Towards the Detection of Malicious Android Applications Through Analysis of Permissions

Alistair Robinson 1703212
Department of Computer Science, University of Warwick

***Abstract*—As the usage of malware by cyber criminals continues to increase year on year, it is highly relevant to consider new, effective methods by which malicious applications can be detected and neutralised before causing damage to users. This report details statistical analysis on a sample of benign and malicious applications, investigating the relationship between the type of application and the permissions it requests. Experimental results suggest that the difference in the distribution of permissions between benign and malicious applications is statistically significant and that some malicious applications share identical permissions signatures. Classification and clustering are found to be accurate methods of distinguishing between benign and malicious applications, with logistic regression, SVM and ensemble methods performing most effective on permissions signatures.**

I. Introduction

Malware is a term used to describe a malicious application, often developed or manufactured by cyber criminals, which is intentionally designed to cause damage to a device or its user. Mobile applications are a common vector for malware to be distributed to unsuspecting users, particularly in the form of Trojans: malicious applications disguised as or contained within benign applications, which are commonly delivered through application marketplaces such as the App Store, Google Play Store or other third-party distributors [1].

In 2019, the Android mobile operating (OS) system was judged by security engineers to be the most insecure OS available on the market, with over 400 individual vulnerabilities identified in its architecture [2]. Despite this, Android remains the most used mobile OS worldwide with 64% market share overall, with high popularity in developing territories such as Asia (72%), South America (84%), Africa (87%) [3].

Malware in Android applications can be used by attackers for a multitude of unethical and illegal practices, including the harvesting of login information from other applications, displaying intrusive advertising on a user's device, performing ransomware attacks on a user's local files or even hijacking a user's device from within for use in a distributed botnet used to perform DDoS attacks [1].

By employing an effective method of detecting malware before it is published to a public marketplace, such as the Google Play Store, users will feel more comfortable installing useful applications to their device without fear of inadvertently installing malware. Similarly, Google will be able to preserve its image as a leader in positive consumer practices through the removal of malware from its marketplace. This report seeks to investigate the efficacy of application permissions as a signature by which malware detection can be performed.

II. Related Work

The detection of malicious applications is a non-trivial task which forms an active research area in the field of digital forensics and cyber security. Malware detection as a task is typically categorised into three general approaches [4]:

- Anomaly-based detection
- Specification-based detection
- Signature-based detection

Anomaly and specification-based approaches rely on measuring the runtime behaviour of an application and identifying malicious activity. These approaches would be inefficient, time-consuming and have a high False Positive Rate (FPR) if used to police a public application marketplace, owing to the quantity and variety of applications available on large platforms. Instead, it may be more relevant to consider the use of a signature-based detection methodology, which uses the permissions requested by an application as a preliminary indicator of whether its behaviour will be malicious or benign.

Android applications are used to restrict the privileges of applications running on the Android OS. If an application makes a system call relating to restricted actions or data, such as sending an SMS message or reading the phone's internal state, without the appropriate permission enabled by the user, the OS will not handle the system call [5].

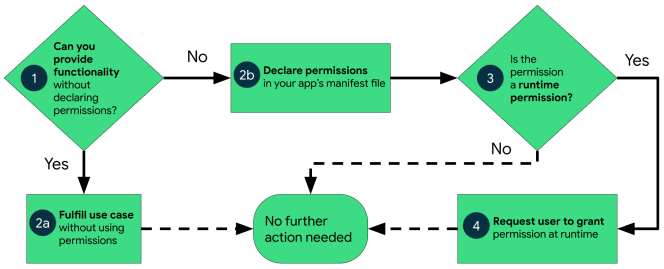

Fig. 1: Permissions workflow for Android applications [5]

Because many malware-based attacks require access to restricted actions or data, malicious applications will need to declare their permissions within their application manifest to perform their attack. As such, it is hypothesised that the permissions requested by malicious applications will be statistically distinguishable (that is, the distribution of permissions will be sufficiently different within a given confidence interval) and that therefore, the permissions requested by an application are a sufficient signature to perform efficient, scalable and accurate malware detection in Android applications.

