Defence Science Journal, Vol. 72, No. 3, May 2022, pp. 371-381, DOI : 10.14429/dsj.72.3.17522

A Novel Traffic Based Framework for Smartphone Security Analysis

Sumit Kumar,[#,*] S. Indu[$] and Gurjit Singh Walia[#]

[#]*DRDO - Scientific Analysis Group (SAG), Delhi - 110 054, India*
[$]*Electronics and Communication, Delhi Technological University, Delhi - 110 042, India*
[*]*E-mail: sumitkr@hotmail.com*

ABSTRACT

Android Operating system (OS) has grown into the most predominant smartphone platform due to its flexibility and open source characteristics. Because of its openness, it has become prone to numerous attackers and malware designers who are constantly trying to elicit confidential information by articulating a plethora of attacks through these designed malwares. Detection of these malwares to protect the smartphone is the core function of the smartphone security analysis. This paper proposes a novel traffic-based framework that exploits the network traffic features to detect these malwares. Here, a unified feature (UF) is created by graph-based cross-diffusion of generated order and sparse matrices corresponding to the network traffic features. Generated unified feature is then given to three classifiers to get corresponding classifier scores. The robustness of the suggested framework when evaluated on the standard datasets outperforms contemporary techniques to achieve an average accuracy of 98.74 per cent.

Keywords: Smartphone; Malware; Network traffic; Android; Fusion

1. INTRODUCTION

Smartphones are replacing conventional mobiles as well as computational devices due to their portability and ease of handling almost everything ranging from storing private data to making banking transactions. Smartphones having Android OS are extensively familiar and have wide usage due to their open architecture and the assortment of apps it affords. As per the recent information by IDC[1] (International Data Corporation), the market share of android smartphones is 83.8 per cent till March 2021 and it will grow to 85 per cent by March'2025. Due to its widespread usage, we are deluged with a variety of smartphone apps that makes our life simple and easier. Flooding of these apps tempts attackers to design a variety of malapp (malicious applications) which are directed toward smartphones to steal vital private information, acquire root privileges, build botnets to attack networks, or extract critical data from smartphones. These malwares after stealing the critical data sent it to the remote servers controlled by hackers. This paper aims at identifying the existence of those malapps which are distantly controlled by hackers via remote servers to collect the critical data. The network traffic-based framework extracts required features that are fused resulting in a unified feature. The unified feature is further given to the optimal classifier for the detection of malwares. This n/w traffic analysis-based framework aids in creating robust solutions for identifying android malwares.

These malapps are the most deterrent to the security of these devices. Smartphone security refers to the measures taken to thwart these malwares to protect sensitive information stored in or communicated by smartphones. The security analysis of smartphones mainly deals with findings of the vulnerabilities and intimidations while using smartphone applications.

Mobile devices including Smartphones generated about 54.8 per cent of worldwide web traffic[2] and analyzing this traffic leads to incredible results in detecting malapps. Analyzing traffic[3,4] is accomplished by studying the patterns in the network traffic for its identification and segregation for further investigation. Numerous traffic features are extracted from the network traffic patterns. Mainly Hypertext Transfer Protocol (HTTP) and Transmission Control Protocol (TCP) are two types of traffic that are prevalent in the smartphone ecosystem. Features extracted from HTTP and TCP are exploited in detecting the malapps. The HTTP header features could not detect the malapps in the encrypted traffic and TCP-based detection models are impervious to encrypted traffic. Therefore, TCP flow-based detection methods are mainly exploited in detecting malapps.

Two widely used malware detection methods employed by researchers pivots around static[5] and dynamic analysis[6-8]. The amalgamation of these widely used detection methods is also exploited by some researcher's resulting in hybrid analysis[9]. Static investigation-based detection techniques failed to detect apps having code obfuscation, and conventional dynamic investigation-based detection needs are quite cumbersome. N/w traffic based dynamic detection excerpts detection attributes from n/w traffic and uses Machine Learning (ML) techniques to categorize mobile apps. Our detection prototype is based on TCP-based features.

In short, the following are the main contributions to the manuscript:

Received : 24 September 2021, Revised : 30 March 2022
Accepted : 11 May 2022, Online published : 01 July 2022

- Proposed a traffic feature-based fusion that comprises of optimal combination of multiple traffic features by cross-diffusion of order and sparse graphs to produce a unified feature.
- The unified feature vector thus generated is given to the three parallel ML classifiers and classifiers scores obtained are fused to enhance the accuracy attained by separate classifiers.
- Presented the performance comparison with existing state-of-the-art methods using standard data sets available.

In Section 2, related work is presented. In Sec 3, we cover the proposed identification framework for malware identification and the assessment of our technique is presented in Sec 4. In Sec 5, we conclude the paper.

2. RELATED WORK

Arora[10], *et al.* examine the TCP-based features based on traffic to shape the classifier for Android malware with more than 90 per cent of detection accuracy. Arora[11] *et al.* came up with a hybrid model named NTPDroid (**N**etwork **T**raffic and **P**ermissions based An**droid** malapp detection framework), that uses permissions and traffic features from the apps and exploits a Frequent Pattern Growth algo to generate frequent patterns of permissions and traffic features to achieve detection accuracy of 94 per cent. Wang[12] *et al.* proposed an efficient malware detection technique by using text semantics of n/w traffic by studying each HTTP flow. These HTTP packets were further processed by NLP (Natural Language Processing) to take out text-level features achieving an accuracy of 99.15 per cent but the method achieves 54.81 per cent for unknown apps in the wild.

Liu[13] *et al.* proposed a malware detection technique built on TCP n/w traffic, where network traffic generated by apps gets a greater number of a TCP flow to extract packet sizes as features. Results achieve 97 per cent of detection accuracy. Ding Li[14] *et al.* introduced a framework named DroidClassifier for the identification of HTTP header fields of n/w traffic created by malapps by using a supervised method to train the malware dataset. Moreover, Clustering is also used to increase the classification efficiency. The results achieve 90 per cent of detection accuracy. Li[15] *et al.* proposed a multilevel detection system named MulAV, in which it obtains info from n/w traffic, App's source code, and geospatial info where n/w traffic is collected by TCPdump Tool. The info is further fed to the ML method to train the model which identifies malapps. The result achieves a detection rate of 97.8 per cent. Wang[16] *et al.* discussed a technique to parse the HTTP packets of n/w traffic where features analysed are packet avg. length, number of upload and download packets, distribution of packet size, etc. Features were further extracted to obtain the pure malicious traffic dataset and this is used to detect malwares. Su[17] *et al.* presented an Android detection method that uses TCP-based behavioral characteristics to detect malapps. Here capturing of n/w traffic is done using NTM [network traffic monitor] tool and training are done via n/w traffic classifier. Results achieve 99.2 per cent and 94.2 per cent of detection accuracy by using Random forest and J48 classifier.

