Methods to Select Features for Android Malware Detection Based on the Protection Level Analysis

Chaeeun Lee, Eunnarae Ko, and Kyungho Lee(✉)

Institute of Cyber Security and Privacy (ICSP), Korea University, Seoul 02841, Korea
{katey95415,eun13,kevinlee}@korea.ac.kr

Abstract. Android's permission system is asked to users before installing applications. It is intended to warn users about the risks of the app installation and gives users opportunities to review the application's permission requests and uninstall it if they find it threatening. However, not all android permissions ask for the user's decision. Those who are defined as 'Dangerous' in the permission protection level are only being confirmed by the users in Android Google Market. We examine whether the 'Dangerous permissions' are actually being a main component of detection when it comes to defining the app as malicious or benign. To collect important features and to investigate the correlation between the malicious app and the permission's protection level, feature selection and deep learning algorithms were used. The study evaluates the feature by using the confusion matrix. We used 10,818 numbers of malicious and benign applications, and 457 permission lists to investigate our examination, and it appeared that 'Dangerous' permissions may not be the only important factor, and we suggest a different perspective of viewing permissions.

Keywords: Android application · Permission · Protection level · Malware detection · Feature selection · Classification · Deep learning

1 Introduction

An investigation report by International Data Corporation (IDC) expects that the overall smartphone market will reach to 1.511 billion units in 2024. In the same year, the Android market will still be in the first place of the OS Market with 87% share, while Apple's iOS will be accounted for second place with 13% share [11]. As of December 2019, the Google Play Store consists of 2.87 million applications which are consisting of a wide range of contents, such as music, magazines, books, film, games, and TV [18].

Permissions are divided into four levels of protection level. Among them, dangerous level permissions are defined as higher-risk permissions, which is required to be confirmed by an Android user since they can cause harmful impact to the user and device. In our paper, we performed a research to examine how effective dangerous level permissions are when detecting malware applications. Therefore, we made a feature selection of our dataset to see which permissions are being important factors. 70 different types of permissions were selected via the Weka tool to detect malware applications. We used four different deep learning algorithms to see the accuracy of the detection and calculated them with the confusion matrix. All of the selected features resulted in a high score; all of them were above 90% accuracy. We now look in-depth to see what kind of permissions were being selected to be important and figured out what particular protection level was considered more when detecting malware applications.

2 Background

2.1 The Android Permission System

Android uses permission to protect an Android user's privacy, so all Android apps must request permission to alert users about the risks that applications can contain [8]. Permissions that are asked must be approved by the user in order to access to user's sensitive data. Android provides API framework for applications to interact with the Android system, and API is consisted of packages and classes [15]. If the device's API level is 23 or higher, app permissions are notified to users at runtime, and if the device is running under API level 22, app permissions are automatically asked to users at install-time. Therefore, users can choose whether they want to accept or deny the permissions request [1]. Permissions that are reviewed by users are defined as "Dangerous" permissions, according to the Android Developer's protection level. Permissions are categorized into four threat levels [3], [6]:

Normal permissions are default values. They are lower-risk permissions that are automatically granted by the system without needing the user's approval. However, if needed, users can review these permissions before installing them.

Dangerous permissions are higher-risk permissions that are needed to be reviewed by the user. These permissions request access to the user's private data or can control the device. Since these permissions can cause a negative impact on the user, they are not automatically granted by the system which means that they are asked for a review before proceeding. Dangerous permissions potentially have harmful API calls that are related to the user's private information. These permissions, for instance, can read and send user's text messages.

Signature permissions are granted by the system only if the requesting application has the same certificate as the application that declared the permission. Permissions are automatically granted without asking for the user's review when the certificates match.

SignatureOrSystem permission is an old synonym for signature|privileged which was deprecated in API level 23. These permissions are used in some special cases and are granted only to applications in the dedicated folder of the Android system image or to applications signed with the same certificate as those that have declared permissions. Since this protection level is sufficient to most cases, permissions in this protection level can be activated regardless of the exact area of installation.