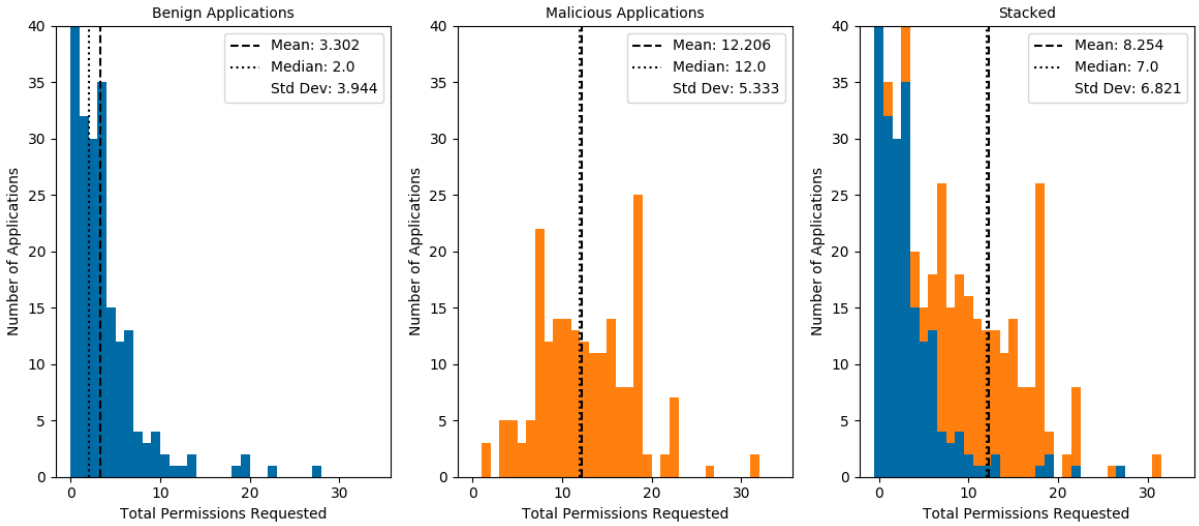

Fig. 2: Frequency distribution of total permissions requested by applications

The usage of application permissions as a means of detecting malware is not a novel concept and has been investigated in scientific literature before [6], [7], [8], [9], however, past experiments have focused efforts on developing high-performance classification systems which can accurately perform malware detection, rather than investigating the underlying statistical relationships between application permissions and safety. This report intends to support existing work in this research area by providing a more statistical analysis into these relationships, thereby demonstrating the robustness of permissions-signature based malware detection approaches.

III. Data Acquisition

To investigate the efficacy and statistical significance of performing malware detection through permissions signatures, a data set was sourced containing samples of the permissions requested by both benign and malicious Android applications. This was sourced from a public repository compiled by an academic security researcher investigating the application of machine learning in the detection of malware [10]. In total, the data set contains 198 samples taken from benign applications and 198 samples taken from malicious applications. Each sample contains a vector p of 330 binary features, each representing whether a single android permission was enabled in the application's manifest, and a binary class which denotes whether the application was benign or malicious. It is possible for two applications in the dataset to have an identical signature if they request identical permissions. This data was fully complete containing no missing values and therefore required minimal preparation and cleaning. It was analysed using statistical analysis and data science packages Numpy, Scipy and SKLearn in a Python-based Jupyter Notebook[1].

The decision to use a pre-existing data set from a secondary source was motivated by the time constraints of the investigation; an area of future work may include the replication of this report's findings on a larger, more recent data set.

[1]Available at github.com/AlistairRobinson/MalwareDetection

IV. Preliminary Analysis

Before considering the relationships between individual application permissions and the application's classification, it is simpler and more intuitive to first consider the total number of permissions $|p|$ requested by an application. By computing the total number of permissions requested by each sample in the dataset, separating samples based on their behaviour type and plotting these results in a histogram chart, we find Fig. 2.

There are a number of immediate observations that can be drawn from this visualisation. On average, benign applications request significantly fewer system permissions than malicious applications, with a median sum of 2 and a mean sum of 3.302. By contrast, malicious applications generally requested more permissions with a slightly larger spread, with a median sum of 12, a mean sum of 12.206 and a standard deviation of 5.333 compared to 3.994 for benign applications.

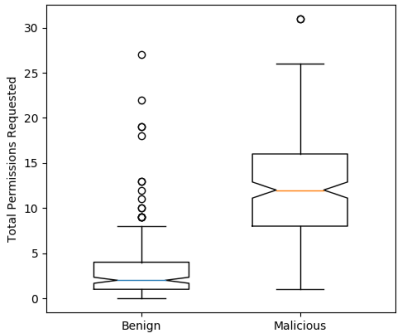

Fig. 3: Box-plot of total permissions requested by applications

Importantly, the shapes of the two frequency distributions are also different. For benign applications, the median (2) and mean (3.302) indicate a clear positive skew towards 0, resulting in an asymmetrical distribution. By clear contrast, the frequency distribution for malicious applications had similar values for the median (12) and mean (12.2), indicating a relatively symmetrical distribution with little skew. This can also be observed through a box plot of the same data, Fig. 3.

