Generate Optimal Number of Features in Mobile Malware Classification using Venn Diagram Intersection

Najiahtul Syafiqah Ismail[1†], Robiah Binti Yusof[2††] and Faiza MA[3†††]
najiahtul.ismail@gmail.com robiah@utem.edu.my faizalabdollah@utem.edu.my
Universiti Teknikal Malaysia Melaka

Summary
Smartphones are growing more susceptible as technology develops because they contain sensitive data that offers a severe security risk if it falls into the wrong hands. The Android OS includes permissions as a crucial component for safeguarding user privacy and confidentiality. On the other hand, mobile malware continues to struggle with permission misuse. Although permission-based detection is frequently utilized, the significant false alarm rates brought on by the permission-based issue are thought to make it inadequate. The present detection method has a high incidence of false alarms, which reduces its ability to identify permission-based attacks. By using permission features with intent, this research attempted to improve permission-based detection. However, it creates an excessive number of features and increases the likelihood of false alarms. In order to generate the optimal number of features created and boost the quality of features chosen, this research developed an intersection feature approach. Performance was assessed using metrics including accuracy, TPR, TNR, and FPR. The most important characteristics were chosen using the Correlation Feature Selection, and the malicious program was categorized using SVM and naive Bayes. The Intersection Feature Technique, according to the findings, reduces characteristics from 486 to 17, has a 97 percent accuracy rate, and produces 0.1 percent false alarms.

Keywords:
Mobile malware, Classification, Permissions, Intersection Technique, Intents

1. Introduction

The usage of mobile devices as personal computers is on the rise in the modern day. A cell phone's fundamental function is to promote human interaction, but as technology develops, it will eventually add more functions and processing power. These days, a mobile phone can perform any task that a desktop or laptop can. Smartphones are equipped with the tools necessary to conduct phone calls, send and receive text messages, make high-definition video calls, access social media, send and receive emails, use financial services, and take, record, and edit high-definition pictures, videos, or music.

The most popular mobile operating system worldwide in 2020, as per [1], was the Android operating system. The main cause is the open Android policy, which permits programmed repackaging and simple publishing, which gave rise to numerous security risks and quick Android malware evolution. Giving to [2], in response to the numerous demands from Android users for the ability to acquire premium software at no cost, numerous third-party app stores have been created, inviting various application developers to publish their repackaged applications. This situation, according to [3] and [4] also encourages malware developers to take a chance by downloading dangerous repackaged versions of expensive software to infect consumer mobiles.

Malware is continually developing even if the existing detection technology can detect Android malware. According to [5] and [6], the malware's primary goals are often to steal user information and make money by sending premium-rate SMS messages without the users' permission. Due to its opensource approach compared to other operating systems, Android has attacked many people; nevertheless, it also turns into a malware attacker. Attackers create malicious copies of good apps, especially premium ones, and release them on the Play Store or other third-party markets to lure customers in. Once malicious applications have been installed on a user's device, harmful actions like data modification or deletion are possible. According to [7], [8] and [9] collecting user information or sending SMS or email without user authorization. To safeguard the privacy and interests of Android users, it is therefore extremely important to identify malware successfully. As described by [10]. The ability to access third-party applications exposes mobile devices to permission-based attacks.

The remainder of the paper is structured as follows: the related works involve in this paper is discussed Section 2 and Section 3 introduced and discuss the detail about the technique proposed. In Section 4, it discussed and shows the results and evaluation. Section 5 summarization and suggest ideas for further research.

2. Related Works

Mobile malware detection techniques, according to [11], [12] and [13], are comparable to desktop environments where they may be divided into three primary categories: signature-based detection, anomaly-based detection, and specification-based detection. The signature-based detection method matches malware feature patterns to rules-based malware pattern patterns in the database. Any

Manuscript received July 5, 2022
Manuscript revised July 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.7.48

activity that differs from the typical profile behaviour is examined using an anomaly-based detection approach. The specification-based detection method, meanwhile, derives from anomaly-based detection and includes a training step when users are taught the anticipated behaviour of the application or system under scrutiny. Anomaly-based and specification-based detection, as stated by [14], may identify unknown malware, but they take a long time and have a high false-positive rate. In contrast, a signature-based assault has a low percentage of false positives and produces excellent accuracy for known malware. The feature vectors derived from malicious programs determine the accuracy of the signature-based detection method. As a result, it has emerged as a crucial stage in malware detection for efficiently extracting a crucial characteristic.

