Triceratops

Privacy-protecting Mobile Apps

Edward Wu

University of Washington

Sai Zhang, Ravi Bhoraskar, Rene Just, Mike Ernst

Motivation

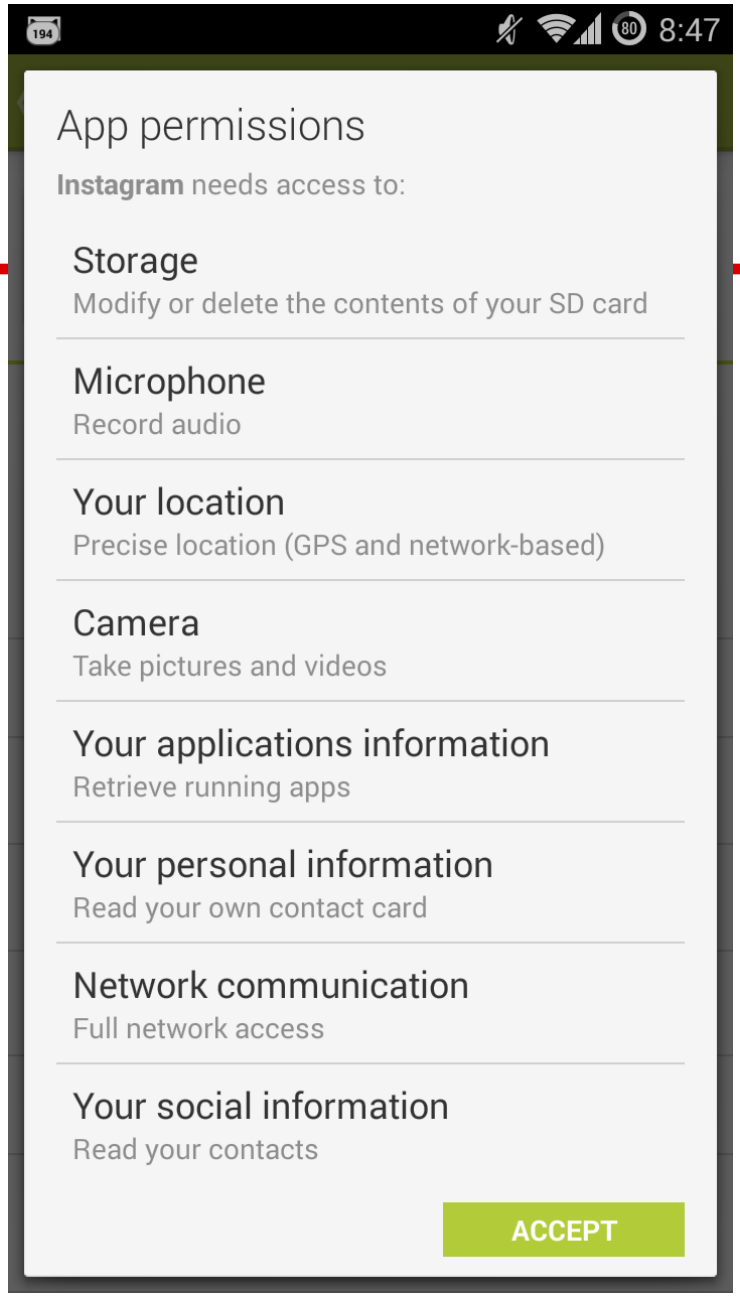
- Mobile security is becoming increasingly important
  - In 2013, there are over 1 billion smartphone users around the globe another billion users by 2015
  - F-Secure identified 275 new mobile threat families in Q1 2014, up from 149 last year
- Mobile privacy is a leading concern
  - Over 50% of the Android malware has some private information collection capabilities

Threat Model

- Mobile privacy: Leakage of personal or sensitive information
  - GPS coordinates
  - Audio recordings
  - Contacts list
  - SMS messages
- Not focusing on:
  - Attacks that tries to take over the device
  - Phishing, social engineering attacks
  - Denial of Service

Android Permission

- Coarse-grained permission system
  - Possible to hide malicious behavior
- Weak enforcement
  - All or nothing



Malware Example

Kittey Kittey

- A real Android malware, designed to evade detection tools

READ_FILESYSTEM **INTERNET_ACCESS**

Approach

An **enforcement tool** that allows users to enforce **fine-grained privacy policies** on a given mobile app

Design challenges:

- What is a easy-to-write and expressive syntax for privacy policies?
- How to build a tool that precisely and effectively enforce these policies?

Outline

- Privacy policy
- Enforcement tool
  - Survey of existing techniques
  - Static optimized dynamic enforcement
- Implementation
- Demo
- Preliminary Evaluations

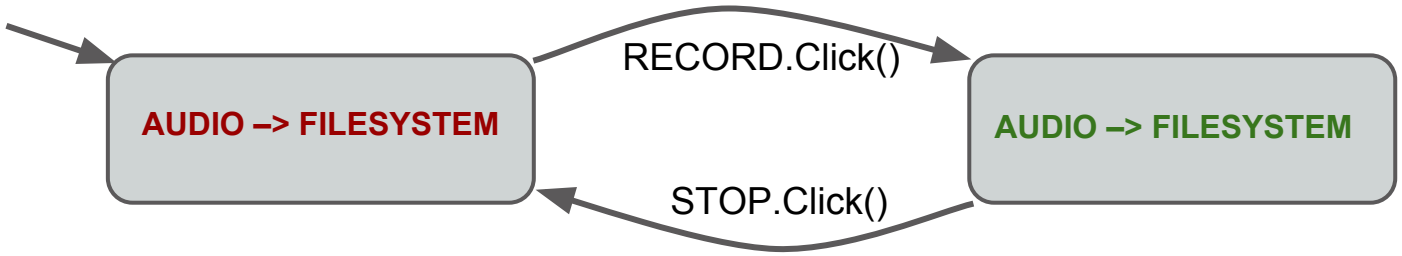
What is Privacy Policy

A specification determining how sensitive information is allowed or not allowed to be used within the app.

