SANER-45: The Detailed Description of the Dataset

I. Statistical Feature of DateSet

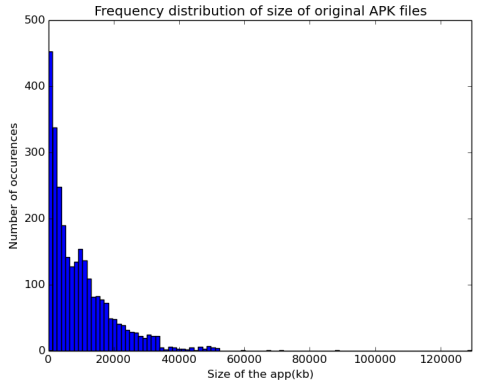
We decompile all the apps in our dataset. We extract all the opcode of apps and give a specific report on these apps. To gain a general understanding of the repackaged pairs, we thoroughly analyzed the dataset we collected from the Androzoo. The total number of the apps is 18,073. The number of original apps is 2,776 and the number of repackaged apps is 15,295. Therefore, there are 15,295 repackaged apps pairs.

1) size of apps: The largest size of the original app is 129.5MB, and its SHA256 prefix number is 81EC8. The smallest size of the original app is 51.6KB and its SHA256 prefix number is 3E45B. The largest size of the repackaged app is 91.5MB, and its SHA256 prefix number is 9F392D while the smallest size of the repackaged app is 53.5KB and its SHA256 number is 3D856. The specific situation we can see from the Table I.The original APKs and repackaged APKs have a median size of 9.72MB and 8.6MB, respectively. The total size of the original APK files is 26.97GB and the total size of the repackaged APK files is 131.6GB.

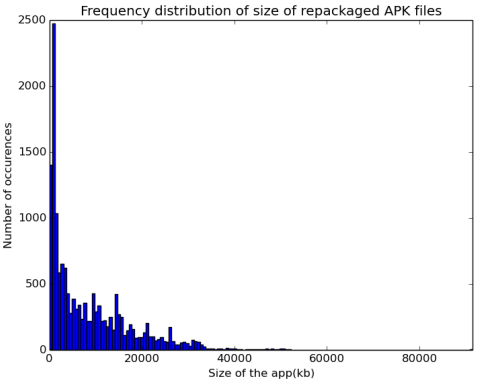
TABLE I
THE STATISTICAL DATA OF THE SIZE OF THE GROUND TRUTH APPS

The size of apps	original apps	repackaged apps
largest size	129.5MB	91.5MB
smallest size	51.6KB	53.5KB
average size	9.72MB	8.60MB
total size	26.97GB	131.8GB

Figure 1 shows some statistical characteristics. Figure 1 (a) and Figure 1(b) are the frequency distribution of size of the original and repackaged APK files, respectively. Both of them the range of x-axis is from the smallest size to the largest size of APK files. The number of bins is 100 in Figure 1(a) while the number of bins is 150 in Figure 1 (b). As can be seen from the Figure 1(a) and (b), the distribution is skewed to right. The size of most of original apks is smaller than 40MB. Besides, sizes of 80% original apps are smaller than 20MB. we also can find that the size range from 400KB to 1MB has the majority of apps. As shown in the Figure 1(b), the distribution is also skewed to right.The size of most of repackaged apks is smaller than 37MB. Besides, the sizes of 80% repackaged apps are smaller than 20MB.



(a) Frequency distribution of size of original apps



(b) Frequency distribution of size of repackaged apps

Fig. 1. Frequency distribution of ground truth apps' size

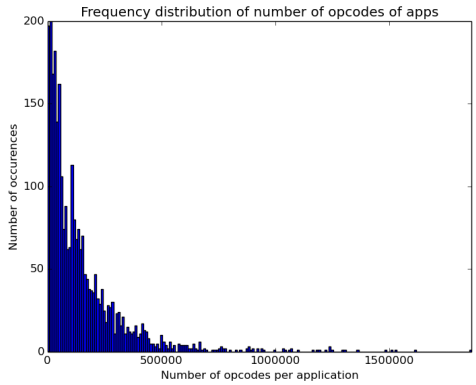
A. Opcode Statistics

The total number of opcode of original apps and repackaged apps are 0.416 billion and 1.87 billion. The average number of opcode of original apps and repackaged apps are about 150117 and 122265. The smallest and the largest number of opcode of original apps are 37 and 1,862,721 respectively. The smallest and the largest number of opcode of original apps are 891 and 1,869,051 respectively. The specific situation we can see from the Table II.

