S2LAB

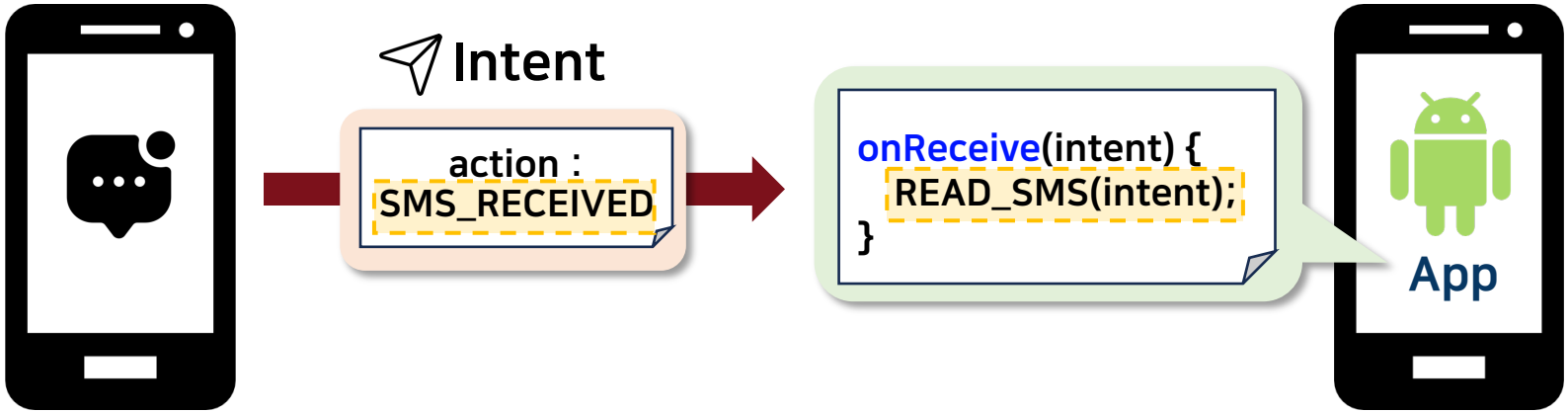
Intent-aware Fuzzing for Android Hardened Application

Seongyun Jeong[1,2], Minseong Choi[1,2], Haehyun Cho[3],
Seokwoo Choi[4], Hyungsub Kim[5] , Yuseok Jeon[1,2]

[1] KOREA UNIVERSITY [2] UNIST [3] Soongsil University [4] NSR National Security Research Institute [5] INDIANA UNIVERSITY

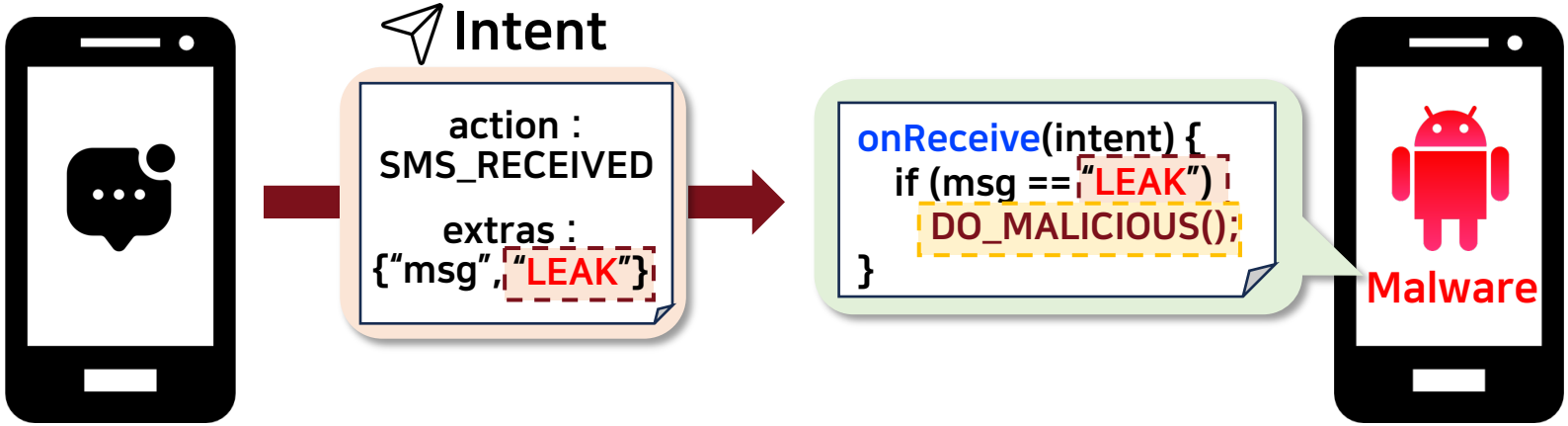
Intent

- Intent is a message object used for inter-component communication



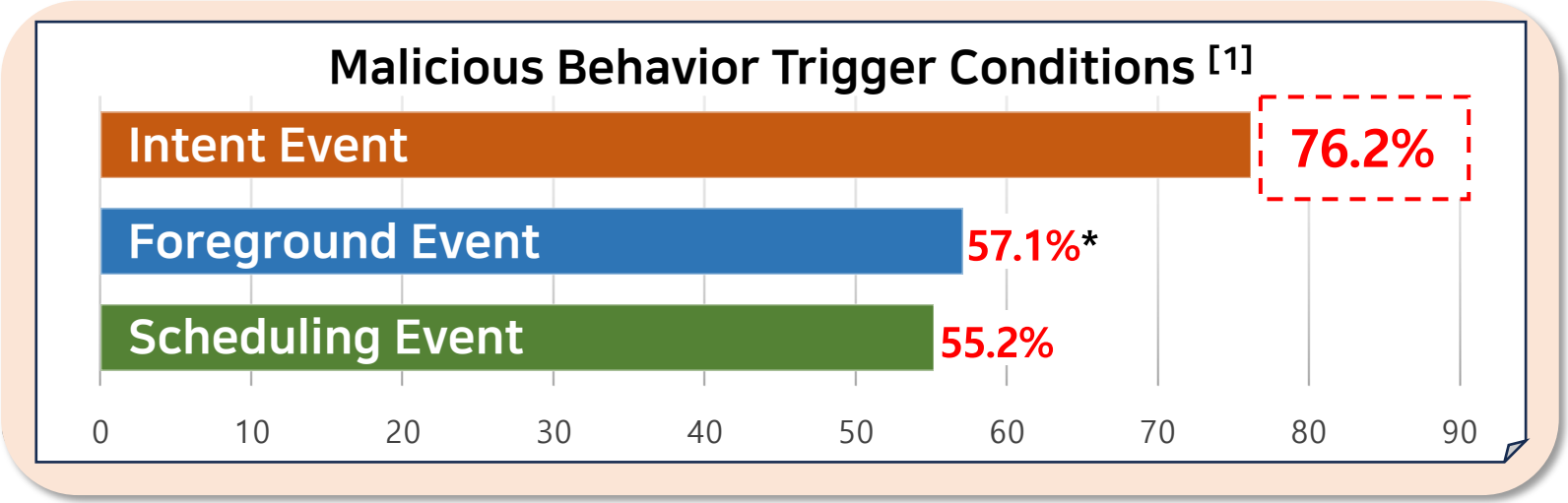
Importance of Intent

- Intent can contain extra data and influence an app's execution flow



Importance of Intent

- Intent can contain extra data and **influence an app's execution flow**

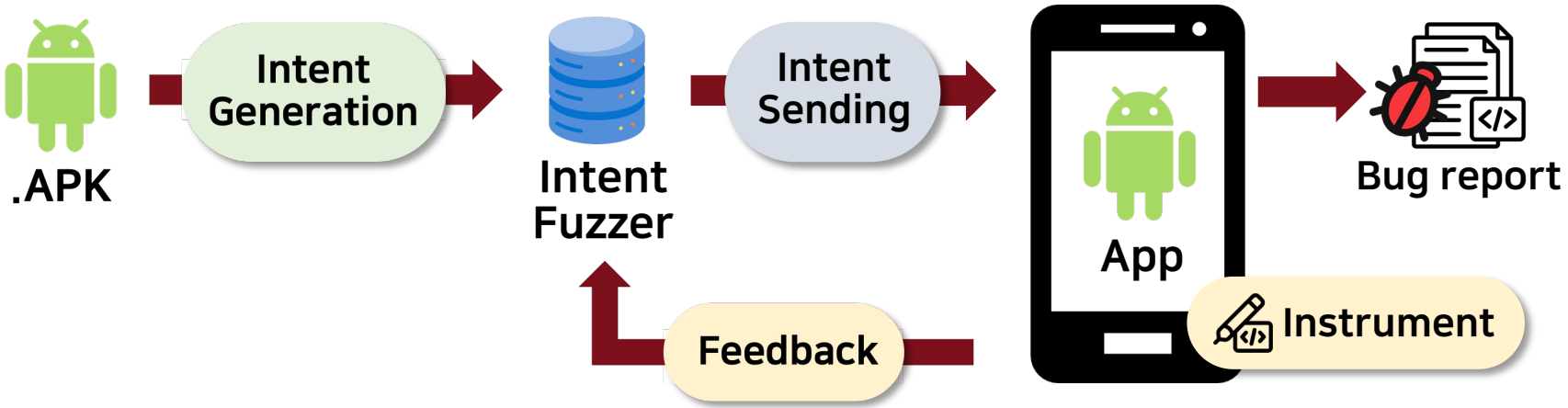


76.2% of malware leverage Intent

[1] : Rotten Apples Spoil the Bunch: An Anatomy of Google Play Malware (ICSE'22)
* Excluded foreground events triggered by app launch

Intent Fuzzing

- For Intent fuzzing, generating valid, interesting Intent is important



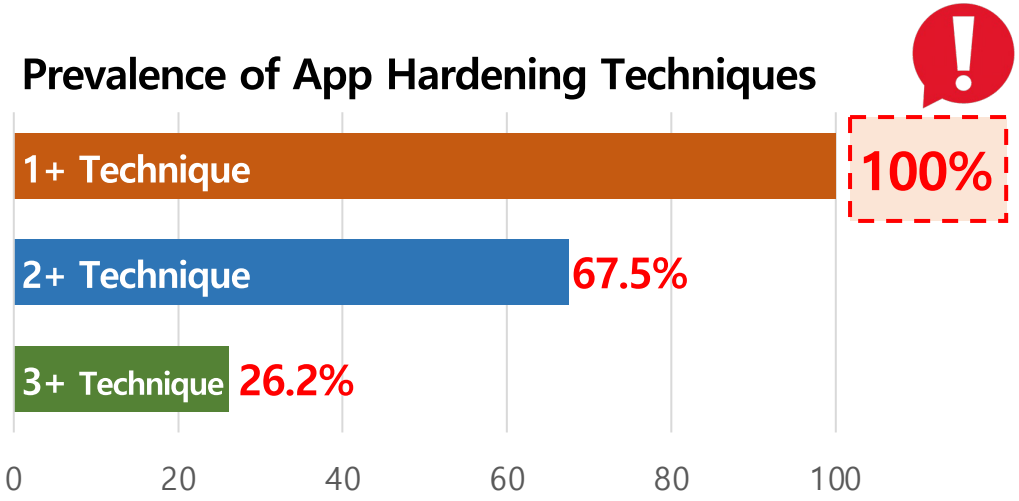
Intent Fuzzing

- For Intent fuzzing, generating valid, interesting Intent is important

What happens if the target application uses hardening techniques?