2.2 Feature Selection

When using machine learning, large data can be redundant or irrelevant. These data can cause overfitting, increase run time and model complexity, and mislead the learning algorithm. In order to run files in an efficient way, reducing unnecessary features can help the program perform well with a higher level of accuracy. In particular, permission-based analysis require feature selection methods with classification algorithms. However, choosing the right feature selection method is a challenge because the result of feature selection can be impacted with not only the characteristic of the dataset but also the interaction with the classification algorithms [17].

The feature selection method is consisted of three types of methods; filter, wrapper, and embedded method. The filter method removes the least important features one by one, the wrapper method finds which feature subset has the best performance in running the algorithm, and the embedded method adds itself a feature selection function [4], [20].

In our paper, to reduce the dimension size of the dataset, we performed a feature selection method with the Weka tool. 16 different algorithms were used to get the best feature set, and with the selected features, we were able to perform permission-based malware application detection.

3 Related Work

Barrera et al. performed an empirical analysis of the permission-based security models by analyzing 1,100 popular Android apps with the SOM algorithm. The study was to find what permissions were used by the Android developers. They found out that out of many permissions, only a small amount of numbers of permissions are used by them. Also, they found that permissions do not exactly matter or correlate with the category of the application [2]. Jesse Obiri-Yeboah et al. discussed how people relate their privacy issues with the Android permission system, and studied about security issues that come along [12]. Ontang et al. [14] propose access control policy infrastructure to prevent applications. They also propose that Android permissions should be in more detail with notifying some specific requirements for configurations or software versions for instance. Felt et al. conducted a usability study on the effectiveness of the Android permissions. The study came up with the result that permissions do not help users to be informed about the malware threat that could impact their security [7]. Aung et al. detected malware applications by extracting permission-based applications at Google's Play market. They extracted the data from AndroidManifest.xml files. In order to calculate the accuracy of their detection method, they compared the performances with true positive, false positive, precision, and recall rates. J48 and Random Forest outperformed CART [21]. Enck et al. paper proposes Kirin security service to check on the permissions when installing applications to reduce the risks of malware applications. The study is relied on developer's perspective of requesting permissions [5].

4 Data

In order to conduct this research, 6414 benign apps and 4404 malware apps were used. These applications were collected from 2017 to 2019 by NSHC and Google. Out of 457 lists of permission that were provided by Android API features, we have figured out what permissions were being used for each application that we have. For those permissions that were being used, we marked them as “1”, and “0” for those that weren’t being used, and saved as a CSV file. In order to perform various algorithms for Feature Selection, we used all of 457 permissions for features.

10,818 applications that we have collected used 182 different kinds of permissions. We have matched all of used permissions with protection levels. 23 dangerous permissions, 54 normal permissions, 49 signature permissions, and 56 signature|privileged permissions were used.

We wanted to check how many protection levels existed in each benign and malware app in total. We counted the actual numbers of permissions by its protection level. Out of 6,414 benign applications, and 4,404 malware applications, 10,699 and 22,283 dangerous permissions were used respectively, and the detailed numbers are in Table 1.

Table 1. Percentage of Dangerous Protection Level Permissions

	Dangerous	Normal	signature	signature|privileged
Benign App (6,414)	10,699	17,286	800	1,391
Malware App (4,404)	22,283	37,460	8,386	8,829

5 Methodology

5.1 Weka Attribute Selection

To perform Feature Selection, we used WEKA’s Attribute Selection, which is great for measuring the utility of attributes and finding the subsets that are predictive of the data [9], [10]. Both wrapper and filter approaches are included in Attribute Selection. For Attribute Evaluator, for example, correlation-based feature selection, chi-square statistic, gain ratio, information gain, symmetric uncertainty, and support vector machine-based criterion is provided. For search methods, the best-first search, ranker method, and random search methods are provided, for example.