Zulkifli[18], *et al.* proposed a detection process based on n/w traffic which registers the app behavior and considered 7 TCP-based n/w traffic features from Contagio dumpset and Drebin dataset in which Drebin dataset achieved 98.4 per cent of detection accuracy on J48 decision tree algo. Malik[19], *et al.* proposed a pattern-based detection method CREDROID which identifies malapps based on the DNS (Domain Name Service) queries, data it transfers to the remote server from n/w traffic logs, and also the protocol used for communication for identifying the credibility of the app. Moreover, the Android app can be checked without rooting the android phone. Wang[20] et.al. proposed a malapp detection framework exploiting the URLs (Uniform Resource Locator) visited by them. Here the malapp detection model is based on a multi-view neural network with a detection accuracy of 98.35. Multiple views maintain copious semantic info from inputs for segregating the apps. Wang[21], *et al.* suggested a framework for android malapp identification leveraging both the TCP and HTTPS features. Here, the app detection was done on the server-side without affecting the user experience. C4.5 ML algo is used to train the model with 8312 benign and 5560 malign apps for identifying unknown apps with an accuracy of 97.89 per cent. Sanz[22], *et al.* offered a lightweight malapp detection framework using TCP-based network features with an accuracy of 90 per cent and a false-positive rate of less than 3 per cent. Here, the total number of 359 malapp and benign apps are used along with two Random forests and AdaBoost ML algorithms.

Upadhayay[23] et.al. suggested a hybrid-based malware detection model using network traffic and permissions with a higher frequency of occurrence to achieve the detection accuracy of 95.96 per cent. Alshehri[24], *et al.* proposed an innovative method to detect the repackaged apps by investigating the network traffic behavior of smartphones. Here authors exploited the request traffic generated by the apps. A total number of 8645 applications were used for experimentation. Here the accuracy of request flows attained is 95.1 per cent an improvement of 18.3 per cent of accuracy when compared with contemporary methods. Sihag[25], *et al.* proposed network packet-based investigations of captured traffic of the smartphone. Here, the authors represent the captured network packet interactions as images. These images were given to CNN (Convolution Neural Network) to achieve a detection accuracy of 99.12 per cent. Norouzian[26], *et al.* offered a hybrid detection method based on network traffic flow and static features. Here network flow features are combined with static graph vectors to detect malapps with 97 per cent accuracy.

3. IDENTIFICATION FRAMEWORK PROPOSED

The proposed traffic analysis framework for smartphone security analysis is elucidated in Fig. 1. The framework consists of four blocks viz. traffic feature fusion, classifier score-fusion, decision criteria, and reference apps update to accomplish efficient malapp detection. Extracted traffic features are converted into three traffic feature vectors. All the three traffic feature vectors are used for constructing similarity graphs by the means of cosine similarity. Similarity graphs are

again converted to normalised graphs by using the anchored normalisation technique. Normalised graphs are again used to form the sparse and order graphs. These obtained sparse and order graphs are further cross diffused to generate three fused traffic feature vectors. The three fused feature vectors are further concatenated to form the highly distinct unified feature vector. This distinct unified feature is discriminatory and given to three classifiers. The scores obtained from these classifiers are again optimally combined to classify a given test app.

Because of the numerous attributes encapsulated in the android based smartphone apps, individual ML algo shows its incompetence to categorize these apps accurately. To achieve overall detection accuracy, the framework uses three classifiers viz. Random Forest (RF), k-Nearest Neighbor (KNN), and Support Vector Machine (SVM) for app categorisation. RF performs superbly when the dataset is large and it is not susceptible to outliers. SVM performs better in the limited dataset and it is optimal for binary classification. If there is no training period, then KNN performs best. The framework leverages modified PCR-5 rules for score fusion. Finally, in decision block score w_{PCR5} is matched with the threshold w_{th} and a particular test app is categorised as benign if $w_{PCR5} \geq w_{th}$ or otherwise malign.

3.1 Traffic Feature Fusion

It is apparent from Fig. 1 that traffic feature fusion comprises feature extraction followed by similarity, normalised, sparse, and order graph generation. The generated sparse and order graphs are further cross diffused giving unified feature vector U as output.

3.1.1 Traffic Feature Extraction

The traffic gathering platform is used to collect the malign and benign traffic data produced by malign and benign apps, respectively. A firewall is installed on the platform to guarantee its security. Figure 2 shows our methodology for traffic gathering. The traffic gathering platform comprises four constituents, i.e., the control server, traffic collection module, app repository having downloaded malign and benign apps, and TCP traffic storage module containing only filtered TCP flows. These four components converse with the aid of a LAN switch. The control server is controlling the traffic gathering job in the platform by assigning the job to the different modules. The apps from the apps repository are directed to the traffic collection module, where the android virtual machine (AVM) is used to run the apps and collect the corresponding traffic. The collected traffic is further directed to the TCP traffic storage module, where only TCP flows are stored and the rest of the traffic is filtered out. Here android emulators are used to install and running apps on AVM. AVM comprises of packed android s/w stack and it runs just like a physical smartphone. Apps are run on an emulator. An emulator is restarted to fuel the malign apps to generate malicious behavior in the network traffic. A script in python was written to extract the features from the TCP flows. The extracted TCP features used are tabulated in Table 1.

Fifteen traffic features were extracted for a test app ***t*** along with N apps from the reference dictionary, $d \in \{D^+, D^-\}$, D^+ which comprises of benign apps and D^- comprises of the malign app. The traffic feature extraction procedure is shown in Fig. 2. Update in Fig.1 shows that the reference dictionary

