X-ANOVA Ranked Features for Android Malware Analysis

P. Vinod
SCMS School of Engineering & Technology, Ernakulam, Kerala

2014 Annual IEEE India Conference (INDICON)

X–ANOVA Ranked Features for Android Malware Analysis

Rincy Raphael
Department of Computer Science
and Engineering
SCMS School of Engineering
and Technology
Ernakulam,Kerala
rincyraphael2012@gmail.com

Vinod P.
Department of Computer Science
and Engineering
SCMS School of Engineering
and Technology
Ernakulam,Kerala
pvinod21@gmail.com

Bini Omman
Department of Computer Science
and Engineering
SCMS School of Engineering
and Technology
Ernakulam,Kerala
binireni@gmail.com

***Abstract*—The proposed framework represents a static analysis framework to classify the Android malware. From each Android .apk file, three distinct features likely (a) opcodes (b) methods and (c) permissions are extracted. Analysis of Variance (X-ANOVA) is used to rank features that have high difference in variance in malware and benign training set. To achieve this conventional ANOVA was modified; and a novel technique referred to us as X–ANOVA is proposed. Especially, X–ANOVA is utilized to reduce the dimensions of large feature space in order to minimize classification error and processing overhead incurred during the learning phase. Accuracy of the proposed system is computed using three classifiers (J48, ADABoostM1, RandomForest) and the performance is compared with voted classification approach. An overall accuracy of 88.30% with opcodes, 87.81% with method and an accuracy of 90.47% is obtained considering permission as features, using independent classifiers. However, using voted classification approach, an accuracy of 88.27% and 87.53% are obtained respectively for features like opcodes and methods. Also, an improved accuracy of 90.63% was ascertained considering permissions. Initial results are promising which demonstrate that the proposed approach can be used to assist mobile antiviruses.**

Keywords - Android Malware, X–ANOVA, Feature Ranking, Classifiers, Mobile Malware

I. Introduction

Android is an open–source mobile operating system, based on a modified version of the Linux kernel. A total market share of 79.3% holds smartphones and tablets of different Android versions [7]. This makes the application developers to design a huge variety of Android apps to attract the users. Google provides an easy way to develop applications to expand the functionality of the devices. This is achieved by using Android Software Development Kit (SDK) and the Native Development Kit (NDK). Developers can release applications on the official Android Market and/or on any number of available third-party stores. There are more than 500 different Android markets that may contain malicious applications [6] that can infect the Android platform by propagating through various Appstores as well as Android devices. According to the F-Secure's Mobile Threat Report [8] in 2014, a total of 275 new malware families were discovered and 88.6% of them were found to be profit motivated families and variants.

The lack of security features in Android platform resulted in the introduction of diverse forms of attacks. Malware writers attempt to inject malicious payload by decompiling a legitimate application and in later stage repackaging them to a downloadable archive. These applications disguised as genuine by naive users are installed in their mobile phones which results in the exposure to unseen attacks. Repackaging may dramatically change the function name or code layout, thereby making it difficult to detect the app from a large collection. Permission misusing is another form of threat in Android environment. The Android permission system grant access to the user's credential information including contacts, photos, position etc. Also during the time of installation, the user unknowingly approve some permissions defined by the attacker thereby causing the device to be affected [14].

To mitigate the above limitations, we implemented a static malware analysis approach to classify the unknown mobile malware using opcode, method and permission features. The dimensionality reduction is achieved by ranking features by extending ANOVA.

In order to summarize the work we make following contributions:

- A statistical scanner for mobile malware identification with fewer features.
- Optimal features are pruned from a large attribute space using statistical technique known as extended ANOVA (X-ANOVA).
- Different categories of features (opcode, permission and method) are evaluated on varied length.
- A voted classification techniques which capture the strength of independent classifier to predict unseen sample is also implemented.

The remaining sections are organized as follows: Section II introduces the related work in the domain of mobile malware analysis. In Section III our proposed methodology is discussed. Experiments are presented in Section IV. Inference of our work is covered in Section V. Finally, concluding remarks and pointers to future work are discussed in Section VI.