Components:

- **Information Flow**: how sensitive data can be exfiltrated
  - Filesystem -> Internet
  - Call logs -> SMS
- **Control Flow:** specific code paths or preconditions
  - Not allowed to upload GPS coordinate till a button is pressed

Privacy Policy Example



Audio recording is only allowed after RECORD is clicked and before STOP is pressed

A FSM that describes both the information flow and the control flow specifications

- State: a list of allowed or disallowed information flows
- Edge: a specific program instruction that causes the state change

Who will write the privacy policy

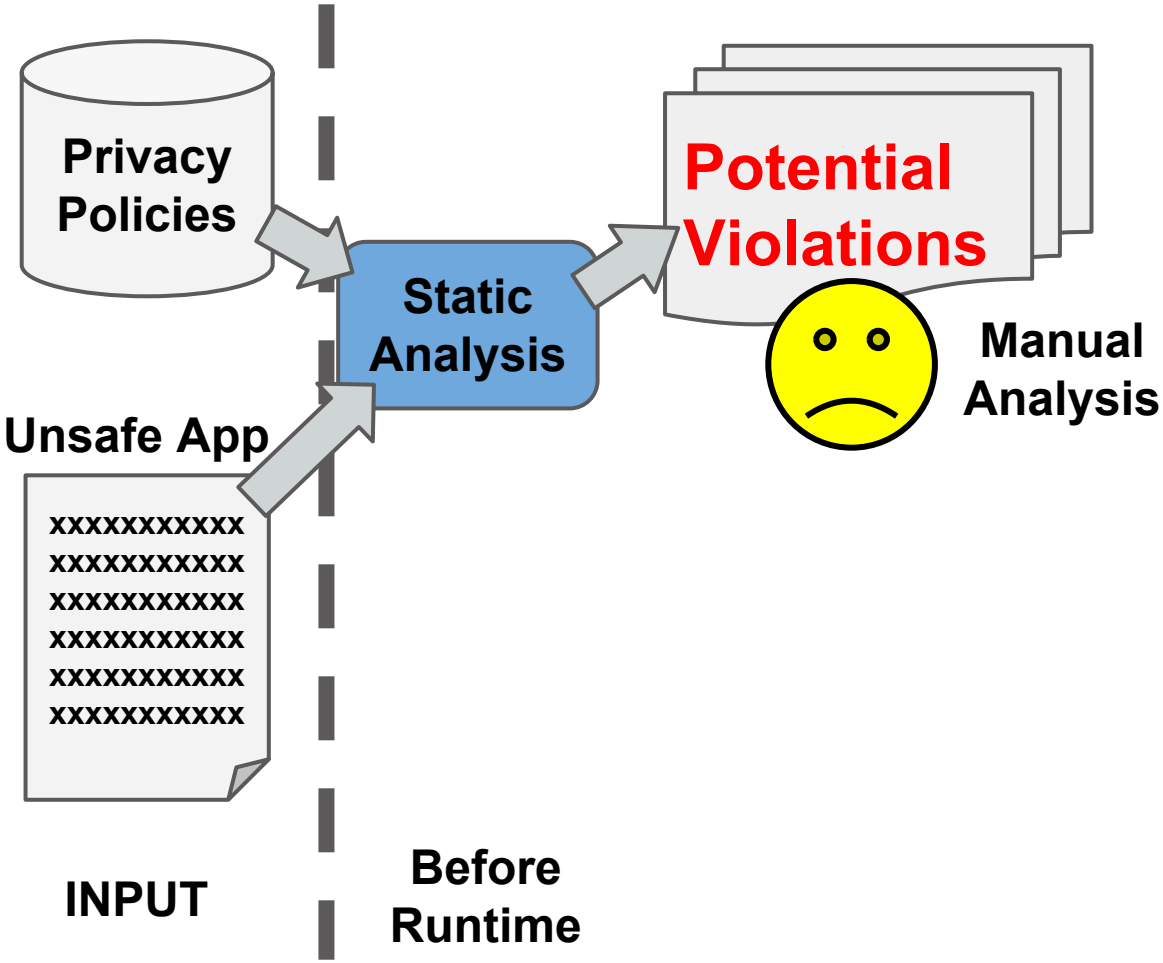
- App developer
  - Specifies how sensitive data are used in more detail
  - “Enhanced permission system”
- Sysadmins
  - Apply set of default “not-allowed” policies based on app’s permission
- User
  - All sensitive data flow is not-allowed by default
  - Ask user’s permission when a flow first occurs
  - Next time this specific flow occurs, it will be automatically allowed or blocked

