Adding semantic to level-up graph-based Android malware detection

Roxane Cohen, Florian Yger, Fabrice Rossi

March 9, 2023

LAMSADE UMR 7243

Quarkslab

Dauphine | PSL UNIVERSITÉ PARIS

Motivations

Cybersecurity application

Given an Android app, how to detect a hidden malware if any without executing the app ?

Answer

Static analysis ! Widely used in reverse-engineering. Many tools : Function Call Graph

Function Call Graph

Definition : Function Call Graph (FCG)

From a program P, one can extract its Function Call Graph, an oriented graph $G = (V, A)$, where vertices represent program functions and edges denote interprocedural calls.

```
1 int main(int argc, char*
      argv[]) {
2     A();
3     B();
4 }
5 void A(){ D(); }
6 void B(){ D(); }
7 void D(){}
```

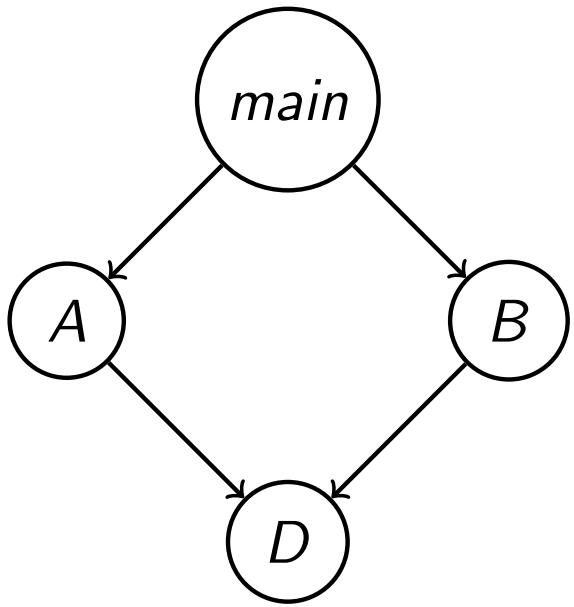


Figure: A Java program and its associated FCG

MalNet dataset

Obfuscation

Alterating a program code to make it hard to understand without modifying its behavior. Per function or on the whole program.

MalNet [1] :

- More than 1,200,000 Function Call Graphs extracted from Android apps
- 47 app types or 696 families
- Tiny version : 5,000 graphs evenly distributed over 5 types : adware, addisplay, trojan, downloader, benign.

[1] Scott Freitas and al. A Large-Scale Database for Graph Representation Learning. 2020

