In-depth Feature Selection and Ranking for Automated Detection of Mobile Malware

Alejandro Guerra-Manzanares, Sven Nõmm and Hayretdin Bahsi
Department of Software Science, TalTech University, Tallinn, Estonia

Keywords: Machine Learning, Mobile Malware, Feature Selection.

Abstract: New malware detection techniques are highly needed due to the increasing threat posed by mobile malware. Machine learning techniques have provided promising results in this problem domain. However, feature selection, which is an essential instrument to overcome the curse of dimensionality, presenting higher interpretable results and optimizing the utilization of computational resources, requires more attention in order to induce better learning models for mobile malware detection. In this paper, in order to find out the minimum feature set that provides higher accuracy and analyze the discriminatory powers of different features, we employed feature selection and ranking methods to datasets characterized by system calls and permissions. These features were extracted from malware application samples belonging to two different time-frames (2010-2012 and 2017-2018) and benign applications. We demonstrated that selected feature sets with small sizes, in both feature categories, are able to provide high accuracy results. However, we identified a decline in the discriminatory power of the selected features in both categories when the dataset is induced by the recent malware samples instead of old ones, indicating a concept drift. Although we plan to model the concept drift in our future studies, the feature selection results presented in this study give a valuable insight regarding the change occurred in the best discriminating features during the evolvement of mobile malware over time.

1 INTRODUCTION

Mobile phone users are increasingly facing the risks of malware. McAfee stated that “2018 could be the year of mobile malware” as they detected 16 million infections in the third quarter of 2017 alone, twice the figure in 2016 (McAfee, 2018). This enormous increase was also confirmed by Kaspersky who identified an 80% rise in mobile malware attacks (Unuchek, 2018). In addition to these spikes, malware detection software has been proved to be inefficient in tackling this threat (Fedler et al., 2013).

Traditional detection approaches based on signatures fail to detect unknown malware due to the improved obfuscation or stealth techniques employed by malware creators (Fedler et al., 2013). On the other side, machine learning techniques have been perceived as a promising approach for detecting previously unseen malware samples and many studies have shown that they could provide high detection accuracy (Sahs and Khan, 2012; Yuan et al., 2014; Arp et al., 2014). These studies created learning models using dynamic, static or both (namely hybrid) features extracted from legitimate applications and malware samples. Static features such as permissions, java codes or intent filters, are extracted directly from APK files whereas dynamic features, e.g. system calls or network traffic patterns, are derived from the interaction of programs with OS or network (Feizollah et al., 2015).

Feature selection, eliminating irrelevant or redundant features that do not improve the classification performance, is an essential step of machine learning workflow due to three reasons: (1) Representing the problem domain with high dimensions requires more data for learning (commonly known as the curse of dimensionality) and may disturb the accuracy of the classifier, (2) Models using higher dimensions cannot be easily interpreted by the experts, which may create enormous problems in detecting falsely classified instances or profoundly investigating a cyber incident, (3) Higher dimensional data requires more computational resources for constructing and using the learning model on a mobile device. On the other side, feature selection could be more complicated in problem domains where the behaviour of the subjects may vary in time, i.e., a selected feature set may no longer have its discriminatory power, which may be one of the main concerns in malware detection.

In this study, our primary objectives are to iden-

Guerra-Manzanares, A., Nõmm, S. and Bahsi, H.
In-depth Feature Selection and Ranking for Automated Detection of Mobile Malware.
DOI: 10.5220/0007349602740283
In *Proceedings of the 5th International Conference on Information Systems Security and Privacy (ICISSP 2019)*, pages 274-283
ISBN: 978-989-758-359-9

tify the minimum feature set that provides higher accuracy, compare the discriminatory powers of feature categories and analyze the results of models induced by datasets belonging to different time-frames. For these purposes, we applied a two-step procedure to the dataset that is composed by system calls (i.e., a dynamic behaviour) and permissions (i.e., a static behaviour), extracted from malware samples and legitimate applications. In the first step, we used statistical hypothesis testing methods to identify the feature set that may have a significant contribution to the classification. In the second step, we employed Fisher's Score and Gini Index which enabled to rank the selected features according to their discriminatory power. We in turn induced machine learning models with different combinations of datasets with varying feature sets. As Android is the most used mobile operating system worldwide, we focused on detection of Android malware (Statista, 2018). For this research, we formed two malware datasets. "Old dataset" which consists of randomly selected apps from Drebin malware dataset, collected between 2010 and 2012 (Arp et al., 2014)."New dataset" formed by randomly choosing samples, belonging to years 2017 and 2018, from VirusTotal Academic malware dataset (VirusTotal, 2018). Third one is called "legitimate dataset" which is composed by benign applications. We utilized various combinations of these datasets for inducing learning models.

