Semantics-Aware Android Malware Classification Using Weighted Contextual API Dependency Graphs

Mu Zhang

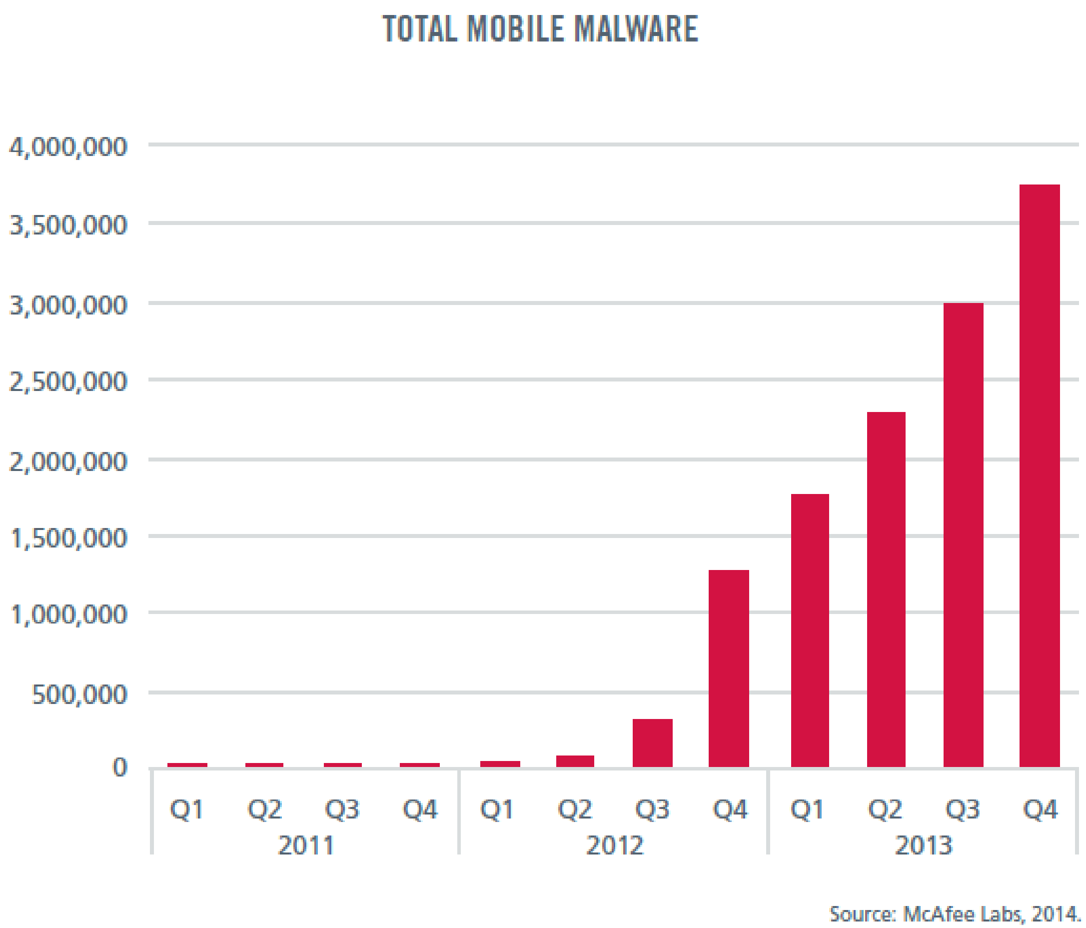
Yue Duan

Heng Yin

Zhiruo Zhao

Department of Electrical Engineering and Computer Science,
Syracuse University

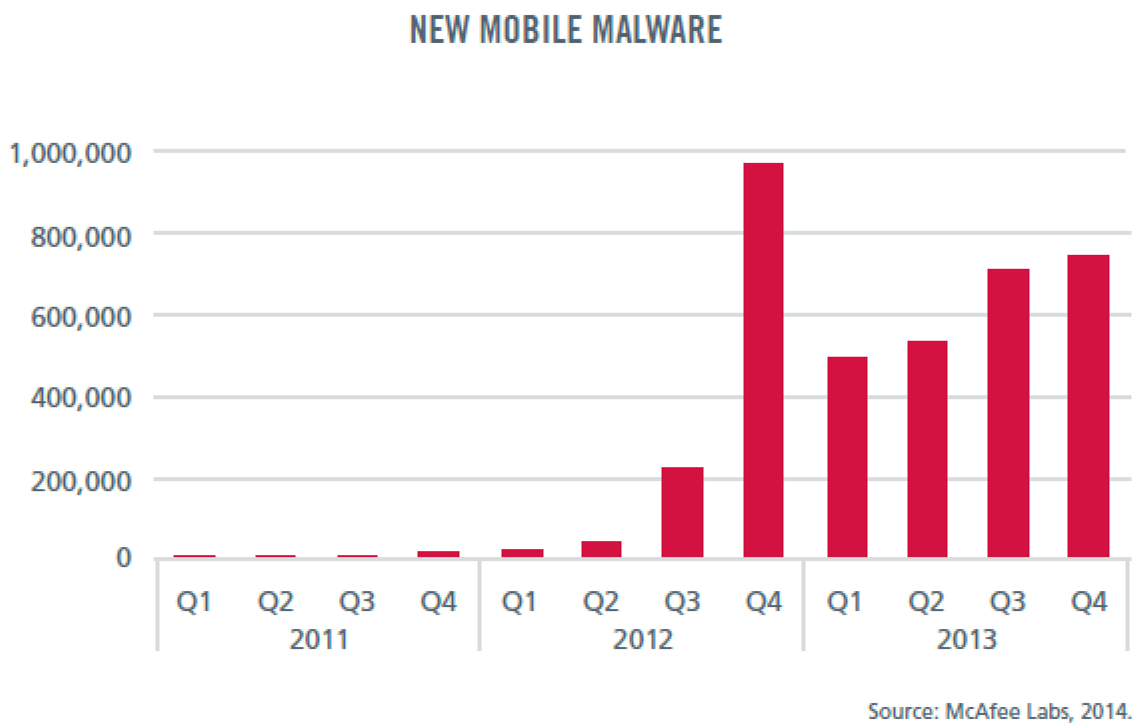
Android Malware Detection: Need For Speed