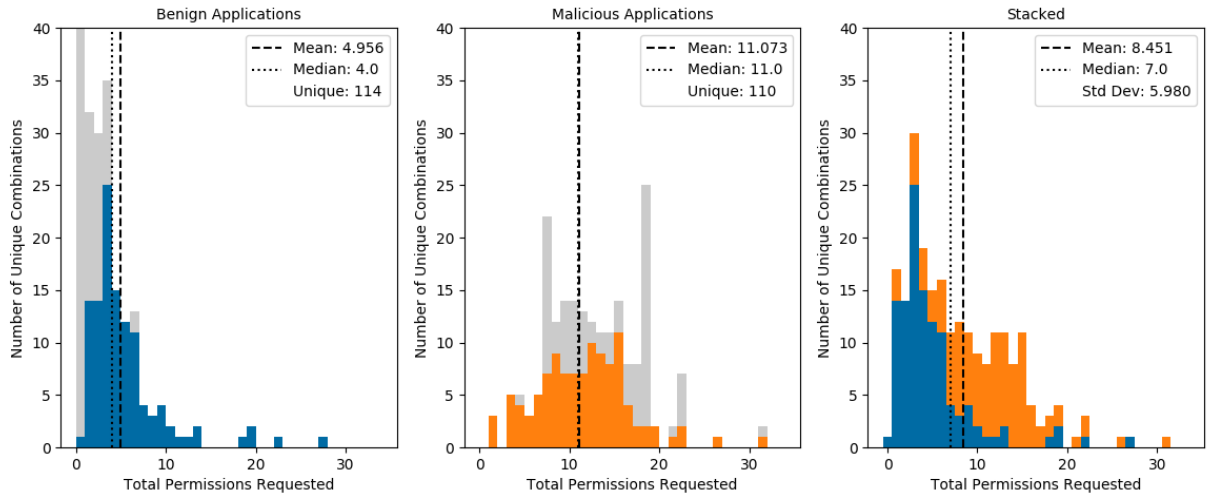

Fig. 4: Frequency distribution of unique permission combinations (shaded sections represent data from Fig. 2)

Initial analysis of the frequency distribution of the data highlighted anomalies in the distribution of the total number of permissions requested by malicious applications at values of 7 and 18, which are visualised as unusual peaks in the frequency distribution plot (Fig. 2). To investigate the cause of these anomalous peaks, the same analysis was performed with duplicate samples removed from both the set of benign and malicious applications (Fig. 4). This expresses the number of unique combinations of permissions in the data set.

For benign applications, the differences between Fig. 2 and Fig. 4 are within expected boundaries. When the total number of permissions requested by an application is low, there are fewer unique combinations which can represent these samples. For example, 40 benign applications in the dataset did not request any permissions, however, since there is only one possible combination of permissions which has a total of 0, all samples with $|p| = 0$ share a permissions signature. As such, we do not draw significant conclusions from this data.

However, for malicious applications, it is clear that the peaks in Fig. 2 are no longer present in Fig. 4, and that the distribution in Fig. 4 has fewer outliers overall than Fig. 2. The largest difference between the distributions occurs at $|p| = 18$, where a total of 25 samples requested exactly 18 permissions, but out of these samples, only two unique signatures were identified. On closer inspection, it was found that 23 of the samples shared identical permissions signatures, displayed in Fig. 5. Based on their permissions, these applications would be able to perform several attacks against a user, for example:

1) **Escalation of privilege**: By accessing the internet, the malicious application may be able to install other, more dangerous application files to a user's device.
2) **Unauthorised service usage**: By sending SMS messages and calling premium rate phone numbers controlled by an attacker, the user could be forced to pay the attacker unauthorised charges for network usage [11].
3) **Side-channel attacks**: By disabling the user's lock screen (the keyguard), an attacker could physically steal the user's phone and use it without authentication.

```
'android.permission.ACCESS_NETWORK_STATE'
'android.permission.ACCESS_WIFI_STATE'
'android.permission.CALL_PHONE'
'android.permission.DISABLE_KEYGUARD'
'android.permission.INTERNET'
'android.permission.READ_CONTACTS'
'android.permission.READ_LOGS'
'android.permission.READ_PHONE_STATE'
'android.permission.READ_SMS'
'android.permission.RECEIVE_BOOT_COMPLETED'
'android.permission.RECEIVE_SMS'
'android.permission.RESTART_PACKAGES'
'android.permission.SEND_SMS' 'android.permission.VIBRATE'
'android.permission.WAKE_LOCK'
'android.permission.WRITE_APN_SETTINGS'
'android.permission.WRITE_CONTACTS'
'android.permission.WRITE_SMS'
```

