Apposcopy: Automated Detection of Android Malware

{ Yu Feng, Isil Dillig }@UT Austin {Saswat Anand, Alex Aiken}@Stanford University

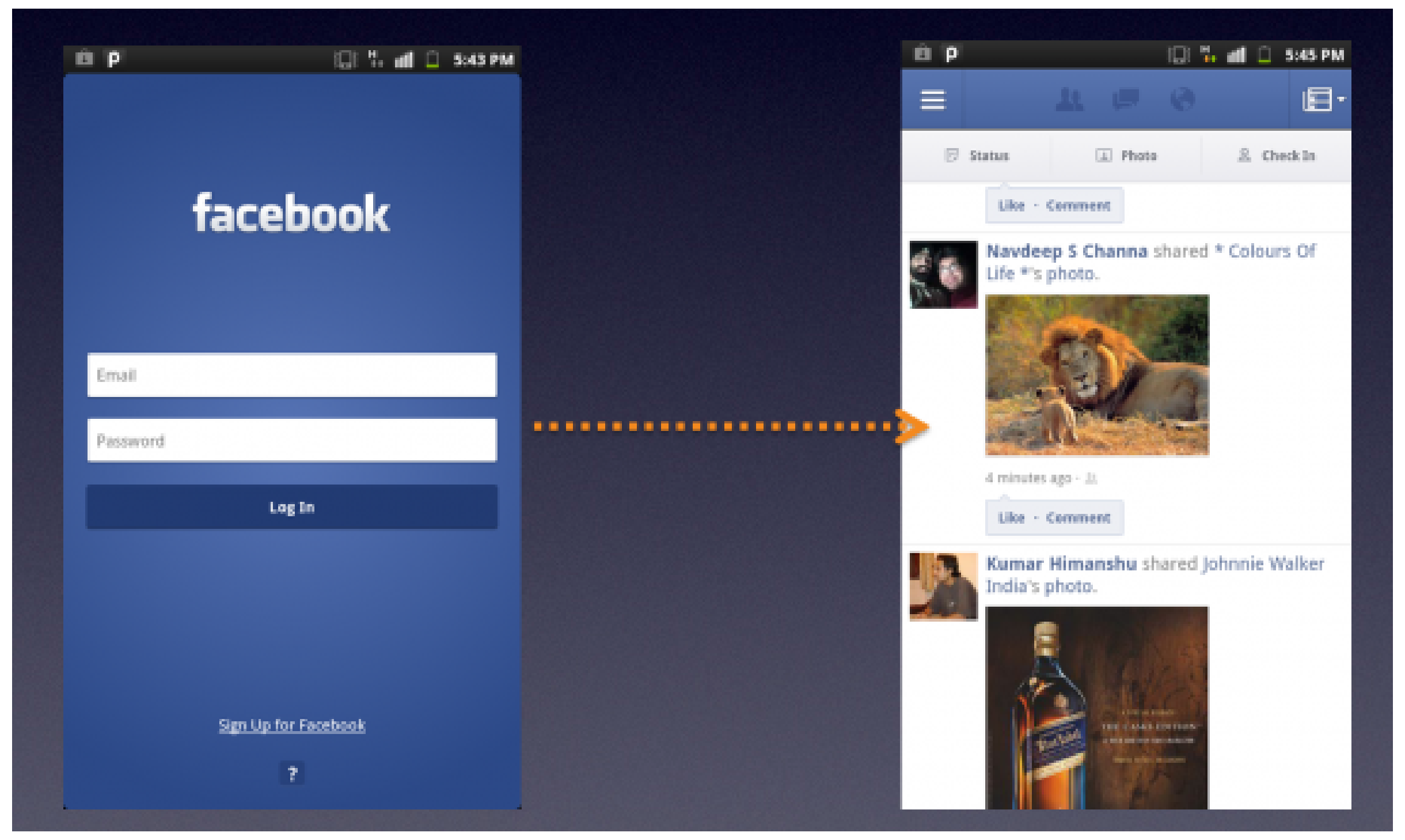
Problem

The Android platform is a growing target for mobile malware. Today, many of the malicious applications that afflict Android users exploit the private and monetized information stored in a user's smartphone.

