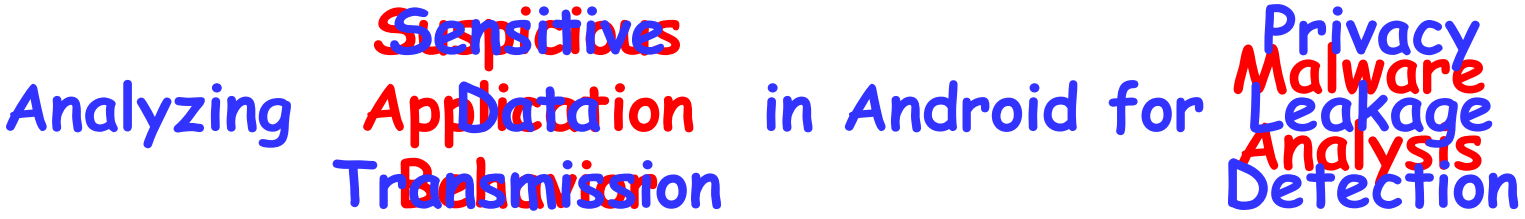
AppIntent: Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection

ZheMin Yang[+], Min Yang[+], Yuan Zhang[+],
Guofei Gu[*], Peng Ning[**], X.Sean Wang[+]
[+]Fudan University
[*]Texas A&M University
[**]North Carolina State University

Beyond privacy leakage



- Recent malware
  - Do suspicious behavior
  - Stealthy(To evade the detection/analysis)

Evasion Techniques

- To Evade the Dynamic Analysis:
  - Anti-virtualization, Anti-debugging, Anti-dumping, Anti-intercepting
  - Packing
  - **Hide deep**(Hide after registration)

- To Evade the Static Analysis:
  - Obfuscation, Packing
  - **Utilize the gap between suspicious and malicious**