McAfee Threat Report:

Totaled **3.73 million** samples at the end of 2013, a **197% increase** over 2012

Malware Variants and Zero-day Malware



McAfee Threat Report:

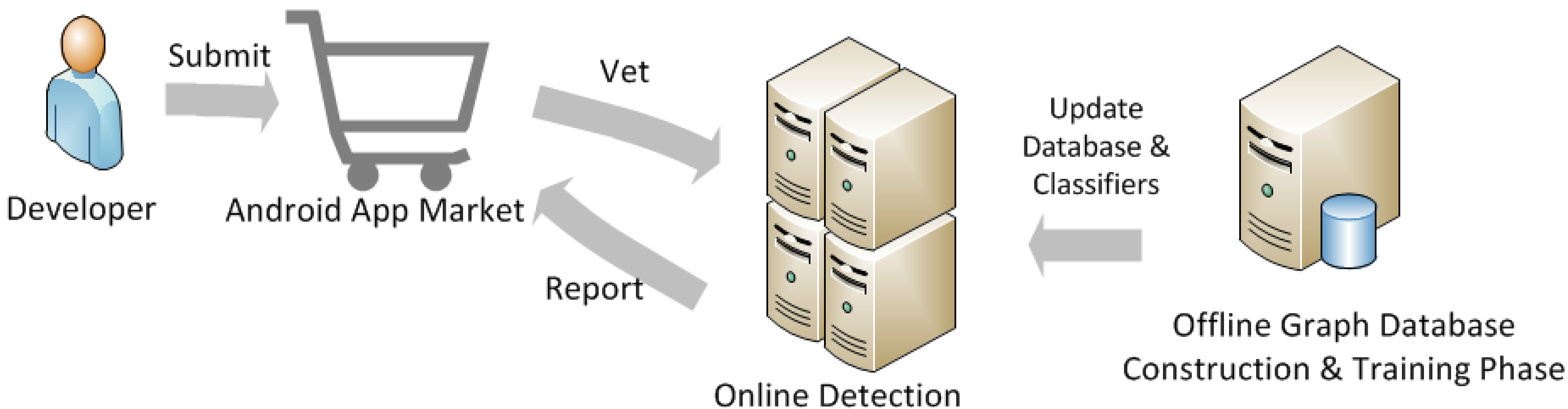
2.47 million new mobile malware samples were collected in 2013

Motivation: Existing Techniques have Limitations

- **Code Pattern-based**
  - Riskranker [MobiSys'12], DroidRanger [NDSS'12], Antivirus Software, etc.
  - Rely on ***code patterns***
  - Evaded by transformation attacks (DroidChameleon [TIFS'14, ASIACCS'13])
- **Machine Learning-based**
  - DroidMiner [ESORICS'14], Drebin [NDSS'14], DroidAPIMiner [SecureComm'13], Peng et al. [CCS'12], etc.
  - Rely on ***application syntax*** rather than ***program semantics***
  - Susceptible to evasion

DroidSIFT: Semantics-Aware Malware Classification

- **Deployment**
  - Complement to Bouncer
  - Signature detection: new variants
  - Anomaly detection: zero-day
- **Design Goals**
  - Semantic-based Detection
  - High Scalability
  - Variant Resiliency

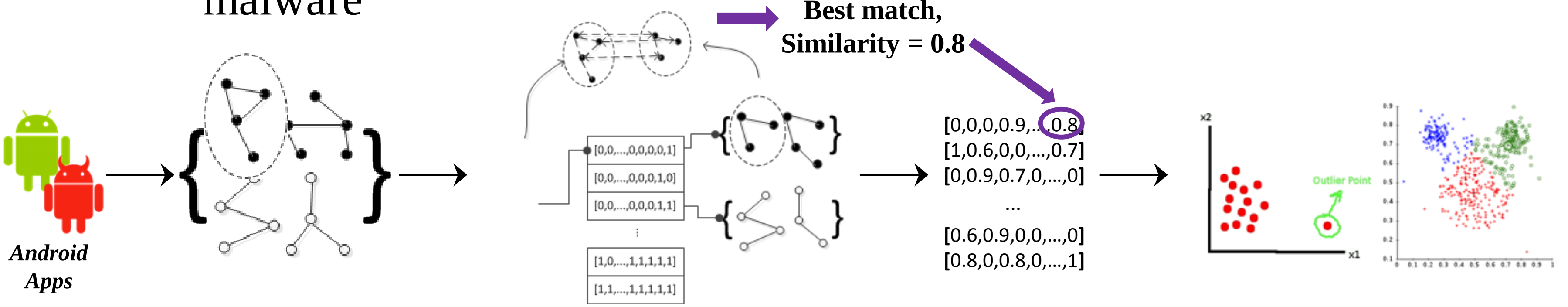


Related Work: Semantic-based Malware Detection

- **Semantic-based Approaches**
  - Control-flow Graph: M. Christodorescu et al. [Oakland'05]
  - Data Dependency Graph: M. Fredrikson et al. [Oakland'10], C. Kolbitsch et al. [Usenix Security'09]
  - Permission Event Graph: K. Z. Chen et al. [NDSS'13]
- **Limitations**
  - Manually crafted specifications
  - Specifications are produced from known malware
  - To pursue exact matches