Fig. 5: A set of permissions used by 23 malicious applications

Without additional information on the 23 applications which shared identical permissions, it is impossible to know for certain why this anomaly occurred and what attack the applications sought to perform against users. However, it is very unlikely that these 23 applications requested the same permissions independently. A possible explanation for this observation mass-manufactured malware, whereby in order to maximise the number of victims for a malware attack, an attacker will repackage the same Trojan malware programs into different shell programs. The resulting applications may appear different on the surface, and are therefore more likely to draw in a wider range of victims; however, the actual attack performed by the applications are the same [1]. It is also possible, though unlikely, that this anomaly arose as a result of an error in data collection; if researchers counted the same applications multiple times during data collection, this could account for the anomalies in malicious application signatures.

The final area of preliminary analysis on the data set is a regression analysis, whereby the Pearson correlation coefficient is calculated between each permission and the class of each application. The motivation for this analysis is to identify whether any individual permissions were highly correlated with, and therefore potentially indicative of, an application's class. These results are displayed in Table I; a positive correlation indicates a positive relationship between the permission and malicious applications, and vice versa.

Pearson r	Permission
0.837	'android.permission.READ_PHONE_STATE'
0.645	'android.permission.READ_SMS'
0.587	'android.permission.WRITE_SMS'
0.541	'android.permission.ACCESS_WIFI_STATE'
0.533	'android.permission.ACCESS_NETWORK_STATE'
...	...
-0.112	'android.permission.CAMERA'
-0.118	'android.permission.BLUETOOTH'
-0.134	'android.permission.NFC'
-0.134	'android.permission.USE_CREDENTIALS'
-0.134	'android.permission.AUTHENTICATE_ACCOUNTS'

TABLE I: Correlation between permissions and class

The most highly correlated permissions include the ability to read sensitive information about a user's device, read a user's SMS messages, send SMS messages and access information on the device's network connectivity [5]. From this, we may infer that many malicious applications execute attacks which involve information gathering or SMS-based fraud [11].

The least correlated permissions include the ability to activate the device's camera without alerting the user, connect to other devices via Bluetooth or NFC and manage the accounts on the device. The negative correlation between camera permissions and malicious behaviour was unexpected, particularly due to the recent increase in cyber-crime activities utilising device cameras to blackmail victims [12]. However, this result does not mean that no malicious applications perform attacks using a device's camera; it expresses that, within the provided dataset, more benign applications requested camera permissions than malicious applications. Similarly, the negative correlation for account permissions does not indicate that no malware targets the account manager; it may instead indicate that most malicious applications perform rudimentary attacks which are less complex but target a wide range of users.

From this analysis, we may conclude from our data that most malicious applications perform simple attacks such as information gathering on a user's device or SMS-based fraud, both of which can be used for immediate financial gain or to facilitate other attacks, such as phishing.

V. Hypothesis Testing

To investigate the significance of the trends identified through preliminary analysis, hypothesis testing is used to provide a statistical justification of each claim. Two tests are performed: the first investigates whether the increase in mean permissions requested between benign (3.302) and malicious (12.206) applications is statistically significant, the second investigates whether the observation of 23 malicious samples with identical permissions signatures is statistically significant.

Because the first hypothesis test requires us to compare a statistic over two samples of data, a one-tailed Welch t-test was used to evaluate the difference between the mean of both data sets [13]. This test was selected over a standard Student's t-test because the latter is not considered robust when samples have different variances (Fig. 2) and because the samples sizes are considered sufficiently large that a normal distribution can be approximated for the purposes of this test.

$$H_0 : \mu_{\text{Benign}} \geq \mu_{\text{Malicious}}$$
$$H_1 : \mu_{\text{Benign}} < \mu_{\text{Malicious}}$$
$$\alpha : 0.01$$
$$p : 3.42 \times 10^{-56}$$
$$p < \alpha \implies \text{Reject } H_0$$

The p value shown above was found using the `scipy` implementation of the Welch t-test, which calculates a p value based on two input data sets. Because the p value of our samples is below our significance level α, we may conclude that there is sufficient evidence to reject our null hypothesis, suggesting that the mean number of permissions for malicious applications is greater than the mean for benign applications. Therefore, our observation of a larger mean in malicious applications compared to benign applications in Fig. 2 can be considered statistically significant.

