SigPID: Significant Permission Identification for Android Malware Detection

Authors: Lichao Sun, Zhiqiang Li, Qiben Yan, Witawas Srisa-an and Yu Pan

Department of Computer Science and Engineering

University of Nebraska Lincoln

Presenters: Yu Pan

Android OS

- Android is the most popular operating system for smart-mobile devices
- Android is also widely used in other mobile platforms, such as tablets, smart tvs, and smartwatches, etc.

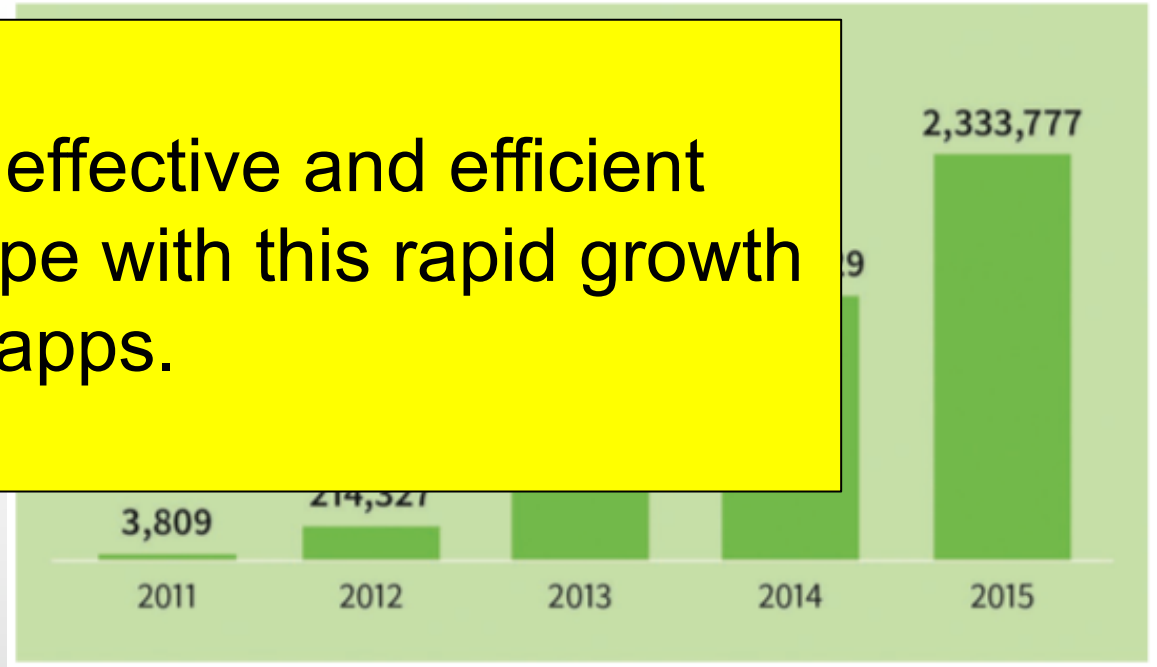
Period	Android	iOS	Windows Phone	BlackBerry OS	Others
2015Q2	82.8%	13.9%	2.6%	0.3%	0.4%
2014Q2	84.8%	11.6%	2.5%	0.5%	0.7%
2013Q2	79.8%	12.9%	3.4%	2.8%	1.2%
2012Q2	69.3%	16.6%	3.1%	4.9%	6.1%

