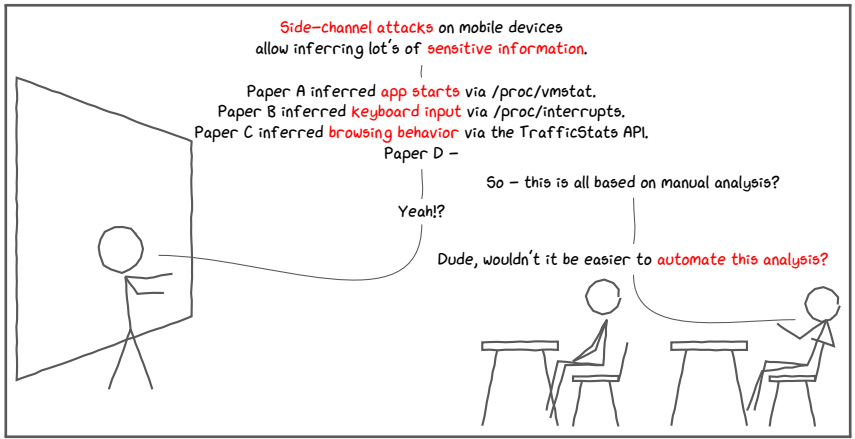
# SCAnDroid: Automated Side-Channel Analysis of Android APIs

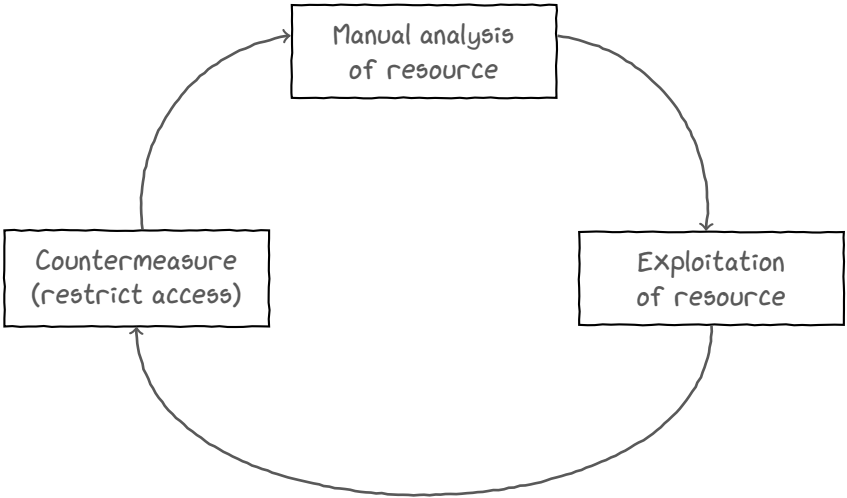
**Raphael Spreitzer**, Gerald Palfinger, Stefan Mangard
IAIK, Graz University of Technology, Austria

WiSec 2018, Stockholm, Sweden, 20th June 2018

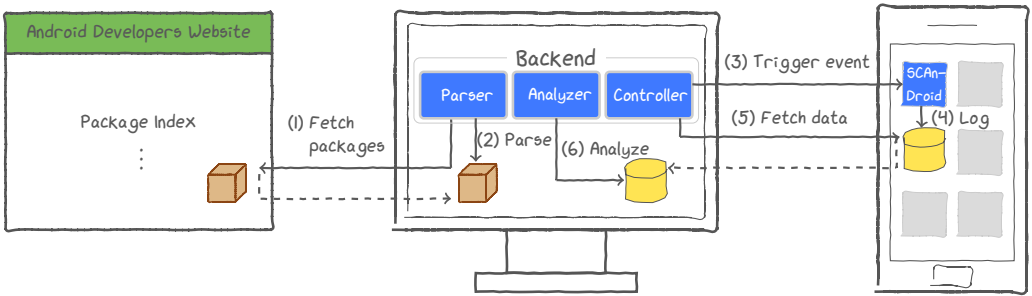
# Motivation and Contribution



# Cat and Mouse Game?



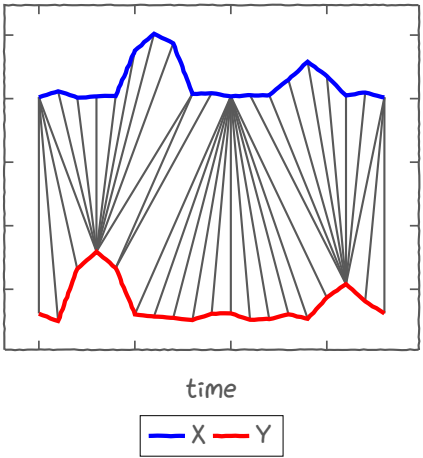
# SCAnDroid


Android Developers Website
Package Index
(1) Fetch packages
Backend
Parser
Analyzer
Controller
(2) Parse
(6) Analyze
(3) Trigger event
(5) Fetch data
SCAn-Droid
(4) Log