In this study, the ideas of Venn Diagrams are introduced, and the Intersection technique, permission-based feature extraction, and intent-based feature extraction are proposed. The intersection technique involves creating a feature vector including features for permissions and intents, which is then subjected to feature tuning before being selected as a feature. These feature vectors will be used in a classification procedure to assess the proposed approach [15]. According to the experimental findings, the Intersection Technique of Permission and Intent Feature performed better than other approaches and could reach high accuracy based on our data set.

The use of Venn diagrams to find and overlap area is not uncommon in another research field. The Venn Diagram is popular in genetic research. It is proven by [16], where they use the Venn Diagram to find gene sets that describe particular processes using these cross-comparisons by layered gene lists from several pair-wise experimental contrasts. Besides, [17] implement a Venn diagram-based analytic approach that selects genes from experiments using network propagation on protein-protein interaction networks. According to [18], they applied the Venn Diagram to find common gene from various platform that consists of thousands of genes. Meanwhile,

Android had built a permission-based paradigm in order to prevent an application from accessing system-sensitive resources. According to [19], in order to access sensitive resources, application developers must clearly specify their request in the Android Manifest File. Consistent with [20], an Android application with restricted access employs a permission mechanism to access mobile critical resources. For instance, an application has to ask for Android.permission in order to send messages using an SMS application. During installation, SEND SMS is sent from the mobile device.

In addition to permission, it also depends on other elements like accessing system resources intents [21]. As mentioned by [22], the requester's activity is described in the requester's intent, which serves as a communication tool. In contrast, intent, according to [23] is a request from an application to carry out a certain activity. Explicit and implicit intent are the two categories of intent. As stated in [24], asserts that an explicit intent component can be started by requesting a particular receiver name, such as the names of classes. Thus, the system is aware of just which application to ask for. While implicit intent asks for generic action to be taken, this intent is used by developers to launch an activity or service in their application. This intent will be handled by other components that can carry out such activities. For instance, an application can utilise an implicit intent (Android.intent.action.DIAL) to launch a dialer activity if it wishes to make a phone call. However, the system will prompt users to choose whether to install additional dialer applications.

Based on [25], the permission-based paradigm is continually vulnerable to security concerns such permissions leaking to take advantage of mobile devices' weaknesses. Permission escalation, in accordance with [26], permits malware to access sensitive resources without permission. On the contrary stated by [27] malware will often declare many permissions in order to complete its purpose, hence over-claimed permissions are used to get around permission-based detection mechanisms. If too many rights are claimed, hostile applications may seek access to personal user data in attacks that include permission escalation. Few studies have been done on android intents, despite major study being done to examine the permission model and Intent functionality for identifying mobile malware.

The goal of this study is to use the integration of permission and intent characteristics to defeat the permission-based attack. However, too many sub-features are generated by the Permission and Intent usage, which has an impact on accuracy and false-positive rate. The goal of the suggested approach is to produce the ideal amount of features from permission and intent while simultaneously ensuring an improvement in accuracy rate and a decrease in false-positive rate.

Users of the Android operating system have the freedom of being able to install applications from the third-party market. Android devices are secured via a permission-based architecture that limits third-party applications' access to critical resources, such short messaging system databases and external storage, in order to ensure the user's safety. However, mobile malware still poses a concern via misuse of permissions.

Static characteristics have been employed as part of several detection techniques. as an analogy, to identify Android malware, [25], [28], [29], [30], [31], [32], [33] and [34] employed permissions, intents, or a combination of manifest file components.

As claimed by [35], The misuse of permission may result in catastrophic security breaches, therefore the permission induced attack is one of the most frequent and dangerous vulnerabilities that puts Android users in peril. It

explains four types of permission-induced attacks: passive data leak, identical custom permission, unsafe pendingIntent, and privilege escalation. According to [29], which provides evidence in support of this claim, application developers may seek or define unauthorized privileges, leading to overprivileged programs, since there is a dearth of reliable permission data.

