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

Roxane Cohen, Florian Yger, Fabrice Rossi

March 9, 2023
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.

```java
int main(int argc, char* argv[]) {
    A();
    B();
}

void A(){ D(); }

void B(){ D(); }

void D(){}
```

Figure: A Java program and its associated FCG
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.

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

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

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

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

![Diagram with nodes and arrows connecting them.](image)

```
1 | 1 1 2
2 | 0 1 1
```
## PCA-reduced features

### PCA reduction

1. $G \rightarrow \text{matrix } (n_{encoded}, 42, 000) \rightarrow \text{PCA} \rightarrow (n_{encoded}, 32)$
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

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

<table>
<thead>
<tr>
<th>Features</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mean</td>
<td>0.906 ± 0.0068</td>
</tr>
<tr>
<td>Mean + covariance</td>
<td>0.9214 ± 0.0067</td>
</tr>
<tr>
<td>Mean + covariance + graph features</td>
<td>0.9208 ± 0.0049</td>
</tr>
<tr>
<td>Max</td>
<td>0.9238 ± 0.0039</td>
</tr>
<tr>
<td>Max + mean</td>
<td><strong>0.9332 ± 0.0076</strong></td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Features</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>GNN : 1.1</td>
<td>0.3106 ± 0.1736</td>
</tr>
<tr>
<td>GNN : 1.2</td>
<td>0.3976 ± 0.1598</td>
</tr>
<tr>
<td>GNN : 2.1</td>
<td>0.4657 ± 0.1471</td>
</tr>
<tr>
<td>GNN : 2.2</td>
<td>0.8413 ± 0.0226</td>
</tr>
<tr>
<td>GNN : 2.3</td>
<td>0.7198 ± 0.0818</td>
</tr>
<tr>
<td>GNN : 2.4</td>
<td>0.7896 ± 0.0231</td>
</tr>
</tbody>
</table>

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

---

*a* Yue Duan and al. Deepbindiff: Learning program-wide code representations for binary differencing. 2020.
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

- betweenness 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
- betweenness centrality (min)
Graphlet density extraction

Figure: Graphlets example [1]
Figure: A Java program and its obfuscated version with an opaque predicate that is always true, even with integer overflows