The second test considers the expected number of times the identical signature identified in Section IV would appear. To evaluate this test statistic, we model the probability of a single permission being requested as a Bernoulli distribution with probability 0.165, the average probability of a permission being enabled across all malicious applications excluding the permissions which were never requested anywhere in the dataset (leaving a total of 93 permissions)[2]. From this, we model the probability of exactly replicating the repeated signature over 93 Bernoulli trials: $r = 0.165^{18} \cdot (1-0.165)^{93-18}$. We then consider the number of times n the identical signature will appear in a data set of size 198: $n \sim \text{B}(198, r)$.

$$H_0 : r \leq 0.165^{18} \cdot (1 - 0.165)^{93-18}$$
$$H_1 : r > 0.165^{18} \cdot (1 - 0.165)^{93-18}$$
$$\alpha : 0.01$$
$$\text{Critical Region} : n > \text{PPF}(\text{B}(198, r), 1 - \alpha) = n > 0$$
$$\text{Observed Value} : 23$$
$$23 > 0 \implies \text{Reject } H_0$$

Our test is single-tailed and uses a discrete probability distribution, as such there is a single critical value at the upper end of the distribution of n. The `PPF` function is used to find the discrete value at which the CDF of the distribution reaches a critical value. In this case, the probability r under the null hypothesis is sufficiently low that any value of n strictly greater than 0 is within the critical region. The observed value of n, 23 occurrences of the signature, clearly lies within the critical region. Therefore, there is sufficient evidence to reject the null hypothesis. This result suggests that r is larger than the value given in H_0, or that one or more of our assumptions in the test are incorrect (such as the assumption that all permissions are selected independently from the same distribution; we have already seen from Table I that this is not the case). Regardless, this suggests that the observation of 23 identical signatures can be considered statistically significant.

[2]This only provides us with a rough approximation for how permissions are distributed in the data, however, this generalisation is necessary for us to model the probability of achieving the desired signature under H_0

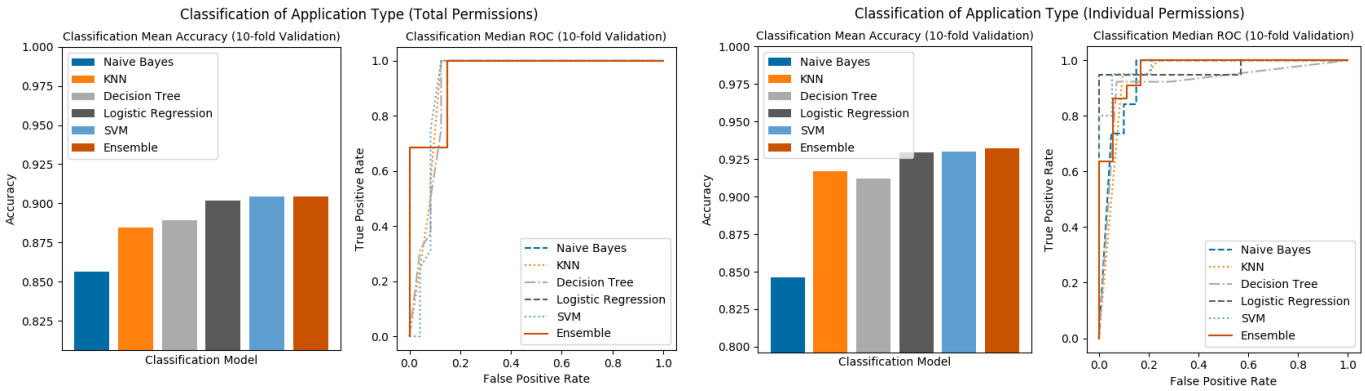

Fig. 6: Classification mean accuracy and median ROC on total (left) and individual (right) permissions

All hypothesis tests assume the independent and identical distribution of data in the malicious and benign samples, as well as conditional independence between permissions. Calculations are available in the project repository[1].

VI. Classification

It is also important to demonstrate the application of identified statistical relationships in permissions data to classify applications as malicious or benign. By performing classification on an application's manifest, users and application marketplace operators would be able to accurately, efficiently and scalably perform malware detection on untrusted applications.

Two classification experiments were performed: in the first, models are given a single feature, the total number of permissions an application requested, and are trained to predict the application's class; in the second experiment, models are instead given a large binary vector of features representing the individual permissions requested by an application.

10-fold validation was used on each experiment to provide a suitable number of experimental repeats, to minimise the risk of overfitting models and to isolate test data for validation purposes. In addition to classification algorithms, an ensemble classifier was also developed. This ensemble method is designed to take a weighted vote of predictions made by other models based on their validation accuracy.