As mentioned by [36], they introduced a behaviour detection approach based on the Latent Semantic Indexing concept in accordance with. By selecting which permission category to match from and then analysing behaviors in an emulation environment, the suggested technique sought to find anomalous Android applications. Both static and dynamic analysis is used by them. Latent Semantic Indexing will be used to retrieve data during the static phase. Obtaining a list of applications that are comparable is the aim of this method. In the meanwhile, they used dynamic analysis to check the projected behaviors of the permission list by looking at the invokes functions.

They suggest identifying cooperating programs as well as unidentified malicious applications since permissions and intents enable application collusion, as noted in [37]. Applications are categorized by the proposed malware detection system based on particular permissions and intents that are only present in malware. Use of ensemble methods optimizes the classification results.

Despite the fact that many signature-based detection systems use static features, and it is effectively increased the accuracy, nevertheless it generates a considerable number of false alarm and number of \features. The introduce technique in this paper is to resolve the problems.

3. Proposed Intersection Features Technique

Permission and intent are static features, as stated by [38]. Both features are important in distinguishing between applications that are malicious and those that are benign. In this research, we applied static analysis by reverse engineering the process to extract the features from application. Static analysis, as describe by [39], leverages the resources of the program to identify it as malicious or benign while examining it without running it. The datasets utilized in this study are from the University of New Brunswick (Abdul Kadir et al., 2015) and they contain 2909 Android malware programs from 14 distinct families that have existed between 2010 and 2019.

The Intersection Technique consists of several stages to generate quality and optimal number of features. There are four stages in the Intersection Technique: Feature Extraction, Feature Vector, Feature Tuning and Feature Selection. The details will discuss below. The details of the stages will discuss in the subsection.

3.1. Feature Extraction

Mobile devices running Android use the .APK file type to download and install applications. The Android application files were disassembled into a readable format using the ApkTool (iBotPeaches, 2019), which produced dex files, manifest.xml files, and smali files. The ApkTool Command example is shown in Fig. 1.

```
>> java –jar apktool.jar
>> apktool d appname.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources... I: Decoding values */*
XMLs...
I: Baksmaling classes.dex...
...
...
>> apktool b appname
I: Smaling smali folder into classes.dex... I: Building resources...
I: Building apk file...
I: Built apk...
```

Fig.1 Example of ApkTool Command

Fig. 1 shows the example of Apktool and the command use to decompile the file in an application. The manifest file, where the developer specifies the request, Permission, and Intent filters used in the application, is the subject of concern [22].

```
<?xml version="1.0" encoding="utf-8" standalone="no"?><manifest
xmlns:android="http://schemas.android.com/apk/res/android" package="com.ducky.tracedrawingLite">
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
  <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>

  <uses-permission
android:name="com.samsung.android.providers.context.permission.WRITE_USE_APP_FEATURE_SURVEY"/>
  <application android:allowBackup="true" android:icon="@drawable/ic_launcher"
android:label="@string/app_name" android:largeHeap="true" android:theme="@style/AppTheme">
    <activity android:configChanges="keyboardHidden|orientation|screenSize" android:label="■ Trace
Artist!" android:name=".MainActivity" android:screenOrientation="portrait"
android:theme="@android:style/Theme.Holo.Light.NoActionBar.Fullscreen">
      <intent-filter>
        <action android:name="android.intent.action.MAIN"/>
        <category android:name="android.intent.category.LAUNCHER"/>
      </intent-filter>
      <intent-filter>
        <action android:name="android.intent.action.SEND"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <data android:mimeType="image/*"/>
      </intent-filter>
    </activity>
    <activity android:configChanges="keyboardHidden|orientation|screenSize"
android:label="@string/title_activity_editor" android:name=".Editor" android:screenOrientation="portrait"
android:theme="@android:style/Theme.Holo.Light.NoActionBar.Fullscreen"/>
  </application>
</manifest>
```