# Analysis

## Dynamic time warping (DTW)

- Compare time series
  - $X = (x_1, \ldots, x_n)$
  - $Y = (y_1, \ldots, y_m)$
- No background information
- No human interaction
- Ignoring misaligned, stretched, or compressed traces



# Classification

## DTW-based approach (template attacks)

- Training data: $T = \{(e_i, X_i)\}$
- Test sample $s = (e_j, X)$: $i = \text{argmin } DTW(X, Y_i)$
- $\Rightarrow$ two time series result from the same event if they yield a low distance to each other

## K-fold cross validation

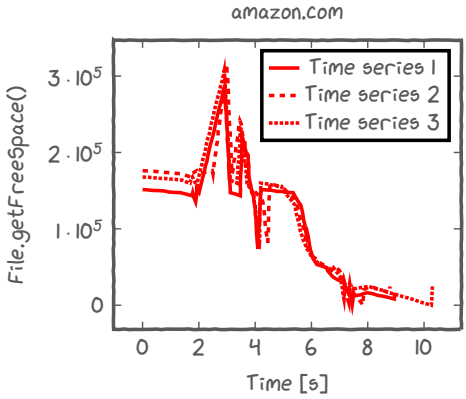
- Accuracy better than random guessing?
- $\Rightarrow$ information leak identified

# Coverage of Analyzed Methods

| Methods | # | % |
|---|---|---|
| Documented in the Android API | 36339 | |
| Relevant (get, is, has, query) | 12012 | 100% |
| In abstract classes or interfaces | 2860 | 23.8% |
| Removed (crashed, missing constructors, etc) | 5075 | 42.3% |
| Theoretically to be profiled | 4077 | 33.9% |
| Actually profiled | 5046 | 42.1% |
| Methods that "react" to events | 36 | |