978-1-4799-5364-6/14/$31.00 ©2014 IEEE

II. Related Work

In [14], authors conducted survey of mobile malware in three different mobile platform (iOS, Android and Symbian). Malicious and non–malicious apps requested on an average of 6.18 and 3.46 dangerous permissions. The complexity of Android security was revealed in [13]. Different components (Permission, API Call, Service Hook) in android apps defeat application security. The authors in [15] developed a framework that applies various transformation techniques (also known as obfuscation) to identify if the generated malicious samples bypass anti–malware products. They have found that 43% signatures used by the scanners could not capture code level changes (such as repacking, disassembling & reassembling). Also, simple modification to Android manifest file could thwart detection and 90% of signatures were not based on static analysis of bytecode.

SAAF [17] (Static Android Analysis Framework) is a graphical tool that support automated analysis of malware that unpacks and disassembles each application. Dataflow analysis is performed on various slices of smali code of the apk file. The investigation was performed on large number of application without decompilation. DroidMOSS [16] is a similarity measurement system to identify repacking of legitimate applications to localize the region of modification using fuzzy hashing techniques. The experiment was performed on applications downloaded from six third party Android market places. Authors demonstrated that 5% to 13% of applications were modified version of the existing archetype.

In [18] authors characterized malware based on the installation, activation and the features of malicious payload. Study was conducted on 1260 Android samples belonging to 49 different families. Results depicted that 86.0% of legitimate applications were repackaged version of the original application, with the malicious payload. Experiments were conducted with four existing anti–virus softwares where, the best and worst case detection rate of 79.6% and 20.2% was respectively obtained.

In [20] authors suggested a permission based scheme to detect unseen samples of known Android malware families using heuristics filtering scheme. From the collection of 204,040 apps, the proposed system identified 211 malicious apps, where 32 of them from official Android market and 179 appeared in other Android markets. Proposed scheme discovered two zero day malware in 40 application. Another machine learning based malware detection system was introduced in [21] The developed system used permissions and events as features. The experiments were conducted on two dataset with 200 and 500 samples and obtained 91.78% and 91.57% accuracies respectively with RandomForest Classifier.

The [19] proposed a classification method based on application bytecode that contains actual behaviour of an app. Study conducted on 1,300 malware belonging to 26 families. Malware and benign models were developed using RandomForest classifier. An overall weighted average accuracy of 94% was obtained with 218 instructions.

III. Proposed Methodology

This section describes our proposed method for Android malware analysis. The malicious samples are collected from [3] and benign samples are gathered from Google play [4]. Smali/Backsmali [1] tool is used to disassemble .apk file into Dalvik byte code (i.e, dex format). A single .apk file has the multiple .smali files associated with it. From each .smali file opcodes and methods are acquired. The other features are extracted from apps using Androguard [2]. We modified Analysis of Variance (ANOVA) [9] extended by us (X–ANOVA) is employed to rank different categories of features (i.e. opcodes, methods and permissions), and to determine significant features for the identification of application. Figure 1 represents the architecture of the proposed methodology. The working of X–ANOVA is explained in Algorithm **??** which has the complexity of $O(D)$. The creation of feature vector table results in complexity of $O(log\ d)$ where $d << D$.

A. Data Pre–processing

In the data preprocessing phase, weights corresponding to each attribute are obtained from the training instances. The weight is determined by estimating the product of frequency of each variable with the joint probability of a feature with respect to number of files in which they appear.

Subsequently, features are sorted in the non–increasing order of their weights. This results in features that appear with high frequency in maximum number of instances. Consequently, we selected prominent 156 opcodes and 69 methods using an assumed threshold value 0.0001. A total of 78 prominent permissions $(M \cap B)$ that have reasonable difference in frequencies in both the target classes are filtered out.

