FSE 2024

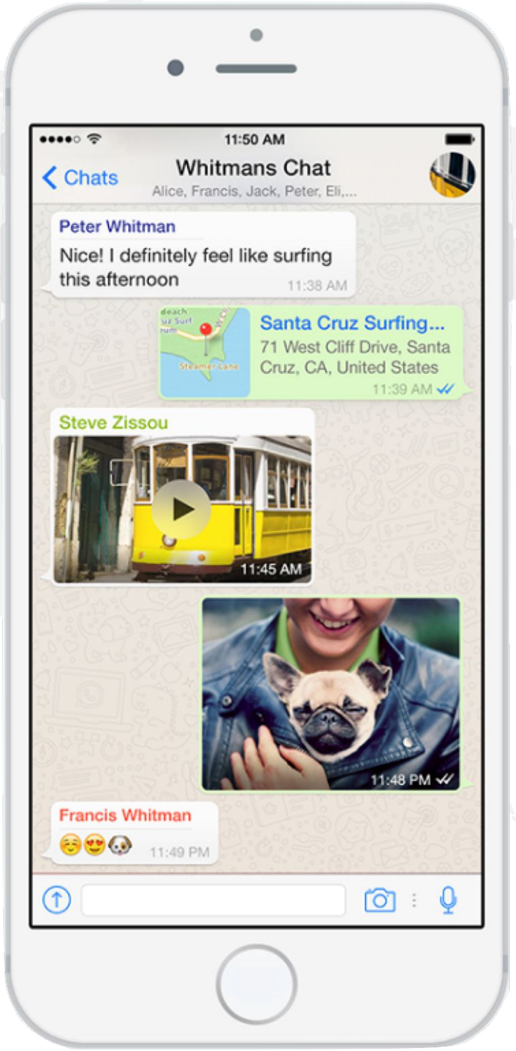
Automated End-to-End Dynamic Taint Analysis for WhatsApp

Sopot Cela, Andrea Ciancone, Per Gustafsson, Ákos Hajdu, Yue Jia,
Timotej Kapus, Maksym Koshtenko, Will Lewis, Ke Mao, Dragos Martac

Meta

2 Billion

People around the world use WhatsApp daily



100 Billion

Messages daily

Simple

So anyone can use it.

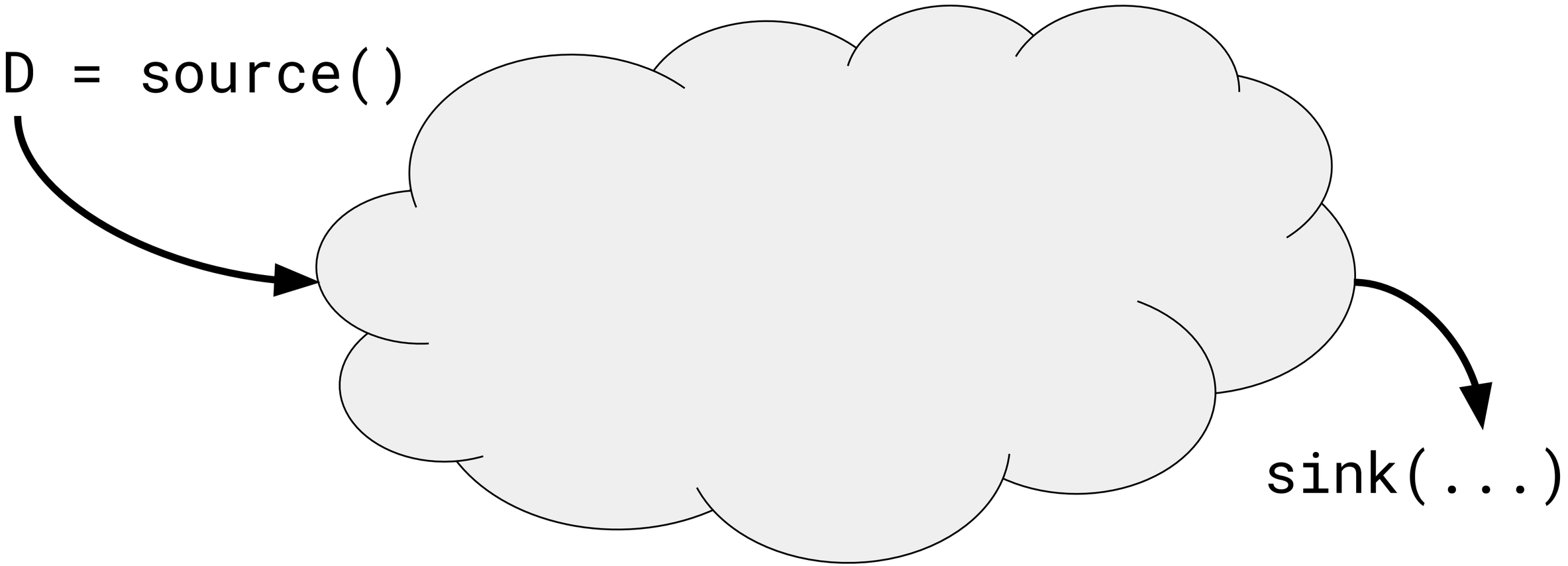
Reliable

So that messages go through no matter what.

