Mobile Security: Malware Detection & Analysis

Yue Duan

Outline

- Research paper:
  - DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android
  - Semantics-aware android malware classification using weighted contextual api dependency graphs
  - Things You May Not Know About Android (Un)Packers: A Systematic Study based on Whole-System Emulation

DroidAPIMiner: Mining API-Level Features for Robust Malware Detection in Android

Yousra Aafer, Wenliang Du, and Heng Yin

SecureComm 2013

Overview

- Existing Android malware detection systems:
  - Permission-based [Peng et.al CCS'12]
    - over-privilege issue
    - used only by ad packages
    - malware can attack permission [Grace et.al NDSS'12]
  - Heuristics-based [DroidRanger NDSS'12]
    - not robust

Overview

- Goal
  - Overcome the shortcomings of the existing techniques
    - train a robust and lightweight classifier
    - focus on critical API calls

- Insights:
  - APIs are more semantic-rich than permissions
  - machine learning approach

Approach

- Feature extraction
- Feature refinement
- model learning and generation

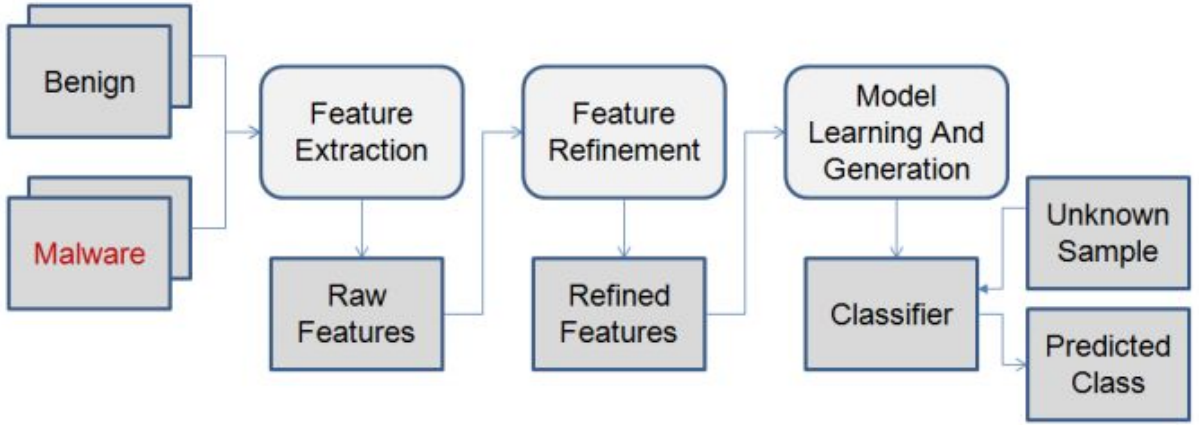


Fig. 1. Our Approach

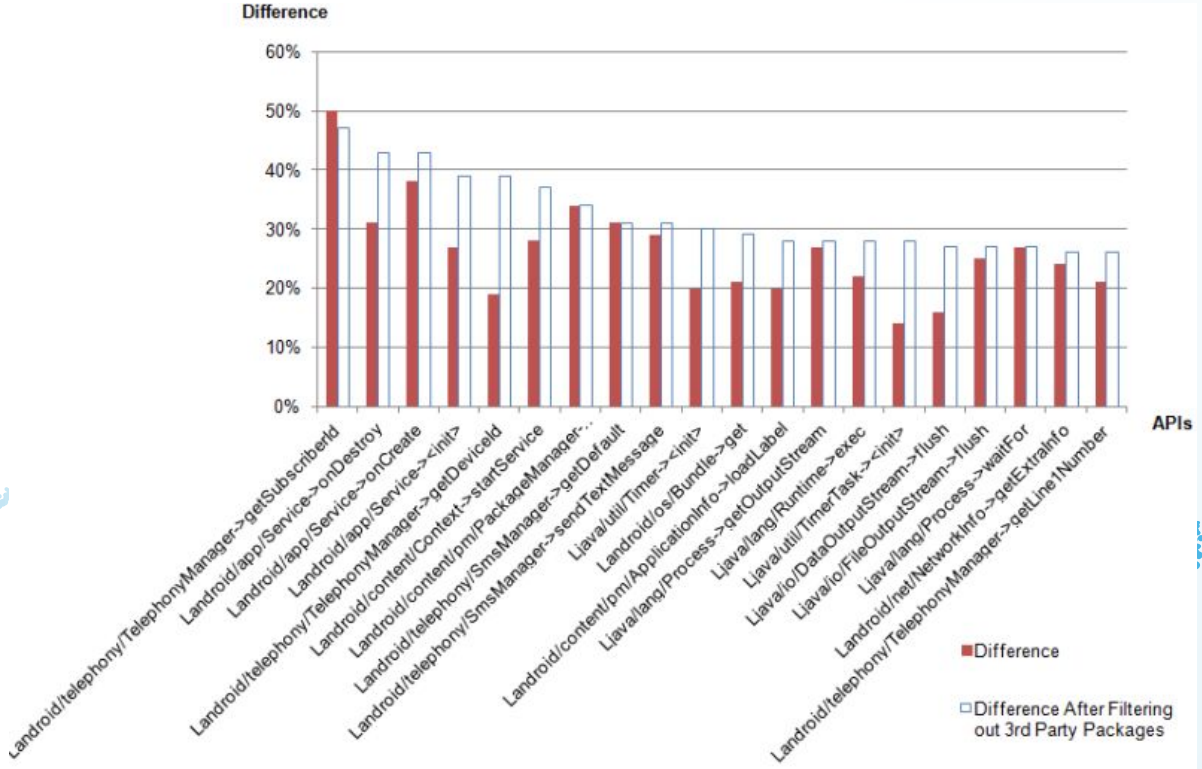
Feature Extraction

- statically extract API calls and their package level info
  - value of API parameters ⇐ data-flow analysis

Classes	Methods	Parameter Category
Intent IntentFilters	setFlags, addFlags, setDataAndType, putExtra, init	Flag is either: CALL, CONNECTIVITY, SEND, SENDTO, or BLUETOOTH
ContentResolver	query, insert, update..	URI is either: Content://sms-mms, Content://telephony, Content://calendar, Content://browser/bookmarks, Content://calllog, Content://mail, or Content://downlaods
DataInputStream BufferedReader DataOutputStream DataOutputStream	init, writeBytes...	Reads from process Reads from connection Uses SU command
InetSocketAddress	init	parameter IP is explicit or port is 80
File Stream StringBuilder String StringBuffer	init, write, append, indexOf, Substring	Dangerous Command such as: su, ls, loadjar, grep, /sh, /bin, pm install, /dev/net, insmod, rm, mount, root, /system, stdout, reboot, killall, chmod, stderr Accesses external storage or cache Contains either: An identifier (e.g. Imei), an executable file(e.g. .exe, .sh), a compressed file (e.g. jar, zip), a unicode string, an sql query, a reflection string, or a url

