Finding Unknown Malice in 10 Seconds: Mass Vetting for New Threats at the Google-Play Scale

Kai Chen[‡,†], Peng Wang[†], Yeonjoon Lee[†], XiaoFeng Wang[†], Nan Zhang[†], Heqing Huang[§], Wei Zou[‡], Peng Liu[§]

[†]Indiana University, Bloomington

[‡]Institute of Information Engineering, Chinese Academy of Sciences

[§]College of IST, Penn State University

http://www.appomicsec.com

Background

- Android Malware
  - Billions of mobile computing devices. 70% are Android.
  - In 2014, 99% of mobile malware targets Android system
- Current Approaches
  - Signature-based detection & Behavior-based detection
- **Are they effective in malware detection?**

Are they effective?

- Signature-based detection
  - Cannot detect new malware: **Over 160,000 new malware samples created every day (Panda Security, 2014)**
  - Code obfuscation, e.g., DroidChameleon (AsiaCCS 2013)
- Behavior-based Detection
  - **Heavyweight** information-flow analysis
  - Require **known suspicious behaviors** (e.g., Dynamic code loading)

Can we design an approach that is:

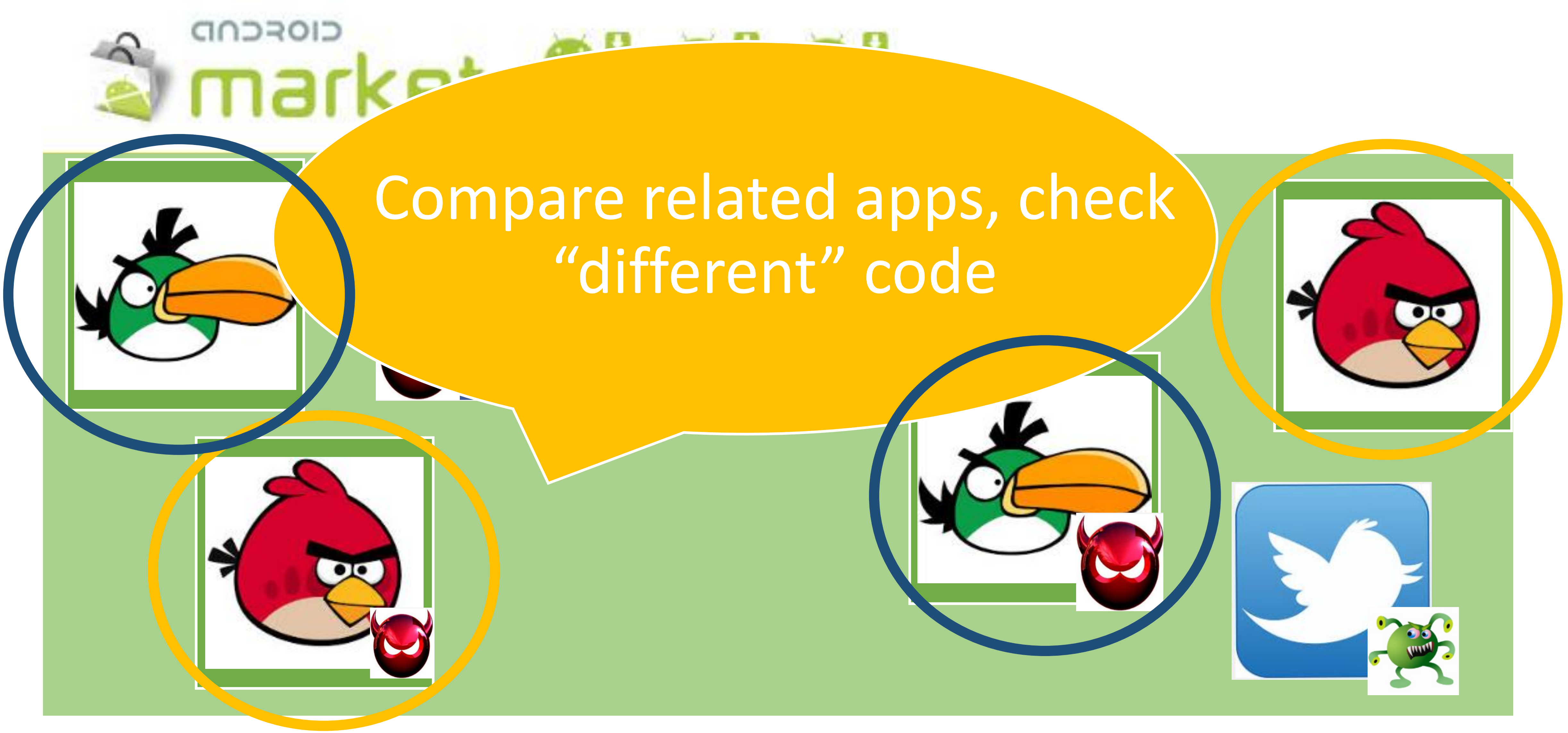
- Highly efficient
- Detect malware with unknown behaviors

We achieve this goal using neither signatures nor behaviors. But only code comparison.

Observation: a unique business model

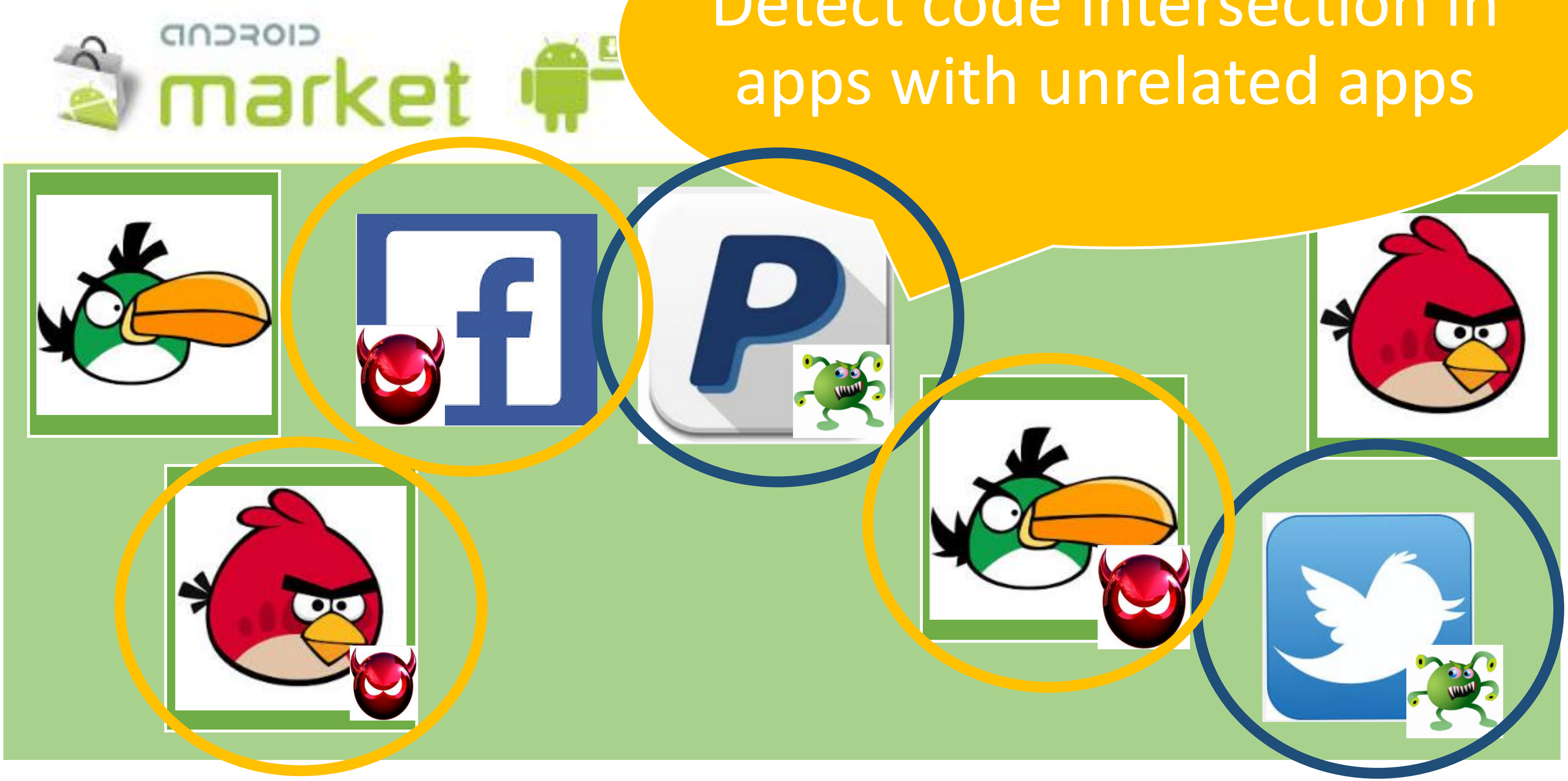
Attackers like to attach the same attack payload to legitimate apps.

