A Summary of **TriFlow**: Triaging Android Applications using Speculative Information Flows

Omid Mirzaei, Guillermo Suarez-Tangil, Juan Tapiador, Jose M. de Fuentes

uc3m | Universidad Carlos III de Madrid

UCL

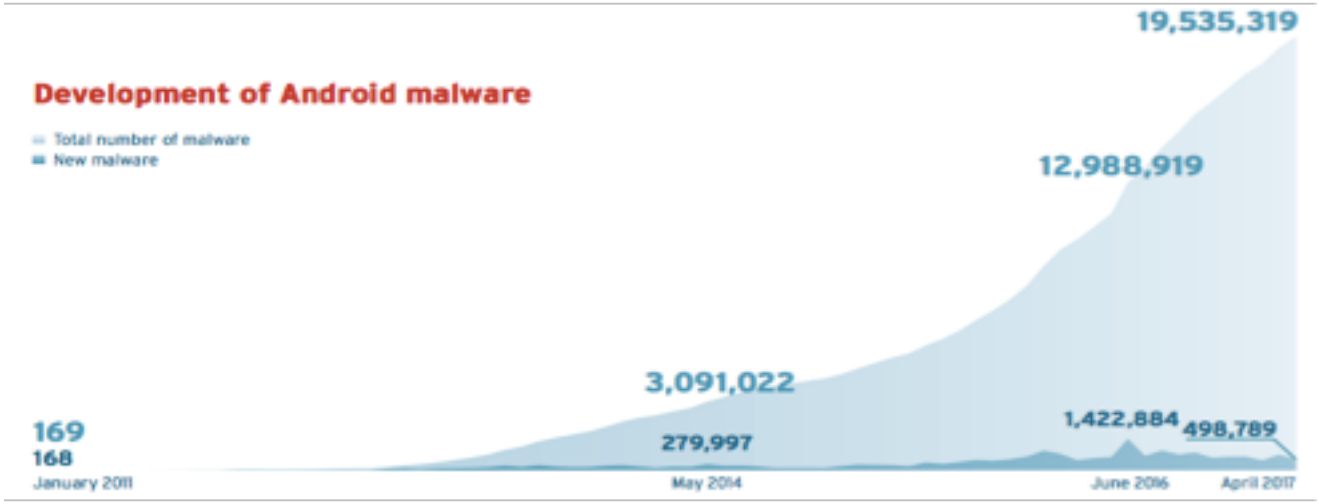
IV Jornadas Nacionales de Investigación en Ciberseguridad

JNIC 2018

3rd place at CSAW-Eruope'17 Best Applied Security Research Competition

Motivation

- Android OS market share
- Malicious and Potentially Harmful Apps (PHAs)
  - 5,730,916 malicious installation packages
  - 94,368 mobile banking Trojans
  - 544,107 mobile ransomware Trojans
- Limitations in the analysis tools



Outline

- **Background**
- TriFlow Overview
- Results
- Conclusions

Background
Info-flows

- Information Flow Analysis
- How to extract flows?
  - Static Analysis (e.g. FlowDroid)
  - Dynamic Analysis (e.g. TaintDroid)
- Limitations of info-flow analysis techniques:
  - Accuracy (false positives, false negatives)
  - Scalability (memory, time)

Background
Risk metrics

- Info-flows are good to study the behavior of apps
- What if we rely upon flows to measure the risk posed by an app?
  - Identifying apps with potentially dangerous behaviors
- What are the applications?
  - Communicate risk to final users
  - Triage
- Previous risk metrics:
  - Meta-data from Manifest file (permissions and components)
  - Market data (review score, size, developer reputation)
  - Features in the source code (API methods)
  - Others (Apps' behavior, network usage, root exploits)

Background
Main contributions

- Info-flow based risk scoring mechanism
- Key Features:
  - Notion of speculative info-flows (predicting the presence of flows)
  - Fast: Applications to triage
  - Complement to other analysis tools
- Tool:
  - **TriFlow** is publicly available at:

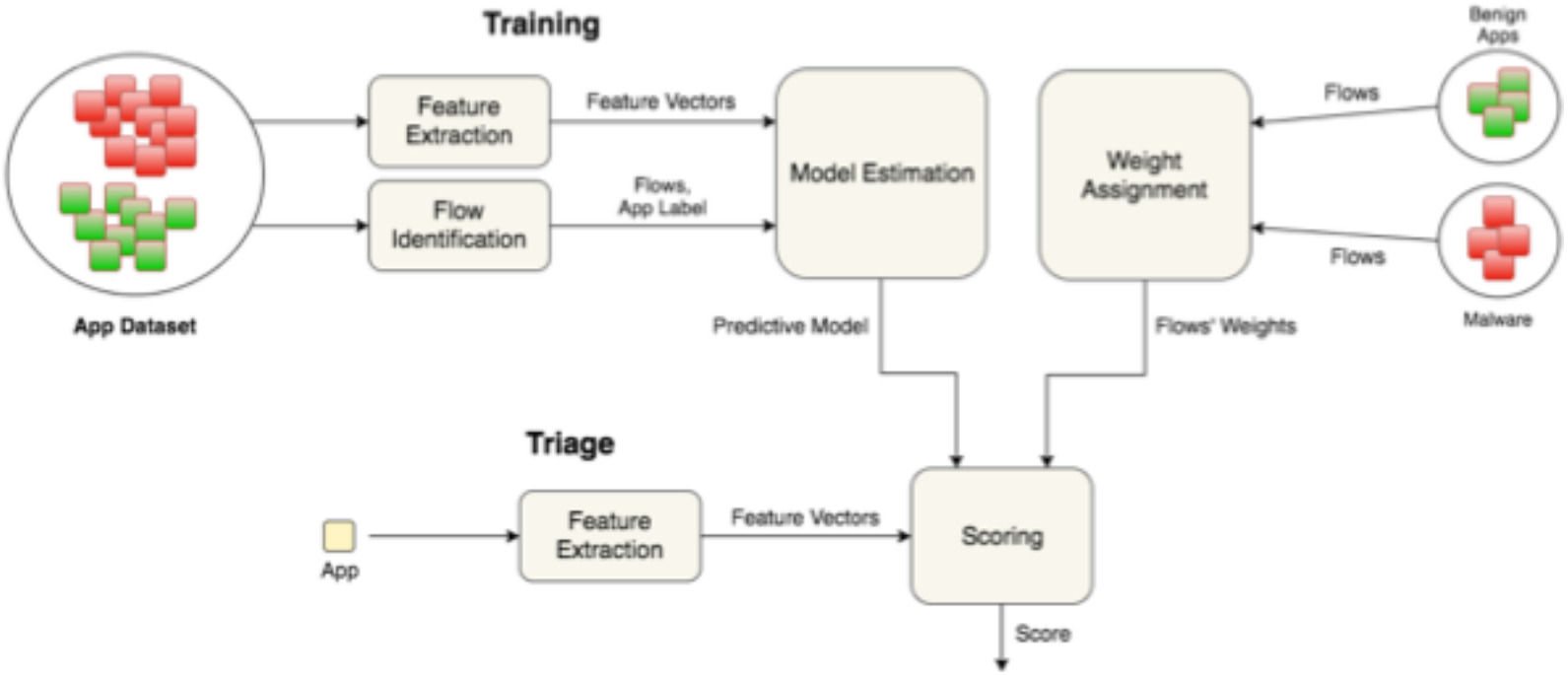
https://github.com/OMirzaei/TriFlow

Outline

- Background
- **TriFlow Overview**
- Results
- Conclusions

TriFlow Overview

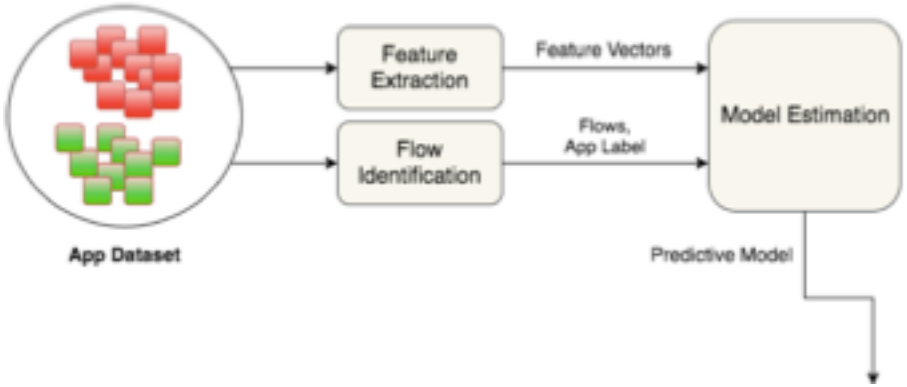
Overall System



$$R = \sum_{j=1}^{N} P_j * I_j \qquad j = 1, 2, 3, \dots, N \ (Total\ number\ of\ flows)$$