In this paper, we have used 16 algorithms for Feature Selection that were provided in Weka’s interface, and 3 search methods. We are to select those that are the main features to detect malware applications. The following Table 2 shows which algorithms and methods were used in Feature Selection.

12 out of 16 algorithms were required to use the ranker method for the search methods in Weka. Ranker methods rank all the features by their individual evaluations. Therefore, they ranked all 457 features, which were not intended, so we made another experiment with python to select the right features for malware detection. We calculated the accuracy of the first ranked feature. Then, the next ranked feature was added to calculate the accuracy, and then the next ranked feature was added, and so on until the program calculates all 457 features. We continued this for 12 algorithms. After all the features were calculated, we figured out the best feature selection for each algorithm by the accuracy that they have reached (see Fig. 1).

Table 2. Algorithms used for Feature Selection in Weka.

Attribute Evaluator	Search Method
CfsSubsetEval	BestFirst
ChiSquaredAttributeEval	Ranker
ClassifierAttributeEval	Ranker
ClassifierSubsetEval	GreedyStepwise
CorrelationAttributeEval	Ranker
FilteredAttributeEval	Ranker
GainRatioAttributeEval	Ranker
InfoGainAttributeEval	Ranker
OneRAttributeEval	Ranker
ReliefAttributeEval	Ranker
ConsistencySubsetEval	GreedyStepwise
SignificanceAttributeEval	Ranker
SVMAttributeEval	Ranker
FilteredSubsetEval	GreedyStepwise
SymmetricalUncertAttributeEval	Ranker
WrapperSubsetEval	GreedyStepwise

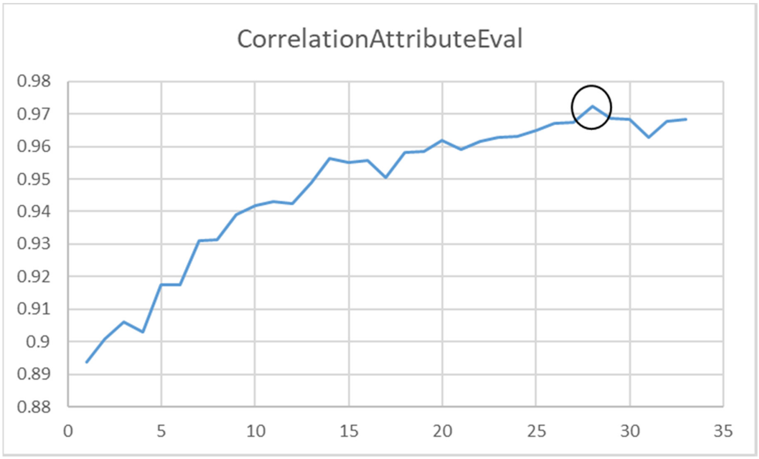

Fig. 1. An example of Ranker Method feature selection. The circle shows the peak of accuracy. We selected all features until this point.

In Fig. 1, we are showing one example of how we performed a feature selection for the ranker method. This figure is about the CorrelationAttributeEval algorithm. The circle that surrounds the peak of the graph is the 28th feature. It was the highest peak, which eventually means that 28 features all together scored the highest accuracy. We used this 28 features as a feature set to perform malware detection.

After the peak, when we continue to add permissions to calculate its accuracy, some do reach up to the point where we first considered to be the highest accuracy. However, we considered all the results after the peak to be overfitting, so they were not reviewed in our study.

5.2 Deep Learning Algorithms

After selecting all the features that are relevant to detect malware applications, we now put it into practice to see the accuracy of detection by using deep learning algorithm. This method was intended to check if selected features are actually being meaningful when detecting malware applications. The detection rate could differ by the dataset and algorithm's characteristics; we chose four different deep learning algorithms to see the detection rate; MLP (Multilayer Perceptron), CNN (Convolutional Neural Networks), LSTM (Long Short-Term Memory), and DBN (Deep Belief Network) [16].