Hardening Techniques

- At least one hardening technique is adopted in most downloaded 200 apps
- Hardening techniques (e.g., obfuscation, packer) hinder Intent fuzzing



	Benign (%)	Malicious (%)
Obfuscation	39	22
Packer	22	94
Anti-Debugging	74	9
Anti-Emulator	98	33

Hardening Techniques

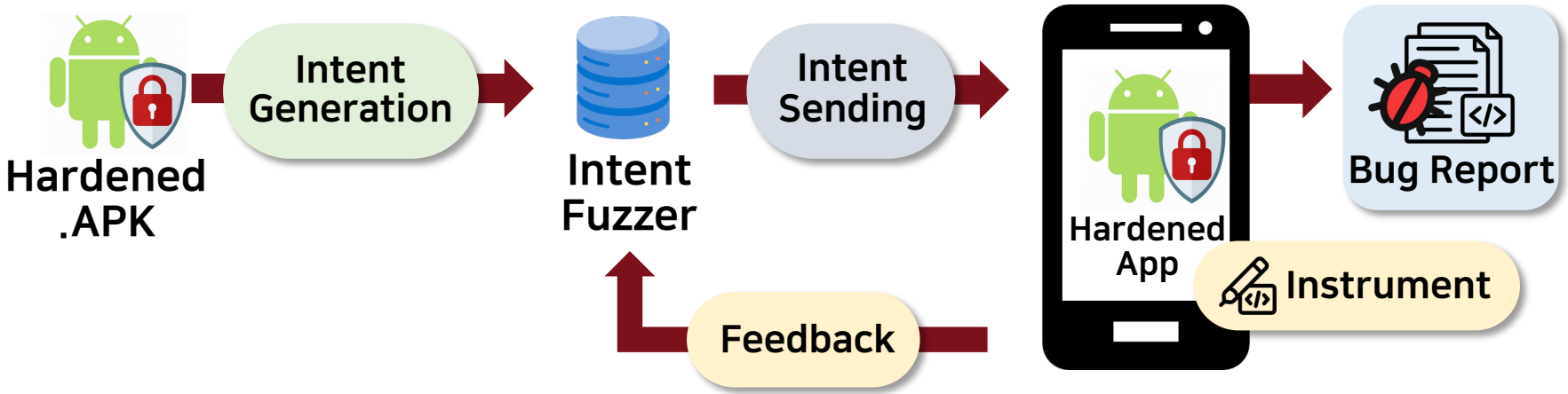
- At least one hardening technique is adopted in most downloaded 200 apps

However, existing Intent fuzzers do not consider hardening techniques

Anti-Emulator	98	33

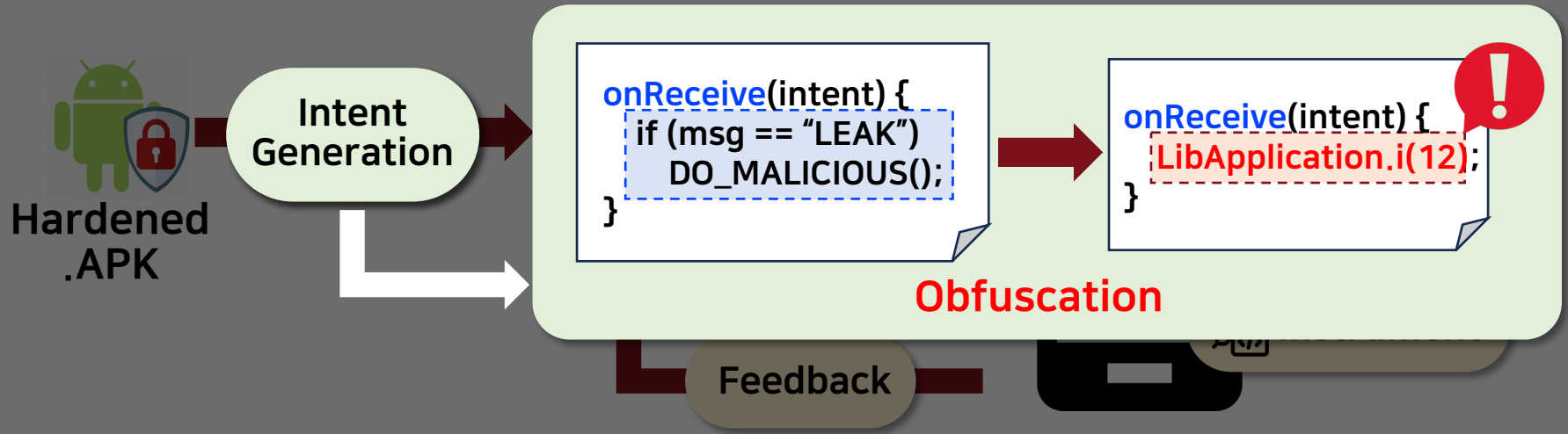
Challenge #1: Valid Intent Generation

- To generate valid Intent, extracting Intent-related information is needed



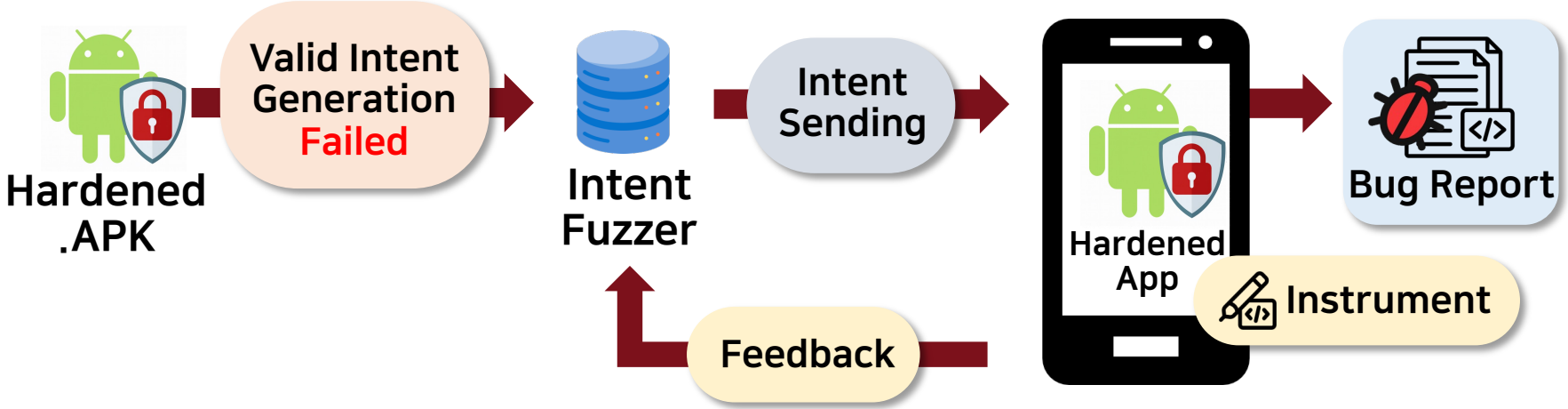
Challenge #1: Valid Intent Generation

- To generate valid Intent, extracting Intent-related information is needed



Challenge #1: Valid Intent Generation

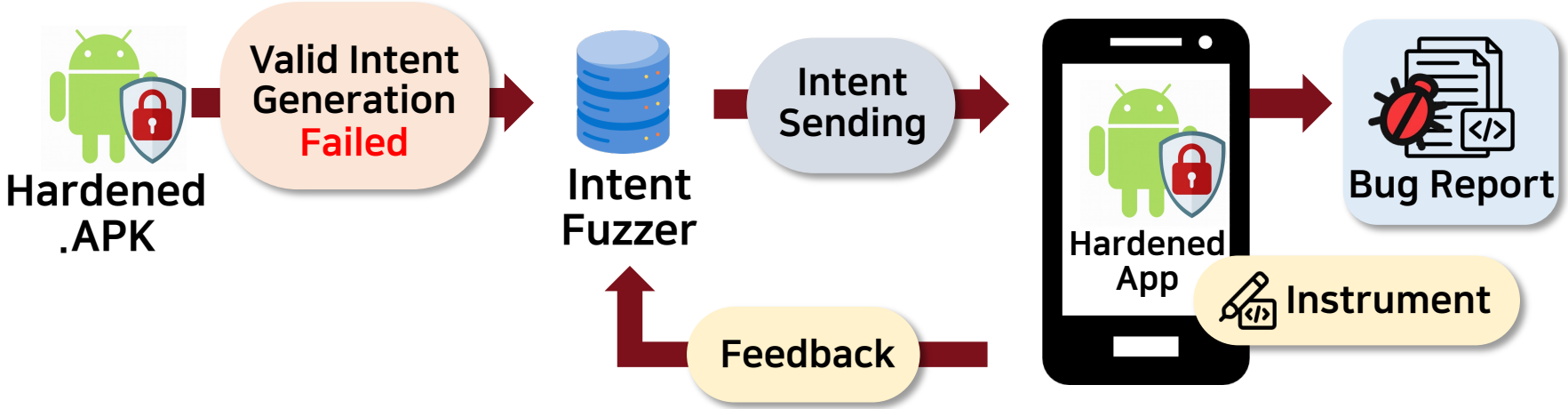
- To generate valid Intent, extracting Intent-related information is needed



