Experiences of Landing Machine Learning onto Market-Scale Mobile Malware Detection

Liangyi Gong, Zhenhua Li, Feng Qian, Zifan Zhang, Qi Alfred Chen, Zhiyun Qian, Hao Lin, Yunhao Liu
Mobile Malware Detection

**Android App Markets**

- Fingerprint-based Antivirus Checking
- Expert-informed API inspection
- User-report-driven Manual Examination
- API-based Dynamic Analysis

"lend credibility"
Mobile Malware Detection

- Android App Markets
  - Amazon Appstore
  - GETJAR
  - Opera Mobile Store
  - Google Play
  - Huawei
  - Vivo
  - myapp.com
  - Oppo
  - MI

- "lend credibility"

- ML-based Mobile App Review Techniques
  - Fingerprint-based Antivirus Checking
  - Static Code Inspection
  - Dynamic Behavior Analysis
ML-based Detection at Market Scales

Widely explored in the past decade

Real-world Challenges?

No existing report of the effectiveness

ML-based Malware Detection

ML-based Solutions at Market Scales
Large-scale Dataset: API-centric, Dynamic

- **500K apps** submitted to Tencent Market
- From March to December 2017
- Containing apps’ malice labels

**App Emulation**

**Commodity servers**

**Tencent Market**
https://sj.qq.com/

**API Invocation Log**

**One-hot Feature Vector**
API Selection: Correlation

- APIs’ correlations with the malice of apps
  - Using SRC (Spearman’s rank correlation coefficient) to evaluate APIs’ correlation with apps’ malice
  - 260 APIs pose non-trivial correlation (|SRC| ≥ 0.2)
API Selection: Correlation

- APIs’ correlations with the malice of apps
  - Using SRC (Spearman’s rank correlation coefficient) to evaluate APIs’ correlation with apps’ malice
  - 260 APIs pose non-trivial correlation ($|SRC| \geq 0.2$)

- Time consumption of tracking different API sets
  - Fitting a tri-modal distribution
  - Indicating a complex relationship
    $$ t = \begin{cases} 
    a_1 \cdot n + b_1, & n \in [1, 800); \\
    a_2 \cdot n^{b_2}, & n \in [800, 1K]; \\
    a_3 \cdot \log(n) + b_3, & n > 1K. 
    \end{cases} $$

- Graph showing the time consumption for tracking different numbers of APIs.

- Graph showing the ranking of APIs based on SRC.
API Selection: Model & Accuracy

Machine Learning Model & Detection Accuracy

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>60.4%</td>
<td>59.6%</td>
<td>3.6 min</td>
</tr>
<tr>
<td>LR</td>
<td>81.2%</td>
<td>70.3%</td>
<td>10.4 min</td>
</tr>
<tr>
<td>SVM</td>
<td>87.9%</td>
<td>71.6%</td>
<td>~27K min</td>
</tr>
<tr>
<td>GBDT</td>
<td>88.4%</td>
<td>74.3%</td>
<td>364 min</td>
</tr>
<tr>
<td>kNN</td>
<td>86.5%</td>
<td>83.7%</td>
<td>~1.8K min</td>
</tr>
<tr>
<td>CART</td>
<td>87.6%</td>
<td>84.3%</td>
<td>11.6 min</td>
</tr>
<tr>
<td>ANN</td>
<td>90.8%</td>
<td>89.9%</td>
<td>~1.2K min</td>
</tr>
<tr>
<td>DNN</td>
<td>91.5%</td>
<td>90.9%</td>
<td>~1.9K min</td>
</tr>
<tr>
<td>Random Forest</td>
<td>91.6%</td>
<td>90.2%</td>
<td>29.1 min</td>
</tr>
</tbody>
</table>
### API Selection: Model & Accuracy

#### Machine Learning Model & Detection Accuracy

<table>
<thead>
<tr>
<th>Model</th>
<th>Precision</th>
<th>Recall</th>
<th>Training Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Naive Bayes</td>
<td>60.4%</td>
<td>59.6%</td>
<td>3.6 min</td>
</tr>
<tr>
<td>LR</td>
<td>81.2%</td>
<td>70.3%</td>
<td>10.4 min</td>
</tr>
<tr>
<td>SVM</td>
<td>87.9%</td>
<td>71.6%</td>
<td>~27K min</td>
</tr>
<tr>
<td>GBDT</td>
<td>88.4%</td>
<td>74.3%</td>
<td>364 min</td>
</tr>
<tr>
<td>kNN</td>
<td>86.5%</td>
<td>83.7%</td>
<td>~1.8K min</td>
</tr>
<tr>
<td>CART</td>
<td>87.6%</td>
<td>84.3%</td>
<td>11.6 min</td>
</tr>
<tr>
<td>ANN</td>
<td>90.8%</td>
<td>89.9%</td>
<td>~1.2K min</td>
</tr>
<tr>
<td>DNN</td>
<td>91.5%</td>
<td>90.9%</td>
<td>~1.9K min</td>
</tr>
<tr>
<td>Random Forest</td>
<td>91.6%</td>
<td>90.2%</td>
<td>29.1 min</td>
</tr>
</tbody>
</table>

Tracking top-490 correlated APIs achieves the highest precision/recall.