Two major existing approaches and their disadvantages.

1. Taint analyses: Could generate a lot of false positives without context;
2. Signature-based malware detectors: Classify malware based on sequences of low-level instructions. Compromised by common obfuscations.

Android Background



There are four types of components in Android and they communicate with each other through the *Intent* object.

1. Activities form the basic user interface;
2. Service components run in the background even if windows are switched;
3. BroadcastReceiver components react asynchronously to messages from other apps;
4. ContentProviders store data for the app.

Contributions

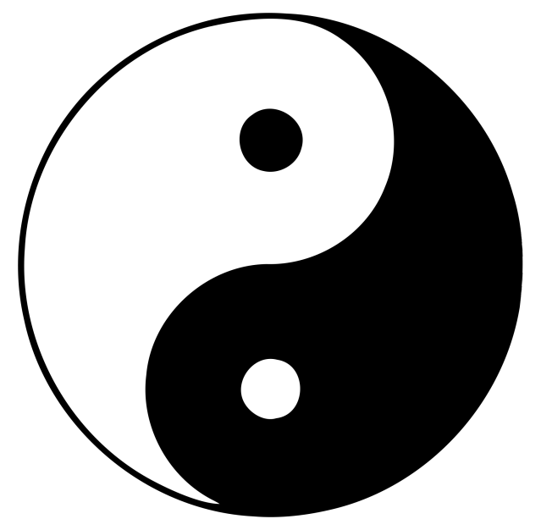
We design a high-level signature language for describing semantic characteristics of Android malware families. Such as:

1. Control-flow properties
2. Data-flow properties

We perform powerful static analysis for deciding if a given app matches signature of a malware family. In order to detect malware precisely,

1. We use a hybrid pointer analysis for the taint analyses;
2. We build a precise Inter-component Call graph, which is our own high-level abstraction for Android apps.