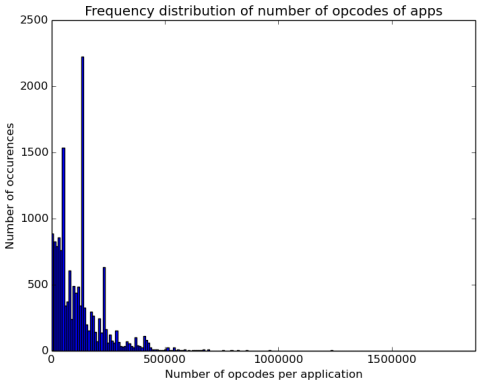
TABLE II
THE NUMBER OF OPCODE OF THE GROUND TRUTH APPS

The number of opcode	original apps	repackaged apps
largest	1,862,721	1,869,051
smallest	37	891
average size	150,117	122,265
total size	0.416 billion	1.87 billion

Figure 2 (c) and (d) show the distribution of the number of opcode per original and repackaged app, respectively. Both of the two distributions are skewed to the right too. At the same time, the number of the opcode of most of these apps is smaller than 1 million. the number of opcode of 80% apps is smaller than 500 thousand.



(a) Frequency distribution of number of opcode of original apps



(b) Frequency distribution of number of opcode of repackaged apps

Fig. 2. Frequency distribution of the number of opcode of ground truth app

Conclusion: According to our experimental result, we can find the size distribution of apps in our dataset is reasonable. The dataset contains different size of apps. Different size of apps contain a wide range of opcode.

II. PDGs Statistics

Figure3 illustrates the frequency distribution histogram of the average node number in each PDG. When we construct the PDGs for each app, we have filtered the third-party It can be seen from the Figurefig:nodeAVGPDG, the number of most PDGs in our data set range from 5 to 10 and only a few PDGs have more than 25 nodes, which means using the subgraph isomorphism algorithm(VF2) can get a preferable experiment performance.

Conclusion: According to our statistical results, we can find the apps in our dataset is reasonable. The average node number of PDGs obey normal distribution.

III. CFGs Statistics

We construct CFGs for apps in dataset and count the number of node in CFGs.

Result: The Figure 4(a) and (b) show the distribution of the number of the CFGs of original apps.The biggest difference

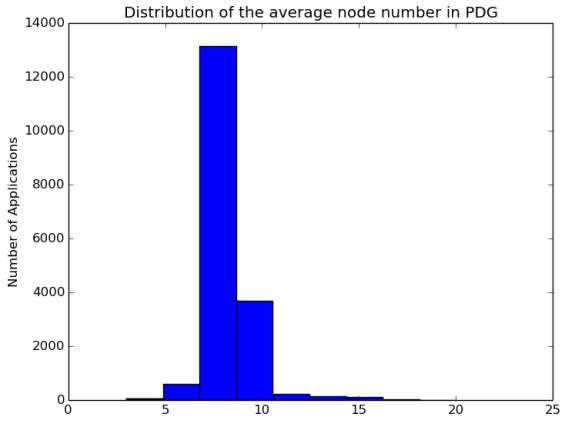


Fig. 3. Frequency distribution Histogram of the average node number of PDG.

between these two figures is that Figure 4(b) shows the histogram of original apps with no third-party libs. For better visualization, we choose the number of CFGs of apps is smaller than 4000 and 2000 to analyze. As we can see from the Figure 4(a), the range of x-axis is from the smallest number of CFGs to 4000. We set the number of bins is 40 and the size of each bin is 100 in Figure 4 (a) and (b). Most of the original apps the number of CFGs are smaller than 1500. As shown in the 4(b),the range of x-axis is from the smallest number to the max number 2000. The number of bins is 20 and the size of each bin is 100. We can find that most of the original apps the number of CFGs is smaller than 1000. The number of CFGs of original apps is smaller than 100 have the majority number of apps.