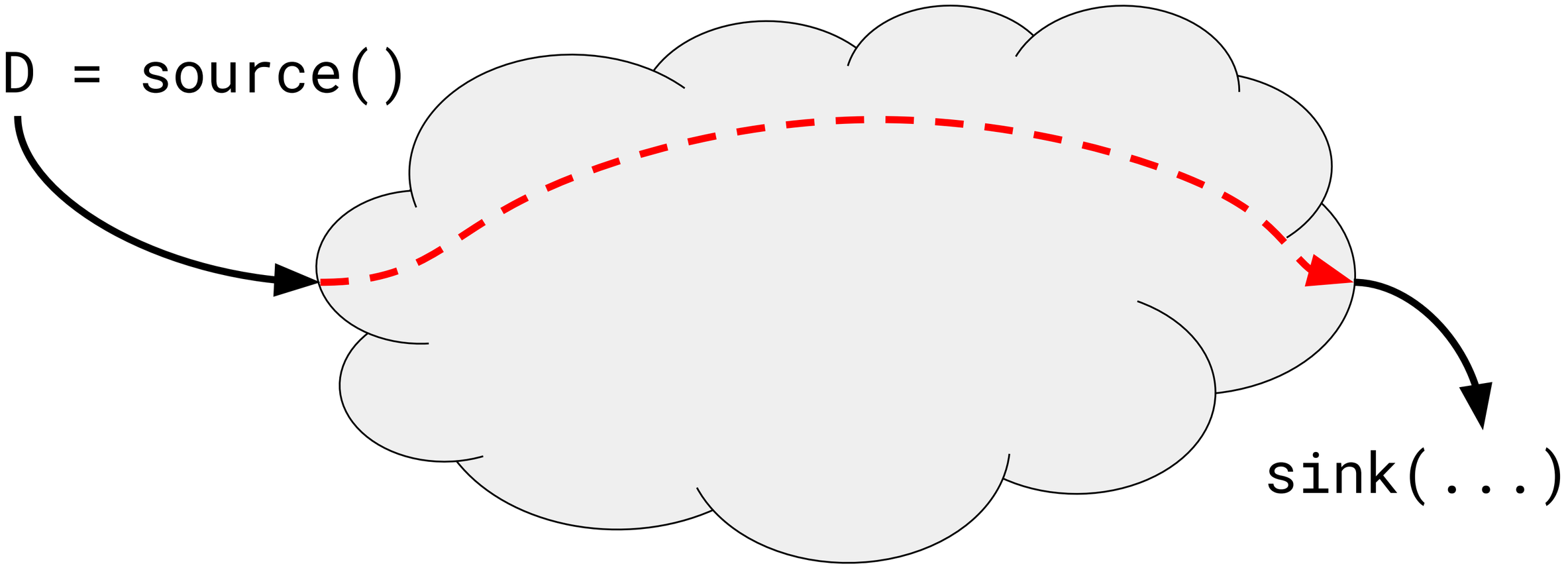
End-to-end Encrypted

So only sender and receiver of the message can see its content.