Performance Evaluation Criteria. To evaluate the performance of our features, we used a confusion matrix, which is a summary of prediction results that are used in the field of machine learning. We used recall, precision, f-measure, and accuracy [19].

Accuracy calculates the percentage of correctly identified applications as shown in Eq. (1):

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}} \tag{1}$$

Recall calculates the percentage of correctly identifying malicious applications as malicious as shown in Eq. (2):

$$Recall = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{2}$$

Precision calculates the percentage of actual malicious applications among those predicted to be malicious as shown in Eq. (3):

$$Precision = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{3}$$

F-measure calculates the harmonic average of precision and recall as shown in Eq. (4):

$$F - measure = \frac{2 * \mathrm{Recall} * \mathrm{Precision}}{\mathrm{Recall} + \mathrm{Precision}} \tag{4}$$

6 Results and Evaluation

6.1 Accuracy of Malware Detection

With selected features from Weka's attribution selection, we evaluated its efficiency with deep learning algorithms. According to Table 3, all of the detection rates are above

90% which scored a high percentage of malware detection. This percentage proves that the selected features are meaningful to determine malware applications. Selected features included all protection levels, and 324 permissions were used, with 70 types of permissions. The confusion matrix was used to calculate the accuracy, and we only scored the accuracy part on our Table 3.

Table 3. Malware application detection results

	CNN (%)	MLP (%)	LSTM (%)	DBN (%)
CfsSubsetEval	94.4778	94.5779	94.2814	92.7911
ChiSquaredAttributeEval	95.5869	96.1799	95.7139	94.4778
ClassifierAttributeEval	94.9399	96.5496	95.0439	93.2070
ClassifierSubsetEval	95.1017	95.1325	94.9168	95.0323
CorrelationAttributeEval	96.7421	97.2274	96.8807	94.8013
FilteredAttributeEval	96.3725	97.0425	95.9681	94.7551
GainRatioAttributeEval	94.9630	96.1491	95.2403	93.7384
InfoGainAttributeEval	96.1414	96.4571	96.1183	94.5702
OneRAttributeEval	96.0259	96.1183	95.9681	93.6922
ReliefAttributeEval	97.2505	97.0425	97.3198	95.6099
ConsistencySubsetEval	97.0425	96.9808	96.9039	95.6099
SignificanceAttributeEval	95.1248	96.2723	95.8872	94.4778
SVMAttributeEval	95.4251	95.0708	95.3905	94.1312
FilteredSubsetEval	94.4547	94.3931	94.0850	92.4445
SymmetricalUncertAttributeEval	97.0656	97.1965	96.6266	94.9399
WrapperSubsetEval	96.7190	97.2581	96.8115	95.5175

From the process of selecting features that are important to detect malware applications, MLP's accuracy was all above 94% which performed better than the other three methods. The highest detection rate was conducted with a combination of ReliefAttributeEval and LSTM methods.

Out of selected permissions, that were consisted of 370 permissions with 70 different types, we counted each protection level to see which one was more used to detect malware applications. Table 4 shows exactly counted numbers of protection levels, and it appears that the normal protection level is more used than a dangerous level. This is based on permissions.

Out of 70 types of permissions, 19 were dangerous, 22 were normal, 10 were signature, and 19 were signature|privileged (Table 5). We divided 70 types of permissions into four protection levels to use for malware application detection.

Four different deep learning algorithms were used for the detection, and the confusion matrix was used to calculate the accuracy. Table 6 shows the accuracy.