B. Feature Ranking

Consider a set $A = \{a_1, a_2, \cdots, a_D\}$ representing a feature space with D dimensions. The goal is to determine a set A' such that $|A'| = d$ with $d << D$ and $f(A') >= f(A)$, where f designate the objective function. In other words, feature selection is a process of determining a subset of relevant features from a large feature space for constructing robust classification model. The significance of feature selection techniques are (a) provides better visualization of data (b) occupies less storage space; results in effective retrieval of data (c) improve speed in learning and thus scale up predictive accuracy and (d) few dimension may contain less number of correlated attributes thereby precisely eliminate noise which may otherwise lead to misclassification of instances.

We improved one–way or single factor *Analysis of Variance* (ANOVA) to rank features. This extended version of ANOVA is referred to us as X–ANOVA. Conventional ANOVA is used to test the differences among the means of population by computing the amount of variance within and between samples. However, X–ANOVA is employed to determine all those attributes that have minimal intra–class (malware or benign) and high inter–class variance (malware and benign). The two estimates of variances are used to determine

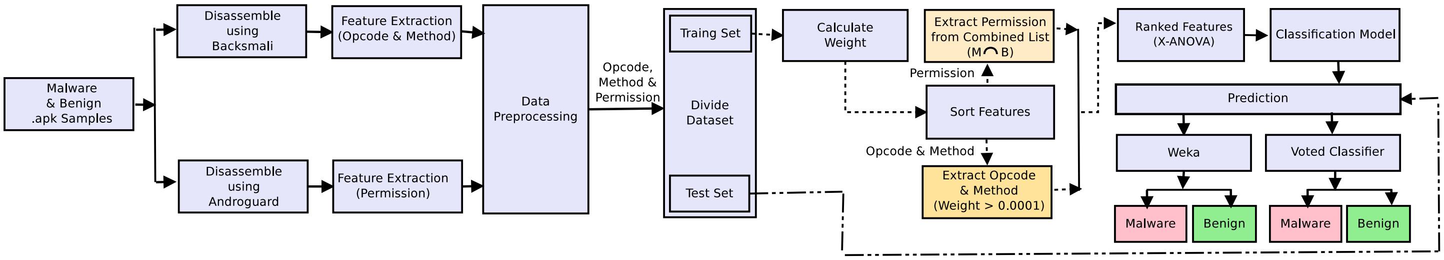


Fig. 1: Architecture of Proposed Methodology

Algorithm 1 : X-Anova Feature Selection

INPUT: $F = \{f_1, f_2......, f_D\}$ **// Set of attributes for D dimensions**

OUTPUT: $S = \{x_1, x_2......, x_d\}$ **// SCF with d dimensions.**

```
// Compute the mean of each feature $O(D)$
for $i \leftarrow 1to|C|$ do                ▷ Class i=1 for M and i=2 for B
    for $j \leftarrow 1to|D|$ do
        $tempsum \leftarrow 0$
        for $k \leftarrow 1to|N|$ do
            $tempsum = tempsum + f^i_{j,k}$
        end for
        $\bar{f^i_{j,k}} \leftarrow tempsum/N$
    end for
end for
// Compute Global Mean
for $j \leftarrow 1to|D|$ do
    $\bar{\bar{f}}_j \leftarrow (\bar{f^M_j} + \bar{f^B_j})/2$
end for
// Compute $SS_{between}$ and $MS_{between}$ for each attribute
for $i \leftarrow 1to|D|$ do
    $SS_{between i} \leftarrow |M|(\bar{f^M_i} - \bar{\bar{f}}_i)^2 * |B|(\bar{f^B_i} - \bar{\bar{f}}_i)^2$
    $MS_{between i} \leftarrow \frac{SS_{between i}}{|C|-1}$
end for
// Compute $SS_{within}$ and $MS_{within}$
for $i \leftarrow 1to|D|$ do
    $tempsumM \leftarrow 0$
    for $j \leftarrow 1to|M|$ do
        $tempsumM \leftarrow tempsumM + (f^M_j - \bar{f^M_i})^2$
    end for
    $tempsumB \leftarrow 0$
    for $j \leftarrow 1to|B|$ do
        $tempsumB \leftarrow tempsumB + (f^B_j - \bar{f^B_i})^2$
    end for
    $SS_{within i} \leftarrow tempsumM + tempsumB$
    $MS_{within i} \leftarrow \frac{SS_{within i}}{(|M|+|B|-|C|)}$
    $F_{value i} \leftarrow \frac{MS_{between i}}{MS_{within i}}$
    if $(F_{value i} > Standard\,Value)$ then
        $S = S \cup \{f_i\}$
    end if
end for
```

