Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket

Daniel Arp, Michael Spreitzenbarth, Malte Hübner, Hugo Gascon, Konrad Rieck
Android-Malware

- Android-Malware
  - Rapid growth in the past few years
  - Mostly distributed through alternative markets

- Mobile Antivirus-Scanners
  - Signature-based detection
  - Unable to identify unknown malware samples
Drebin

- Detection of unknown malware samples
  - Analysis of known malware
  - Adaptive detection using machine learning techniques

- Detection directly on the smartphone
  - Apps can be installed from many different sources

- Technical Challenges
  - Limited resources of mobile devices
Drebin Learning Dataset Feature Extraction

Server

Feature Extraction → Embedding into Vector Space → Learning

Model Transmission

Client

Feature Extraction → Embedding into Vector Space → Classification → Explanation

Drebin
Drebin
Static Analysis

- Lightweight Analysis of Android Applications
  - Extraction of features (strings) from 8 different categories

- APK File
  - Manifest
    - App Components
    - Filtered Intents
    - Hardware Components
    - Requested Permissions
  - Dexcode
    - Protected API Calls
    - Used Permissions
    - Suspicious API Calls
    - Network Addresses

Drebin
Drebin

Server

Dataset

Feature Extraction ➔ Embedding into Vector Space ➔ Learning

Classification ➔ Explanation

Client

App

Feature Extraction ➔ Embedding into Vector Space

Model Transmission
Embedding in Vector Space

- Embedding of Apps into a vector space

- Vector representation of an App
  - Extracted features are set to 1
  - Small distance between Apps with similar characteristics

\[ \varphi(x) \rightarrow \begin{bmatrix} \ldots \\ 0 \\ 0 \\ \ldots \\ 1 \\ 0 \\ \ldots \end{bmatrix} \begin{align*} \text{hardware} &: \text{android.hardware.wifi} \\ \text{hardware} &: \text{android.hardware.telephony} \\ \text{permission} &: \text{SEND_SMS} \\ \text{permission} &: \text{DELETE_PACKAGES} \end{align*} \]
Drebin Learning Dataset Feature Extraction

Server Dataset

Feature Extraction → Embedding into Vector Space → Learning

Client App

Feature Extraction → Embedding into Vector Space → Classification → Explanation

Model Transmission
Dataset

- **Dataset**
  - Training and testing is done on large dataset
  - Collected by Mobile Sandbox project [5]
  - Consists of 123,453 benign and 5,560 malware samples

- **Malware Samples available at**
  - [http://user.cs.uni-goettingen.de/~darp/drebin/](http://user.cs.uni-goettingen.de/~darp/drebin/)
Learning

- Linear 2-Class Support Vector Machine
  - Hyperplane, which separates both classes with maximum margin
  - Can be described by model vector $w$
Drebin Learning Dataset Feature Extraction Server Dataset

Client App

Feature Extraction Embedding into Vector Space Learning

Feature Extraction Embedding into Vector Space Classification Explanation

Model Transmission
Classification

- Classification Score
  - Inner product of model and app vector
  - Sign indicates class of particular sample

\[ f(x) = \langle \varphi(x), \vec{w} \rangle \]
Classification

- Detector Calibration
  - FP-Rate should be less than 1%
  - Choice of threshold unequal to zero
Classification

- Detector Calibration
  - FP-Rate should be less than 1%
  - Choice of threshold unequal to zero
Detection Performance

![Detection Performance Chart]

- DREBIN: 94%
- Peng et al.: 47%
- RCP: 12%

Drebin
Detection Performance

![Detection Performance Graph](image)

