The 32nd International Conference on Software Engineering & Knowledge Engineering, July 9 - July 19, 2020

Significant API Calls in Android Malware Detection

(Using Feature Selection Techniques and Correlation Based Feature Elimination)

Author 1

Asadullah Hill Galib
Masters Student
Institute of Information Technology
University of Dhaka

Author 2

Dr. B. M. Mainul Hossain
Associate Professor
Institute of Information Technology
University of Dhaka

Outline

- Introduction
- Research Questions
- Significant API Calls Identification Approach
- Evaluation
- Limitation
- Conclusion

Introduction

- API Calls are commonly used for characterizing and separating malware from benign applications
- Android OS uses a lot of API Calls and the number continues to expand
- This large API features set may overfit the model or complicated the classification

Introduction

- This study dealt with examining API Calls for reducing the irrelevant ones without tampering significant API Calls
- It proposes and assesses a feature reduction approach for identifying significant API Calls
- Its reduced features set performs notably in terms of accuracy, precision, recall, f-1 score, AUC, and execution time

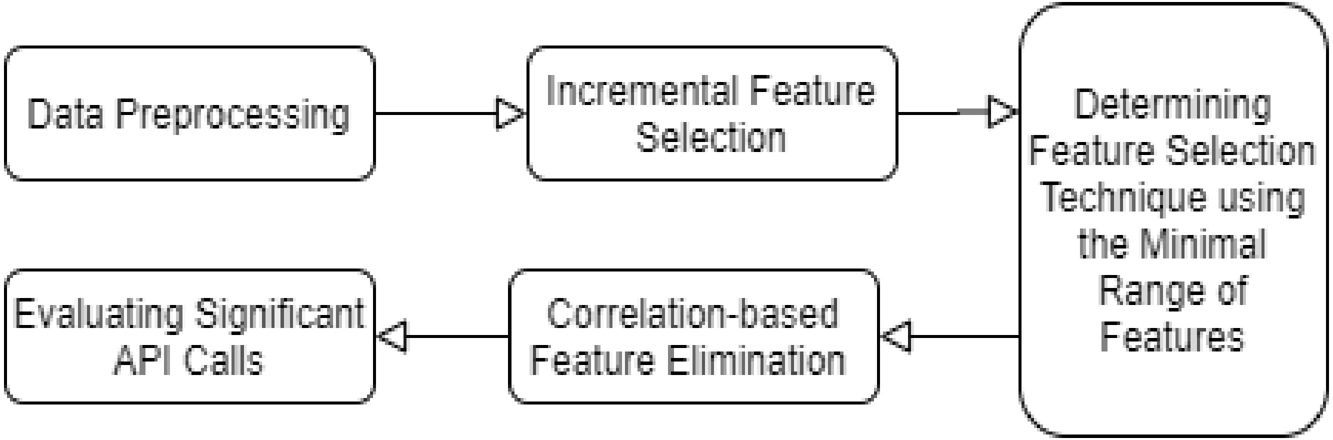
Introduction

- It provides a list of the top significant API Calls in Android malware detection
- It is the first study on significant API Calls in Android malware detection

Research Questions

- RQ1: How can we sort out the significant API Calls (features) in Android malware detection?
- RQ2: How do the significant API Calls perform in detecting Android Malware?
- RQ3: Which API Calls are Significant in Android Malware Detection?

Significant API Calls Identification Approach



Incremental Feature Selection (IFS)

- First, how many numbers of API Calls should we choose?
  - For each feature selection technique, from one to the highest number of API Calls are assessed
  - The optimal/minimal number of API Calls is evaluated by analyzing performance for different numbers of API Calls

Incremental Feature Selection (IFS)

- Second, which feature selection technique is more suitable in reducing API Calls?
  - Feature Selection Techniques analyzed:
    - Mutual Information Gain (Entropy Based)
    - Univariate ROC-AUC Score
    - SelectKBest with chi-squared distribution

Incremental Feature Selection (IFS)

- Feature Selection Techniques analyzed:
  - SelectFromModel (Tree-Based)
    - Classifier: Random Forest and Extra Trees
  - Recursive Feature Elimination (RFE)
    - Classifier: Random Forest and Gradient Boosting