Feature Refinement

- Top commonly used malware features



Feature Refinement

- Application-specific resources APIs
  - Content Resolver:
    - insert(), delete(), query()
  - Context:
    - startService(), getFilesDir(), openFileOuput(), getApplicationInfo()
  - Intents:
    - setDataAndType(), setFlags(), addFlags(), setDataAndType()

Feature Refinement

- Android framework resources APIs
  - ActivityManager:
    - getRunningServices(), getMemoryInfo(), restartPackage()
  - PackageMnanger:
    - getInstalledPackages()
  - SmsManager:
    - sendTextMessage()
  - TelephonyManger:
    - getSubscriberId(), getDeviceId(), getLine1Number(), getSimSerialNumber (), getNetworkOperator(), getCellLocation()

Feature Refinement

- DVM related resources APIs
  - DexClassLoader:
    - loadClass()
  - Runtime and System:
    - Runtime.getRuntime.exec()
    - loadLibrary()
- System resources APIs
- Utilities APIs

Parameter Features

- Top invoked parameters types that yield to the highest support difference between the malware and benign sample

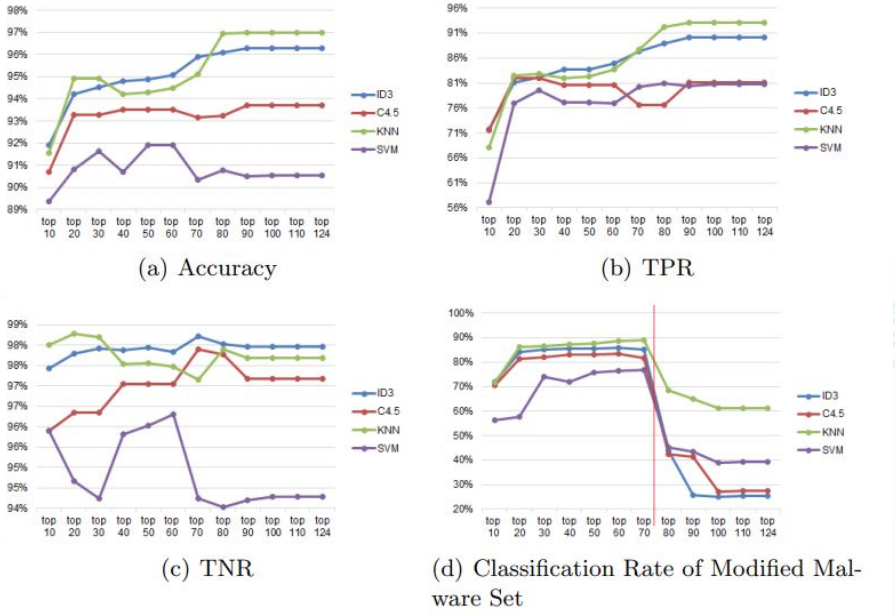
Class	Method	Parameter type	Difference (%)
StringBuilder	append	Dangerous command	35.95
ContentResolver	query	SMS or MMS	23.65
StringBuilder	append	Unicode string	23.6
StringBuilder	init	Dangerous command	23.07
DataOutputSream	writebytes	Reads from process	21.80
DataOutputSream	init	Reads from process	21.62
runTime	exec	Dangerous command	21.27
InetSocketAddress	init	Port 80	19.91
StringBuilder	append	Compressed file	19.58
DataInputStream	init	Reads from connection	19.27
String	valueOf	Unicode string	18.05
StringBuilder	append	File manupilation	17.79
File	init	Accesses external storage	16.92
InetSocketAddress	init	Explicit IP	14.87
String	getBytes	URL manupilation	14.05
Intent	setFlags	SendTo	12.94
Intent	setFlags	Call	11.67
ContentResolver	query	Telephony	10.88
Intent	setFlags	Send	10.47
ContentResolver	query	Call_log	10.12

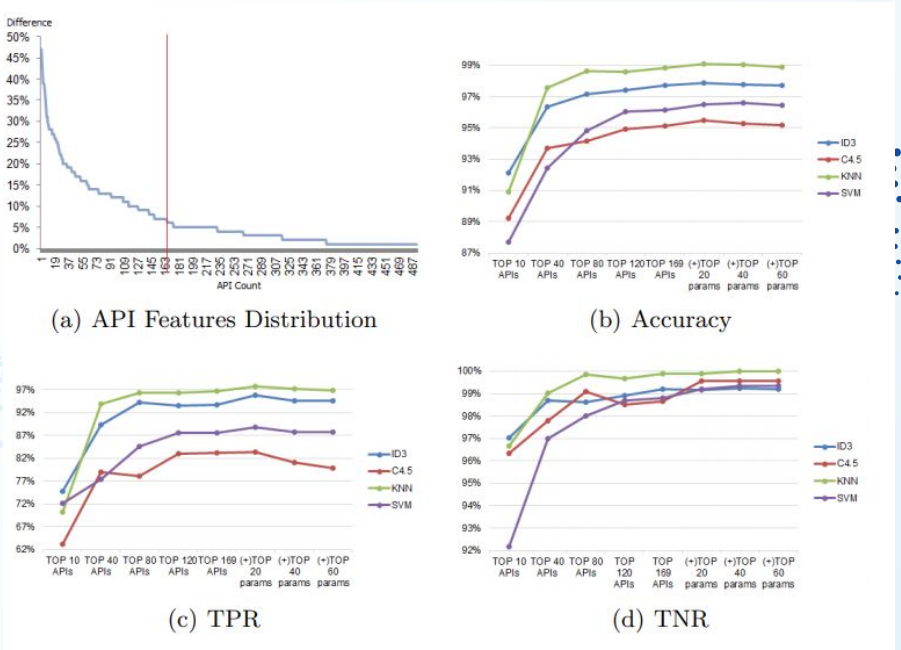
Classification Models

- Four different algorithms
  - decision trees: ID3 DT, C4.5 DT
  - lazy classifier: KNN
  - linear SVM