Source: IDC, Aug 2015

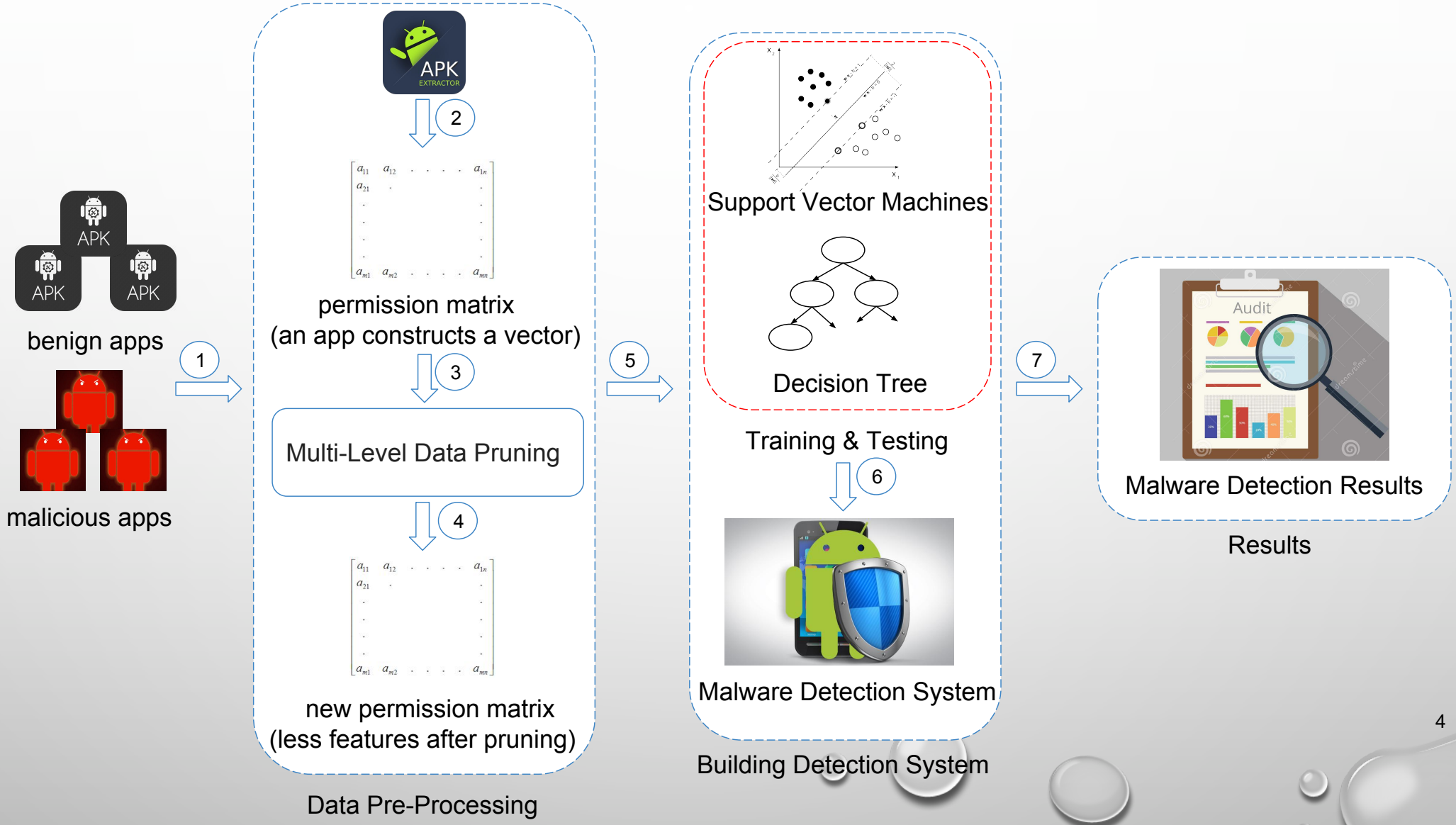
Growth of Android Malware

- Android allows to install applications from uncerti
- 97% of target A
- A new seconds

There is a need to create an effective and efficient malware detection system to cope with this rapid growth of malicious apps.