Survey of existing enforcement techniques

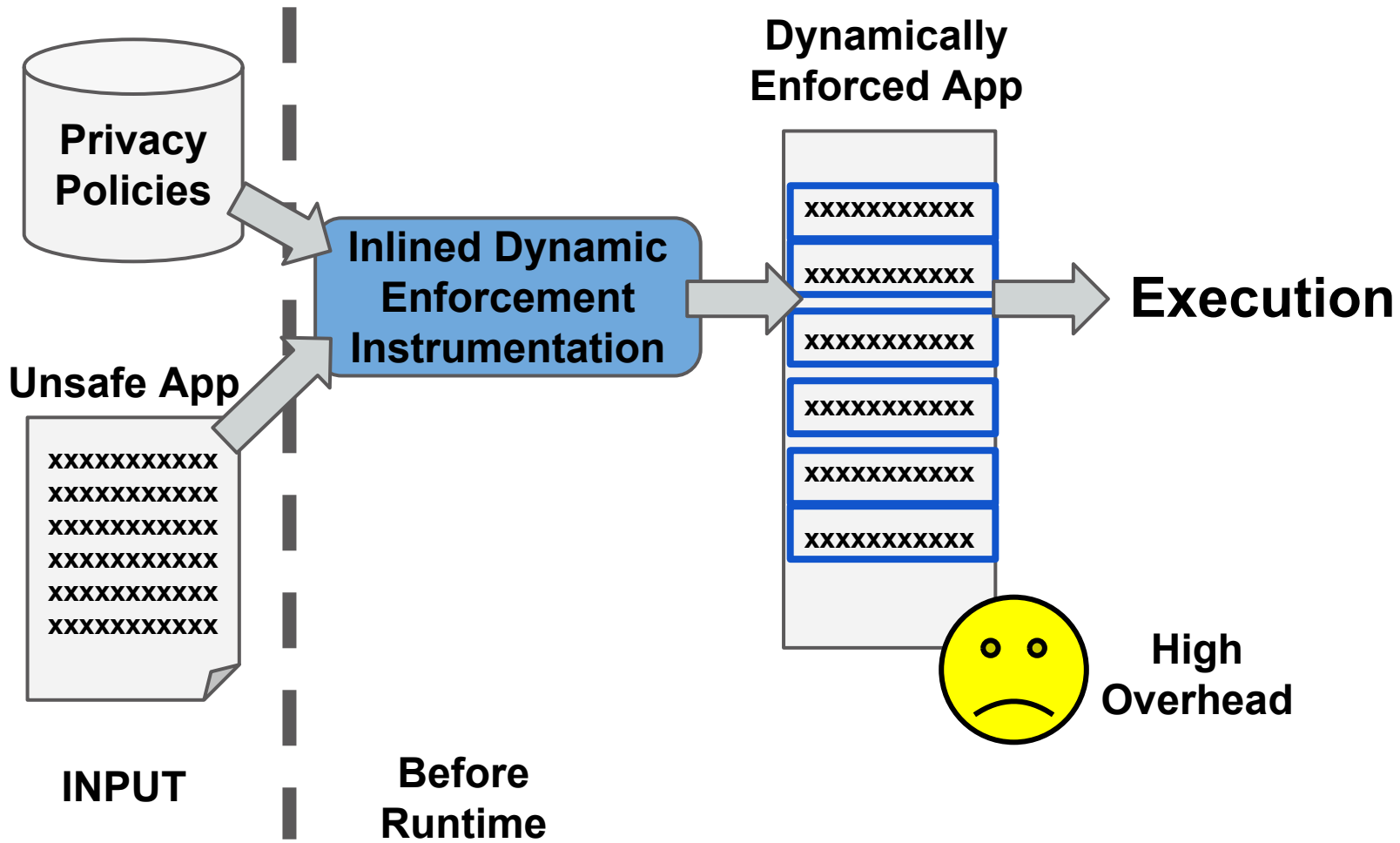
Metrics:

- Precision
  - No false positive
- Usability
  - Small runtime overhead
- Practicality
  - Automated
  - Does not require modification to the runtime system