Evaluation

- Baseline
  - permission-based feature set





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

Mu Zhang, Yue Duan, Heng Yin, Zhiruo Zhao

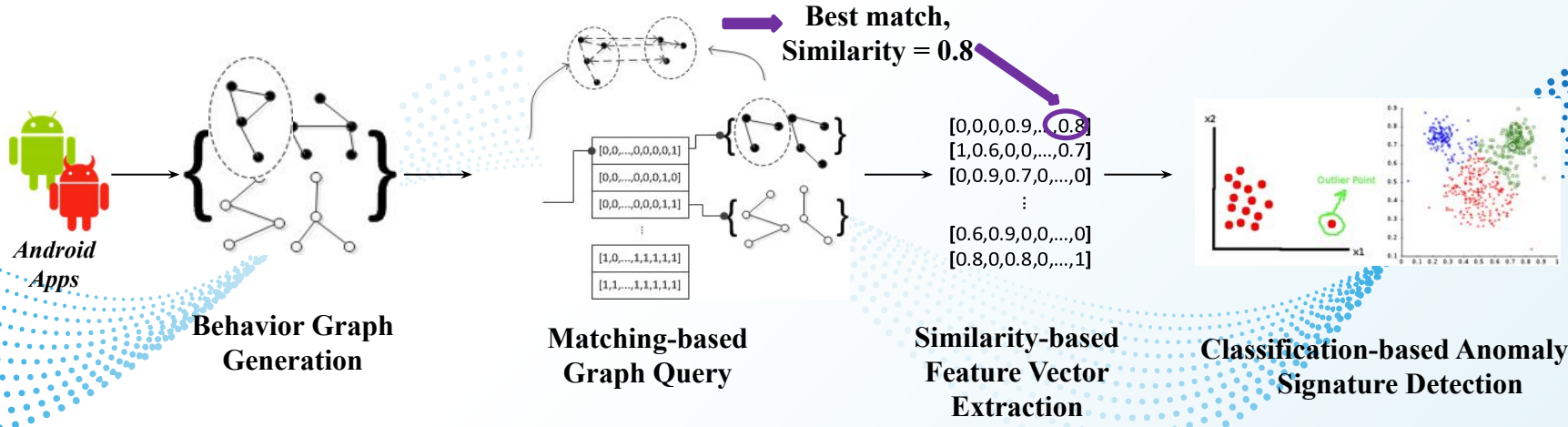
CCS 2014

Motivation

- Signature-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

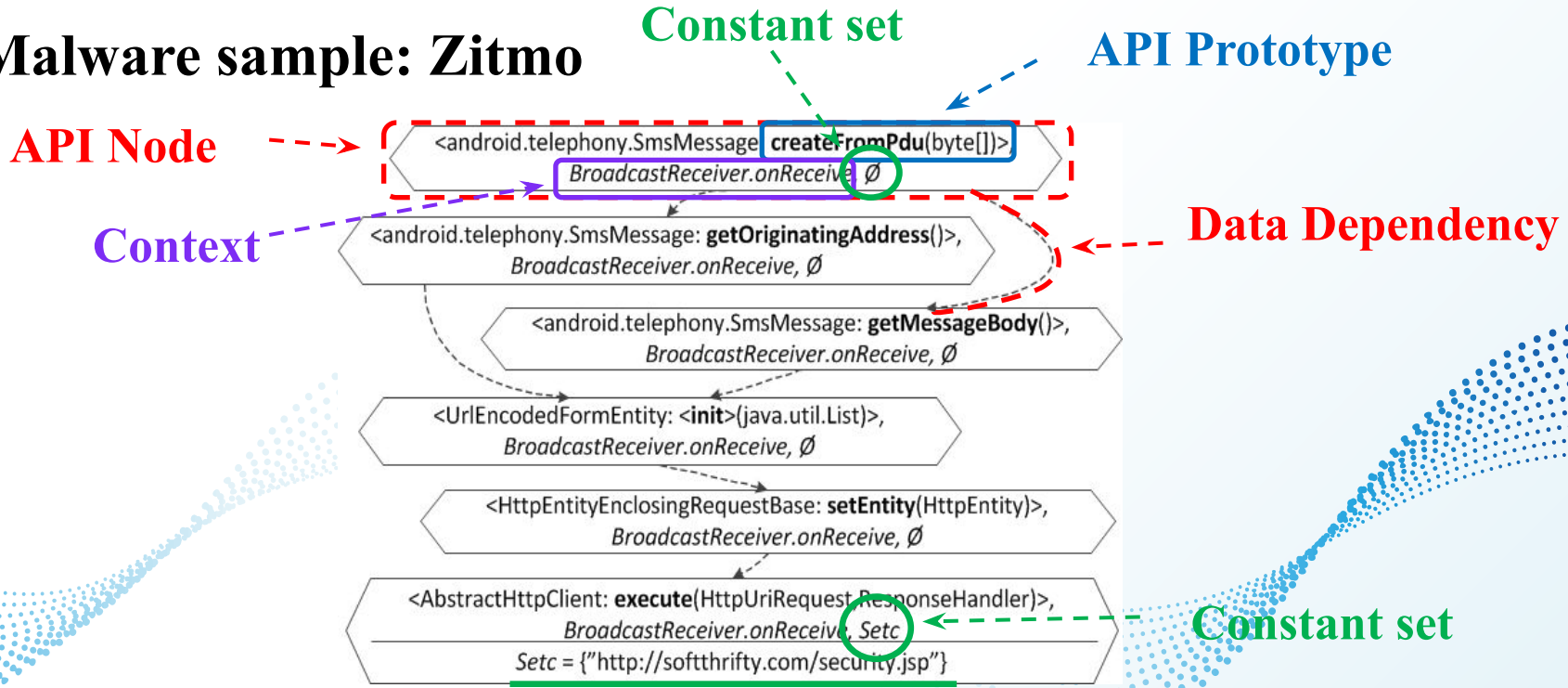
Approach Overview

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



Weighted Contextual API Dependency Graph

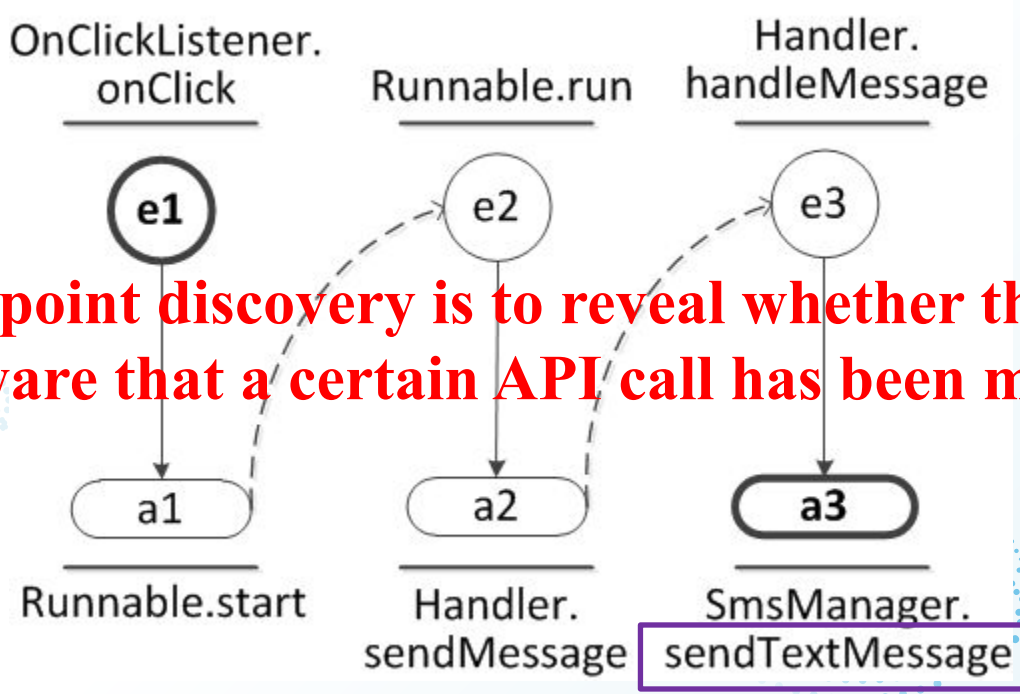
Malware sample: Zitmo



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

Weighted Contextual API Dependency Graph

- Context (Entry Point) Discovery



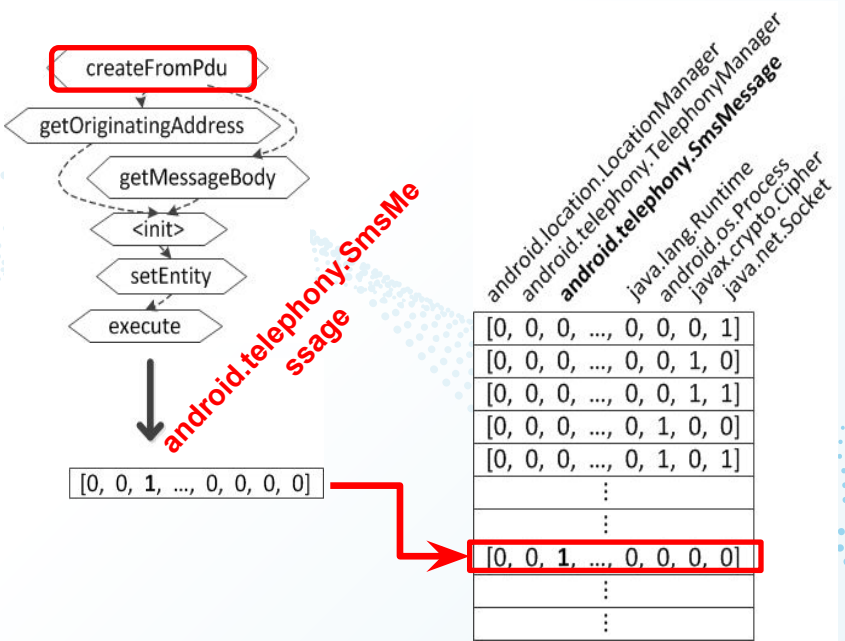
Entry point discovery is to reveal whether the user is aware that a certain API call has been made.

Weight Assignment

- Selection of Critical API Labels
  - Sensitive to Malware
  - Concept Learning