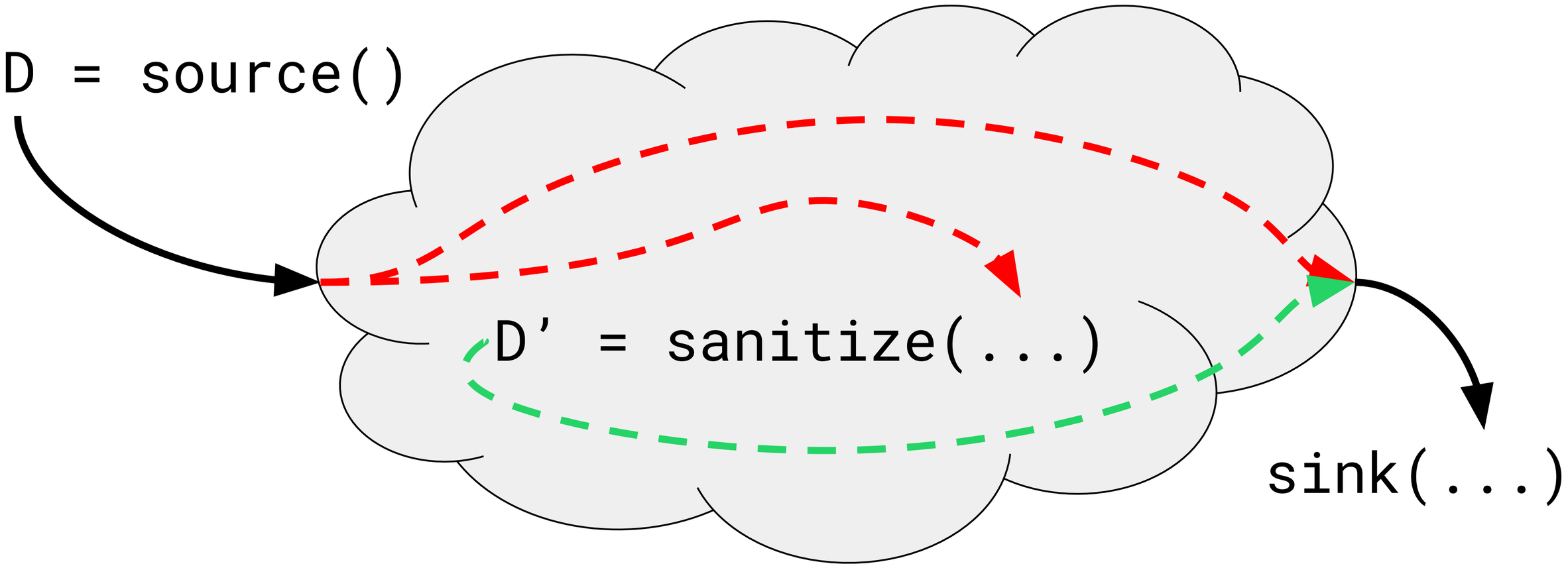
Taint Analysis



Taint Analysis



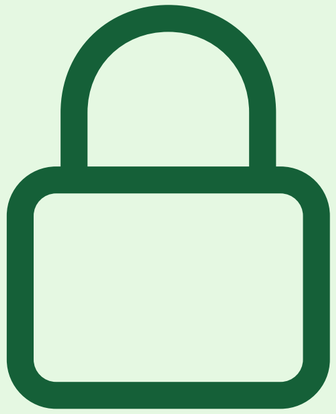
Taint Analysis



Challenges

Privacy

Cannot record and replay real traffic



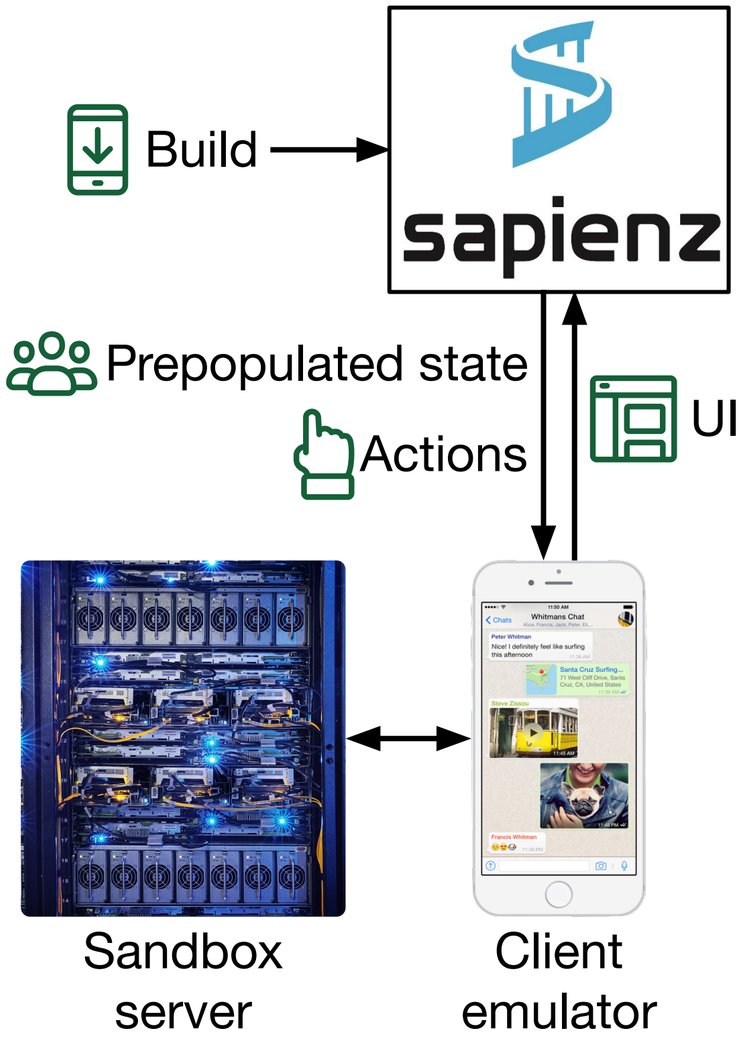
False positives

Due to non-release builds and artificial data

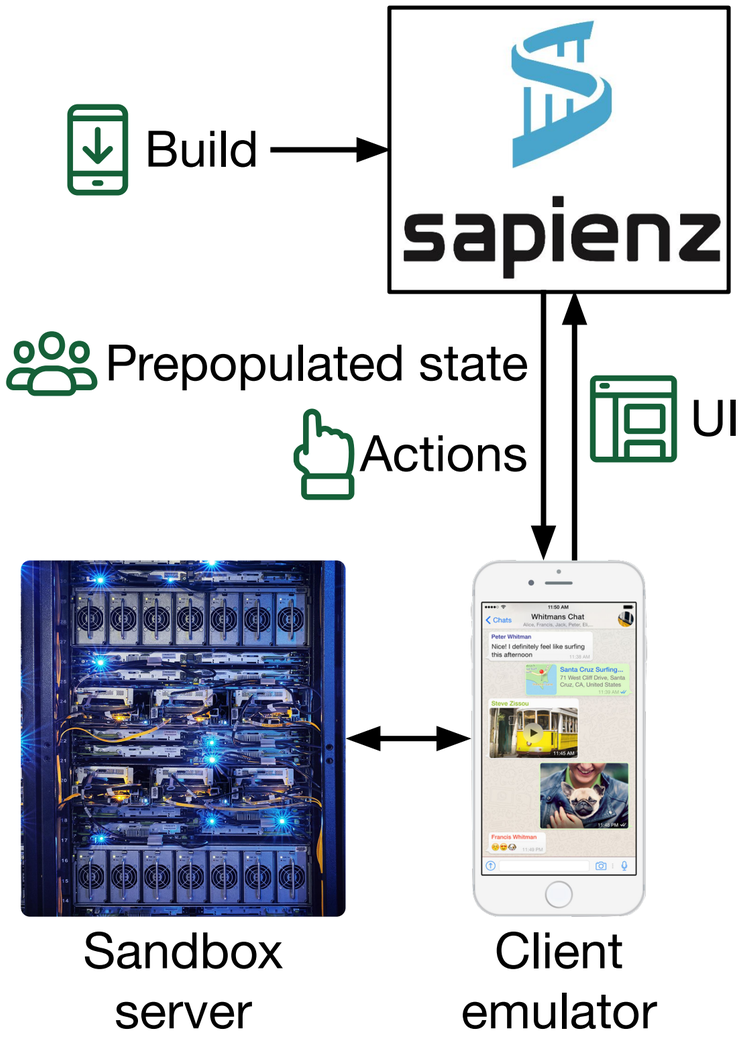
Ecosystem

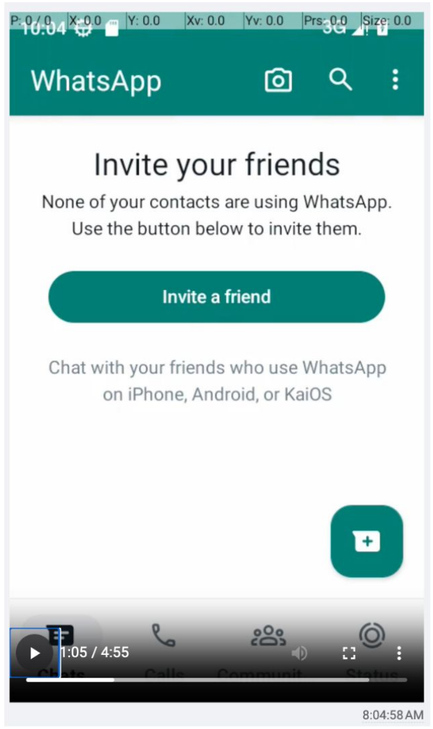
Multiple platforms (client/server) and languages (Java, Kotlin, Obj-C, Swift, Erlang)