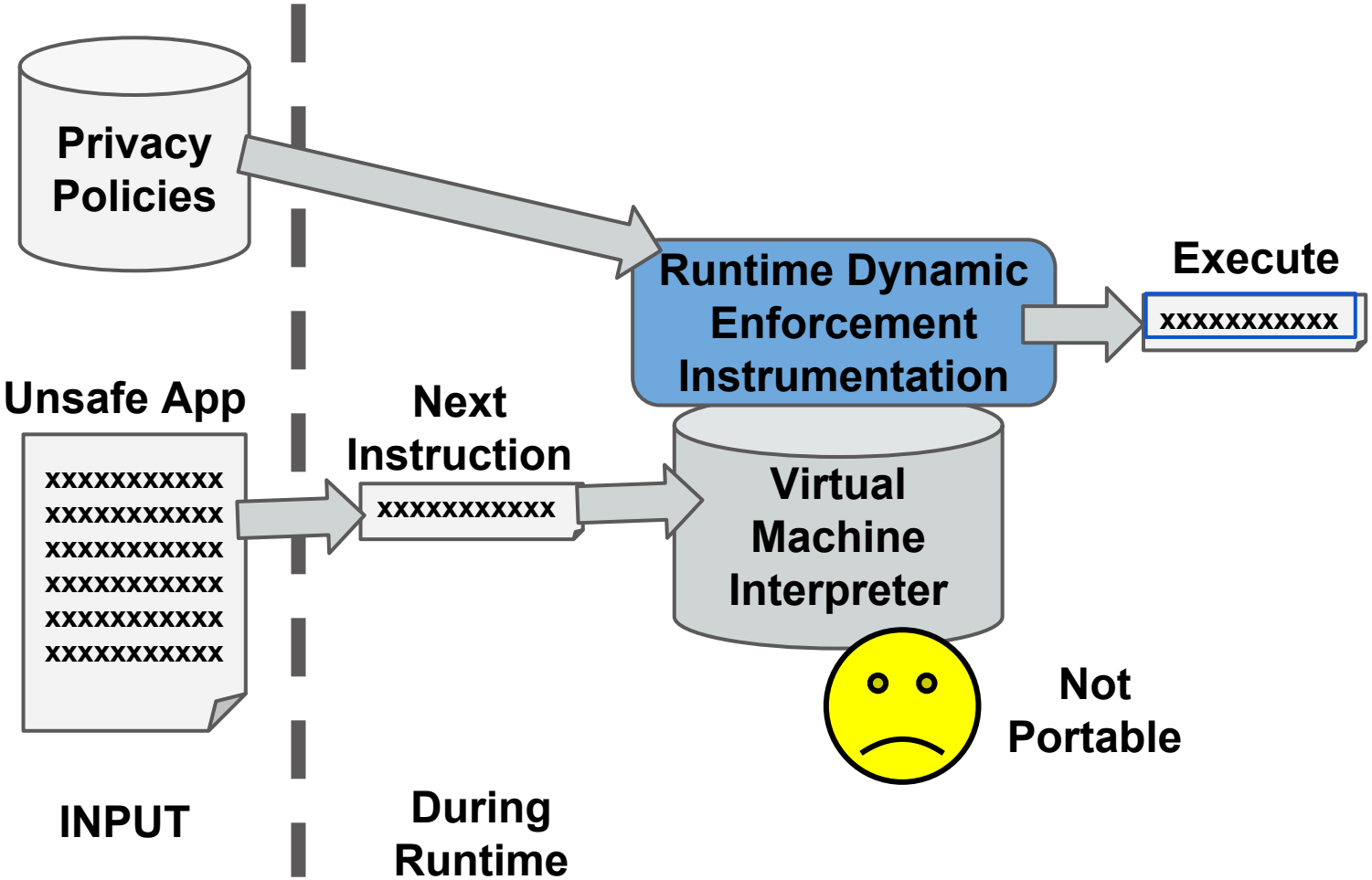
Static Analysis



Inlined Dynamic Enforcement



Runtime Dynamic Enforcement

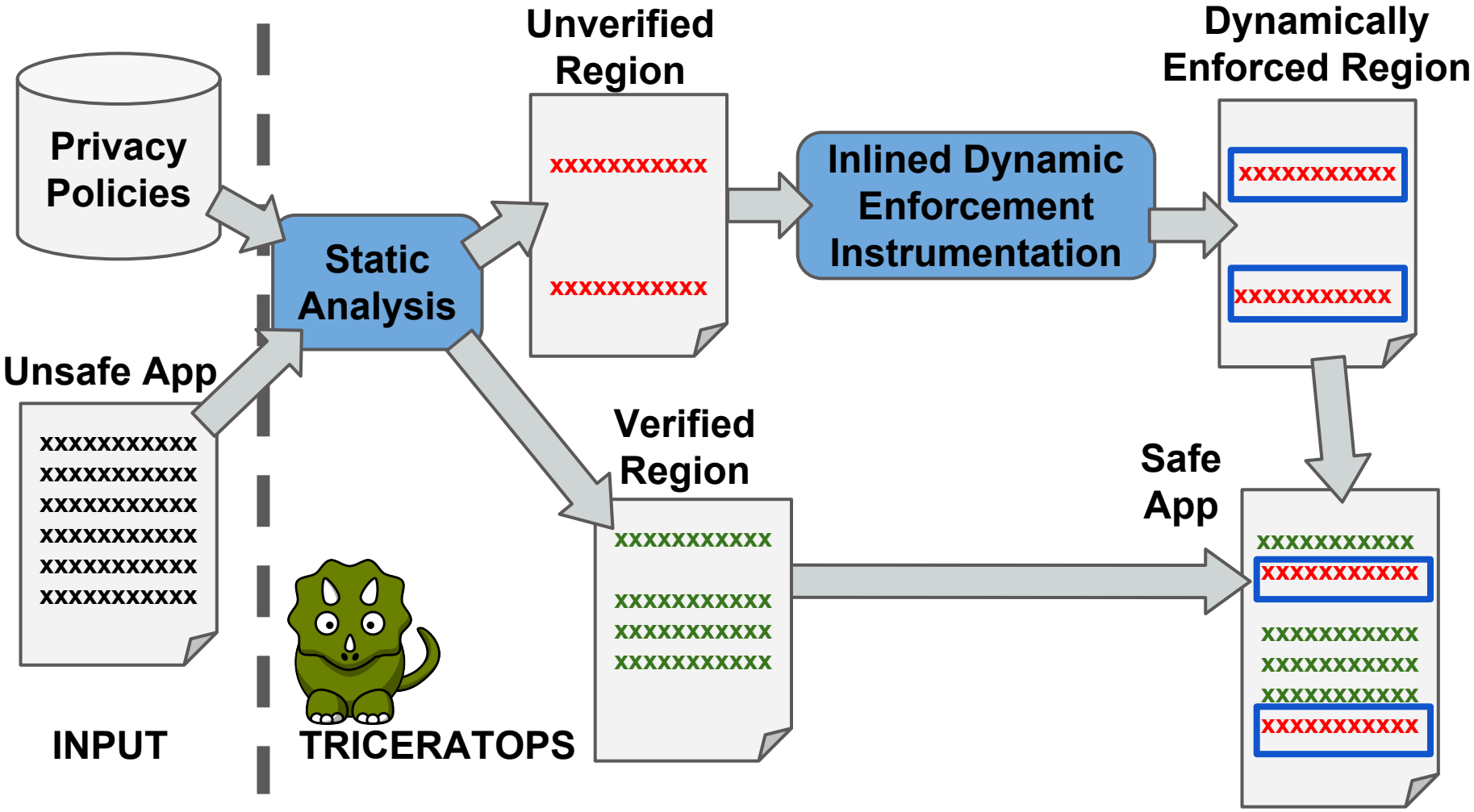


Comparison

Techniques	Runtime Overhead	Portable	False Positives
Static Analysis (Conservative)	N/A	N/A	YES
Runtime Dynamic Enforcement	Low	NO	NO
Inlined Dynamic Enforcement	High	YES	NO
Triceratops	Low	YES	NO

Key idea: Combine static analysis and inlined dynamic enforcement

Intuition


Privacy Policies
Unsafe App
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
INPUT
Static Analysis
TRICERATOPS
Unverified Region
xxxxxxxxxxx
xxxxxxxxxxx
Verified Region
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
Inlined Dynamic Enforcement Instrumentation
Dynamically Enforced Region