F–test value which is later compared with the standard F–value in the 0.05 significant table [5].

$$F = \frac{Variance\ of\ attributes\ between\ two\ classes}{Variance\ of\ attribute\ within\ class} \qquad (1)$$

Consider an example (refer Table I and II) which depict the implementation of X–ANOVA. Feature vector tables shows 5 malware, 4 benign instances along with the frequency of attributes $(A_1, A_2, A_3, A_4, A_5)$.

TABLE I: Attribute values of Malware Samples

	A_1	A_2	A_3	A_4
M_1	3	2	0	5
M_2	4	0	3	1
M_3	0	5	2	4
M_4	1	3	4	2
M_5	1	4	5	0
$\bar{A_i}^M$	1.8	2.8	2.8	2.4

TABLE II: Attribute values of Benign Samples

	A_1	A_2	A_3
B_1	2	4	0
B_2	1	0	3
B_3	0	5	2
B_4	5	3	4
$\bar{A_i}^B$	2	3	2.2

The global attribute mean of first attribute is $\bar{\bar{A}}_1 = 1.9$. Using the steps of X–ANOVA as discussed in algorithm 1, compute $SS_{between}$ (i.e., $SS_{between}[A_1] = 5(1.8 - 1.9)^2 * 4(2 - 1.9)^2 = 0.002$). Likewise, with the known information of $SS_{between}[A_1]$ and the degree of freedom, $MS_{between}$ for each attribute is computed. Similarly, SS_{within} for each instance and a attribute (A_{ij}) using class mean $\bar{A_i^M}$ (i.e, $SS_{within}[A_1] = (3 - 1.8)^2 + (4 - 1.8)^2 + (0 - 1.8)^2 + (1 - 1.8)^2 + (1 - 1.8)^2 + (2 - 2)^2 + (1 - 2)^2 + (0 - 2)^2 + (5 - 2)^2 = 24.8$) is computed. Evaluate the MS_{within}. Thus, value $MS_{within} = 3.54$. Finally, $F - ratio$ is determined by dividing $MS_{between}$ by MS_{within}. For attribute A_i, the $F - ratio = 1.77$. Check if $MS_{between} > MS_{within}$, then consider the degree of freedom between the classes (i.e., $C - 1 = 2 - 1 = 1$) in the x-axis and degree of freedom within the class (i.e., $N - C = 9 - 2 = 7$) indexing at y-axis. As $F - ratio > F_{Table\ value}$ (i.e., $1.77 > 5.59$) is false, hence the attribute A_i is considered as insignificant.

Top 78 common permissions, 67 methods, 156 unigram and 800 bigram opcodes extracted from training dataset are independently supplied as input to X–ANOVA to obtain reduced attribute space. After pruning the features, 47 permissions, 49 methods, 64 unigram and 130 bigram opcodes are obtained. For methods as the attribute, the classification model is developed by considering the boolean and frequency based feature vector table.

C. Evaluation of Accuracy

Accuracy of proposed framework is calculated by two different methods (i) using individual classifier implemented in WEKA [10] and (ii) with voted classification approach.

