TriFlow: Triaging Android Applications using Speculative Information Flows

Omid Mirzaei[1*], Guillermo Suarez-Tangil[2], Juan Tapiador[1], Jose M. de Fuentes[1]
[1] *Universidad Carlos III de Madrid*
[2] *University College London*
[*] *Corresponding Author: omid.mirzaei@uc3m.es*

Introduction & Contributions

- **Motivations:**
  1. The current pace of Android app creation makes it unfeasible to analyze all of them in real time.
  2. Limitations of automated (static or dynamic) analysis techniques.
- **Primary Goal:** Developing a lightweight system to identify Android apps with potentially dangerous behaviors.
- **Approach:** An information flow based risk scoring mechanism which is based on two key ideas:
  1. Predicting information flows in Android applications.
  2. Estimating the maliciousness of information flows.

TriFlow Overview

- **Creating a predictive model:** Based on some *static features* whose presence correlates with the presence of flows (API methods).
- **Weighting flows:** $I(f) = -P_M(f) \log_2 P_B(f)$

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

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

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

Table 1: Dataset.

Type	Dataset	No. apps
Malware (MW)	Drebin	5,560
Goodware (GW)	Google Play	11,456
	Total	17,016

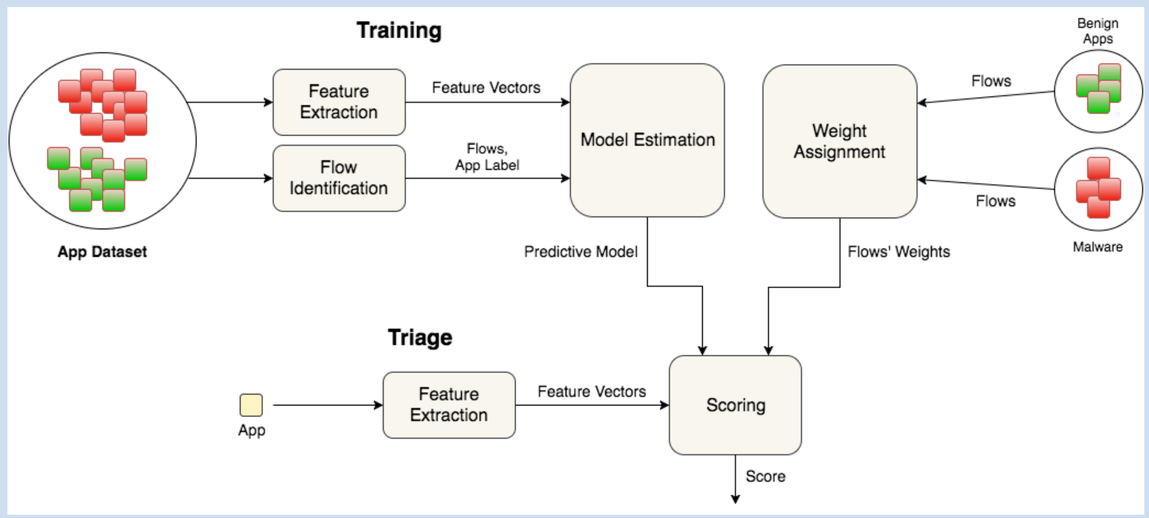


Figure 1: TriFlow architecture for flows prediction and weighting.

Flows Prediction

Logarithmic scale of prediction errors show that 90% of flows are predicted with error < 0.25 in both malware and benign apps.

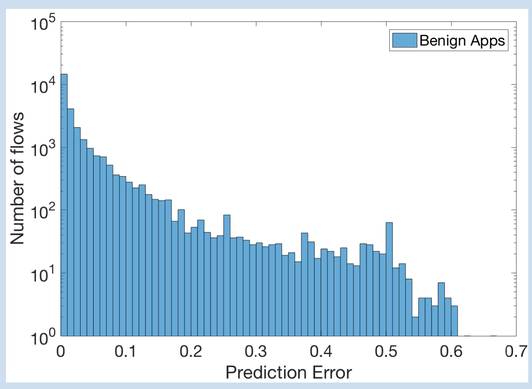


Figure 2: Flow prediction errors in benign apps.