Hardening techniques hinder the extraction of Intent-related information

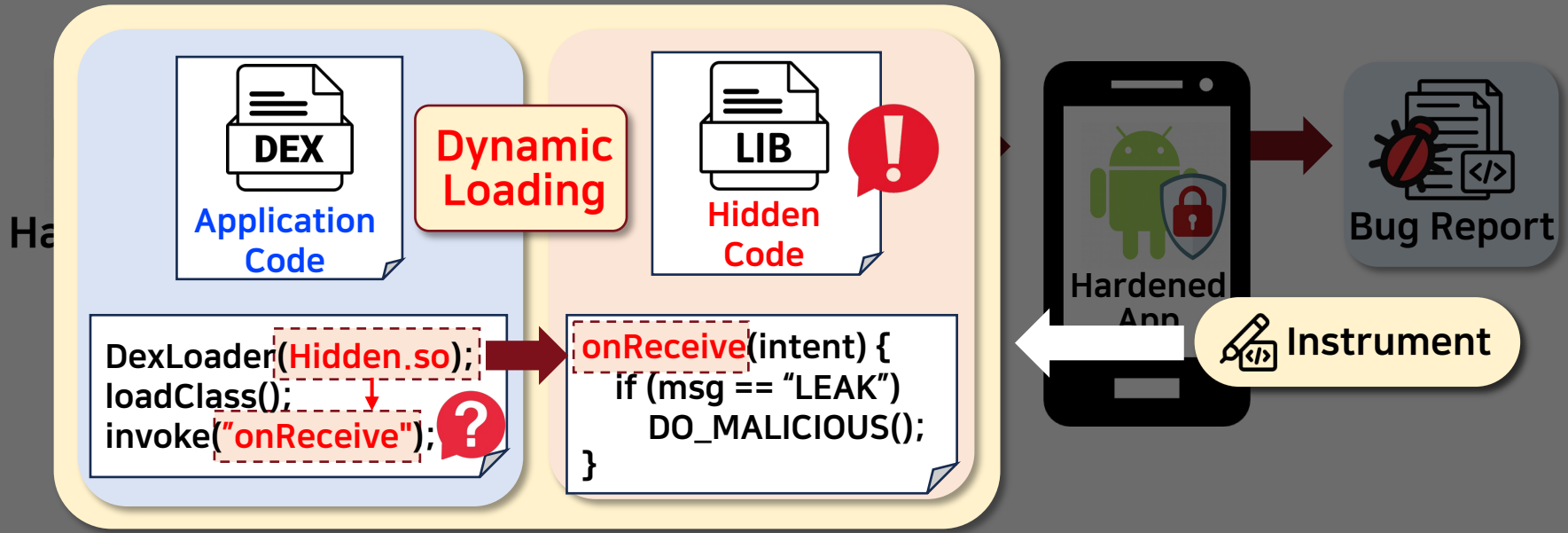
Challenge #2: Coverage Feedback Generation

- To measure the impact of Intent inputs, coverage feedback is needed



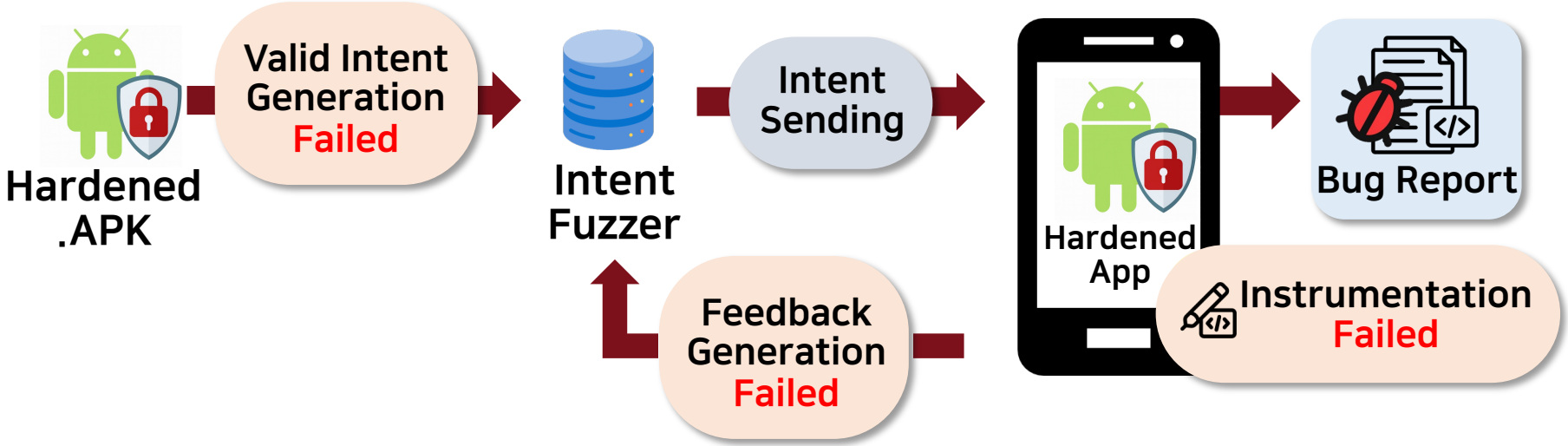
Challenge #2: Coverage Feedback Generation

- To measure the impact of Intent inputs, coverage feedback is needed



Challenge #2: Coverage Feedback Generation

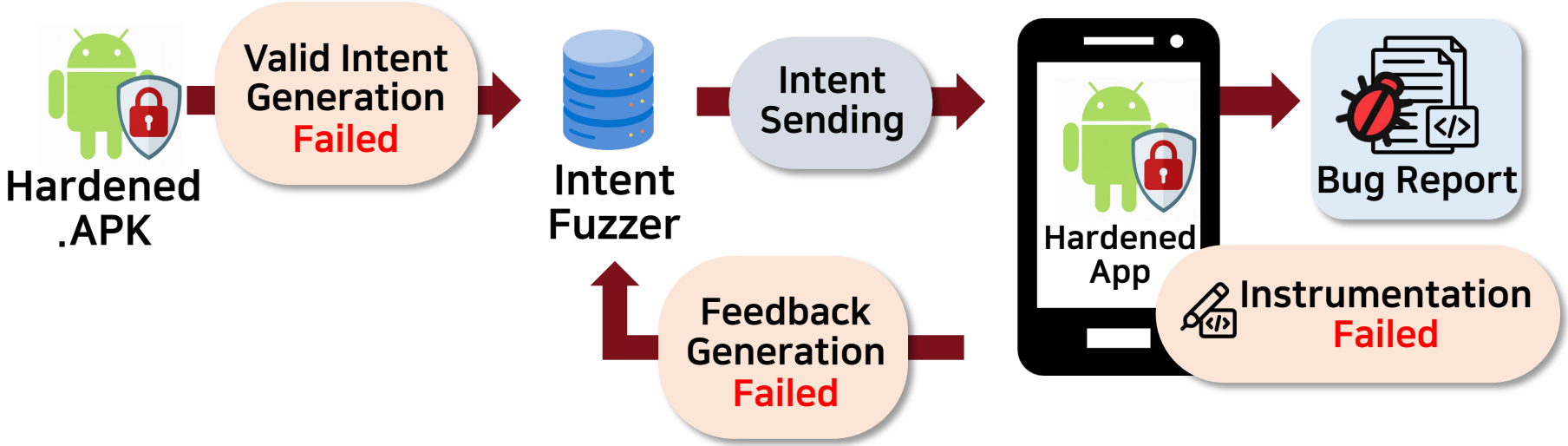
- To measure the impact of Intent inputs, coverage feedback is needed



Hardening techniques complicate identifying instrumentation target

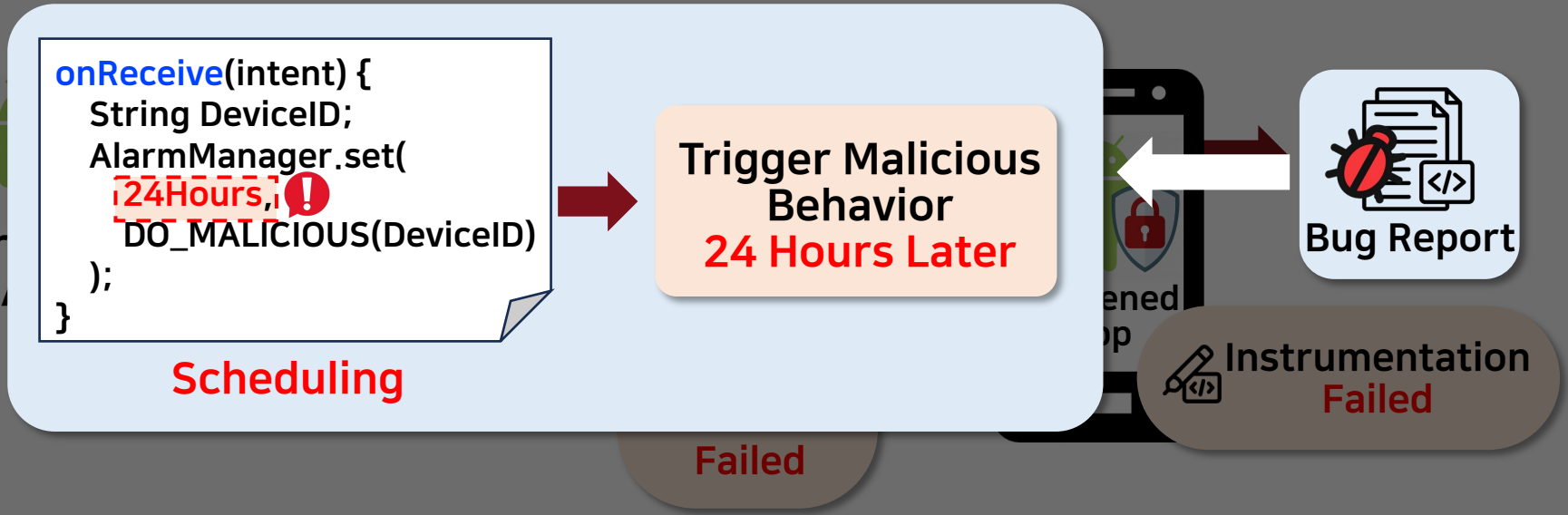
Challenge #3: Hidden Bugs Triggering & Detecting