    - Rarely occur in benign apps
    - Happen more frequently in malware
  - Hill climbing algorithm
  - 108 Critical APIs

Graph Database Query

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



Malware Classification

- Anomaly Detection
  - Empirically: all similarity scores <70% = Anomaly
- Signature Detection
  - Generate feature vectors to train a Naive-Bayes classifier

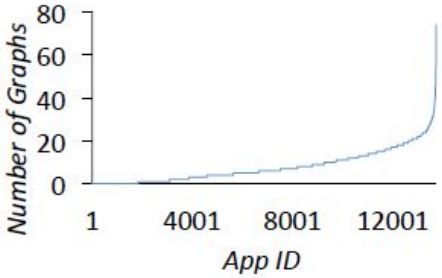
	G1	G2	G3	G4	G5	G6	G7	G8	...	G861	G862
ADRD	0	0	0	0	0	0.8	0.9	0	...	0	0
DroidDream	0.9	0	0	0	0.8	0.7	0.7	0	...	0	0
DroidKungFu	0	0.7	0	0	0.6	0	0.6	0	...	0	0.9

Evaluation

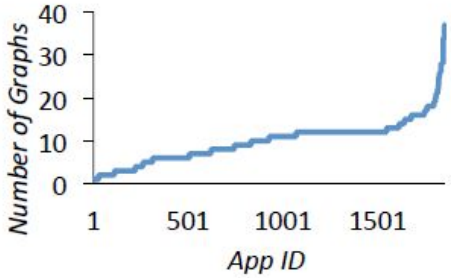
- Dataset
  - 2200 malware instances
    - Android Malware Genome Project, McAfee Labs
  - 13500 benign samples
    - Google Play

Evaluation: Measurements of Graphs

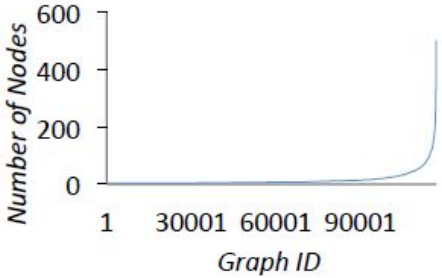
- The amount of graphs/nodes is manageable.