Results of Repackaging

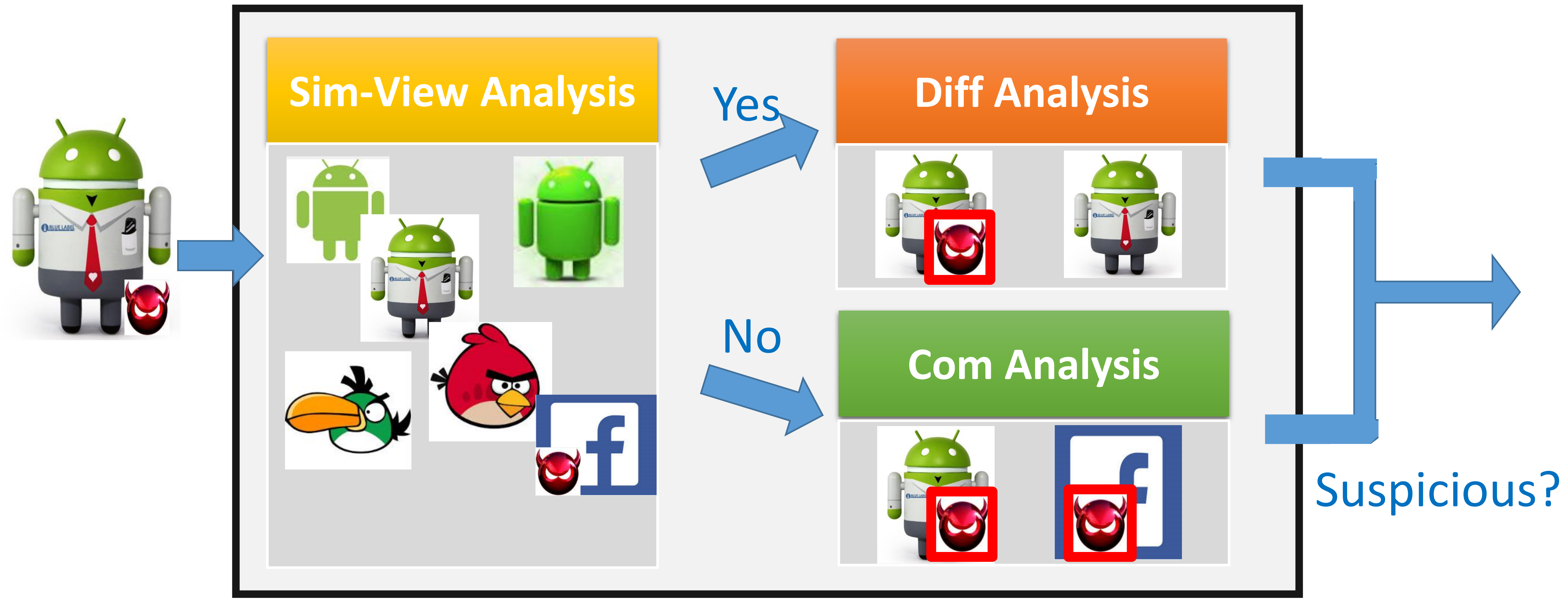


Results of Repackagin

Detect code intersection in apps with unrelated apps



Our approach: DiffCom Analysis



Sim-View Analysis: An example

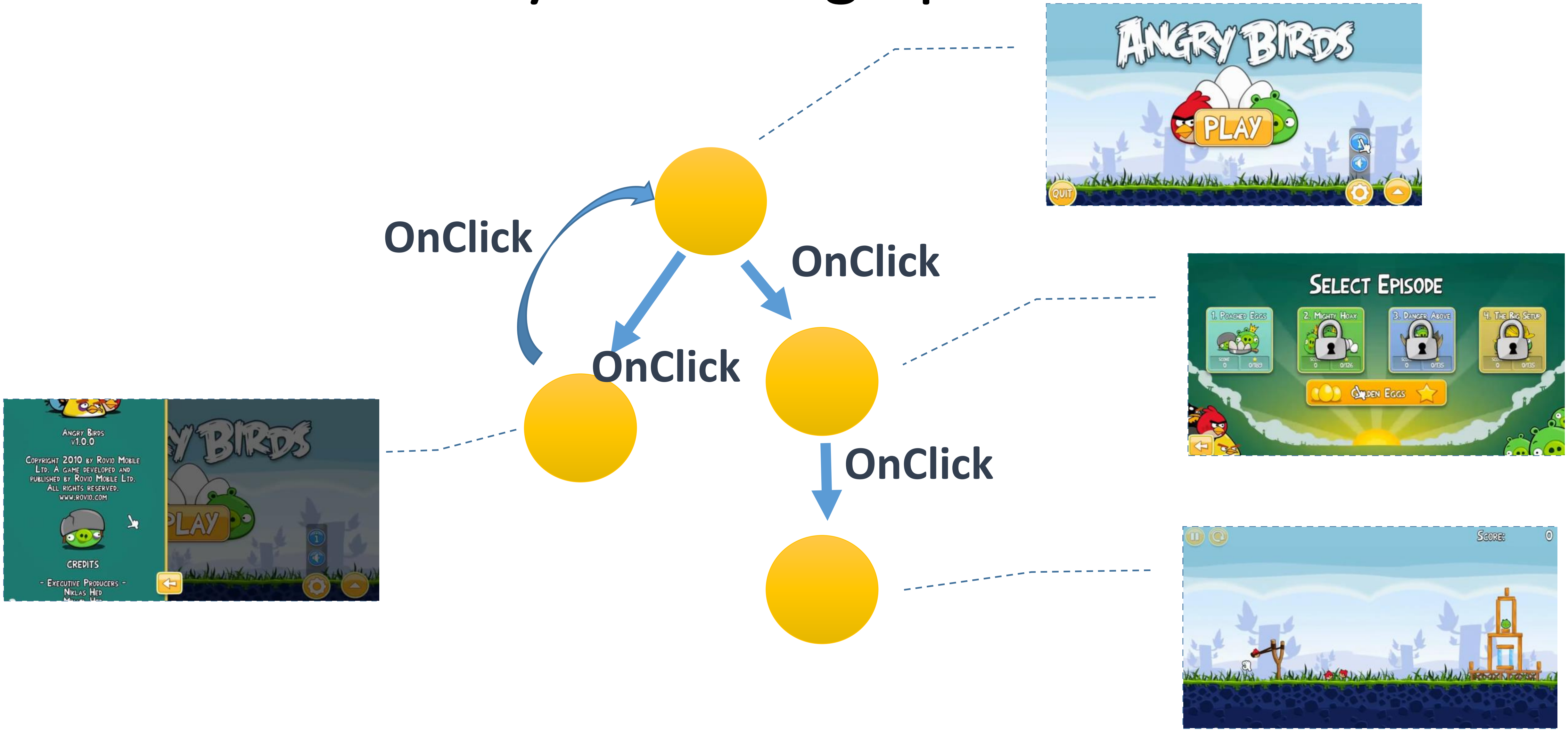
Sim-View Analysis: An example

Sim-View Analysis: An example

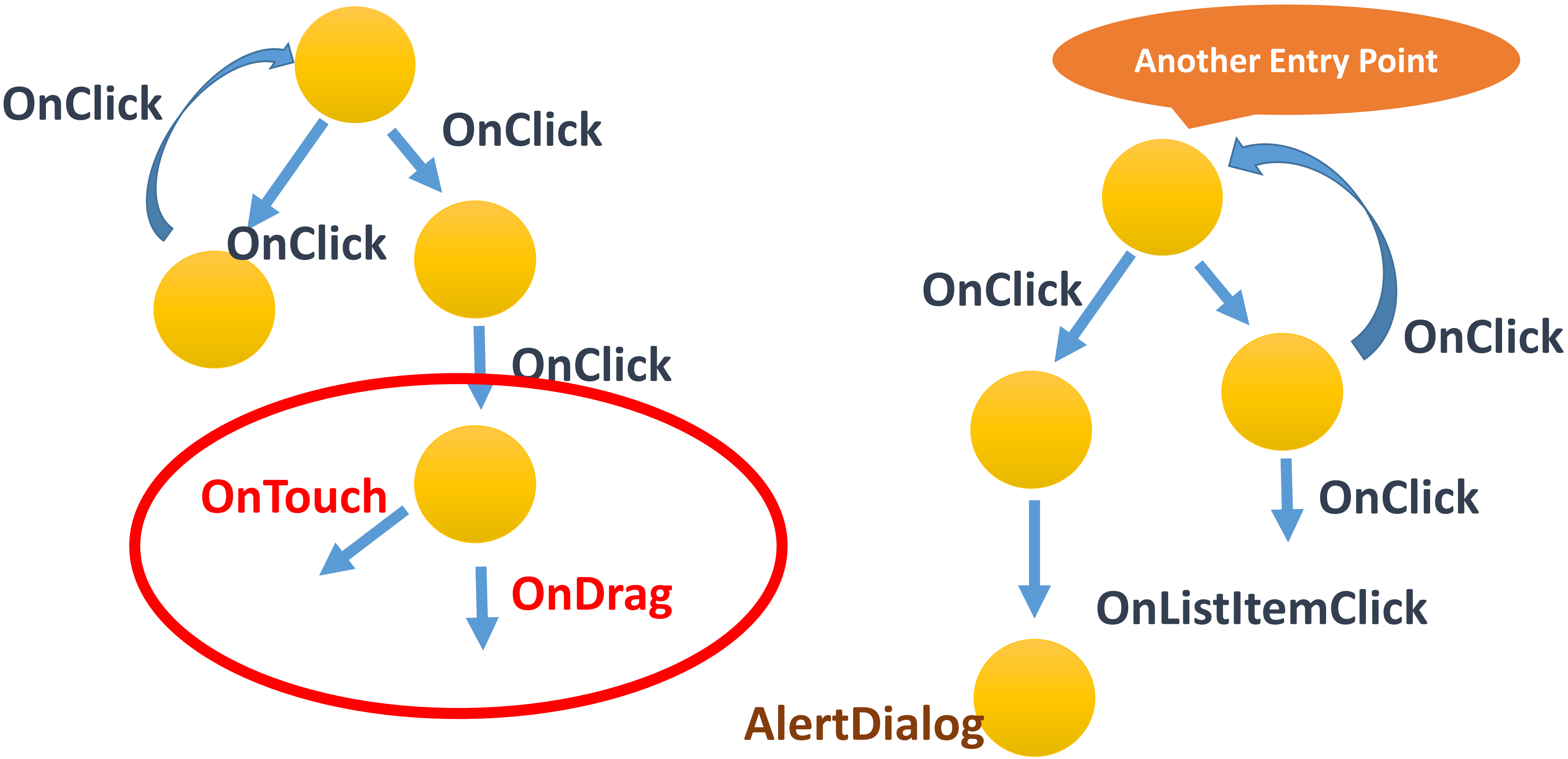
Sim-View Analysis: An example

Sim-View Analysis: An example

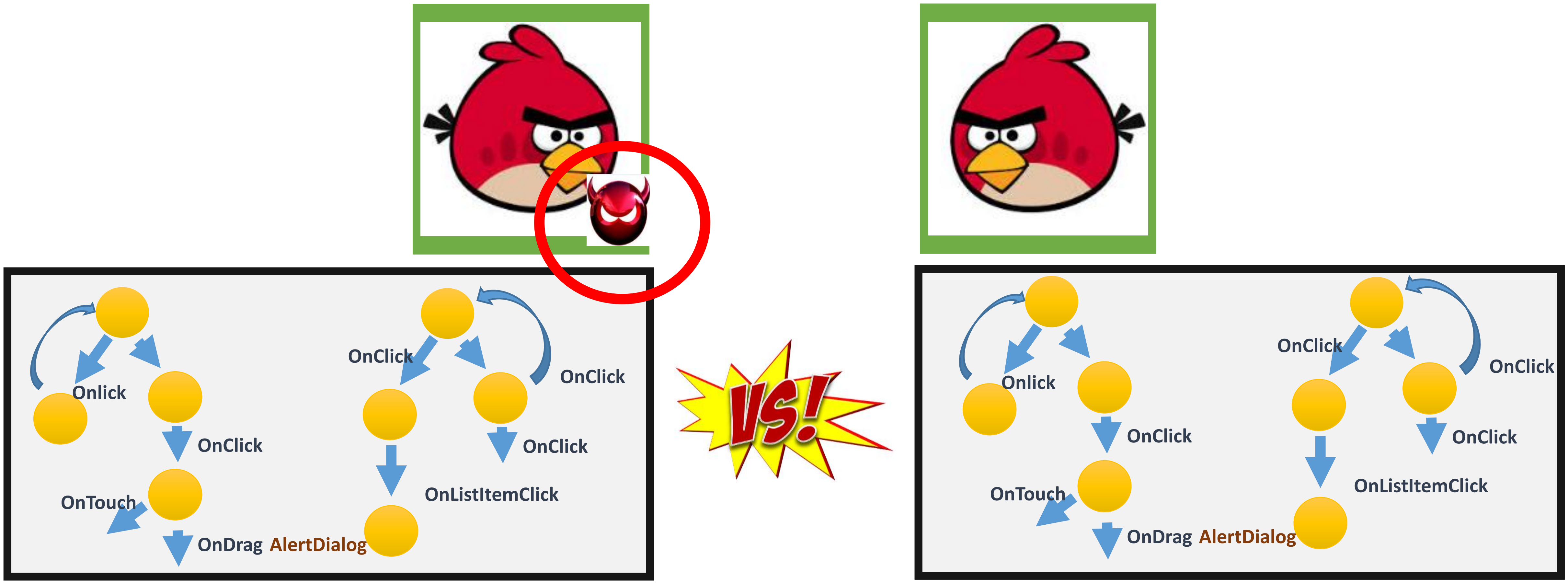
Sim-View Analysis: View graph



Sim-View Analysis: View Graph



Sim-View Analysis: Compare View Graphs



Can we avoid graph isomorphism analysis?