xxxxxxxxxxx
xxxxxxxxxxx
Safe App
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx
xxxxxxxxxxx

Static Optimized Dynamic Enforcement

- Minimizes the instrumentation needed to enforce a set of policies by using static analysis to:
  - Apply API summaries
  - Identify unsafe code regions
  - Optimize enforcement code

API Summary

- Allows static analysis to reason about API's effect without executing the app
- Remove the need to instrument API bodys

```
File f=sensitiveFile
String x= Long.toString(f.lastModified())  ⇨  String x= f
uploadToInternet(x)
```

```
Long.toString(long)
File.lastModified()
```

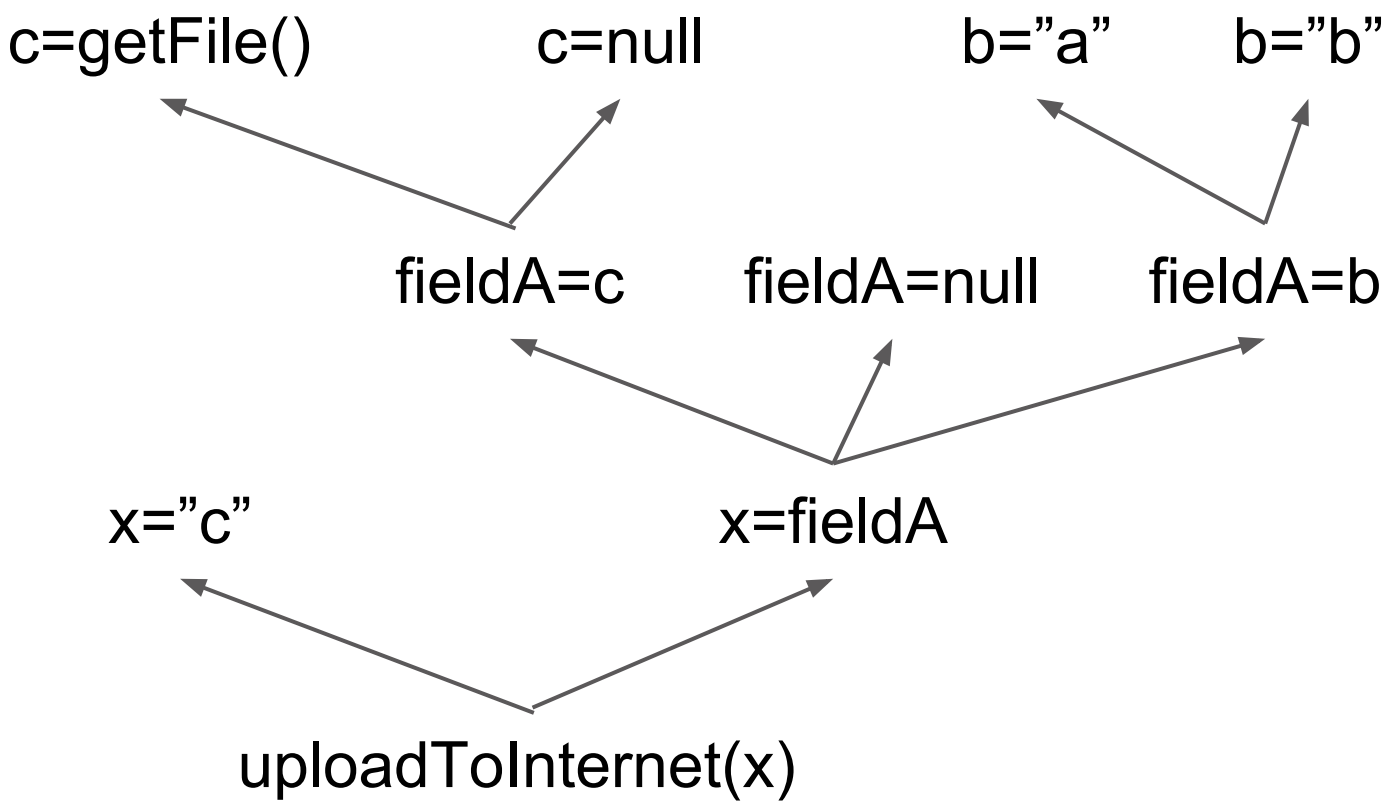
```
if (parameter.isSensitive)
    return Sensitive
else
    return NotSensitive
```