This study shows that feature selection and ranking process can significantly reduce the number of features required in a classifier that provides high accuracy for the detection of mobile malware. We found that features possessing most discriminatory power in classification may differ as new malware types evolve over time, indicating a concept drift. Results suggest that behaviour of mobile malware in terms of system calls and permissions has become more similar to legitimate apps over time although there are some variations among the extent of this evolvement in both feature categories.

Our main contribution is a detailed analysis and comparison of feature selection and ranking results obtained for two types of feature categories. One of the distinctive properties of the present paper is that, in addition to the optimization of number of predictors, we analyzed the change in selected features that has occurred due to the evolvement of malware over time.

This paper is organized as follows: Section 2 presents a review of related literature. Method employed in the study is described in Section 3. Results of our experiments are presented and discussed in Section 4 whereas Section 5 concludes the study.

2 LITERATURE REVIEW

Feature selection and ranking methods have been used in various machine learning-based malware detection studies. In Yan et al. (2013) discriminatory power of malware features such as hexdump of binaries, disassembly codes, PE header and system calls are measured by three filter methods, i.e., ReliefF, Chi-squared, F-statistics, and two embedded methods, i.e., L1 regularized methods, L1-logreg and L1-SVC. In this study, it is identified that PE header and system calls are very beneficial to discern malware from legitimate software, and that L1 regularized methods with 100 features provided higher detection rates (Yan et al., 2013). In Ahmadi et al. (2016) discriminatory powers of various static feature categories are measured and compared by using mean decrease impurity notion and random forest classifier in a multi-class malware family classification.

Utilization of feature ranking methods is considerably less common in those studies which provide classifiers specifically for mobile malware detection (Feizollah et al., 2015). Lindorfer et al. (2015) applied Fisher's Score to evaluate the discriminatory power of dynamic and static feature categories. This study found out that required permissions and some dynamic features related to SMS sending and dynamic loading of code have higher discriminatory powers (Lindorfer et al., 2015). Cen et al. (2015), created a classifier using Regularized Logistic Regression with Lasso Norm for source code features (java package, class and function levels). Information Gain, Chi-Square and an embedded method of logistic regression were utilized for feature selection. It was found that 10% of the features selected by Information Gain or Chi-Square are sufficient for high detection rates (Cen et al., 2015). Similarly, in Shabtai et al. (2012) filter methods such as Chi-Square, Fisher's Score and Information Gain were applied to some system metric features (e.g., CPU consumption, number of running processes, battery level) in the early times of Android.

Pehlivan et al. (2014) applied feature selection methods such as Information Gain, ReliefF, Correlation Feature Selection (CFS) and consistency-based selection to permissions with different classification models. Random forest classifier that selected 25 permission features with CFS provided the best accuracy. In a similar study by Nezhadkamali et al. (2017), three feature selection methods, L1-based feature selection, Information Gain and Gini Impurity, were used with permissions. All three methods were tested using different machine learning algorithms, such as decision tree, SVM and Random forest. Best results were obtained using Random Forest as classification algorithm and Information Gain as feature selection method (Nezhadkamali et al., 2017).

Sing and Hofmann (2017) used three feature selection methods (Chi-Square, Information gain, and correlation analysis) to select variables and form system calls vector. In Ferrante et al. (2016), an embedded feature selection method was used for classifying the dataset that consisted of features such as system calls, memory usage and CPU usage. Kim and Choi (2014) used Linux kernel features related to memory, CPU and network (summing up to 59 features) to perform malware detection. This study used an embedded model to perform feature selection, ending up eliminating 23 features and using 36 features for their detection system (Kim and Choi, 2014). In Qiao et al. (2016) combined API calls and permissions were processed by two feature selection methods, one-way analysis of variance (ANOVA) (i.e., a filter method) and Support Vector Machine—Recursive Feature Elimination (i.e., a wrapper method). They ended up with top 300 features from API set and 80 from permissions set (Qiao et al., 2016).

Although previously mentioned studies applied feature selection methods and some of them provided considerably detailed analysis about discriminatory powers of used features, none of them analyses the character change and its impact on feature selection.

In Hu et al. (2017) concept drift of mobile malware was modelled with an ensemble learning model in which the feature selection is based on Information Gain. In Jordaney et al. (2017) a concept drift detection method that was based on conformal evaluator is applied to two cases, a binary classification for mobile malware and a multi-class classification for malware. These studies focus on enhancing the detection performance of classifiers with concept drift. However, they do not provide an in-depth analysis of discriminatory powers of feature categories and their impact on concept drift.

