Computer Systems Science & Engineering
DOI: 10.32604/csse.2023.028227
Article

Tech Science Press

Investigation of Android Malware with Machine Learning Classifiers using Enhanced PCA Algorithm

V. Joseph Raymond[1,2,*] and R. Jeberson Retna Raj[1]

[1]Department of Computer Science and Engineering, Sathyabama Institute of Science and Technology, Chennai, 600119, Tamilnadu, India
[2]School of Computing, SRM Institute of Science and Technology, Chennai, 603203, Tamilnadu, India
*Corresponding Author: V. Joseph Raymond. Email: josephrv@srmist.edu.in
Received: 05 February 2022; Accepted: 30 March 2022

Abstract: Android devices are popularly available in the commercial market at different price levels for various levels of customers. The Android stack is more vulnerable compared to other platforms because of its open-source nature. There are many android malware detection techniques available to exploit the source code and find associated components during execution time. To obtain a better result we create a hybrid technique merging static and dynamic processes. In this paper, in the first part, we have proposed a technique to check for correlation between features and classify using a supervised learning approach to avoid Multicollinearity problem is one of the drawbacks in the existing system. In the proposed work, a novel PCA (Principal Component Analysis) based feature reduction technique is implemented with conditional dependency features by gathering the functionalities of the application which adds novelty for the given approach. The Android Sensitive Permission is one major key point to be considered while detecting malware. We select vulnerable columns based on features like sensitive permissions, application program interface calls, services requested through the kernel, and the relationship between the variables henceforth build the model using machine learning classifiers and identify whether the given application is malicious or benign. The final goal of this paper is to check benchmarking datasets collected from various repositories like virus share, Github, and the Canadian Institute of cyber security, compare with models ensuring zero-day exploits can be monitored and detected with better accuracy rate.

Keywords: Zero-day exploit; hybrid analysis; principal component analysis; supervised learning; smart cities

1 Introduction

The smartphone generation started later this generation and have achieved the peak in the recent past years, especially we focus more on Android stack used by the end user's all over the world at different levels for different categories, some like to more time on Social Applications, some on Business Applications and others in Gaming and Media Applications and to certain extend research activities and projects done with Android platform. The framework used in Android API (Application Programming Interface) is more vulnerable because of open-source nature and stack can be modified henceforth resulting in security breach when distributed across third party and the major source of advertisement. The attacker is keener on finding useful information from the victim by sending fake Short Messaging Service (SMS), recording, and access call logs, dropping malicious activities using payload with the help of Trojan and Backdoor mechanisms [1]. The process of finding and detecting android malicious applications traditionally is done by static and dynamic analysis. The researchers explore vulnerabilities using a platform like Kali Linux and emulators like Genymotion or Android Studio henceforth achieving the target by identifying the threat. The first approach is the static analysis where the payload will be analyzed using tools like OllyDbg, Integrated Disassembler (IDA) Pro, Java Debugger (JAD) Debugger and checking the internal functionalities and finding the weakness in code [2]. The second approach is executing the Android Application Package (APK) application in sandboxing environment and monitoring events with the help of log cat applications. The combination of static and dynamic analysis mechanisms called as hybrid approach gives better accuracy and result in detecting the malicious payload. There is a multicollinearity problem that can come in machine learning classifiers because of correlational between features which can affect the performance resulting in producing a low score. According to Yongshuang Liu et al. [3], understanding the semantics of an application will find information about the API Calls. There are a few permissions that are protected available in the manifest file. We have to give importance by combining API calls and Permissions resulting in better accuracy. There is a special kind of approach called Principal Component Analysis (PCA) is a statistical method that helps us in converting the uncorrelated data to correlated data. This is unsupervised learning where we understand features and their interrelations. PCA helps us in improving the logistic regressions. Nowadays most of the developed and developing countries going for smart cities where most people might use Smartphone's more specifically android based lead to more vulnerability and threat. This can cause a major issue for IoT (Internet of Things) concerning security.

In this paper, we have added these sections focusing on related works, understanding existing methodologies, proposed work, implementation and results, conclusion, and further directions can be done for research work. In the upcoming sections, we discuss static, dynamic, and hybrid analysis of android applications applying reverse engineering using disassembly tools. The results obtained from sandboxing environment are taken along with dataset repositories collected from various sources for creating the model. One of the major issues inside the dataset is the dependency between variables that can lead to Multicollinearity problem and henceforth using principal component analysis we handle that issue in the proposed system we build a classifier based on sensitive android permission for handling and decision making on zero-day exploits and recent threats which is not updated with most of the antivirus scanning tool. In the upcoming chapter's, we discuss about static and dynamic analysis under related work. The study about data set is done with various machine learning classifiers and its terminologies. In the proposed work we implement enhanced PCA to overcome multicolinearity issue shows the novelty of research done and with future conclusions.

2 Related Work

2.1 System Architecture

The application classified as the label and unknown application performed a dynamic analysis based on behaviour extraction and from there generate feature extraction using vector template and feed into the base classifier. The probability prediction is based on learning and stacking and decides whether the given application is Benign or Malicious as shown in the Fig. 1.

