Correlating UI Contexts with Sensitive API Calls: Dynamic Semantic Extraction And Analysis

Jie Liu, **Dongjie He**, Diyu Wu and Jingling Xue

Problem: Is the Sensitive API Invocation legal?

Existing Solution I: app-level techniques

Apps Legitimacy

Sensitive API Invocation,
e.g., sendTextMessage

Message Apps

✔

Non-messaging Apps

✖

Could not distinguish API invocation with its calling context

Suffer from many false positive or many false negative.

Existing Solution II: API-level techniques

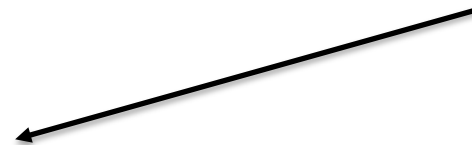**Sensitive API Invocation,**
e.g., sendTextMessage

static analysis →

UI components,
e.g., text, …

Dynamic features, e.g.,
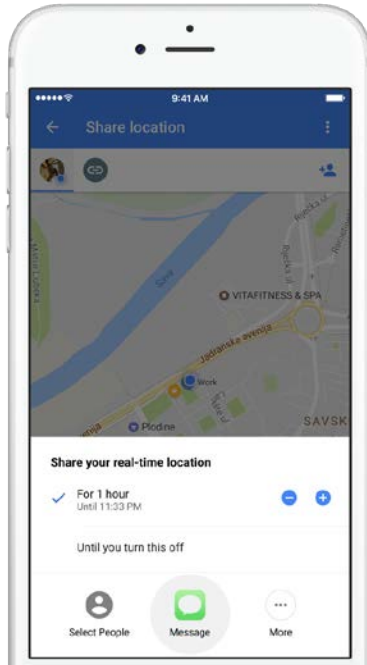reflection, dynamic class loading

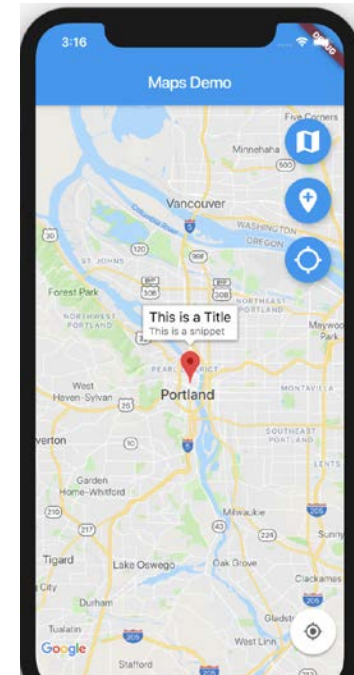Attributes, e.g., text, could be null
or statically unknown.

- ICSE 14, USENIX Security 18

Our Key Insight: Correlating UI Contexts with Sensitive API Calls dynamically.
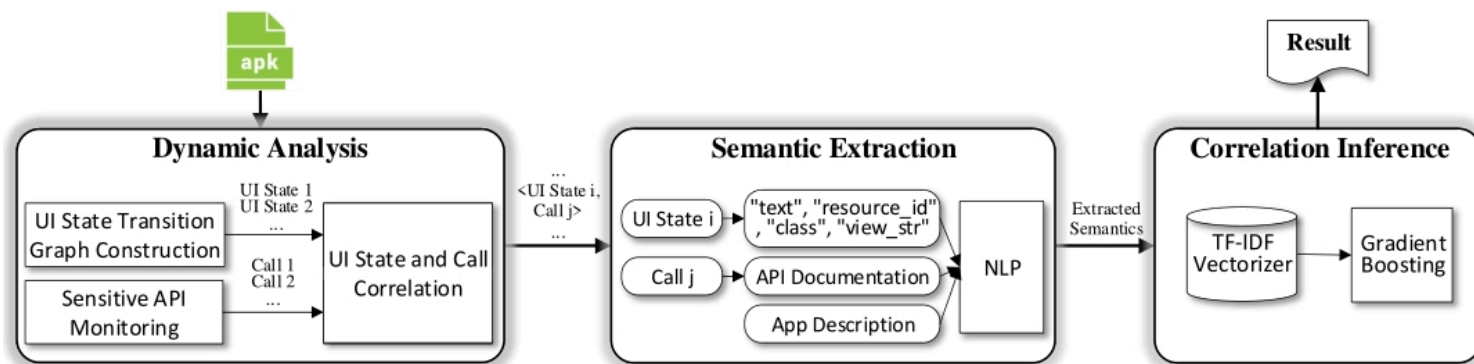
Our Approach: APICOG Overview