$O(n^2 \cdot M^2)$ $\Longrightarrow$ $O(c \cdot M)$

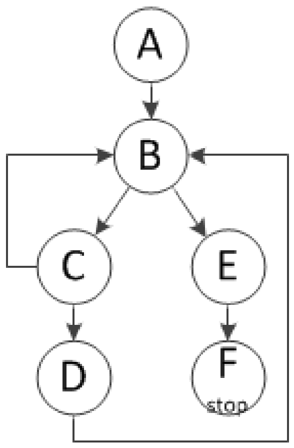
"Enemy" for scalability

Goal

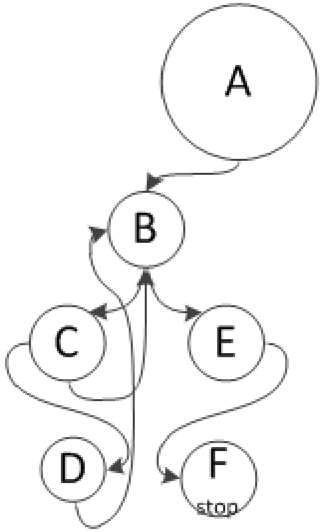
Sim-View Analysis: Challenge

- **Challenge 1: A Graph edge = abstract relation**
  - The abstract relation could have arbitrary length

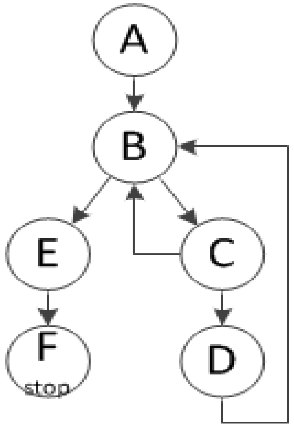
- **Challenge 2: Switching branches changes node positions**



Original Graph



Challenge 1



Challenge 2

Our idea: Fix the nodes in the graph

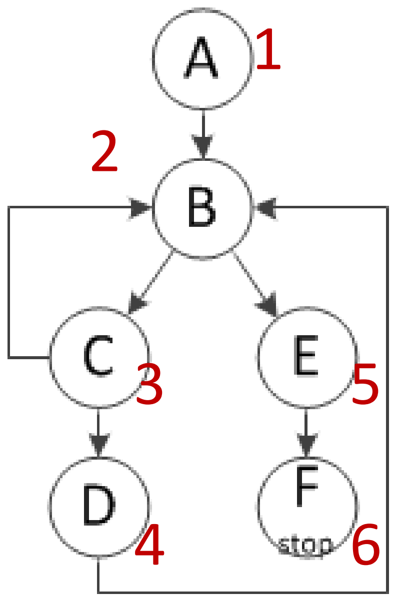
- **Step 1: view graph→3D-view-graph→v-core**
- **Step 2: Scalable comparison**