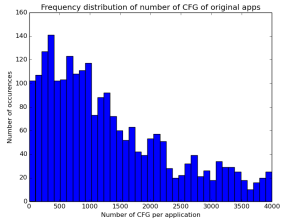
The Figure 4(c) and (d) show the distribution of the number of the CFGs of repackaged apps. Figure 4(d) shows the histogram of the number of the CFGs of the apps without third-party libraries. According to the Figure 4(c)the number of CFGs from 1300 to 1400 of repackaged apps has the largest percentage of the whole apps. The number of these apps is about 2500. The distribution of the rest applications is more uniform. According to the 4(d), the number of CFGs from 800 to 900 of repackaged apps account for the largest percentage of the whole apps. The number of these apps is about 2500. The distribution of the rest applications is more uniform. After deleting the third-party libraries methods from test apps,we total decrease 17,652,009 CFGs.

Conclusion: According to our statistical results, we can find the apps in our dataset is reasonable. The dataset contains different size of apps. The node of CFGs can reflect some advantages and disadvantages of different techniques.

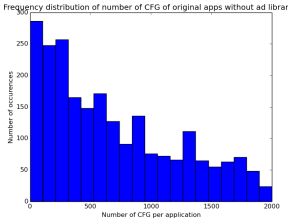
Activity Statistics

We use analysis to construct Activity Invocation Graph(AIG) for our dataset and count the number of node in AIGs.

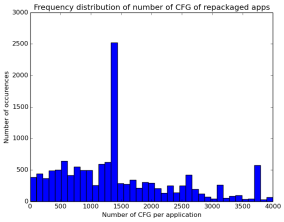
Result: Figure 5 shows the frequency distribution histogram of the number of activities nodes. The numbers on the x-



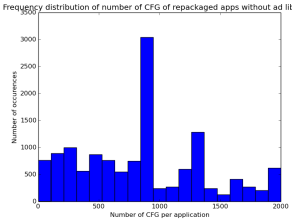
(a) Frequency distribution of number of CFGs of original apps



(b) Frequency distribution of number of CFGs of original applications without third-party libs



(c) Frequency distribution of number of CFGs of repackaged applications



(d) Frequency distribution of number of CFGs of repackaged applications without third-party libs

Fig. 4. Frequency distribution of number of CFGs of apps

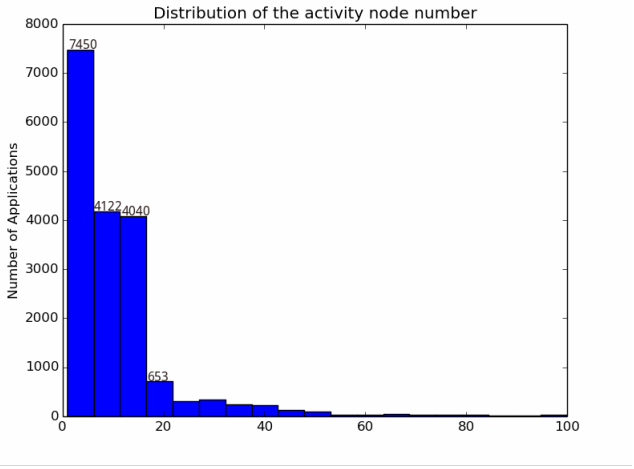


Fig. 5. Histogram of the number of activities nodes of dataset

axis are the lower bounds of the bins, and the size of each bin is 5. This distribution is skew to right, more than 93% of our test samples' number of activities nodes is less than 20. About 7,450 apps' the number of activities nodes is less than 5. About 8,200 apps' number of activities nodes is from 5 to 15. According to some researchers [1], VF2 subgraph isomorphism algorithm can handle pretty fast when the nodes number between 5 and 10. However, there is a serious limitation in view-based method; if the view graphs are small, it is easy to find (sub)graphs isomorphic to themselves. In our dataset, the view graphs with the number of nodes less than 2 are 2216. Besides, there are 407 view graphs with one node and 1809 view graphs with two nodes, respectively.