3 METHOD

We formulated mobile malware detection as a binary classification problem that requires the discrimination of benign mobile applications from mobile malware samples. As we were able to obtain labelled data, supervised machine learning methods were applied. We followed machine learning workflow, that mainly involves five steps: (1) Data Acquisition, (2) Data Cleaning and Preparation, (3) Feature Selection, (4) Classifier training and Evaluation, (5) Interpretation (Robert, 2014). Sometimes tuning could be applied to the trained classifier, but within the framework of the present study, this step was omitted as it was deemed as unnecessary.

We tested *k*-nearest neighbours (kNN), logistic regression, decision tree, and support vector machines (SVM) for building the classifiers, and used Python programming language and Sci-kit learn library in our implementation. Data acquisition and feature selection stages are detailed in Sections 3.1 and 3.2. We covered two types of feature categories in our datasets: absolute frequency of system calls (numerical features) encountered during the execution of the applications and requested Android standard permissions (categorical features).

3.1 Data Acquisition

In this study, we collected 3000 Android x86 architecture compatible applications as the details are given below:

- 1000 benign applications which were randomly downloaded by the authors from APKMirror repository. They were verified as malware free applications with VirusTotal AntiVirus engine. Legitimate applications date between April 2017 and February 2018. Named as ”legitimate dataset” in this research.
- 1000 malware applications which were randomly selected from Drebin malware dataset. These samples date between August 2010 and October 2012 (Arp et al., 2014). We named this dataset as ”old malware dataset”, and refer to each element in the set as ”old malware”.
- 1000 malware applications which were randomly selected from VirusTotal Academic malware dataset. This dataset, shared by VirusTotal, dates between the end of 2016 and beginning 2018 (VirusTotal, 2018). We named this dataset as ”new malware dataset”, and refer to each element in the set as ”new malware”.

Android requested permissions were directly extracted from AndroidManifest.xml file, included in every application APK file, using Android Asset Packaging Tool (*aapt*). The recent Android distribution, Android 8.0, defines 147 Android standard permissions. A permission profile vector that is composed of the data regarding the presence/absence of each Android standard permission was created for each application.

As the collection of system calls requires to run the application itself, we used an Android emulation environment and Android Debug Bridge (*ADB*) to install, execute, monitor, log and uninstall each application. During the execution, *strace* tool was attached to the main process to obtain the first 2000 system calls. 212 distinct system calls are defined in Bionic x86 library. A frequency vector that included the number of each system call made by the application was formed from the logged data. Prior research have demonstrated that malware could be effectively discriminated with a reduced amount of system calls acquired during the application's boot up and that acquisition of the first 2000 system calls provided the best detection results (Vidal et al., 2017).

Although we selected malware samples from two different time-frames, composing two different malware datasets, we used only one benign dataset comprised of recent applications. In this study, we focused on the analysis of change in selected features according to the evolvement of malware with respect to recent benign applications. This approach is in line with malware detection practices happening in the field as mobile phones are usually not compatible with older applications due to frequent operating system and hardware changes and also changes in applications' installation requirements but the detection systems usually include signatures of all malware samples including the old ones. The impact of the evolvement in benign applications will also be analyzed in the context of concept drift within our future studies.

3.2 Feature Selection and Ranking

We employed a two-step procedure that consists of conducting statistical hypothesis testing for feature selection and applying feature ranking method. The former one chooses the features which significantly differ between the two classes (i.e., legitimate and malware), and the latter one orders the features according to their discriminatory power. Order provided in this step is necessary to optimize the number of features used as predictors and describe behavioural evolvement of malware belonging to different time-frames.

There are three feature selection techniques that can be widely utilized in identifying the features (Aggarwal, 2015). Filter techniques evaluate the suitability of a feature by using a statistical criterion which can be applied irrespective of the classification method used. Wrapper techniques iteratively extend the feature set and evaluate the accuracy of each identified set in a classification model. Embedded techniques also evaluate suitability of the feature set with respect to a particular classification model, but unlike the wrapper one, they attempt to prune the features within the classification process itself. Since wrapper and embedded techniques have higher computational complexity, we utilized filter techniques in the second step.

It is important to emphasize that feature categories used in this study, system calls and permissions, do not have the same data type. System calls are numeric values (i.e., amount of calls issued for each system call) and permissions are categorical (i.e., permission request was present/absent for each standard permission). In both steps, we employed different techniques that are more appropriate for each feature category and its data type. The procedure was performed as follows:

- Step 1: Feature selection by statistical hypothesis testing
  - **System Calls.** System calls which differ between malicious and legitimate applications in terms of mean values were selected. To perform statistical hypothesis testing Welch's Test was used. This test provides more reliable results for the cases of unequal variances (Welch, 1947). The statement of the null (base) hypothesis H_o is that mean values of for the number of system calls among first 2000 calls are the same for legitimate μ_L and malicious μ_M applications, and the statement of the alternative hypothesis H_1 is that mean values are different.