Sim-View Analysis: v-core

Step 1: Accurate mapping: **view graph→3D-view-graph**→v-core

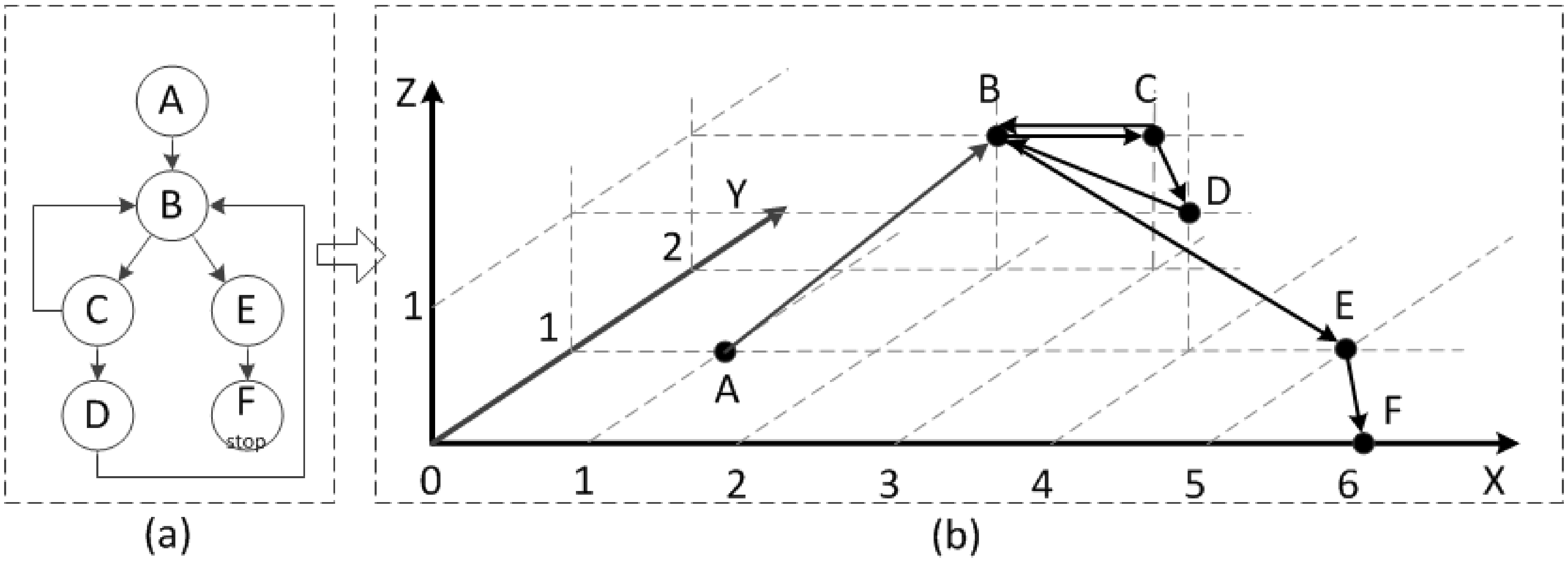
3D-View-Graph is a View Graph in which each node has a unique coordinate.

- The coordinate is a vector <x,y,z>
- x is the *sequence number* in the view graph
- y is the number of outgoing edges of the node
- z is the depth of loop of the node



Sim-View Analysis: v-core

Step 1: Accurate mapping: **view graph→3D-view-graph**→v-core

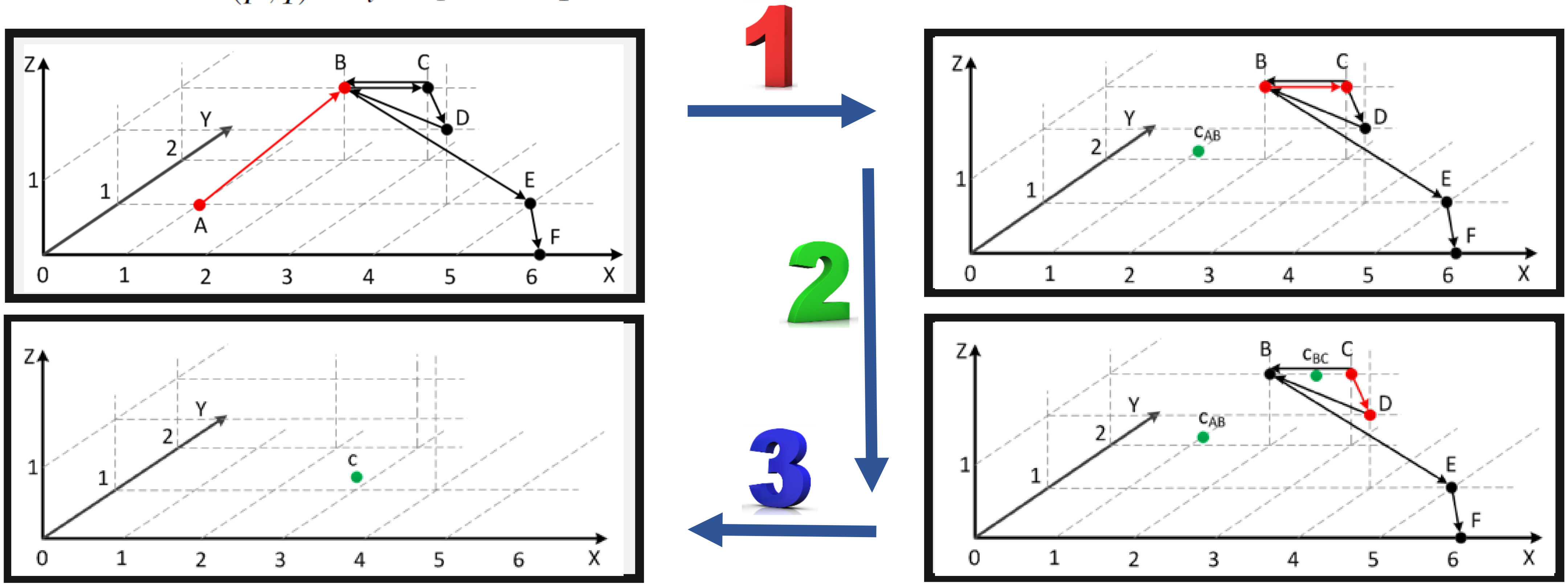


A <1, 1, 0>; B <2, 2, 1>; C <3, 2, 1>; D <4, 1, 1>; E <5, 1, 0>; F <6, 0, 0>

Sim-View Analysis: v-core

Step 1: Accurate mapping: view graph→**3D-view-graph→v-core**

$$vc_i = \frac{\sum_{e(p,q)\in G_i}(w_p\vec{c}_p + w_q\vec{c}_q)}{\sum_{e(p,q)\in G_i}(w_p + w_q)}$$



Sim-View Analysis: v-core

Step 2: Scalable comparison

- First, sub-graph-level comparison

$$|vc_i - vc_t| \leq \tau$$

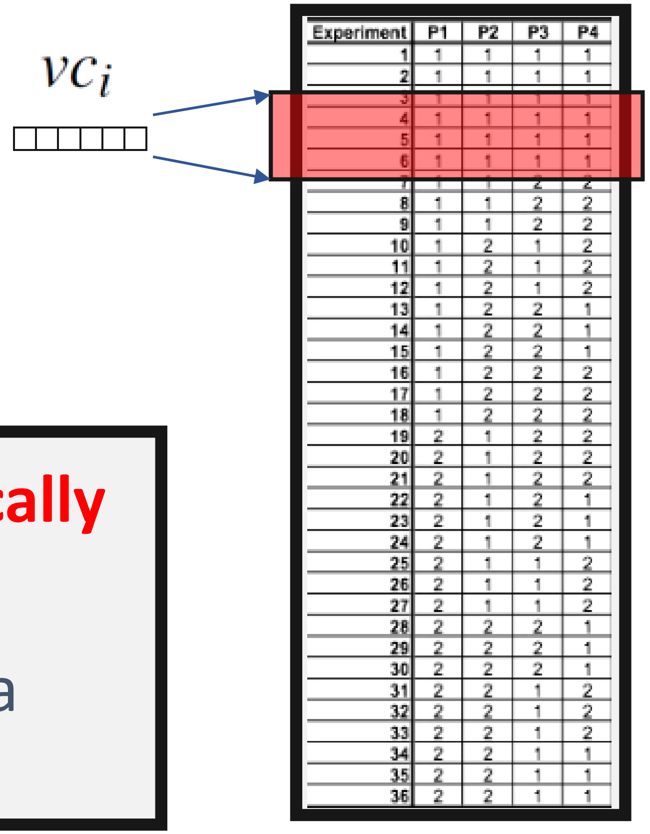
- Second, app-level comparison

$$\sum_l |G_{i(l)}| / \sum_i |G_i| \geq \theta$$

Feature 1: The similarity between two graphs is **monotonically** correlate to the “distance” between two v-cores.