Determining Feature Selection Technique using the Minimal Range of Features

- Analysis of performance metrics to identify the minimal range of features for each techniques
- Selection of feature selection technique which gives the lowest minimal range

Correlation-based Feature Elimination

- Pearson correlation-based feature elimination strategy is applied for further reduction of API Calls
  - If two features are highly correlated, then removing one of them would not affect the classification
  - The less important feature from each pair is removed

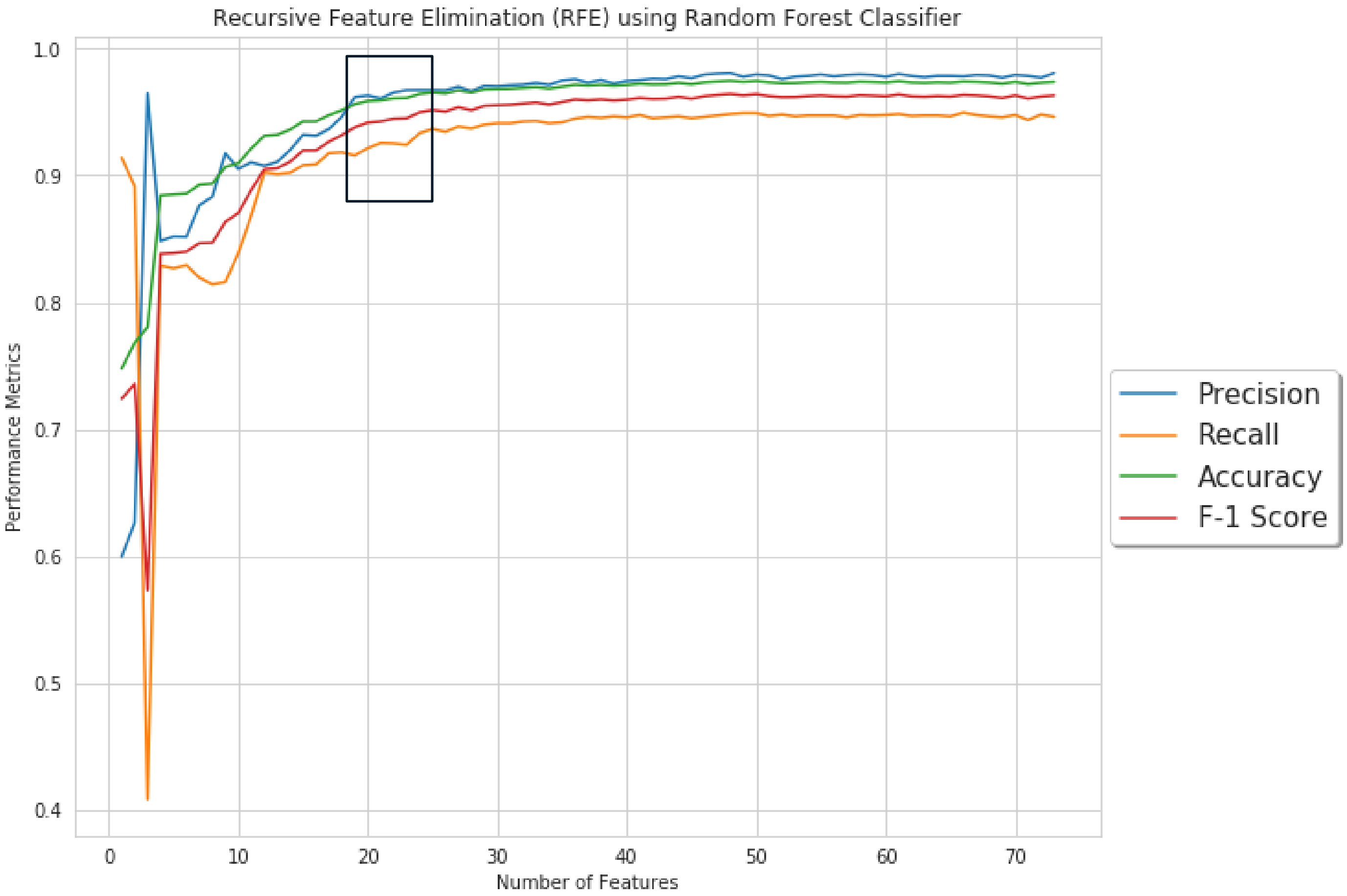
Evaluation (Dataset)