- **Dynamic Analysis:**
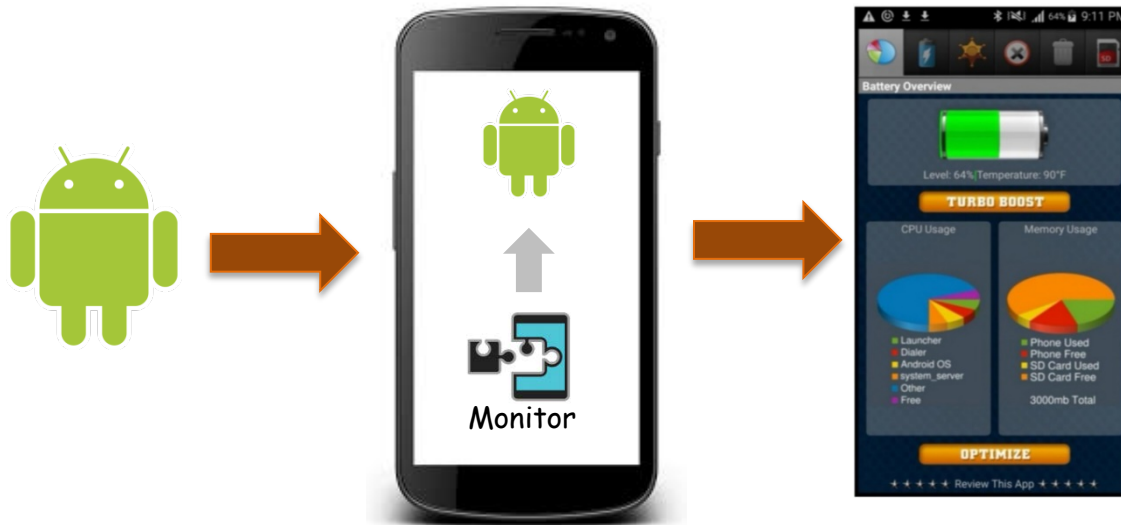 - Associate sensitive API call with its related top Activity
- **Semantic Extraction:**
  - Extract Activity semantics from saved screenshot, UI layout and App description.
  - Extract API semantics from API documentation.
- **Correlation Inference:**
  - Determine if semantics provides enough information to justify the legitimacy of the usage.

Our Approach: Dynamic Analysis

<BoosterActivity, LocationManager getLastKnownLocation()>

Snippet from BoosterActivity:onClick

```
19   String service = CipherUtil.decrypt("bG9jYXRp");// service = "location";
20   Object lm = ctx.getSystemService(service); // get LocationManager object
21   // mtdName = "getLastKnownLocation";
22   String mtdName = CipherUtil.decrypt("WpMUG6kCL/VztBsv");
23   Method mtd = lm.getClass().getMethod(mtdName, String.class);
24   // get the current location
25   Object location = mtd.invoke(lm, provider);
```