Fig.2 Example of Manifest File

Fig. 2 shows the example of Permission and Intent features extracted in this stage. There are 150 manifest files generated from 150 malicious Android applications. The

extracted Permissions and Intent were matched with standard Permission and Intent release by Android system using string similarity method, and the additionally extracted sub features also recorded. The equation shown in Eq.1 where if extracted Permission and Intent features exist, it denoted as 0 for the nonexistence of the feature. Let R be a vector containing a set of 466 Android features consist of Permission and Intent features. For every ith application in the Android application dataset (malware and benign),

$$R_i = \{r_1, r_2, r_3, \ldots r_j\} \text{ and}$$

$$j \begin{cases} 1, & \text{If feature } \mathbf{j}_{th} \text{ exist} \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

Based on The Eq.1, in this stage, all identified features from an application will be labelled as 1, while the unused features will be labelled as 0. The result of this phase will be used in the feature vector phase.

3.2. Feature Vector

This stage will create a comma-separated value (CSV) file from the binary value of an extracted feature from Permission and Intent. The vector's final result was 1 or 0, indicating the presence of malware or benign application. [40]. Fig. 4 shows the example of feature vectors result.

0,	0,	1,	1,	0,	0,	….	1

Fig.4 Feature vector

3.4. Feature Tuning

When two or more features from different feature sets that go through the same procedure intersect, an Intersection Feature is formed. For example, two features, sendTextMessage API function and SEND SMS Permission, go through the same process, and intersect, resulting in the creation of an Intersection Feature containing these two features. The detection system can benefit from features in the Intersection Feature. Table 1 shows the concept of Intersection Feature.

In the same Intersection, features are included. Features may alter one other's feature values. Malware, for example, can get around the Permission-based detection process by not requesting the SEND_SMS permission. As a result, the value of this permission in the malware feature vector was 0. Even yet, if malware used the sendTextMessage API Call method, the SEND_SMS Permission value was set to 1. Fig. 5 illustrates the concept.

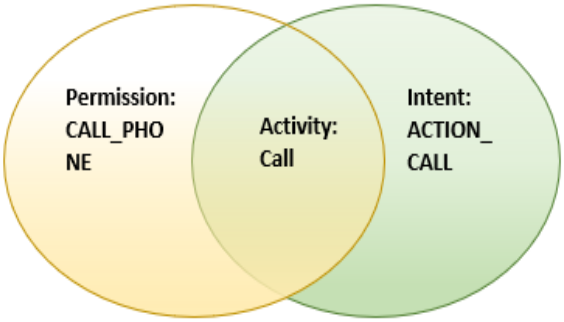

Fig.5 Demonstrated of Feature Intersection diagram

Fig. 5 depicts the Venn Diagram of the Intersection Feature method proposal, which consists of the intersection of two feature sets that share the same resource: Permission and Intent. A resource is a process that occurs on a smartphone, such as dialling a phone number, sending an SMS, watching a movie, and so on. Based in Fig. 5, the application request to make a call activity. So, both permission and Intent will access to the same resources to complete the activity. The activity will be the key to find the intersection of the features.

Based on [41], asserts that some features offer extremely good discrimination. However, the selection of features during the feature selection step will be impacted by the dataset's few apps' requests for it. Enhancing the quality of features is the aim of feature tuning. The removal of unused additional features and the alteration of feature values within the same Intersection feature make up this phase's two parts. Table 1 shows the example of Intersection Features between Permission and Intent.

Table 1: Example Permission and Intent from The Same Activity

Permission	Intent
read_phone_contacts	Action.phone_state
Send_sms	Action.sendto
call_phone	Action_call
Access_wifi_state	Connectivity_change

```
1 : xlxs > an incoming excel file
2 : malicious_intent [n1]= List of dangerous intent value
3:    If malicious_intent = 1,
4:    then
5:       intersection_permission=1
6:    end if
7:       end for
```

Fig.6 Pseudocode of Changing Permission Value Based on Intent Value

Fig.6 and and Fig.7 shows the pseudocode implementation of the Intersection Technique Fig. 5 shows the pseudocode of changing permission value based on intent value. Only features that use the same resources, like Table 1, are impacted by the changing value. The permission value is equal to 1 for each specified Intent value of 1.

```
M = malicious applications
N = benign applications
Sa = sets of attributes
i = i-th attributes in Sa (Si = S1, s2, s3..., si+1)
S0 = 0.1

Input : F1, f2, f3,.., fM+N
Output : S'a (with value for each i-th)
S'p ←Ø
For i-the attributes in Sa do
Si ←(Mi+Ni)/M+N
    If Si ≥0.1      then
      Add i-th attributes in Sa into S'a
    End if
    Print S'a
End for
```