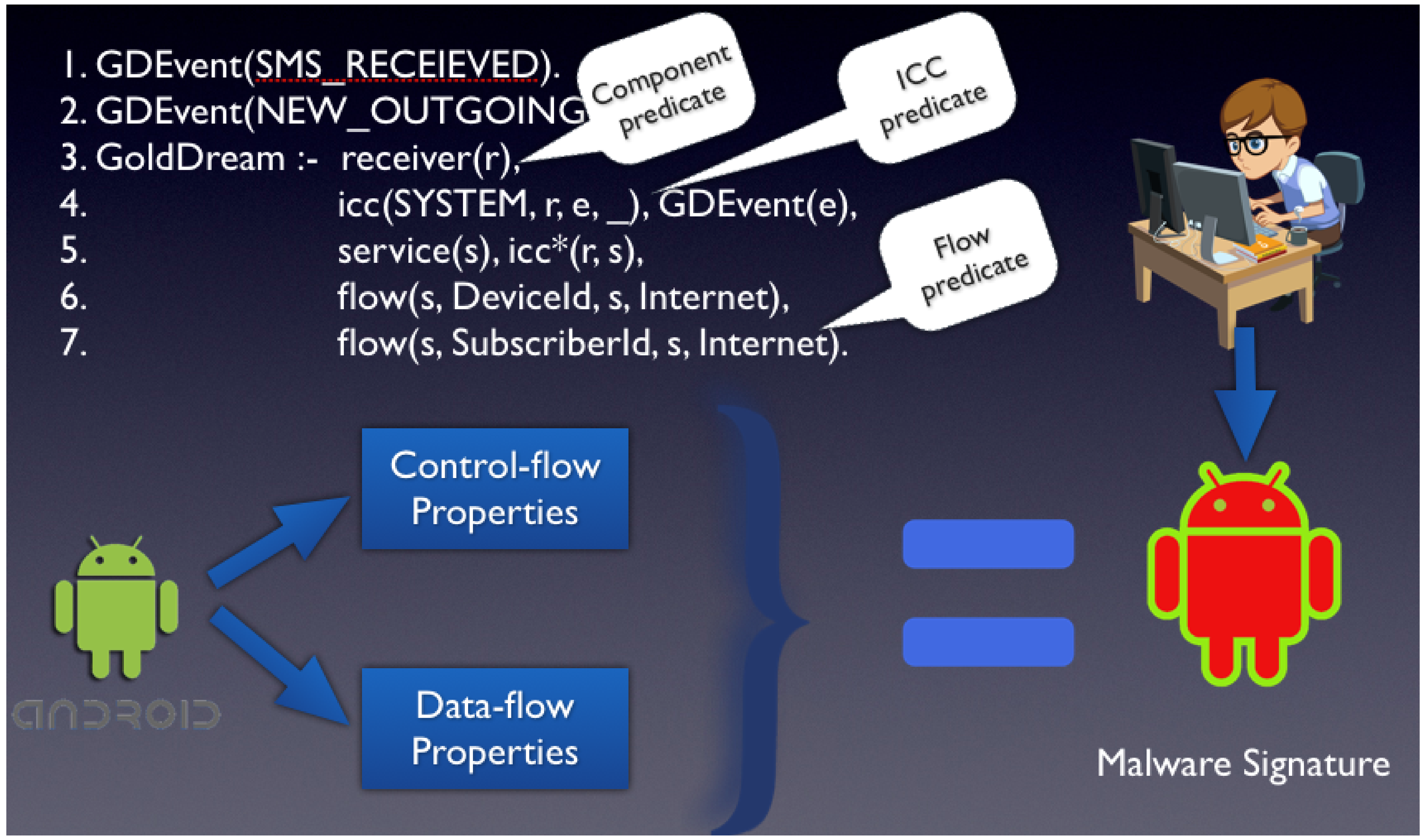
Methodology



We combine the advantages of taint analyses and signature-based techniques and overcome their disadvantages.

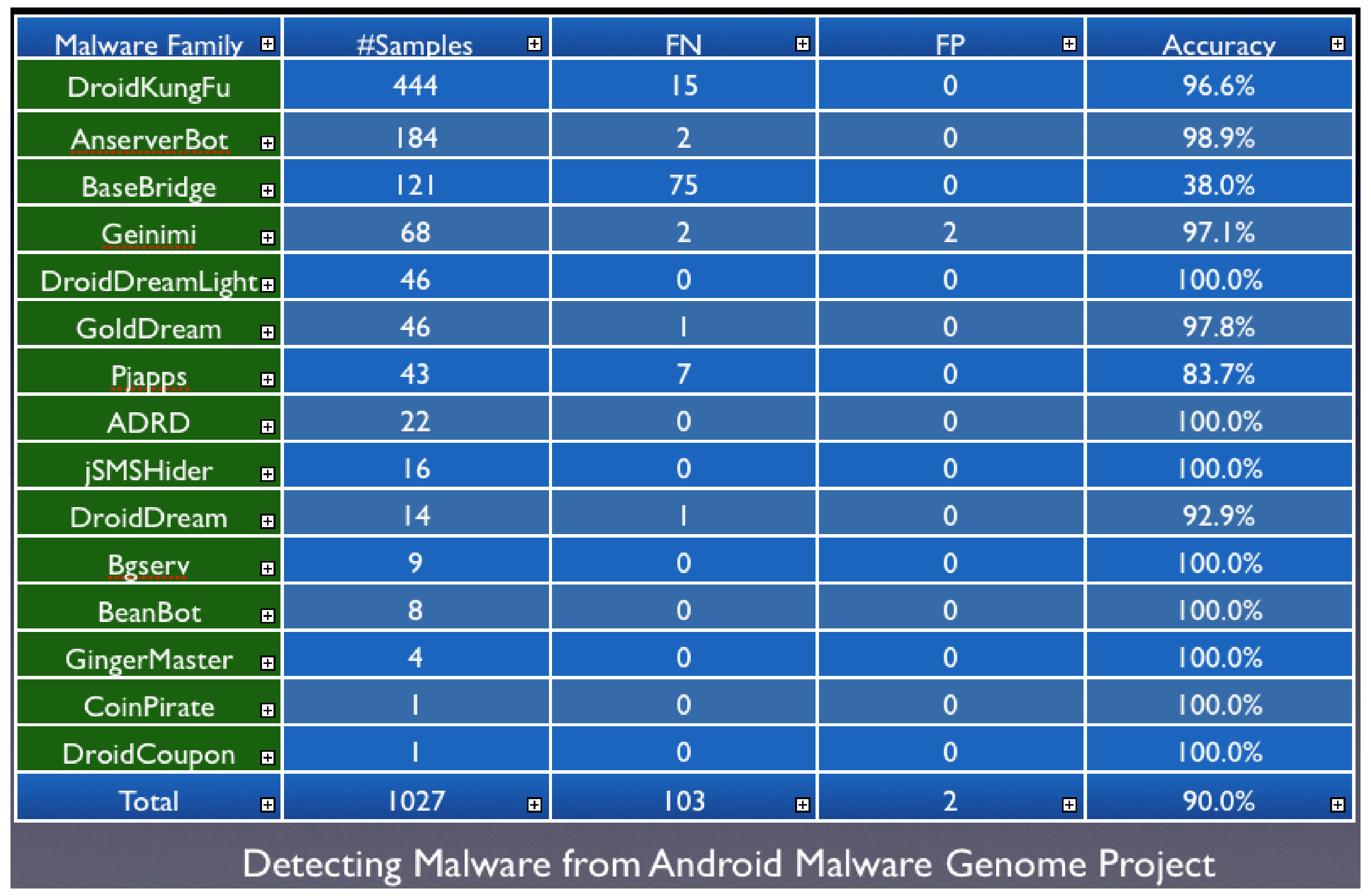
1. Represent a corpus of malware through semantic signatures in terms of data-flow and control flow properties.
2. We extract the data-flow properties of an app by performing taint analyses;
3. We extract the control-flow properties of an app by constructing its Inter-Component Call Graph(ICCG), which is a high-level representation for Android apps.
4. Decide if the extracted control and data-flow properties of the app match any malware signatures in database.