Identify Unsafe Code Regions

- Because the tool knows exactly what data flow it needs to track
- Conservatively identify code regions that help compute or propagate data from the source

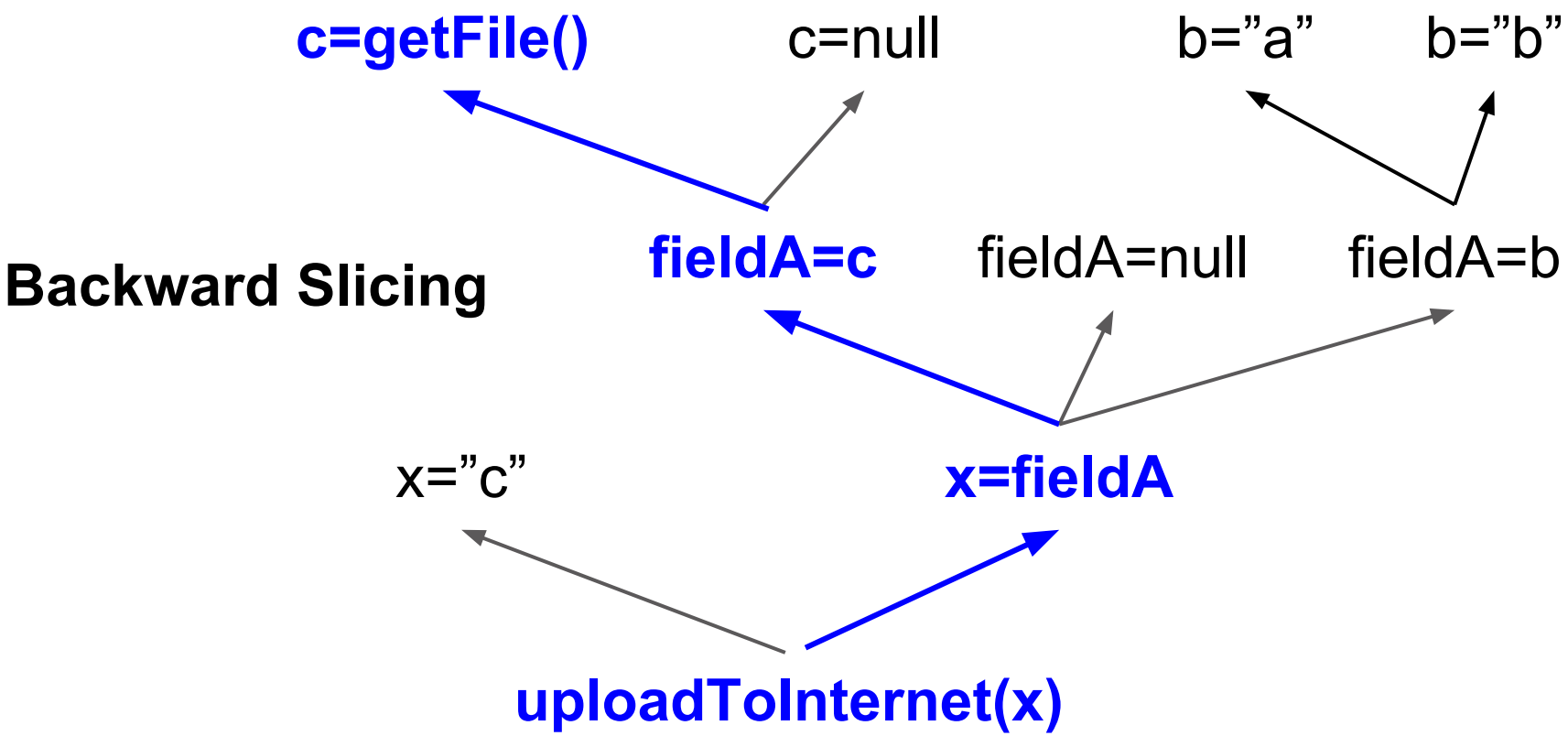