System Overview



Introducing SIGPID

- Multi-Level Data Pruning (MLDP)
- Malware Detection using Significant Permission
- Advanced MLDP with Fusion of Multiple Lists and X-value

Multi-Level Data Pruning (MLDP)

- Motivation: 135 permissions + huge number of applications = long processing time
- Three levels of data pruning

Balance Benign and Malicious Matrixes

- Two matrixes:
  - Matrix of original malware: M
  - Matrix of original benign apps: B
- Permission support formalization:

$$S_B(P_j) = \frac{\sum_i B_{ij}}{size(B_j)} * size(M_j)$$

Permission Ranking with Negative Rate (PRNR)

- No need to consider all 135 permissions
- Extract significant permissions:
  - Highly risky permission requested by malware
  - Rarely touched permission by malware
  - Remove permissions equally used by benign and malicious applications

Permission Ranking with Negative Rate (PRNR)

- $R(P_i) = [-1,1]$
  - -1 means non-risky permission
  - 1 means risky permission
  - 0 means lowest impact
- -1 to 0
- 0 to 1
- Near 0

$$R(P_j) = \frac{\sum_i M_{ij} - S_B(P_j)}{\sum_i M_{ij} + S_B(P_j)}$$

Permission Incremental System (PIS)

- Two sorted permission lists based on PRNR
- Choose the top permissions in benign and malware permission lists and evaluate malware detection
- Choose top three permissions in both lists and evaluate malware detection
- Repeat until f-measure becomes stable
- Remove 40 insignificant permissions from the total of 135 permissions

Remaining Permission : 135 – 40 = 95

Support Based Permission Ranking (SPR)

- Prune permissions with low impact
- Two policies
  - Use PIS to find the least number of permissions
  - Set a very small threshold of support
- Remove 70 more permissions

Remaining Permission: 135 – 40- 70 = 25

Permission Mining with Association Rules (PMAR)

- Some permissions are always used together
  - We can use the one with higher support to represent both
- Use Apriori with 95% minimum confidence and 3% minimum support
- Remove 3 additional permissions

Remaining Permission: 135 – 40 - 70 = 25

Evaluation

- Data Set
- MLDP Effectiveness
- Malware Detection Performance with Different Machine Learning Algorithms
- Comparison with Other Approaches

Evaluation Criterion

- Precision
- Recall
- F-Measure

		prediction	
		malicious	benign
actual	malicious	TP	FN
	benign	FP	TN
200 apps (100 malicious apps + 100 benign apps)			
		prediction	
		malicious	benign
actual	malicious	85	15
	benign	5	95
Precision = TP/(TP+FP)=94.4%			
Recall = TP/(TP+FN)=85%			
FM = 2*Precision*Recall/(Precision+Recall)= 89.7%			

Data Set

- 1,661 and 5,494 malicious applications
- 310,926 benign applications
- Extract permission information from the Android Manifest file of each app
- One vector represent an app with 1s and 0s, where 1 represents required permission and 0 otherwise

Multi-level Data Pruning Effectiveness

- Permission ranking with negative rate (PRNR) effectiveness
- Support Based Permission Ranking(SPR) effectiveness
- Permission mining with association rules (PMAR) effectiveness

Multi-level Data Pruning Effectiveness

Number of Permissions	Status	Precision	Recall(TPR)	FPR	F-measure	Accuracy
135	Original	98.81%	83.73%	1.01%	90.65%	91.36%
95	PRNR	96.39%	85.78%	3.22%	90.77%	91.28%
25	PRNR+PMAR	90.64%	91.77%	9.56%	91.17%	91.10%
22	PRNR+PMAR+SPR	91.55%	91.22%	8.54%	91.34%	91.34%

Malware Detection using Significant Permissions

- Implement 67 machine learning algorithms
- Compare 22 permissions with 135 permissions