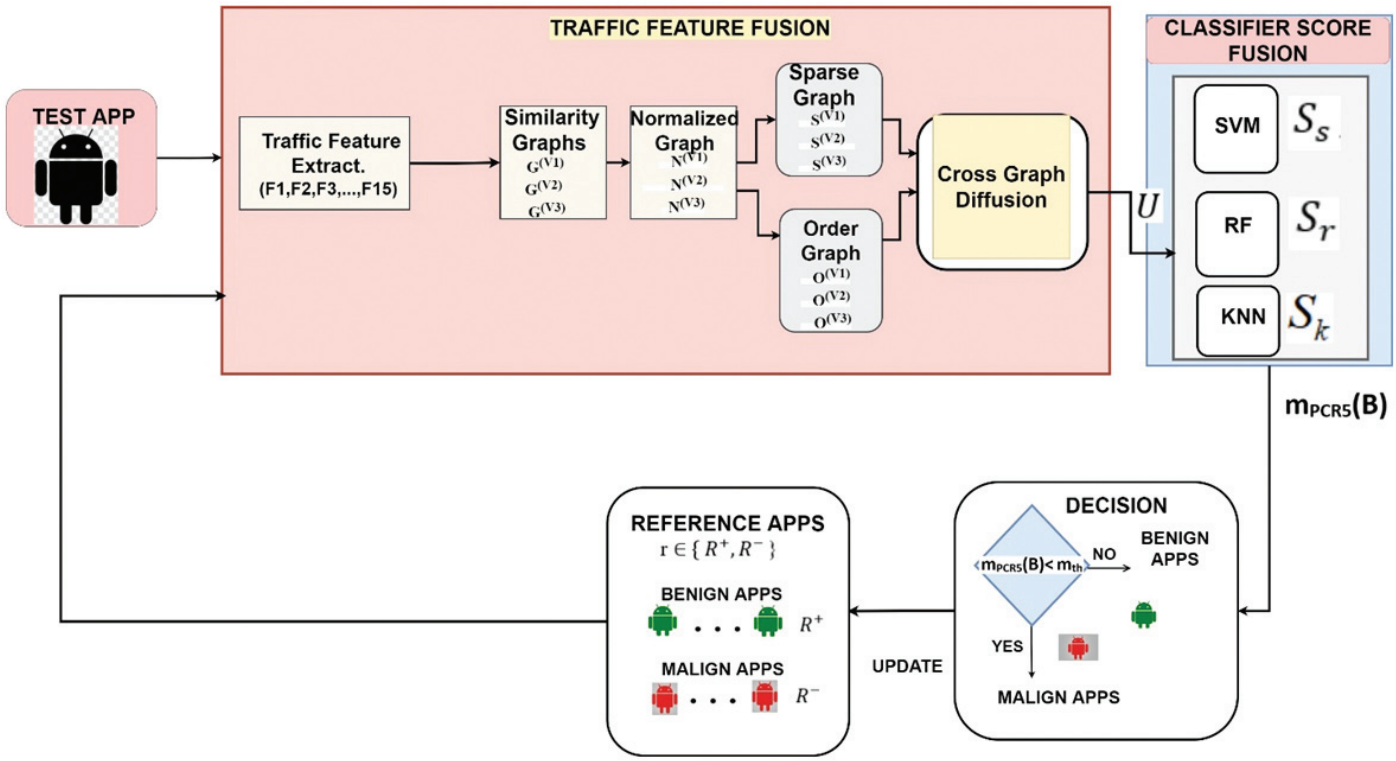

Figure 1. Proposed traffic-based framework for smartphone security analysis.

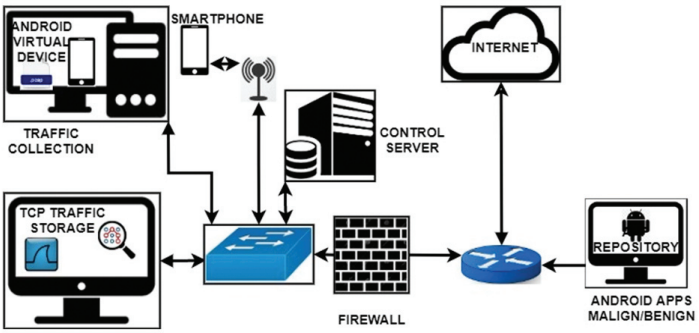

Figure 2. Traffic gathering platform.

Table 1. Extracted TCP based traffic features

Feature symbol	Feature description
$F1$	Avg. no. of bytes sent
$F2$	Avg. no. of bytes received
$F3$	Total no. of headers bytes sent
$F4$	Avg. no. of bytes per second
$F5$	The ratio of the no. of incoming to outgoing bytes
$F6$	Avg. no. of the packet sent per second
$F7$	Avg. no. of the packet received per second
$F8$	The ratio of the no. of incoming to no. of outgoing packets
$F9$	Std. deviation of the packet- size sent
$F10$	The standard deviation of the packet- size received
$F11$	Avg. no. of the packet sent per-flow
$F12$	Avg. no. of the packet received per-flow
$F13$	Avg. no. of bytes sent per-flow
$F14$	Avg. no. of bytes received per-flow
$F15$	Std. deviation of the length of the flow

apps are constantly updated with the most recent malign and benign apps to improve the detection ability of the framework. Here, only TCP packets and TCP-related traffic features are considered. Traffic features exploited in our framework are depicted in Table 1. From the above 15 traffic features, we form three vectors such that each vector complements the other in detecting the malicious app. The vectors formed are as follows:

$$V^1 = \{F1, F2, F3, F4, F5\} \quad (1)$$

$$V^2 = \{F6, F7, F8, F9, F10\} \quad (2)$$

$$V^3 = \{F11, F12, F13, F14, F15\} \quad (3)$$

The proposed solution is realised by the formation of the three complementary traffic feature vectors, namely V^1 (byte-based features), V^2 (packet-based features), and V^3 (flow-based features). We have built three feature vectors as stated in Eqn. (1) - Eqn. (3) for each test app and reference app. In feature-fusion, traffic-features vectors for reference and test apps are utilised for graph formation. The test app's traffic-feature vectors represent one node and the reference app's traffic-feature vector represents other nodes. Consequently, non-linear graphs are formed for all test app ***t*** corresponding to three traffic feature vectors.

3.1.2 Graph Generation

This subsection mainly comprises the generation of similarity, normalised, sparse, and order graphs. For traffic feature vectors, V_t^1, V_t^2 and V_t^3 of test app ***t*** corresponding to three traffic-based features, we construct graphs $G^\phi = (Ver^\phi, E^\phi, w^\phi)$, where $\phi \in \{1,2,3\}$, corresponding to three traffic-based features and w^ϕ are edge weights acting as similarity betwixt traffic-feature vectors of apps ***t*** and d where $d \in \{D^+, D^-\}$, Ver^ϕ correlates to the vertices of the created similarity graphs, E^ϕ correlates to the edges of the similarity graphs that characterize the association between test apps and the reference apps. In the suggested framework, similarity matrices $G^\phi \in \mathbb{R}^{n \times n}$ are constructed by calculating the cosine similarity between the three traffic-feature vectors of the test app t and reference apps d. For feature set values $\left(V_t^\phi, V_d^\phi\right)$, where $\phi \in \{1,2,3\}$ corresponds to three traffic-feature vectors, the similarity edge weights are symbolised by the vector $w^\phi(t,r)$ and are derived by the cosine similarity between the pair $\left(V_t^\phi, V_d^\phi\right)$ from the following Eqn. (4).

$$w^\phi(t,d) = \frac{V_t^\phi * V_d^\phi}{\left\|V_t^\phi\right\| \left\|V_d^\phi\right\|} \quad (4)$$

Similarity graphs created using Eq.(4) for three traffic feature vectors are further normalised using an anchor $A^\phi = A_{avg}^\phi + A_{std}^\phi$, where A_{avg}^ϕ A_{std}^ϕ and are average and standard deviation of malicious score distribution for ϕ traffic feature vector. We consider only malicious scores for calculating the anchor. The normalised graphs N^ϕ, whose edge weights matrix $\bar{w}_j^\phi$, where $\forall j \in \{1,2,.....,n\}, \forall \phi \in \{1,2,3\}$ is constructed by Eqn. (5),

$$\bar{w}_j^\phi = \begin{cases} \dfrac{w_j^\phi - \min(w^\phi)}{2(A^\phi - \min(w^\phi))}, w_j^\phi \le A^\phi \\ 0.5 + \left(\dfrac{w_j^\phi - A^\phi}{\max(w_j^\phi) - A^\phi} \right), w_j^\phi > A^\phi \end{cases} \quad (5)$$

The obtained normalised graphs were transformed to obtain the sparse S^ϕ and order O^ϕ graphs. Sparse graphs guarantee robustness to noise while boosting the strong info and suppressing the weak info corresponding to each traffic feature vector. A sparse graph $S^\phi = \{Ver^\phi, E^\phi, \eta^\phi\}$ is built using KNN by Eqn. (6).

$$\eta^\phi = \begin{cases} \bar{w}_j^\phi, \left(\bar{w}_j^\phi \in KNN_{N^\phi} \mid \nu\right) \\ 0, Otherwise \end{cases} \quad (6)$$

ν is the parameter controlling the sparseness of the graph. Edges corresponding to reference apps that are similar to the test apps are retained and the rest of the edges are removed to ensure robustness to noise. To discriminate the significant reference apps from the insignificant ones, each of the reference apps is assigned a weight according to its order. Therefore, order(s) is

assigned to each reference app based on its similarity with the test app, which is calculated by the edge weight between the test app and the reference app in the normalised graph. The order graph $O^{\phi} = \{Ver^{\phi}, E^{\phi}, \mu^{\phi}\}$ having the weight matrix is constructed by Eqn.(7) :

$$\mu^{\phi} = order(\overline{w_j}^{\phi}), \forall j \in \{1,2,...n\} \tag{7}$$

Where function *order*, allocates order(s) to each reference app.