To evaluate each classifier, accuracy and the Area Under Curve (AUC) of the Receiver Operating Characteristic (ROC) curve were calculated. Accuracy provides an estimate of accuracy on unseen test examples based on validation accuracy and the AUC describes how clearly a classifier can distinguish between classes. AUC is relevant in the context of this problem case as we may have a preference between minimising the True Positive Rate and False Positive Rate of the classifier. Because malware carries a high risk, it may be preferable to accept a higher number of false-positive classifications in exchange for a lower number of false-negative classifications. This trade-off is displayed in ROC visualisations in Fig. 6. AUC and accuracy results are shown in Table II.

The performance of models on total permissions was strong, though the accuracy and AUC of all classifiers clearly increased when models were given access to individual permissions with the exception of the Naive Bayes classifier.

Model	Total		Individual	
	ACC	AUC	ACC	AUC
Naive Bayes	0.856	0.942	0.846	0.932
KNN (K = 5)	0.884	0.921	0.917	0.951
Decision Tree	0.889	0.927	0.912	0.910
Logistic Regression	0.902	0.942	0.930	0.974
SVM	0.904	0.924	0.930	0.977
Ensemble	0.904	0.942	0.932	0.972

TABLE II: Mean accuracy (ACC) and AUC of classifiers

This may have arisen because individual permissions violate the conditional independence assumption made by the Naive Bayes classifier; an assumption which is satisfied when trained on a single feature representing total permissions.

Although training on individual permissions allows models to identify underlying trends between specific permissions and application behaviour (for example, the relationships and correlations identified in Table I) and increase prediction accuracy as a result, this comes at the cost of increased time and memory performance overheads. Particularly for models such as K Nearest Neighbours (KNN), which suffer greatly from the curse of dimensionality [14], there is a large performance overhead between training and predicting using individual permissions ($O(|p|)$ dimensions) and the total number of permissions ($O(1)$ dimensions).

Logistic regression and Support Vector Machine (SVM) approaches were consistently performant across the two experiments performed. Both methods were similarly accurate in classification. Logistic regression distinguished between behaviour types better when trained on total permissions; conversely, the SVM approach's distinguishability outperformed logistic regression when trained on individual permissions.

The weighted ensemble method also proved highly performant across both experiments. The ensemble approach achieved the highest accuracy for both experiments but was less discriminatory between behaviour types than logistic regression and SVM approaches on individual permissions. This suggests that considering the output of multiple different methods in this classification problem is beneficial, because as the methods independently identify different trends within the data, the overall performance of the ensemble increases.

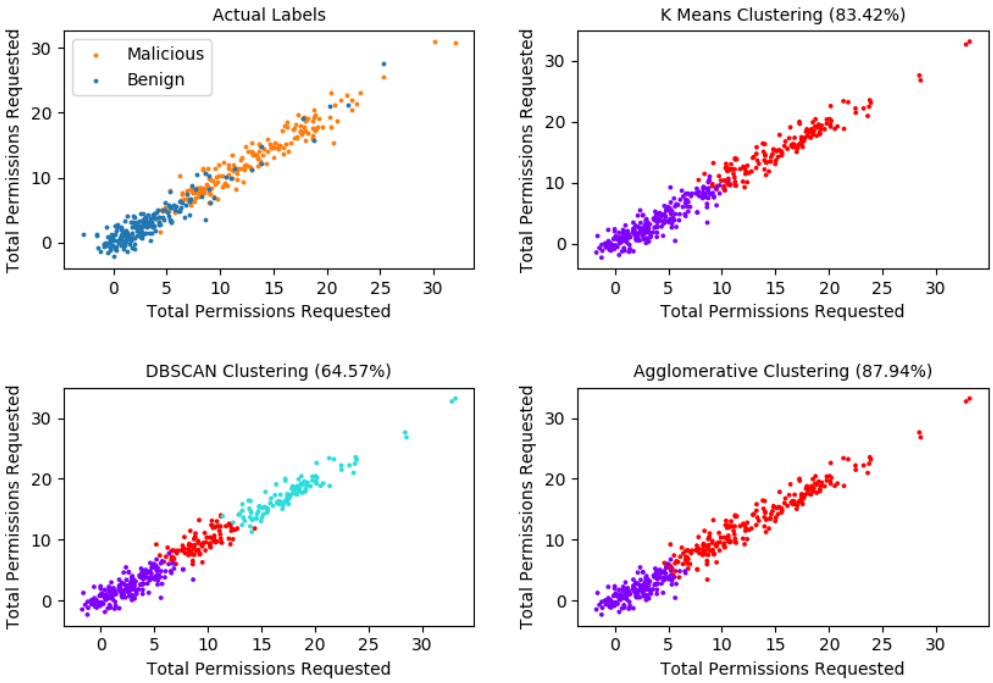

Fig. 7: Clustering results and accuracy based on the total number of permissions requested by applications. Points are distorted with a small amount of Gaussian noise $\sim N(0, 1)$ to improve the graphical visualisation of clusters

VII. Clustering

Clustering algorithms were also evaluated as approaches by which the behaviour of applications may be grouped or distinguished based on their permissions.