Example

Filesystem -> Internet



Example

Filesystem -> Internet

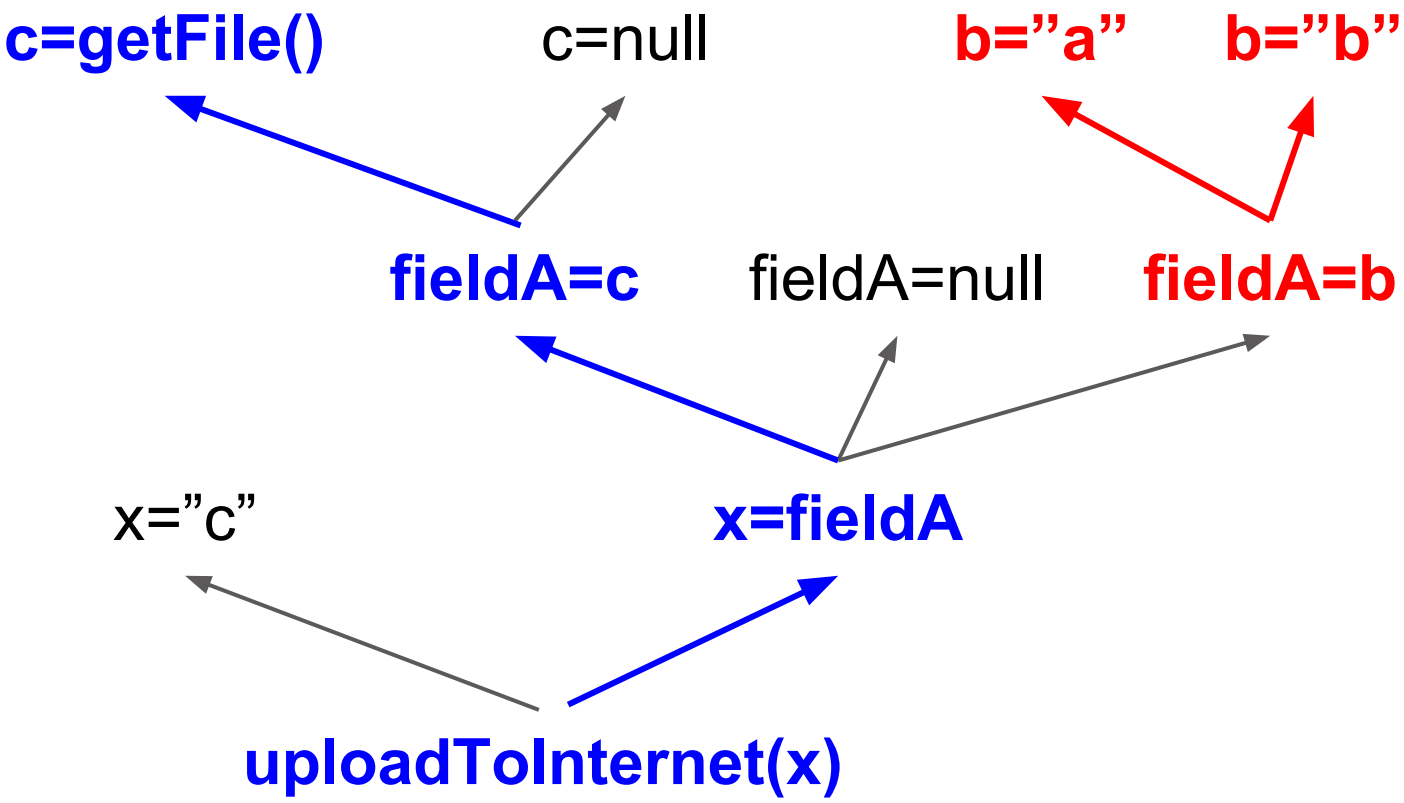


Enforcement Code Optimization

- Static taint propagation
- Constant folding
- Copy propagation
- Dead code elimination