MalNet

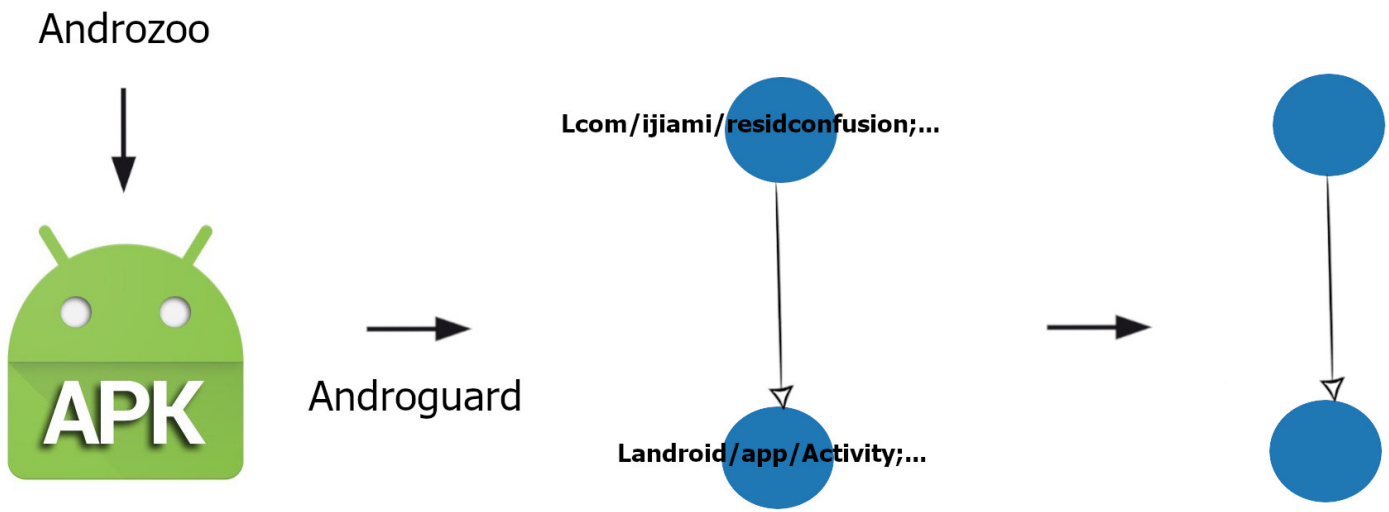


Figure: MalNet creation process

No semantic in MalNet

Node labels were deleted. They can help to classify Android apps. We decided to recreate MalNetTiny, but this time, by keeping semantic data.

Objectives

Use the two MalNet datasets to quantify how much semantic can help in a 5-class classification task.

- Original MalNetTiny : 5000 FCGs, without semantic labels, 5 classes
- Custom MalNetTiny : 4986 Attributed FCGs, with semantic labels, 5 classes.

Graph classification

Features

- MalNetTiny without labels
  - Graph feature extraction
  - Graphlet density vector extraction
- MalNetTiny with labels
  - PCA-based external call feature

Methods

- RandomForest
- SVM
- Graph Neural Networks (GIN, Sage, GCN)

Metric

Accuracy (balanced classes)

Graph features

37 structural graph features : number of nodes, edges, cyclomatic complexity, density...

Statistics	Nodes	Edges	CC
addisplay	1631 ± 1213	2920 ± 2373	1425 ± 1351
adware	2507 ± 1339	5597 ± 3012	3245 ± 1882
benign	2064 ± 1510	3887 ± 2921	2037 ± 1686
downloader	49 ± 5	55 ± 6	11 ± 1
trojan	800 ± 1313	1840 ± 3274	1084 ± 2058

Figure: Statistical elements per class about MalNetTiny graphs

Methodology

5x train/test split, 10-fold cross validation on training set

RandomForest : 0.87 ± 0.0078

Graphlet density feature vector[2]