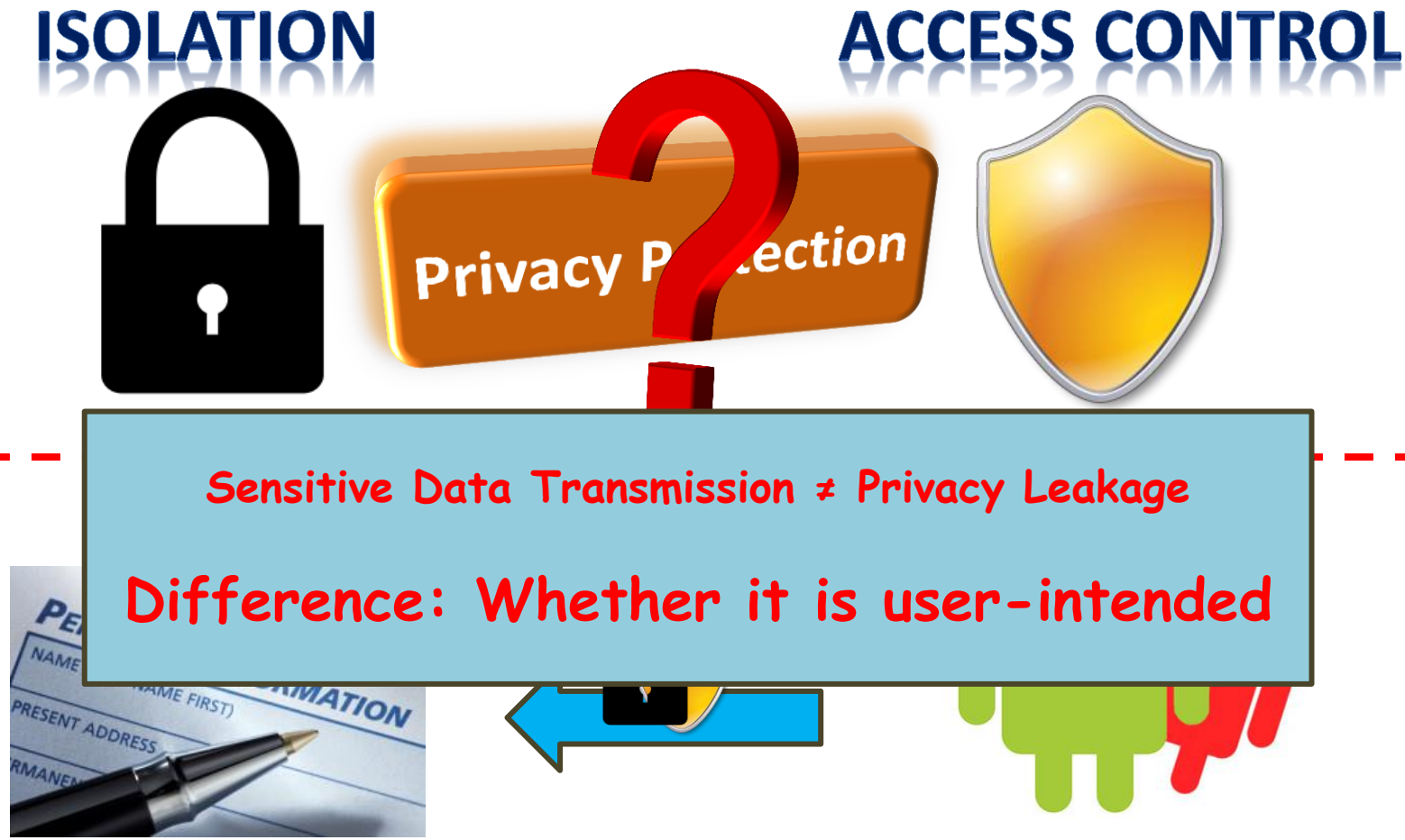
From suspicious to malicious

- Reveal suspicious
  - Android permission system
  - Static analysis techniques
- **Why not analyze malicious behavior instead of suspicious?**

  Because it's hard to automatically judge the intention.

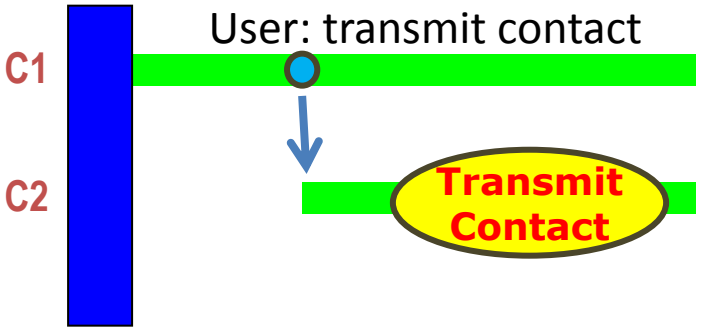
  Focus of AppIntent: visualize the intention.