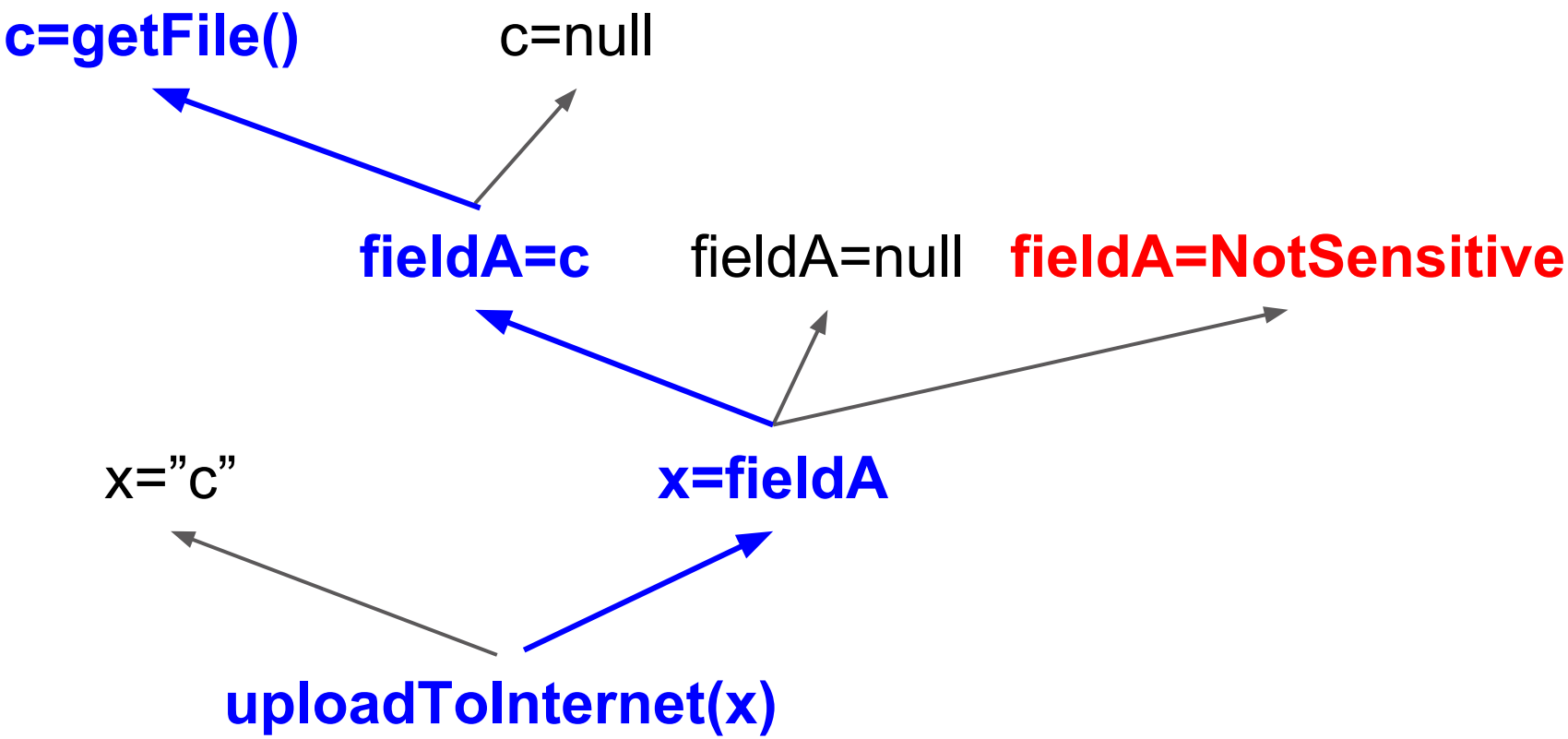
Static Taint Propagation Example

Filesystem -> Internet



Static Taint Propagation Example

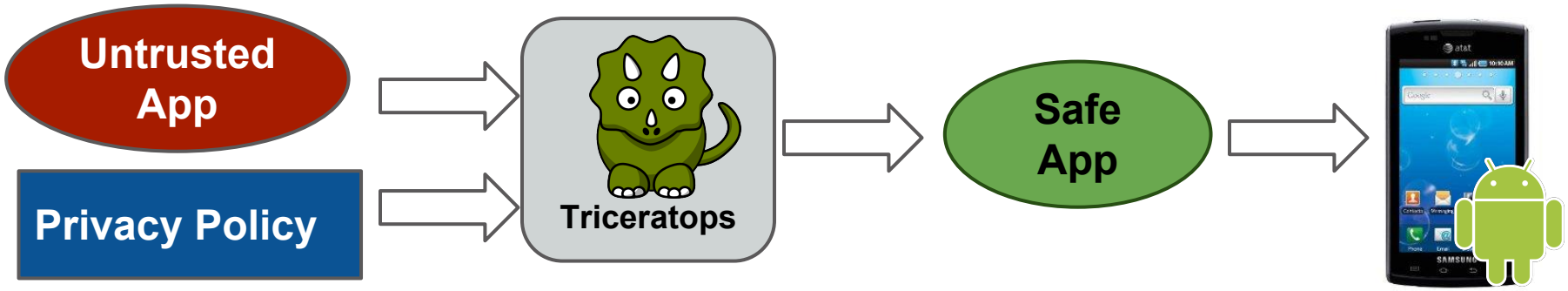
Filesystem -> Internet



Implementation

- Mainly built on top of Wala analysis framework
- Directly perform analysis on Dalvik bytecode (no need for source code)
- Use smali assembler and disassembler toolchain for instrumentation
- Existing API summary from SPARTA project

Triceratops Demo



Preliminary Evaluations

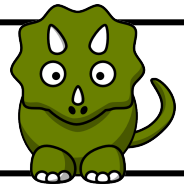
- Kittey Kittey
  - No Filesystem -> Internet
- SMS replicator
  - No SMS -> SMS before a button is clicked

Enforcement Overhead (# of additional instructions)

App	No Optimization	API Summary Relevant Code	Full Optimization
Kittey Kittey	2757	75/61	6/4
SMS replicator	886	20/13	4/3

Very low runtime overhead!

Preliminary Evaluations

Tools	Kittey Kittey	SMS Replicator	Root Cause
Android Permission System	✗	✗	No IF, CF
Pegasus [Chen'13]	☹	☹	Multiple code path to potential violation
TaintDroid [Enck'10]	☺	☹	No CF
Aurasium [Xu'12]	✗	☹	No IF
Triceratops 	☺	☺	Finer-grained privacy policy IF+CF

Supports more types of malware

Limitations

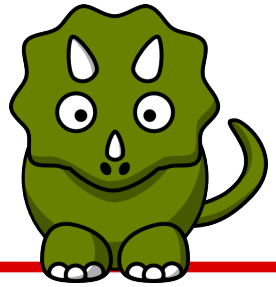
- Classical Java static analysis challenges
  - Reflection
  - Precision of points-to analysis
- Static modeling of Android runtime behavior
  - Dynamically register a callback function to a button
- Completeness of the API summary
- Native code

- Can be addressed by other research

Future Work

- Implicit Flow
  - Static analysis assisted dynamic analysis can be used to track implicit flow while achieving high precision
- Data tracking mechanisms for persistent storage mediums and side channels
  - Databases and file systems
  - Displaying sensitive information on screen, then take a screenshot

Conclusion



A powerful **enforcement tool** that allows users to enforce **fine-grained privacy policies** on a given mobile app

- Finer-grained privacy policy (IF+CF)
  - Defend against more types of malicious apps
- Static optimized dynamic enforcement
  - Portable, low runtime overhead, and no false positives