Methodology

5x train/test split, 10-fold cross validation on training set

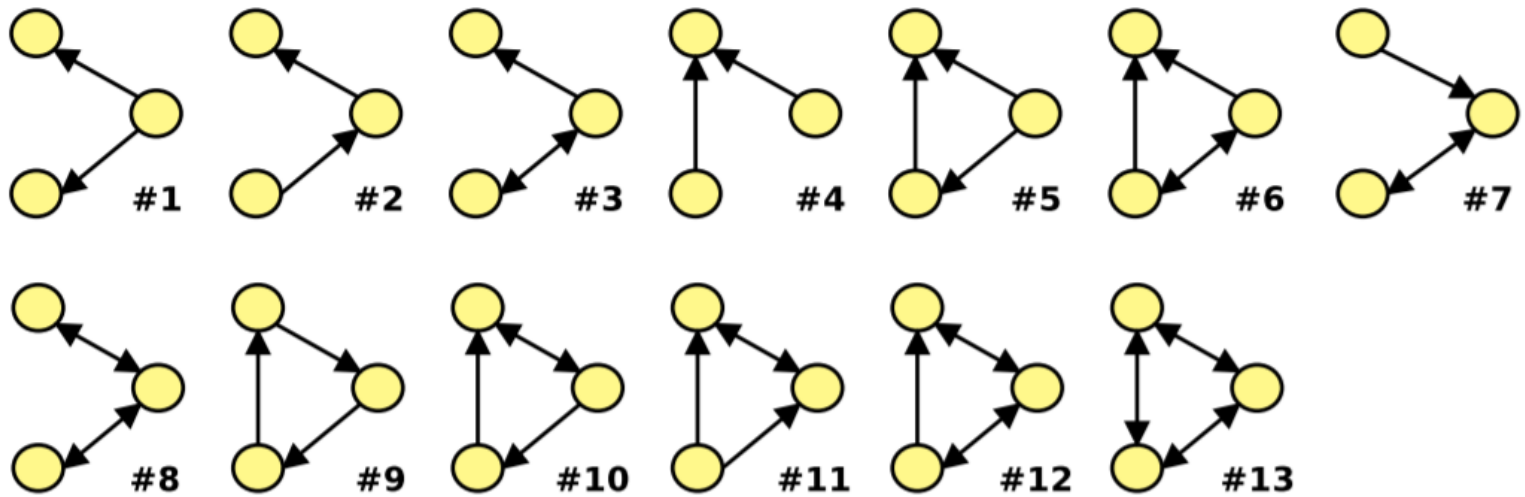


Figure: The 13 3-oriented graphlets

RandomForest : 0.8322 ± 0.012 d'accuracy

[2]Tianchong Gao and al. Android Malware Detection via Graphlet Sampling. IEEE Transactions on Mobile Computing: 18.12 (2019), pages 2754–2767.

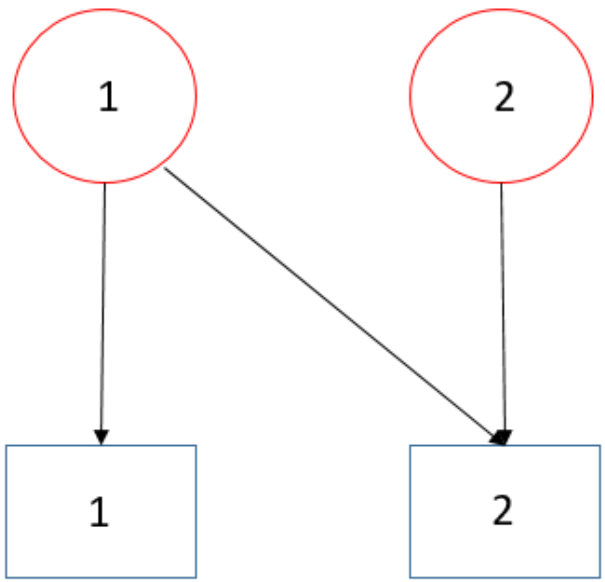
Adding semantic data ?

Motivations

Combining structure and semantic. Node : encoded (app node) or external

Labeled MalNet

1. More than 42 000 external classes
2. An encoded node can be associated to a counting vector of its calls to external classes



	1	2
1	1	1
2	0	1

PCA-reduced features

PCA reduction

1. $G \rightarrow$ matrix $(n_{encoded}, 42,000) \rightarrow$ PCA $\rightarrow (n_{encoded}, 32)$
2. Around 0.75 of explained variance.

Recall

Each graph must be associated to a single vector, regardless of the number of nodes per graph.

PCA-reduced features

Features	Accuracy
Mean	0.906 ± 0.0068
Mean + covariance	0.9214 ± 0.0067
Mean + covariance + graph features	0.9208 ± 0.0049
Max	0.9238 ± 0.0039
Max + mean	$\mathbf{0.9332} \pm 0.0076$

Table: Accuracy score comparison between different features based on RandomForest applied on PCA reduction of external calls.

Graph Neural Networks

1. Original MalNet
   1. Unoriented. Degree
   2. Oriented. In degree, out degree
2. Custom MalNet
   1. Oriented. In degree, out degree, encoded or external node, number of calls to external classes.
   2. Oriented. PCA on the counting matrix of calls to externals classes
   3. Oriented. DeepBinDiff [a]
   4. Oriented. PCA + DeepBinDiff

[a] Yue Duan and al. Deepbindiff: Learning program-wide code representations for binary diffing. 2020.

Features	Accuracy
GNN : 1.1	**0.3106** $\pm$ 0.1736
GNN : 1.2	**0.3976** $\pm$ 0.1598
GNN : 2.1	0.4657 $\pm$ 0.1471
GNN : 2.2	**0.8413** $\pm$ 0.0226
GNN : 2.3	0.7198 $\pm$ 0.0818
GNN : 2.4	0.7896 $\pm$ 0.0231

Table: Test accuracy comparison with a GIN-based GNN with different initial features.

Conclusion

Sum-up

- Robust and strong baselines
- Promising GNN results
- Best results on PCA-reduced features based on external class calls. Full autonomous procedure.
- Structure only is not sufficient. It is fundamental to use program semantic.