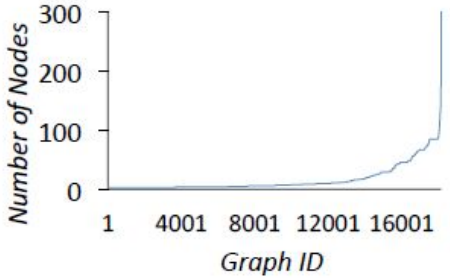
(a) **Graphs per Benign App.**



(b) **Graphs per Malware.**



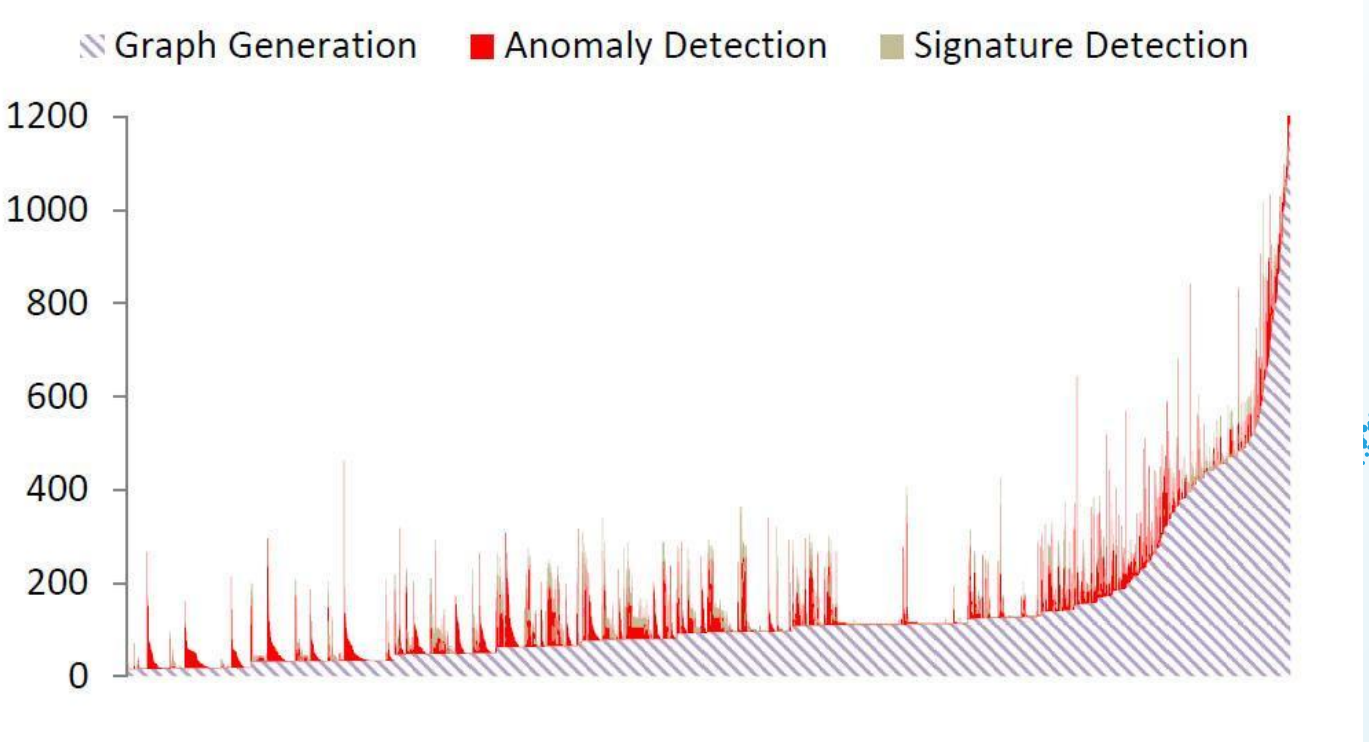
(c) **Nodes per Benign Graph.**



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

Evaluation: Runtime Performance

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



Evaluation: Classification Results

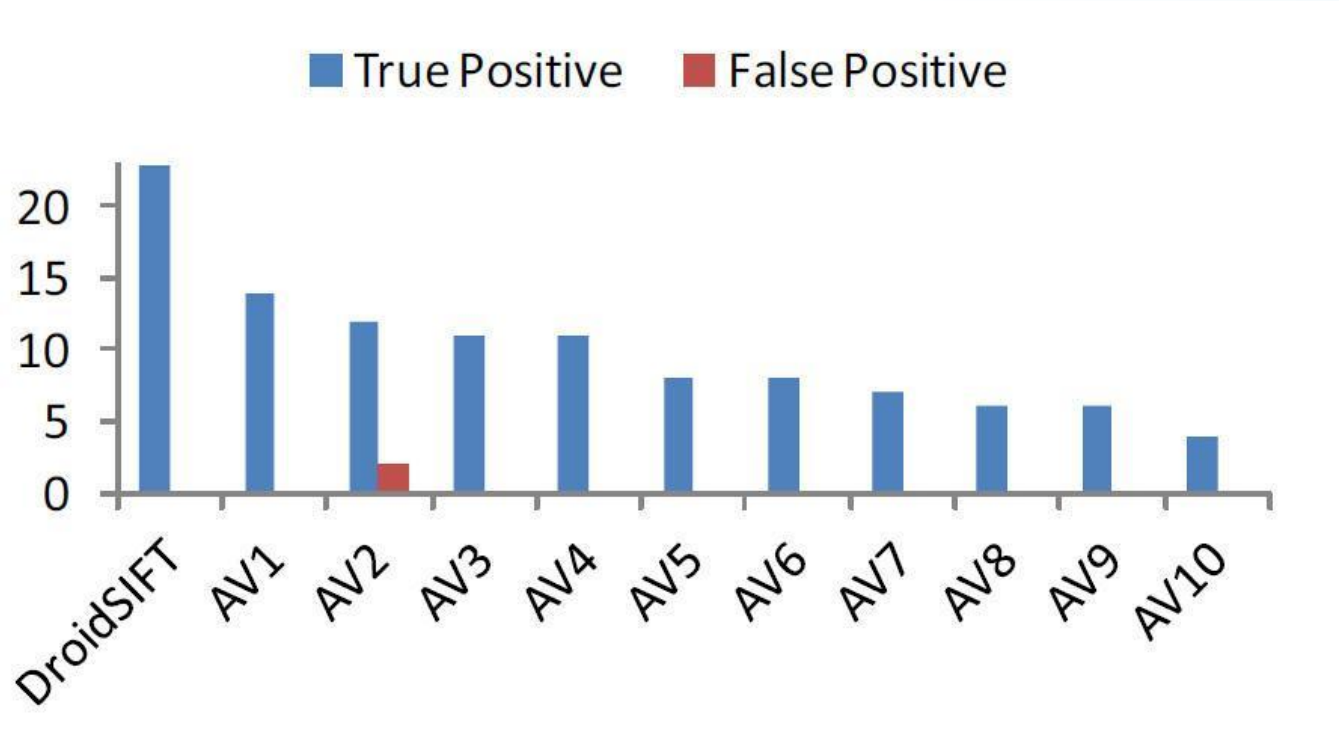
- Signature Detection
  - Multi-label Classification
  - Database: 862 unique graphs from Android Malware Genome Project
  - 1050 malware training samples, 193 testing samples
  - Correctly label the families of 93% malware
  - Mislabeled cases:
    - DroidKungFu, DroidDream
    - Zitmo, Zsone, YZHC

Evaluation: Classification Results

- Anomaly Detection
  - Binary 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: Obfuscation

- Detection of Transformation Attacks (TIFS'14)