Table 4. Total number of Protection Levels in Selected Features via Weka

	Dangerous	Normal	Signature	signature|privileged
# of Protection Levels	119	147	54	50

Table 5. Protection Level counted for 70 types of permissions in Feature Selection

	Dangerous	Normal	Signature	signature|privileged
# of Protection Levels	19	22	10	19

Table 6. Malware application detection with protection level

	CNN (%)	MLP (%)	LSTM (%)	DBN (%)
Dangerous	92.4676	92.2982	92.3406	89.2329
Normal	94.0619	94.3007	94.1774	90.5961
Signature	89.2791	88.3240	88.5166	88.3318
signature|privileged	91.1275	91.5280	91.5434	90.1571

When we detected malware applications with protection levels, CNN's overall performance was good, but the highest accuracy was detected with LSTM, 94.1774%, with Normal protection level. This would let us assume that permissions with normal protection are important factors to consider when detecting malware applications.

6.2 Classification

With the accuracy results from what we performed with various feature selection algorithms and deep learning methods, it appeared that Relief Attribute Evaluator and LSTM method detects higher accuracy. Relief Attribute Evaluator repeatedly samples the instance and evaluates the value of the attribute by taking the value into account of the given attribute for the nearest instance of the same and different classes. It can perform on both discrete and continuous class data [13].

Out of 70 types of permissions that were selected with feature selection, we wanted to investigate which are more important to consider when detecting malware applications, and figure out what kinds of protection levels exist. Therefore, we chose to follow the steps we made in 5.1, and used Relief Attribute Evaluator and LSTM for detection.

Relief Attribute Evaluator uses Ranker for the search method, and Fig. 2 shows the peak of accuracy; 94.66%. 11 permissions as a set scored the highest accuracy when we ran it through LSTM. 11 permissions are listed in Table 7. We also counted how many applications have used them, and counted them in two ways; malware applications, and both malware and benign applications. It appears that selected permissions are way more used in malware applications.

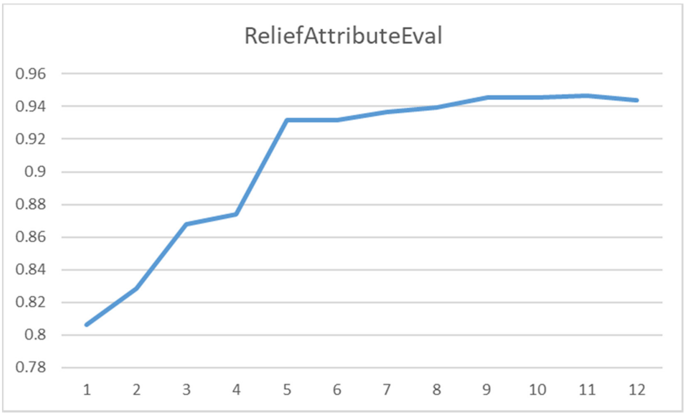

Fig. 2. ReliefAttributeEval's accuracy. 11 permissions together scored highest.

Table 7. Relief Attribute Evaluator Feature Selection results

ReliefAttributeEval	Protection Level	Malware/Total
android.permission.BIND_DEVICE_ADMIN	signature	2012/2032
android.permission.GET_TASKS	normal	2638/2965
android.permission.VIBRATE	normal	1780/3420
android.permission.WRITE_SETTINGS	signature	2343/2564
android.permission.SEND_SMS	dangerous	3668/4055
android.permission.RECEIVE_SMS	dangerous	3436/3729
android.permission.READ_SMS	dangerous	2832/2960
android.permission.WRITE_SMS	normal	2308/2384
android.permission.WRITE_EXTERNAL_STORAGE	dangerous	3793/6025
android.permission.READ_PHONE_STATE	normal	4077/6073
android.permission.RECEIVE_BOOT_COMPLETED	normal	3572/4284

Permissions that were selected again with Relief Attribute Evaluator algorithm consists of 4 dangerous, 5 normal, and 2 signature levels. These selected permissions represent important factors in determining malware applications. Once again here, the normal protection level was the most. While dangerous permissions are to ask users to review their usage, other levels in these selected features are not notified to a user unless users intend to look up for them. Permissions in Table 7 are all related to user's privacy and security, and many permissions were related to SMS and the device. Not only dangerous level permissions are important factors that users should be noticed and be aware of, but also all other permission levels. Therefore, we will name lastly selected permissions to be "Risky", and other remaining ones rather "Safe".