- Even if fuzzer reaches suspicious code, triggering & detecting bugs is needed



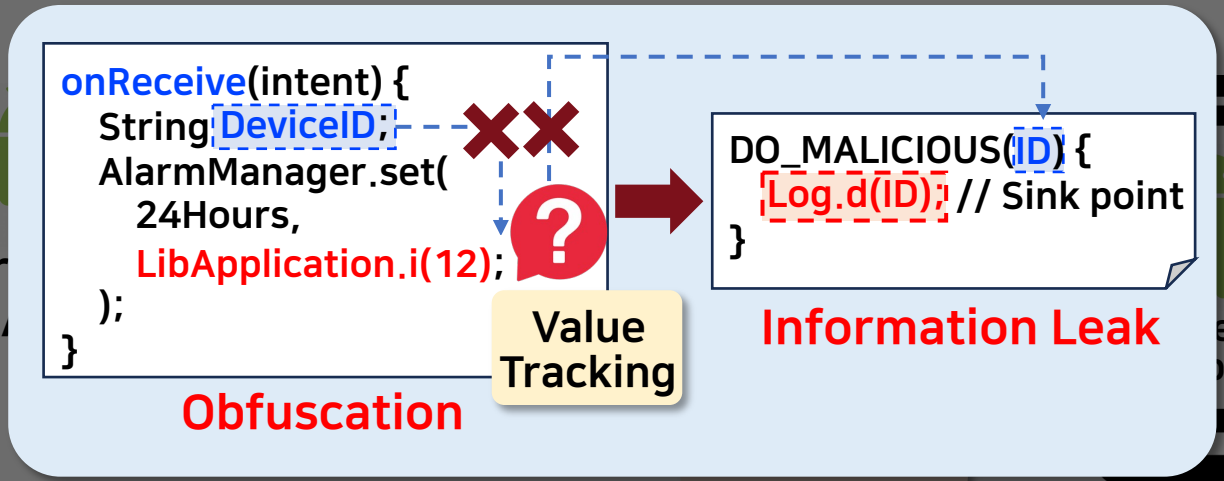
Challenge #3: Hidden Bugs Triggering & Detecting

- Even if fuzzer reaches suspicious code, triggering & detecting bugs is needed



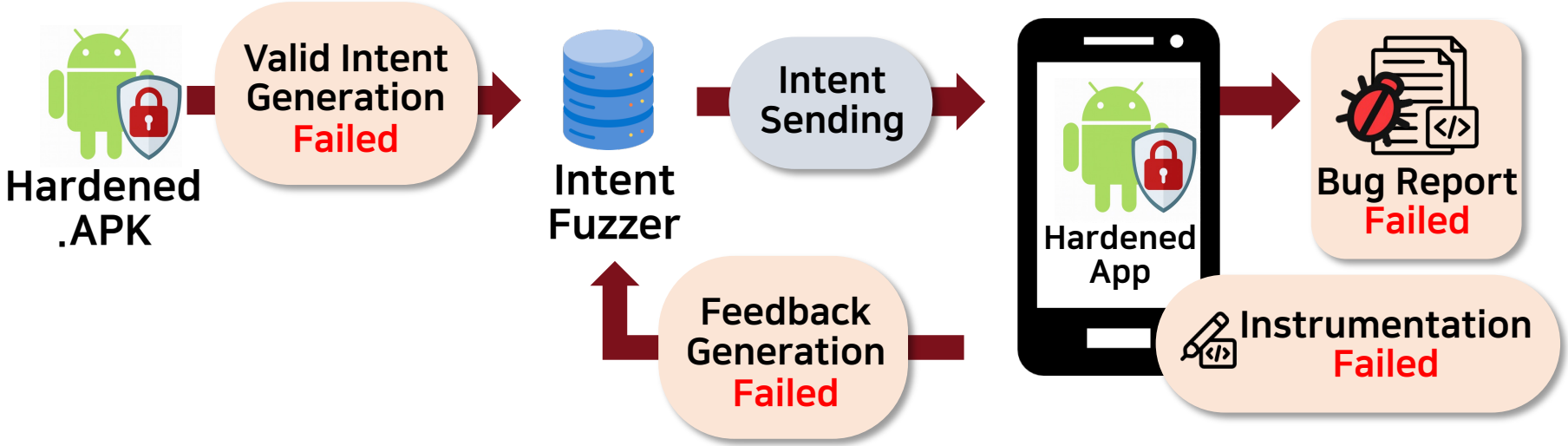
Challenge #3: Hidden Bugs Triggering & Detecting

- Even if fuzzer reaches suspicious code, triggering & detecting bugs is needed



Challenge #3: Hidden Bugs Triggering & Detecting

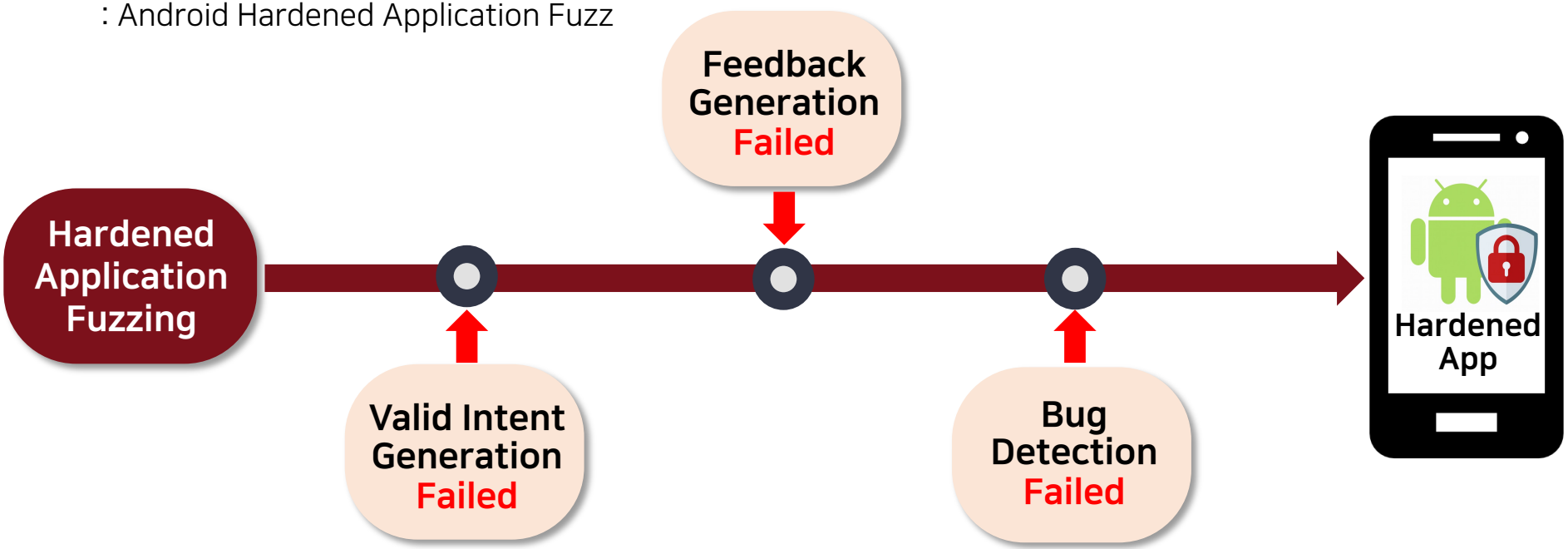
- Even if fuzzer reaches suspicious code, triggering & detecting bugs is needed