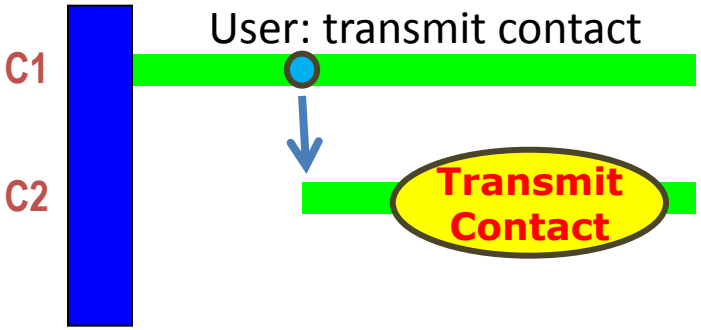
Privacy Protection



User-intended & Unintended Data Transmission

User-intended Data Transmission

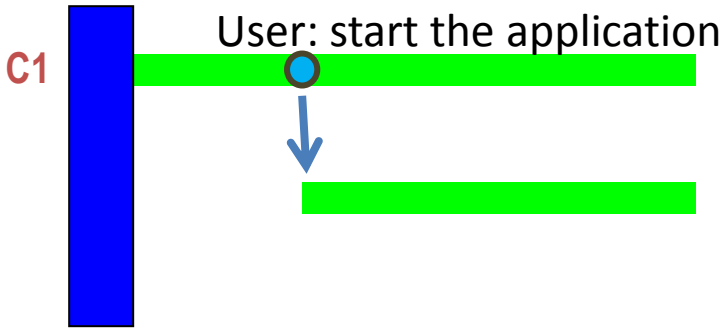


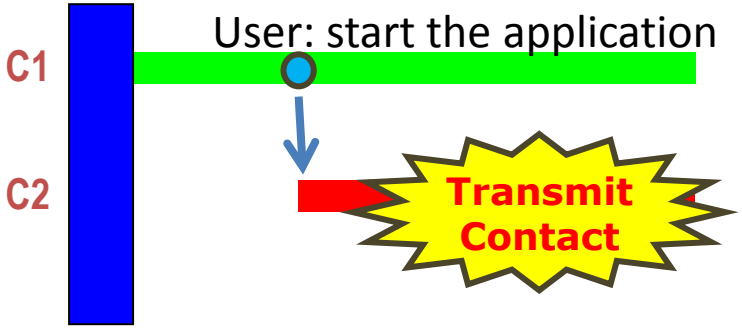


What you expect

What the app functions

Unintended Data Transmission(Privacy Leakage)





Outline

- Why we need AppIntent
- Demos
- Using AppIntent to analyze sensitive data transmission
- Evaluation Result

State-of-the-art

- Static Analysis(*PLDI'09*, *NDSS'11*, etc.)
  - **No user intention or context information**
  - Cannot separate user-intended operations from unintended ones
- Dynamic Taint Tracking(*MICRO'04*, TaintDroid *OSDI'10*, etc.)
  - **Irrelevant events**

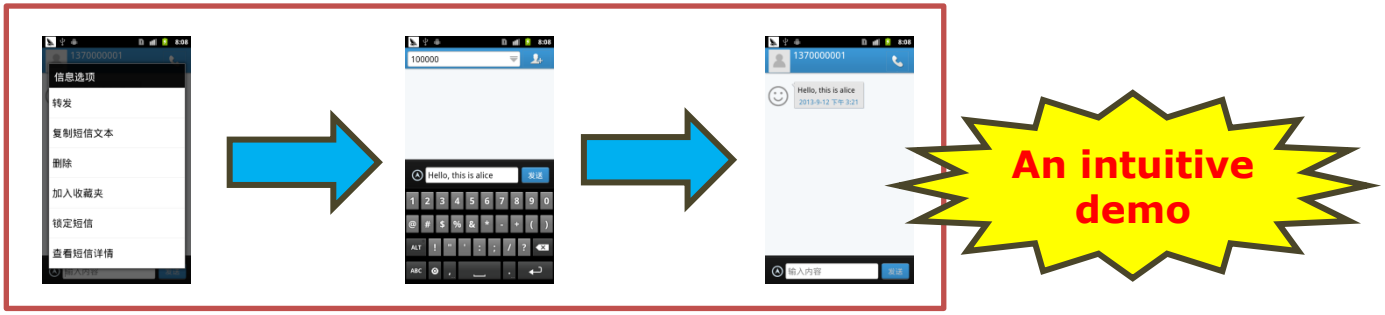
State-of-the-art

- BLADE(*CCS'10*) / Vision(*MCS'11*)
  - Works only if app contains End-user license agreements(EULA) or explicit notification
- Pegasus(*NDSS'13*)
  - Only work on permissions
  - Need define application-specific properties for evaluated apps