Approach Overview

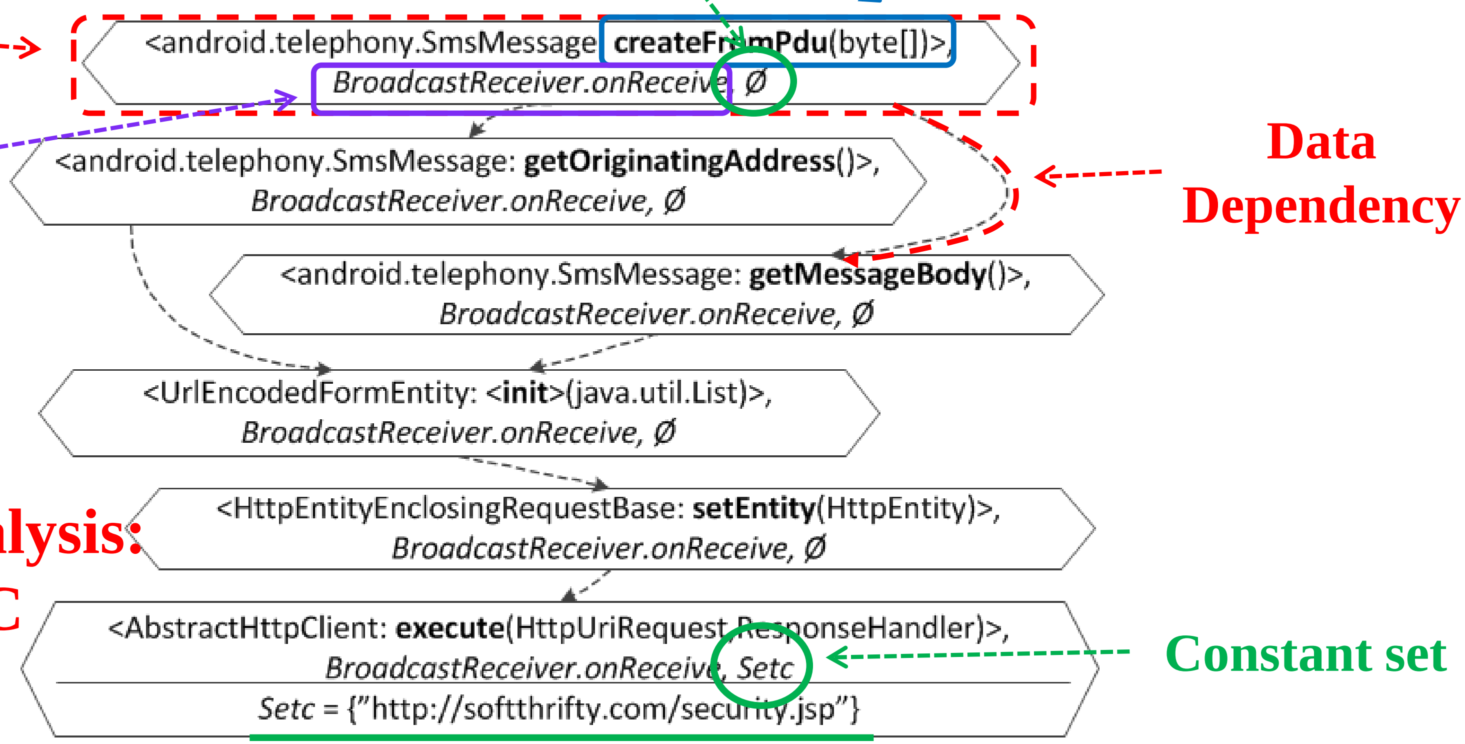
- **DroidSIFT**
  - ***Contextual API Dependency Graphs***, automatically and statically extracted “specifications”
  - ***Weighted Graph Similarity***, to address malware variants & zero-day malware



Behavior Graph Generation | **Matching-based Graph Query** | **Similarity-based Feature Vector Extraction** | **Classification-based Anomaly & Signature Detection**

Weighted Contextual API Dependency Graph

Malware sample: Zitmo

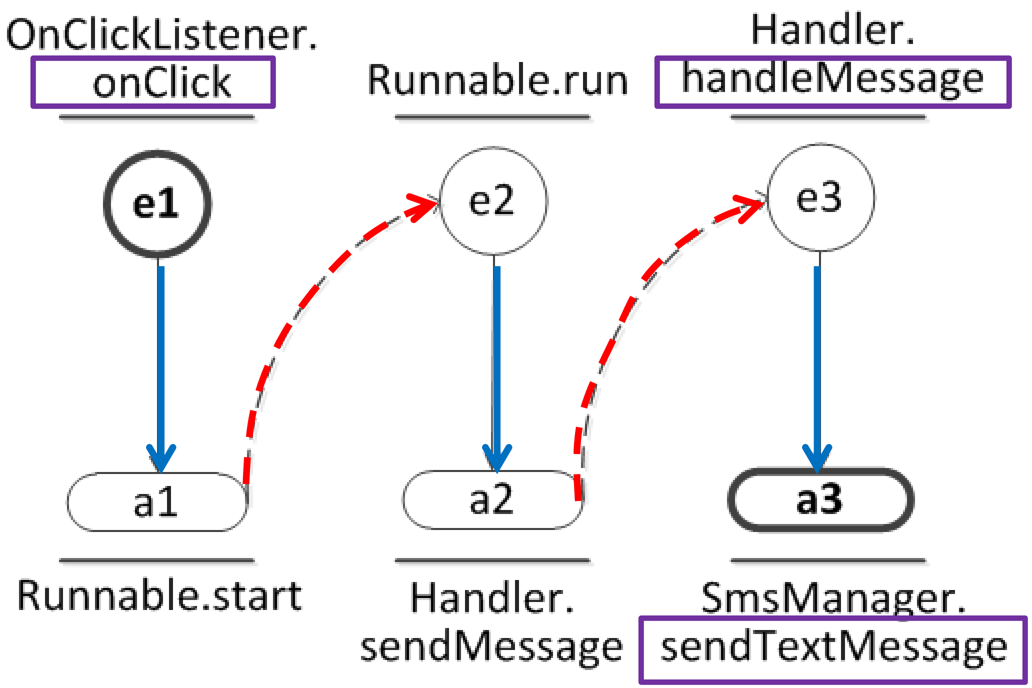


Dataflow Analysis: 20K LOC

Weights are assigned to API nodes, giving greater weights to the nodes containing critical calls

Weighted Contextual API Dependency Graph

- **Context (Entry Point) Discovery**



Algorithm in the paper

Entry point discovery is to reveal whether the user is aware that a certain API call has been made.