Miss bugs triggered under scheduling, or fail to detect triggered bug

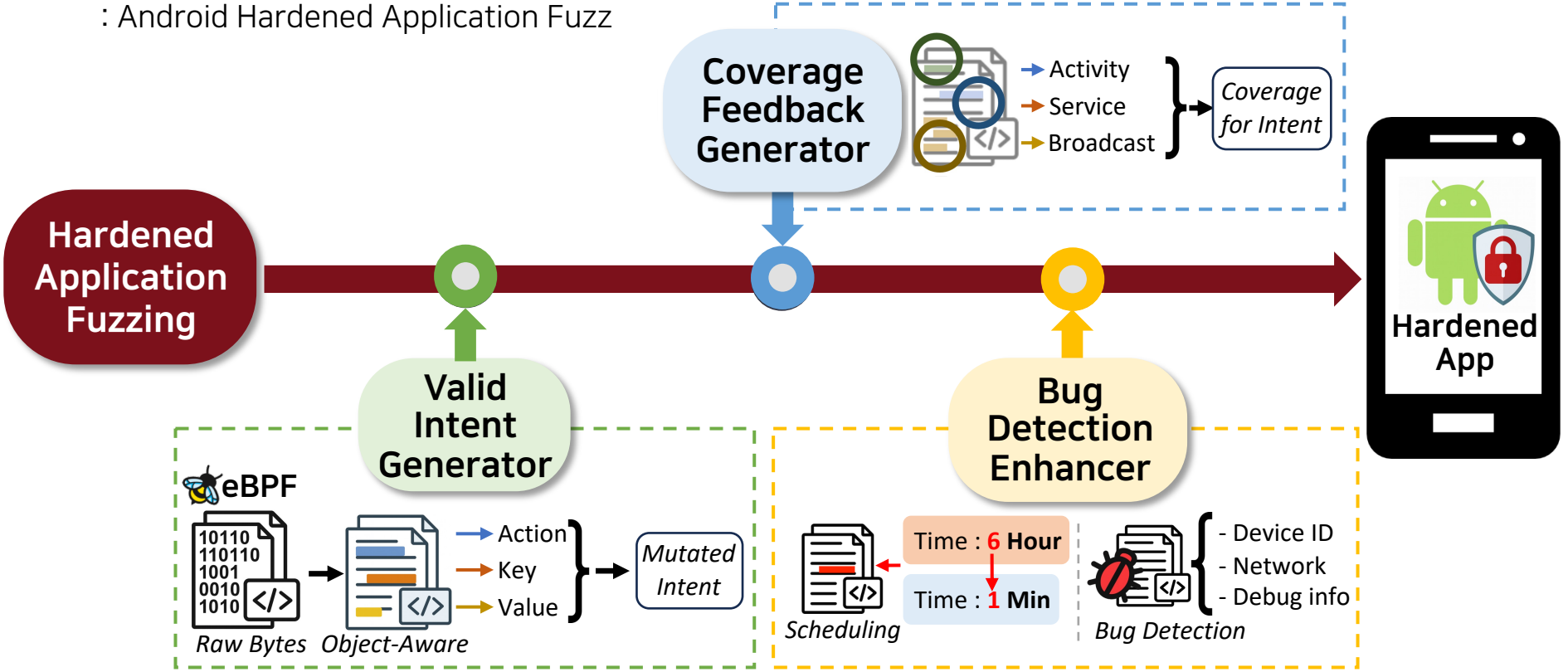
AHA-Fuzz Overview

: Android Hardened Application Fuzz



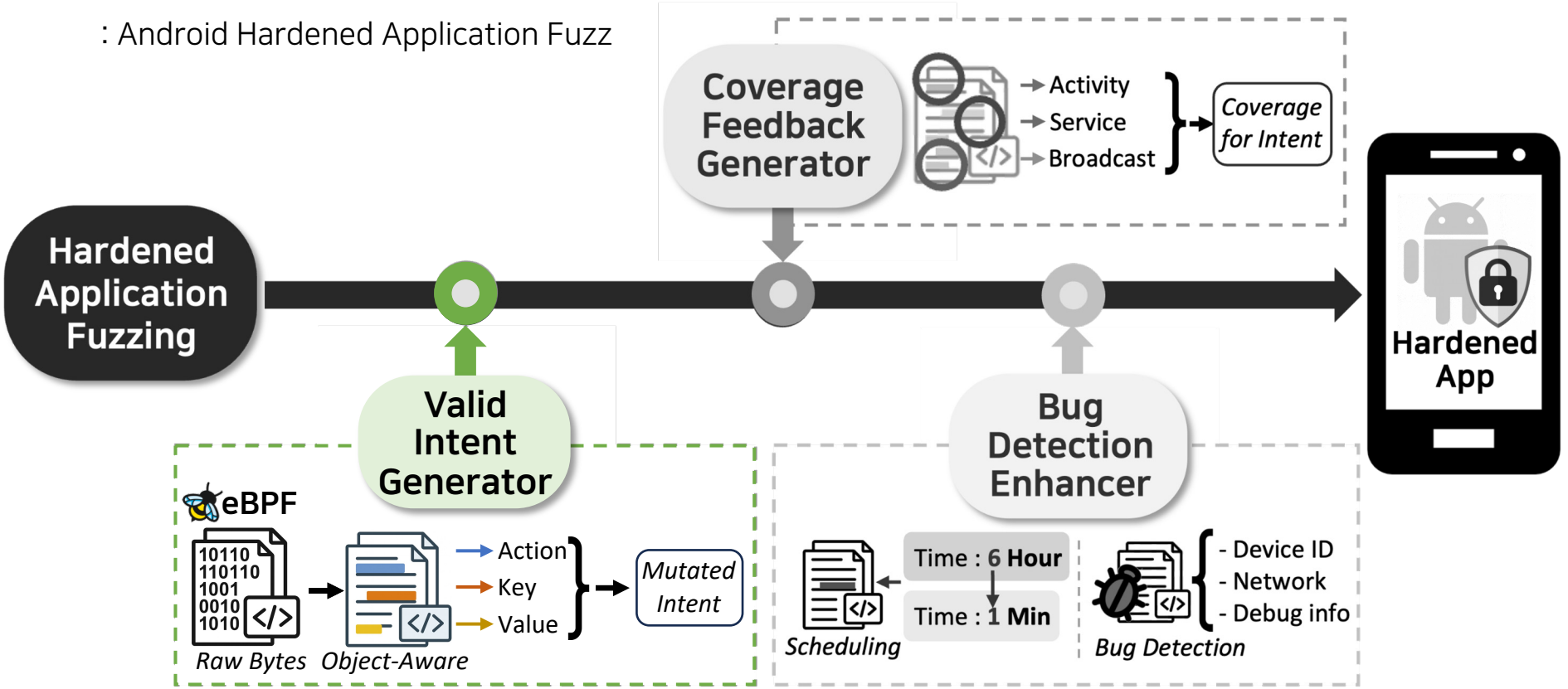
AHA-Fuzz Overview

: Android Hardened Application Fuzz



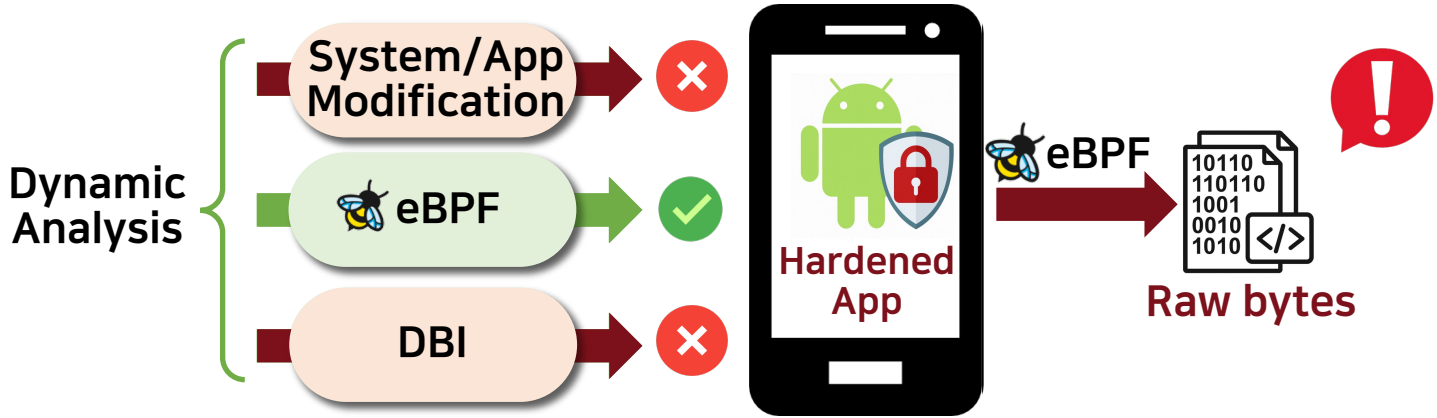
AHA-Fuzz Overview

: Android Hardened Application Fuzz



Extracting Information in Hardened App

- **Step 1. extract information at runtime in hardened environment**
  - **eBPF enables instrumentation without interference** from hardening techniques
  - **Limitation** : can **only access raw bytes** in memory, making Java objects **hard to observe**

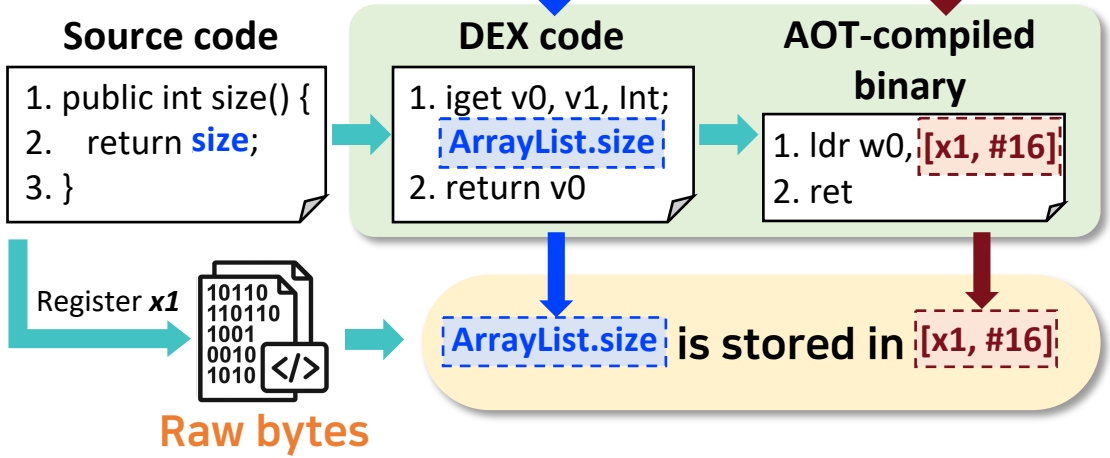


Need to recover Java object layout information

Recovering Java Object Layout

- **Solution : recover layout by mapping DEX code and AOT-compiled binary**
  - **DEX code**: contains field name, but **not offset information**
  - **AOT-compiled binary :** contains offset information, but **not field name**

- Example in ArrayList.size



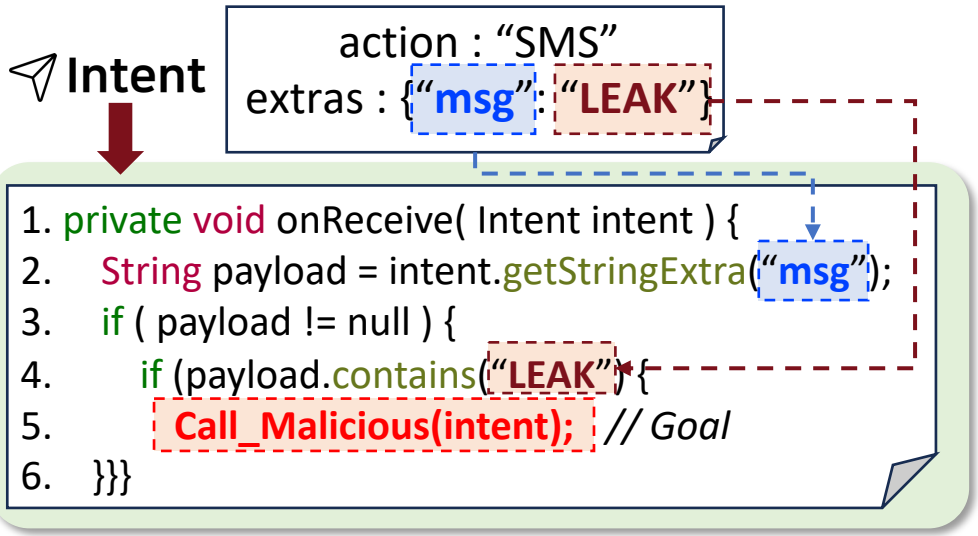
Recover 82.8% of Java object layout

⬇

Information extraction success

Valid Intent Generator

- **Step 2. generate Key-Value feedback that can trigger deeper code**
  - **Key-Value feedback** : generate **Intent metadata** (key-value pair) and guide **input mutation**
  - **Coverage feedback** : measure **input impact** and guide **effective input selection**
- Example of how key-value pairs are used in Intent