AppIntent

- Help analyst determine:
  - whether the transmission is user intended

- Present context information in which:
  - Sensitive data transmission occurs

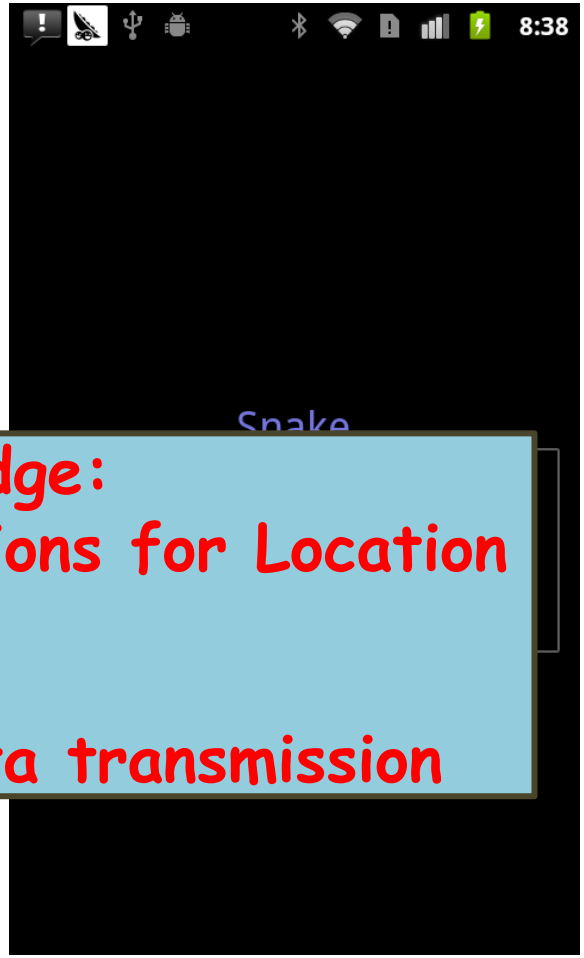


Outline

- Why we need AppIntent
- Demos
- Using AppIntent to analyze sensitive data transmission
- Evaluation Result

Demos

- TapSnake
  - An Android game
  - Output of AppIntent:
    - Automatic driven execution that:
      - Start Application

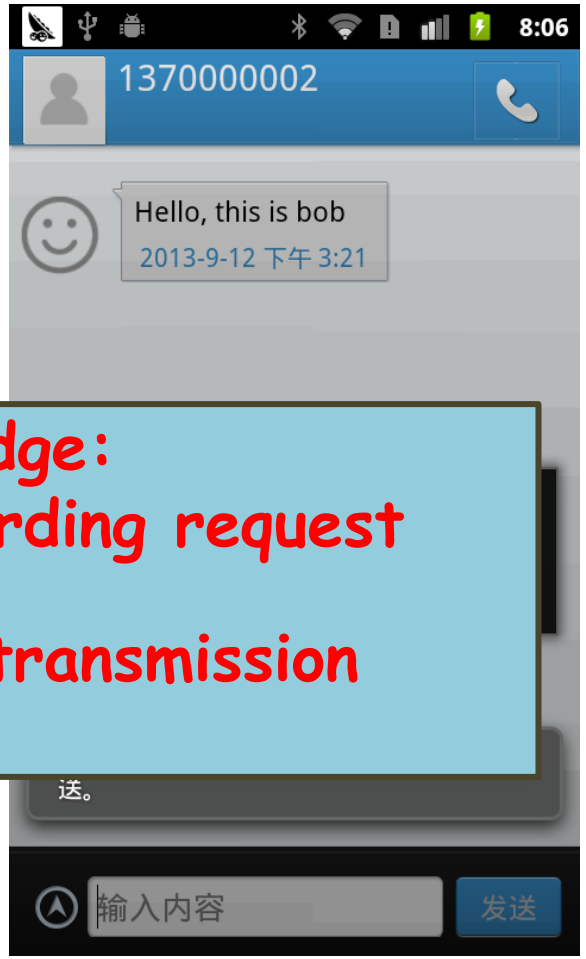


Analyst's Judge:
No user triggered operations for Location send

NOT user-intended data transmission

Demos

- Anzhuoduanxin
  - An SMS management app
  - Output of AppIntent:
    - Automatic driven execution that:



Analyst's Judge:
This is an SMS forwarding request

user-intended data transmission
Video

Click Button "Send"