Fig.7 Removing extra features infrequently used

The pseudocode to eliminate unused features is shown in Fig. 6. If each feature's value, Si, is below the threshold, S0, the features will be eliminated. This is due to the fact that some Permissions and Intent features were developed alone by their developer and that their feature's name may have been misspelt in some applications. The features that will be employed in the next phases will be determined at this decisive phase. Furthermore, not all malicious applications will use the same features, but employing a large number of samples in an experiment can assist to identify the features that malware most frequently requests.

3.5. Feature Selection

The subset selection of critical characteristics is the aim of the feature selection step. Feature selection is a process to get a subset from an original feature that chooses the pertinent characteristics of the dataset, according to [42]. Supported by [43], feature selection poses a serious problem in classification since it may have an impact on accuracy result. The feature selection helps reduce the dataset's number of features and helps to use less memory and CPU. Gain Ration, Chi-Square, and Information Gain were three types of feature selection algorithms that were taken into account in this study to assess how well the Intersection Technique worked. As stated by [44], Gain Ratio, Chi-square Information Gain algorithm gives a great performance. Additionally, as confirmed by [45], Information Gain offers consistent performance for a set of features, whereas Gain Ratio exhibits superior results for the selection of a sizable number of sensitive features. Furthermore, [46] demonstrated that Chi-Square generates good accuracy with less features. Chi-Square was thus employed in this research as the feature selection approach. Only high-quality features will be chosen during the feature selection step since the unusable features in the prior process were discarded.

4. Results and Discussions

The purpose of this research is to find whether the generated optimal number of features proposed by Intersection Technique will increase the accuracy detection results and lower the false alarm result. In order to fit the purpose, several evaluation metrics are modified to serve the objective of evaluating the success of the intersection feature technique. The measures for assessment and the findings are explained in this section.

4.1. Evaluation Metrics

In order to evaluate the detection method, several evaluation metrics were used.

- •True Positive (TP): number of malicious applications precisely classified as positive.
- •False Positive (FP): number of benign applications falsely classified as positive.
- •False Negative (FN): number of malicious applications falsely classified as negative.
- •True Negative (TN): number of benign applications precisely classified as negative.

$$\text{False Positive} = \frac{FP}{FP+TN}$$

$$\text{Accuracy} = \frac{TP+TN}{TN+FN+TP-FP}$$

$$\text{Recall} = \frac{TP}{FN+TP} \quad (2)$$

4.2. Evaluation Results

The results of the proposed technique's implementation are highlighted in this section. Two scenarios have been created in order to assess the improved performance while employing Permission and Intent features which are classification result before feature tuning and classification result after feature tuning.

To determine the effectiveness of the feature intersection technique, the SVM classifier was used in the classification phase on the created dataset. Table 2 displays the accuracy and false positive attained by applying feature

tuning to the Permission feature and the Intent feature in two separate classification methods.

Table 2. Effectiveness Result of Intersection Technique

	With Intersection Feature (%)	Without Intersection Feature (%)
Accuracy	94.8	86
False Positive	0.07	0.18

In Table 2, the accuracy of classification for Intersection Feature is higher than without using the proposed technique. The experiment has been done three times. The showing result in Table 2 is the mean results of Accuracy and False Positive Value. The technique also achieves to reduce the false-positive rate to 0.07. The lowest false positive results and the highest accuracy result prove the effectiveness of the Feature Intersection Technique.

An independent samples T-test was used to compare the mean accuracy (ACC) score of Without Intersection Feature (n=3) and With Intersection Feature (n=3). Levene's test was significant; thus, an equal variance assumed for both groups. As P value= 0.001 (Sig. 2 tailed value) is less than 0.05, t test is statistically significant. So, we managed to reject the null hypothesis. Therefore, there is significant difference in average accuracy (ACC) exists between Without Intersection Feature and With Intersection Feature. So, we can conclude that average accuracy (ACC) of Without Intersection feature and With Intersection Feature are statistically significant.