Graph Similarity based Classification

- **Graph Similarity-based Feature Extraction**
  - Generate behavior graphs for dataset
  - Each unique graph → A feature
  - Example:

Index of Graph in DB

G1	G2	G3	G4	G5	G6	G7	G8	…	G861	G862
0	0	0	0.9	0	0	0.9	0.7	…	0	0

Similarity to the Graphs of a given APP

Graph Similarity Score

- **Weighted Graph Similarity (WGS)**

$$wgs(G,G',\beta)=1-\frac{wged(G,G',\beta)}{wged(G,\phi,\beta)+wged(\phi,G',\beta)}$$

- **Weighted Graph Edit Distance (WGED)**

$$wged(G,G',\beta)=\min\left(\sum_{v_I\in\{V'-V\}}\beta(v_I)+\sum_{v_D\in\{V-V'\}}\beta(v_D)+|E_I|+|E_D|\right)$$

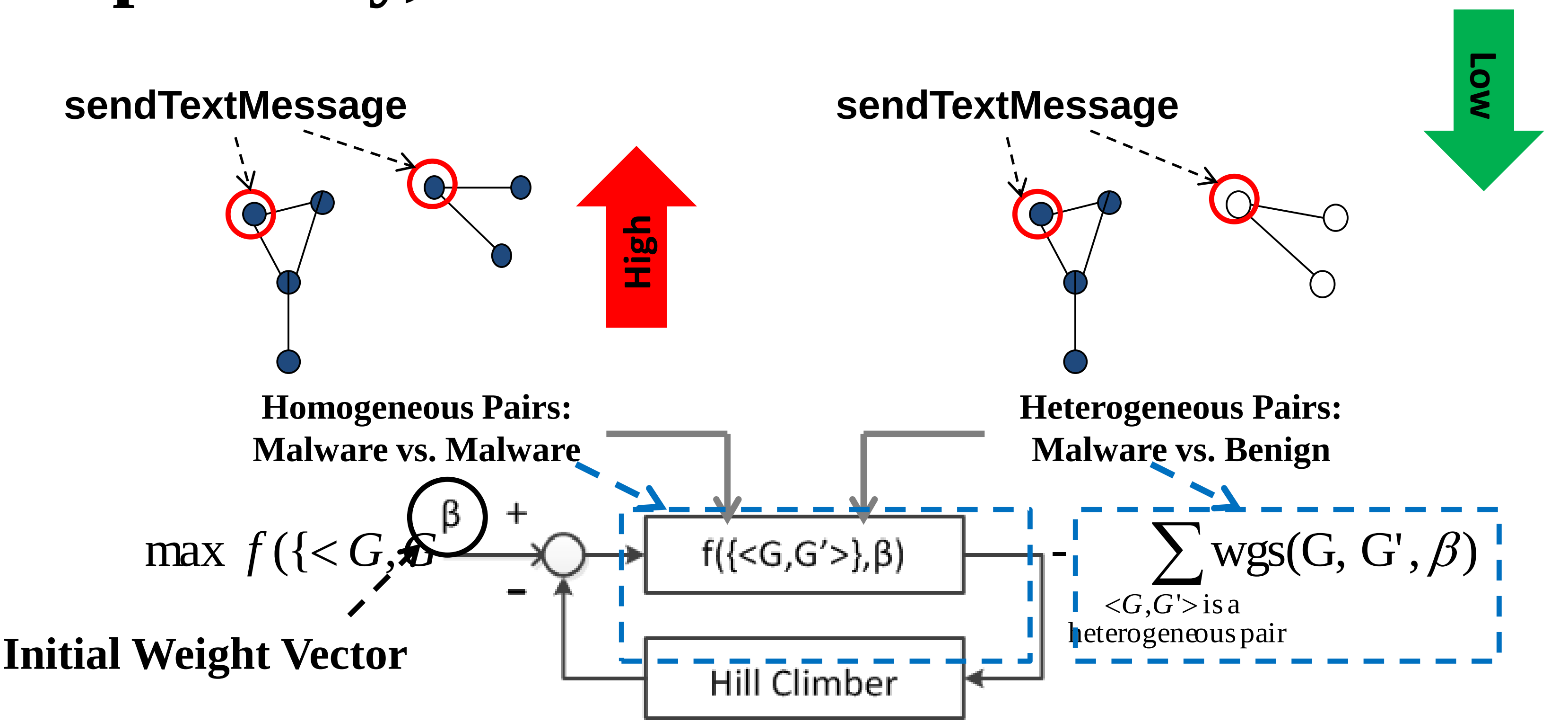
  - Weight only on vertices
  - Need to enhance Bipartitie algorithm

Weight Assignment

- **Selection of Critical API Labels**
  - **Sensitive to Malware**
  - **Concept Learning**
    - Rarely occur in benign apps
    - Happen more frequently in malware
  - **108 Critical APIs, automatically assigned weights > > 1**
  - **The rest, assigned a weight of 1**