Message sent

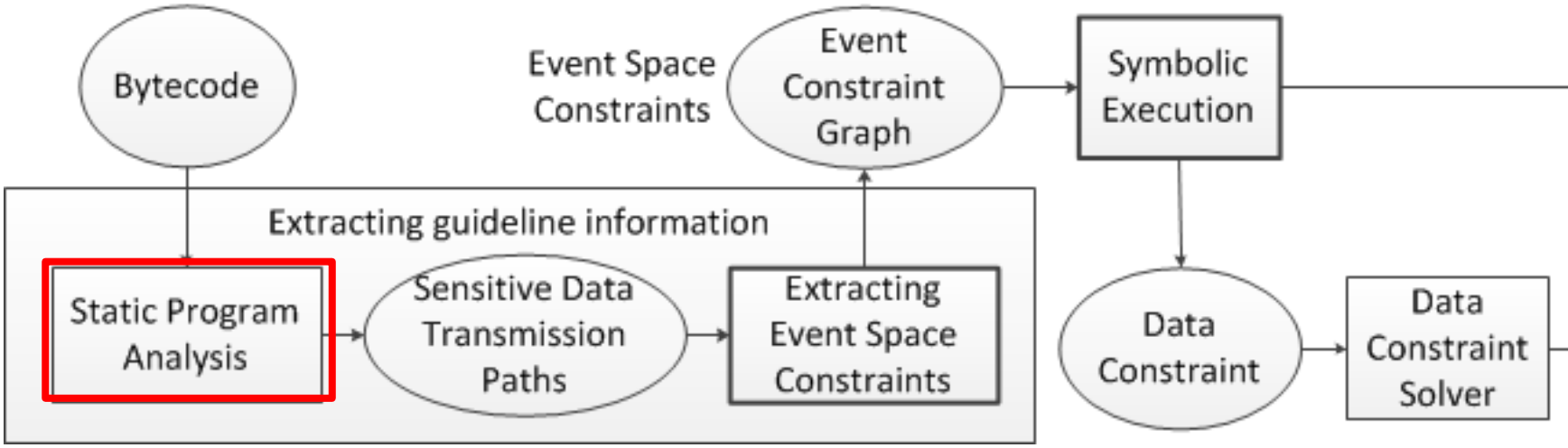
Outline

- Why we need AppIntent
- Demos
- Using AppIntent to analyze sensitive data transmission
- Evaluation Result

Overall architecture

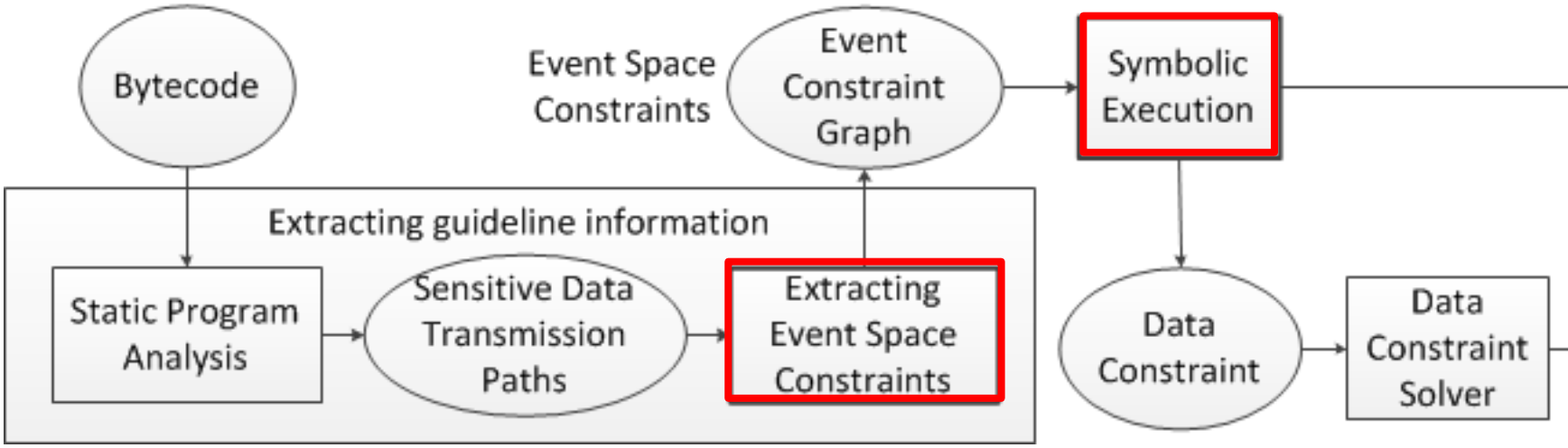
- Goal of AppIntent
  - Generate and present context information
- Context information ← app inputs
  - ***Data inputs*** which contain text inputs from outside and
  - ***Event inputs*** from user interactions by GUI interface and from system through IPC
- Precise context information
  - AppIntent focuses on ***critical app inputs*** in which:
  - Irrelevant events are not included