Perspectives

- Scalability
- Unstable results with GNN
- PCA reduction : choose parameters with cross-validation. Replace external class calls by methods. Better use of code in each function.

Thank you for your attention. Questions ?

Cyclomatic complexity

$$C = E - N + 2P$$

with C = cyclomatic complexity,
E = number of edges in the graph ;
N = number of nodes in the graph ;
P = number of components (weakly) connected in the graph.

Graph features

37 structural features :

- mean clustering number
- degree centrality (max)
- number of entry points
- mean degree
- density
- p-value of the out-degree powerlaw
- degree centrality (mean)
- number of nodes inside the large weakly connected component
- number of edges
- cyclomatic complexity
- degree assortativity
- alpha estimation of the degree-out powerlaw
- number of weakly connected components
- cutoff estimation of the degree-out powerlaw
- number of strongly connected components
- number of nodes

Graph features

- betweeness centrality (mean, max)
- degree centrality (min)
- alpha estimation of the in-degree powerlaw
- graph clique number
- number of attractive components
- mean length of the shortest path
- p-value of the in-degree powerlaw
- radius
- cycle number
- periphery
- center
- selfloop number
- diameter
- cutoff estimation of the in-degree powerlaw
- algebraic connectivity
- node connectivity
- betweeness centrality (min)

Graphlet density extraction

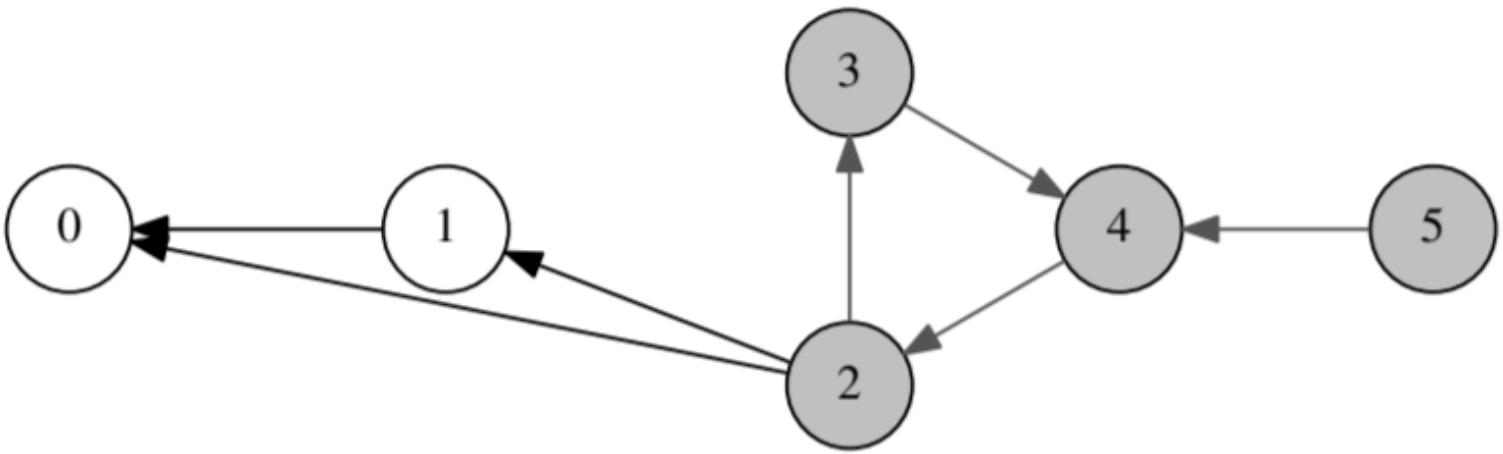


Figure: Graphlets example [1]

Obfuscation techniques

```
1  int main(int argc, char*
       argv[]) {
2      int a = 1;
3      a = D(a);
4      return a;
5  }
6  int D(int a){
7      a = a * 2;
8      return a;
9  }
```

```
1  int main(int argc, char*
       argv[]) {
2      int a = 1;
3      if 7*y**2-16= x**2{
4          a=D(a);
5      }
6      else{
7          a = A();
8      }
9      return a;
10 }
11 void A(){}
12 int D(int a){
13     a = a * 2;
14     return a;
15 }
```

Figure: A Java program and its obfuscated version with an opaque predicate that is always true, even with integer overflows