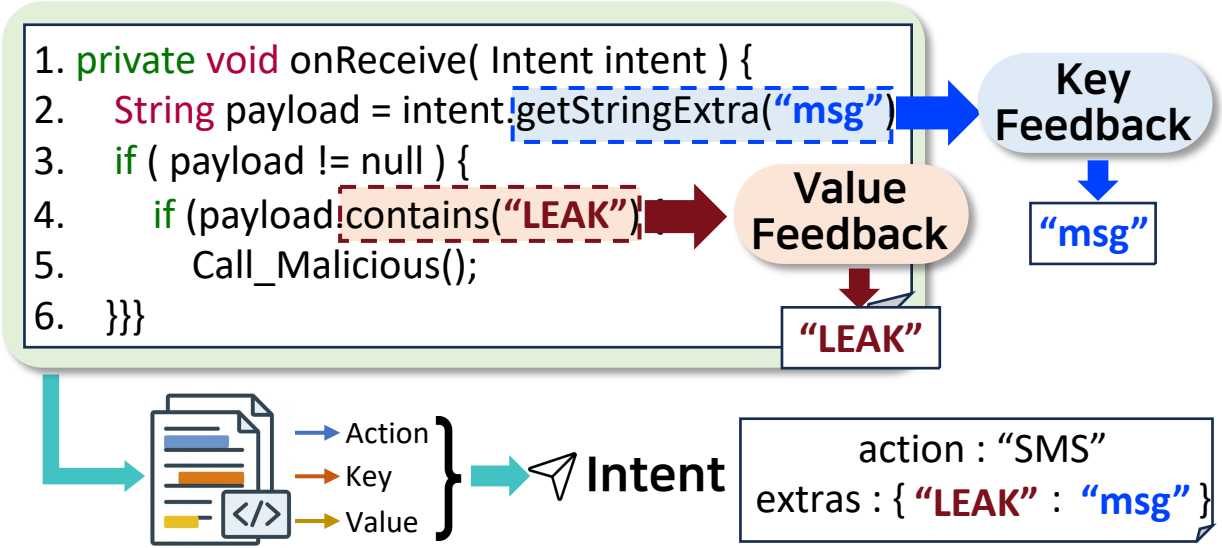
Intent validation code are stored in application

⬇

Key-Value feedback is needed

Key-Value Feedback

- **Hook Intent-related operation & generate feedback**
  - **Observation :** Intent metadata (extras) are **used in comparison operation**
  - **Key feedback** : hooking **predefined extras operation** (e.g., getStringExtra)
  - **Value feedback :** hooking **comparison operation** (e.g., contains, equals)