Overall architecture



- Static Taint Analysis
  - preprocess and extract all possible data transmission paths
  - Existing techniques

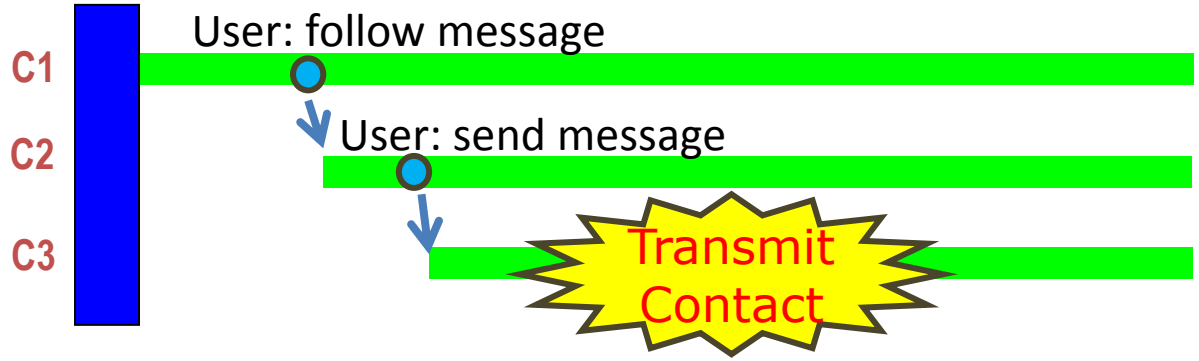
Overall architecture



- Generate context information
  - ***Event-Space Constraint Guided Symbolic Execution*** *(introduce below)*
- Present context information
  - The controlled execution (The demos)

AppIntent

Step1: Generate context information



- Given a possible sensitive data transmission
- Extract critical inputs through ***Event-Space Constraint Guided Symbolic Execution***