$$H_0: \quad \mu_L = \mu_M$$
$$H_1: \quad \mu_L \neq \mu_M$$

  - **Permissions.** As these features are categorical, we employed χ^2 (chi-squared independence test) which can answer the question if two categorical variables are related or not. The statement of the null hypothesis is that there is no relation between the particular permission and class of the application. The statement of the alternative hypothesis is that there is a relation between particular permission and class.
- Step 2. Feature ranking by Fisher's Score and Gini Index
  - **System Calls.** Fisher's Scores of system calls with mean values that differ significantly between malicious and legitimate applications were computed (i.e., higher Fisher's score values indicate higher discriminatory power).
  - **Permissions.** As permissions are categorical, Gini Index suited better for ordering these features (i.e., lower values of the Gini Index indicate higher discriminatory power).

At first glance, a two step procedure may seem unnecessary. One may suggest ordering features with respect to only their *p*-values, computed during the hypothesis testing step. It should be noted here that linear relationship between the values of Fisher's Score and *p*-values is not strong enough to lead exactly to the same feature orderings. Simulations performed by the authors demonstrated that for numeric values Fisher's Score based selection led to better orderings with respect to classifier accuracy. This fact justifies a two-step feature selection procedure for system calls. Regarding permissions, *p*-values and Gini Index based selection procedures did not lead to sufficient difference in detection accuracy. Nevertheless, a two-step selection procedure was used for the sake of method coherence.

In relation to classifier training, one has to choose desired number of predictors either on the basis of Fisher's Score values or Gini Index values. Note that there are no universal or generic valid thresholds for Fisher's Score and Gini Index values indicating suitability or unsuitability of a particular feature. Based on the outcomes of the feature selection process, we provided our expert judgement to determine the thresholds, selected the sets and verified their prediction performance by creating and testing the learning model.

4 RESULTS & DISCUSSION

4.1 Results of Feature Selection and Ranking

We applied feature selection and classification methods to two different compound datasets: First one (namely L/O) includes 1000 legitimate and 1000 old malware samples, and second one (namely L/N) is composed by 1000 legitimate and 1000 new malware samples. Let us remind that each particular system call was treated as a numeric feature which results in 212 numeric features. Each particular permission was treated as a categorical feature (set or unset), which leads to 147 categorical features. Following the feature selection procedure described in Section 3.2, Welch's test demonstrated that for L/O dataset, 38 numeric features differed significantly between the legitimate and malicious applications for level of significance $\alpha = 0.05$, whereas this number was 43 for L/N dataset. In a similar manner, for the same level of significance, χ^2 filtered out 85 permissions for L/O dataset and 79 permissions for L/N dataset.

In the feature ranking step, Fisher's Score and Gini Index values were computed for numeric and categorical features respectively. This allowed ordering the features with respect to their discriminatory power. As mentioned before, there is no specific threshold on any of the methods performed to select or discard any particular feature, only data knowledge and expertise helps in this selection step. As all Fisher's Score (F) values were relatively low, we selected those system calls having $F > 0.15$. Regarding permissions, all Gini Index (G) values were relatively

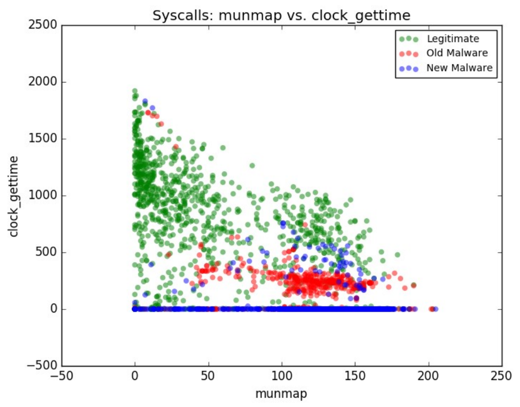

Figure 1: Scatter plot munmap vs clock_gettime.

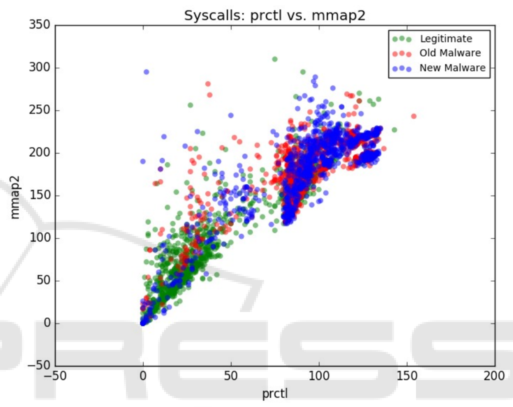

Figure 2: Scatter plot prctl vs mmap2.