Feature 2: V-cores are sortable. We only need to compare a v-core with its **neighbors**, but not all v-cores.

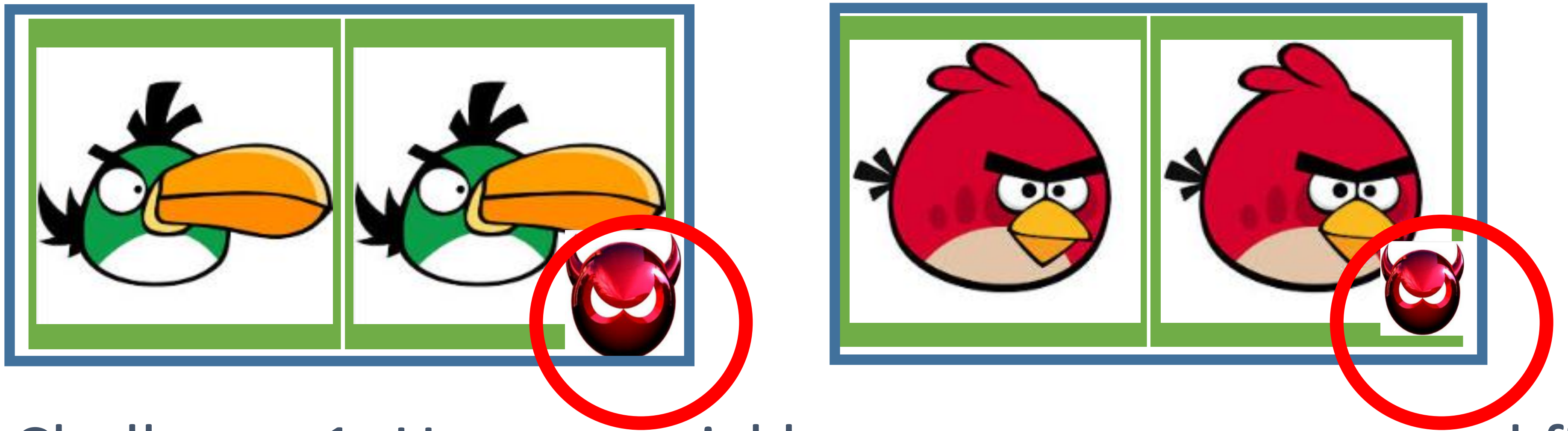
Localized global comparison



Experiment	P1	P2	P3	P4
1	1	1	1	1
2	1	1	1	1
3	1	1	1	1
4	1	1	1	1
5	1	1	1	1
6	1	1	1	1
7	1	1	2	2
8	1	1	2	2
9	1	1	2	2
10	1	2	1	2
11	1	2	1	2
12	1	2	1	2
13	1	2	2	1
14	1	2	2	1
15	1	2	2	1
16	1	2	2	2
17	1	2	2	2
18	1	2	2	2
19	2	1	2	2
20	2	1	2	2
21	2	1	2	2
22	2	1	2	1
23	2	1	2	1
24	2	1	2	1
25	2	1	1	2
26	2	1	1	2
27	2	1	1	2
28	2	2	2	1
29	2	2	2	1
30	2	2	2	1
31	2	2	1	2
32	2	2	1	2
33	2	2	1	2
34	2	2	1	1
35	2	2	1	1
36	2	2	1	1

Diff Analysis

- For apps having **the same view** and **different signatures**, the different methods between the two apps may be malicious

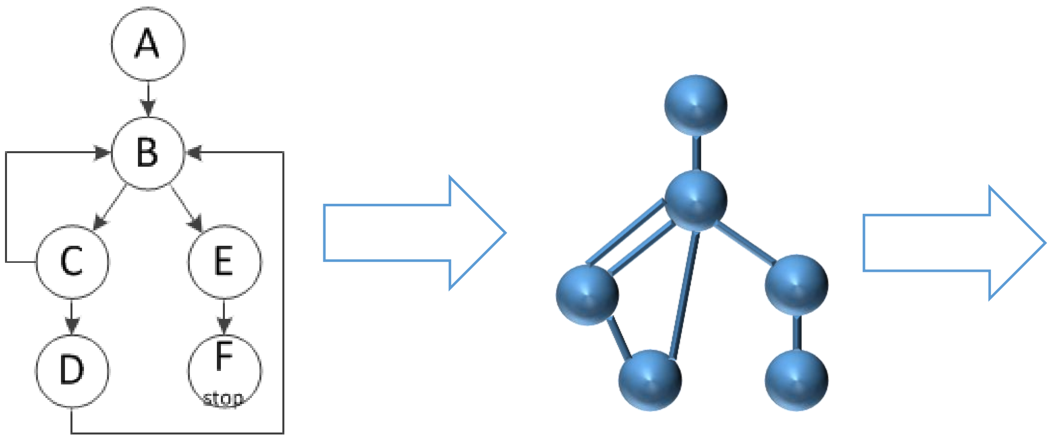


- Challenge 1: How to quickly compare two apps and find the different methods?
- Challenge 2: Are the different methods malicious?

Diff Analysis

- Challenge 1: How to quickly compare two apps and find the different methods?
- Centroid on methods:

Control flow graph (CFG)→3D-CFG→m-core

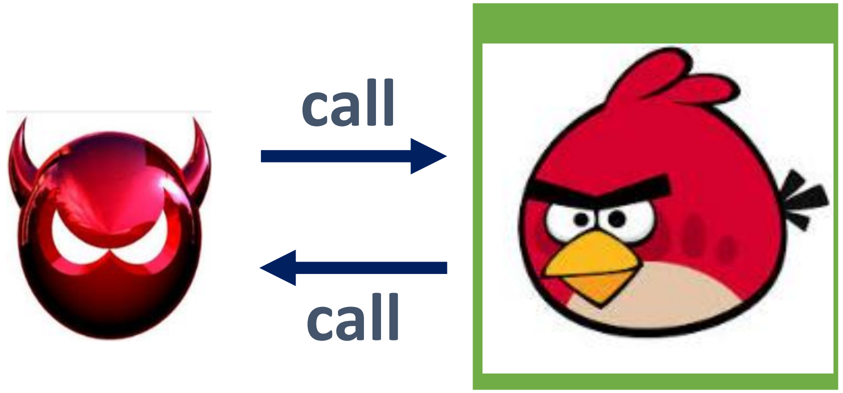


m-core

<x,y,z,w>

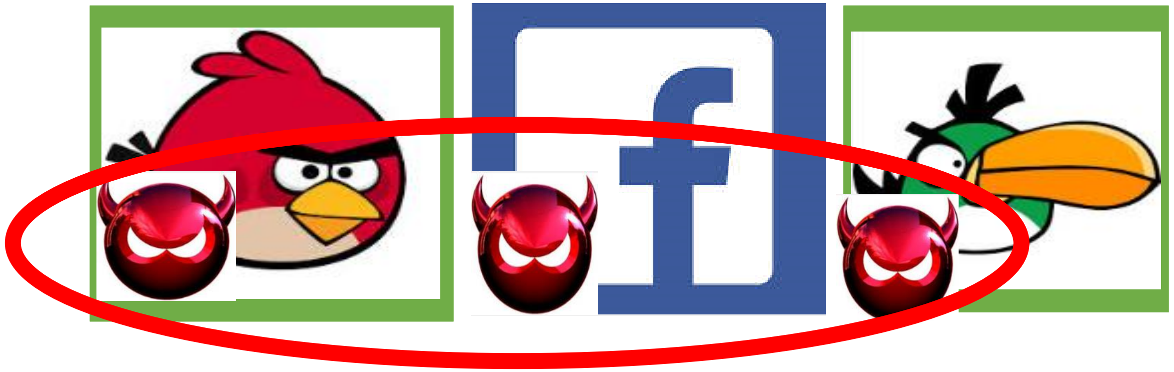
Diff Analysis

- Challenge 2: Are the different methods malicious?
  - Ads and other libraries
  - Updated code (from the same author)
  - Unharmful code
- Solution
  - White-list of libraries
  - Stand-alone analysis
  - Sensitive APIs
    - e.g., GetSimSerialNumber
    - Avoid heavy-weight information flow analysis