- Drebin
  - 5560 malware samples, 9470 benign samples
  - 73 API Calls
- Android Malware Genome Project
  - 1200 malware samples, 2539 benign samples
  - 69 API Calls

Evaluation (RQ1)

- RQ1: How can we sort out the significant API Calls (features)?
  - Identifying Feature Selection Technique using Minimal Range of Features
  - Correlation-based Feature Elimination on the Minimal Features Sets

Identifying Minimal Range of Features for RFE with Random Forest



Minimal Range of Features for Different Feature Selection Techniques

TABLE I. MINIMAL RANGE OF FEATURES FOR DIFFERENT FEATURE SELECTION TECHNIQUES

Feature Selection Technique	Minimal Range for Drebin	Minimal Range for Malgenome
Mutual Information Gain	37-42	23-28
Univariate ROC-AUC Score	35-38	25-30
RFE with Gradient Boosting Classifier	23-28	20-25
RFE with Random Forest Classifier	18-25	18-21
SelectKBest with chi2	47-50	30-33
SelectFromModel with Random Forest Classifier	25-30	17-22
SelectFromModel with Extra Trees Classifier	28-33	20-23

Correlation-based Feature Elimination on the Minimal Features Sets

TABLE II. CORRELATION-BASED FEATURE ELIMINATION (CFE) ON THE MINIMAL FEATURE SETS

Number of API Calls		
Minimal Feature Sets	**With CFE for Drebin**	**With CFE for Malgenome**
18	15	16
19	15	17
20	17	18
21	19	19
22	20	20
23	21	21
24	21	21
25	22	22

Evaluation (RQ2)

- RQ2: How do the significant API Calls perform in detecting Android Malware?
  - Performance Evaluation
  - Execution Time
  - Comparison with the Existing Works

Performance Evaluation using Malgenome Project

TABLE IV. PERFORMANCE EVALUATION OF THE SIGNIFICANT API CALLS (MALGENOME)

Feature Selection Technique	# of API Calls	Acc (%)	Pre (%)	Rec (%)	F-1	AUC
All Features	69	98.71	98.87	97.22	0.980	0.998
RFE with Random Forest Clas-sifier	25	98.12	98.16	96.19	0.972	0.998
	23	98.03	98.15	95.87	0.970	0.998
	21	98.10	98.07	96.10	0.971	0.996
	19	96.16	97.92	96.51	0.972	0.996
	17	97.97	97.82	96.03	0.969	0.995
	15	97.26	97.62	94.05	0.958	0.993

Performance Evaluation using Drebin

TABLE III. PERFORMANCE EVALUATION OF THE SIGNIFICANT API CALLS (DREBIN)

Feature Selection Technique	# of API Calls	Acc (%)	Pre (%)	Rec (%)	F-1	AUC
All Features	73	98.32	98.62	96.17	0.974	0.996
RFE with Random Forest Classifier	25	97.07	97.41	94.60	0.960	0.993
	23	96.69	96.92	94.06	0.955	0.993
	21	96.26	96.64	93.15	0.949	0.992
	19	96.17	96.26	93.27	0.947	0.991
	17	95.62	95.58	92.43	0.940	0.988
	15	95.38	96.03	91.29	0.936	0.986

Execution Time

TABLE V. EXECUTION TIME OF THE SIGNIFICANT API CALLS

# of API Calls	Execution Time (s) for Drebin	Execution Time (s) for Malgenome
All	6.48	5.76
25	4.15	4.02
23	4.11	3.98
21	4.11	3.88
19	3.95	3.85
17	3.91	3.78
15	3.90	3.74

Comparison with the Existing Works

TABLE VI. COMPARISON WITH THE EXISTING WORKS

Works	Detection Rate (%)
SigAPI (20)	$\boxed{96.30}$
Drebin [10]	93.90
SigPID [15]	93.62
Yerima et al. [11]	92.1%
Yerima et al. [12]	97.5%
Peiravian et al. [13]	95.75%
DroidAPIMiner [14]	~99%
Altaher et al. [16]	91%