Overview

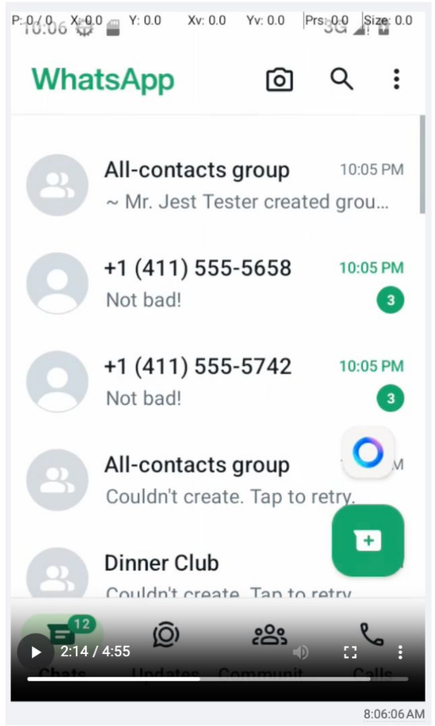

Build
sapienz
Prepopulated state
Actions
UI
Sandbox server
Client emulator

Overview

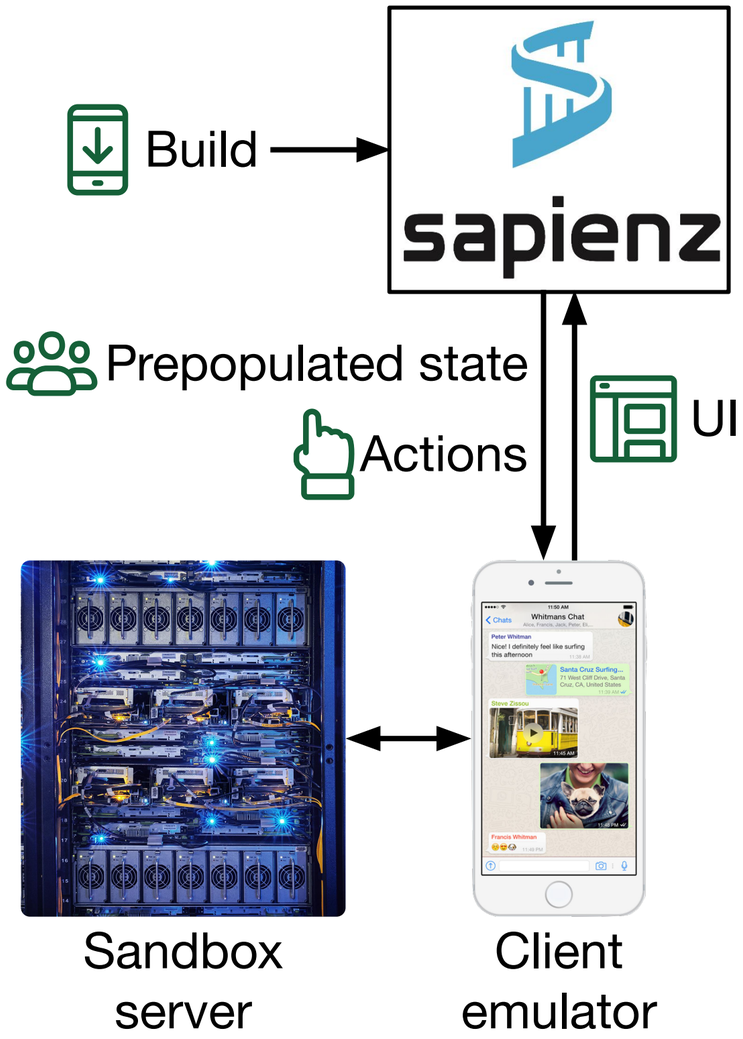


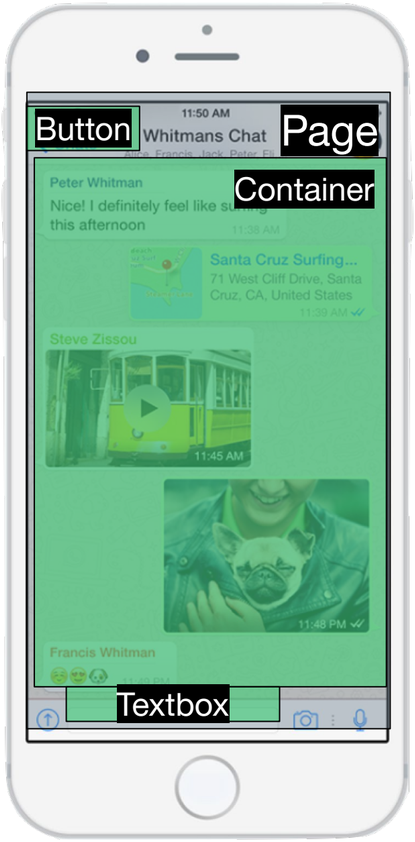


Prepopulate client



Overview

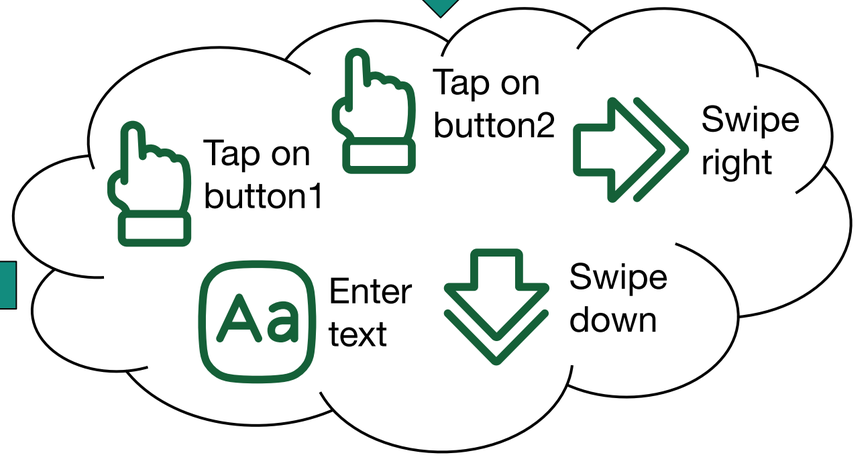




Fetch UI hierarchy

```
Page
└ Button(id=1, label="Chats")
└ Container
   └ Container
   └ Label(text="Peter")
   ...
└ Textbox
```