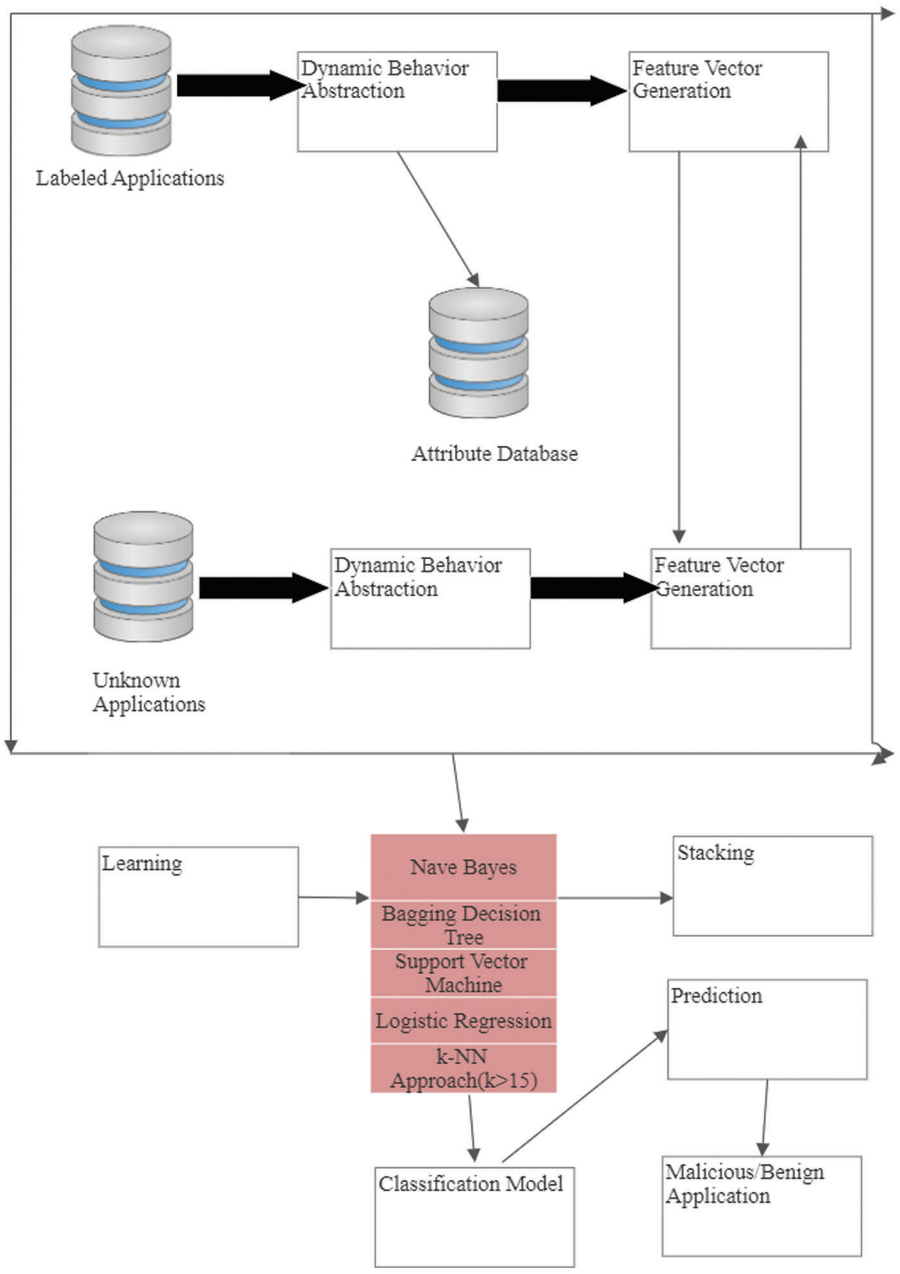


Figure 1: Proposed framework

2.2 Static Analysis

Static analysis is the process of analyzing a binary without executing it. It is easiest to perform and allows you to extract the metadata associated with the suspect binary. Static analysis might not reveal all the required information, but it can sometimes provide interesting information that helps in determining where to focus your subsequent analysis efforts to extract useful information from malware binary. According to V. Joseph Raymond et al. [2], static analysis works based on code segments and finding out vulnerable threats. We have used tools like OllyDbg and IDA pro for exploring the implementation work and finding out modules that can lead to a breach of security. When we discuss in terms of Android application in static analysis mainly focus on permissions available in the manifest file. In, Arp et al. [3] suggested the usage of Drebin gather static features like intent filters, permissions and app components. The output of static features is given as input for logistic regression and using the classification approach decides whether the APK application is malicious or benign [1]. In, Shahriar et al. [4] implemented a machine learning detection approach using trained probabilistic features taking frequently used permissions. In Yerima et al. [5] detection of android malware based on decompiling source code using ensemble techniques. In MaMadroid [6], the detection of malicious applications is done based on abstracted API calls using the Markov chain technique.

2.3 Dynamic Analysis

Dynamic Analysis is the process of executing the suspect binary in an isolated environment and monitoring its behavior. This analysis technique is easy to perform and gives valuable insights into the activity of the binary during its execution. This analysis technique is useful but does not reveal all the functionalities of the hostile program. According to V. Joseph Raymond et al. [2], dynamic analysis is done by executing the android application in Santoku Operating System (OS) and android emulator. In that work, code reverse engineering is done and injected malicious payload in the benign application and monitored activities using log cat tools. The process is done sandboxing environment and ensured a safe home environment. In, Crowdoid [7] detection is based on a cloud architecture by collecting system call events using a K-mean clustering algorithm. In, Yu et al. [8] detection is done based on system call traces applying Support Vector Machine (SVM) and naive Bayes classifiers in [9,10], Dmjsevac et al. by understanding the nature of system call decisions are made for detection also considering the occurrence of repeated requested to the kernel. In [11], Dash et al. suggested reconstructing high-level behavior from an application. The demerit point with dynamic analysis is lack of concentration on code can impact the quality of work [12]. The merit points on both static and dynamic analysis are based on the target and the exploitation. Fig. 2 shows how feature selection helps in reducing columns in choosing whether the payload is malicious or not [13,14]. The goal of static and dynamic analysis is to ensure that malware payload detection done with at most care of host system not getting affected.

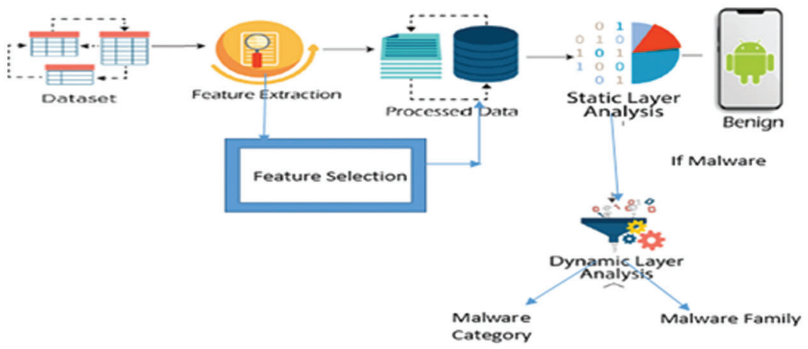


Figure 2: A hybrid approach for malware detection

2.4 Hybrid Analysis

Cuckoo Sandboxing [15] is an automated malware analysis system that helps in finding suspicious activities in an isolated environment. Malware in the modern world is used as a cyber weapon by attackers used on various platforms. The goal is to analyze different malicious files, trace API calls, dump and analyze network traffic and perform advanced memory analysis using YARA. In [2,16] AspectDroid, monitoring is done by injecting payload. In [17], Onwuzurike et al. suggested a hybrid mechanism using

Markov chain sequences. In Samadroid [18] they have done hybrid analysis in two steps static features gathered from APK file shown in Fig. 3.