# Case Study: Website Inference

## Correlations between website launches and API calls

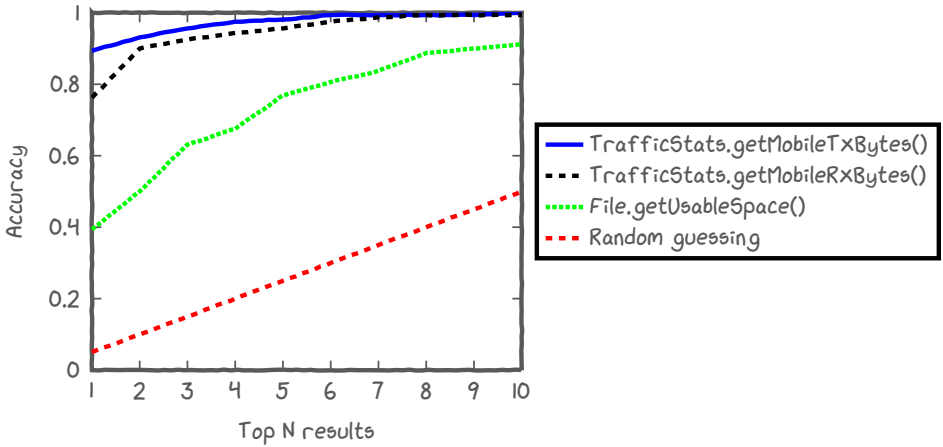



# Website Inference on Android 8

## 20 websites, 8 samples, 10 seconds

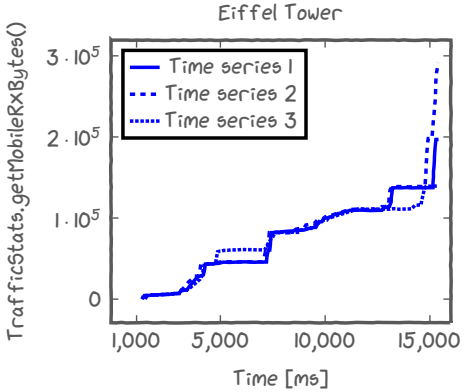
| API | Accuracy |
|---|---|
| android.net.TrafficStats.getMobileTxBytes() | 89.4 % |
| android.net.TrafficStats.getTotalTxBytes() | 88.8 % |
| android.net.TrafficStats.getMobileTxPackets() | 86.2 % |
| android.net.TrafficStats.getTotalRxPackets() | 85.6 % |
| android.net.TrafficStats.getTotalTxPackets() | 85.0 % |
| android.net.TrafficStats.getMobileRxPackets() | 83.1 % |
| android.net.TrafficStats.getTotalRxBytes() | 79.4 % |
| android.net.TrafficStats.getMobileRxBytes() | 76.2 % |
| android.app.usage.StorageStatsManager.getFreeBytes(java.util.UUID) | 46.9 % |
| java.io.File.getUsableSpace() | 39.4 % |
| java.io.File.getFreeSpace() | 38.1 % |
| android.os.storage.StorageManager.getAllocatableBytes(java.util.UUID) | 36.2 % |
| android.os.Process.getElapsedCpuTime() | 21.9 % |

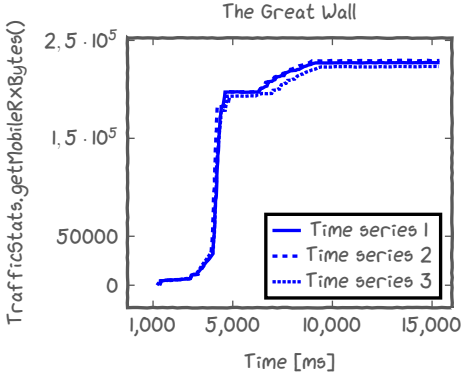
# Case Study: Website Inference on Android 8



# Case Study: Google Maps Search Inference

## Correlations between Google Maps search queries and API calls



# Google Maps Search Inference on Android 8

## 20 POIs, 8 samples, 15 seconds

| API | Accuracy |
|---|---|
| android.net.TrafficStats.getTotalRxBytes() | 87.5 % |
| android.net.TrafficStats.getMobileRxBytes() | 83.8 % |
| android.net.TrafficStats.getMobileRxPackets() | 76.2 % |
| android.net.TrafficStats.getTotalRxPackets() | 73.1 % |
| android.net.TrafficStats.getTotalTxPackets() | 68.1 % |
| android.net.TrafficStats.getMobileTxPackets() | 66.9 % |
| android.net.TrafficStats.getTotalTxBytes() | 49.4 % |
| android.net.TrafficStats.getMobileTxBytes() | 48.8 % |
| android.app.usage.StorageStatsManager.getFreeBytes(java.util.UUID) | 16.2 % |
| android.os.storage.StorageManager.getAllocatableBytes(java.util.UUID) | 13.1 % |
| android.os.Process.getElapsedCpuTime() | 13.1 % |
| java.io.File.getFreeSpace() | 11.9 % |
| java.io.File.getUsableSpace() | 10.6 % |

# Discussion

## Limitation: false negatives

- No leaks identified → secure?
- More specialized features
- Timing side channels not considered
  - iOS: fileExistsAtPath API [ZWB+18]

## Countermeasures

- Restrict access to APIs
- SCAnDroid could be used to eliminate side channels in upcoming Android versions (before they are released)

# Take-Home Message

## Manual analysis of side-channel leaks

- Tedious and error-prone

## SCAnDroid

- Framework to scan the Java-based Android APIs automatically
- Identified several side-channel leaks
- Available at https://github.com/IAIK/SCAnDroid

# SCAnDroid: Automated Side-Channel Analysis of Android APIs

**Raphael Spreitzer**, Gerald Palfinger, Stefan Mangard
IAIK, Graz University of Technology, Austria

WiSec 2018, Stockholm, Sweden, 20th June 2018

# Disclaimer

The xkcd comic, in particular the stick figures, and the plots have been drawn based on StackExchange [stal2] and the xkcd comic "Teaching Physics" [xkcll].

# Bibliography

[stal2] StackExchange: Create xkcd style diagram in TeX.
https://tex.stackexchange.com/questions/74878/create-xkcd-style-diagram-in-tex/74881#74881, 2012.
Accessed: May 31, 2018.

[xkcl1] xkcd Comic: Teaching Physics.
https://xkcd.com/895/, 2011.
Accessed: May 31, 2018.

[ZWB+18] Xiaokuan Zhang, Xueqiang Wang, Xiaolong Bai, Yinqian Zhang, and XiaoFeng Wang.
OS-level Side Channels without Procfs: Exploring Cross-App Information Leakage on iOS.
In Network and Distributed System Security Symposium — NDSS 2018, 2018.