Visualizing Android Feature Maliciousness through time

Michael Tegegn
Content

1. Introduction and Motivation
   a. Android
   b. Android Malware Detection

2. Visualizing Android Features over time
   a. Related Work
   b. Dataset and Abstraction
   c. Design Decisions
   d. Demo

3. Limitations and Future work
Android

- Most widely used Mobile OS

- 12,000 new Android malware instances every day. unb
Android Malware Detection

- Machine Learning Based Techniques perform well

Why visualize android features?

- Aid in Feature Selection
  - Eg. Selecting features more common in malware applications can help boost robustness [1]

Sample Selection: Which samples to train on?

- We want to identify future malware
- We can just train on all existing benign and malware samples

Why visualize android features through time?

- Identify features that can help detect future malware [2]

Related Work: Visualizations

1. Correlation Between Features

2. Model Performance through time
Data:

- Android Applications represented as a set of (DREBIN) features
  - Benign and Malware Android applications
    - Eg. app1: \{ feature1, feature3, feature5 \} label: benign
- Feature selection metrics and their results
  - Features ranked according the feature selection metrics
    - Eg. Mutual info metric: \{feature3, feature1, feature2\}

Derived Attributes:

\[
\text{Maliciousness of Feature } x = \frac{\text{Malicious Apps with } x - \text{Benign Apps with } x}{\text{Total Number of Apps with } x}
\]

\[
\text{Normalized freq of Feature } x = \frac{\text{Apps with Feature } x}{\text{Total Number of Apps}}
\]

- \(-1 \Rightarrow x \) only in malware apps
- \(1 \Rightarrow x \) only in benign apps
- \(1 \Rightarrow \text{All apps contain } x\)
- \(0 \Rightarrow \text{No apps contain } x\)
### Data Abstraction Summary

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Kind</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feature</td>
<td>Categorical</td>
</tr>
<tr>
<td>Feature Set / Feature Family</td>
<td>Categorical</td>
</tr>
<tr>
<td>App Development Year</td>
<td>Ordinal</td>
</tr>
<tr>
<td>Feature Maliciousness</td>
<td>Quantitative</td>
</tr>
<tr>
<td>Feature Normalized Frequency</td>
<td>Quantitative</td>
</tr>
</tbody>
</table>

**Goal:** Identify feature trends in android applications

**Target Group:** Malware detection tool developers
Visualization Design: Encodings

- Feature Maliciousness over time using heatmap
  - **y-axis:** Feature  **x-axis:** Development year
  - Maliciousness encoded by **Blue-Red Diverging Scale**

- Normalized frequency over time using bar charts
  - **y-axis:** Feature  **x-axis:** Development year
  - **Bar height:** Normalized frequency of a feature
Visualization Design: View Manipulation

- **Requirement 1**: Accommodate for large number of features
  - **Design Solution**
    - **Filter** features based on feature set
    - **Order** features based on popularity, maliciousness, ...
    - **Alter view** using scrolling

- **Requirement 2**: Show selected features
  - **Design solution**
    - Show selection using **dashed border lines**

- **Requirement 3**: Display normalized frequency of a feature on demand
  - **Design solution**
    - Add side view bar chart upon selection
http://127.0.0.1:5500/index.html
Limitations and Future Work

Limitations:

- Sampling bias in dataset
  - Eg. Drebin Dataset:
- Short time duration for the two datasets (VT: 4 years, DREBIN: 5 years)
- Still a large number of features (Scrolling required)

Future work:

- Hand select features and train models directly on the vis interface
- Extend for any domain that requires analysis of features for feature selection
Thank you!

Questions?