Weight Assignment

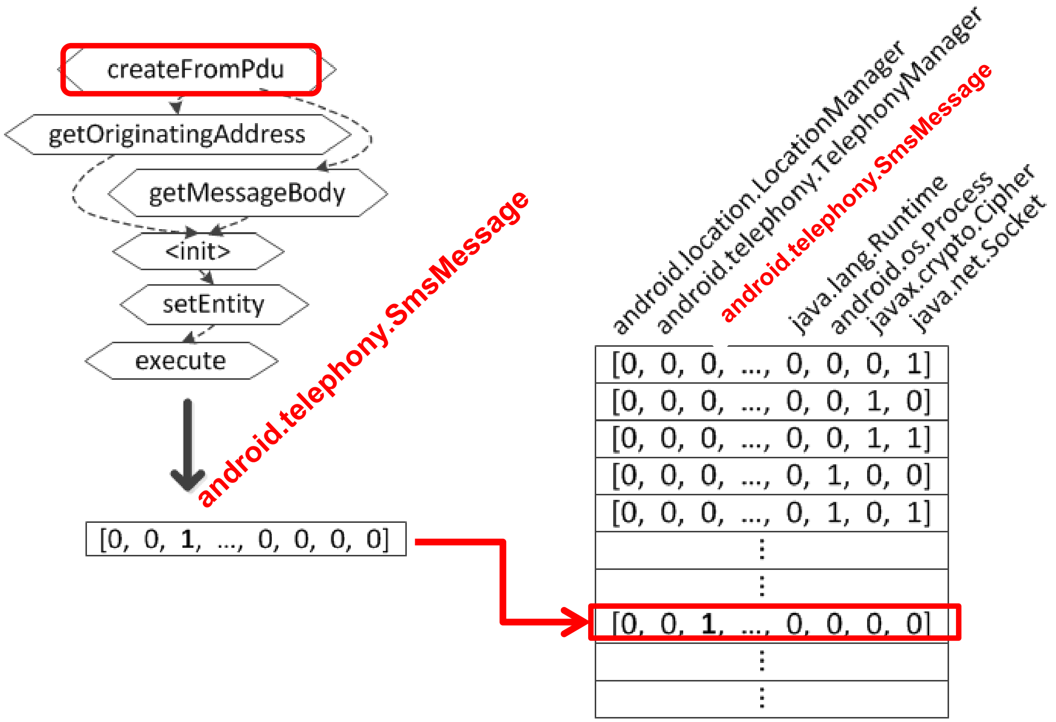
Intuitively, Optimization Problem:



Output: Optimal Weight Vector

Graph Database Query

- **Bucket-based Indexing**
  - Bitvector of Critical API Package Names as Index
  - **Exact** match on index



Malware Classification

- **Anomaly Detection**
  - Binary detector: compare against benign graphs
  - Empirically: all similarity scores <70% = Anomaly
- **Signature Detection**
  - Multi-label detector: compare against malware graphs
  - Generate feature vectors to train a Naive-Bayes classifier

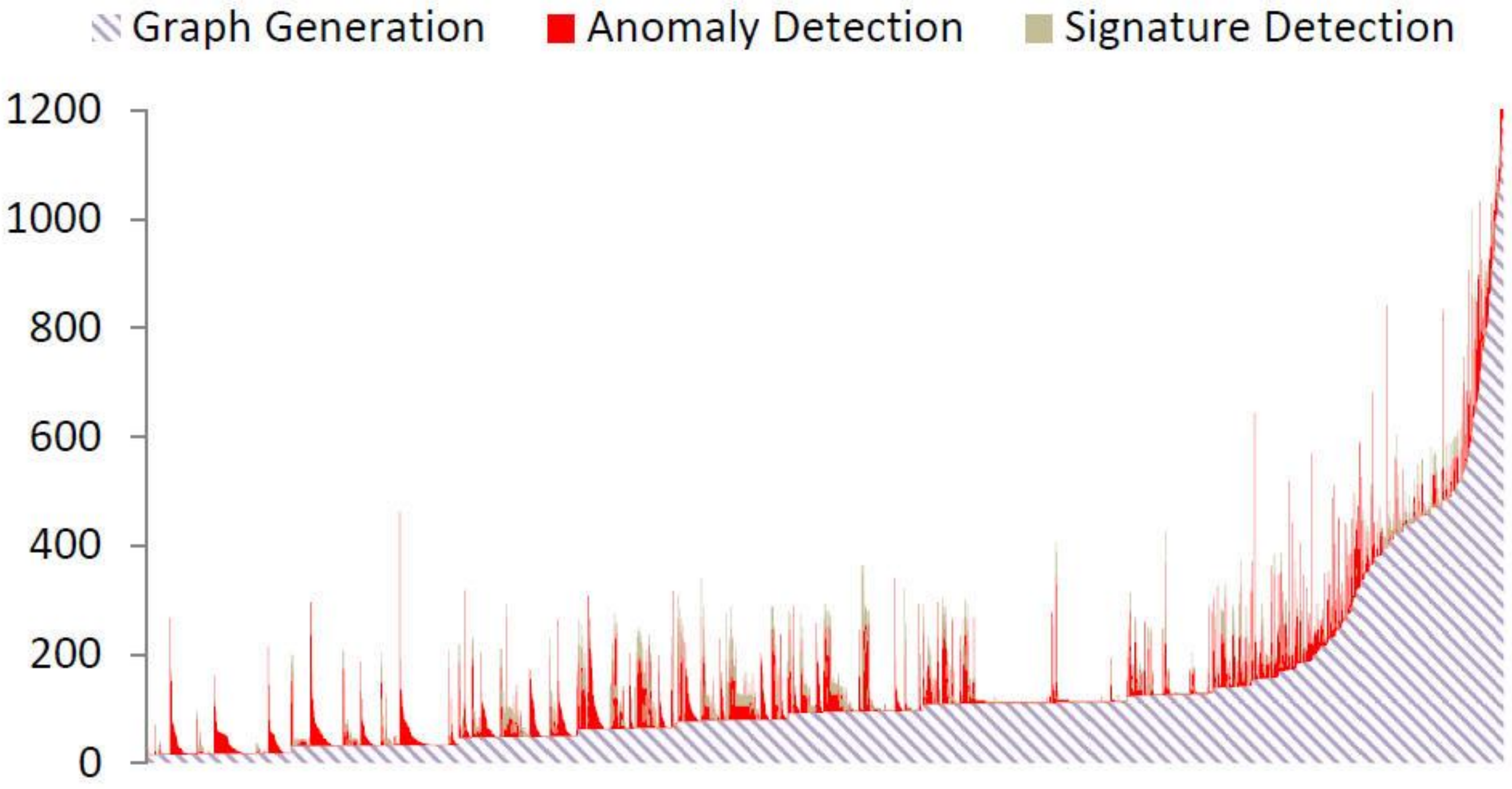
	G1	G2	G3	G4	G5	G6	G7	G8	…	G861	G862
ADRD	0	0	0	0	0	0.8	0.9	0	…	0	0
DroidDream	0.9	0	0	0	0.8	0.7	0.7	0	…	0	0
DroidKungFu	0	0.7	0	0	0.6	0	0.6	0	…	0	0.9