TriFlow Overview
Prediction of flows

- Key idea:
  - Static features whose presence correlates with the presence of flows
- Creating a predictive model:
  - Using a dataset of apps
  - Needs ground truth, i.e. actual flows for each app (FlowDroid)
  - Estimating **P(flow | features)**



TriFlow Overview
Weighting flows

- Goals:
  - Identifying “malicious” info-flows:

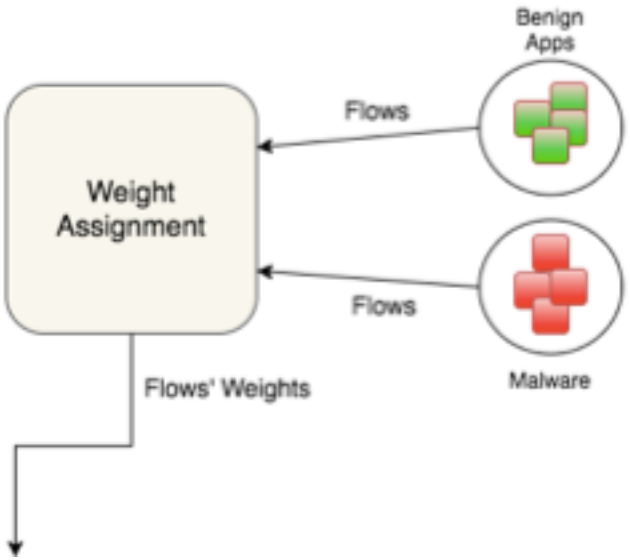
    Flows appearing mostly in malware but not in benign apps

  - Filtering out info-flows which are common in both classes
- $I(f) = -P_M(f) \log_2 P_B(f)$

$P_M(f)$: Frequency of flow in malware

$P_B(f)$: Frequency of flow in Benign apps

$I(f)$: Weight of flow (Maliciousness)



Outline

- Background
- TriFlow Overview
- **Results**
- Conclusions

Results
Datasets

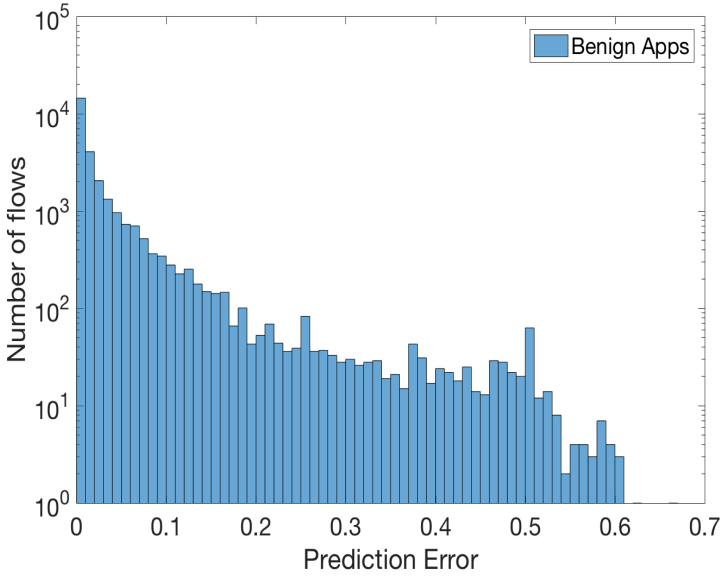
Type	Dataset	Type	Samples
Malware (MW)	Drebin [4]	Malware	5,560
Goodware (GW)	Google Play	Goodware	11,456
		Total	17,016

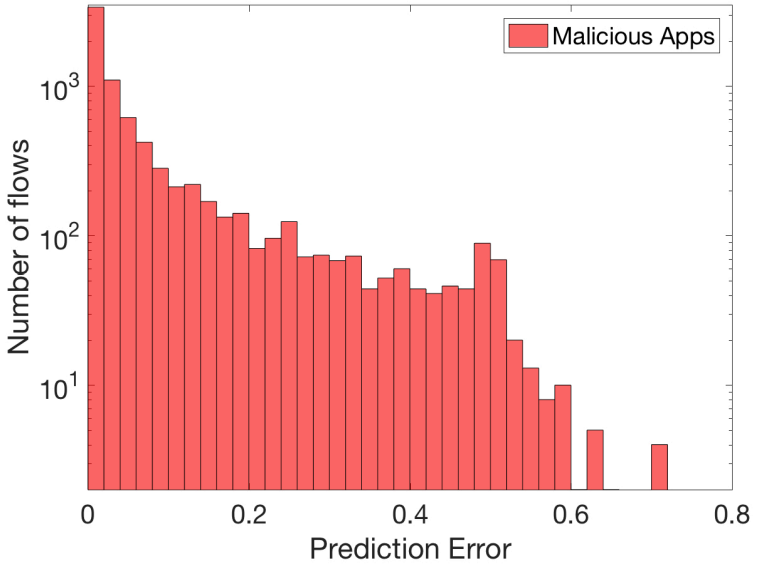
Mode	Split	Ratio	Samples
Modeling (Training)	4,000 MW 4,000 GW	1:1	8,000
Triage (Testing)	1,560 MW 7,456 GW	1:5	9,016