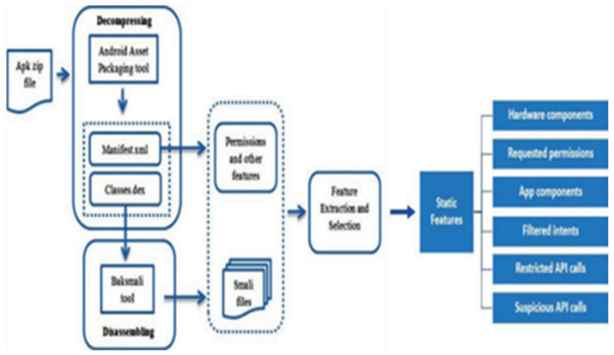


Figure 3: Static features extracted from APK Application

In the second phase, the hybrid approach is done with data processing, cleaned, and given input to machine learning classifiers such as logistic regression and target is considered when both approaches are termed as malicious. It will be less inaccurate in the dynamic analysis [19]. In StormDroid [20] features collected from sensitive permissions like Short Messaging Service (SMS), intent filters tec. are collected as given as input as machine learning classifier. The malware detection process is happening by taking conditional dependency of logistic regression classifiers based on the features shown in above figure [21]. There is a possibility of data overfitting that can come classification algorithm we propose a reduced feature selection find the correlation between them apply principal component analysis and make the model which will discuss in the upcoming sections [22].

2.5 Dataset Information

The experimental setup is done by taking the dataset from payloads collected from virus share, Github, and Canadian Institute for Cyber security [23]. We collected 1403 samples and used 1100 for training and 303 samples for validation implemented machine algorithm applications using Google Colab as shown in Tab. 1.

Table 1: Dataset information

Details	Malware Family	Number of samples
CCCS-CIC-AndMal2020	Adware	102
CICMalDroid 2020	Backdoor	105
Darknet 2020	Dropper/Trojan	204
Investigation of the Android Malware (CIC-InvesAndMal2019)	File Infector	167
Android Malware Dataset (CIC-AndMal2017)	Ransomware	107
Android Adware and General Malware Dataset (CIC-AAGM2017)	Scare ware	106
ISCX Android Botnet dataset 2015	SMS Attack	102
ISCX Android Validation dataset 2014	Spyware	106
Virus Share 2021	Zero-Day	100
CCCS-CIC-AndMal2020	Benign	304

3 Existing Methodologies

In the existing approach, our paper we are discussing supervised and unsupervised learning without feature reduction.

3.1 Logistic Regression

In the traditional approach, under classification, the primary classifier is linear or logistic regression based on the need of the user for training the model. The dataset is made by collecting datasets from repositories along with the own result obtained from the hybrid analysis. In the first part, logistic regression is supervised classification where our target variable is a discrete value stating whether the application is malicious or not called binary classification. This model uses the sigmoid function given below in figure.

$$g(z) = \frac{1}{1 + e^\wedge{-z}} \tag{1}$$

The logistic regression is categorized based on either low/high or high/low precision-recall where in the first case we reduce the false negative for sensitive data and in the latter reduce the false positives. The presence of android malware comes under binomial logistic regression where the presence is treated as '1' and not present as '0'.

$$\mathrm{y} = \{0,\ \text{if fail and } 1,\ \text{if pass}\} \tag{2}$$

The data has 'm' feature variables and 'n' observations and the matrix is represented as shown in figure below.

$$\mathrm{X} = \begin{pmatrix} 1\,x11 & \cdots & x1m \\ \vdots & \ddots & \vdots \\ 1\;\; xn1 & \cdots & xnp \end{pmatrix} \tag{3}$$

We can define conditional probabilities for two labels (0 and 1) for the $i^{th.}$

$$\mathrm{P}\ (\mathrm{yi} = 1 \mid \mathrm{xi};\ \beta)\ = \mathrm{h}\ (\mathrm{xi}) \quad \mathrm{P}\ (\mathrm{yi} = 0 \mid \mathrm{xi};\ \beta)\ = 1 - \mathrm{h}\ (\mathrm{xi}) \tag{4}$$

Here, y and h(x) present the vector and predicted response. Xj representing the observation values of the j$^{\text{th}}$ feature.

$$\beta^j = \beta j - \alpha \tag{5}$$

The learning is defined from α and the value is set explicitly. The merits of using this approach are we need not take a learning rate, are always quick in execution, and get an appropriate numerical response. The demerits are a bit complex and look more like black-box testing [24]. The important points to be noticed are independent variables and non -linear transformation. The dependent variables need not be normally distributed based on the binomial distribution. The homogeneity and errors need not be focused on in this type of approach based on large sample approximations. Tabs. 2–4 shows the selection of system call, API, and permission for detection of android malware. In this paper, we have created a dataset ensuring system calls, API calls, and permissions with recent exploits and zero-day vulnerabilities. The next step is the creation of a model using supervised learning [25].

Table 2: Selection of system calls for malware detection

ID	System Call	ID	System Call	ID	System Call
1	writev	11	pread	21	close
2	Unlink	12	unmask	22	lseek
3	Socket	13	bind	23	connect
4	recvfrom	14	write	24	ioctl
5	readv	15	chdir	25	execve
6	read	16	sendto	26	dup
7	Open	17	Rename	27	fchown
8	Mkdir	18	access	28	Chmod
9	Fcntl	19	Recvmsg	29	sendmsg
10	epoll	20	dup2	30	fchown

Table 3: Selection of permission for malware detection

ID	API CALLS	ID	API CALLS
1	_WRITE_SMS	18	_PROCESS_OUTGOING_CALLS
2	_WRITE_SETTINGS	19	_MODIFY_PHONE_STATE
3	_WRITE_HISTORY_BM	20	_INTERNET
4	_WRITE_EXTERNAL_STORAGE	21	_INSTALL_PACKAGE
5	_WRITE_CONTACTS	22	_HARDWARE_TEST
6	_WRITE_APN_SETTINGS	23	_HARDWARE_TEST
7	_VIBRATE	24	_GET_ACCOUNTS
8	_USE_CREDENTIALS	25	_FACTORY_TEST
9	_SEND_SMS	26	_EXPAND_STATUS_BAR
10	_RESTART_ PACKAGE	27	_DIABLE_KEYGUARD
11	_RECEIVE_SMS	28	_DEVICE_POWER
12	_RECEIVE_BOOT_CMD	29	_CHANGE_WIFI_STATE
13	_READ_SMS	30	_CHANGE_NETWORK_STATE
14	_READ_PHONE_STATE	31	_CALL_PHONE
15	_READ_LOGS	32	_ACCESS_NETWORK_STATE
16	_READ_EXTERNAL_STORAGE	33	_ACCESS_LOCATION
17	_READ_CONTACTS	34	_ACCESS_GPS