Generate actions

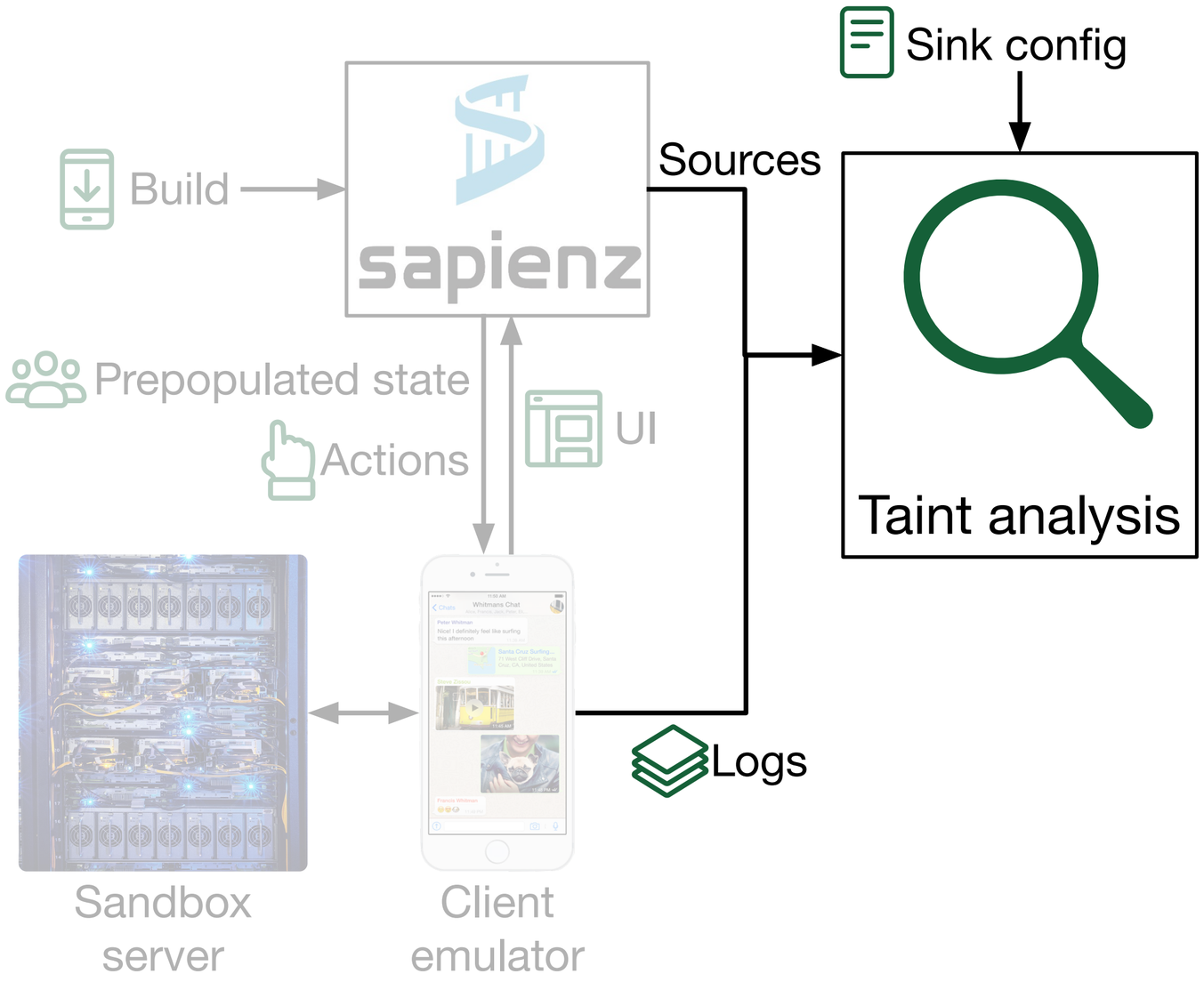


Rank & pick

Act

Tap on button2

Overview

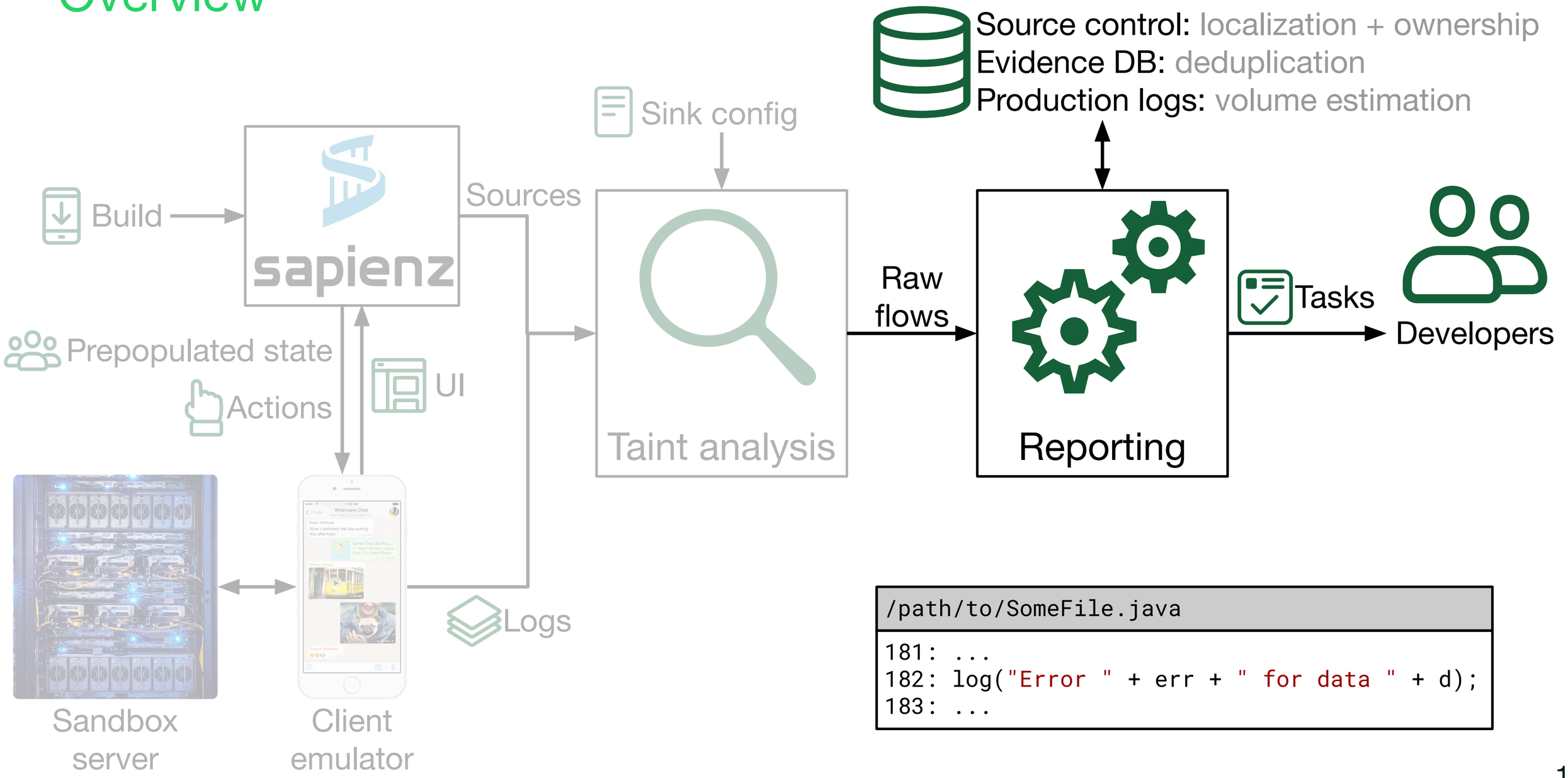


Match sources against sinks

- Interface for rules
- Reusable rules (e.g. regexp)

Group evidences by code pointers

Overview




```
/path/to/SomeFile.java
181: ...
182: log("Error " + err + " for data " + d);
183: ...
```

Results

Results - Overview

Reporting

- Tasks triaged to developers
- Monitor outcomes

Android

68 out of 178

tasks closed with a fix

iOS

21 out of 33

tasks closed with a fix

Results - Overview

Reporting

Tasks triaged to developers

Monitor outcomes

False positives

Tainted flows in initialization

Android

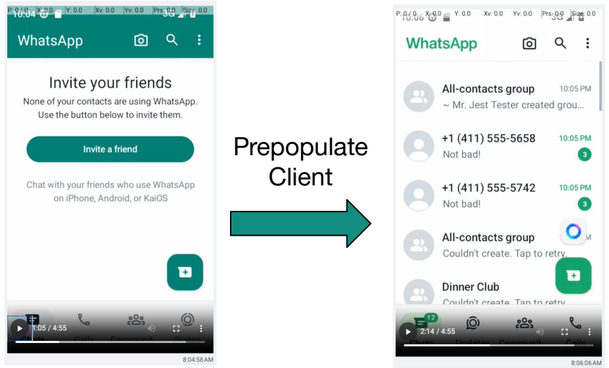
68 out of 178

tasks closed with a fix

iOS

21 out of 33

tasks closed with a fix



Results - Overview

Reporting

Tasks triaged to developers

Monitor outcomes

False positives

Tainted flows in initialization

Inconsistent use of APIs

Android

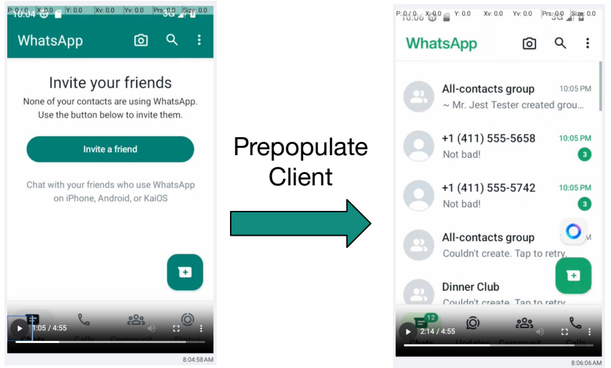
68 out of **178**

tasks closed with a fix

iOS

21 out of **33**

tasks closed with a fix




```
if isDebug():
  Log.production(...)
```

Instead of

```
Log.debug()
```

Results - Overview

Reporting

Tasks triaged to developers

Monitor outcomes

Android

68 out of 178

tasks closed with a fix

iOS

21 out of 33

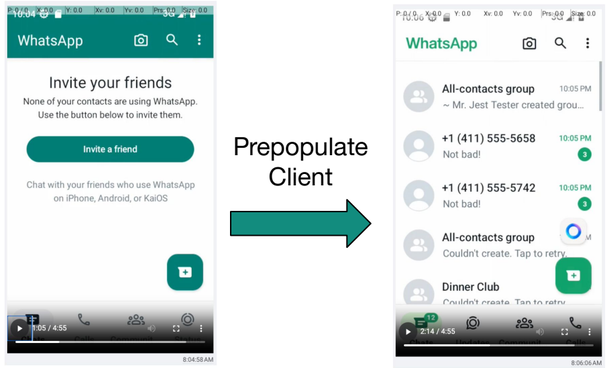
tasks closed with a fix

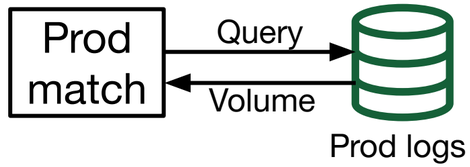
False positives

Tainted flows in initialization

Inconsistent use of APIs

Production matching heuristics






```
if isDebug():
  Log.production(...)
```

Instead of

```
Log.debug()
```

Escalation and False Negatives

SEV: Escalation process

False negatives

- Incremental development
- Very specific (e.g. country-specific)
- Gated features