Tracking fewer APIs benefits both detection accuracy and speed!
Key API Selection Strategy

- Step 1. Selecting APIs with the highest correlation with malware (Set-C).
- Step 2. Selecting APIs that relate to restrictive permissions (Set-P).
- Step 3. Selecting APIs that perform sensitive operations (Set-S).
- Step 4. Combining the above.
Key API Selection Strategy

- **Step 1.** Selecting APIs with the highest correlation with malware (Set-C).
- **Step 2.** Selecting APIs that relate to restrictive permissions (Set-P).
- **Step 3.** Selecting APIs that perform sensitive operations (Set-S).
- **Step 4.** Combining the above.

**Performance:**
- **Analysis time:** 4.3 minutes
- **Precision/Recall:** 96.8% / 93.7%
- **Training time:** 14.4 seconds
Further Enriching the Feature Space

- Hidden features – API invocation hidden by certain techniques

  - Hidden and internal APIs triggered by special techniques like Java reflection
  - IPC through intents leveraging other apps/services to perform sensitive actions

- Checking Permissions
- Checking Used Intents

- Precision: 96.8%
- Recall: 93.7%
Further Enriching the Feature Space

- Hidden features – API invocation hidden by certain techniques

Hidden and internal APIs triggered by special techniques like Java reflection

- Key APIs alone
  - Precision: 96.8%
  - Recall: 93.7%

- Checking Permissions

IPC through intents leveraging other apps/services to perform sensitive actions

- API + Permission + Intents
  - Precision: 98.6%
  - Recall: 96.7%

- Checking Used Intents
System: Emulation Optimization

- Default Google Android Emulator: full-system emulation
- Result: 30% of apps require ≥5-minute analysis time
- Solution: lightweight emulation on powerful x86 server
- Architect: native x86 Android + Dynamic Binary Translation
System: Emulation Optimization

- **Configuration:** 5x4-core x86 server with CPU pinning
- **Compatibility:** ≤1% incompatible apps
- **Roll back to the Google Emulator for incompatible apps**
- **Performance:** saving around 70% of the detection time

Able to analyze an app in around 1.3 minutes
System: Real-world Deployment

Integration to Tencent Market

- Running since March 2018
- Checking ~10K apps per day using a single commodity server
- Over 98%/96% online precision/recall
System: Real-world Deployment

**Integration to Tencent Market**
- Running since March 2018
- Checking ~10K apps per day using a single commodity server
- Over 98%/96% online precision/recall

**System Evolution**
- Monthly updating the key APIs with the original dataset and newly submitted apps
- Fluctuating between 425 and 432
System: Addressing FPs & FNPs

○ False Positives

- 2% FP apps as complained by developers
- All using a few top-ranking APIs
- Most are quickly vetted based on previous versions

Manual Inspection: acceptable workload

Active & complete avoidance of FPs
**System: Addressing FPs & FNs**

### False Positives
- 2% FP apps as complained by developers
- All using a few top-ranking APIs
- Most are quickly vetted based on previous versions

**Manual Inspection:** acceptable workload

**Active & complete avoidance of FPs**

### False Negatives
- 4% FN apps reported by end users
- Hard to avoid
- Most (87%) barely use key APIs
- They have fairly simple functionalities, posing little threat

**Report-driven:** mild impact on users

**Passive mitigation of FNs**
Revealed Important Features

- Attempting to acquire privacy-sensitive information of user devices
- Tracking or intercepting system-level events
- Enabling certain types of attacks such as overlay-based attacks
Experiences of APICHECKER

Feature Selection
Principled, data-driven

Feature Engineering
Adversary’s perspective

Analysis Speed
Efficient app emulation on powerful x86 servers

Model Evolution
Monthly update with novel apps & SDK APIs

Developer Engagement
Active & complete avoidance of FPs vs. Passive mitigation ofFNs
Conclusion & Dataset

- We conduct a large-scale study to understand and overcome real-world challenges of developing ML-based malware detection solutions at market scales.

- We showcase several key design decisions we make towards implementing, deploying, and operating a production market-scale mobile malware detection system – APICHECKER.

- Our system has been operational at Tencent Market since March 2018, vetting around 10K apps per day on a single commodity server.

