# Android-Application Rewriting Guided by Quantitative Information Flow

Ke Tian, Danfeng (Daphne) Yao
Dept. of Computer Science
Virginia Tech
{ketian, danfeng}@cs.vt.edu

Gang Tan
Dept. of Computer Science & Engineering
Penn State University
gtan@cse.psu.edu

## Motivation:

- Conventional app-screening approaches are passive as they are not designed to make security enhancements to the app code.
- Current all-or-nothing verification cannot prevent vulnerable apps that are in the gray area.

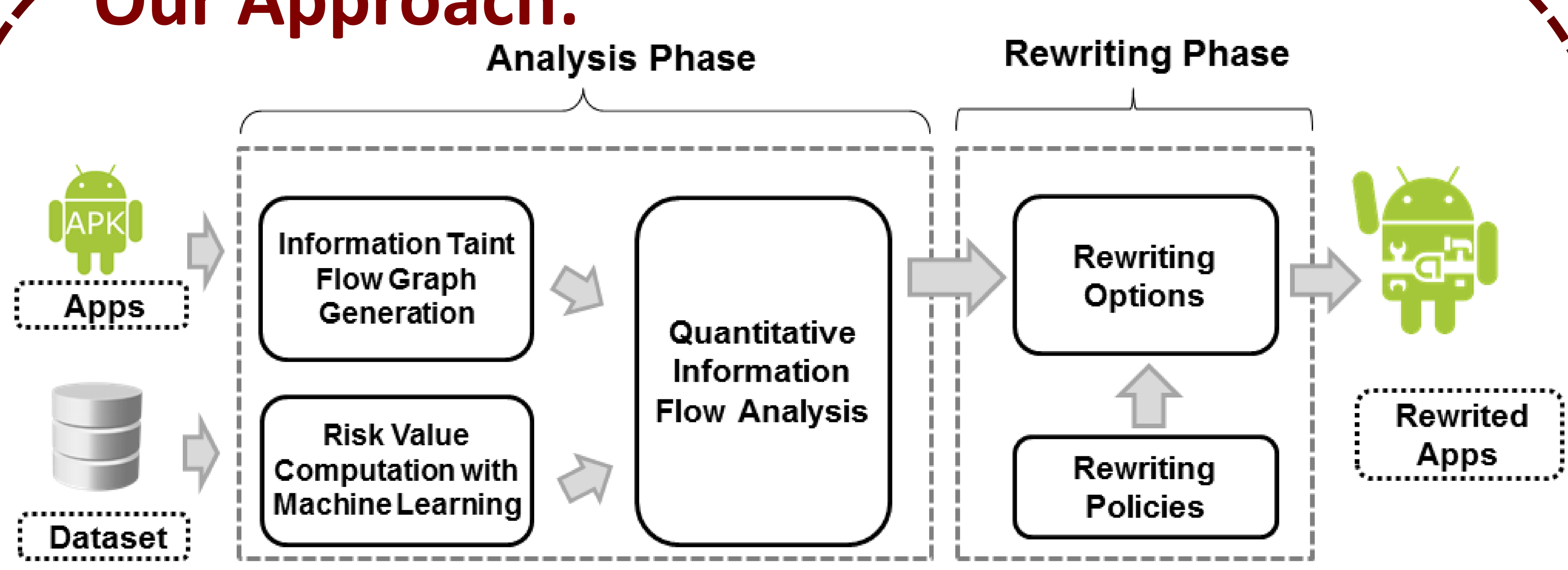
## Security Applications of Android Rewriting:

- External runtime monitoring (e.g., preventing data exfiltration and privacy leakage).
- Code reduction (e.g., removing certain code to eliminate apps' the overall risk).
- Inlined code insertion for monitoring Insert security checks and assertions (e.g., for authentication, logging).

## Threat Model:

- Vulnerable Android apps can expose and exfiltrate sensitive data (privacy leakage, e.g., sending sensitive device ID through a HTTP connection).
- Vulnerable interfaces of privileged Android apps can be exploited by malicious apps (confused deputy, e.g., intercepting communication channels for the malevolent purpose).

## Our Approach:




### Analysis Phase:

- Utilizing machine learning to map permissions to quantitative values representing security risks.
- Constructing the information taint flow graph and Initializing the graph with risk value assignment.
- Analyzing propagation of permissions and calculating risk scores of sinks.

### Rewriting Phase:

- Generating rewriting policies with constraints (e.g., register integrity, execution completeness).
- Extracting rewriting rules combined with analysis results to make optimal rewriting decisions.

