2008.

[24] Y. Bengio, R. Ducharme, P. Vincent and C. Jauvin, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[25] J. Singh and N. Modi, "Use of information modelling techniques to understand research trends in eye gaze estimation methods: An automated review," *Heliyon*, vol. 5, no. 12, e03033, 2019.

[26] T. F. Abidin, B. Yusuf and M. Umran, "Singular value decomposition for dimensionality reduction in unsupervised text learning problems," in *Proc. 2nd Int. Conf. on Education Technology and Computer*, Shanghai, China, vol. 4, pp. 422–426, 2010.

[27] 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.

[28] A. A. Kuandykov, S. B. Rakhmetulayeva, Y. M. Baiburin and A. B. Nugumanova, "Usage of singular value decomposition matrix for search latent semantic structures in natural language texts," in *Proc. 54th Annual Conf. of the Society of Instrument and Control Engineers of Japan*, Hangzhou, China, pp. 286–291, 2015.

[29] J. Kim, Y. Yoon, K. Yi, J. Shin and S. W. R. D. Center, "Scandal: Static analyzer for detecting privacy leaks in android applications," *Mobile Security Technologies*, vol. 12, no. 110, pp. 1, 2012.

[30] W. Enck, P. Gilbert, S. Han, V. Tendulkar, B. G. Chun *et al.,* "Taintdroid: An information-flow tracking system for realtime privacy monitoring on smartphones," *ACM Transactions on Computer Systems*, vol. 32, no. 2, pp. 5:1–5:29, 2014.

[31] H. S. Ham and M. J. Choi, "Analysis of android malware detection performance using machine learning classifiers," in *Proc. Int. Conf. on ICT Convergence*, Jeju Island, South Korea, pp. 490–495, 2013.

[32] S. Y. Yerima, S. Sezer and I. Muttik, "Android malware detection using parallel machine learning classifiers," in *Proc. 8th Int. Conf. on Next Generation Mobile Apps Services and Technologies*, Oxford, UK, pp. 37–42, 2014.

[33] N. Milosevic, A. Dehghantanha and K. K. R. Choo, "Machine learning aided android malware classification," *Computers and Electrical Engineering*, vol. 61, pp. 266–274, 2017.

[34] W. C. Wu and S. H. Hung, "DroidDolphin: A dynamic android malware detection framework using big data and machine learning," in *Proc. Conf. on Research in Adaptive and Convergent Systems*, Towson, Maryland, USA, pp. 247–252, 2014.

[35] B. Amos, H. Turner and J. White, "Applying machine learning classifiers to dynamic android malware detection at scale," in *Proc. 9th Int. Wireless Communications and Mobile Computing Conf.*, Sardinia, Italy, pp. 1666–1671, 2013.

[36] M. Raza, M. Awais, K. Ali, N. Aslam, V. V. Paranthaman *et al.,* "Establishing effective communications in disaster affected areas and artificial intelligence-based detection using social media platform," *Future Generation Computer Systems*, vol. 112, pp. 1057–1069, 2020.

[37] G. Ali, A. Ali, F. Ali, U. Draz, F. Majeed *et al.,* "Artificial neural network based ensemble approach for multicultural facial expressions analysis," *IEEE Access*, vol. 8, pp. 134950–134963, 2020.

[38] R. Pandita, X. Xiao, W. Yang, W. Enck and T. Xie, "WHYPER: Towards automating risk assessment of mobile applications," in *Proc. 22nd USENIX Security Sym.*, Washington, DC, USA, pp. 527–542, 2013.

[39] K. Tam, A. Feizollah, N. B. Anuar, R. Salleh and L. Cavallaro, "The evolution of android malware and android analysis techniques," *ACM Computing Surveys*, vol. 49, no. 4, pp. 76:1–76:41, 2017.

[40] Y. Zhou and X. Jiang, "Dissecting android malware: Characterization and evolution," in *Proc. IEEE Sym. on Security and Privacy*, San Francisco, California, USA, pp. 95–109, 2012.

[41] Z. Yang, M. Yang, Y. Zhang, G. Gu, P. Ning *et al.,* "Appintent: Analyzing sensitive data transmission in android for privacy leakage detection," in *Proc. ACM SIGSAC Conf. on Computer & Communications Security*, Berlin, Germany, pp. 1043–1054, 2013.

[42] C. Zheng, S. Zhu, S. Dai, G. Gu and X. Gong, "Smartdroid: An automatic system for revealing UI-based trigger conditions in android applications," in *Proc. 2nd ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, Raleigh, North Carolina, USA, pp. 93–104, 2012.

[43] V. Afonso, A. Bianchi, Y. Fratantonio, A. Doupe, M. Polino *et al.,* "Going native: Using a large-scale analysis of android apps to create a practical native-code sandboxing policy," in *Proc. 23rd Annual Network and Distributed System Security Sym*, San Diego, California, USA, pp. 1–15, 2016.

[44] S. Rasthofer, S. Arzt, M. Miltenberger and E. Bodden, "Harvesting runtime values in android applications that feature anti-analysis techniques," in *Proc. 23rd Annual Network and Distributed System Security Sym.*, San Diego, California, USA, pp. 21–24, 2016.

[45] T. Vidas, J. Tan, J. Nahata, C. L. Tan, N. Christin *et al.,* "A5: Automated analysis of adversarial android applications," in *Proc. 4th ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, Scottsdale, Arizona, USA, pp. 39–50, 2014.

[46] I. Burguera, U. Zurutuza and S. Nadjm-Tehrani, "Crowdroid: Behavior-based malware detection system for Android," in *Proc. 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices*, Chicago, Illinois, USA, pp. 15–26, 2011.