Evaluation: Overview

- **Dataset**
  - **2200** malware instances
    - Android Malware Genome Project, McAfee Labs
  - **13500** benign samples
    - Google Play

Evaluation: Runtime Performance

- **Most apps (96%) can be processed within 10 minutes.**

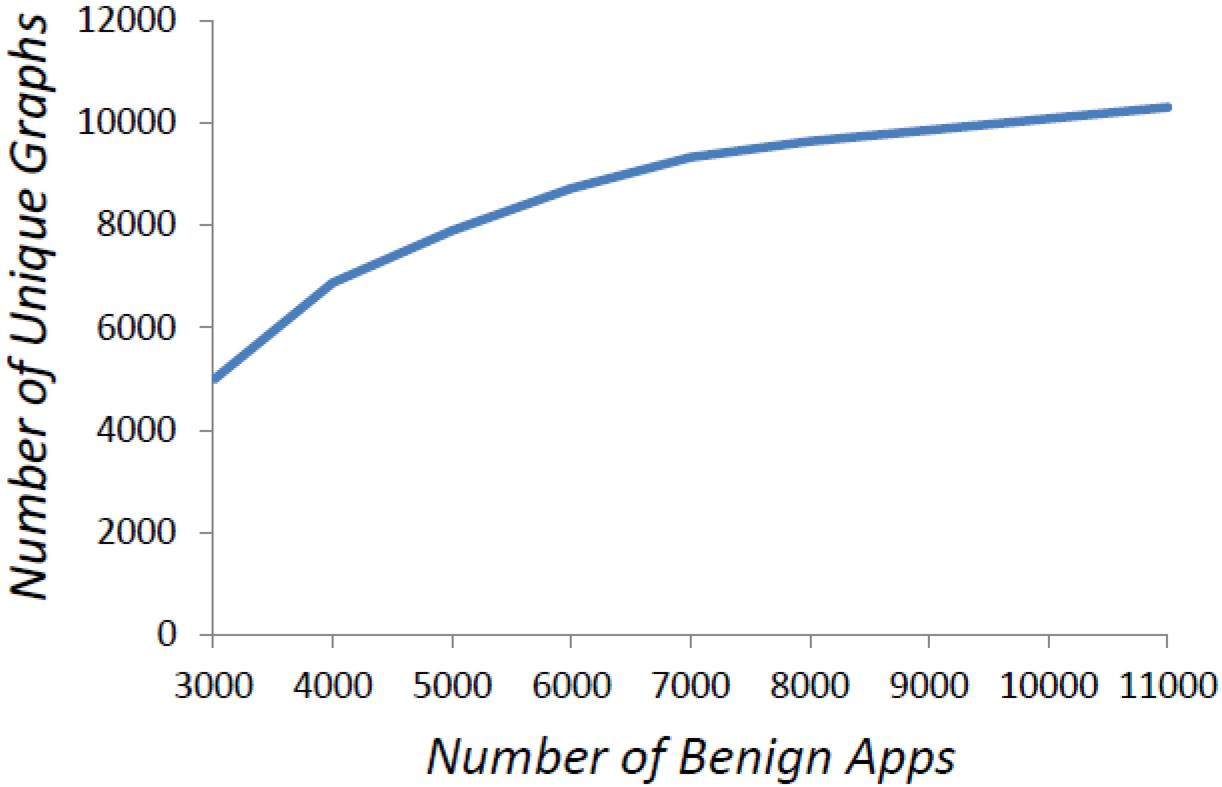


Evaluation: Classification Results

- **Signature Detection**
  - Database: **862** unique graphs from Android Malware Genome Project
  - **1050** malware samples to train classifier
  - **193** testing samples
  - Correctly label the families of **93%** malware
  - Mislabeled cases:
    - DroidKungFu ←→ DroidDream
    - Zitmo, Zsone, YZHC

Evaluation: Classification Results

- **Anomaly Detection**
  - **Convergence** of unique behavioral graphs for benign apps