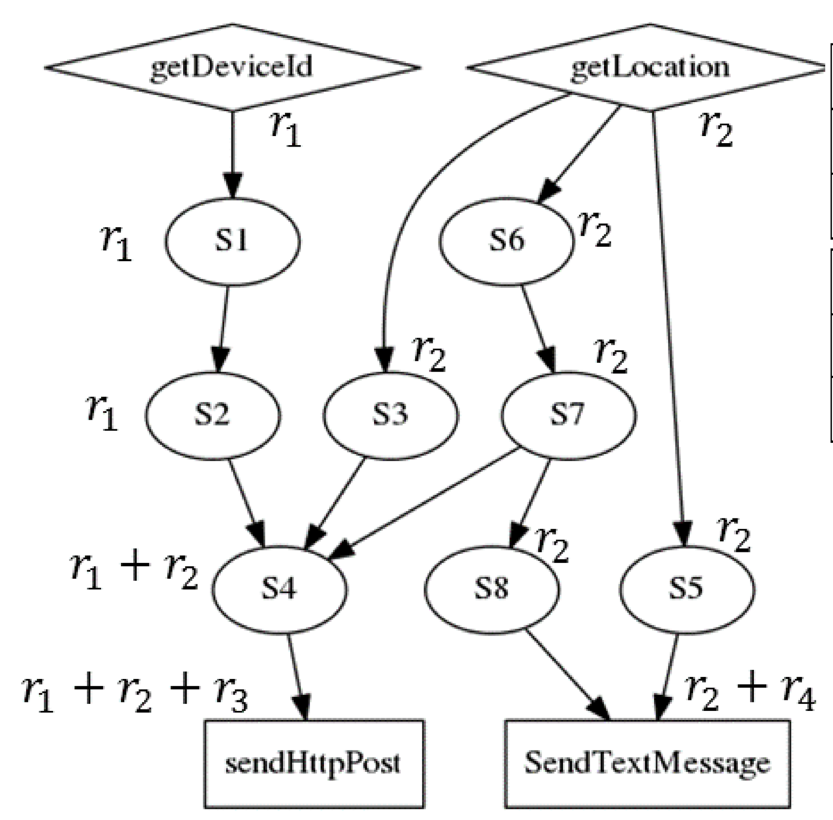
## Conclusion:

- We provide an efficient quantitative analysis to characterize apps' internal behaviors and rank risk scores of nodes.
- We provide a general rewriting framework with our quantitative analysis to enforce apps' security properties.

## Purposes of Quantifying Risks of Flows:

- Quantitative risk analysis of flows enables one to efficiently identify the most critical sets of sinks to cut or modify.
- There are too many sensitive sink nodes as possible rewriting options. A find-all-occurrences approach would be expensive.
- Alternative approaches such as choosing sinks with the minimum in-degrees often give imprecise results.

*We utilize graph algorithms and machine-learning methods to compute and propagate permission-based risk scores over data-flow graphs of apps.*




| Source | Permission | Risk Score |
|---|---|---|
| getDeviceID | READ_PHONE_STATE | $r_1$ |
| getLocation | ACCESS_FINE_LOCATION | $r_2$ |
| Sink | Permission | |
| sendHttpPost | INTERNET | $r_3$ |
| sendTextMessage | SEND_SMS | $r_4$ |