1) Using Individual Classifier: The performance of classification model developed using tree based classifier (J48, ADABoostM1 (ADA) [11] and RandomForest (RF) [12]) is evaluated by estimating the number of false positives (FP), true positives (TP), false negatives (FN) and true negatives (TN). **Accuracy**: Degree of correctness of the model in classifying the unseen samples

$$Accuracy(ACC) = (TP + TN)\ /\ (TP + TN + FP + FN) \quad (2)$$

True Positive Rate (TPR): Rate of malware correctly classified as malware

$$TPR = TP\ /\ (TP + FN) \quad (3)$$

Precision: Fraction of relevant instance retrieved from sample set

$$Precision = TP\ /\ (TP + FP) \quad (4)$$

where, FP denote the misclassified benign samples, TP indicate correctly classified malware instances, FN represents wrongly classified malware samples and TP denotes correctly identified malware files.

2) Using Voted Classifier: In this approach, maximum vote from the classifiers (J48, ADABoost and Random Forest) are collected to arrive at a conclusion for any test instance. The obtained result from voted scheme is then compared with the actual class of test sample (as in case of supervised learning approach) to determine the final outcome.

$$Accuracy = (TD/N) * 100 \quad (5)$$

TD represents the number of true detection or number of correctly detected files and N shows the total number of test samples.

IV. Experiments and Results

Experiments are performed in Ubuntu 12.10 platform with the support of Intel core i5 processor and 4GB RAM. The investigations are carried out on different classification models constructed using diverse category of features with variable feature length. Through the study we answer the following research questions:
(a) Which feature set results in better performance? (b) What is the effect of varying feature length on the classification accuracy? (c) Does dimensionality reduction using proposed X–ANOVA improves the classification accuracy compared to the complete feature space? and (d) Which classifier depicted better detection rate?

A. Experiment with Unigram opcode

From 401 training samples, 435 unique opcodes are extracted. Additionally, weight for each opcode is determined (using equation 1, Section III). The weighted opcodes are sorted in non–increasing order. Subsequently, 156 opcodes are selected by fixing a threshold on weight i.e. 0.0001. Afterwards the application of X–ANOVA resulted in reduced number of opcodes (i.e. 64 features) from a higher dimensional opcode space (156 opcodes).

Figure 2a demonstrates the accuracy of different classifiers such as J48, ADABoost and RandomForest for variable feature length (10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 64, 156). The result explains that, AdaBoost resulted in the highest accuracy 87.06% for feature length 64. Followed by accuracy value of 85.82% obtained with J48 and RandomForest using feature length 25 and 50 respectively. We can also visualize with full 156 features, AdaBoost exhibited 86.81% accuracy which is lesser compared the pruned feature space with 64 significant opcodes.

B. Experiment with Bigram opcode

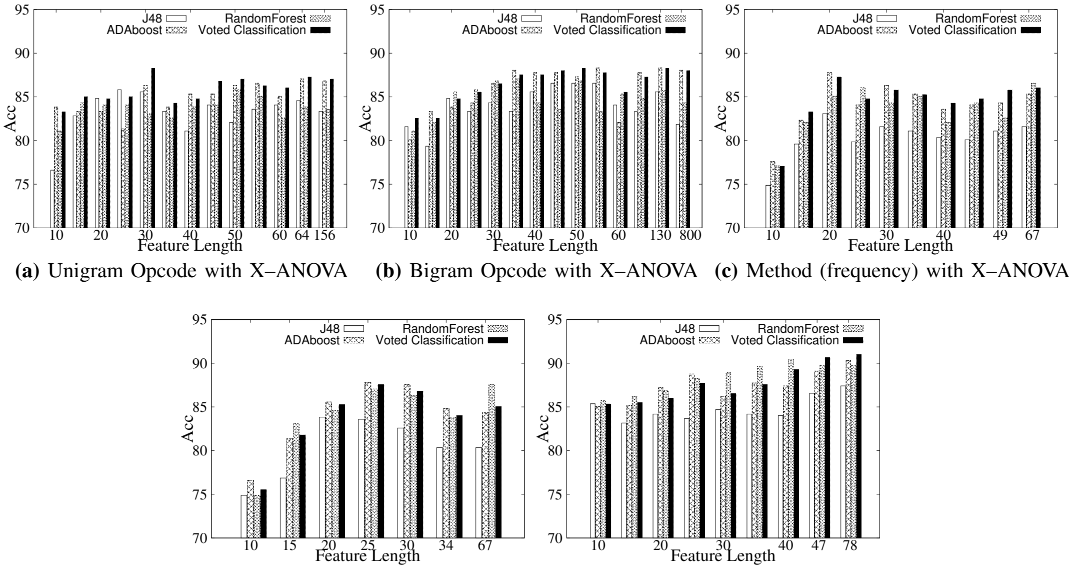
We obtained 800 bigram opcodes out of 14,461 unique bigram opcodes on comparing with the determined threshold. After pruning the feature length by applying X–ANOVA, 130 bigram opcodes are acquired. The accuracies for variable feature lengths (10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 64, 130, 800) are exhibited in the Figure 2b. Highest accuracy of 88.30% is obtained with Adaboost classifier considering prominent 55 opcodes. Accuracies of 86.56% and 87.06% are obtained respectively with J48 and RandomForest with 45 and 35 bigram opcodes. Prior to feature synthesis, 88.05% of accuracy is estimated with 800 features. On comparing the accuracy values of smaller attribute set with that of higher dimensional space we observed that X–ANOVA resulted in better performance.