Android

4 out of 10

privacy SEVs detected

iOS

6 out of 10

privacy SEVs detected

Performance and Coverage

Android

18 mins
p90 execution time

1920 jobs/day
100 actions/job

34%
Activity coverage

iOS

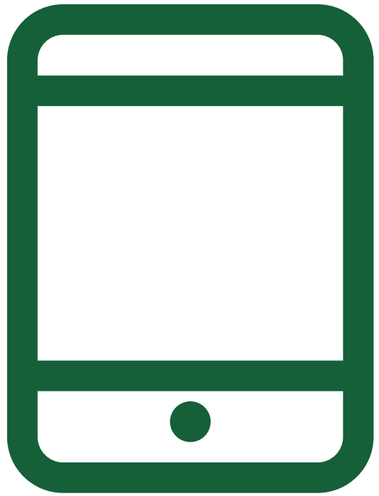
21 mins
p90 execution time

1920 jobs/day
100 actions/job

46%
UIViewController coverage

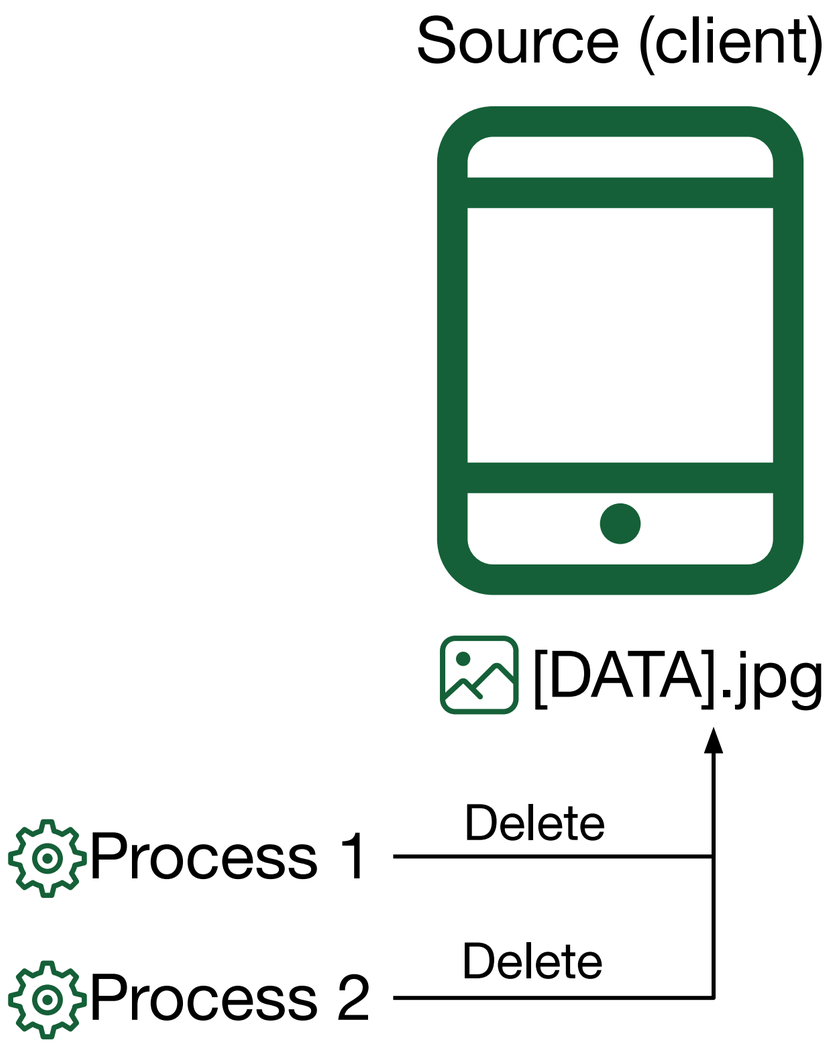
Example

Source (client)