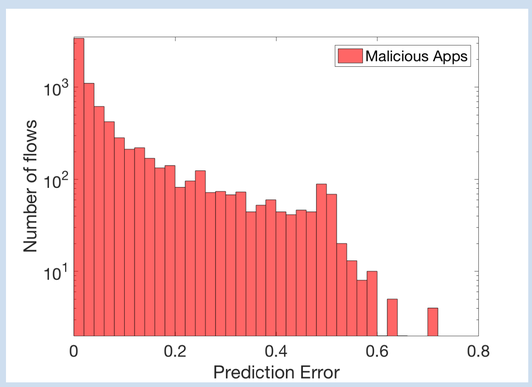


Figure 3: Flow prediction errors in malware.

Table 2: Statistics of 5-fold cross-validation.

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

Flows Weighting & Apps Triage

The higher a flow's weight, the more frequent is that flow in malware than in benign apps.

MEAN

	LOG	FILE	NETWORK	SMS_MMS	AUDIO	NO_CATEGORY	LOCATION_INFORMATION
NETWORK_INFORMATION	0,0369	0,0074	0,0111	0,1767	N/A	0,044	N/A
CALENDAR_INFORMATION	0,0104	0,0096	0,0063	N/A	N/A	0,0148	N/A
LOCATION_INFORMATION	0,0342	N/A	0,031	0,0054	N/A	0,0173	N/A
DATABASE_INFORMATION	0,0277	0,0157	0,022	0,0655	0,0032	0,0179	N/A
ACCOUNT_INFORMATON	0,0027	N/A	N/A	N/A	N/A	0,032	N/A
UNIQUE_IDENTIFIER	0,0824	0,0079	0,3059	0,0919	N/A	0,0508	N/A
BLUETOOTH_INFORMATION	N/A	N/A	N/A	N/A	N/A	0,0031	N/A
NO_CATEGORY	0,0284	0,0173	0,0382	0,0799	0,0097	0,0222	0,0088

MAX

	LOG	FILE	NETWORK	SMS_MMS	AUDIO	NO_CATEGORY	LOCATION_INFORMATION
NETWORK_INFORMATION	0,684	0,0096	0,0257	1,8161	N/A	1,1881	N/A
CALENDAR_INFORMATION	0,0421	0,0128	0,0075	N/A	N/A	0,1284	N/A
LOCATION_INFORMATION	0,1403	N/A	0,1175	0,0128	N/A	0,1626	N/A
DATABASE_INFORMATION	0,2092	0,0544	0,0471	0,2336	0,0032	0,2766	N/A
ACCOUNT_INFORMATON	0,0028	N/A	N/A	N/A	N/A	0,1056	N/A
UNIQUE_IDENTIFIER	0,5216	0,0187	0,4279	0,1536	N/A	1,0901	N/A
BLUETOOTH_INFORMATION	N/A	N/A	N/A	N/A	N/A	0,0032	N/A
NO_CATEGORY	0,6032	0,4618	0,5292	1,3315	0,0376	1,1776	0,016

Figure 4: Flow weight distribution grouped by SuSi categories.

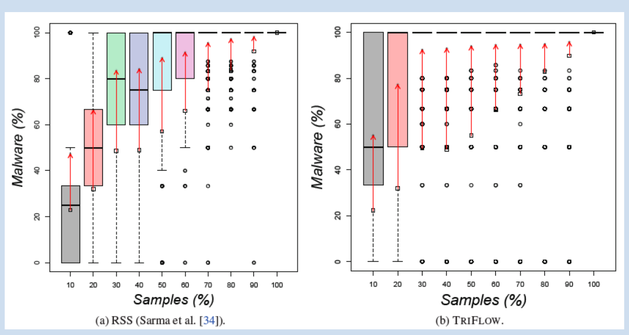
In very rare flows $I(f) > 1$ which are observed mainly in malware.

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.

Figure 5: Top ranked flows.

TriFlow can prioritize more malware samples per batch than RSS for every single workload value from 10 to 100.



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

Figure 6: Triage evaluation.

The scoring report contains the overall score, a break down of scores into the SuSi categories, and, also, flows that contribute to the score.

```
Application Name = 55d7e1c74ed3630f7c8d7a892c049aca.apk   (Trackplus family)
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
...
```

Figure 7: Snippet of TriFlow report for a malware

Conclusion

- **Flow prediction:** Flows can be predicted efficiently based on precise static analysis tools and sufficient ground truth.
- **Flow weighting:** Weights based on a simple yet complete intuition–*rare equals risky*.
- **Risk metric:** A new risk metric has been proposed based on information flows.
- **Triage:** An efficient triage system has been developed which can prioritize analysis and save valuable resources. Applications include initial triage of large scale sets of apps as a lightweight pre-analyzer.

Further Information