Our Approach: Semantics Extraction

Activity information

API description

App description

NLP

(boost, battery),
(null, memory),
…
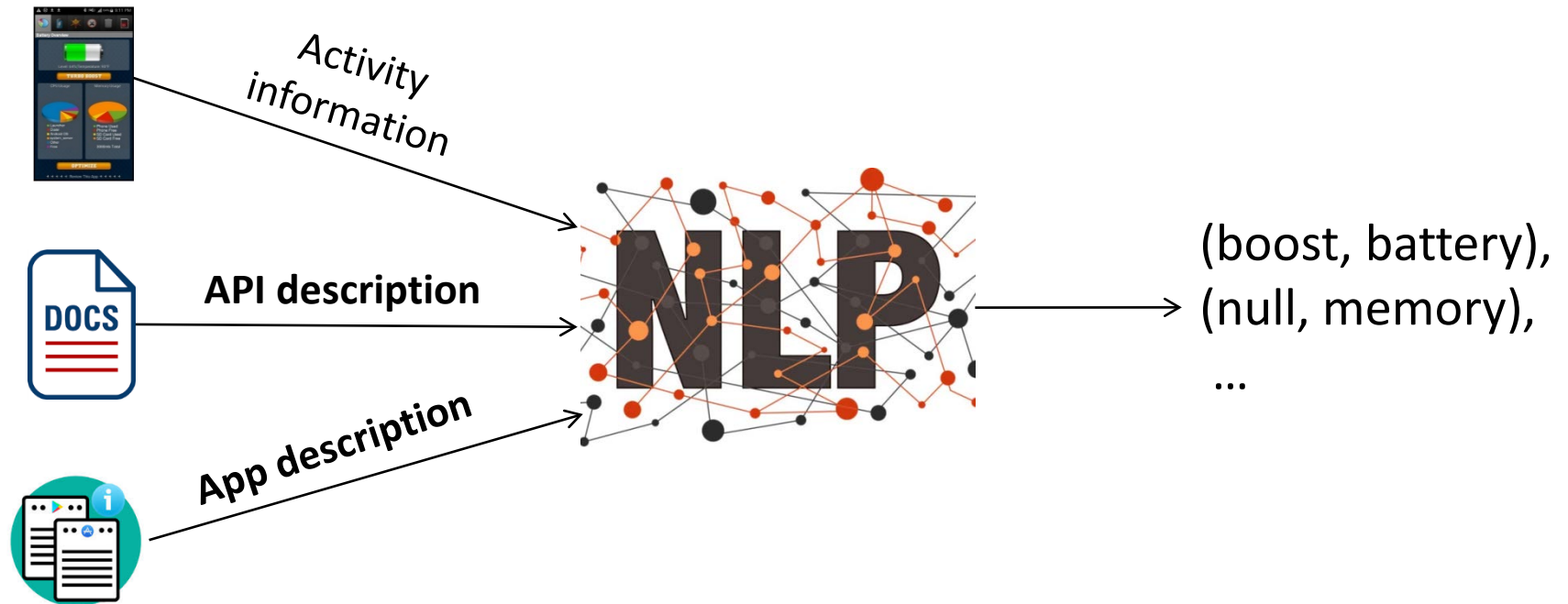
Our Approach: Correlation Inference

callsite:

getLastKnownLocation

action resources:

(boost, battery),
(null, memory),
…

Classifier

TF-IDF Vectorizer

Gradient Boosting

√ ✕
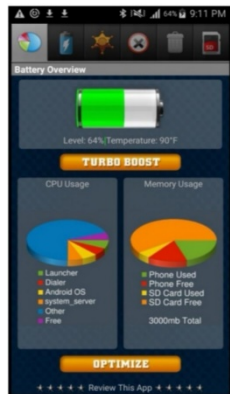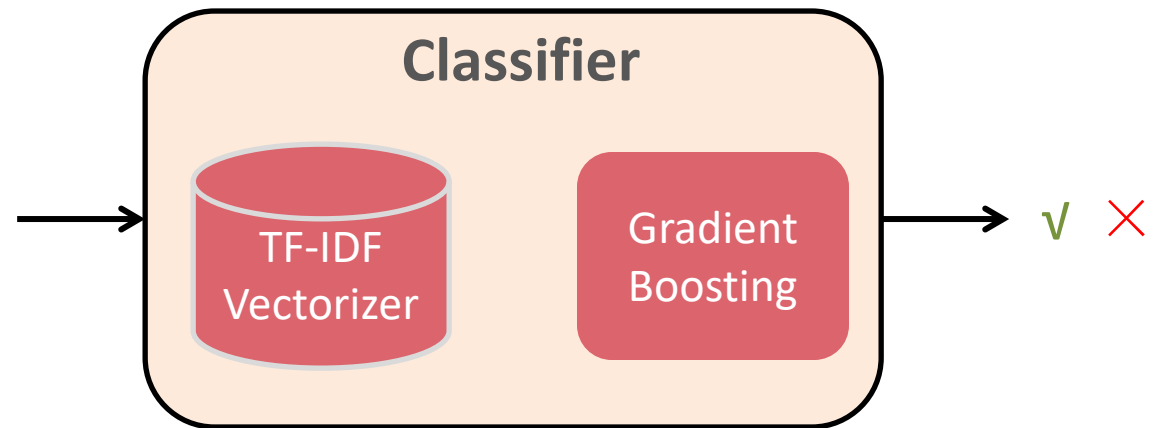
Evaluation

- 1500 malware apps from Drebin dataset [1]

- All 1625 benign apps from F-Droid[2]

Manually-annotated Ground Truth

App Type	# of Apps	# of Apps with Activity-callsite Pairs	# of Total Activity-callsite Pairs	# of Positive Activity-callsite Pairs	# of Negative Activity-callsite Pairs
Benign	1625	251	805	696	109
Malware	1500	725	4294	191	4103
Total	3125	976	5099	887	4212

Half pairs for training and half for testing.