<table>
<thead>
<tr>
<th>Id</th>
<th>Family</th>
<th>#</th>
<th>Id</th>
<th>Family</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>FakeInstaller</td>
<td>925</td>
<td>K</td>
<td>Adrd</td>
<td>91</td>
</tr>
<tr>
<td>B</td>
<td>DroidKungFu</td>
<td>667</td>
<td>L</td>
<td>DroidDream</td>
<td>81</td>
</tr>
<tr>
<td>C</td>
<td>Plankton</td>
<td>625</td>
<td>M</td>
<td>LinuxLotoor</td>
<td>70</td>
</tr>
<tr>
<td>D</td>
<td>Opfake</td>
<td>613</td>
<td>N</td>
<td>GoldDream</td>
<td>69</td>
</tr>
<tr>
<td>E</td>
<td>GingerMaster</td>
<td>339</td>
<td>O</td>
<td>MobileTx</td>
<td>69</td>
</tr>
<tr>
<td>F</td>
<td>BaseBridge</td>
<td>330</td>
<td>P</td>
<td>FakeRun</td>
<td>61</td>
</tr>
<tr>
<td>G</td>
<td>Iconosys</td>
<td>152</td>
<td>Q</td>
<td>SendPay</td>
<td>59</td>
</tr>
<tr>
<td>H</td>
<td>Kmin</td>
<td>147</td>
<td>R</td>
<td>Gappusin</td>
<td>58</td>
</tr>
<tr>
<td>I</td>
<td>FakeDoc</td>
<td>132</td>
<td>S</td>
<td>Imlog</td>
<td>43</td>
</tr>
<tr>
<td>J</td>
<td>Geinimi</td>
<td>92</td>
<td>T</td>
<td>SMSreg</td>
<td>41</td>
</tr>
</tbody>
</table>
Detection Performance

94%
Drebin Learning Dataset

Feature Extraction → Embedding into Vector Space → Learning

Model Transmission

Server

Dataset

Feature Extraction

Embedding into Vector Space

Learning

Client

App

Feature Extraction → Embedding into Vector Space → Classification → Explanation
Explainability

- Interpretation of Results
  - Insights into characteristics of malware
  - Analysis of false positives

- SVM assigns weight to each feature
  - Features with high weight → characteristic for class
  - Only consider features with high weights
  - Interpretation of malware characteristics
## Example: DroidKungFu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Set</th>
<th>Average Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG_STR</td>
<td>Filtered Intents</td>
<td>2.02</td>
</tr>
<tr>
<td>system/bin/su</td>
<td>Suspicious Calls</td>
<td>1.30</td>
</tr>
<tr>
<td>BATTERY_CHANGED_ACTION</td>
<td>Filtered Intents</td>
<td>1.26</td>
</tr>
<tr>
<td>READ_PHONE_STATE</td>
<td>Requested Permissions</td>
<td>0.54</td>
</tr>
<tr>
<td>getSubscriberId()</td>
<td>Suspicious Calls</td>
<td>0.49</td>
</tr>
</tbody>
</table>
### Example: DroidKungFu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Set</th>
<th>Average Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG_STR</td>
<td>Filtered Intents</td>
<td>2.02</td>
</tr>
<tr>
<td>system/bin/su</td>
<td>Suspicious Calls</td>
<td>1.30</td>
</tr>
<tr>
<td>BATTERY_CHANGED_ACTION</td>
<td>Filtered Intents</td>
<td>1.26</td>
</tr>
<tr>
<td>READ_PHONE_STATE</td>
<td>Requested Permissions</td>
<td>0.54</td>
</tr>
<tr>
<td>getSubscriberId()</td>
<td>Suspicious Calls</td>
<td>0.49</td>
</tr>
</tbody>
</table>
### Example: DroidKungFu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Set</th>
<th>Average Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG_STR</td>
<td>Filtered Intents</td>
<td>2.02</td>
</tr>
<tr>
<td>system/bin/su</td>
<td>Suspicious Calls</td>
<td>1.30</td>
</tr>
<tr>
<td>BATTERY_CHANGED_ACTION</td>
<td>Filtered Intents</td>
<td>1.26</td>
</tr>
</tbody>
</table>