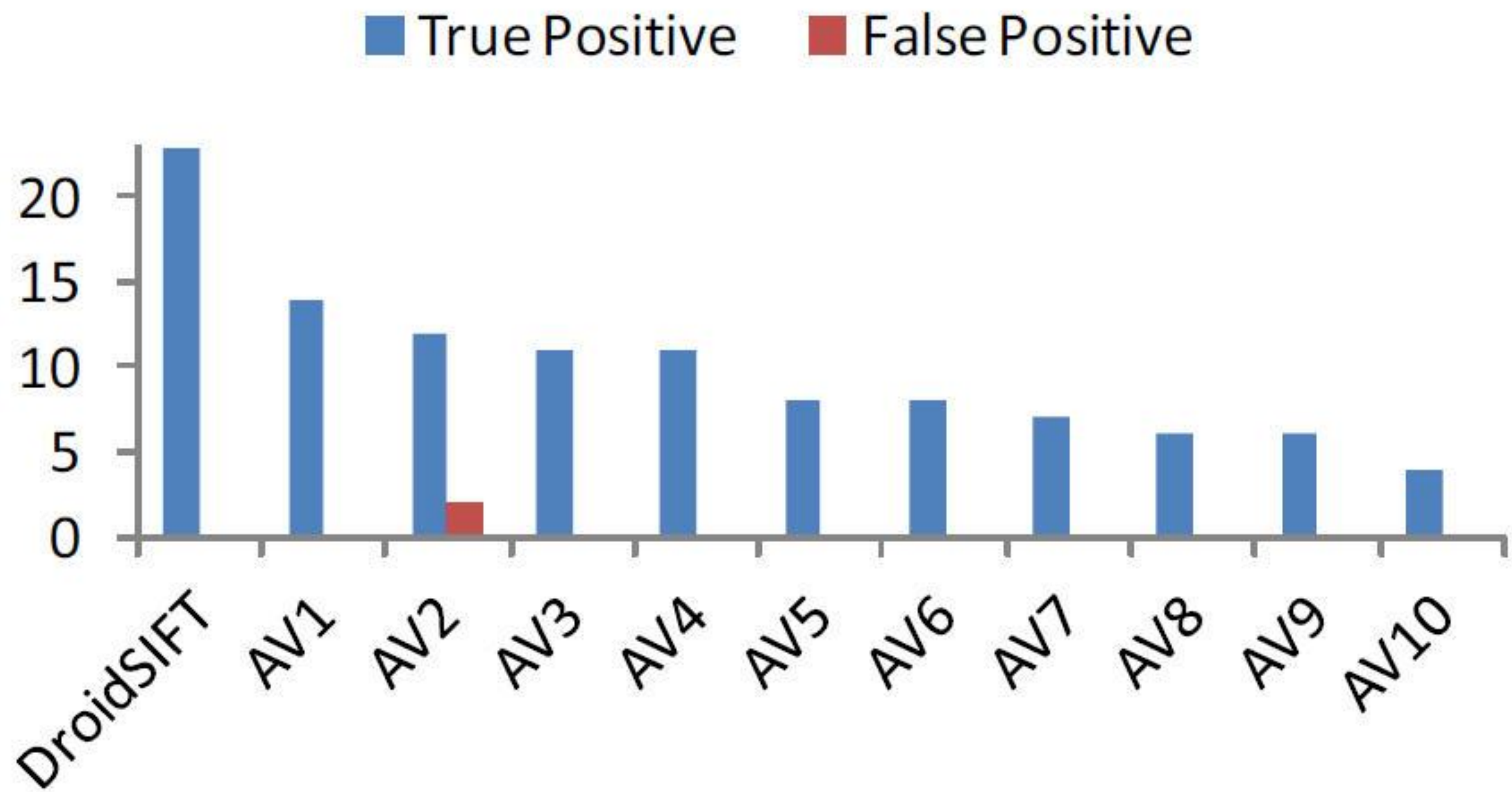


Evaluation: Classification Results

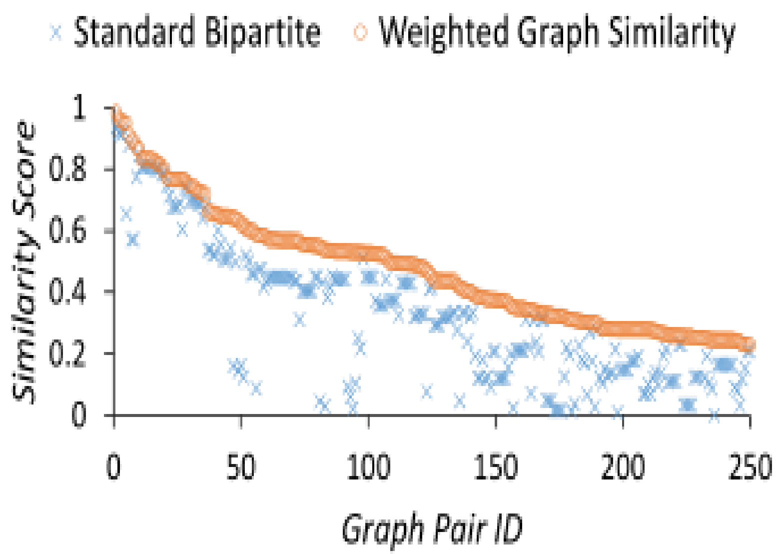
- **Anomaly Detection**
  - Database: **10420** unique graphs from **11400** benign apps
  - **2200** malware testing sample
    - False negative rate: 2% (Exploits and Downloaders)
  - **2100** benign testing sample
    - False positive rate: 5.15%
  - Detection of new malware (Android.HeHe)