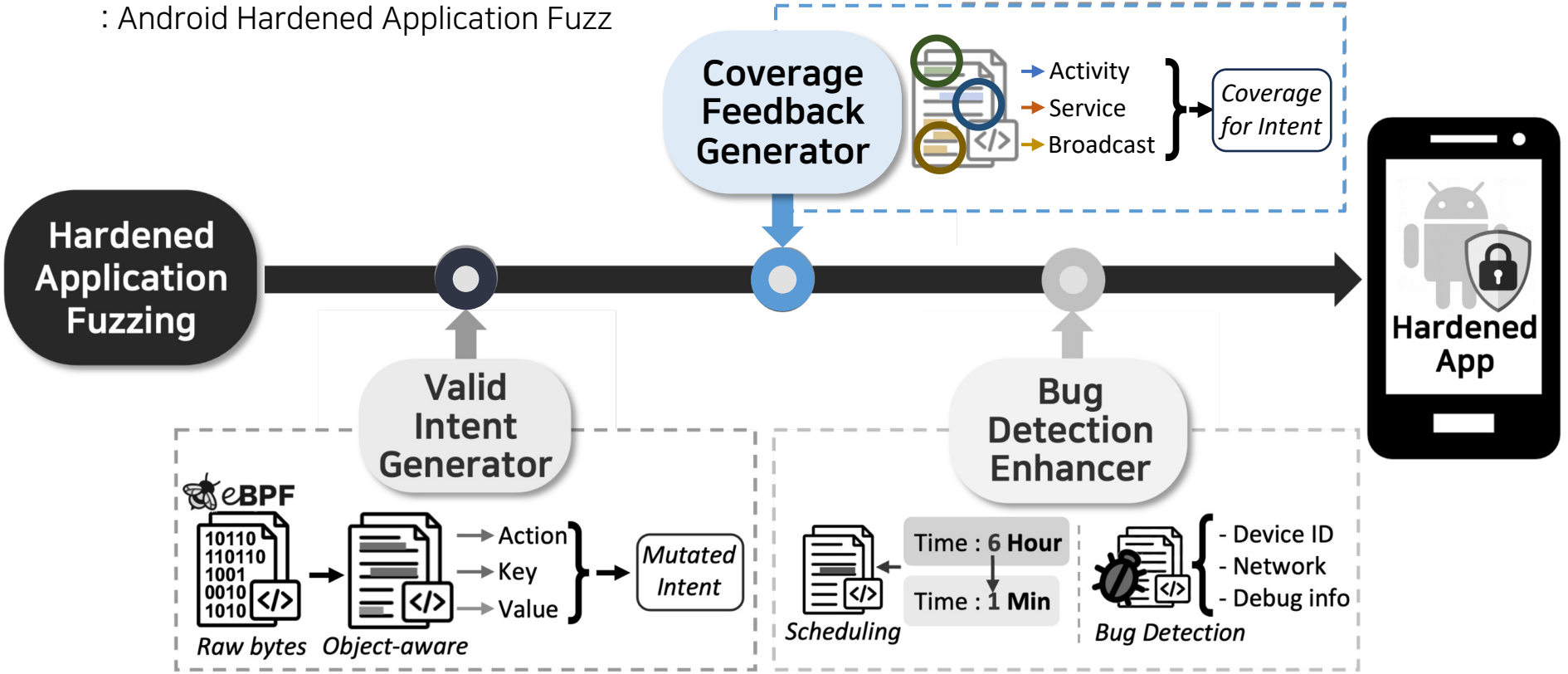
Cover 93.5% extras values

⬇

Generate valid Intent

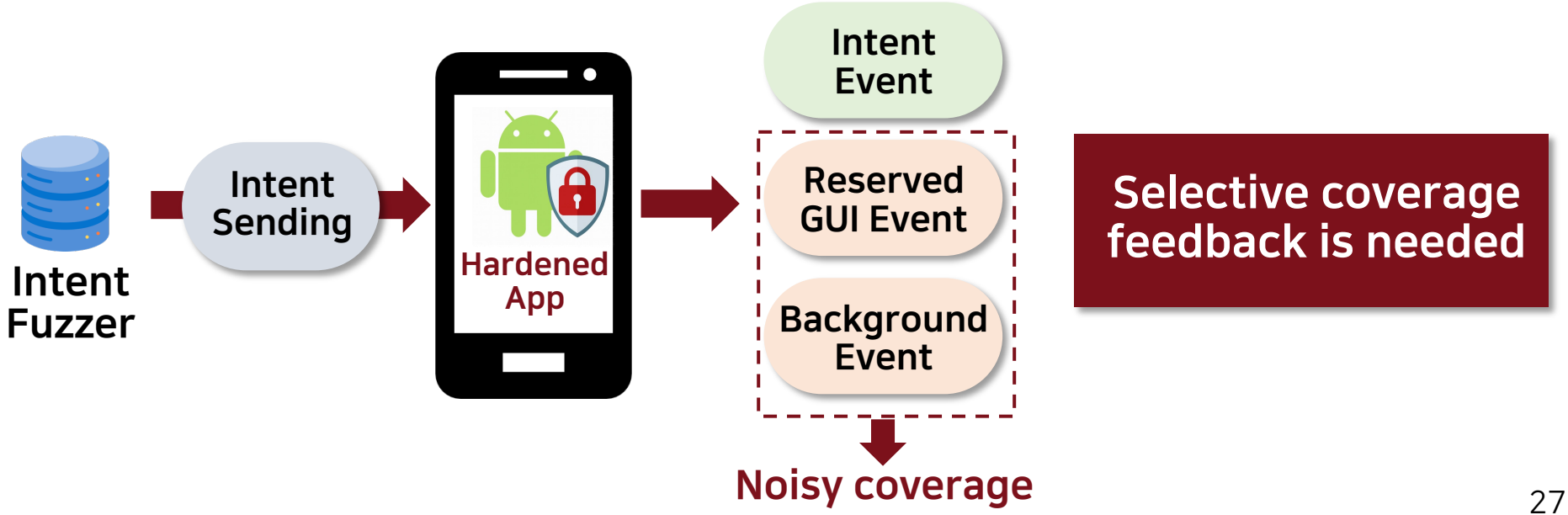
AHA-Fuzz Overview

: Android Hardened Application Fuzz



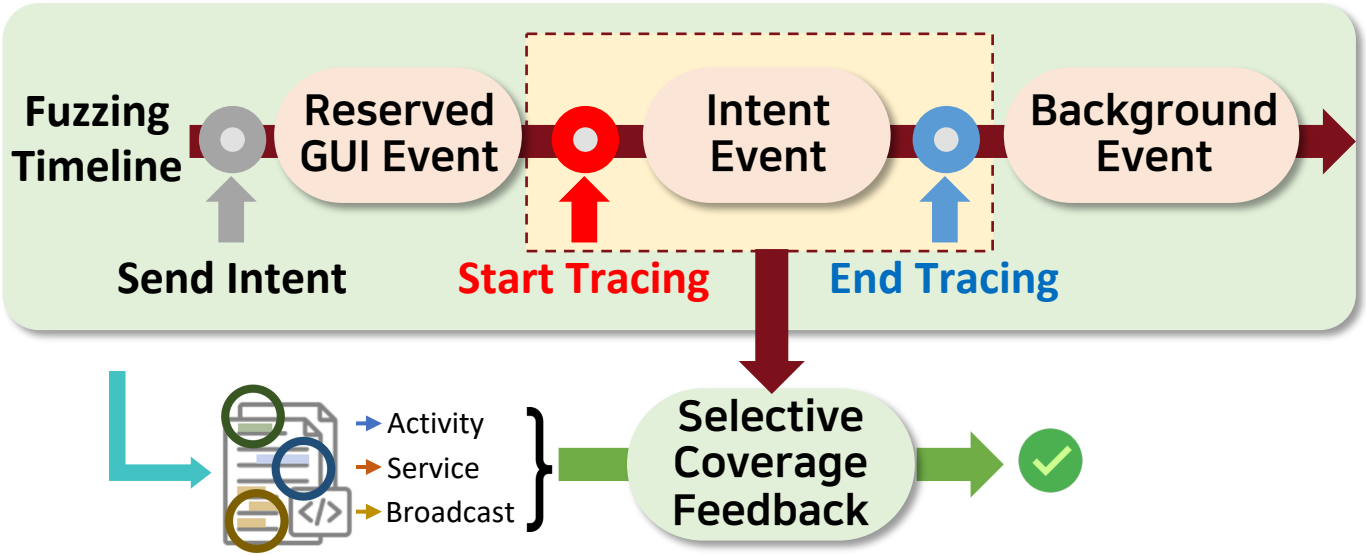
Coverage Feedback Generator

- **For Intent coverage feedback, Intent-specific feedback is needed**
  - **Noisy coverage** (e.g., GUI event) can **negatively impact** **Intent fuzzing**



Selective Coverage Feedback

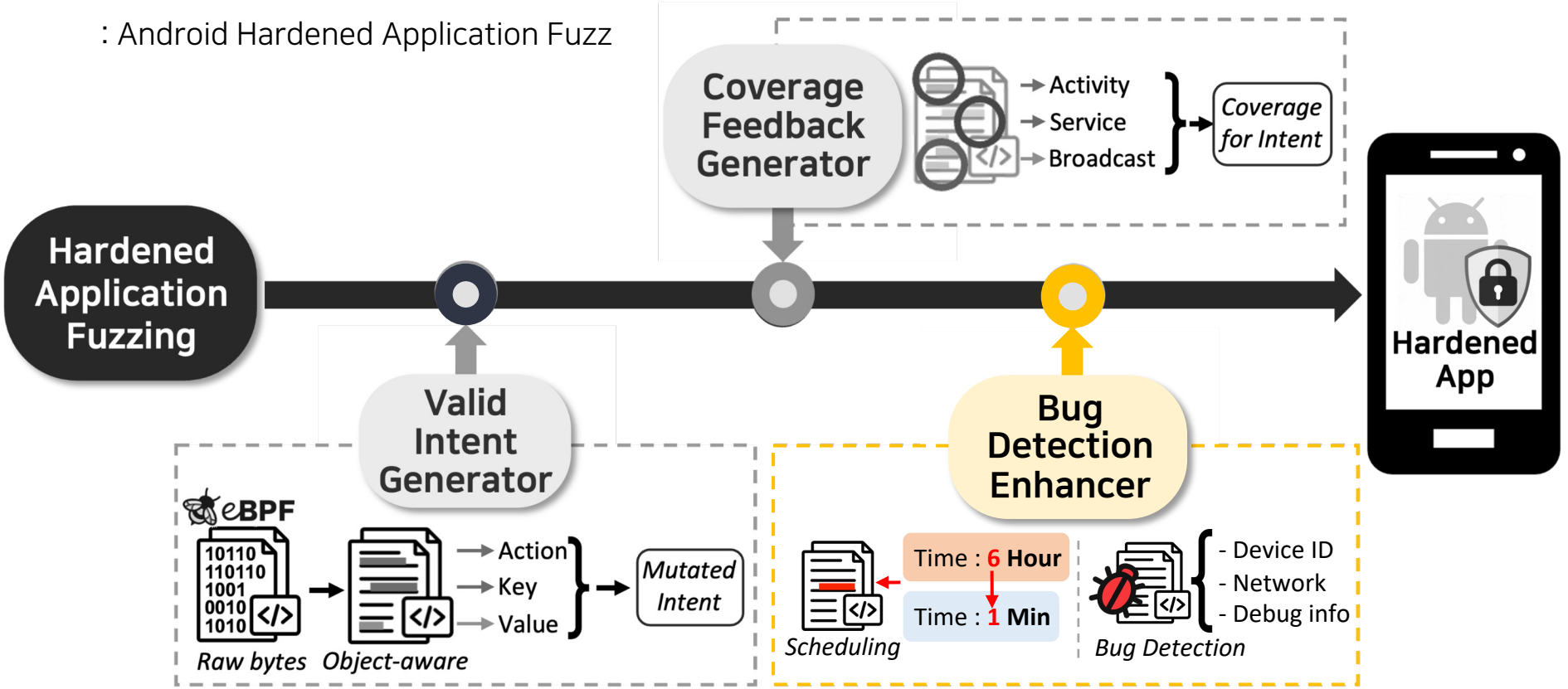
- **Tracing individual Intent event execution flow, generate selective feedback**
  - **Observation** : can **define Intent execution pattern** (e.g., start & end point)