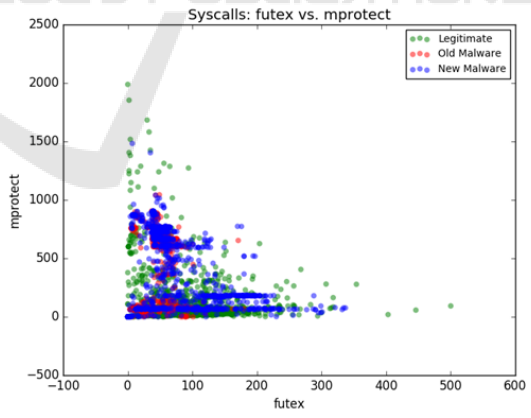

Figure 3: Scatter plot futex vs mprotect.

Table 1: System Calls and Fisher's Score Values.

System Call	L/O	L/N
clock_gettime	0.84	1.11
munmap	0.75	0.57
readlinkat	0.69	0.59
connect	0.67	0.52
mmap2	0.63	0.47
prctl	0.61	0.53
madvise	0.54	0.48
ppoll	0.31	0.25
sigaction	0.29	0.30
sigaltstack	0.23	0.21
openat	0.22	0.16
mprotect	0.15<	0.19
futex	0.30	0.15<
rt_sigprocmask	0.24	0.15<
epoll_create1	0.23	0.15<
eventfd2	0.22	0.15<
getppid	0.22	0.15<
clone	0.21	0.15<
sendto	0.19	0.15<
recvfrom	0.18	0.15<
close	0.17	0.15<
getdents64	0.15	0.15<

Table 2: Permissions and Gini Index Values.

Permission	L/O	L/N
access_network_state	0.46	0.41
wake_lock	0.45	0.39
install_packages	0.42	0.41
read_phone_state	0.32	0.45
get_accounts	>0.47	0.47
system_alert_window	>0.47	0.46
get_tasks	>0.47	0.45
mount_unmount_file_systems	>0.47	0.44
vibrate	>0.47	0.44
access_fine_location	0.47	>0.47
bind_remoteviews	0.47	>0.47
use_fingerprint	0.47	>0.47
camera	0.47	>0.47
bluetooth	0.46	>0.47
read_logs	0.44	>0.47
send_sms	0.43	>0.47
read_contacts	0.43	>0.47
read_external_storage	0.33	>0.47

high so we selected those with $G < 0.47$. System calls possessing higher discriminatory power are listed, together with their Fisher's Score values, in Table 1. Similarly, Table 2 gives the selected permissions with their Gini Index values.

As a result of the second step, 21 features were selected for L/O dataset and 12 for L/N dataset among the system calls (11 of them were common in both datasets). All common system calls in L/N except `clock_gettime` have lower Fisher's Score values. Furthermore, there is only one additional discriminatory system call, `mprotect`, which has a relatively low score, that has been developed in the course of time (appears as potentially discriminatory feature in L/N dataset but not in L/O dataset). Based on that, it can be argued that separability between legitimate and new malware is less obvious, meaning that system call behaviour of malware has become more similar to legitimate as time has passed. Additionally, it can also be argued that beyond this separability fact, new malware has not developed a robust novel character.

Scatter plot graph given in Figure 1 shows an easily recognizable well-defined decision boundary that is formed by two of the most discriminatory system calls, `clock_gettime` and `munmap2`. As shown, old malware is gathered in a cluster which is located between legitimate and new malware regions. On the other side, decreased separability formed by system calls with relatively less Fisher's Score values, such as `prctl` and `mmap2`, is demonstrated in Figure 2. Although most of legitimate and new malware samples form their own clusters which can be separable from each other, boundaries are not so clear when compared to the graph given in Figure 1. Figure 3 shows the graph for two system calls having lower scores such as `futex` and `mprotect`. It is observed that despite some condensed regions occupied by one class, boundaries between old malware, new malware and legitimate apps mostly disappear.

According to Fisher's Score values, it can be derived that system calls that possess best discriminatory power are related to socket connection, process management or file operations. However, best predictor is the one which is related with clock time, showing the most different behaviour between malware and legitimate applications.

Based on Gini Index values (see Table 2) and the established threshold value, we identified that 13 permissions in L/O possess greater discriminatory power whereas 9 permissions have greater power in L/N (among the 147 permissions in total). New malware gained more separability from legitimate applications in features such as `wake_lock`, `access_network_state`, `install_packages`. They exceeded the threshold value in an additional five features which were below that value in old malware. On the other side, it has become closer to legitimate apps in 10 features (for instance, `read_phone_state`, `camera`, `send_sms`, or `read_contacts`). It can be argued that total discriminatory power of new malware has diminished to some extent due to a reduction in the number of selected features, but in contrast to system calls, it gained new character.