Results

Prediction of flows

Dataset	Mean	Std. Dev.	Median
Drebin	0.0861	0.1272	0.0278
GooglePlay	0.0361	0.0734	0.0094
All	0.0376	0.0784	0.0089





1.04% of flows are predicted with error = 0
90% of flows are predicted with error < 0.25

4.31% of flows are predicted with error = 0
83% of flows are predicted with error < 0.1
90% of flows are predicted with error < 0.25

Results
Weighting flows

- In 75% of flows: $I(f) = 0$
- In almost 25% of flows: $0 < I(f) \leq 0.5$
- In almost 1% of flows: $0.5 < I(f) \leq 1$
- In very rare flows: $I(f) > 1$

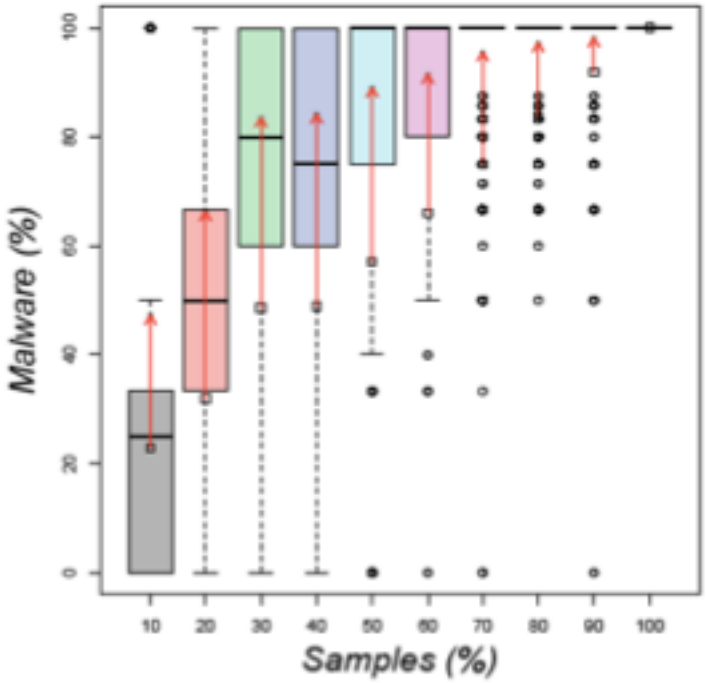
Observed mainly in Malware

Source	Sink	$I(f)$
TM.getDeviceId()	String.startsWith()	0.69
TM.getDeviceId()	OutputStream.write()	0.26
TM.getDeviceId()	Intent.putExtra()	0.52
TM.getDeviceId()	String.substring()	0.28
TM.getDeviceId()	URL.openConnection()	0.37
TM.getSubscriberId()	String.startsWith()	0.88
TM.getSubscriberId()	OutputStream.write()	0.24
TM.getSubscriberId()	HttpURLCon.setRequestMethod()	0.25
TM.getSubscriberId()	URL.openConnection()	0.42
TM.getSubscriberId()	Intent.putExtra()	0.58
TM.getSimCountryIso()	Log.i()	0.37
TM.getSimCountryIso()	String.substring()	0.25
TM.getSimOperator()	Log.v()	0.31
TM.getNetworkOperator()	String.startsWith()	0.32
TM.getNetworkOperator()	String.substring()	1.18
TM.getLine1Number()	URL.openConnection()	0.20
TM.getLine1Number()	Log.v()	0.52
TM.getLine1Number()	String.startsWith()	0.53
TM.getSimSerialNumber()	String.startsWith()	0.98
TM.getSimSerialNumber()	String.substring()	1.09
gsm.SM.getDefault()	gsm.SM.sendTextMessage()	0.82
SM.getDefault()	SM.sendTextMessage()	1.81
NetworkInfo.getExtraInfo()	Log.d()	0.68
NetworkInfo.getExtraInfo()	String.startsWith()	0.45
WebView.getSettings()	WebS.setAllowFileAccess()	0.67
WebView.getSettings()	WebS.setGeolocationEnabled()	0.46
WebView.getSettings()	WebS.setPluginsEnabled()	0.50
System.getProperties()	String.substring()	0.45
PI.getBroadcast()	SM.sendTextMessage()	1.28
HashMap.get()	SM.sendTextMessage()	1.33