Filter out noisy event, focus on only Intent-related event

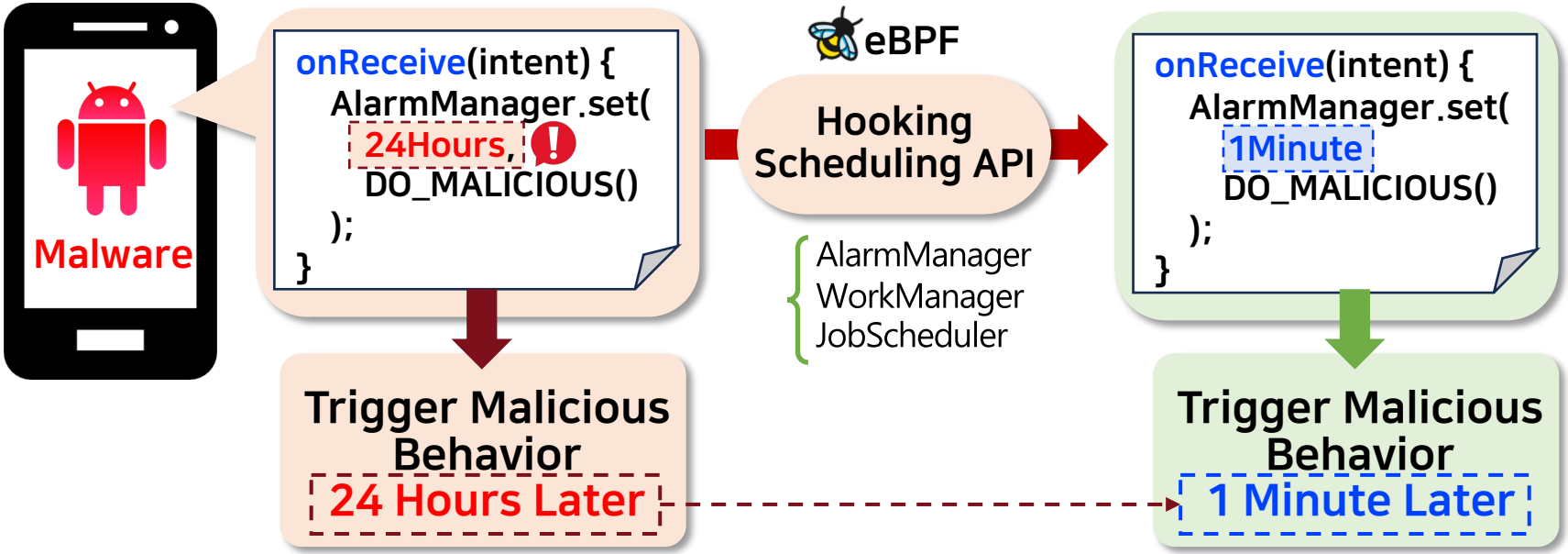
AHA-Fuzz Overview

: Android Hardened Application Fuzz



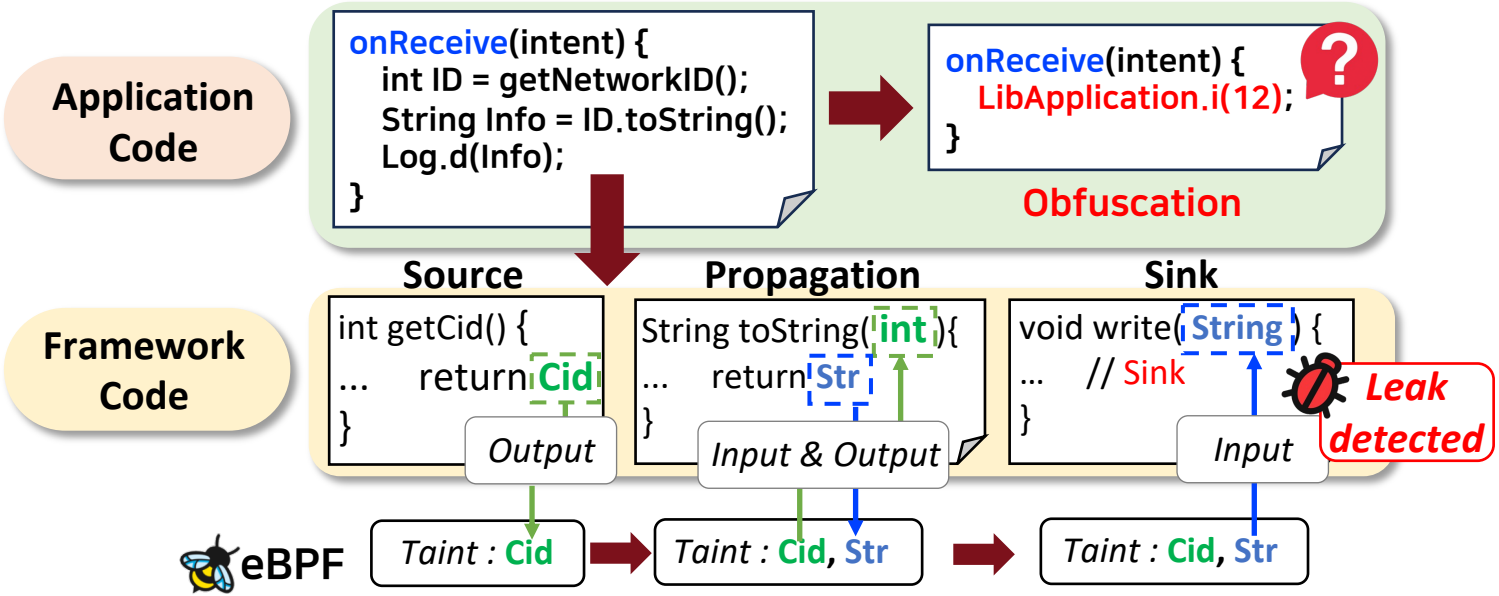
Triggering Scheduling Event

- **Handle scheduling event** : **overwrite scheduling time** **for fast-forward invoking**
  - **Hooking all scheduler-related APIs**, adjust the associated scheduling times



Detecting Information Leak

- **Detect information leak : lightweight method-level taint analysis**
  - **Tracking** primitive type values pass **through input & output**
  - **Observation** : **sensitive information** typically stored and manage as **primitive type**



Evaluation

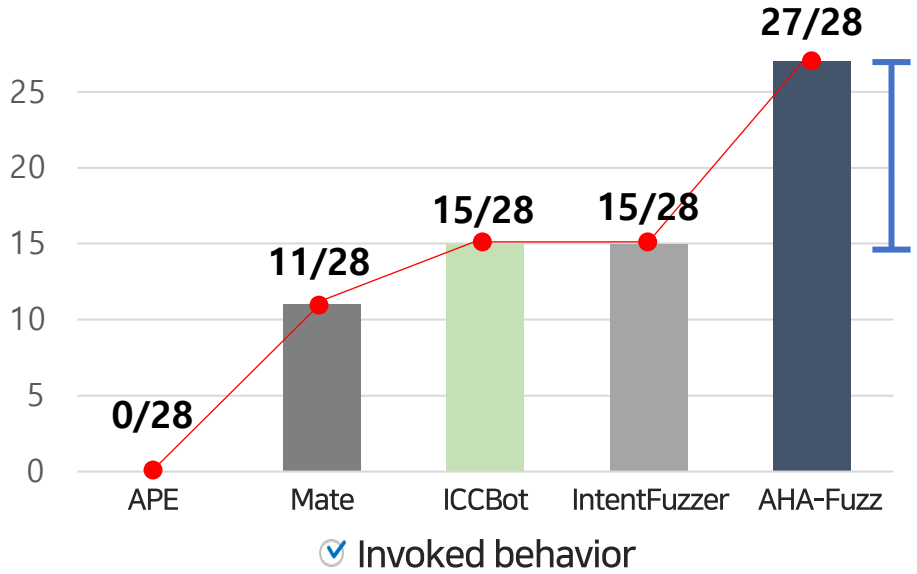
AHA-Fuzz Evaluation

- **RQ #1 : Effectiveness of fuzzing**
- **RQ #2 : Code Coverage**
- **RQ #3 : Bug Detection**

- **Baseline Fuzzers**
  - **APE** (ICSE'19) : **GUI** Fuzzer
  - **Mate** (ICST'23) : GUI + Intent fuzzer with **intra-procedural** analysis
  - **ICCBot** (ICSE'22) : Intent fuzzer with **inter-procedural** analysis
  - **IntentFuzzer** (AsiaCCS'14) : Intent fuzzer with **hybrid** analysis
  - **AHA-Fuzz** : **GUI +** Intent fuzzer with **key-value & coverage feedback**

Effectiveness of Intent Fuzzing

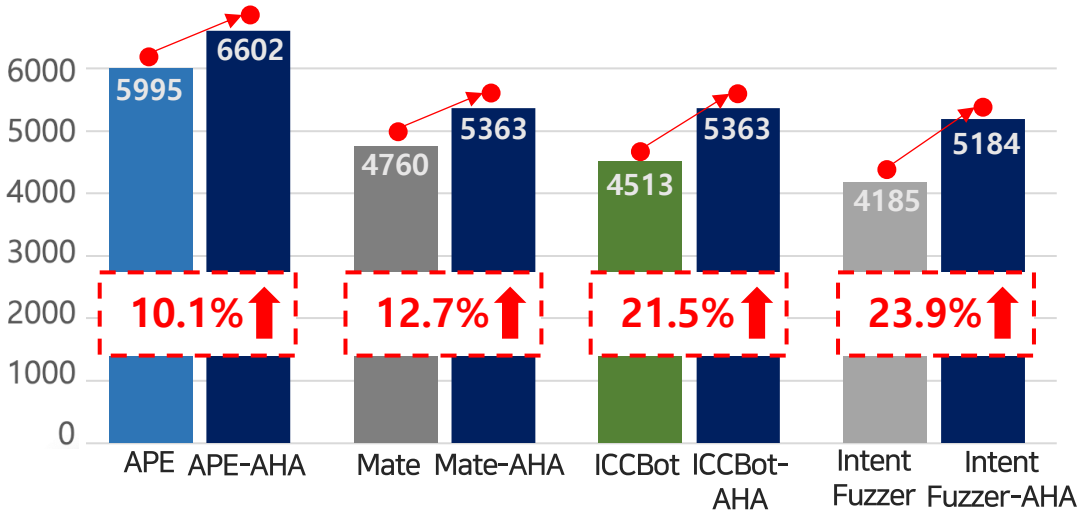
- **Evaluated how Intent fuzzer can create valid intents used in malware**
  - Target : Among 105 Google Play malware, find 13 reproducible apps
  - Select **28** malicious behavior triggered by Intent event
  - **AHA-Fuzz triggered all malicious behavior except for one**



Trigger 92.3% more Intent events

Code Coverage

- **Evaluated method coverage increase in benign/malicious apps**
  - Target : Top downloaded 20 benign / 20 malicious apps
  - **AHA-Fuzz triggered 23.9% more method than previous approach**



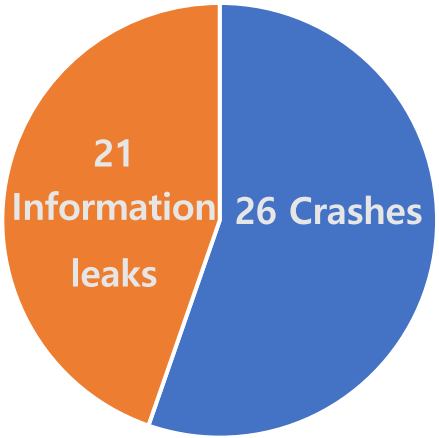
Average of invoked method both benign/malicious apps

Trigger 23.9% more method

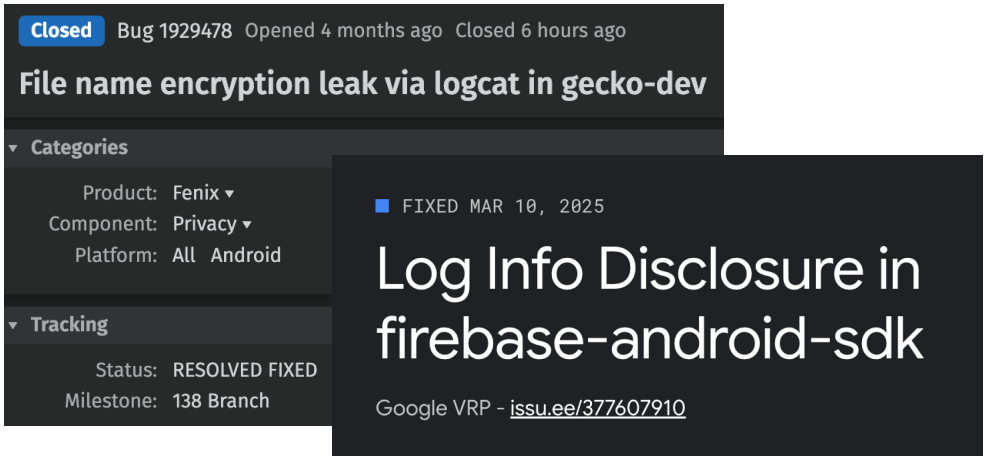
- -AHA : Variant version of each fuzzer, replace each Intent fuzzer to AHA-Fuzz algorithm

Discovering Unknown Bugs

- **Evaluated AHA-Fuzz can discover previously-unknown bugs**
  - Target : Top downloaded 300 google play apps
  - AHA-Fuzz discovered **47 unknown bugs** that previous works cannot detect
  - **Google, Firefox, and Facebook have acknowledged 6 bugs, 3 of them fixed**



- **Total 47 bugs**



- **Bug fixed in Firefox, Google SDK**

Conclusion

- **Introduce AHA-Fuzz, the Intent-aware greybox fuzzer for hardened android app**
  - Valid Intent generator by recovering object layouts and leveraging key-value feedback
  - Selective coverage feedback for effective fuzzing
  - Triggering hard-to-trigger events, increase bug detection capability

- **AHA-Fuzz shows better performance compared to previous works**
  - Trigger 92.3% more Intent patterns in malware (3.45X faster)
  - Invoke 23.9% more methods
  - Find 47 previously unknown bugs that previous works cannot find
  - 6 bugs have been acknowledged by Google, Firefox and Facebook, and 3 have been fixed

https://github.com/S2-Lab/AHA-fuzz