C. Experiment with Method

A total of 81,052 unique method names are retrieved from the training set. Subsequently, application of X–ANOVA resulted in 49 notable method attributes. Learning models are developed by using frequency and boolean based FVT. ADABoost Classifier obtained the highest accuracy of 87.81% with the 20 significant methods, if the number of occurrence of method is represented in FVT. Likewise, with distinguishable 25 methods on boolean FVT resulted in similar accuracy (refer, Figure 2c and 2d).

D. Experiment with Permission

The top common 78 permissions are obtained from the total of 195 and 190 unique permissions respectively from the malware and benign training sets. By applying X–ANOVA, the feature length is reduced from the 78 to 47. The Figure 2e conveys the accuracy result of the different feature length (10, 15, 20, 25, 30, 35, 40, 47, 78). The highest accuracy of 90.47% is obtained for the feature length 40 using RandomForest.



(a) Unigram Opcode with X–ANOVA **(b)** Bigram Opcode with X–ANOVA **(c)** Method (frequency) with X–ANOVA

(d) Method (boolean) with X–ANOVA **(e)** Permission with X–ANOVA

Fig. 2: Accuracy of feature ranking methods with different feature length

E. Experiment with Voted Classification

The performance of voted tree based classifier is computed by applying the equation 11. Figure 2a exhibits the accuracy of unigram opcode for the feature length 30. The 354 samples are correctly identified from the total of 401 test files and it has the highest accuracy as 88.27%. Figure 2b demonstrate the bigram opcode accuracy (88.27%) for the feature set 50, where 354 files of 401 instances are correctly identified.

Likewise considering methods as features, 87.28% and 87.53% of accuracies are respectively acquired for frequency based FVT (feature length 20) and with boolean based FVT (reduced feature set of 25 significant methods). The results shown in Figure 2c and Figure 2d respectively. From the total 587 samples in test data set (with extracted permission), 523 files are identified correctly and the highest accuracy of 90.63% with notable 47 permissions are ascertained (refer Figure 2e).

F. Comparative Analysis of Voted Classifier with Individual Classifier

Table III and IV shows the comparative analysis of evaluation metrics for individual classifiers as well as the voted tree based classifier. The results depict that the dimensionality reduction improves the classification accuracy. The voted classification scheme exhibits better accuracy in the case of opcode (88.27%) and permission (90.63%) features. However, performance of voted scheme with method as features show insignificant decrease i.e. 87.28% (-0.53 with respect to individual classification approach) and 87.53% (-0.08 with respect to individual classifier) respectively for frequency and boolean FVT representation. In the case of TPR, boolean method depicts highest value (92.09%) and lower value is obtained with unigram opcode (85.58%). However, permission results in higher precision value 92.30% and the lesser value (86.08%) is attained with boolean method.