Evaluation (RQ3)

- RQ3: Which API Calls are Significant in Android Malware Detection?
  - Top 25 significant API Calls are reported

Top 25 Significant API Calls

TABLE VII. TOP 25 SIGNIFICANT API CALLS (DREBIN)

TOP 25 SIGNIFICANT API CALLS (DREBIN)	
transact	ClassLoader
onServiceConnected	Landroid.content.Context.registerReceiver
bindService	Ljava.lang.Class.getField
attachInterface	android.content.pm.PackageInfo
ServiceConnection	TelephonyManager.getLine1Number
android.os.Binder	Ljava.lang.Class.getMethod
Ljava.lang.Class.getCanonicalName	android.telephony.gsm.SmsManager
Ljava.lang.Class.getMethods	TelephonyManager.getSubscriberId
Ljava.lang.Class.cast	Ljava.lang.Object.getClass
Ljava.net.URLDecoder	TelephonyManager.getDeviceId
android.content.pm.Signature	HttpUriRequest
android.telephony.SmsManager	Runtime.exec

Limitation

Threats to Internal Validity	Threats to External Validity
The parameters of different techniques and algorithms, execution time measurement are susceptible to bias	The datasets are subject to bias and induce lack of generalizability

Related Works

- Li et al.
  - Reduced 135 Permission features to 22 features.
  - Using Permission ranking with negative rate, support based Permission ranking, and Permission mining with association rules for feature selection.

Related Works

- Wang et al.
  - Implemented three measures of scoring on the permission features
  - Discovered dangerous permission subsets using Sequential Forward Selection (SFS) and Principal Component Analysis (PCA).

Related Works

- Altaher et al.
  - based on ANFIS with fuzzy c-means clustering using significant Permissions.

Conclusion

- Reduced features set of significant API Calls would be convenient in classifying Android malware considering performance and computational complexity
- In the future
  - Reevaluation using different and large datasets
  - Deep learning and ensemble learning will be employed

References

1. D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, “Drebin: Effective and Explainable Detection of Android Malware in Your Pocket,” Proceedings 2014 Network and Distributed System Security Symposium, 2014.
2. Yajin Zhou, Xuxian Jiang, "Dissecting Android Malware: Characterization and Evolution," Proceedings of the 33rd IEEE Symposium on Security and Privacy (Oakland 2012), San Francisco, CA, May 2012
3. Yerima, S. Y., Sezer, S., McWilliams, G., & Muttik, I. (2013, March). A new android malware detection approach using Bayesian classification. In 2013 IEEE 27th international conference on advanced information networking and applications (AINA) (pp. 121-128). IEEE.
4. Peiravian, N., & Zhu, X. (2013, November). Machine learning for android malware detection using permission and api calls. In 2013 IEEE 25th international conference on tools with artificial intelligence (pp. 300-305). IEEE.
5. Yerima, S. Y., Sezer, S., & Muttik, I. (2014, September). Android malware detection using parallel machine learning classifiers. In 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies (pp. 37-42). IEEE.

References

6. Aafer, Y., Du, W., & Yin, H. (2013, September). Droidapiminer: Mining api-level features for robust malware detection in android. In International conference on security and privacy in communication systems (pp. 86-103). Springer, Cham.

7. Li, J., Sun, L., Yan, Q., Li, Z., Srisa-an, W., & Ye, H. (2018). Significant permission identification for machine-learning-based android malware detection. IEEE Transactions on Industrial Informatics, 14(7), 3216-3225.

8. Altaher, A., & BaRukab, O. (2017). Android malware classification based on ANFIS with fuzzy c-means clustering using significant application permissions. Turkish Journal of Electrical Engineering & Computer Sciences, 25(3), 2232-2242.

9. Wang, W., Wang, X., Feng, D., Liu, J., Han, Z., & Zhang, X. (2014). Exploring permission-induced risk in android applications for malicious application detection. IEEE Transactions on Information Forensics and Security, 9(11), 1869-1882

Thank you!

Any questions?