Table 4: Selection of API calls for malware detection

ID	API CALLS	ID	API CALLS
1	setSerialNumber	18	getMethod
2	sendTextmessage	19	getMessage
3	RequestFocus	20	getLongitude
4	loadClass	21	getLoaction
5	killProcess	22	getLineNumber
6	isProviderEnabled	23	Getlatitude
7	getWifiState	24	getInputStream
8	getSubscriberID	25	getDisplayAddress
9	getSIMSerialNumber	26	getDeviceID
10	getSimOperatorName	27	getCredential
11	getSession	28	getCookies
12	getPackageName	29	getClassLoader
13	getPackageInfo	30	getCertStatus
14	getOutputStream	31	getAppPackageName
15	getNetworkType	32	exec
16	getNetworkOperator	33	CreateFromPdu
17	getMsgBody	34	abortBroadCast

3.2 Naive Bayes Classifiers

This approach collection of algorithms where every pair classification is done independently, we divide into first feature matrix second response vector. The first part contains dependent features and later contains prediction in simple terms output. The equations given below explains the working principle of the classifier.

$$\mathrm{P(X|Y)} = \frac{P(Y|X)P(X)}{P(Y)} \tag{6}$$

where X and Y are events and checking whether P(Y)? 0 considering the probability of occurrence of X assuming Y is true termed as evidence. The prior and posterior probability have to be monitored. In our paper, we have used a Gaussian classifier. The demerit with this approach model will assume ‘0’ as output in case the dataset having errors or missing values.

3.3 Bagging Decision Tree Classifiers

This approach ensembles with meta-estimator based on voting make a prediction, works similar to black-box testing where the system is based on input and output comparing random dataset with actual entities. The merit of this approach will reduce overfitting taking into account variance by removing multiple entries of the same record. The implementation is done by resampling taking instance and present multiple times will not be considered.

3.4 Support Vector Machine Classifiers

This approach is non-linear where we map high dimensional features with input data set. The outcome is non-probabilistic binary classification. The optimization of linear discriminant represents perpendicular distance. This classifier works on black-box testing where the input training data is compared with the output label and achieves the result.

3.5 K-Nearest Neighbours Classifiers

In the second part of the experiment, we have implemented using K-Nearest Neighbours supervised machine algorithm one of the most essential classification algorithm mainly used for intrusion detection part of cyber security. This algorithm can be used for real-time data and henceforth is mostly suitable for hybrid analysis. Here we classify the data sets identified with the help of attributes.

3.6 Evaluation Measures and Experimental Setup

The table shown above gives us about selected permissions, selected API (Application Programming Interface) calls and selected system calls for android malware detections considered for our dataset created by exploring through hybrid analysis. We can be considered different categories of APK files like social engineering app, banking, and financial app, games, and sports app. Media app, educational app, etc. and ensure to take around 800 samples with a combination of malware and benign app from various repositories like Canadian university of cyber security, virus share, virus total, Drebin, MalDrozer, and Android Tracker and gathered recent 2020 Applications [26]. The dataset is created by taking features from the above table and label deciding whether the application is malicious or not. We have used the google colab platform for implementing the logistic regression using the sklearn kit. We used also used Numpy and Pandas packages for implementing the mathematical approach. We have used Seaborn packages for visualizing the accuracy of the model. The selection of input and target label is done by choosing all features as 'X' for the training set and y as the label showing whether the application is malware or benign. The usage of label encoder for y ensuring the dataset can be understood by the interpreter before fitting the model. The next step is to split the training set into test and train data considering 80–20 ratio and assigning random states as zero. The data is transformed into a scalar feature before choosing the classifier [27].

From the above formula [, True Positives (TP) results in actual prediction same as the expected outcome. False Positives [FP] actual prediction not as the expected outcome. True Negative [TN} were predicted not in YES but not in it. Finally, False Negative [FN} predicted not in Yes but actually in it. We can be considered the same dataset as used for the earlier approach and how this essential approach can give us better accuracy than the previous model. It is purely based on the number of points from the data set that creates the training model. We have implemented using the google colab platform importing all the necessary packages. The first step is to fix the 'X' and y variables for the training set and output label. Here also we can use a label encoder for predicting the label to be malicious or not. The scalar features are transformed for the input training data set. We split the training data set into train and test based on 80–20 approach before we fit it into the model. We apply KNeighbors Classifiers and predict the accuracy score which results in both classification and regression problems. We have to focus more on the scale of variables and the distance between the observations. First, we find the K value so that compare it with the error rate assuming n_neigbors as "I" ranging from 1 to 40. This shows Error Rate *vs*. K Value as shown in Fig. 5.

From the above graph, we can see that the error rate tends to get slower after k > 15, so we fix n_neigbors as '15' and get the accuracy rate as 0.76 as shown in Figs. 4 and 5. The existing approach is compared with Naïve Bayes, Bagging Decision Tree, Support Vector Machine, logistic regression, and k-NN classifiers along with linear regression as shown in Tab. 5. The demerit point with the existing approach is the Multicollinearity problem where independent variables are highly correlated with each other than the dependent variables. To overcome this issue researches have been done by tuning the performance of machine learning classifier like using augmented Naïve Bayes approach. But in our proposed approach in the next section, we will be implementing principle component analysis to perform feature reduction find the correlation between dependent and independent variables on our data set, and compare performance with the existing approach [28].

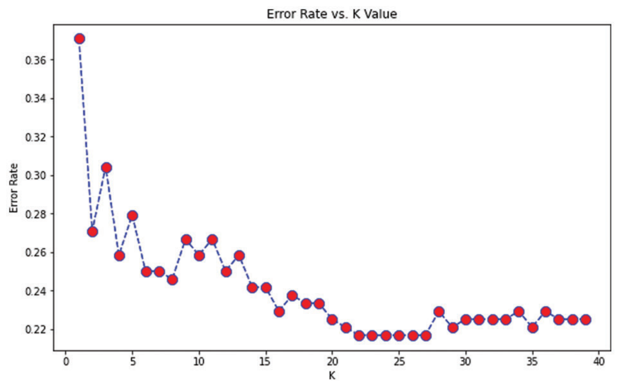


Figure 4: Error rate in k-NN approach

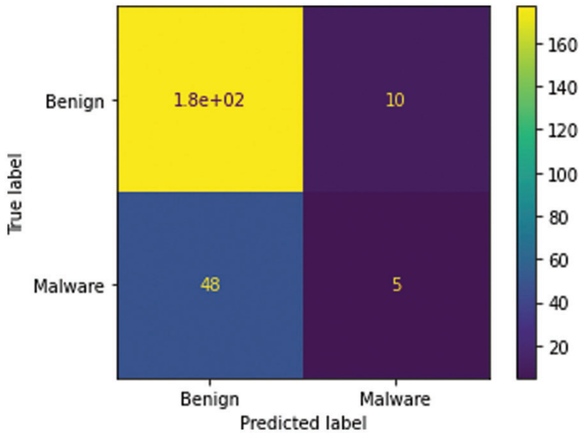


Figure 5: Visualization of accuracy score

Table 5: Binary malware classification using machine learning classifiers without PCA