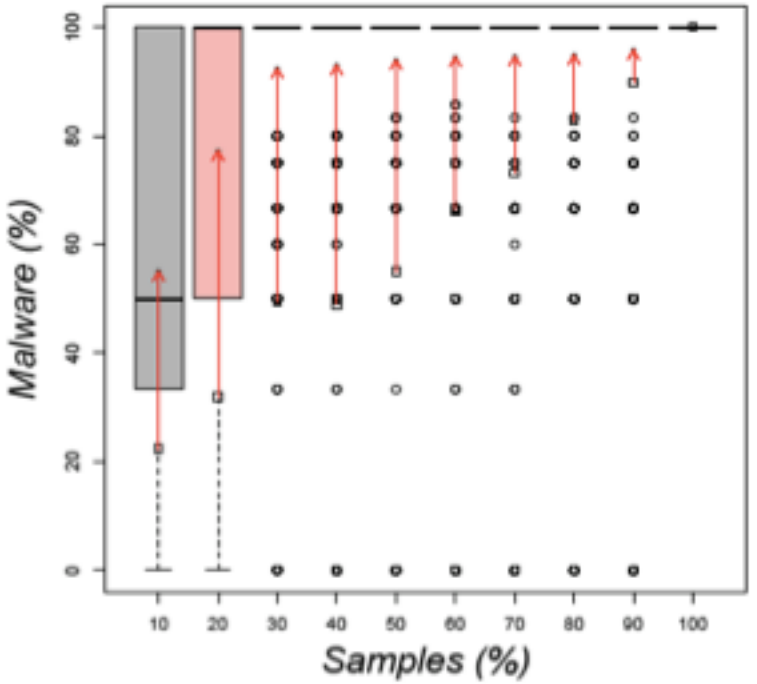
TM: TelephonyManager, SM: SmsManager, PI: PendingIntent, HttpURLCon: HttpURLConnection, WebS: WebSettings.

Results

Scoring and Prioritizing



(a) RSS (Sarma et al. [34]).



(b) TRIFLOW.

Results

Explanation of Scores

(Trackplus family)

Application Name = 55d7e1c74ed3630f7c8d7a892c049aca.apk

Total Score = 8.002e-05

[UNIQUE_IDENTIFIER, LOG] = 2.17e-05 (27.09% of the score)

<android.telephony.TelephonyManager: java.lang.String **getDeviceId()**>,
<android.util.Log: int **i**(java.lang.String,java.lang.String)>, 1.20e-05

<android.telephony.TelephonyManager: java.lang.String **getDeviceId()**>,
<android.util.Log: int **d**(java.lang.String,java.lang.String)>, 8.03e-06

<android.telephony.TelephonyManager: java.lang.String **getDeviceId()**>,
<android.util.Log: int **e**(java.lang.String,java.lang.String)>, 1.58e-06

<android.telephony.TelephonyManager: java.lang.String **getDeviceId()**>,
<android.util.Log: int **w**(java.lang.String,java.lang.String)>, 2.80e-08

[DATABASE_INFORMATION, LOG] = 1.43e-08 (0.018% of the score)

<android.database.sqlite.SQLiteDatabase: android.database.Cursor
query(java.lang.String,java.lang.String[],java.lang.String,java.lang.String
[],java.lang.String,java.lang.String,java.lang.String,java.lang.String)>,
<android.util.Log: int **d**(java.lang.String,java.lang.String)>, 1.43e-08

…

Outline

- Background
- Contributions
- TriFlow Overview
- Results
- **Conclusions**

Conclusions

- **Predicting** info-flows in Android apps
- **Weighting** flows based on **"rare equals risky"** intuition
- A **new risk metric** based on info-flows
- An efficient **triage** system
- A **complement** to static and dynamic analysis tools

Your Questions

Thanks!

Email: omid.mirzaei@uc3m.es
Website: http://www.seg.inf.uc3m.es/~omirzaei/