Com Analysis

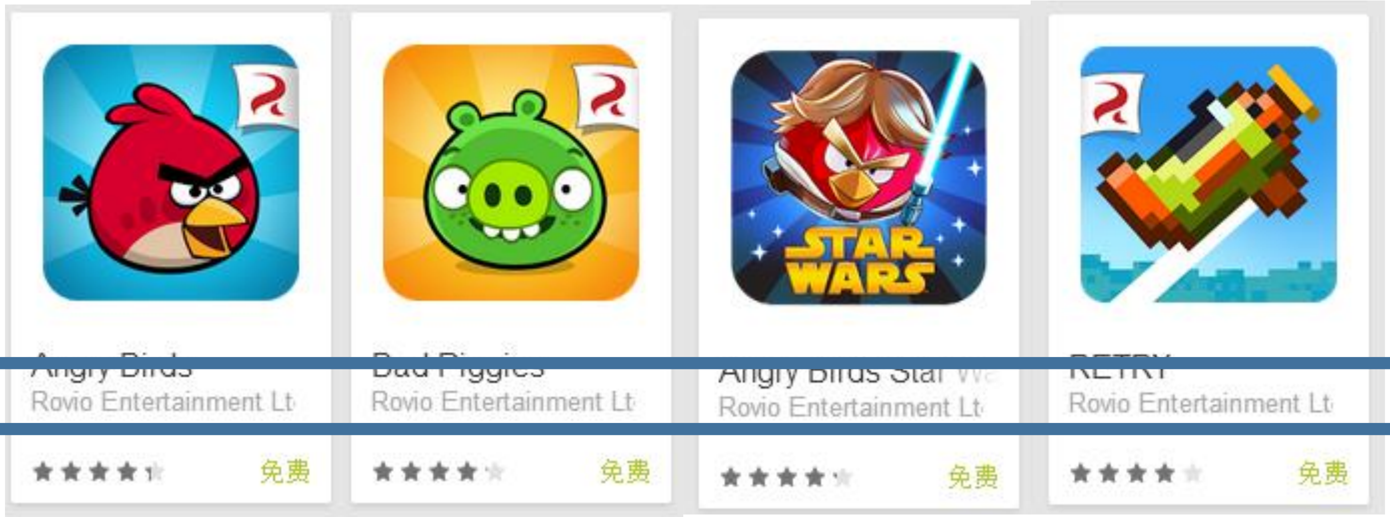
- For the apps with different views, find the common code



- Challenge 1: Are the two apps really unrelated?
- Challenge 2: Is the common code really malicious?

Com Analysis

- Challenge 1: Is the two apps really unrelated?
- Correlation check
  - Similar ideas with “Diff”



Rovio Entertainment

Com Analysis

- Challenge 2: Is the common code really malicious?
  - Library code: Ads, third-party libraries
  - Code reuse: templates
- Approach
  - White-listing popular libraries
  - Training set: the methods not viewed as malicious by virustotal
- Report suspicious code: the method with dangerous APIs

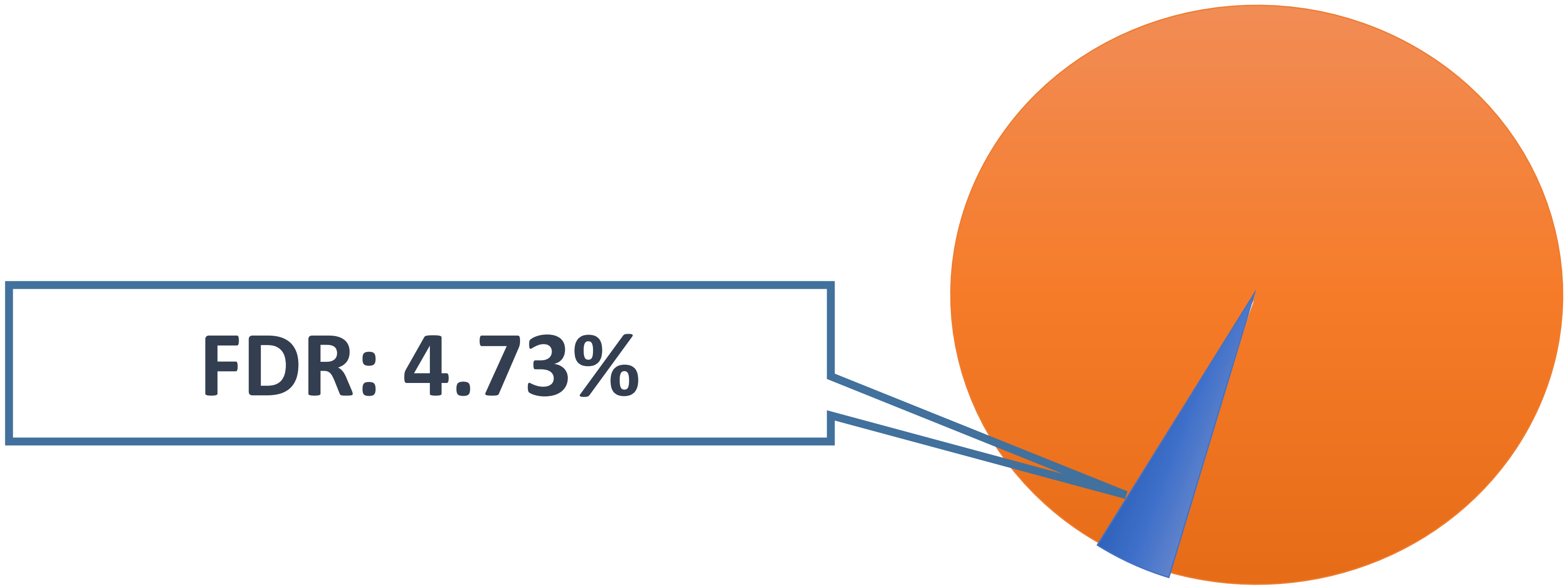
Measurement – Scale of study

- Total apps collected : 1.2 million apps
  - Duplicates removed using MD5
- App markets covered : 33
- # of apps collected from different markets and region
  - GooglePlay : 400,000+ apps
  - Chinese app markets : 596,437 apps
  - European app markets : 61,866 apps
  - Other US stores : 27,047 apps

Appstore	# of malicious apps	# of total apps studied	Percentage	Country
Anzhi	17921	46055	38.91	China
Yidong	1088	3026	35.96	China
yy138	828	2950	28.07	China
Anfen	365	1572	23.22	China
Slideme	3285	15367	21.38	US
AndroidLeyuan	997	6053	16.47	China
gfun	17779	108736	16.35	China
16apk	4008	25714	15.59	China
Pandaapp	1577	10679	14.77	US
Lenovo	9799	68839	14.23	China
Haozhuo	1100	8052	13.66	China
Dangle	2992	22183	13.49	China
3533_world	1331	9886	13.46	China
Appchina	8396	62449	13.44	China
Wangyi	85	663	12.82	China
Youyi	408	3628	11.25	China
Nduo	20	190	10.53	China
Sogou	2414	23774	10.15	China
Huawei	148	1466	10.1	China
Yingyongbao	272	2812	9.67	China
AndroidRuanjian	198	2308	8.58	China
Anji	3467	41607	8.33	China
AndroidMarket	1997	24332	8.21	China
Opera	4852	61866	7.84	Europe
Mumayi	6129	79594	7.7	China
Google	30552	401549	7.61	US
Xiaomi	832	12139	6.85	China
others	2377	38648	6.15	China
Amazon	59	1001	5.89	US
Baidu	831	21122	3.93	China
7xiazi	898	26195	3.43	China
Liqu	394	26392	1.49	China
Gezila	30	5000	0.6	China

Measurement – False Positive

- Flagged apps by MassVet : 127,429 apps (10.93%)
- **FDR** (false-positive VS all detected) : 4.73%
- **FPR** (false-positive VS all apps analyzed) : < 1%
- Manually studied: 20/40 malware



Measurement – Coverage

- 2700 Randomly sampled apps
  - Virustotal: 281 apps
  - MassVet: 197 apps (70.11%)
  - NOD32: 171 apps (60.85%)
  - McAfee: 45 apps (16.01%)
  - 21 apps (11%) apps missed by Virustotal

AV Name	# of Detection	% Percentage
Ours (MassVet)	197	70.11
ESET-NOD32	171	60.85
VIPRE	136	48.40
NANO-Antivirus	120	42.70
AVware	87	30.96
Avira	79	28.11
Fortinet	71	25.27
AntiVir	60	21.35
Ikarus	60	21.35
TrendMicro-HouseCall	59	21.00
F-Prot	47	16.73
Sophos	46	16.37
McAfee	45	16.01
DrWeb	45	16.01
Baidu-International	44	15.66
AVG	40	14.23
Comodo	32	11.39
Cyren	29	10.32
F-Secure	22	7.83
AhnLab-V3	20	7.12
Tencent	16	5.69
Symantec	15	5.34
Alibaba	15	5.34
Commtouch	13	4.63
GData	10	3.56

Measurement – Performance

- A server with 260 GB memory, 40 cores at 2.8 GHz and 28 TB hard drives
- 9 seconds from the submission of the app to the completion of the whole process on it.