4. Conclusion and Future Work

Compared to other forms of Android malware, the Android botnet poses a major threat to smartphone users. Numerous Android malware detection techniques are put into place based on earlier studies without taking into account diverse feature set circumstances. Additionally, the majority of detection methods just employ basic Android botnet characteristics without taking into account their unique traits. As a consequence, the Android botnet is able to avoid most suggested techniques. The Intersection Features technique is suggested in this work while taking into account various Permissions and Intent features. Some intersection characteristics' values are changed by the proposed technique. As a result, the accuracy rate rises to 94.8 percent and the false positive rate reduces from 1.8 to 0.07. Concurrently, the main objective to generate optimal number of features has been achieved where the number of features has decreased from a total of 486 features to 17.

For future research, features from dynamic analysis such as system call should be applied to achieve better performance and increase detection accuracy. Although this technique is achieved excellence results, however the number of malicious application samples during experiment also play a major role. This experiment was done using 150 samples from the same sources' dataset, but in the future more samples will be collected from various sources to distinguish mobile malware features.

Acknowledgments

We want to express our appreciation to the Universiti Teknikal Malaysia Melaka (UTeM) for their constant support in getting the authors to publish this work.

References

[1] StatCounter, "Mobile Operating System Market Share Worldwide," *StatCounter*, 2021. .

[2] W. J. Buchanan, S. Chiale, and R. Macfarlane, "A methodology for the security evaluation within third-party Android Marketplaces," *Digit. Investig.*, vol. 23, pp. 88–98, 2017.

[3] F. Martinelli, F. Mercaldo, V. Nardone, A. Santone, and C. A. Visaggio, "Identifying mobile repackaged applications through formal methods," *ICISSP 2017 - Proc. 3rd Int. Conf. Inf. Syst. Secur. Priv.*, vol. 2017-Janua, no. Icissp, pp. 673–682, 2017.

[4] S. Kandukuru and R. M. Sharma, "Android malicious application detection using permission vector and network traffic analysis," *2017 2nd Int. Conf. Converg. Technol. I2CT 2017*, vol. 2017-Janua, pp. 1126–1132, 2017.

[5] J. Duarte, "A Survey of Android Attacks Detection Techniques," in *Digital Privacy and Security Conference (DPSC)*, 2020, no. February, pp. 106–117.

[6] C. Chen, J. Lin, and G.-H. Lai, "Detecting Mobile Application Malicious Behaviors Based on Data Flow of Source Code," in *International Conference on Trustworthy Systems and their Applications Detecting*, 2014, pp. 1–6.

[7] W. Wang, Z. Gao, M. Zhao, Y. Li, J. Liu, and X. Zhang, "DroidEnsemble: Detecting Android Malicious Applications with Ensemble of String and Structural Static Features," *IEEE Access*, vol. 6, no. c, pp. 31798–31807, 2018.

[8] G. Kirubavathi and R. Anitha, "Structural analysis and detection of android botnets using machine learning techniques," *Int. J. Inf. Secur.*, vol. 17, no. 2, pp. 153–167, 2018.

[9] P. Feng, J. MA, C. SUN, X. XU1, and Y. MA, "A Novel Dynamic Android Malware Detection System With Ensemble Learning," *IEEE Trans. JOURNALS*, vol. 4, no. c, 2018.

[10] F. M. Faqiry, "Scrutinizing Permission Based Attack on Android Os Platform Devices," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 7, pp. 421–426, 2017.

[11] V. Kouliaridis, G. Kambourakis, D. Geneiatakis, and N. Potha, "Two anatomists are better than one-Dual-level android malware detection," *Symmetry (Basel).*, vol. 12, no. 7, pp. 1–21, 2020.

[12] M. Al Ali, D. Svetinovic, Z. Aung, and S. Lukman, "Malware Detection in Android Mobile Platform using

Machine Learning Algorithms," in *International Conference on Infocom Technologies and Unmanned Systems (Trends and Future Directions) (ICTUS)*, 2017, pp. 4–9.

[13] P. Yan and Z. Yan, "A survey on dynamic mobile malware detection," *Softw. Qual. J.*, vol. 26, no. 3, pp. 1–31, 2017.