The result of choosing the right feature selection method with the right deep learning study varies with the dataset. The suggested study set that we propose is not definitely the right answer, but we propose a way to select risky features with a way to follow along (see Fig. 3).

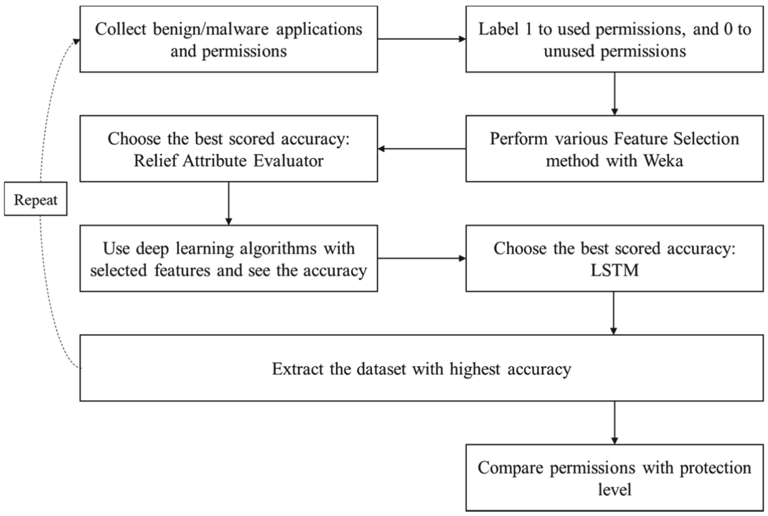
Collect benign/malware applications and permissions
Label 1 to used permissions, and 0 to unused permissions
Perform various Feature Selection method with Weka
Choose the best scored accuracy: Relief Attribute Evaluator
Repeat
Use deep learning algorithms with selected features and see the accuracy
Choose the best scored accuracy: LSTM
Extract the dataset with highest accuracy
Compare permissions with protection level

Fig. 3. How to choose risky features when detecting malware applications.

7 Conclusion

As Android has defined each permission with its permission levels, some are required to be reviewed by the user, and some are not. Dangerous permissions are correlated with user's privacy issues, which we would all consider it being a really dangerous factor when it comes up to detecting malware applications. In our research, we wanted to find out what features are being considered to be the main elements to detect malware applications, and proposed a way to follow. To detect malware applications, feature selection and a way to evaluate it needs to be done. With our data, feature selection was well done with the Relief Attribute Evaluator method, and malware detection was done well with LSTM. However, we emphasize that this combination varies with the character of the dataset. Selected features were closely related to the user's privacy and device, but most of the permissions were not required to be reviewed by the user. Four levels of protection level are classified by the Android Google Homepage, but they are divided into developer-friendly ways. Users must understand that other levels can also be dangerous, and change their view of understanding the protection level. Therefore, in our perspective, we view selected features as the same level, and define it "Risky".

Our study was to investigate important features and see how these feature's protection level was organized by the developer's view. How we defined some permissions as

"Risky" is just a start of defining them in the user's view. In future work, some user-friendly views should be made, and we would like to categorize the views in more detail. Usability studies and empirical analysis should be conducted to fully understand the user's view. Dividing up permissions with the user's view would be another challenge coming up.