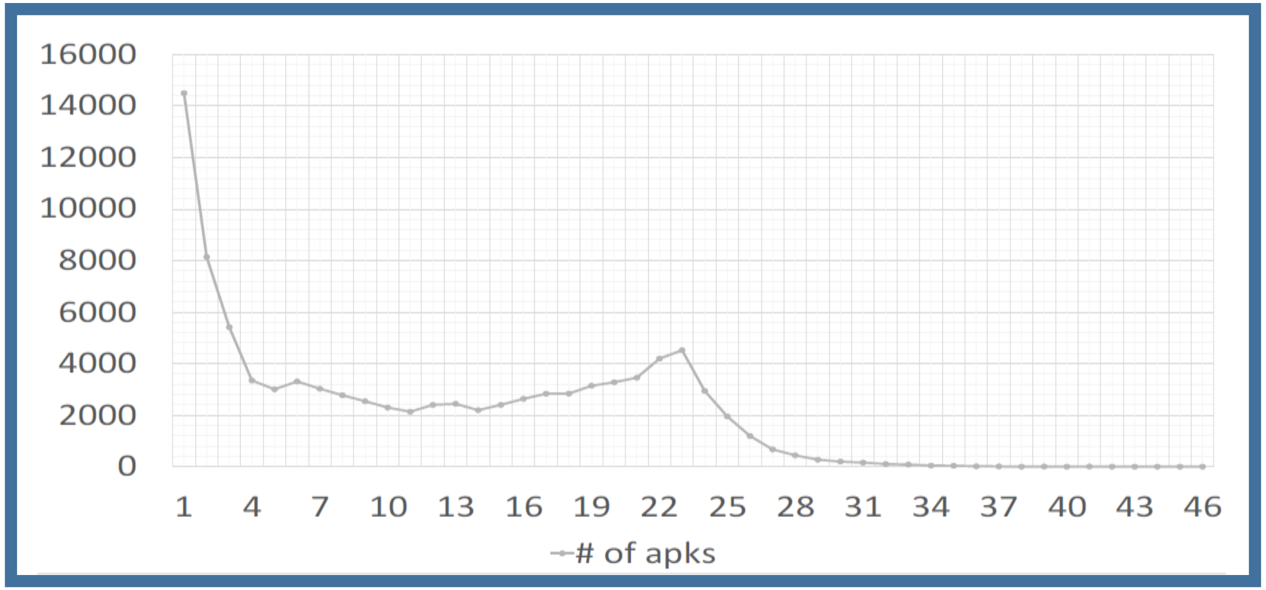
9.95 seconds

500 apps concurrent

# Apps	Pre-Processing analysis	v-core database search	differential	m-core database search (Intersection)	sum
10	5.84	0.15	0.33	1.80	8.12
50	5.85	0.15	0.34	1.99	8.33
100	5.85	0.14	0.35	2.23	8.57
200	5.88	0.16	0.35	3.13	9.52
500	5.88	0.16	0.35	3.56	9.95

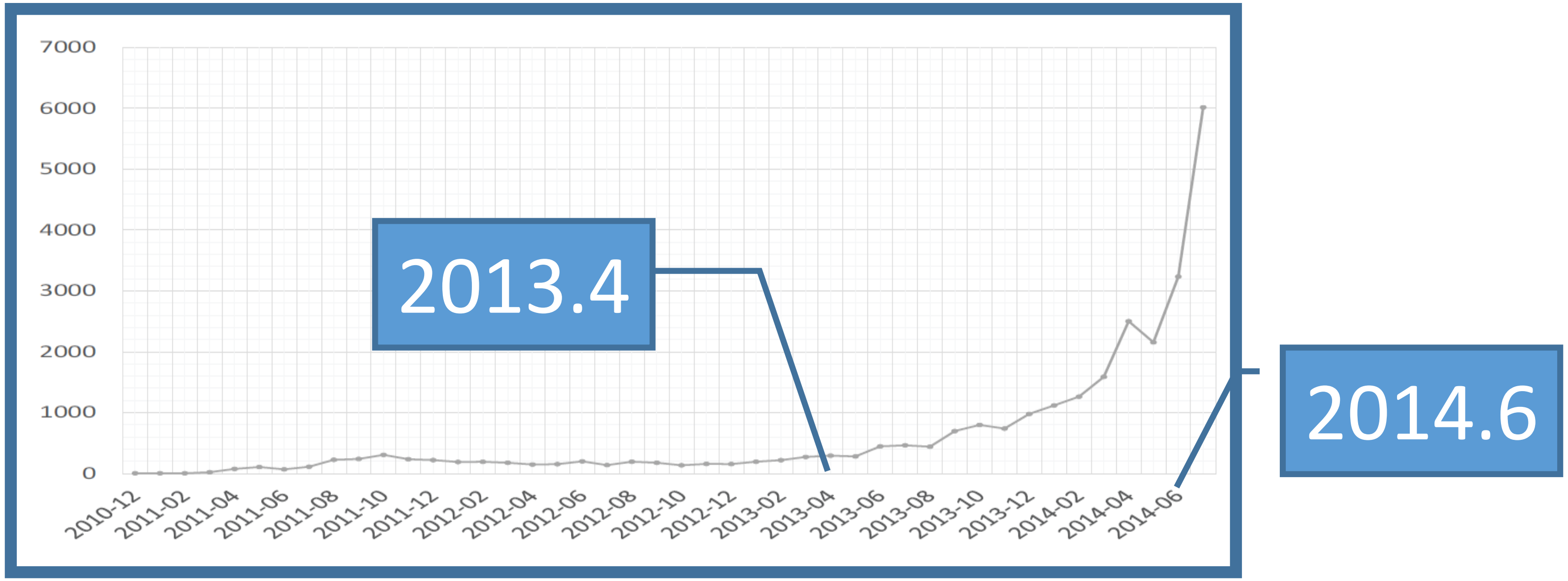
Measurement – Landscape

- 35,473 (north America), 4,852 (Europe), 87,104 (Asia)
- Apps from Google Play: 7.61% are potentially harmful
- Virustotal confirmed 91,648 malware
  - 4.1% were alarmed by at least 25 out of 54 scanners



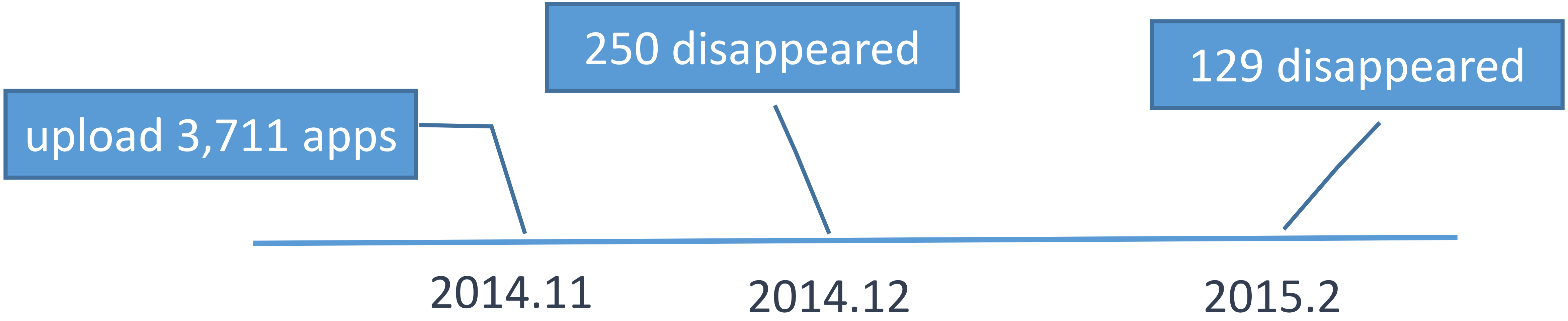
Measurement – Existing defense

- Existing defense: Google Play indeed makes effort to mitigate the malware threat
- Most malware we discovered were uploaded in the past 14 months