[14] A. K. Chakravarty and S. Paul, "A study of signature-based and behaviour-based malware detection approaches," *Int. J. Adv. Res. Ideas Innov. Technol. ISSN*, vol. 5, no. 3, pp. 1509–1511, 2019.

[15] D. Ö. Şahin, O. E. Kural, S. Akleylek, and E. Kılıç, "A novel permission-based Android malware detection system using feature selection based on linear regression," *Neural Comput. Appl.*, vol. 1, p. 5875, 2021.

[16] P. Wijesekera, A. Baokar, L. Tsai, and J. Reardon, "The Feasibility of Dynamically Granted Permissions: Aligning Mobile Privacy with User Preferences," *Proc. - IEEE Symp. Secur. Priv.*, pp. 1077–1093, 2017.

[17] Z. Abdullah and M. M. Saudi, "RAPID-Risk Assessment of Android Permission and Application Programming Interface (API) Call for Android Botnet," no. October, 2018.

[18] H. Bagheri, E. Kang, S. Malek, and D. Jackson, "A formal approach for detection of security flaws in the android permission system," *Form. Asp. Comput.*, vol. 30, no. 5, pp. 525–544, 2018.

[19] B. F. Demissie, M. Ceccato, and L. K. Shar, "Security analysis of permission re-delegation vulnerabilities in Android apps," *Empir. Softw. Eng.*, vol. 25, no. 6, pp. 5084–5136, 2020.

[20] M. W. Afridi, T. Ali, T. Alghamdi, T. Ali, and M. Yasar, "Android application behavioral analysis through intent monitoring," in *International Symposium on Digital Forensic and Security, ISDFS 2018*, 2018, vol. 6, pp. 1–8.

[21] A. Feizollah, N. B. Anuar, R. Salleh, G. Suarez-Tangil, and S. Furnell, "AndroDialysis: Analysis of Android Intent Effectiveness in Malware Detection," *Comput. Secur.*, vol. 65, no., pp. 121–134, 2017.

[22] R. Chang *et al.*, "Towards a multilayered permission-based access control for extending Android security," *Concurr. Comput.*, vol. 30, no. 5, 2018.

[23] L. Shi, J. Fu, Z. Guo, and J. Ming, "'Jekyll and hyde' is risky: Shared-everything threat mitigation in dual-instance apps∗," in *MobiSys 2019 - Proceedings of the 17th Annual International Conference on Mobile Systems, Applications, and Services*, 2019, pp. 225–235.

[24] S. Kumar, R. Shanker, and S. Verma, "Context aware dynamic permission model: A retrospect of privacy and security in android system," in *Proceedings - 2nd International Conference on Intelligent Circuits and Systems, ICICS 2018*, 2018, pp. 330–333.

[25] J. Xiao, S. Chen, Q. He, Z. Feng, and X. Xue, "An Android application risk evaluation framework based on minimum permission set identification," *J. Syst. Softw.*, vol. 163, p. 110533, 2020.

[26] R. Kumar, X. Zhang, R. Khan, and A. Sharif, "Research on Data Mining of Permission-Induced Risk for Android IoT Devices," *Appl. Sci.*, vol. 9, no. 2, p. 277, 2019.

[27] A. Bhattacharya and R. T. Goswami, "A Hybrid Community Based Rough Set Feature Selection Technique in Android Malware Detection," in *Smart Trends in Systems, Security and Sustainability.*, 2018, no. 18, pp. 249–258.

[28] C. La, P. Myo, and K. M. Myo, "Permission-based Feature Selection for Android Malware Detection and Analysis," *Int. J. Comput. Appl.*, vol. 181, no. 19, pp. 29–39, 2018.

[29] M. Hammad, H. Bagheri, and S. Malek, "DELDROID: An automated approach for determination and enforcement of least-privilege architecture in android," *J. Syst. Softw.*, vol. 149, pp. 83–100, 2019.

[30] W. Y. Lee, J. Saxe, and R. Harang, "SeqDroid: Obfuscated android malware detection using stacked convolutional and recurrent neural networks," *Adv. Sci. Technol. Secur. Appl.*, pp. 197–210, 2019.