Details of feature unification are presented in the subsequent subsection.

3.1.3 Cross Diffusion

Three traffic feature vectors created are fused in a way to extract complementary info in them by optimal non-linear cross-diffusion of generated sparse and order graphs. Features fusion via cross-diffusion technique[27] was offered and its results confirm that the feature fusion by non-linear graphs techniques are significantly more accurate than linear graph-centered approaches. Graph-built unification retains robust features of apps and rejects all the frail features that add to misclassification.

Order graphs averts biasness and sparse graphs guarantees elimination of any outlier behavior. Cross diffusion [Eq. (9)] basically comprised of addition of sparse graphs of two other feature vectors to form α^{ϕ^t} [Eq.(8)], where α^{ϕ^t} is the t^{th} element of the set ϕ and subsequently their multiplication with order graph of the feature vector with α^{ϕ^t}.

$$\alpha^{\phi^t} = \sum_{Y} \eta^{Y} \quad \text{where, } Y \in \{\phi\} - \{\phi^t\} \tag{8}$$

$$\beta^{\phi^t} = \alpha^{\phi^t} \odot \mu^{\phi^t} \quad \forall \phi \in \{1,2,3\} \tag{9}$$

Here, we add the sparse graph of two other feature vectors and then multiply it with the order graph of the feature vector turn by turn to generate three fused vectors. Eqn. (9) can be further represented [Eqn. (10,11,12)] in the form of fused vectors ($F^{V^1}, F^{V^2}, F^{V^3}$), sparse vectors ($S^{V^1}, S^{V^2}, S^{V^3}$) and order vectors ($O^{V^1}, O^{V^2}, O^{V^3}$) as follows:

$$F^{V^1} = (S^{V^2} + S^{V^3}) \odot O^{V^1} \tag{10}$$

$$F^{V^2} = (S^{V^3} + S^{V^1}) \odot O^{V^2} \tag{11}$$

$$F^{V^3} = (S^{V^1} + S^{V^2}) \odot O^{V^3} \tag{12}$$

The above-fused feature vectors F^{V^1}, F^{V^2} and F^{V^3} are concatenated to form the unified feature vector, U by Eqn. (13).

$$U = F^{V^1} + F^{V^2} + F^{V^3} \tag{13}$$

This unified feature vector is given to three parallel classifier(s) i.e. SVM (Support Vector Machine), RF (Random Forest) & KNN (K Nearest Neighbors). The classifiers scores of these parallel classifiers are further fused to classify apps. Details of Optimal classifier fusion follow in the next sub-section.

3.2 Classifier Score-Fusion

Created U is fed to three classifiers connected in parallel. Obtained classification scores [S_s (SVM), S_r (RF), S_k (KNN)] are again fused by the classifier score fusion technique. There are various score fusion techniques[28,29] reported in the literature. Here we have chosen the PCR-5[30] to solve the highly contradictory scores of the three classifiers. PCR-5 is used to solve ambiguous problems in multi-sensor score fusion. Android app detection is a certainly ambiguous problem as we are uncertain whether the app is malicious or not. In the suggested model, three classifiers are selected and the fusion of the output of these classifiers can be modeled as a multi-sensor score fusion problem as their outputs are independent of each other. Therefore, Android app detection satisfies all the conditions of the PCR-5 theory. In our framework, the frame of discernment Θ has two elements B M and corresponding benign and malign. Classifier scores [S_s, S_r, S_k] are converted individual Basic Belief assignments or belief masses by the Eqn.(14).

$$\begin{aligned} m_i(\mathrm{B}) &= C_i \times S_i(\mathrm{B}) \\ m_i(\mathrm{M}) &= 1 - C_i \times S_i(\mathrm{B}) \end{aligned} \tag{14}$$

Where $i \in (s,r,k)$ & C_i denotes the confidence measure of the single classifier.

These belief masses are combined by PCR-5 rules. The conjunctive consensus among the classifiers is assessed by Eqn. (15, 16)

$$m_{srk}(\mathrm{B}) = m_s(\mathrm{B}) * m_r(\mathrm{B}) * m_k(\mathrm{B}) \tag{15}$$

$$m_{srk}(\mathrm{M}) = m_s(\mathrm{M}) * m_r(\mathrm{M}) * m_k(\mathrm{M}) \tag{16}$$

Overall conflict among the classifiers is estimated by Eqn. (17). It comprises six partial conflicts-masses.

$$\begin{aligned} m_{srk}(\mathrm{B} \cap \mathrm{M}) = {} & m_s(\mathrm{B}) * m_r(\mathrm{M}) * m_k(\mathrm{M}) + m_r(\mathrm{B}) * m_s(\mathrm{M}) * m_k(\mathrm{M}) + \\ & m_k(\mathrm{B}) * m_s(\mathrm{M}) * m_r(\mathrm{M}) + m_s(\mathrm{M}) * m_r(\mathrm{B}) * m_k(\mathrm{B}) + \\ & m_r(\mathrm{M}) * m_s(\mathrm{B}) * m_k(\mathrm{B}) + m_k(\mathrm{M}) * m_s(\mathrm{B}) * m_r(\mathrm{B}) \end{aligned} \tag{17}$$

Six partial conflict masses are further redistributed using PCR-5 rules in ratio to masses assisting these partial conflicts. The values of p_i and q_i are a contribution to Benign and Malign masses following reallocation of partial conflicts, where i = 1,…,6 and are determined by Eqn. (18-23).