Acknowledgements. "This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the ITRC (Information Technology Research Center) support program (IITP-2020-2015-0-00403) supervised by the IITP (Institute for Information &communications Technology Planning &Evaluation)"

"This study was supported by a grant of the Korean Heath Technology R&D Project, Ministry of Health and Welfare, Republic of Korea. (HI19C0866)"

References

1. Android Developers Homepage. https://developer.android.com/guide/topics/permissions/overview. Accessed 28 May 2020
2. Barrera, D., Kayacik, H.G., van Oorschot, P.C., Somayaji, A.: A methodology for empirical analysis of permission-based security models and its application to android. In: Proceedings of the 17th ACM Conference on Computer and Communications Security, CCS 2010, pp. 73–84 (2010)
3. Chin, E., Felt, A. P., Greenwood, K., Wagner, D.: Analyzing inter-application communication in Android. In Proceedings of the 9th International Conference on Mobile systems, Applications, and Services, pp. 239–252 (2011)
4. DAS, Sanmay.: Filters, wrappers and a boosting-based hybrid for feature selection. In: ICML, pp. 74–81 (2001)
5. Enck, W., Ongtang, M., McDaniel, P.: On lightweight mobile phone application certification. In Proceedings of the 16th ACM Conference on Computer and Communications Security, pp. 235–245 (2009)
6. Enck, W., Ongtang, M., McDaniel, P.: Understanding android security. IEEE Secur. Priv. **7**(1), 50–57 (2009)
7. Felt, A. P., Finifter, M., Chin, E., Hanna, S., & Wagner, D.: A survey of mobile malware in the wild. In Proceedings of the 1st ACM Workshop on Security and Privacy in Smartphones and Mobile Devices, pp. 3–14 (2011)
8. Felt, A. P., Ha, E., Egelman, S., Haney, A., Chin, E., Wagner, D.: Android permissions: user attention, comprehension, and behavior. In Proceedings of the Eighth Symposium on Usable Privacy and Security, pp. 1–14 (2012)
9. Frank, E., et al.: WEKA-a machine learning workbench for data mining. In: Maimon, O., Rokach, L. (eds.) Data Mining And Knowledge Discovery Handbook, pp. 1269–1277. Springer, Boston, MA (2009)
10. Hall, M., et al.: The WEKA data mining software: an update. ACM SIGKDD Explorations Newsletter 11.1 (2009)
11. IDC Homepage. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. Accessed 02 April 2020
12. Obiri-Yeboah, J., Man, Q.: Data security of android applications. In: 2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). IEEE (2016)

13. Kononenko, I.: Estimating attributes: analysis and extensions of RELIEF. In: Bergadano, F., De Raedt, L. (eds.) ECML 1994. LNCS, vol. 784, pp. 171–182. Springer, Heidelberg (1994). https://doi.org/10.1007/3-540-57868-4_57
14. Ongtang, M., McLaughlin, S.E., Enck, W., McDaniel, P.D.: Semantically rich application-centric security in android. In: ACSAC, IEEE Computer Society, pp. 340–349 (2009)
15. Peiravian, N., Zhu, X.: Machine learning for android malware detection using permission and API calls. In: 2013 IEEE 25th International Conference on Tools with Artificial Intelligence, pp. 300–305. IEEE (2013)
16. Samira, P., et al.: A survey on deep learning: algorithms, techniques, and applications. ACM Comput. Surv. (CSUR) **51**(5), 1–36 (2018)
17. Shabtai, A., et al.: "Andromaly": a behavioral malware detection framework for android devices. J. Intell. Inf. Syst. **38**(1), 161–190 (2012)
18. Statista Homepage. https://www.statista.com/statistics/266210/number-of-available-applications-in-the-google-play-store/. Accessed 28 May 2020
19. Story, M., Congalton, R.G.: Accuracy assessment: a user's perspective. Photogramm. Eng. Rem. Sens. **52**(3), 397–399 (1986)
20. Lei, Y., Liu, H.: Feature selection for high-dimensional data: a fast correlation-based filter solution. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), pp. 856–863 (2003)
21. Zarni Aung, W.Z.: Permission-based android malware detection. Int. J. Sci. Technol. Res. **2**(3), 228–234 (2013)