Dataset	Accuracy (%)	F-Score	Recall	Precision
Naive Bayes	76%	0.70	0.76	0.74
Bagging Decision Tree	68%	0.69	0.68	0.66
Support vector machines	76%	0.69	0.76	0.74
Logistic Regression	69%	0.65	0.69	0.64
k-NN Approach (K = 15)	**76%**	**0.70**	**0.76**	**0.69**

4 Proposed Work and Methodologies

The proposed approach implemented with Principal Component Analysis (PCA) is one of the best feature extraction techniques well suited for a hybrid approach for detecting whether the android application is malicious or not. This approach helps in lowering the dimensions giving focus to very important and critical attributes from the training data set also helps in finding linear combinations which can countermeasure Multicollinearity problems. The flowchart below given shows how the enhanced PCA can reduce search time in Fig. 6.

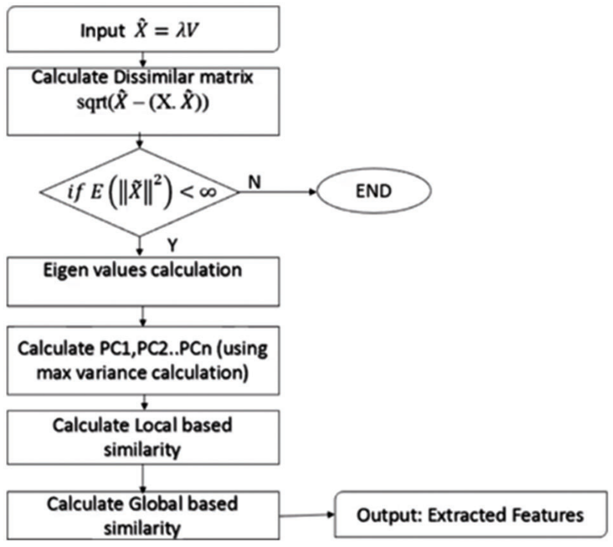

Input $\hat{X} = \lambda V$
Calculate Dissimilar matrix sqrt($\hat{X}$ – (X. $\hat{X}$))
$if\ E(\|\hat{X}\|^2) < \infty$
N
END
Y
Eigen values calculation
Calculate PC1,PC2..PCn (using max variance calculation)
Calculate Local based similarity
Calculate Global based similarity
Output: Extracted Features

Figure 6: Flowchart for enhanced PCA

The input variable 'X' defines the number of features in the dataset. The original dataset is transformed as N × d matrix X into an N × m matrix Y. The calculate covariance or correlation matrix using the equation given below

$$\mathrm{C} = \frac{1}{N-1} X^T X \tag{7}$$

$$\mathrm{Ci, j} = 1 \frac{1}{N-1} \sum Xq.i.Xq.j \tag{8}$$

We calculate the Eigenvector and values from the covariance matrix as ~X = λV and then calculate dissimilar matrix and local-based similarity calculation. We find local feature minimum distance and global feature minimum distance. The output of the hidden layer is computed by summing input multiplied with weights shown in the below equation.

$$y_c = f(\sum_{i=1}^{n} w_i x_i) \tag{9}$$

The error is then used to adjust the weights of the input vector according to the delta learning rule. The outcome will be based on weight-based features. The value of 'm' feature variables and 'n' observations based on coefficient's obtained from maximum variance calculations.

4.1 Feature Selection and Extraction

The feature selection approach also called the variable selection approach or attribute selection approach can be used to choose features that are most relevant to the predictive modelling problems. Some irrelevant features and redundant features may appear in feature sets. Irrelevant features should be reduced because they will have a low correlation with the class. Redundant features should be screened out as they will be highly correlated with one or more of the remaining features. Since the feature selection approach can remove irrelevant and redundant features, it usually gives a good or better accuracy whilst requiring fewer data. We have taken one example of Logistic regression which is a linear classifier whose parameters are weights, usually in terms of the weight vector, and the regularization parameter to explain the importance of feature selection for creating models. After training logistic regression is estimated, and the value of each weight represents how important that weight is for classification. The logistic Regression model uses Akaike Information Criterion for feature selection. The feature selection algorithm reduced the number of features to the eight most relevant ones. The experiments finally achieve better accuracy as shown in Fig. 7 below. The static features extracted from permission, APP Component, Filtered Intent, API are the major source of extraction and minor features considered are network address, operation code, data flow, and system components. These features are extracted using a disassembly tool namely the IDA pro demo version. After executing payload in sandboxing environment extracts features like system call, phone and SMS event, File Operations, Hook and text analysis using tools like AF Logical and log cat tools forensics tools. The diagram given below shows the working model of feature selection and extraction with sensitive permission taken from Android App and Meta data gives data about extracted information.