V. Inferences

From Table III and Table IV, we identified that permissions resulted in high accuracy and high precision rates compared with all other categories of features used in this study. It follows that metadata information of Android Manifest file such as permission are significant for identification of mobile malware in preliminary phase. However, the performance obtained with *ngram* models are not comparable with that of permissions. The prime reason is that opcode can be morphed or obfuscated, whereas it is difficult to obfuscate permission as every application require minimal set of permissions to execute and interact with system services.

As we increase the features in the feature space the classification accuracy delineate and the processing time also increases. This is due to the fact that large attribute space contain insignificant features that appear as noise and lead to high misclassification. Whereas, we visualize that smaller features represent the complete dataset without affecting the performance of the prediction model. This justifies the need of dimensionality reduction such as X–ANOVA. Also, Adaboost and Random Forest resulted in improved performance in all cases. This is because of the notable characteristics of these classifiers such as *bagging, boosting* and *ensemble* properties to arrive at a final decision for any test instance [11] [12].

VI. Conclusion

The proposed static mobile malware analysis model uses three features (opcode, permission and method) to classify the Android malware. The individual tree based classifiers depicts 88.30% and 87.81% accuracy considering learning models

TABLE III: Before Applying X–ANOVA

Feature	Individual Classifier Accuracy	TPR	Precision	Feature Length	Classifier	Voted tree based Classifier Accuracy	Feature Length	TD / Total no of samples
Unigram opcode	86.81	89.30	86.48	156	ADA	87.03	156	349/401
Bigram opcode	88.05	89.30	88.47	800	ADA	88.02	800	353/401
Permission	90.30	87.45	92.27	78	ADA	90.97	78	534/587
Method (Frequency_FVT)	86.56	91.69	84.54	67	RF	86.03	67	345/401
Method (Boolean_FVT)	87.56	90.69	86.66	67	RF	85.03	67	341/401

TABLE IV: After Applying X–ANOVA

Feature	Individual Classifier Accuracy	TPR	Precision	Feature Length	Classifier	Voted tree based Classifier Accuracy	Feature Length	TD / Total no of samples
Unigram opcode	87.06	85.58	89.75	64	ADA	88.27	50	354/401
Bigram opcode	88.30	88.37	89.62	55	ADA	88.27	30	354/401
Permission	90.47	87.80	92.30	40	RF	90.63	47	532/587
Method (Frequency_FVT)	87.81	90.23	87.38	20	ADA	87.28	20	350/401
Method (Boolean_FVT)	87.61	92.09	86.08	25	ADA	87.53	25	351/401

developed with opcodes and methods features. However, permission based classification model obtained 90.47% accuracy. The dimensionality reduction is achieved by applying the modified version of feature ranking method X–ANOVA which improves the performance in comparison to full feature space and reduces the processing time. In future, we would like to investigate on the use of ensemble features, feature selection methods to improve detection rate with minimum false alarms.

REFERENCES

[1] Smali/Backsmali: https://code.google.com/p/smali/ (Accessed on March 4, 2014)

[2] Androguard: http://code.google.com/p/androguard/ (Accessed on September 12, 2013)

[3] Malware apk: http://contagiominidump.blogspot.in/2011/07/take-sample-leave-sample-mobile-malware.html (Accessed on September 22, 2013).

[4] Google Play Store: https://play.google.com/store/apps/collection/topselling_free?hl=en

[5] The Table for Critical Values of F for the 0.05 Significance Level: www.faculty.citadel.edu/silver/fvalues_five.pdf (Accessed on May 16, 2014)

[6] Juniper: http://www.juniper.net/us/en/local/pdf/additional-resources/3rd-jnpr-mobile-threats-report-exec-summary.pdf (Accessed on June 3, 2014).

[7] Mobile Threat Report 2013: www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q3_2013.pdf (Accessed on June 3, 2014).