Our approach by example(GoldDream malware)



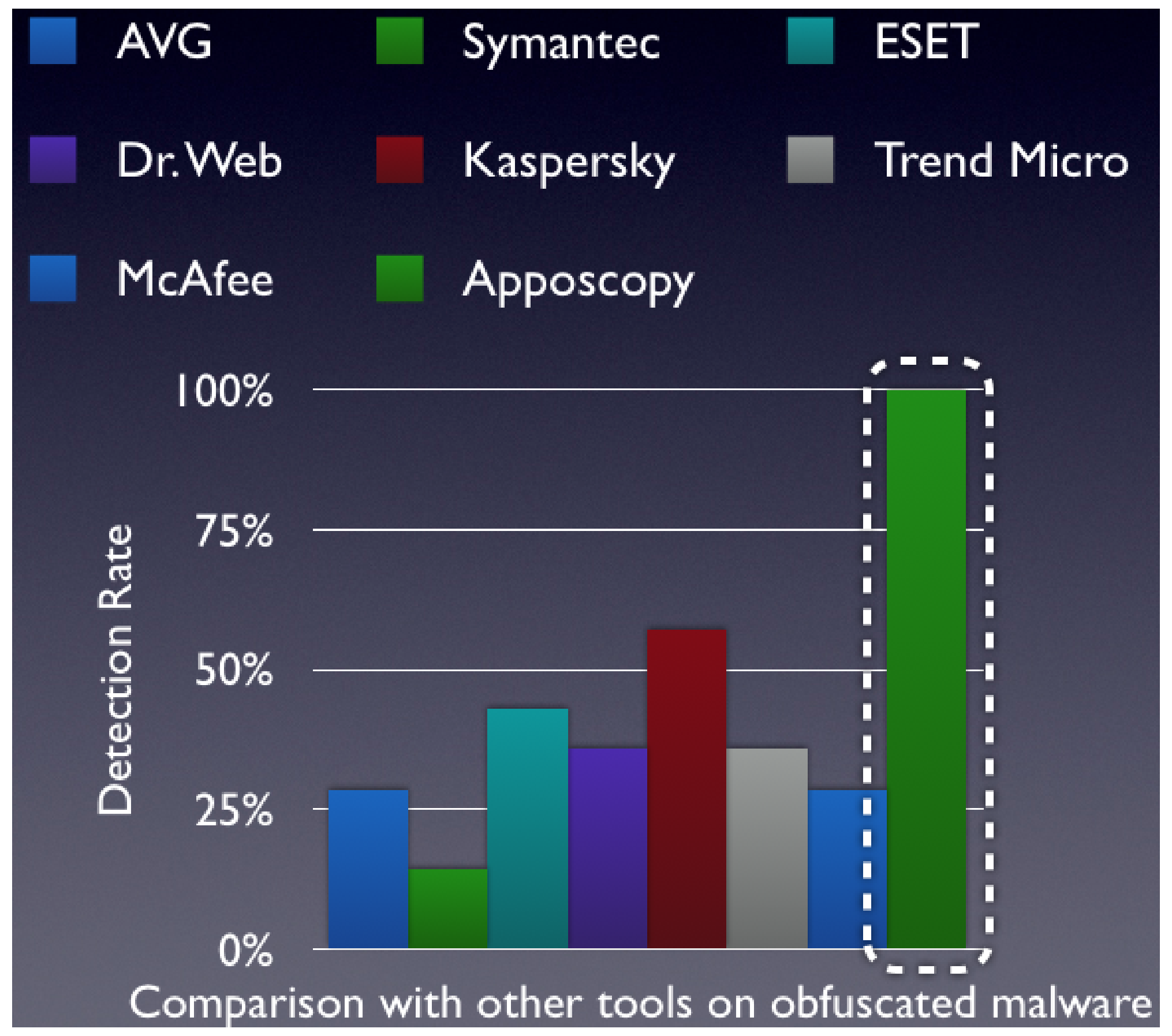
```
1. GDEvent(SMS_RECEIEVED).
2. GDEvent(NEW_OUTGOING
3. GoldDream :-  receiver(r),
4.               icc(SYSTEM, r, e, _), GDEvent(e),
5.               service(s), icc*(r, s),
6.               flow(s, DeviceId, s, Internet),
7.               flow(s, SubscriberId, s, Internet).
```

Results



Malware Family	#Samples	FN	FP	Accuracy
DroidKungFu	444	15	0	96.6%
AnserverBot	184	2	0	98.9%
BaseBridge	121	75	0	38.0%
Geinimi	68	2	2	97.1%
DroidDreamLight	46	0	0	100.0%
GoldDream	46	1	0	97.8%
Pjapps	43	7	0	83.7%
ADRD	22	0	0	100.0%
jSMSHider	16	0	0	100.0%
DroidDream	14	1	0	92.9%
Bgserv	9	0	0	100.0%
BeanBot	8	0	0	100.0%
GingerMaster	4	0	0	100.0%
CoinPirate	1	0	0	100.0%
DroidCoupon	1	0	0	100.0%
Total	1027	103	2	90.0%

Detecting Malware from Android Malware Genome Project



Comparison with other tools on obfuscated malware

References

[1] Y Feng, S Anand, I Dillig, A Aiken. Apposcopy: Semantics-based detection of android malware through static analysis In *SIGSOFT FSE,2014*

[2] Y Feng, S Anand, I Dillig, A Aiken. Apposcopy: automated detection of Android malware (invited talk). In *DeMobile 2014*

Future Work

We will develop techniques to improve the efficiency and precision of Apposcopy's static analyses. We also plan to develop techniques to automatically de-obfuscate apps to enhance Apposcopy's resilience to some types of obfuscations.

Finally, we plan to develop techniques to automatically learn malware signatures from a set of apps labeled with their corresponding malware family (or as benign).

Acknowledgments

This material is based on research sponsored by the Air Force Research Laboratory, under agreement number FA8750-12-2-0020.