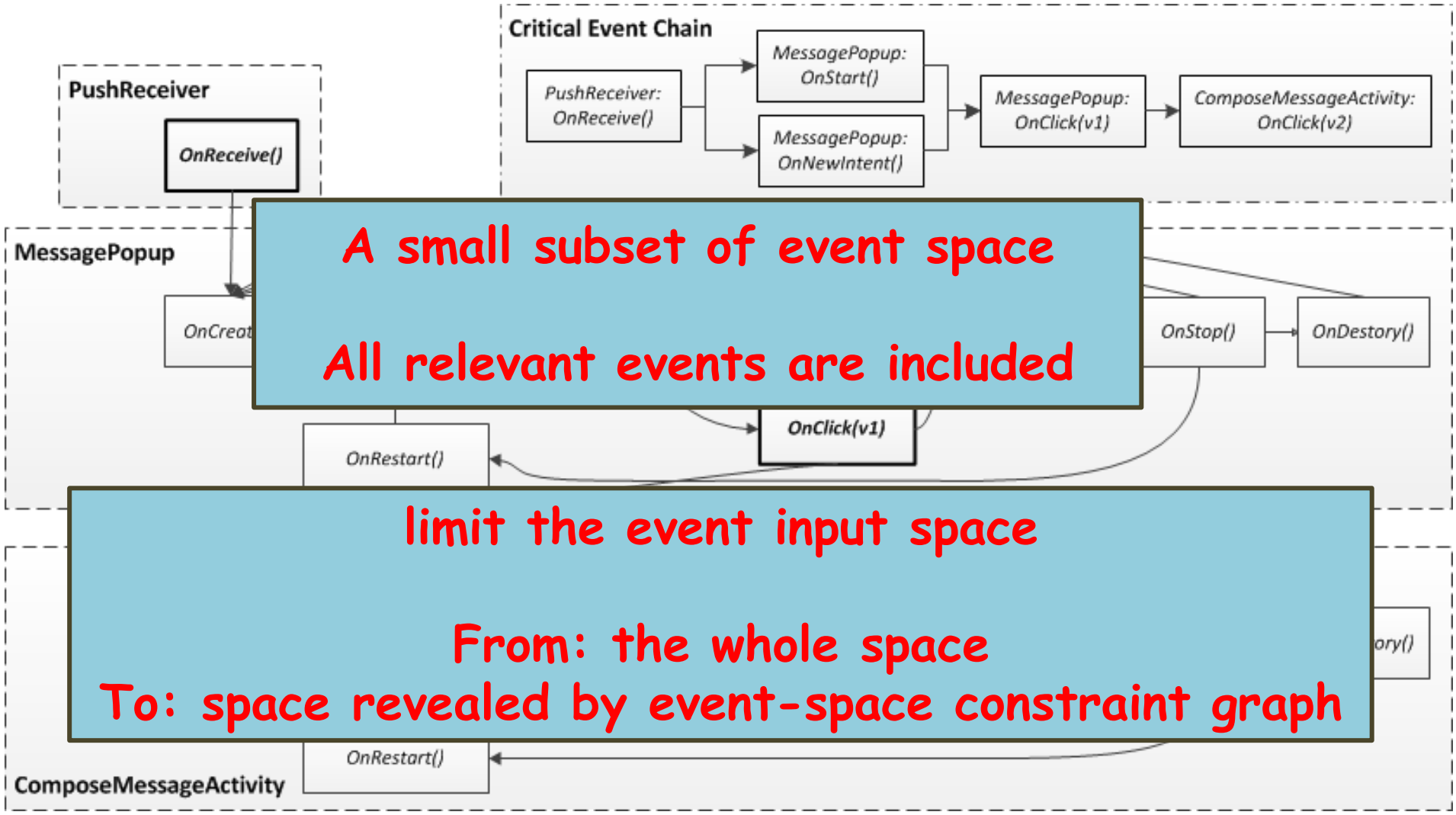
Symbolic Execution *can* and *cannot*

- Symbolic execution is a traversal process.
  - Performance restricted by the size of search space.
- Symbolic Execution can:
  - Produce data inputs for certain app behavior.

We need to limit the event input space

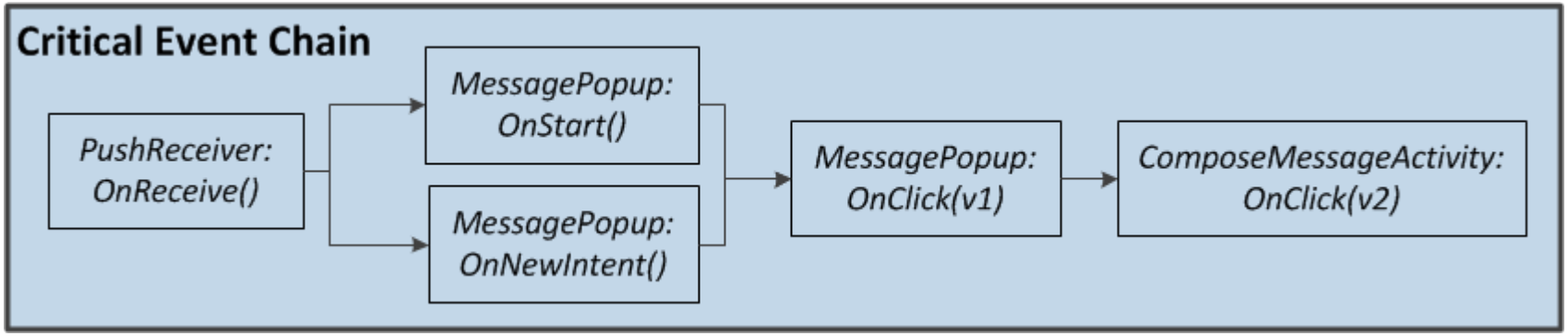
- State-of-the-art Symbolic Execution cannot:
  - Efficiently traverse the ***event input*** space.
    - Explosion of event space
  - That's what AppIntent solved.