Measurement – Disappeared apps

- After uploading 3,711 apps to Virustotal (scan mode)
  - **40 days later:** 250 of them disappeared
  - **90 days later:** another 129 apps disappeared
  - Among the 379 disappeared apps, 54 apps (14%) are detected by Virustotal

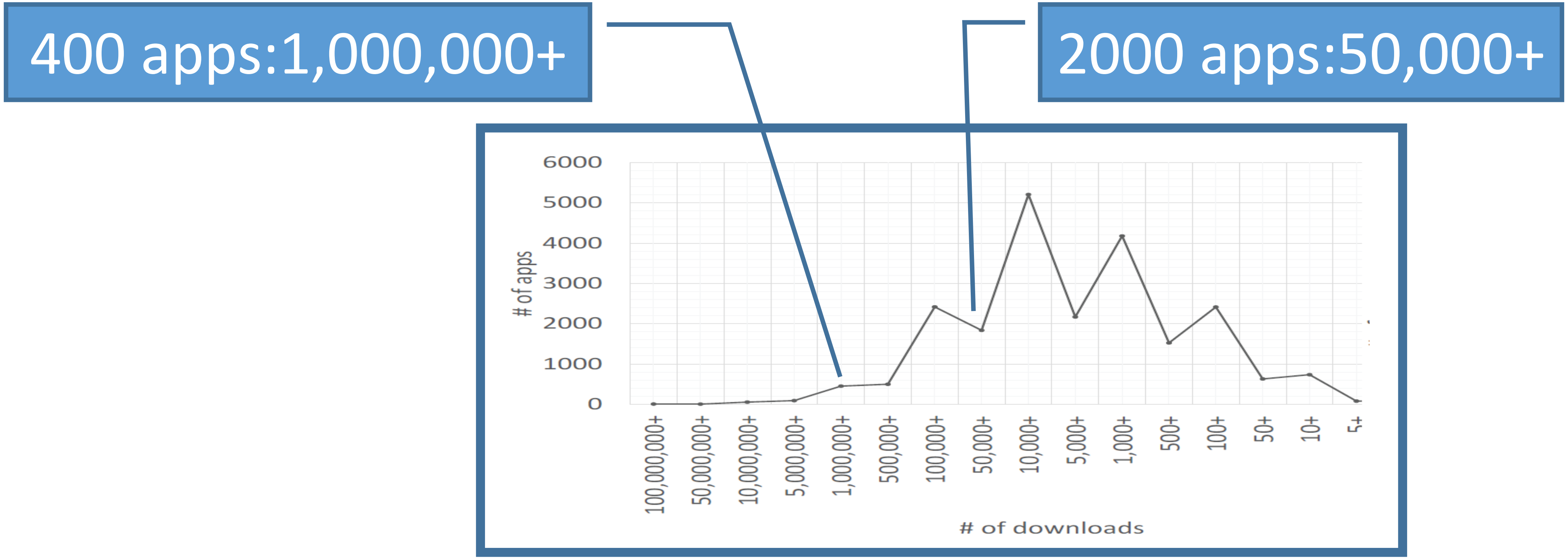


Measurement – Disappeared apps

- Track 2,265 developers of the 3,711 apps (2014/11~2015/02)
  - Additional 2014 apps disappeared (all detected by MassVet)
  - We did **NOT** check them by virustotal
    - Google Play also looked into their common malicious components under the same developers, but not across the whole market (may take long time).
    - Our work is just the one can help them (in several seconds).
- Reappeared apps
  - 604 confirmed malware (28.4%) showed up in Google Play **unchanged**
  - 829 apps showed up using **different names**

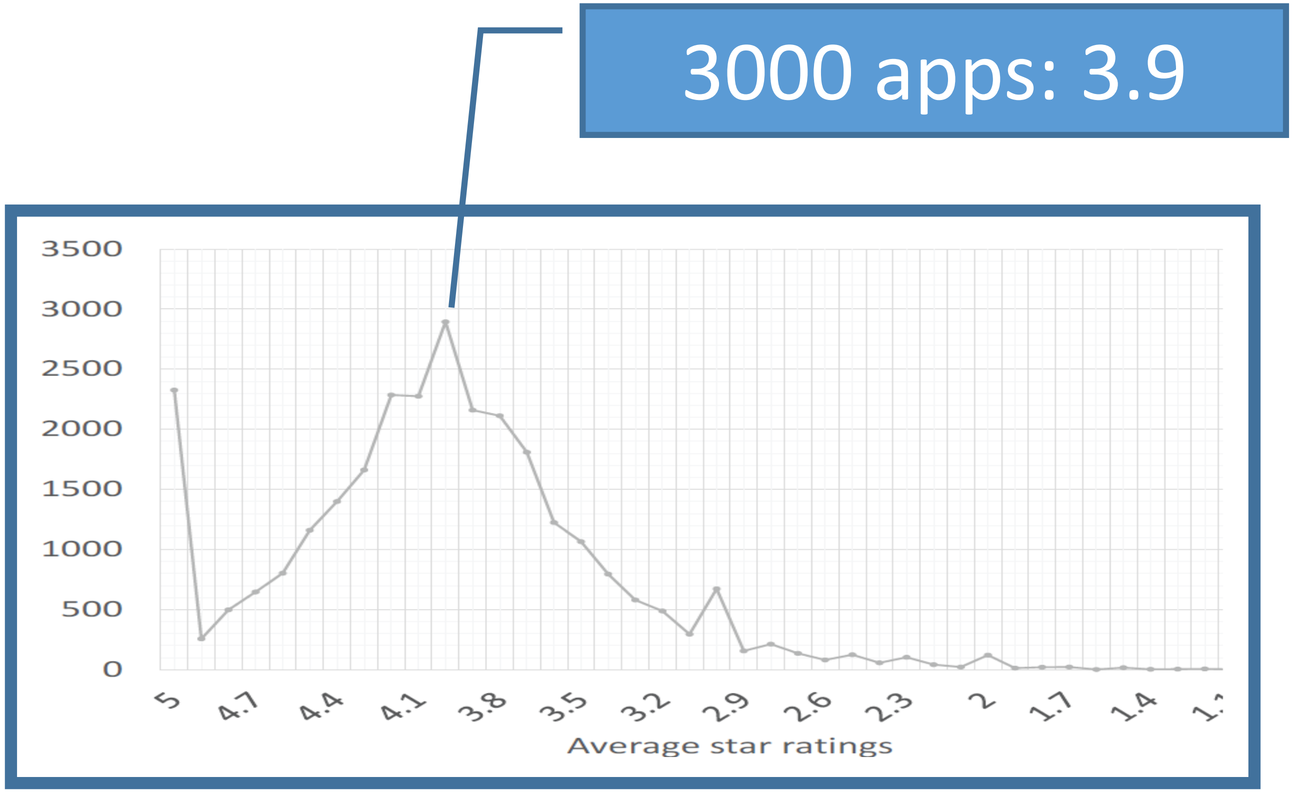
Measurement – Impact

- Distribution of downloads for malicious or suspicious apps in GooglePlay



Measurement – Impact

- The distribution of rating for malicious or suspicious apps in GooglePlay



Measurement – Signatures

- Top 5 signatures used in apps

1644

1258

Signature	# of malicious apps
c673c8a5f021a5bdc5c036ee30541dde	1644
a2993eaecf1e3c2bcad4769cb79f1556	1258
3be7d6ee0dca7e8d76ec68cf0ccd3a4a	615
f8956f66b67be5490ba6ac24b5c26997	559
86c2331f1d3bb4af2e88f485ca5a4b3d	469

Measurement – Identities

- Top 5 signatures used by different identities

604

447

Signature	# of different identities
02d98ddfbcd202b13c49330182129e05	604
a2993eaecf1e3c2bcad4769cb79f1556	447
82fd3091310ce901a889676eb4531f1e	321
9187c187a43b469fa1f995833080e7c3	294
c0520c6e71446f9ebdf8047705b7bda9	145

Conclusion

- **We propose a new technique for efficient vetting of apps for unknown malware**
  - Compare an app with all other apps on a market (DiffCom Analysis)
  - Light-weight code analysis compared with other approaches
- **We implemented MassVet and apply it to analyze 1.2 million apps.**
- **MassVet found 127,429 malware (20 likely to be zero days)**

MassVet Available Now

http://www.appomicsec.com

Thank You! Questions