$$\frac{p_1}{m_s(\mathrm{B})} = \frac{q_1}{m_r(\mathrm{M}) * m_k(\mathrm{M})} = \frac{m_s(\mathrm{B}) * m_r(\mathrm{M}) * m_k(\mathrm{M})}{m_s(\mathrm{B}) + m_r(\mathrm{M}) * m_k(\mathrm{M})} \tag{18}$$

$$\frac{p_2}{m_r(\mathrm{B})} = \frac{q_2}{m_s(\mathrm{M}) * m_k(\mathrm{M})} = \frac{m_r(\mathrm{B}) * m_s(\mathrm{M}) * m_k(\mathrm{M})}{m_r(\mathrm{B}) + m_s(\mathrm{M}) * m_k(\mathrm{M})} \tag{19}$$

$$\frac{p_3}{m_k(\mathrm{B})} = \frac{q_3}{m_s(\mathrm{M}) * m_r(\mathrm{M})} = \frac{m_k(\mathrm{B}) * m_s(\mathrm{M}) * m_r(\mathrm{M})}{m_k(\mathrm{B}) + m_s(\mathrm{M}) * m_r(\mathrm{M})} \tag{20}$$

$$\frac{p_4}{m_r(\mathrm{B}) * m_k(\mathrm{B})} = \frac{q_4}{m_s(\mathrm{M})} = \frac{m_s(\mathrm{M}) * m_r(\mathrm{B}) * m_k(\mathrm{B})}{m_s(\mathrm{M}) + m_r(\mathrm{B}) * m_k(\mathrm{B})} \tag{21}$$

$$\frac{p_5}{m_s(\mathrm{B}) * m_k(\mathrm{B})} = \frac{q_5}{m_r(\mathrm{M})} = \frac{m_r(\mathrm{M}) * m_s(\mathrm{B}) * m_k(\mathrm{B})}{m_r(\mathrm{M}) + m_s(\mathrm{B}) * m_k(\mathrm{B})} \tag{22}$$

$$\frac{p_6}{m_s(\mathrm{B}) * m_r(\mathrm{B})} = \frac{q_6}{m_k(\mathrm{M})} = \frac{m_k(\mathrm{M}) * m_s(\mathrm{B}) * m_r(\mathrm{B})}{m_k(\mathrm{M}) + m_s(\mathrm{B}) * m_r(\mathrm{B})} \tag{23}$$

Approximated contributions p_i and q_i where i = 1,…,6 are added on to their respective conjunctive consensus. The final

belief of the app being benign m_{PCR5} (B) and that of an app being malign m_{PCR5} (M) are determined by Eq. (24) and Eq. (25) respectively.

$$m_{PCR5}(\mathrm{B}) = m_{srk}(\mathrm{B}) + \sum_{i=1}^{6} p_i \tag{24}$$

$$m_{PCR5}(\mathrm{M}) = m_{srk}(\mathrm{M}) + \sum_{i=1}^{6} q_i \tag{25}$$

The decision about the given test app $\boldsymbol{t}$ is taken after a comparison of the m_{PCR5} (B) with m_{th}. If $m_{PCR5}(\mathrm{B}) \geq m_{th}$ then the $\boldsymbol{t}$ is acknowledged as benign else it is taken as malign.

4. EVALUATION

Quantitative analysis on various performance matrices like Precision, Accuracy, F1 Score, Specificity, and Sensitivity was performed on the suggested and two other state-of-the-art methods. Their ROC (Receiver Operating Characteristic) plots were also drawn for comparison. The ROC curve is the plot of FAR (False Acceptance Rate) vs 1-FRR (False Rejection Rate) Following are the details of the evaluation process.

4.1 Databases

We choose about 3000 samples of malign apps and benign apps each downloaded from the benchmarked datasets i.e. Drebin[31], AMD[32], Androzoo[33], and CICMalDroid2020[34]. Benign apps are taken from Google Playstore. The malign apps generated both the benign and malign traffic as most of these malign apps are formed from the benign apps by their reverse engineering and repackaging after inserting maliciousness into them. Finally, a dataset is constructed by TCP-based network flows to assess the performance of the proposed framework. Network flows corresponding to benign and malign apps are stored in pcap format by running selected apps over and over for a period of 6 hrs. We select only those apps from the datasets that produce the network traffic and filter out the TCP-based flows. Traffic features as tabulated in Table 1 are extracted from these flows using the script written in python.

We select about 500 feature-flow vectors each corresponding to benign and malign flows from the total extracted flows corresponding to DB1, DB2, DB3, and DB4 datasets to train our model. We further integrate these 500 feature flow vectors of benign and malign flows into an integrated set of 2000 benign and malign flows each to form the fifth set of TCP features containing 2000 benign and malign feature vectors each. Further, we apply a ten-fold cross-validation procedure for training and testing are done on five sets of feature sets and finally, average values are taken as the results. All investigations are done on MATLAB R2018a on 16GB RAM, i7, and 2.7 GHz processor.

4.2 Performance Assessment

Performance of the suggested framework is realised by the means of TCP features extracted from TCP flows corresponding to malign and benign apps as in Table 2 and calculating evaluation matrices through ten-fold cross-validation. It is compared to training time and mean-time-to-detect (MTTD) against three state-of-the-art approaches. Here, MTTD is the time taken to detect malapps. Evaluation matrices results are also compared with three state-of-the-art techniques FED[12], MMD[20], and LWN[21].

Table 2. Experimentation dataset

App type Dataset	Malign_ Apps(M)	Benign_ Apps(B)	Source
DataBase(DB)1	750	750	Androzoo(M) GooglePlay(B)
DataBase(DB)2	750	750	AMD(M) CICMalDroid2020(B)
DataBase(DB)3	750	750	CICMalDroid2020(M) GooglePlay(B)
DataBase(DB)4	750	750	Drebin(M) GooglePlay(B)

The three state-of-the-art methods viz FED, MMD and LWN are compared with the proposed method. In FED, a malware detection technique is built on TCP n/w traffic, wherein network traffic generated by apps comprises TCP flows. These flows are used to extract packet sizes as features. Standard deviation, Mean, min, and max values of the sizes of the first few packets of TCP flows are taken as features. Here author leverages the RF ML algorithm to train the model to achieve 97 per cent of detection accuracy. In the second state-of-the-art method MMD, the authors proposed a framework for android malapp identification leveraging both the TCP and HTTPS features. Here, the malapp detection is done on the server-side without affecting the user experience. C4.5 Decision tree ML algo is used to train the model with 8312 benign and 5560 malign apps for identifying unknown apps with an accuracy of 97.89 per cent. In the LWN method, a lightweight malapp detection framework using TCP-based network features was given with an accuracy of 90 per cent and a false-positive rate of less than 3 per cent. Here, the total number of 359 malapp and benign apps are used along with two RF and AdaBoost ML algorithms.

Detection Accuracy, F1 Score, Specificity, Sensitivity, and Precision are calculated using equations 26, 27, 28, and 29 respectively.

$$Detection\,Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{26}$$

$$F1\,Score = \frac{2TP}{2TP + FP + FN} \tag{27}$$

$$Specificity = \frac{TN}{TN + FP} \tag{28}$$

$$Sensitivity = \frac{TP}{TP + FN} \tag{29}$$

$$\mathrm{Precision} = \frac{TP}{TP + FP} \tag{30}$$

where, TP, FP, TN, and FN are true positive, false positive, true negative, and false negative respectively. ROC curves for the proposed method and comparative methods FED, MMD, and LWN for flow sets corresponding to datasets DB1, DB2, DB3, and DB4 are depicted in Fig. 3. The experimental results are presented in Table 4. It is observed that the mean value of F