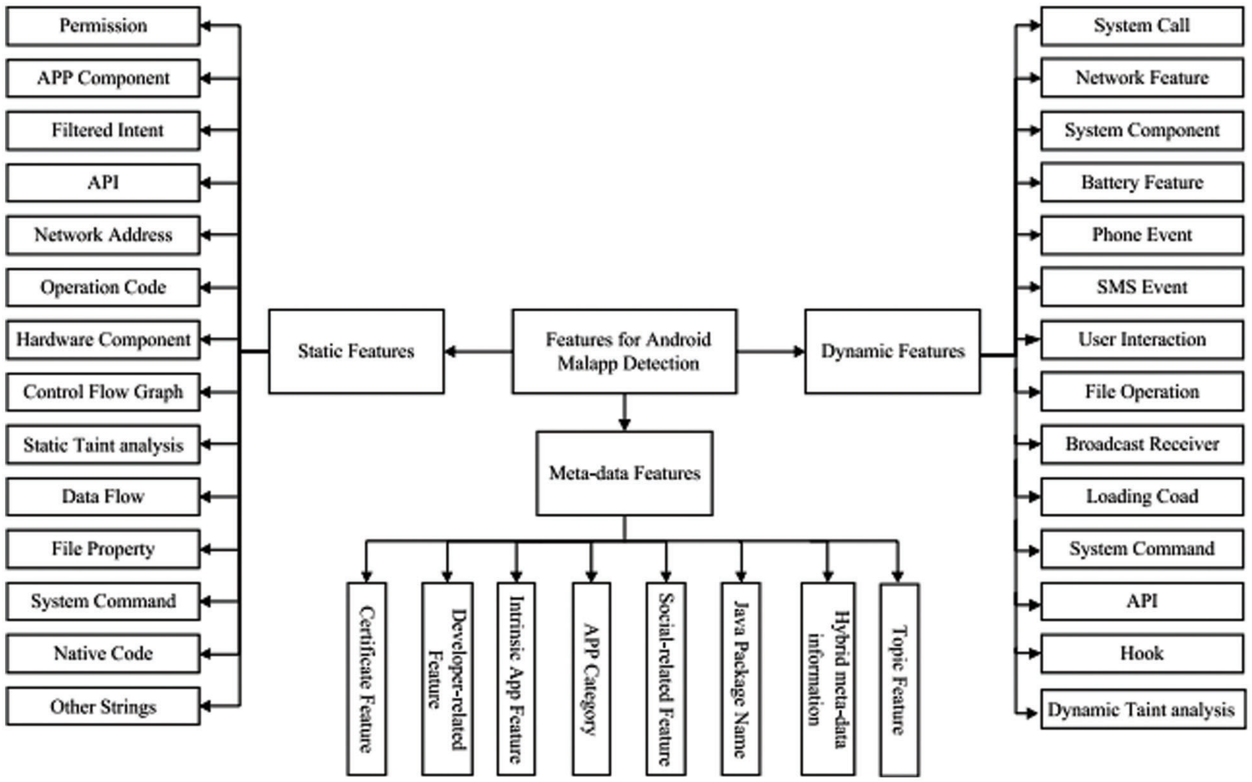


Figure 7: Feature selection and extraction

4.2 Evaluation Measures and Experimental Setup

The experiment is carried out with objective feature reduction with PCA and implements using logistic regression and K-NN machine learning classifiers and seeks for better accuracy time. The first step is data transformation using encoding techniques, find the correlation between variables. This computes the pairwise correlation of columns shown in Fig. 8. The correlation shows how the two variables are related to each other. The positive values show as one variable increases other variable increases as well. The negative values show as one variable increases other variable decreases. The Fig. 9 below shows the correlation between different features.

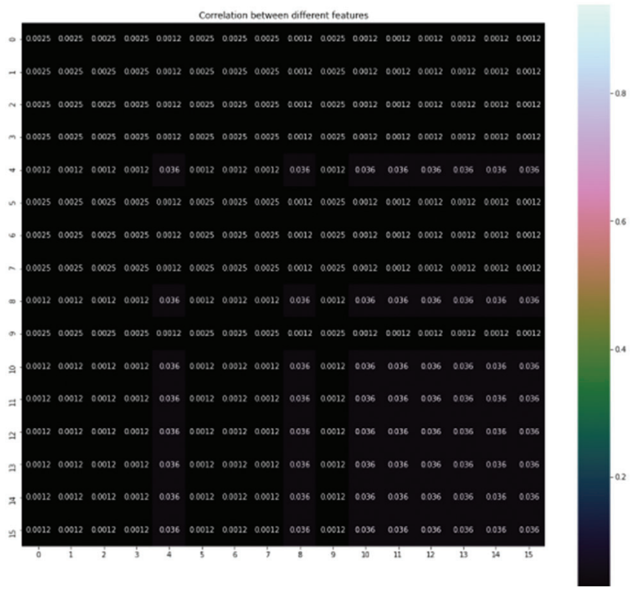


Figure 8: Correlation between features

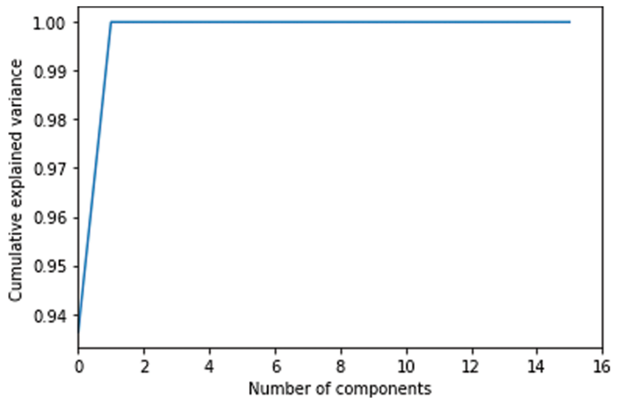
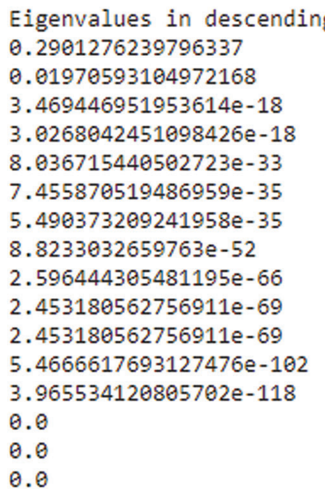


Figure 9: Covariance graph

The bigger the values, the more strongly two variables are correlated and vice-versa. The standardization refers to shifting the distribution of each attribute to have a mean of zero and a standard deviation of one (unit variance). It is useful to standardize attributes for a model. The standardization of datasets is a common requirement for many machine learning estimators implemented in scikit-learn; they might behave badly if the individual features do not more or less look like standard normally distributed data. Before computing Eigenvectors and values we need to calculate the covariance matrix. To decide which eigenvector (s) can be dropped without losing too much information for the construction of lower-dimensional subspace, we need to inspect the corresponding eigenvalues: The eigenvectors with the lowest eigenvalues bear the least information about the distribution of the data; those are the ones can be dropped as shown in Fig. 10. Thus Principal Component Analysis is used to remove the redundant features from the datasets without losing much information. The first component has the highest variance followed by the second, third, and so on. PCA works best on data set having 3 or higher dimensions. In our paper, we have taken 16 features reduced from correlated values. The second part of the proposed work shows the implementation of PCA in the k-NN approach. From the below graph, we can see that the error rate tends to get slower after k > 15, so we fix n_neigbors as '15' in Fig. 10. The output is shown in Figs. 11 and 12. The proposed approach is compared with Naïve Bayes, Bagging Decision Tree, Logistic Regression, Support Vector Machine classifiers and henceforth k-NN approach better accuracy rate of 0.80, and having improved result compared to existing approaches shown in Tab. 6.



```
Eigenvalues in descending order:
0.2901276239796337
0.01970593104972168
3.469446951953614e-18
3.0268042451098426e-18
8.036715440502723e-33
7.455870519486959e-35
5.490373209241958e-35
8.8233032659763e-52
2.596444305481195e-66
2.453180562756911e-69
2.453180562756911e-69
5.4666617693127476e-102
3.965534120805702e-118
0.0
0.0
0.0
```