[1] D. Arp, M. Spreitzenbarth, M. Hubner, H. Gascon, K. Rieck, and C. Siemens, "Drebin: Effective and explainable detection of android malware in your pocket." in NDSS, 2014.
[2] F-Droid. *Free and Open Source App Repository*. https://f-droid.org/. 2019.

RQ1: Precision, Recall and Accuracy

Overall Performance of APICog against the Ground Truth

App Type	# of Testing Activity-callsite Pairs	TP	FN	FP	TN	Accuracy	Precision	Recall
Benign	410	339	12	15	44	93.4%	95.8%	96.6%
Malware	2140	72	20	11	2037	98.6%	86.7%	78.3%
Total	2550	411	32	26	2081	97.7%	94.1%	92.8%

RQ2: Classification Approaches
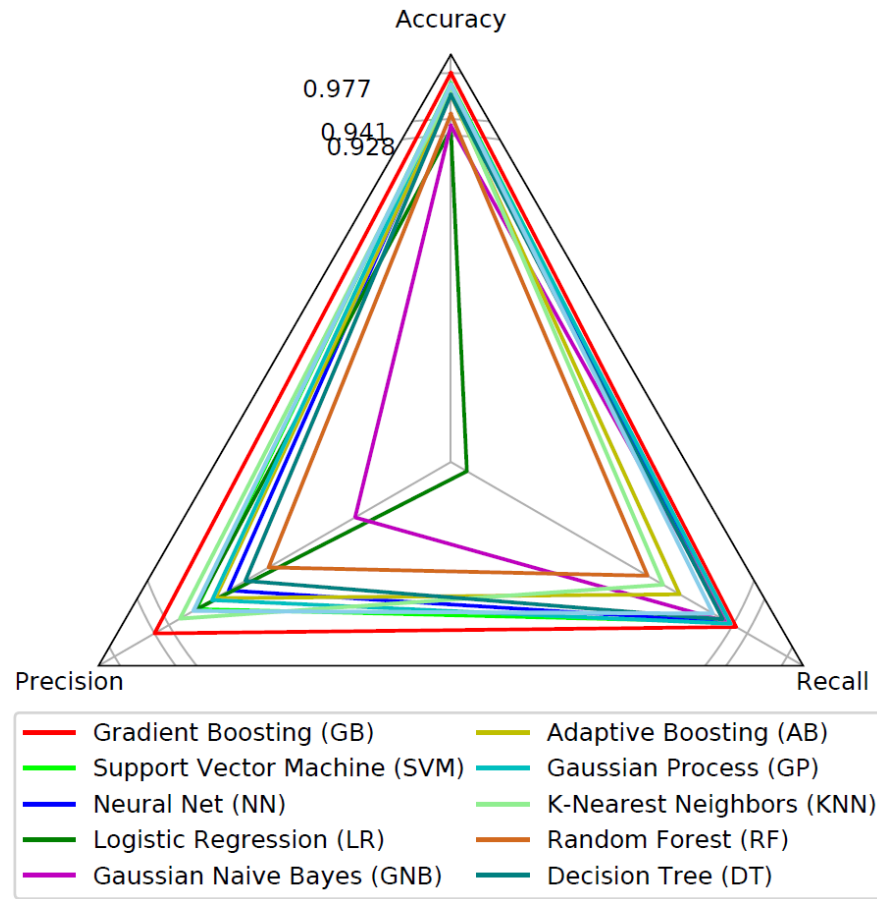
RQ3:Effectiveness of Different Semantics

Compare of APICOG with different Semantics:

- $APICOG_{des}$ = API semantics + App description semantics.
- $APICOG_{asd}$ = API semantics + UI state semantics.
- APICOG = API semantics + App description semantics + UI state semantics.

Tool	Accuracy	Precision	Recall	F1 Score
$APICOG_{des}$	94.8%	91.4%	77.2%	83.7%
$APICOG_{asd}$	95.7%	91.1%	83.5%	87.2%
APICOG	97.7%	94.1%	92.8%	93.4%

Our Contributions

- First dynamic description-to-permission fidelity approach for Android.

- Open-sourced tool:

    - http://www.cse.unsw.edu.au/~corg/apicog/

THANK YOU.