1. Service is triggered by intent messages
### Example: DroidKungFu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Set</th>
<th>Average Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG_STR</td>
<td>Filtered Intents</td>
<td>2.02</td>
</tr>
<tr>
<td>system/bin/su</td>
<td>Suspicious Calls</td>
<td>1.30</td>
</tr>
<tr>
<td>BATTERY_CHANGED_ACTION</td>
<td>Filtered Intents</td>
<td>1.26</td>
</tr>
<tr>
<td>READ_PHONE_STATE</td>
<td>Requested Permissions</td>
<td>0.54</td>
</tr>
<tr>
<td>getSubscriberId()</td>
<td>Suspicious Calls</td>
<td>0.49</td>
</tr>
</tbody>
</table>

2. Malware tries to gain root access on the device
### Example: DroidKungFu

<table>
<thead>
<tr>
<th>Feature</th>
<th>Feature Set</th>
<th>Average Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIG_STR</td>
<td>Filtered Intents</td>
<td>2.02</td>
</tr>
<tr>
<td>system/bin/su</td>
<td>Suspicious Calls</td>
<td>1.30</td>
</tr>
<tr>
<td>BATTERY_CHANGED_ACTION</td>
<td>Filtered Intents</td>
<td>1.26</td>
</tr>
<tr>
<td>READ_PHONE_STATE</td>
<td>Requested Permissions</td>
<td>0.54</td>
</tr>
<tr>
<td>getSubscriberId()</td>
<td>Suspicious Calls</td>
<td>0.49</td>
</tr>
</tbody>
</table>

3. Malware steals sensitive data
Run-time Analysis

- Run-time evaluation using prototype implementation
  - Smartphones: Nexus 4, Galaxy S3, Xperia Mini Pro, Nexus i9250
  - Tablets: Nexus 7
Limitations

- Lack of Dynamic Analysis
  - Encryption of payload
  - Loading of malicious code during run-time

- Pollution Attacks
  - Poisoning of dataset by attacker
Conclusion

- Drebin allows reliable detection of Android malware
- Malware can be detected directly on the device
- Explanations are presented to the user
Thanks for your attention!

Questions?
References

[1] Dissecting Android malware: Characterization and evolution
   ▪ (Zhou and Jiang) (Oakland 2012)

   ▪ (Enck et al.) (USENIX 2011)

   ▪ (Peng et al.) (CCS 2012)

   ▪ (Sarma et al.) (SACMAT 2012)

   ▪ (Spreitzenbarth et al.) (SAC 2013)
Detection Performance

The detection performance for each malware family is shown in the figure. The x-axis represents the malware families, and the y-axis represents the detection rate. There are two conditions shown: 0 samples available and 10 samples available.

Key points:
- **Detection Rate**: The percentage of malware samples correctly identified.
- **Malware Families**: A to T are labeled on the x-axis.
- **Detection Performance**: The height of the bars indicates the detection rate.
- **0 Samples Available**: Indicates the detection rate with no available samples.
- **10 Samples Available**: Indicates the detection rate with 10 samples available.

Key takeaway:
- The detection rate varies significantly across different malware families.
- Families with more available samples generally show higher detection rates.

Experimental Procedure:
- The dataset contains 122,629 benign applications and 6,526 malware samples.
- The dataset is split into a known partition (66%) and an unknown partition (34%).
- The same dataset is used for training and testing.
- The experiment is repeated 10 times, and average results are reported.

Comparison with Related Approaches:
- DREBIN uses static approaches and related static techniques.
- The method by Peng et al. considers all permissions. The other methods consider only a subset of the requested permissions.
- The detection performance of DREBIN varies between 10%–50%.
- The false-positive rate of DREBIN is 1%.
- DREBIN is the only method that successfully detects 93% of the malware samples.

Experimental Results:
- The detection performance is illustrated in the figure for each malware family.
- The bar height represents the detection rate.
- Not all families can be reliably detected with limited samples.
- The detection rate for each family is indicated by the height of the bar.