[31] M. Fan, J. Liu, W. Wang, H. Li, Z. Tian, and T. Liu, "DAPASA: Detecting Android Piggybacked Apps Through Sensitive Subgraph Analysis," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1772–1785, 2017.

[32] A. Sadeghi, R. Jabbarvand, N. Ghorbani, H. Bagheri, and S. Malek, "A temporal permission analysis and enforcement framework for Android," in *Proceedings of the 40th International Conference on Software Engineering - ICSE '18*, 2018, pp. 846–857.

[33] H. Shahriar and M. Islam, "Android Malware Detection Using Permission Analysis," *IEEE Conf. Proc.*, vol. 2017, no. SoutheastCon, pp. 1–6, 2017.

[34] F. I. Abro, M. Rajarajana, and T. Chen, "Mobile Malware detection with permissions and intents analysis," p. 2014, 2014.

[35] M. Yusof, M. M. Saudi, and F. Ridzuan, "A new mobile botnet classification based on permission and API calls," in *Proceedings - 2017 7th International Conference on Emerging Security Technologies, EST 2017*, 2017, pp. 122–127.

[36] A. Talha and I. Alper, "An in-depth analysis of Android malware using hybrid techniques," *Digit. Investig.*, vol. 24, pp. 25–33, 2018.

[37] M. K. Alzaylaee, S. Y. Yerima, and S. Sezer, "Improving Dynamic Analysis of Android Apps Using Hybrid Test Input Generation," in *International Conference On Cyber Security And Protection Of Digital Services*, 2017, pp. 1–8.

[38] C. Wang and Y. Lan, "PFESG : Permission-based Android Malware Feature Extraction Algorithm," in *ICNCC 2017: Proceedings of the 2017 VI International Conference on Network, Communication and Computing*, 2017, pp. 106–109.

[39] J. Cai, J. Luo, S. Wang, and S. Yang, "Feature selection in machine learning: A new perspective," *Neurocomputing*, vol. 300, pp. 70–79, 2018.

[40] E. M. Karabulut, S. A. Özel, and T. İbrikçi, "A comparative study on the effect of feature selection on classification accuracy," *Procedia Technol.*, vol. 1, pp. 323–327, 2012.

[41] A. Sharma and S. Dey, "Performance Investigation of Feature Selection Methods and Sentiment Lexicons for Sentiment Analysis," *Int. J. Comput. Appl.*, no. June, pp. 15–20, 2012.

[42] R. Kaur and M. Sachdeva, "Study and Comparison of

Feature Selection Approaches for Intrusion Detection," in *Proceedings on International Conference on Advances in Emerging Technology*, 2016, vol. 2, pp. 1–7.

[43] X. Liu and J. Liu, "A two-layered permission-based android malware detection scheme," *Proc. - 2nd IEEE Int. Conf. Mob. Cloud Comput. Serv. Eng. MobileCloud*, pp. 142–148, 2014.

Najiahtul Syafiqah Ismail, currently pursuing her PhD degree in the Faculty of Information and Communication Technology, University Technical Melaka (UTeM). She received a BSc (Hons) in Computer Science and the MSc in Information Technology from University Technical Malaysia Melaka (UTeM). Her research interests include computer networking, computer security and mobile security.

Robiah Yusof, currently a Senior Lecturer in the Universiti Teknikal Malaysia Melaka (UTeM). She received the BSc (Hons) of Computer Studies and Master of Information Technology from Liverpool John Moore's University, UK and Universiti Kebangsaan Malaysia, respectively. She obtained the Doctor of Philosophy, Network Security from Universiti Teknikal Malaysia Melaka (UTeM). Her research interests include network security, computer system security, and network design.

Mohd Faizal Abdollah currently working as a Senior Lecturer in the Department of Computer and Communication System, Faculty of Information and Communication Technology, University Technical Malaysia Melaka (UTeM). He obtained his Master and bachelor's degree from University Kebangsaan Malaysia and University Utara Malaysia. He achieved his PhD from University Technical Malaysia Melaka in Computer and Network Security. Formerly, he worked as an MIS Executive at EON Berhad, Selangor and as a System Engineer in Multimedia University, Melaka, for six years. His interest is mainly in network and wireless technology & network and wireless security.