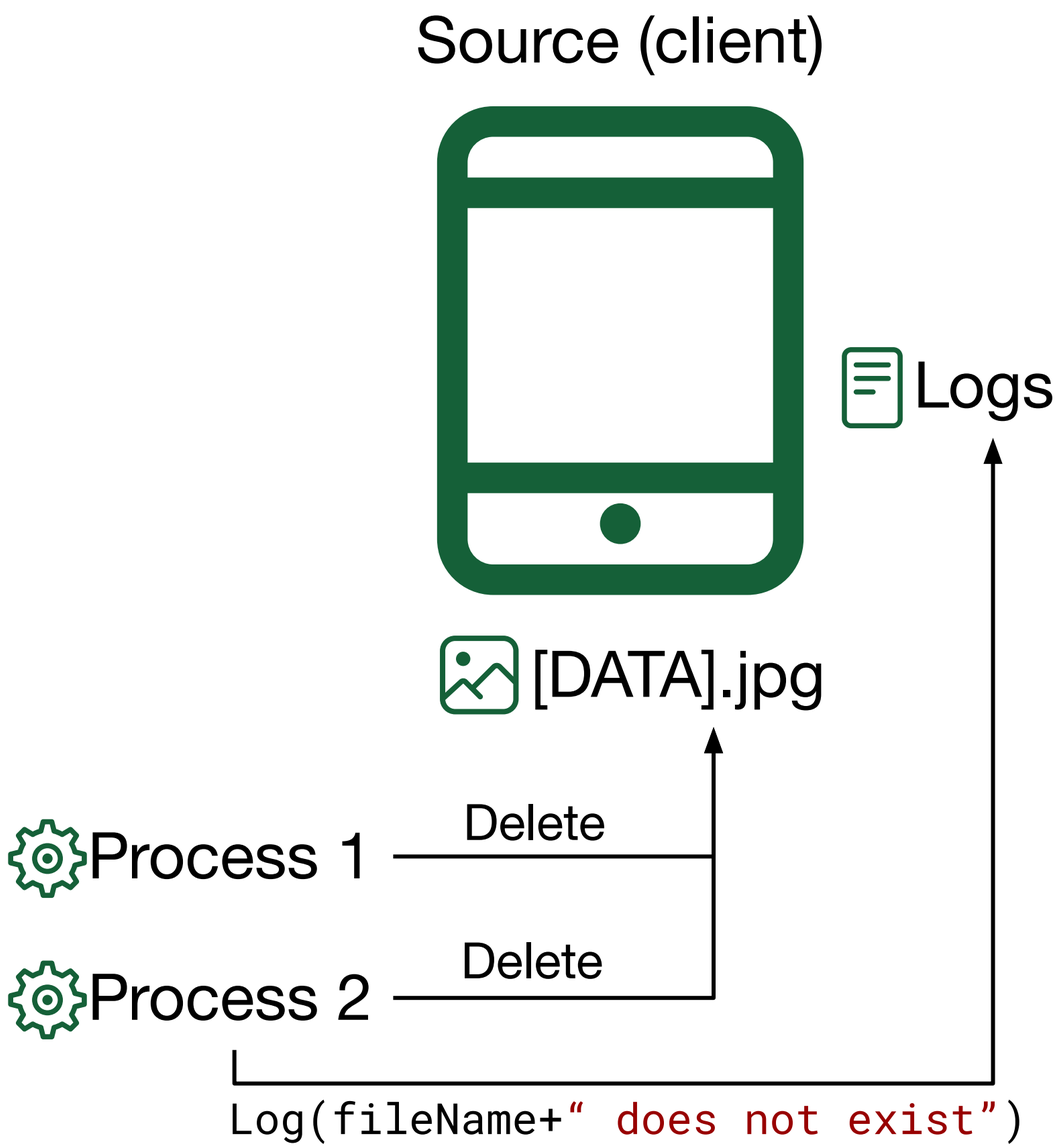


[DATA].jpg

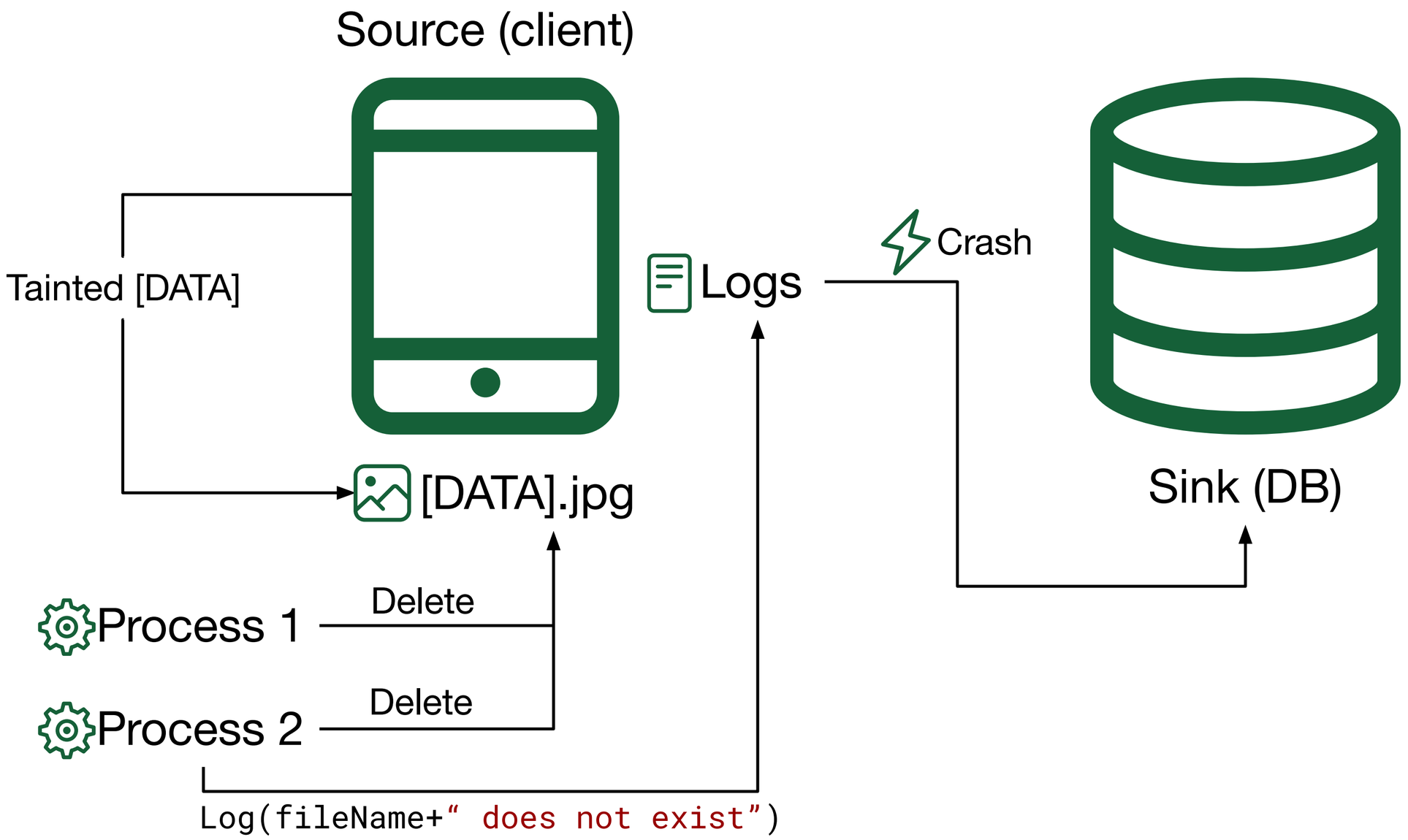
Example



Example


Source (client)
Logs
[DATA].jpg
Process 1
Delete
Process 2
Delete
Log(fileName+" does not exist")

Example


Source (client)
Tainted [DATA]
Logs
Crash
Sink (DB)
[DATA].jpg
Process 1
Delete
Process 2
Delete
Log(fileName+" does not exist")

PrivacyCAT

PrivacyCAT: Privacy-Aware Code Analysis at Scale

Ke Mao
Meta
London, UK
kemao@meta.com

Cons T Åhs
Meta
London, UK
cons@meta.com

Sopot Cela
Meta
London, UK
scela@meta.com

Dino Distefano
Meta UK and Queen Mary University of London
ddino@meta.com

Nick Gardner
Meta
London, UK
nikgardner@meta.com

Radu Grigore
Meta
London, UK
rgrigore@meta.com

Per Gustafsson
Meta
London, UK
pergu@meta.com

Ákos Hajdu
Meta
London, UK
akoshajdu@meta.com

Timotej Kapus
Meta
London, UK
kapust@meta.com

Matteo Marescotti
Meta
London, UK
mmatteo@meta.com

Gabriela Cunha Sampaio
Meta
London, UK
gabrielasampaio@meta.com

Thibault Suzanne
Meta
London, UK
tsuzanne@meta.com

Abstract

Static and dynamic code analyses have been widely adopted in industry to enhance software reliability, security, and performance by automatically detecting bugs in the code. In this paper, we introduce PrivacyCAT[1], a code analysis system developed and deployed at WhatsApp to protect user privacy. PrivacyCAT automatically detects privacy defects in code at early stages (before reaching production and affecting users), and therefore, it prevents such vulnerabilities from evolving into privacy incidents. PrivacyCAT comprises of a collection of static and dynamic taint analysers.