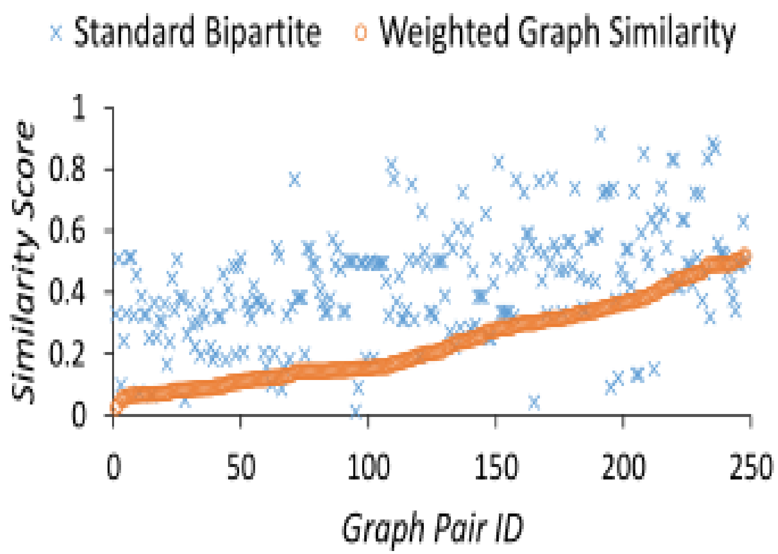
Evaluation: Obfuscated Samples

- **Detection of Transformation Attacks (TIFS'14)**
  - **21 Malware, 2 Benign**



Evaluation: Effectiveness of Weight Generation





Bipartite algorithm produces **73% true positive** rate in signature detection and **10% false negative** rate in anomaly detection

Weighted graph similarity metric is more sensitive to program semantics

Related Work

[1] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang. Riskranker: Scalable and accurate zero-day android malware detection. In Proceedings of MobiSys'12.

[2] Y. Zhou, Z. Wang, W. Zhou, and X. Jiang. Hey, you, get off of my market: Detecting malicious apps in official and alternative android markets. In Proceedings of NDSS'12.

[3] V. Rastogi, Y. Chen, and X. Jiang. Droidchameleon: Evaluating android anti-malware against transformation attacks. In TIFS'14.

[4] Y. Aafer, W. Du, and H. Yin. DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android. In Proceedings of SecureComm'13.

[5] D. Arp, M. Spreitzenbarth, M. HÃijbner, H. Gascon, and K. Rieck. Drebin: Efficient and Explainable Detection of Android Malware in Your Pocket. In Proceedings of NDSS'14.

[6] H. Peng, C. Gates, B. Sarma, N. Li, Y. Qi, R. Potharaju, C. Nita-Rotaru, and I. Molloy. Using Probabilistic Generative Models for Ranking Risks of Android Apps. In Proceedings of CCS'12.

[7] M. Christodorescu, S. Jha, S. A. Seshia, D. Song, and R. E. Bryant. Semantics-aware malware detection. In Proceedings of Oakland'05.

[8] M. Fredrikson, S. Jha, M. Christodorescu, R. Sailer, and X. Yan. Synthesizing near-optimal malware specifications from suspicious behaviors. In Proceedings of Oakland'10.

[9] K. Z. Chen, N. Johnson, V. D'Silva, S. Dai, K. MacNamara, T. Magrino, E. X. Wu, M. Rinard, and D. Song. Contextual policy enforcement in android applications with permission event graphs. In Proceedings of NDSS'13.

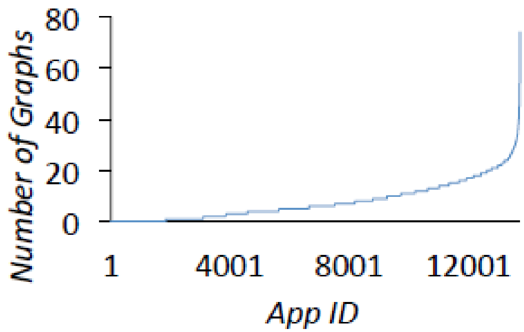
Conclusion

- We propose novel ***semantic-based*** approach that classifies Android malware via dependency ***graphs***.
- To fight against malware variants and zero-day malware, we introduce ***graph similarity metrics*** to uncover homogeneous application behaviors while tolerating minor implementation differences.

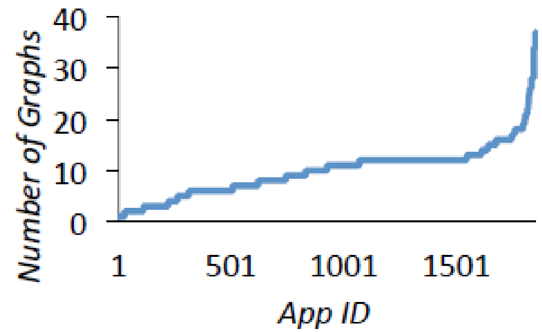
Questions?

Evaluation: Measurements of Graphs

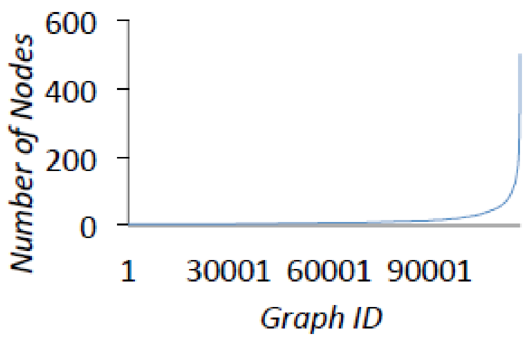
- **The amount of graphs/nodes is manageable.**



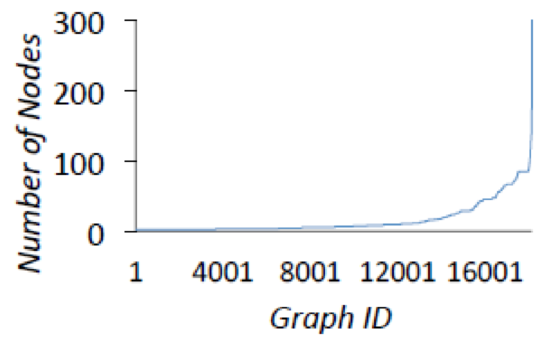
(a) **Graphs per Benign App.**



(b) **Graphs per Malware.**



(c) **Nodes per Benign Graph.**



(d) **Nodes per Malware Graph.**