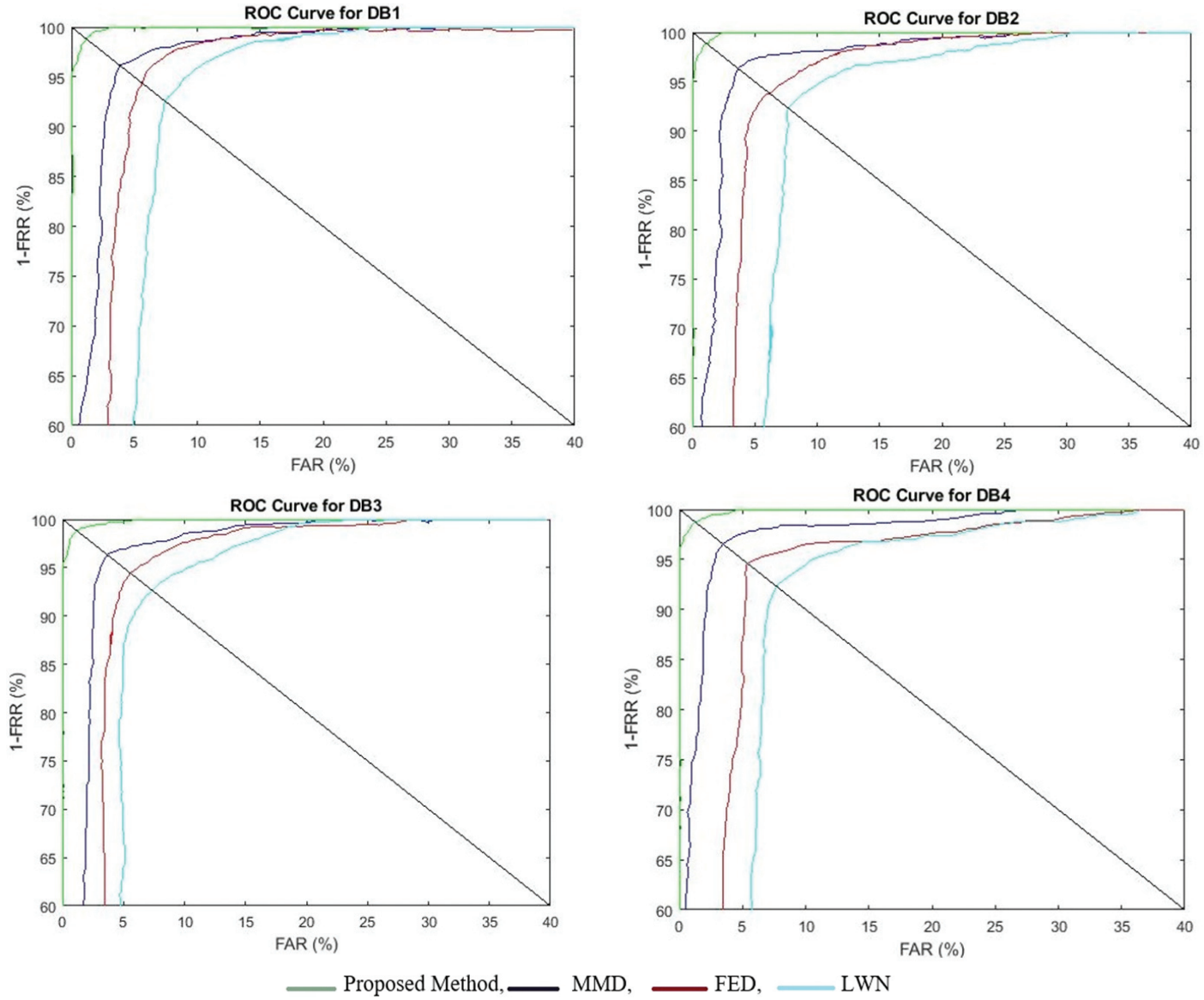

Figure 3. ROC curves for the proposed method and comparative methods MMD, FED, and LWN for flow sets corresponding to datasets DB1, DB2, DB3, and DB4.

score, detection accuracy, sensitivity, specificity, and precision for the proposed framework are 0.98742, 98.74 per cent, 99.04 per cent, 98.44 per cent, and 98.456 per cent respectively. The highest value of the F score, detection accuracy, sensitivity, specificity, and precision is 0.9890, 98.90 per cent, 99.80 per cent, 99.60, and 98.61 respectively.

The proposed technique outshines similar state-of-the-art traffic-based methods when evaluated on extracted TCP features as presented in Table 1. Enhancement for mean accuracy of the suggested scheme is achieved by 6.28 per cent, 4.34 per cent, and 2.3 per cent over LWN, FED, and MMD respectively. This enhanced value of accuracy of the suggested framework is attained by the optimal combination of traffic features using a cross-diffusion strategy and fusion of classifier(s) score using the DSmT-based PCR-5 rule to form a clear-cut border for differentiating malign apps from benign apps. Here, three feature vectors comprising the three sets of complementary features are fused in a nonlinear style through cross-diffusion wherein the rugged features are enhanced and the frail features are repressed resulting in the robust UF. Further, in the optimal score level fusion process, UF is fed to the three ML classifiers where concurrent scores are boosted and discordant classifier scores are suppressed. Also, the conflict between the multiple classifiers is resolved to get the final decision score.

The suggested framework is also compared to MTTD of different state-of-the-art techniques. To calculate the MTTD of contemporary methods, learned models are fed with the 250 arbitrary apps for investigation. Our proposed method attains an amazing average analysis performance of 5.8 seconds per app. Similarly, the average analysis performance of LWN, FED, and MMD comes out to be 6.5 seconds, 7.3 sec., and 6.9 seconds respectively. Hence, our proposed framework outshined other methods in respect of detection time, detection accuracy, and efficacy in real-life app scenarios. MTTD also confirms that the suggested framework detects the apps with high accuracy in a reasonable time.

Table 3. Performance comparison in terms of accuracy with existing methods using captured data from 20 different apps

Apps	UF+RF	LWN	FED	MMD	Proposed method
	Accuracy				
AccuRadio	0.8900	0.8300	0.8500	0.8900	0.9400
Maps	0.8700	0.8100	0.8400	0.8100	0.9100
WhatsApp	0.8600	0.8200	0.8100	0.8700	0.9300
Outlook	0.8300	0.8100	0.8300	0.8500	0.9100
Mail	0.7800	0.8400	0.8500	0.8400	0.9300
Netflix	0.9000	0.7700	0.8900	0.9100	0.9500
Twitter	0.9300	0.7500	0.7600	0.9200	0.9400
Facebook	0.9100	0.9000	0.8700	0.9300	0.9500
Youtube	0.8800	0.8600	0.8800	0.9200	0.9700
Gmail	0.9200	0.9300	0.8900	0.9500	0.9600
DroidDream	0.8100	0.8400	0.9000	0.9200	0.9800
DroidKungFu1	0.8700	0.8800	0.9300	0.9400	0.9600
Buzz	0.8200	0.8400	0.9100	0.8900	0.9400
BlueScanner	0.8500	0.8400	0.9400	0.9300	0.9800
Plankton	0.8200	0.8500	0.9200	0.9100	0.9600
WallpaperGirls	0.8100	0.8200	0.8300	0.8600	0.9300
StylePhotoCollage	0.8700	0.8400	0.9100	0.9200	0.9400
PrivateSms	0.8300	0.8500	0.9400	0.8700	0.9900
PartMessage	0.8800	0.9000	0.9500	0.8900	0.9700
IdeaSecurity	0.8200	0.8800	0.9000	0.9600	0.9800

Table 4. Performance Metrics (PM) for proposed methods and other comparative methods.