Two experiments were performed: in the first, examples were clustered based on total permissions; in the second, examples were clustered based on individual permissions. Clustering algorithms were evaluated based on the accuracy of clusters compared to the actual labels of examples in the data, providing an estimation of accuracy if the clusters were used to classify unseen test examples. Two clustering algorithms, K means and agglomerative clustering, were given the number of actual clusters in the data as a parameter; the parameters of the DBSCAN algorithm were instead optimised by repeating the algorithm on the data multiple times and selecting the values of ϵ and min which maximised accuracy. The results of both experiments are shown in Fig. 7 and Table III.

Data points plotted in Fig. 7 are distorted with a small amount Gaussian noise $\sim N(0, 1)$; without distortion, data points would appear in a perfectly straight line, making it difficult to visually estimate density in areas of the graph. This distortion was not applied to data used by clustering algorithms and only affects the visualisations in Fig. 7.

Algorithm	**Total Accuracy**	**Individual Accuracy**
K Means ($N = 2$)	0.8342	0.9070
DBSCAN ($\epsilon = 1$, min $= 45$)	0.6457	0.5628
Agglomerative ($N = 2$)	0.8794	0.9020

TABLE III: Accuracy of clustering algorithms

From visual inspection, it is clear that the K means (top right) and agglomerative (bottom right) clustering algorithms both identified groups of examples similar to the actual clusters present in the data (top left) when considering total permissions. However, despite having its ϵ and min parameters tuned to maximise accuracy, the groups identified by DBSCAN (bottom left) were not similar to the actual clusters of the data. Malicious (red) and benign (purple) clusters were smaller than expected and a large number of malicious examples were incorrectly discounted as noise (cyan). Across both experiments, the overall accuracy of DBSCAN was similar to a naive or random classifier on the dataset (50%). DBSCAN is designed to be performant in cases where the number of clusters is unknown (in this problem, we know there are exactly two clusters) and in cases where data from different clusters is clearly separable with little overlap (from the top left, there are examples which overlap between the blue and orange clusters in the data). Conversely, K means and agglomerative clustering approaches are performant because they are given the number of clusters in the data ($N = 2$).

The accuracy of K means and agglomerative clustering were higher when grouping examples based on individual permissions signatures, however, because this data is high dimensional, it is difficult to visually represent these clusters as shown in Fig. 7. When performed on high dimensional data, both algorithms suffer from the curse of dimensionality discussed in Section VI. Further, in both experiments, clustering algorithms were outperformed on prediction accuracy by classifiers, including those using distance-based approaches such as KNN. As such, classifiers may be more considered appropriate for malware detection than clustering algorithms.

VIII. Analysis

Based on experimental results from Section VI and Section VII, we may conclude that it is possible to classify malicious applications with high accuracy and distinguishability using both complete permissions signatures or the total number of permissions requested. In particular, SVM, logistic regression and ensemble methods are identified as highly performant methods of performing malware classification and detection. Clustering algorithms such as K means clustering and agglomerative clustering are also fairly effective methods of grouping and separating applications based on their permissions, however, they are consistently outperformed both in terms of accuracy and efficiency by accurate classification models which do not suffer from the curse of dimensionality.

Within the context of malware and security, a benign application being incorrectly classified as malware has a lower impact than a malicious application being incorrectly classified as benign. As such, it may be more relevant to use the classification method which maximises the Positive Predictive Value (PPV) when the Negative Predictive Value (NPV) is 1, or equivalently, when the False Omission Rate (FOR) of predictions is 0; this ensures that, for our data, the model never incorrectly classifies a malicious application as benign. From the ROC curves in Fig. 6, this classifier is shown to be either logistic regression or SVM for total permissions or either the ensemble or Naive Bayes classifier for individual permissions.

The code used to perform the experiments discussed in both sections is available in the project's repository[1].

IX. Limitations & Future Work

Because the data set used by the report [10] only contains samples taken from 398 applications in total, the permissions data within the data set may not be fully representative of the wider population of benign and malicious Android applications. Given that the data set was collected in 2016, it is also possible that the findings of the data are no longer fully relevant in modern applications.