Figure 10: Eigen value in descending order

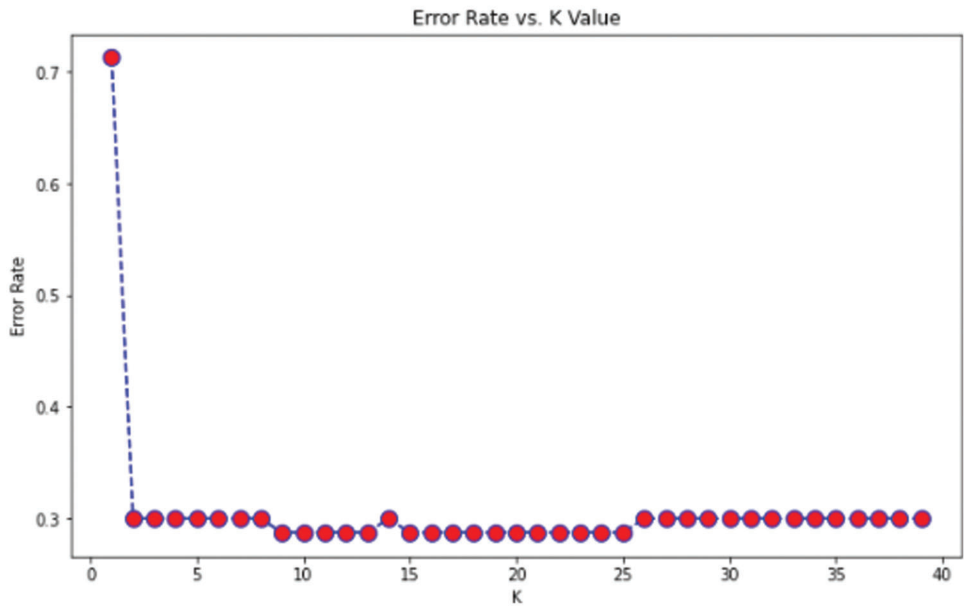


Figure 11: Error rate in k-NN approach

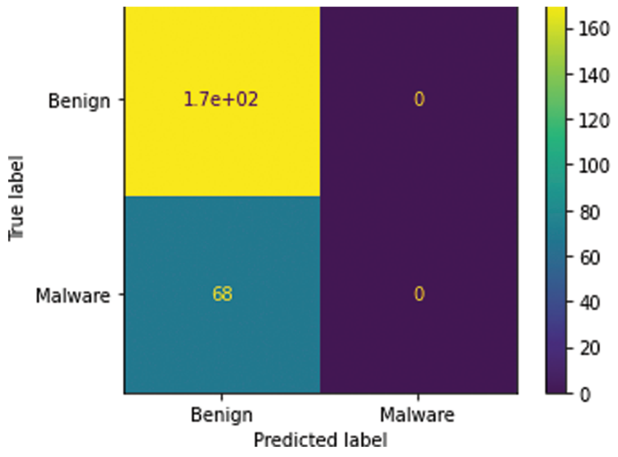


Figure 12: Visualization of the confusion matrix

Table 6: Binary malware classification performance using machine learning with PCA

Dataset	Accuracy (%)	F-Score	Recall	Precision
Naive Bayes	77%	0.71	0.77	0.76
Bagging Decision Tree	76%	0.72	0.76	0.73
Support vector machines	78%	0.72	0.78	0.76
Logistic Regression	75%	0.68	0.75	0.82
k-NN Approach	**80%**	**0.72**	**0.80**	**0.65**

5 Result Discussion

The comparative analysis of Naïve Bayes, Bagging Decision Tree, Logistic Regression, Support Vector Machine classifiers, and k-NN approaches are made without using principal component analysis and with principal component analysis and from the table given below we can see that model has achieved 80% accuracy with PCA for a k-NN approach using k = 15 better than the accuracy of 76% without PCA comparing with other classifier's. We have 100 epochs for validating and testing model. The outputs of static features are given as input for the machine learning classifier and using the classification approach decides whether the APK application is malicious or benign as discussed in the proposed framework. When we compare the performance with the existing approach, the proposed has better accuracy, F-Score, Recall and Precision as shown in Tabs. 5 and 6.

6 Conclusion and Future Work

The detection of android malware using a hybrid approach applying supervised learning for a dataset consisting of zero-day exploits and archives collected from recent payloads achieved an accuracy rate and visualize the model. We have explored and used effective forensics tools like Android Forensics (AF) logical tool for extracting features removes redundant data for creating the model. We filter out redundant features and optimize feature selection for better accuracy which can be considered as an optimization approach. We have also saved time and space by keeping minimum hardware and software requirements for building a model which can be used for a small-scale approach. The Multicollinearity problem is handled using principal component analysis by extracting and reducing features based on correlation. By using PCA (Principal Component Analysis) we ensure that dependency between variables is reduced resulting in better accuracy. The comparison is done with existing classifiers without using PCA and found that the error rate little bit higher. The potential limitation with proposed work might be some error rates achieved while implementing k-NN approach. In future work, we can move forward to implement unsupervised learning like Multilayer Perception apply clustering accuracy and further go for CNN (Convolution Neural Network) by applying XG- Boost Model increases the accuracy and then rank android malware payload and choose based on the level of impact. Henceforth by this approach, we can better the accuracy rate by using this model. As the functions of each application are increasingly powerful it has become mandatory for us to protect the user from vulnerable threats as we know that most of the Applications on the Android platform are not encrypted. As the next generation moving towards smart cities where most users will be using the android application for various purposes like banking, finance, fitness and health, social applications the possibility of threats might increase in the case of unencrypted applications as well as weaker applications. This might be one of the major challenges while establishing IoT platforms where most devices are connected with the internet resulting in breaking of integrity and confidentiality. Our proposed work leading to the threat model can suggest or helps in decision-making for users while installing an application from not trusted resources.

Funding Statement: The authors received no specific funding for this study.