Dataset	PM	LWN	FED	MMD	Proposed Method
DB1 FLOWSET	Accuracy	0.9250	0.9440	0.9620	0.9880
	Specificity	0.9100	0.9280	0.9540	0.9960
	Sensitivity	0.9400	0.9600	0.9700	0.9800
	F1 Score	0.9261	0.9449	0.9623	0.9879
	Precision	0.9126	0.9302	0.9547	0.9959
DB2 FLOWSET	Accuracy	0.9230	0.9380	0.9630	0.9890
	Specificity	0.9060	0.9360	0.9660	0.9860
	Sensitivity	0.9400	0.9400	0.9600	0.9920
	F1 Score	0.9243	0.9381	0.9629	0.9890
	Precision	0.9091	0.9363	0.9658	0.9861
DB3 FLOWSET	Accuracy	0.9260	0.9440	0.9630	0.9881
	Specificity	0.9140	0.9420	0.9460	0.9780
	Sensitivity	0.9380	0.9460	0.9800	0.9980
	F1 Score	0.9269	0.9441	0.9636	0.9881
	Precision	0.9160	0.9422	0.9478	0.9784
DB4 FLOWSET	Accuracy	0.9220	0.9460	0.9660	0.9870
	Specificity	0.8860	0.9260	0.9560	0.9780
	Sensitivity	0.9580	0.9660	0.9760	0.9960
	F1 Score	0.9247	0.9471	0.9663	0.9871
	Precision	0.8937	0.9288	0.9569	0.9784
INTEGRATED FLOWSET	Accuracy	0.9270	0.9480	0.9680	0.9850
	Specificity	0.9405	0.9455	0.9645	0.9840
	Sensitivity	0.9133	0.9505	0.9715	0.9860
	F1 Score	0.9260	0.9481	0.9681	0.9850
	Precision	0.9388	0.9458	0.9647	0.9840

Performance comparison in terms of accuracy with existing methods using online captured data from 20 different apps chosen from diverse sources is shown in Table 3. Here, we have taken Smartphone Samsung Galaxy S9 having android 8 OS with 8 GB RAM and 64 GB storage for generating the network traffic. Our framework detects the apps that generate the flows similar to one generated during the training phase with high accuracy. The trained model is also tested on the network flows captured from the 20 different apps under an unconstrained environment using real smartphones instead of emulators and the performance comparison is done with our framework versus three different state-of-the-art methods viz. LWN, FED, and MMD and one self-proposed method RF+UF i.e. unified feature fed to the random forest algorithm. From Table 3, the average accuracy obtained when the random apps are tested on LWN, FED, MMD, RF+UF, and the proposed method is 85.75 per cent, 84.30 per cent, 88.82 per cent, 89.90 per cent, and 95.10 per cent respectively. Our proposed framework is dependent on the selected number of features. Generating the traffic flows in an unconstrained environment is cumbersome as traffic generation gets hindered when the internet connection is interrupted. Sometimes, the malign app took a significant amount of time to generate malicious behavior in traffic flow.

5. CONCLUSION

A novel network traffic analysis-based framework has been proposed in the manuscript to detect android malapps by exploiting the traffic features. It consists of four blocks viz. traffic feature fusion, classifier score-fusion, decision criteria, and reference apps update to accomplish efficient malapp detection Fifteen TCP-based features have been used in feature fusion to get a unified feature vector which is given to three Multilayer classifiers and find fusion scores for detecting malwares. The reference dictionary apps are regularly updated with zero-day malign and benign apps to improve the detection ability of the framework. It has been shown that the Framework proposed achieves an average detection accuracy of 98.74 per cent and outperforms other existing methods. A comparison of the evaluation matrices of the suggested framework with other contemporary approaches reveals better detection accuracy of malapps. Our future endeavor is to incorporate other dynamic and static attributes for detecting sophisticated malapps and to overcome all the limitations discussed.

REFERENCES

1. IDC. https://www.idc.com/promo/smartphone-market-share (Accessed on August 06, 2021).
2. Statistica. https://www.statista.com/statistics/277125/share-of-website-traffic-coming-from-mobile-devices (Accessed on August 06, 2021).
3. Yadav, A. & Ratan, R. Identifying traffic of same keys in cryptographic communications using fuzzy decision criteria and bit-plane measures. *Int. J. Syst. Assur. Eng. Manag*, 2020, **11**(9), 466–480. doi: 11. 10.1007/s13198-019-00878-7.
4. Ratan, R. & Yadav, A. Security analysis of bit plane level image encryption schemes. *Def. Sci. J.,*, 2021, **71**(2), 209-221. doi:10.14429/dsj.71.15643
5. Kumar, S.; Indu, S. & Walia, G.S. An efficient multistage fusion approach for smartphone security analysis *Def. Sci. J.,* 2021, **71**(4),476–490. doi: 10.14429/dsj.71.15077.
6. Kumar S.; Indu S. & Walia G.S. Smartphone traffic analysis: a contemporary survey of the state-of-the-art. *In* Proceedings of the Sixth International Conference on Mathematics and Computing. Advances in Intelligent Systems and Computing, Springer, Singapore, 2020, **1262**, 325-343 doi:10.1007/978-981-15-8061-1_26.
7. Yerima, S.Y. & Alzaylaee, M.K. & Sezer, S. Machine learning-based dynamic analysis of android apps with improved code coverage. 2019, *Eur. J. Inf. Secur.*, 2019, **1**, 1–24. doi: 10.1186/s13635-019-0087-1.
8. Agrawal, A.; Bhatia, A.; Bahuguna, A.; Tiwari, K.; Haribabu, K.; Vishwakarma, D. & Kaushik, R. A survey on analyzing encrypted network traffic of mobile devices. *Int. J. Inf. Secur.*, 2022, **1**. doi: 10.1007/s10207-022-00581-y.
9. Kumar, S.; Indu, S. & Walia, G.S. Optimal unification of static and dynamic features for smartphone security analysis. *Intell. Autom. Soft Co.*, 2022, **35**(1). doi:10.32604/iasc.2022.024469
10. Arora, A. & Garg, S. & Peddoju, S.K. Malware detection using network traffic analysis in android based mobile devices. *In* Eighth International Conference on Next Generation Mobile Apps, Services and Technologies, 2014, 66–71. doi: 10.1109/NGMAST.2014.57.
11. Arora, A. & Peddoju, S.K. NTPDroid: A hybrid android malware detector using network traffic and system permissions. *In* 17th IEEE Eighth International Conference on Trust. Secur. Priv. Comput. Commun., 2020, 808–813. doi: 10.1109/TrustCom/BigDataSE.2018.00115
12. Wang, S.; Yan, Q.; Chen, Z.; Yang, B.; Zhao, C. & Conti, M. Detecting android malware leveraging text semantics of network flows. *IEEE Trans. Inf. Forensics Secur.*, 2018, **13**(5),1096–1109. doi: 10.1109/TIFS.2017.2771228.
13. Liu, A.; Chen, Z.; Wang, S.; Peng, L.; Zhao, C. & Shi, Y. A fast and effective detection of mobile malware behavior using network traffic. *In* Algorithms and Architectures for Parallel Processing, edited by J. Vaidya and JoLi. 11337, Springer, 2018. doi:10.1007/978-3-030-05063-4_10
14. Li, Z.; Sun, L.; Yan, Q.; Srisaan, W. & Chen, Z. DroidClassifier: Efficient adaptive mining of application-layer header for classifying android malware. *In* International Conference on Security and Privacy in Communication Systems. **2017**, 597–616. doi: 10.1007/978-3-319-59608-2_33.
15. Li, Q.; Chen, Z.; Yan, Q.; Wang, S.; Ma, K.; Shi, Y. & Cui, L. MulAV: Multilevel and explainable detection of