Android OS has mainly three protection levels that determine policies for granting permissions to mobile apps: (1) Normal permissions which are automatically given to applications without explicit consent of the user, (2) Dangerous permissions that require explicit consent of the users to be granted, (3) Signature permissions which require that the app that uses the permission must have the same certificate as the app that defines the permission (Google, 2018). Features with greater discriminatory capabilities, which are identified by Gini Index in our study, do not belong to a single level. Among the 18 listed features in Table 2, only 7 of them belong to the dangerous level. This result indicates that malware and legitimate apps can also differ in permissions which do not seem risky.

It is important to note that, in our context, gaining character or having more discriminatory power means that the referenced dataset can better discriminate malware from legitimate apps by using the corresponding feature. It does not show that, for instance, malware uses that specific system call or permission more (or less) frequently than a legitimate app. However, as we utilized the same legitimate dataset, it is evident that the change in discrimination capabilities relies on the change of malware behaviour over time.

Table 3: Classification with System Calls.

# of features	L/O accuracy	L/N accuracy
Single Best Feature [1]	0.87	0.89
3 Best Common Features [2]	0.90	0.88
6 Best Common Features [3]	0.91	0.89
All 11 Common Features selected in both datasets [4]	0.93	0.89
All 22 Selected Features	0.97	0.91
All 212 Features	0.97	0.93

4.2 Verification of Selected Features with Classifiers

In order to verify the results obtained in Section 4.1, we built and tested classifiers with selected feature sets, grouping them in varied sizes. Recall that the filter methods that we use in this study treat each feature separately while measuring its discriminatory power, meaning that these sets do not guarantee higher accuracy due to, for instance, possible correlations among the selected features. This verification study is needed to show the validity of our findings.

We trained and tested *k*- Nearest Neighbours (kNN), Logistic Regression, Decision Tree, and Support Vector Machines (SVM) machine learning algorithms to the datasets. Among these methods, decision tree model demonstrated best accuracy results, therefore, this method was chosen for further analysis. Then decision tree model was applied to L/O and L/N datasets. As shown in Table 3, we computed accuracy value for different decision tree classifiers as a performance metric (i.e., accuracy is computed as the ratio of correctly classified samples to the total samples), using 5-fold cross-validation with varying feature set sizes for system calls. Corresponding confusion matrix of each classifier is given summarized in Table 4.

Table 4: Confusion Matrices for the Classification of System Calls.

# of features	Actual(L)/ Pred(L)	Actual(M)/ Pred(M)	Actual(L)/ Pred(M)	Actual(M)/ Pred(L)
Single Best L/O	265	265	29	41
3 Best L/O	261	279	31	29
6 Best L/O	293	259	25	23
11 Common L/O	299	262	24	15
22 Selected L/O	303	276	10	11
All (212) L/O	295	290	8	7
Single Best L/N	300	234	27	39
3 Best L/N	263	266	39	32
6 Best L/N	259	269	37	35
11 Common L/N	282	254	32	32
22 Selected L/N	272	268	36	24
All (212) L/N	279	281	19	21

Results of decision tree classifier model regarding system calls show that just a single feature, `clock_gettime` (highest Fisher's score value), was capable of discriminating malware from legitimate apps (in both L/O and L/N datasets) with an accuracy over 87 %. However, this feature provided better classification in L/N, which is in line with the higher Fisher's Score value of this feature in this dataset. In all other classifier models built, selected features provided better outcomes in L/O dataset, justifying that similarity of system calls behaviour between a legitimate app and malware is getting less obvious over time.

Accuracy results of classifiers increase as bigger feature set is covered in both datasets. Just the 22 selected features are enough to give the same accuracy performance than using all system calls (212) in L/O dataset. However, a similar point is not achieved in L/N dataset, indicating a decrease in the discrimina-

[1] clock_gettime

[2] clock_gettime, readlinkat, and munmap

[3] clock_gettime, readlinkat, munmap, connect, prctl and mmap2

[4] clock_gettime, readlinkat, munmap, connect, prctl, mmap2, madvise, ppoll, sigaction, sigaltstack, openat

tory power of the selected features. It can be derived from the confusion matrices given in Table 4 that classifiers are, in general, well-balanced in terms of false positive and false negative results, which are represented in the table as "Actual(L)/Predicted(M)" and "Actual(M)/Predicted(L)" respectively. Note that L refers to legitimate whereas M means malware. However, results of the best feature in L/O and L/N are slightly more skewed to false negatives whereas the classifiers with all 11 common features in L/O and all 22 selected features in L/N are more inclined to false positives.