IV. Resource-based Technique

We found different data set have different top most-referenced resources. Therefore, how to choose the resources to classify repackaged app pairs from other apps is very significant.

RQ1: Are there any difference between the repackaged app pairs and totally independent app pairs cite the statistical features.

1) Answer to RQ1: To answer RQ1, we set two experiments through the following settings.

- **E1:** We extract the 15 statistical features from the real 15,295 repackaged app pairs and calculate their similar value.
- **E2:** We extract the 15 statistical features from 175 totally different apps and we conduct pair wise comparison for them and finally get 15,225 app pairs. We calculate the similar value for these unrelated app pairs.

We extract 15 resources features and use Euclidean distance to calculate the similarity of two apps. The results of this experiment can be seen from Table III.

TABLE III
Results of the similar value between repackaged app pairs and unrelated app pairs

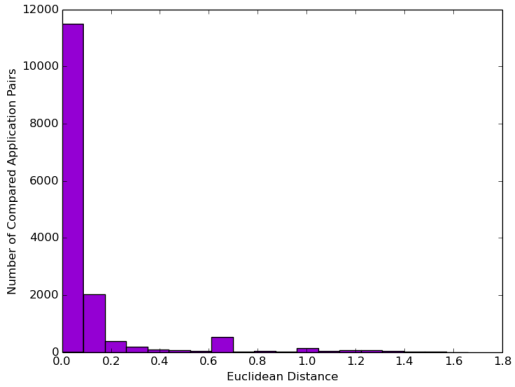
	repackaged pairs	unrelated pairs
the same	8,706	0
max distance	1.657	1.868
min distance	0	0.025
average	0.090	0.54
$< avg\%$	82.3%	63.1%

We can see from the Table III, the second and the third line stand for the maximum distance and minimum distance between two apps,respectively. The fourth line represents the average distance, and we can find that more than 82.3% of repackaged app pairs their similarity value of statistical features are smaller than average value and unrelated app pairs account for about 63.1%. There are 8,706 app pairs have the same 15 statistical features while no unrelated pairs have the same feature vector.

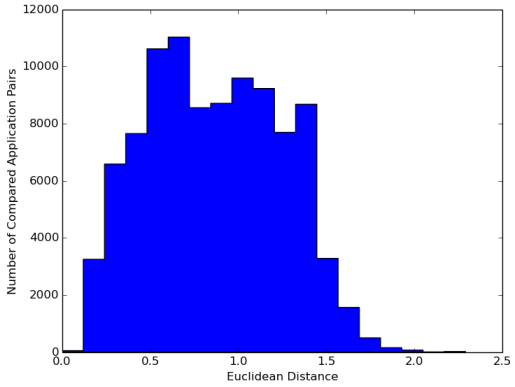
According to the Figure 6, we can clearly see the differences between the repackaged app pairs and unrelated app pairs. Figure 6(a) shows the frequency distribution of the Euclidean Distance between the original apps and repackaged apps. We can see from the Figure 6(a) over 90% repackaged app pairs their feature vectors have closer distance while the unrelated app pairs seldom have very similar feature vectors, their distances between the different app pairs are also various.

References

[1] X. Sun, Y. Zhongyang, Z. Xin, B. Mao, and L. Xie, "Detecting code reuse in android applications using component-based control flow graph," in *IFIP*, 2014.



(a) Repackaged APP Pairs



(b) Unrelated APP Pairs

Fig. 6. The difference between the repackaged app pairs' and unrelated app pairs' Histogram of *Euclidean Distance*