android malware with data fusion. **11337,** 2018 In: *LNCS*. Springer International Publishing.
16. Wang, S.; Hou, S.; Lei, Z.; Chen, Z. & Han, H. Android malware network behavior analysis at HTTP protocol packet level. 2015. *In* Proceedings of the ICA3PP International Workshops and Symposiums on Algorithms and Architectures for Parallel Processing*.,* **9532**, 497–507. doi:10.1007/978-3-319-27161-3_45.
17. Su, X.; Lin, J.; Shen, F. & Zheng, Y. Two-phases detection scheme: Detecting android malware in android markets, *In* ATCI Springer **842**, 389–399, 2019. doi:10.1007/978-3-319-98776-7_41
18. Zulkifli, A.; Hamid, I.; Shah, W. & Abdullah, Z. Android malware detection based on network traffic using decision tree algorithm. *Adv. Intell. Syst. Comput.*, 2018, **700**, 485–494. doi: 10.1007/978-3-319-72550-5_46.
19. Malik, J. & Kaushal, R. CREDROID: Android malware detection by network traffic analysis. *In PAMCO*. 2016, 28-36. doi:10.1145/2940343.2940348.
20. Wang, S.; Chen, Z; Yan, Q.; Ji, K.; Peng, L.; Yang, B. & Conti, M. Deep and broad URL feature mining for android malware detection. *Information Science*, 2000, **513**, 600–613. doi: 10.1016/j.ins.2019.11.008.
21. Wang, S.; Chen, Z.; Yan, Q.; Ji, K.; Peng, L.; Yang, B. & Jia, Z. A mobile malware detection method using behavior features in network traffic. *J. Netw. Comput. Appl*., 2019, **133**, 15–25. doi: 10.1016/j.jnca.2018.12.014.
22. Sanz, I.J.; Lopez, M.A.; Viegas, E.K. & Sanches, V.R. "A lightweight network-based android malware detection system," In: *IFIP Networking Conference (Networking)*, 2020, 695-703.
23. Upadhayay, M.; Sharma, G.; Garg, G. & Arora, A. RPNDroid: Android malware detection using ranked permissions and network traffic, *In* Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*,* 2021, 19-24. doi: 10.1109/WorldS451998.2021.9513992.
24. Alshehri, M. APP-NTS: A network traffic similarity-based framework for repacked Android apps detection. *J. Ambient Intell. Human Comput.*, 2022, **13,** 1537–1546. doi: 0.1007/s12652-021-03023-0
25. Sihag, V.; Choudhary, G.; Vardhan, M.; Singh, P. & Seo, J. T. PICAndro: Packet inspection-based android malware detection 2021, *Secur. Commun. Netw*., 2021, 9099476, 11. doi: 10.1155/2021/9099476
26. Norouzian, M.R.; Xu, P.; Eckert, C. & Zarras, A. Hybroid: toward android malware detection and categorisation with program code and network traffic. *In*: Information Security. ISC 2021 Springer, Cham., *Lecture Notes Comput. Sci.*, **13118**, 259-278. . doi:10.1007/978-3-030-91356-4_14
27. Walia, G.S. *et al.* Unified graph-based multicue feature fusion for robust visual tracking. *IEEE T. Cybernetics*, 2020, **50**(6), 2357-2368. doi: 10.1109/TCYB.2019.2920289.
28. Xiao, Fuyuan. Multi-sensor data fusion based on the belief divergence measure of evidence and the belief entropy. *Information Fusion*, 2019, **46**, 23-32. doi: 10.1016/j.inffus.2018.04.003.
29. Xiao, Fuyuan. A new divergence measure for belief functions in D–S evidence theory for multisensor data fusion. *Information Sciences*, 2020, **514**, 462-483. doi: 10.1016/j.ins.2019.11.022.
30. Smarandache, Florentin & Dezert, J. Advances and applications of DSmT for information fusion, *Am. Res. Press*, 2015, **4**. http://fs.gallup.unm.edu/DSmT-book4.pdf. (Accessed on 10/09/2021).
31. Daniel, A.; Spreitzenbarth, M.; Hubner, M. & Gascon, H. DREBIN: Effective and explainable detection of android malware in your pocket. Symposium on Network and Distributed System Security (NDSS)*,* 2014. doi:10.14722/ndss.2014.23247.
32. Wei, F.; Li, Y.; Roy, S.; Ou, X.; Zhou, W. Deep ground truth analysis of current android malware. In: Polychronakis, M., Meier, M. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA) 2017. Springer, Cham. *Lecture Notes Comput. Sci.*, 2017, **10327**. doi:10.1007/978-3-319-60876-1_12
33. Mahdavifar, S.; Kadir, A.; Fatemi, R. & Alhadidi, D. Ghorbani. Dynamic android malware category classification using semi-supervised deep learning. *In* 18th IEEE International Conference on Dependable, Autonomic, and Secure Computing (DASC)*,* 2020, 17-24.
34. Allix, K.; Bissyandé, T.F.; Klein, J. & Traon, Y.L. AndroZoo: Collecting millions of android apps for the research community. 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories (MSR), 2016, 468-471. doi:10.1109/MSR.2016.056

CONTRIBUTORS

Mr Sumit Kumar received his ME in Electronics and Communication Engineering from Delhi College of Engineering, University of Delhi, Delhi. He is working as a Senior Scientist at DRDO-SAG, Delhi, India. His current areas of research interest include: Traffic analysis, smartphone security, cloud computing, and hardware analysis.
In the present study, he has carried out complete design and development scheme. He has also completed the experimental work by evaluating the suggested framework and comparing the results with other state-of-the-art methods.

Prof S. Indu received her PhD in the area of visual sensor networks from University of Delhi, Delhi, India. She is working as Dean (Student Welfare) and Professor of ECE Department of Delhi Technological University. Her areas of research interest are: Computer vision, sensor networks and image processing.
In the present study, she has provided expert guidance in problem formulation and offered necessary direction and overall support to carry out this study successfully.

Dr Gurjit Singh Walia obtained his PhD in the field of Computer Vision from Delhi Technological University (Formerly Delhi College of Engineering), Delhi. He is working as a Senior Scientist in DRDO, Delhi. His current research interests include: Machine learning, pattern recognition, and information security.

In the present study, he has guided the author and supervised the work.