Results regarding the application of decision tree classifier model to permissions are given in Table 5. Best feature provided accuracy values, 0.79 and 0.73, in L/O and L/N datasets respectively. These values are lower compared to the detection performance of best system call predictor. As shown, accuracy value in L/O was greater than in L/N. This fact was expected as the Gini Index score of the best feature in L/O dataset has a lower value than in L/N dataset, i.e. that it has more discriminatory power. Accuracy of the classifier that uses all selected features, in both datasets, reaches almost the same value obtained when all permissions are used, showing the effectiveness of feature selection in permissions.

Table 5: Classification with Permissions.

# of features	L/O accuracy	L/N accuracy
Single Best Feature [5]	0.79	0.73
4 Common Selected Features in both datasets [6]	0.86	0.85
All 18 Selected Features	0.94	0.92
All 147 features	0.95	0.92

Accuracy values of L/N were slightly lower than values of L/O when common or all selected permissions were used. This result suggests that as time has passed, separability between malware and legitimate applications has partly decreased regarding permissions.

Confusion matrices of classifiers built for permissions are summarized in Table 6. It can be extracted that most of classifiers are not well-balanced compared to the ones built on the basis of system calls. Results of the best and four common features in L/O are skewed to false negatives, but remaining ones are more balanced. L/N dataset provided unbalanced outcomes in each classifier. Best feature in L/N gave more false positives and remaining ones were inclined to false negatives.

Table 6: Confusion Matrices for the Classification of Permissions.

# of features	Actual(L)/ Pred(L)	Actual(M)/ Pred(M)	Actual(L)/ Pred(M)	Actual(M)/ Pred(L)
Single Best L/O	271	201	30	98
4 Common L/O	262	248	23	67
18 Selected L/O	284	280	19	17
All (147) L/O	281	290	14	15
Single Best L/N	186	253	117	44
4 Common L/N	281	227	29	63
18 Selected L/N	274	274	19	33
All (147) L/N	284	268	20	28

When outcomes of system calls and permissions are compared, it can be argued that their amount of loss regarding discriminatory power in L/N is different. All selected system calls in L/N gave an accuracy value of 0.91, showing a decline from 0.97 which was obtained in L/O. This value, 0.91, is below the accuracy result, 0.93, which was obtained in L/N when all system calls were used for the classification. On the other side, accuracy value declines from 0.94 to 0.92 for all selected permissions, which indicates a lower amount of loss than selected system calls. Accuracy value of 0.92, is equal to the result obtained by all permissions in L/N. Recall that, in Section 4.1, we identified a decrease from 21 to 12 in the number of system calls which exceeded the selection threshold in L/O and L/N datasets. Out of 12 system calls, just only two of them have higher Fisher's score in L/N. Contrarily, decline in permissions goes from 13 to 9, and more features, 5 of them, have higher discrimination capability in L/N. These findings support the results obtained in Section 4.1 so that system calls and permissions lost part of their discriminatory power in L/N, being the loss in system calls greater than the loss in permissions.

It is important to highlight here that our results regarding the change in selected feature sets indicate a concept drift. Comparison between system calls and permissions given above provides initial insights into the extent of this phenomenon. However, more complete derivations can be drawn with modelling the drift in the classifier. As we focus on feature selection and ranking in this paper, we postponed this modelling effort to our future work.

Table 7 demonstrates detection performance of a mixture of system calls and permissions (hybrid detection approach). Classifier was constructed using decision tree model within a 5-fold cross-validation setting. As can be seen, in both datasets, detection rates were higher compared to their previously built respective single type classifiers, using only static or only dynamic features.

[5]read_phone_state for L/O and wake_lock for L/N

[6]access_network_state, wake_lock, install_packages and read_phone_state for L/O and L/N

[7]clock_gettime and read_phone_state for L/O and

Table 7: Classification with System Calls and Permissions (Hybrid).

# of features	L/O accuracy	L/N accuracy
Best Two Features [7]	0.90	0.89
$4+11$ Common Selected Features in both datasets	0.95	0.92
$18+22$ Selected Features	0.97	0.94
All Features $(212+147)$	0.98	0.94

5 CONCLUSION & FUTURE WORK

Detection of mobile malware remains a significant challenge due to the rapidly evolving nature of the threat. Machine learning techniques have provided solutions to handle this problem. Although they have provided promising results, there is a room for improvement of the classifiers by the utilization of feature selection to obtain better classification accuracy, present the results in a more interpretable way and reduce required computational resources.

In this paper, we applied a feature selection and ranking procedure that consists of two consecutive steps, statistical hypothesis testing and filter feature selection method. The former enables us to select the features while the latter ranks them according to their discriminatory power. We used system calls and permissions as the feature categories due to their proven success in various research studies. Detection performance of selected features was evaluated in decision tree based classifiers. In order to analyze the impact of the changing behaviour on feature selection process, we induced classifiers with malware samples belonging to different time frames.