We report on the technical development of PrivacyCAT and the results of two years of its large-scale industrial deployment at WhatsApp. We present our experience in designing its system architecture, and continuous integration process. We discuss the unique challenges encountered in developing and deploying such kind of analyses within an industrial context.

Since its deployment in 2021, PrivacyCAT has safeguarded data privacy in 74% of privacy site events (SEVs). It has prevented 493 potential privacy SEVs from being introduced

[1]Authors after the first author are in alphabetical order, which is not intended to denote any information about the relative contribution. All of Per Gustafsson's contribution to this work was conducted at Meta.

FAUSTA: Scaling Dynamic Analysis with Traffic Generation at WhatsApp

Ke Mao
Meta
kemao@fb.com

Timotej Kapus
Meta
kapust@fb.com

Lambros Petrou
Meta
petrou@fb.com

Ákos Hajdu
Meta
akoshajdu@fb.com

Matteo Marescotti
Meta
mmatteo@fb.com

Andreas Löscher
Meta
loscher@fb.com

Mark Harman
Meta
markharman@fb.com

Dino Distefano
Meta
ddino@fb.com

InfERL: Scalable and Extensible Erlang Static Analysis

Ákos Hajdu
Meta
London, UK
akoshajdu@meta.com

Matteo Marescotti
Meta
London, UK
mmatteo@meta.com

Thibault Suzanne
Meta
London, UK
tsuzanne@meta.com

Ke Mao
Meta
London, UK
kemao@meta.com

Radu Grigore
Meta
London, UK
rgrigore@meta.com

Per Gustafsson
Meta
London, UK
pergu@meta.com

Dino Distefano
Meta
London, UK
ddino@meta.com

Abstract

In this paper we introduce InfERL, an open source, scalable, and extensible static analyzer for Erlang, based on Meta's Infer tool. InfERL has been developed at WhatsApp and it is deployed to regularly scan WhatsApp server's Erlang codebase, detecting reliability issues and checking user-defined

1 Introduction

WhatsApp is the largest messaging app on the planet. Over 2 billion people rely on it for their personal and business communication, every day.

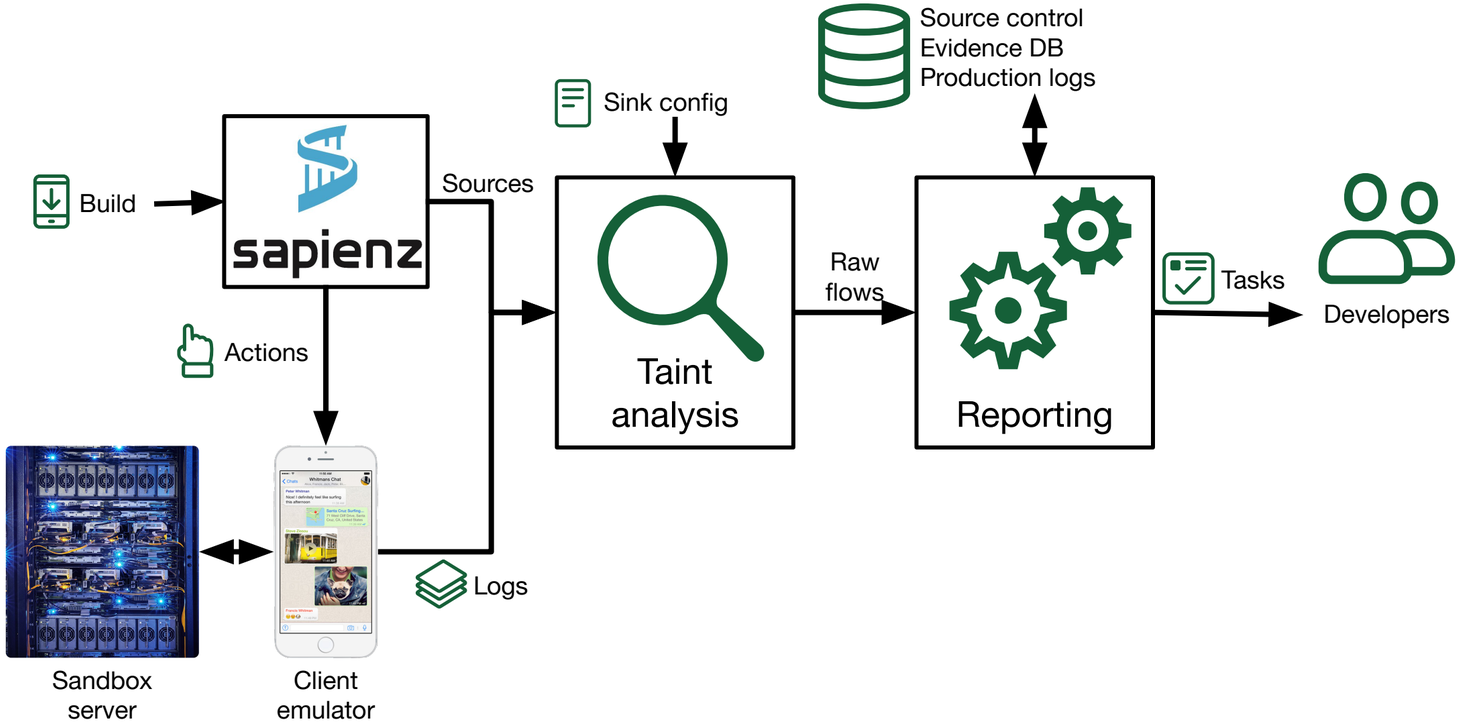
At Meta we develop a variety of tools to help programmers write robust code. One such tool is Infer [5], an open source

into continuous integration in industry at scale (applications consisting of millions of lines of code, used by over two billion users every day).

FAUSTA (**F**ully-**AU**tomated **S**erver **T**esting and **A**nalysis) was developed as a platform for next-generation server testing and analysis of WhatsApp, a well-known communication platform deployed by Meta. FAUSTA provides a framework for dynamic code analysis and testing with the end goal of helping developers gain confidence in their code changes at an early stage of the development cycle. The analysis focuses on helping developers to enforce software reliability, privacy, and performance for WhatsApp backend services, without relying on human engineers to write and maintain the tests themselves. FAUSTA is designed as a platform to allow back-end service owners to onboard their own products and use-cases.

We initially developed FAUSTA for improving code coverage and reducing test maintenance effort. At the time, test code coverage of the server code base was limited, even though there was a non-trivial number of unit and end-to-end tests implemented. Developers also suffered from debugging and maintaining flaky tests. While we encouraged developers to continue writing high-quality tests, we proposed to comple-

Conclusions



Android

68 out of **178**
tasks closed with a fix

4 out of **10**
privacy SEVs detected

18 mins
p90 execution time

34%
Activity coverage

iOS

21 out of **33**
tasks closed with a fix

6 out of **10**
privacy SEVs detected

21 mins
p90 execution time

46%
UIViewController coverage