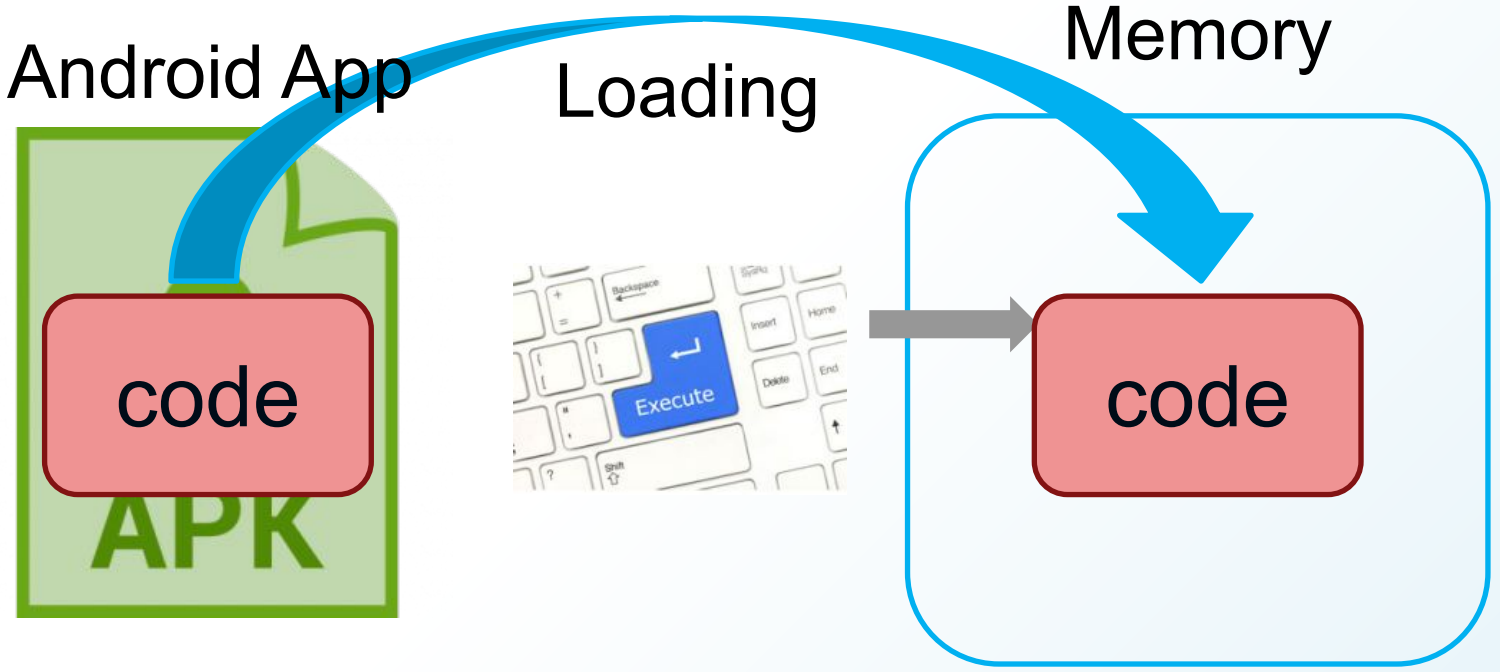


Things You May Not Know About Android (Un)Packers: A Systematic Study based on Whole-System Emulation

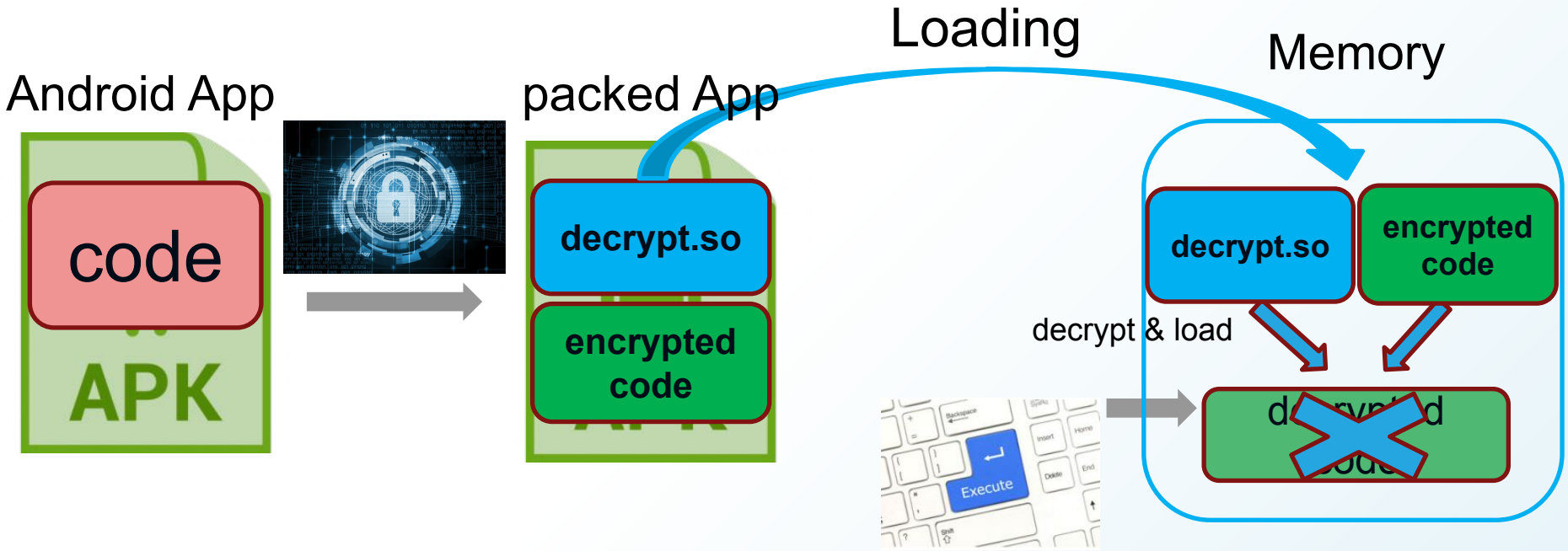
Yue Duan, Mu Zhang,
Abhishek Vasisht Bhaskar, Heng Yin, Xiaorui Pan, Tongxin Li,
Xueqiang Wang, XiaoFeng Wang