Dataset & tool release: https://apichecker.github.io/
Countering Emulator Detection

- **Strategies:**
  - changing the default configurations of emulators
  - tuning the execution parameters of Monkey
  - replaying traces of sensor data collected from real devices
  - obfuscating the existence of Xposed

- **Experiment on real devices, original and enhanced emulator:**
  - original emulator: 86.6% apps invoke the same amount of APIs
  - enhanced emulator: 98.6% apps invoke the same amount of APIs
Comparison with Other Work

Differences:

- The scale of studied apps is much larger
- Innovations in API selection, identifying hidden features
- Optimization in dynamic emulation infrastructure
- Commercial deployment result & online model evolution

<table>
<thead>
<tr>
<th>API Selection Strategy</th>
<th>Related Work</th>
<th>Analysis Method</th>
<th>Analysis Time per App</th>
<th># APIs Used</th>
<th># Apps Studied</th>
<th>Precision, Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>Statistical Correlations</td>
<td>Sharma et al. [35]</td>
<td>static</td>
<td>--</td>
<td>35</td>
<td>1,600</td>
<td>91.2%, 97.5%</td>
</tr>
<tr>
<td></td>
<td>DroidAPIMiner [1]</td>
<td>static</td>
<td>25 sec</td>
<td>169</td>
<td>~20K</td>
<td>--</td>
</tr>
<tr>
<td>Restrictive Permissions</td>
<td>Stowaway [15]</td>
<td>static</td>
<td>--</td>
<td>1,259</td>
<td>964</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>DroidMat [43]</td>
<td>static</td>
<td>--</td>
<td>--</td>
<td>1,738</td>
<td>96.7%, 87.4%</td>
</tr>
<tr>
<td></td>
<td>Yang et al. [46]</td>
<td>dynamic</td>
<td>1080 sec</td>
<td>19</td>
<td>~27K</td>
<td>92.8%, 84.9%</td>
</tr>
<tr>
<td></td>
<td>DroidCat [9]</td>
<td>semi-dynamic</td>
<td>354 sec</td>
<td>27</td>
<td>~34K</td>
<td>97.5%, 97.3%</td>
</tr>
<tr>
<td></td>
<td>IntelliDroid [42]</td>
<td>static + dynamic</td>
<td>138.4 sec</td>
<td>228</td>
<td>2,326</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>Droid-Sec [49]</td>
<td>static + dynamic</td>
<td>--</td>
<td>64</td>
<td>250</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>DroidDolphin [44]</td>
<td>dynamic</td>
<td>1020 sec</td>
<td>25</td>
<td>64K</td>
<td>90%, 82%</td>
</tr>
<tr>
<td>Hybrid</td>
<td>DREBIN [6]</td>
<td>static</td>
<td>10 sec</td>
<td>--</td>
<td>~128K</td>
<td>--</td>
</tr>
<tr>
<td></td>
<td>APICHECKER</td>
<td>dynamic</td>
<td>78 sec</td>
<td>426</td>
<td>~500K</td>
<td>98.6%, 96.7%</td>
</tr>
</tbody>
</table>
UI Exploration & Coverage

- Activity Coverage: pessimistic, only 88% of defined activities are actually referred in source code
- New metric: Referred Activity Coverage (RAC)
- Tradeoff: 5K vs. 100K Monkey Events, sacrificing a small fraction (9.5%) of RAC to largely reduce (94%) of the emulation time

![Graph showing RAC and Emulation Time vs. Number of Monkey Events (K)]
A Smaller API set?

- API selection can affect both the detection accuracy and speed.
- Most of key APIs slightly affect accuracy, greatly impacts speed.
- Tracking top-150 vs. Tracking top-426:
  - Precision/Recall: 98.3%/96.6% vs. 98.6%/96.7%
  - Analysis Time: 2.5 m vs. 4.3 m (without efficient emulation)
Integration to Other Markets

- Expected to be an easy process
- Implementation: mature analysis tool chain + machine learning
- Training: APKs + ground-truth data
- Possible for large markets to distribute pre-trained models
Robustness of APICHECKER

- Our key API set: 426 APIs, 0.85% of the 50K APIs in SDK
- 4,816 APIs depend on the key APIs, a total of 5,242 (10.5%) APIs
- Reimplementing all the APIs: high technical bar
- Possible workaround – NDK: high usage is also an indicator
Online Evaluation & Evolution

**Evaluation:**
- based on other components in T-Market’s app review process
- ≥4 SOTA fingerprint-based antivirus checking (all claim ≤5% FP)
- expert-informed API inspection
- user-report-driven manual examination

**Evolution:**
- dataset: original dataset & newly submitted apps
- labels: flagged by both APICHECKER and manual inspection