Fig.1 Android information taint flows with permission related risk scores

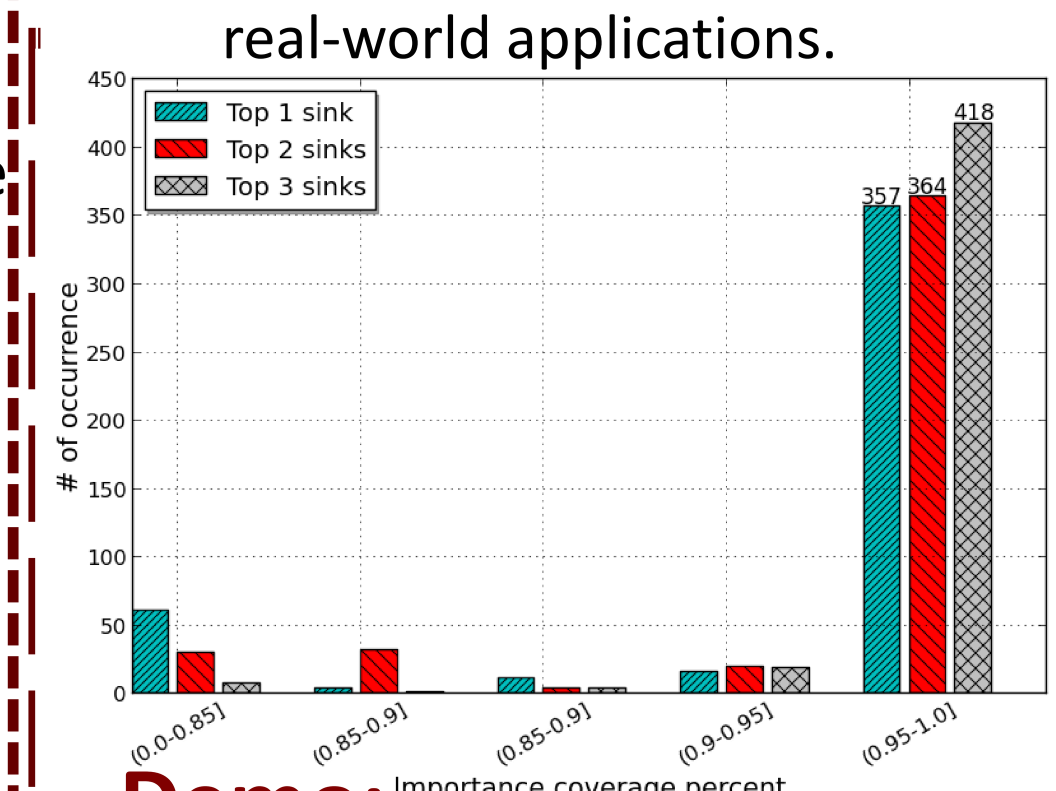
### Risk Scores Associated with:

- inherited permissions (passed down by ancestor nodes).
- inherent permissions (required for completing API invocations).

## Experiments:

### Evaluation Goals:

- To discover properties of the apps with our quantitative information flow analysis.
- To demonstrate the feasibility of our rewriting techniques on real-world applications.



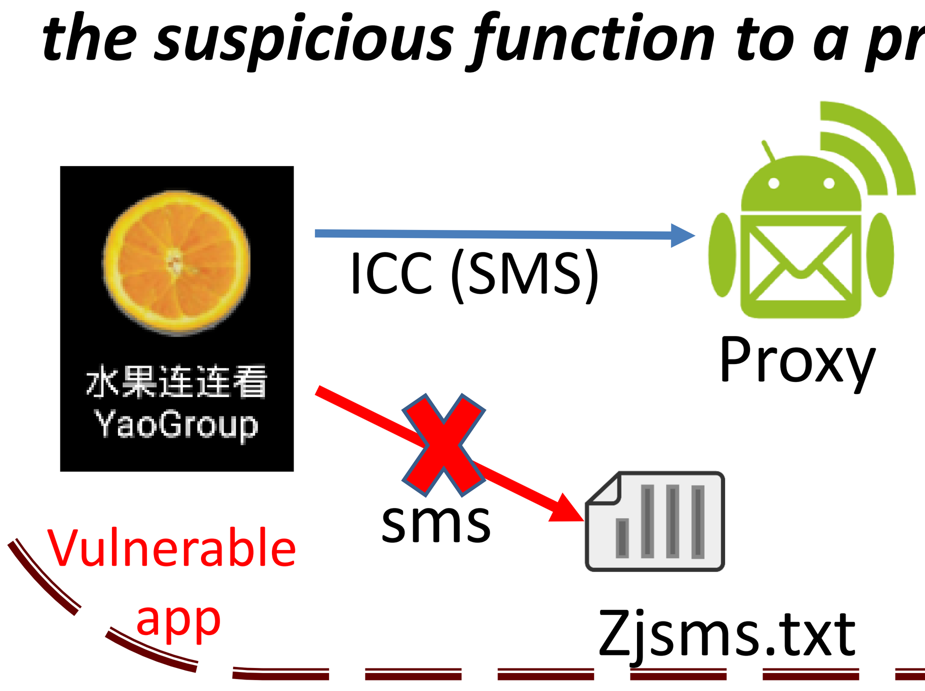

### Risk Inequality:

79% of the apps, the riskiest node has a risk score of 0.95 or higher.

This inequality may be due to the excessive permission requests in malicious code.

## Demo:

***Rewriting rule: remove unsafe permissions and redirect the suspicious function to a proxy***




1. The original app will record SMS content and store it into a local file called zjsms.txt.
2. After the rewriting, the writing function (with privacy info) is redirected to a proxy, the WRITE permission is removed.