[8] Mobile Threat Report 2014: www.f-secure.com/static/doc/labs_global/Research/Mobile_Threat_Report_Q1_2014_print.pdf (Accessed on June 3, 2014).

[9] ANOVA: C.R. Kothari, *“Research Methodology-Methods and Techniques”*, published by New Age Publications (Academic), India, Volume 6, March 2006.

[10] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, Ian H.Witten, *“The WEKA Data Mining Software: An Update”*, SIGKOD Explorations, vol. 11, Issue. 1 (2009)

[11] Y. Freund and R.E Schapire, *“Experiments with a new Boosting Algorithm, Machine Learning”*, In Proceedings of the Thirteenth International Conference, pp. 148156, 1996.

[12] A.Liaw and M. Wiener, *“Classification and Regression by Random Forest”*, R News, pp. 18-22, December, 2002.

[13] William Enck, Machigar Ongtang and Patrick Drew McDaniel, *“Understanding Android Security”*, IEEE Security & Privacy, Journal, Volume 7, pp. 50-57, January 2009.

[14] A.P. Felt, M. Finifter, E. Chin, S. Hanna and D. Wagner, *“A Survey of Mobile Malware in the Wild”*, In the proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, SPSM’11, pp. 3-14, New York, October 17,2011.

[15] Vaibhav Rastogi, Yan Chen and Xuxian Jiang, *“Catch Me If You Can: Evaluating Android Anti-Malware Against Transformation Attacks”*, IEEE Transaction on Information Forensics and Security, Volume 9, pp 99-108, January 2014.

[16] Wu Zhou, Yajin Zhon, Xuxian Jiang and Peng Ning, *“Detecting Repacked Smartphone Applications in Third-Party Android Marketplaces”*, CODASPY ’12 Proceedings of the second ACM conference on Data and Application Security and Privacy, pp. 317-326, ACM, New York, USA, 2012.

[17] Johannes Hohhman, Martin Ussath, Thorsten Holz and Michael Spreitzenbarth *“Slicing Droids: Program Slicing for Smali Code”*, SAC ’13 Proceedings of the 28th Annual ACM Symposium on Applied Computing, pp. 1844-1851, USA, 2013.

[18] Y. Zhou and X. Jiang, *“Dissecting Android malware: Characterization and evolution”*, IEEE Symposium on Security and Privacy’12”,pp. 95-109, 2012.

[19] Byeongho Kang, BooJoong Kang, Jungtae Kim, Eul Gyu Im *“Android Malware Classification Method: Dalvik Bytecode Frequency Analysis”*, RACS ’13 Proceedings of the 2013 Research in Adaptive and Convergent Systems, pp. 349-350, ACM, New York, 2013.

[20] Yajin Zhou, Zhi Wang, Wu Zhou and Xuxian Jiang, *“Hey, You, Get off of My Market: Detecting Malicious Apps in Official and Alternative Android Markets”*, In Proceedings of the 19th Network and Distributed System Security Symposium (NDSS 2012) San Diego, CA, February 2012.

[21] Chun-Ying Huang and Yi-Ting Tsai and Chung-Han Hsu, *“Performance Evaluation on Permission-Based Detection for Android Malware”*, In Proceedings of International Computer Symposium (ICS), December 2012.

TABLE V: Top 5 Relevant Opcode Features

Opcode	Description
ushr-int/lit8	Unsigned right shift of integer value
throw	Throws an exception object
sub-long/2addr	Substract two long address values
sub-long	Substract values in long format
sub-float/2addr	Substract two address float value

TABLE VI: Top 5 Relevant Permission Features

Permission	Description
WRITE_SYNC_SETTINGS	Write the sync settings
WRITE_SETTINGS	Read or write system settings
WRITE_SECURE_SETTINGS	Read or write secure system settings
WRITE_EXTERNAL_STORAGE	Write to external storage
WRITE_CONTACTS	Write (but not read) the user’s contacts data