NDSS'18

Android packing



Android packing



Android packing

4.1 MB

```
.
./apktool.yml
./AndroidManifest.xml
./smali
./smali/com
./smali/com/example
./smali/com/example/hellojni
./smali/com/example/hellojni/R$color.smali
./smali/com/example/hellojni/R$layout.smali
./smali/com/example/hellojni/R$string.smali
./smali/com/example/hellojni/HelloJni.smali
./smali/com/example/hellojni/R$dimen.smali
./smali/com/example/hellojni/R$mipmap.smali
./smali/com/example/hellojni/R$integer.smali
./smali/com/example/hellojni/R.smali
./smali/com/example/hellojni/R$style.smali
./smali/com/example/hellojni/R$id.smali
./smali/com/example/hellojni/R$bool.smali
./smali/com/example/hellojni/R$anim.smali
./smali/com/example/hellojni/R$styleable.smali
./smali/com/example/hellojni/R$drawable.smali
./smali/com/example/hellojni/R$attr.smali
./smali/com/example/hellojni/BuildConfig.smali
./original
./original/META-INF
./original/META-INF/ALIAS_NA.SF
./original/META-INF/MANIFEST.MF
./original/META-INF/ALIAS_NA.RSA
./original/AndroidManifest.xml
./lib
./lib/armeabi-v7a
./lib/armeabi-v7a/libhello-jni.so
```

After packing

1 KB

```
.
./apktool.yml
./AndroidManifest.xml
./smali
./smali/com
./smali/com/ali
./smali/com/ali/fixHelper.smali
./smali/com/example
./smali/com/example/hellojni
./smali/com/example/hellojni/R$color.smali
./smali/com/example/hellojni/R$layout.smali
./smali/com/example/hellojni/R$string.smali
./smali/com/example/hellojni/HelloJni.smali
./smali/com/example/hellojni/R$dimen.smali
./smali/com/example/hellojni/R$mipmap.smali
./smali/com/example/hellojni/R$integer.smali
./smali/com/example/hellojni/R.smali
./smali/com/example/hellojni/R$style.smali
./smali/com/example/hellojni/R$id.smali
./smali/com/example/hellojni/R$bool.smali
./smali/com/example/hellojni/R$anim.smali
./smali/com/example/hellojni/R$styleable.smali
./smali/com/example/hellojni/R$drawable.smali
./smali/com/example/hellojni/R$attr.smali
./smali/com/example/hellojni/BuildConfig.smali
./original
./original/META-INF
./original/META-INF/ALIAS_NA.SF
./original/META-INF/MANIFEST.MF
./original/META-INF/ALIAS_NA.RSA
./original/AndroidManifest.xml
./lib
./lib/armeabi-v7a
./lib/armeabi-v7a/libpreverify1.so
./lib/armeabi-v7a/libdemolishdata
./lib/armeabi-v7a/libdemolish.so
./lib/armeabi-v7a/libdemolishdata.so
./lib/armeabi-v7a/libhello-jni.so
```

Why important

tion

Check Po
SOFTWARE TECHNOLOGI

Charger, however, uses a l st compensate with other means. The developers of Charger gave it everything they had to boost its evasion capabilities and so it could stay hidden on Google Play for as long as possible.

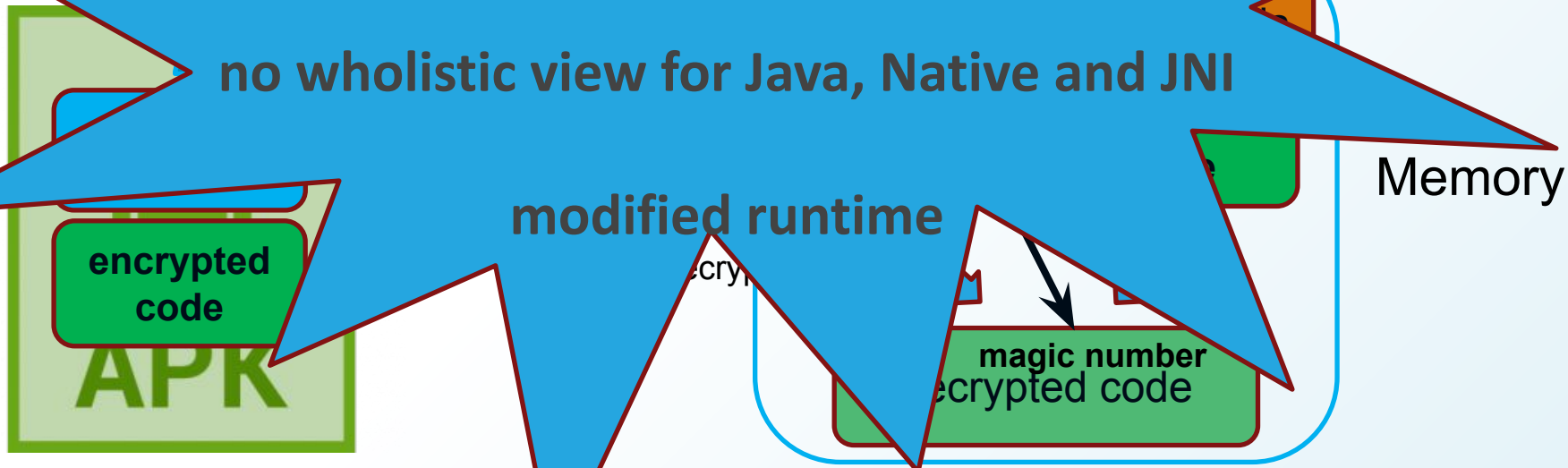
Existing unpackers

unreliable

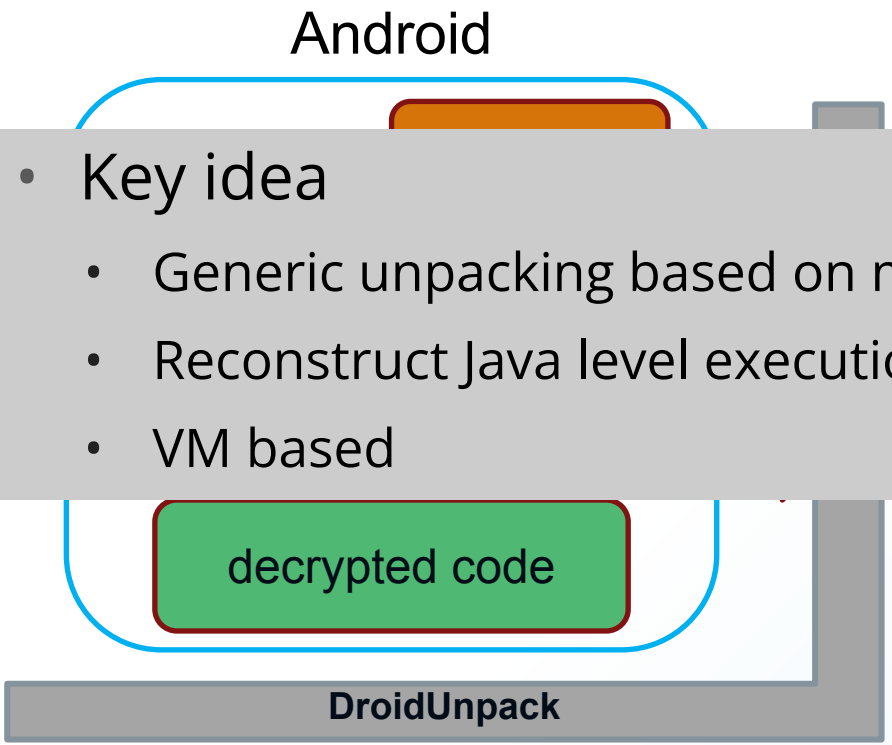
heuristics-based, only known packers

no wholistic view for Java, Native and JNI

modified runtime



DroidUnpack



- Key idea
  - Generic unpacking based on memory operation monitoring
  - Reconstruct Java level execution
  - VM based