This study shows that a small number of selected features, such as 3-6 features, provide relatively high accuracy results. Even a single system call, the one possessing best Fisher's Score value in our feature domain, `clock_gettime`, provided accuracy values over 87%. We identified that 10-12% of the features are able to provide a discriminatory power which is very close to the power of using all features in both feature categories (system calls and permissions). Moreover, we identified that system calls and permissions of new malware samples are more similar to legitimate apps than the old ones. This result suggests a concept drift in these features. Additionally, feature rankings and classifier outputs indicate that system calls have lost more discriminatory power over time compared to permissions.

In this paper, we concentrated on feature selection and its implications on accuracy of machine learning classifiers. Findings regarding concept drift can be better explored and enhanced by precisely modelling this learning aspect in the classifier itself. Feature sets used in the classifiers could be enhanced by adding other static or dynamic categories. Also, required length of collection's time period for dynamic attributes such as system calls could be further investigated.

[7] clock_gettime and wake_lock for L/N

REFERENCES

Aggarwal, C. (2015). *Data Mining: The Textbook*. Springer International Publishing.

Arp, D., Spreitzenbarth, M., Hübner, M., Gascon, H., and Rieck, K. (2014). Drebin: Effective and Explainable Detection of Android Malware in Your Pocket. In *Proceedings 2014 Network and Distributed System Security Symposium*, number February.

Cen, L., Gates, C. S., Si, L., and Li, N. (2015). A probabilistic discriminative model for android malware detection with decompiled source code. *IEEE Transactions on Dependable and Secure Computing*, 12(4):400–412.

Fedler, R., Schütte, J., and Kulicke, M. (2013). On the Effectiveness of Malware Protection on Android. Technical report, Fraunhofer, AISEC.

Feizollah, A., Anuar, N. B., Salleh, R., and Wahab, A. W. A. (2015). A review on feature selection in mobile malware detection. *Digital Investigation*, 13:22–37.

Google (2018). Permissions overview. Retrieved from: https://developer.android.com/guide/topics/permissions/overview.

Kim, H.-H. and Choi, M.-J. (2014). Linux kernel-based feature selection for android malware detection. In *Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific*, pages 1–4. IEEE.

Lindorfer, M., Neugschwandtner, M., and Platzer, C. (2015). Marvin: Efficient and comprehensive mobile app classification through static and dynamic analysis. In *2015 IEEE 39th Annual Computer Software and Applications Conference*, volume 2, pages 422–433.

McAfee (2018). McAfee Mobile Threat Report Q1 2018. Retrieved from: https://www.mcafee.com/es/resources/reports/rp-mobile-threat-report-2018.pdf.

Nezhadkamali, M., Soltani, S., and Hosseini Seno, S. A. (2017). Android malware detection based on overlapping of static features. In *7th International Conference on Computer and Knowledge Engineering (ICCKE 2017), October 26-27 2017, Ferdowsi University of Mashhad*.

Qiao, M., Sung, A. H., and Liu, Q. (2016). Merging permission and api features for android malware detection. In *2016 5th IIAI International Congress on Ad-*

vanced Applied Informatics (IIAI-AAI), pages 566–571. IEEE.

Robert, C. (2014). *Machine learning, a probabilistic perspective*. Taylor & Francis.

Sahs, J. and Khan, L. (2012). A Machine Learning Approach to Android Malware Detection. In *2012 European Intelligence and Security Informatics Conference*, pages 141–147.

Statista (2018). Mobile os market share 2017. Retrieved from: https://www.statista.com/statistics/266136/global-market-share-held-by-smartphone-operating-systems/.

Unuchek, R. (2018). Mobile Malware Evolution 2017. Retrieved from: https://securelist.com/mobile-malware-review-2017/84139/.

Vidal, J. M., Orozco, A. L. S., and Villalba, L. J. G. (2017). Malware detection in mobile devices by analyzing sequences of system calls. *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, 11(5):606 – 610.

VirusTotal (2018). How to use VirusTotal Community - VirusTotal. Retrieved from: https://www.virustotal.com/es/documentation/virustotal-community/.

Welch, B. L. (1947). The generalization ofstudent's' problem when several different population variances are involved. *Biometrika*, 34(1/2):28–35.

Yan, G., Brown, N., and Kong, D. (2013). Exploring discriminatory features for automated malware classification. In *International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment*, pages 41–61. Springer.

Yuan, Z., Lu, Y., Wang, Z., and Xue, Y. (2014). Droid-Sec : Deep Learning in Android Malware Detection. In *Sigcomm 2014*, pages 371–372.