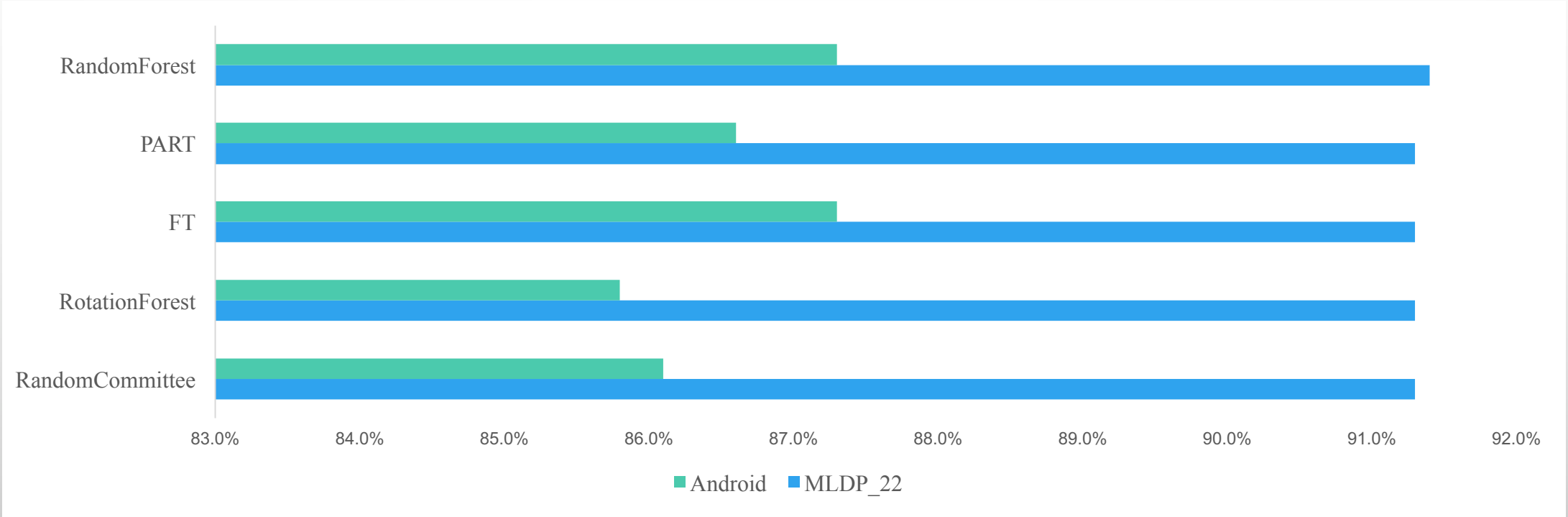
Performance of Machine Learning Algorithms

# of Permissions	22	40		135	
Name of Algorithm	Time(Seconds)	Time	More Time	Time	More Time
RandomCommittee	1.376	2.078	51.02%	7.995	481.03%
RotationForest	47.303	71.887	51.97%	236.944	400.91%
FT	0.731	2.14	192.75%	24.55	3258.41%
PART	16.673	24.645	47.81%	104.74	528.20%
RandomForest	14.028	20.045	42.89%	59.991	327.65%
SVM	2.4722	2.7604	11.66%	3.6773	48.75%

Optimal ML Algorithms For SigPID and Android Dangerous Permissions

#of Permissions	Best ML	Precision	Recall(TPR)	FPR	F-measure	Accuracy
SigPID (24)	FT	97.54%	93.62%	2.36%	95.54%	95.63%
Android (22)	Random Forest	98.61%	90.35%	1.27%	94.30%	94.54%

Detection Performance using Unknown Real-World Malware



Comparison with other approaches

Method	Recall
SigPID with FT	93.62
SigPID with SVMs	91.22
Mutual Information	86.4
Drebin	93.9
AV1	96.41
AV2	93.71
AV3	84.66
AV4	84.54
AV5	78.38
AV6	64.16
AV7	48.5
AV8	48.34
AV9	9.84
AV10	3.99

Future Work

- Enlarge dataset for malware and collect more features of the original dataset
- Develop a new machine learning algorithm
- Use additional information, such as that obtained through static program structure information (e.g., Static call graphs and calling context) and runtime information (e.g., Dynamic call graphs) to further classify behavior and pinpoint locations of malicious code

Conclusion

- We have developed a malware detection system based on permission
  - We are able to only consider a fraction of permissions to provide effective malware detection
  - Our approach performs as well as or better than techniques that consider more permissions or all permissions
  - By using significant permissions, we can improve performance a lot