It is possible, though unlikely, that because the samples within the data set were collected by a third-party, the data collection methodology used to source the data could have been incorrect or otherwise violate the assumptions made in Section V, such as the Independent and Identically Distributed (IID) assumption needed to model properties of the data set to probability distributions and to fit classifiers to training data.

The findings of this report would be supported by repeating experiments on a larger, more recent data set to verify experimental findings and statistical conclusions. Furthermore, by collecting a new data set, it would be possible to use additional contextual information alongside permissions signatures to detect malicious applications. For example, an application marketed as a mobile game may be considered suspicious if it requests access to SMS permissions. Evaluating the use of modern machine learning approaches to malware detection, such as neural networks or kernel trick, could improve accuracy further and alleviate issues arising from the high dimensionality of permissions data.

X. Conclusion

This report has evaluated the efficacy of identifying malicious behaviour in mobile applications using permissions data as a method of signature-based malware detection.

Analysis on a data set of applications suggests that, on average, malicious applications request more permissions than benign applications, and that many malicious applications share identical permissions signatures; a potential byproduct of mass-produced malware. Some permissions are found to be highly correlated with malicious applications. The statistical significance of observations was supported by performing hypothesis tests with reasonable assumptions on the data.

Classification and clustering methods were evaluated as methods of identifying malicious applications using the total number of permissions requested and the permissions signature of the application. Experimental analysis suggests that classification methods were more accurate and distinguishing between classes, with logistic regression, SVM and ensemble approaches proving most effective on permissions signatures.

From these analyses, it is concluded that applications permissions are an accurate and scalable method of performing signature-based malware detection on untrusted mobile applications. To support these findings, experiments should be repeated on a larger, more recent data set of applications.

References

[1] AV-TEST, *Security Report 2019/2020*. AV-TEST, 2020. [Online]. Available: www.av-test.org/fileadmin/pdf/security_report/AV-TEST_Security_Report_2019-2020.pdf

[2] N. I. for Standards and Technology, *National Vulnerability Database*. US Department of Commerce, 2020. [Online]. Available: nvd.nist.gov/

[3] ScientaMobile, *Mobile Overview Report January - March 2020*. ScientaMobile, 2020. [Online]. Available: www.scientiamobile.com/wp-content/uploads/2020/05/MOVR-2020-Q1_Final.pdf

[4] N. Idika and A. P. Mathur, "A survey of malware detection techniques," *Purdue University*, vol. 48, pp. 2007–2, 2007.

[5] A. Developers, "Permissions on android — android developers," 2020. [Online]. Available: developer.android.com/guide/topics/permissions/overview

[6] C. Urcuqui López and A. Navarro, "Framework for malware analysis in android," *Sistemas & Telemática; Vol 14, No 37 (2016)DO - 10.18046/syt.v14i37.2241*, vol. 14, pp. 45–56, 08 2016.

[7] S. Liang and X. Du, "Permission-combination-based scheme for android mobile malware detection," in *2014 IEEE international conference on communications (ICC)*. IEEE, 2014, pp. 2301–2306.

[8] B. Sanz, I. Santos, C. Laorden, X. Ugarte-Pedrero, J. Nieves, P. G. Bringas, and G. Álvarez Marañón, "Mama: manifest analysis for malware detection in android," *Cybernetics and Systems*, vol. 44, no. 6-7, pp. 469–488, 2013.

[9] W. Z. Zarni Aung, "Permission-based android malware detection," *International Journal of Scientific & Technology Research*, vol. 2, no. 3, pp. 228–234, 2013.

[10] C. Urcuqui López, "Dataset malware/beningn permissions android," 2016. [Online]. Available: www.kaggle.com/xwolf12/datasetandroidpermissions

[11] M. Labs, "Sms trojan," 2020. [Online]. Available: blog.malwarebytes.com/threats/sms-trojan/

[12] N. C. Agency, "Sextortion (webcam blackmail)," 2020. [Online]. Available: www.nationalcrimeagency.gov.uk/what-we-do/crime-threats/kidnap-and-extortion/sextortion-webcam-blackmail

[13] B. L. Welch, "The generalization of student's' problem when several different population variances are involved," *Biometrika*, vol. 34, no. 1/2, pp. 28–35, 1947.

[14] M. Radovanovic, A. Nanopoulos, and M. Ivanovic, "Hubs in space: Popular nearest neighbors in high-dimensional data," *Journal of Machine Learning Research*, vol. 11, pp. 2487–2531, 09 2010.