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

References

[1] R. Surendran, T. Thomas and S. Emmanuel, "A TAN based hybrid model for android malware detection," *Journal of Information Security and Applications*, vol. 54, no. 5, pp. 102483–102495, 2020.

[2] V. J. Raymond and R. J. R. Raj, "Reversing and auditing of android malicious applications using sandboxing environment," *International Journal of Electronic Security and Digital Forensics*, vol. 12, no. 4, pp. 386–396, 2020.

[3] Y. Zheng and B. Liu, "Fuzzy vehicle routing model with credibility measure and its hybrid intelligent algorithm," *Applied Mathematics and Computation*, vol. 176, no. 2, pp. 673–683, 2006.

[4] H. Shahriar, M. Islam and V. Clincy, "Android malware detection using permission analysis," *IEEE Access*, vol. 12, no. 6, pp. 1–6, 2017.

[5] S. Y. Yerima, S. Sezer and I. Muttik, "High accuracy android malware detection using ensemble learning," *IET Information Security*, vol. 9, no. 6, pp. 313–320, 2017.

[6] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro and G. Ross, "Mamadroid: Detecting Android malware by building Markov chains of behavioral models," *arXiv preprint arXiv*, vol. 16, no. 12, pp. 4433–4445, 2016.

[7] 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*, German, vol. 23, no. 6, pp. 15–26, 2011.

[8] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song *et al.,* "A novel 3-level hybrid malware detection model for android operating system," *IEEE Access*, vol. 6, no. 16, pp. 4321–4339, 2018.

[9] V. Bulakh and M. Gupta, "Countering phishing from brands' vantage point," in *Proc. ACM Int. Workshop on Security and Privacy Analytics*, Italy, vol. 12, no. 2, pp. 17–24, 2016.

[10] G. Canfora, E. Medvet, F. Mercaldo and C. Vissagio, "Vissagio detection of malicious web pages using system calls sequences," in *Proc. Int. Conf. on Availability, Reliability, and Security*, United States of America, vol. 4, no. 7, pp. 226–238, 2014.

[11] S. K. Dash, G. S. Tangil, S. J. Khan, K. Tam, M. Ahmadi *et al.,* "Classifying android malware based on runtime behavior," in *Proc. IEEE Security and Privacy Workshops*, China17, 5, pp. 252–261, 2016.

[12] J. NewSom, "Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation of Exploits on Commodity Software, Network and Distributed System Security," in *Proc. Symp. Conf.*, China, vol. 7, pp. 231–245, 2005.

[13] K. Tam, S. J. Khan and A. Fattori, “Copperdroid: Automatic reconstruction of android malware behaviors,” in *Proc. NDSS Symp. 2015*, California, USA, vol. 5, no. 9, pp. 371–389, 2015.

[14] D. Wagner and R. Dean, “Intrusion detection via static analysis,” in *Proc. IEEE Symp. on Security and Privacy*, vol. 4, no. 9, pp. 156–168, 2001.

[15] S. Jamalpur, S. Navya, Y. S. Raja, P. Tagore and G. R. K. Rao, “Dynamic malware analysis using cuckoo sandbox,” *Proc. Second Int. Conf. on Inventive Communication and Computational Technologies*, USA, vol. 12, no. 4, pp. 1056–1060, 2018.

[16] A. Ali-Gombe, I. Ahmed and I. Richard, “Aspectdroid: Android app analysis system,” *Proc. Sixth ACM Conf. on Data and Application Security and Privacy*, German, vol. 7, no. 11, pp. 145–147, 2016.

[17] E. Mariconti, L. Onwuzurike, P. Andriotis and E. De Cristofaro, “Detecting android malware by building Markov chains of behavioral models,” *arXiv preprint arXiv*, vol. 16, no. 8, pp. 1612–1625, 2016.

[18] S. Arshad, M. A. Shah, A. Wahid, A. Mehmood, H. Song *et al.,* “A novel 3-level hybrid malware detection model for android operating system,” *IEEE Access*, vol. 6, no. 9, pp. 4321–4339, 2018.

[19] M. Ajdani and H. Ghaffary, “Design network intrusion detection system using support vector machine,” *International Journal of Communication Systems*, vol. 34, no. 3, pp. 4689–4694, 2021.

[20] S. Chen, M. Xue, Z. Tang, L. Xu and H. Zhu, “A streaminglized machine learning-based system for detecting android malware,” *Proc. 11th ACM on Asia Conf. on Computer and Communications Security*, China, vol. 37, no. 8, pp. 377–388, 2016.

[21] X. Liao, Y. Xue and L. Carin, “Logistic regression with an auxiliary data source,” *Proc. 22nd Int. Conf. on Machine Learning*, vol. 5, no. 18, pp. 505–512, 2005.

[22] S. Wold, S. Esbensen and P. Geladi, “Principal component analysis,” *Chemometrics and Intelligent Laboratory Systems*, vol. 2, no. 3, pp. 37–52, 1987.

[23] B. Abu-Shaqra and R. Luppicini, “Technoethical inquiry into ethical hacking at a Canadian University,” *International Journal of Technoethics (IJT)*, vol. 7, no. 1, pp. 62–76, 2016.

[24] A. M. Abdelrahman, J. J. Rodrigues, M. Mahmoud, K. Saleem, K. A. *et al.,* “Software-defined networking security for private data center networks and clouds: Vulnerabilities, attacks, countermeasures, and solutions,” *International Journal of Communication Systems*, vol. 34, no. 4, pp. 7746–7758, 2021.

[25] R. Wazirali and R. Ahmed, “Hybrid feature extractions and CNN for enhanced periocular identification during Covid-19,” *Computer Systems Science and Engineering*, vol. 41, no. 1, pp. 305–306, 2022.

[26] M. Baz, S. Khatri, A. Baz, H. Alhakami, A. Agrawal *et al.,* “Blockchain and artificial intelligence applications to defeat COVID-19 pandemic,” *Computer Systems Science and Engineering*, vol. 40, no. 2, pp. 691–702, 2022.

[27] W. Sun, G. Z. Dai, X. R. Zhang, X. Z. He and X. Chen, “TBE-Net: A three-branch embedding network with part-aware ability and feature complementary learning for vehicle re-identification,” *IEEE Transactions on Intelligent Transportation Systems*, vol. First Online, pp. 1–13, 2021.

[28] W. Sun, L. Dai, X. R. Zhang, P. S. Chang and X. Z. He, “RSOD: Real-time small object detection algorithm in UAV-based traffic monitoring,” *Applied Intelligence*, vol. 92, no. 6, pp. 1–16, 2021.