ort

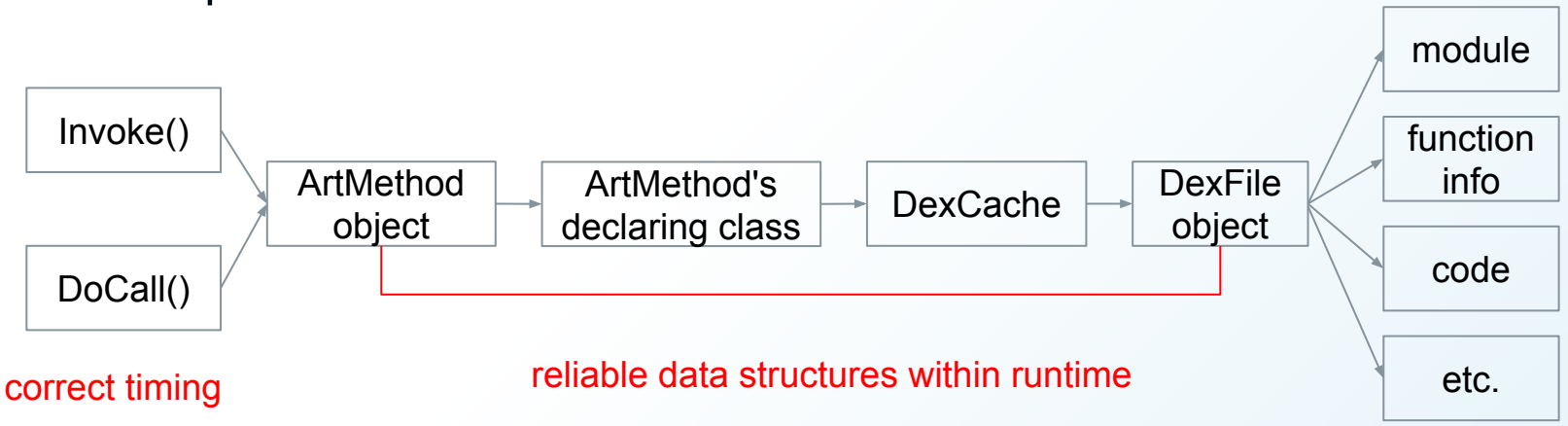
code and JNI

Monitor memory ops for packing behaviors

intrinsic characteristics

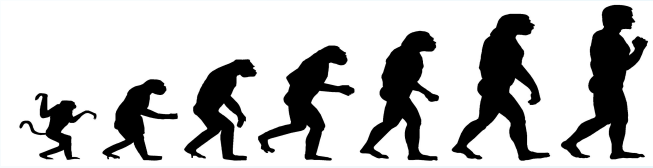
DroidUnpack

- Reconstructing ART Semantic View
  - Compiled Java functions
  - Interpreted Java functions



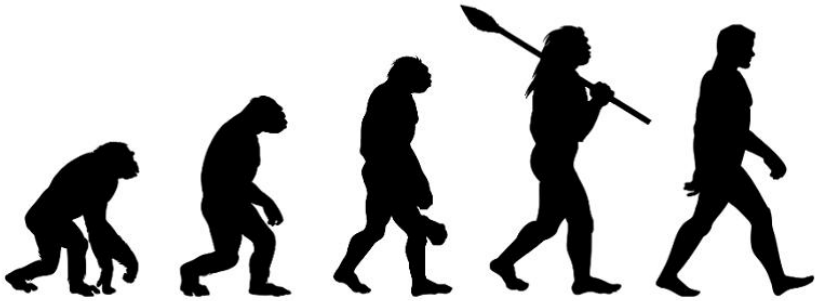
Research Questions

- Question set 1: High Level Landscape
- Question set 2: Detailed Behavioral Analysis
- Question set 3: Evolution of Android packing
- Question set 4: Existing defeating techniques



Evolution

Have Android packers evolved?



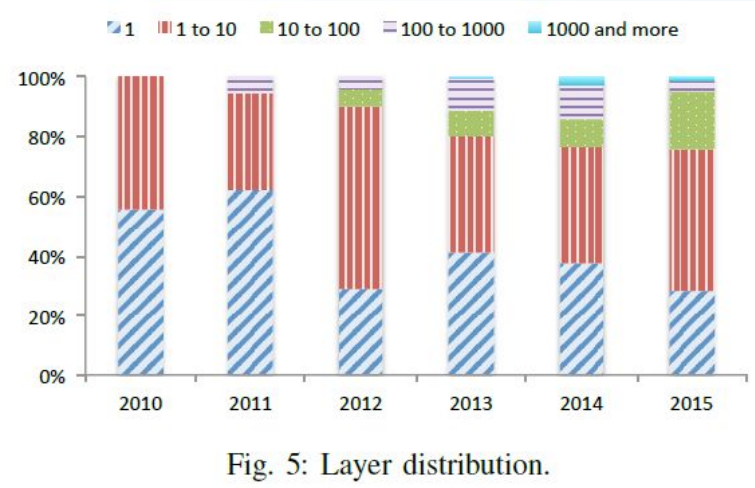


Fig. 5: Layer distribution.

Commercial packers

Commercial packers are supposed to protect your apps.
They do so by changing your apps.

Are the changes secure?

Commercial packers safety

Arbitrary code execution vulnerability

one vulnerable component is inserted stealthily

turn a secure app into a vulnerable one

acknowledged as highest priority vulnerability

won vulnerability discovery reward ~$8000

Commercial packers safety

Tencent

Privacy leakage

adds six new permissions

collect sensitive user data

send back via an insecure connection

Impact

Gaode Navi: **500 million**

Qianniuniu finance: **3 million**

QQ: **800 million**

Thank you!

Questions?