Event-space constraint graph

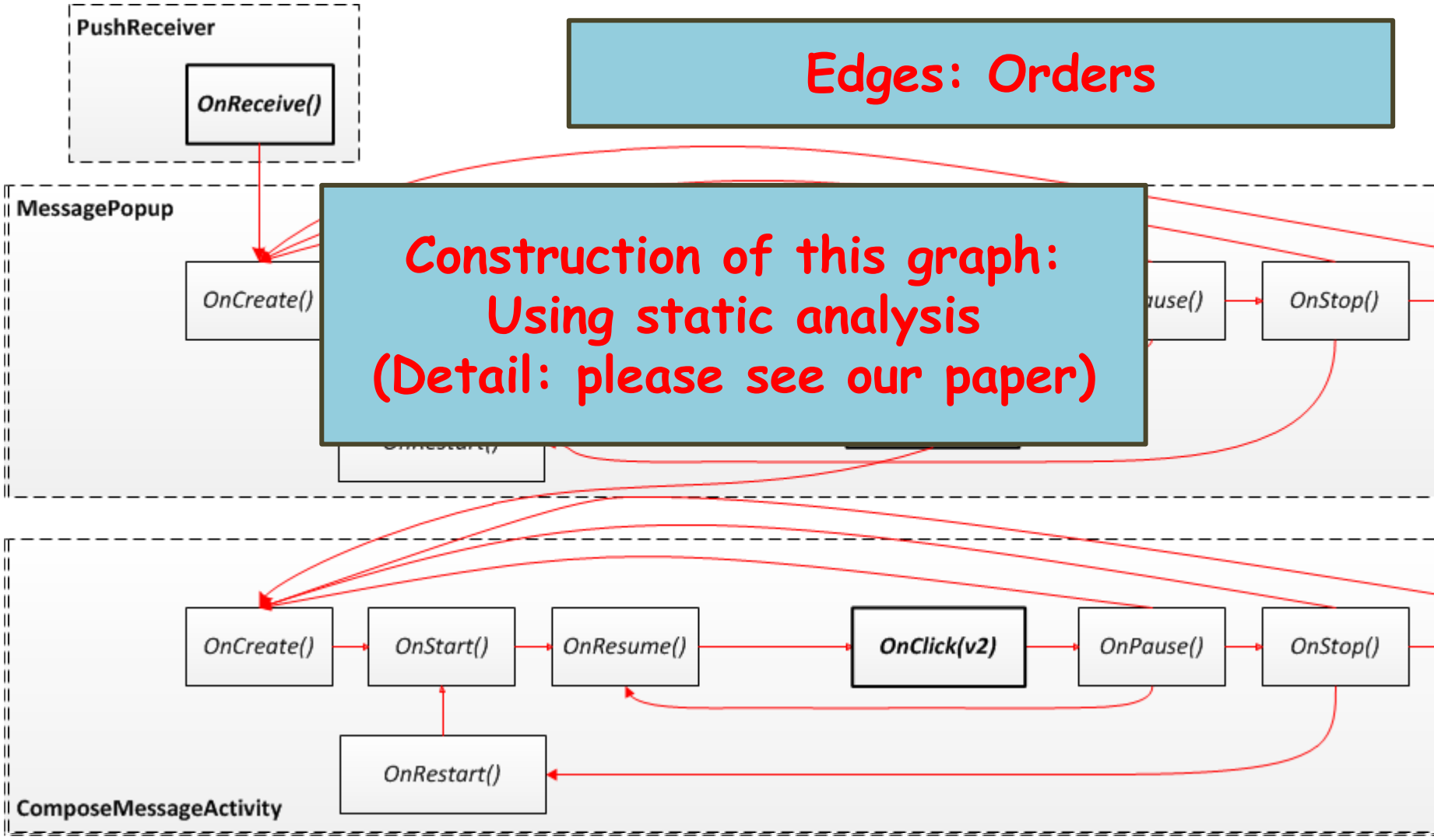


Event-space constraint graph

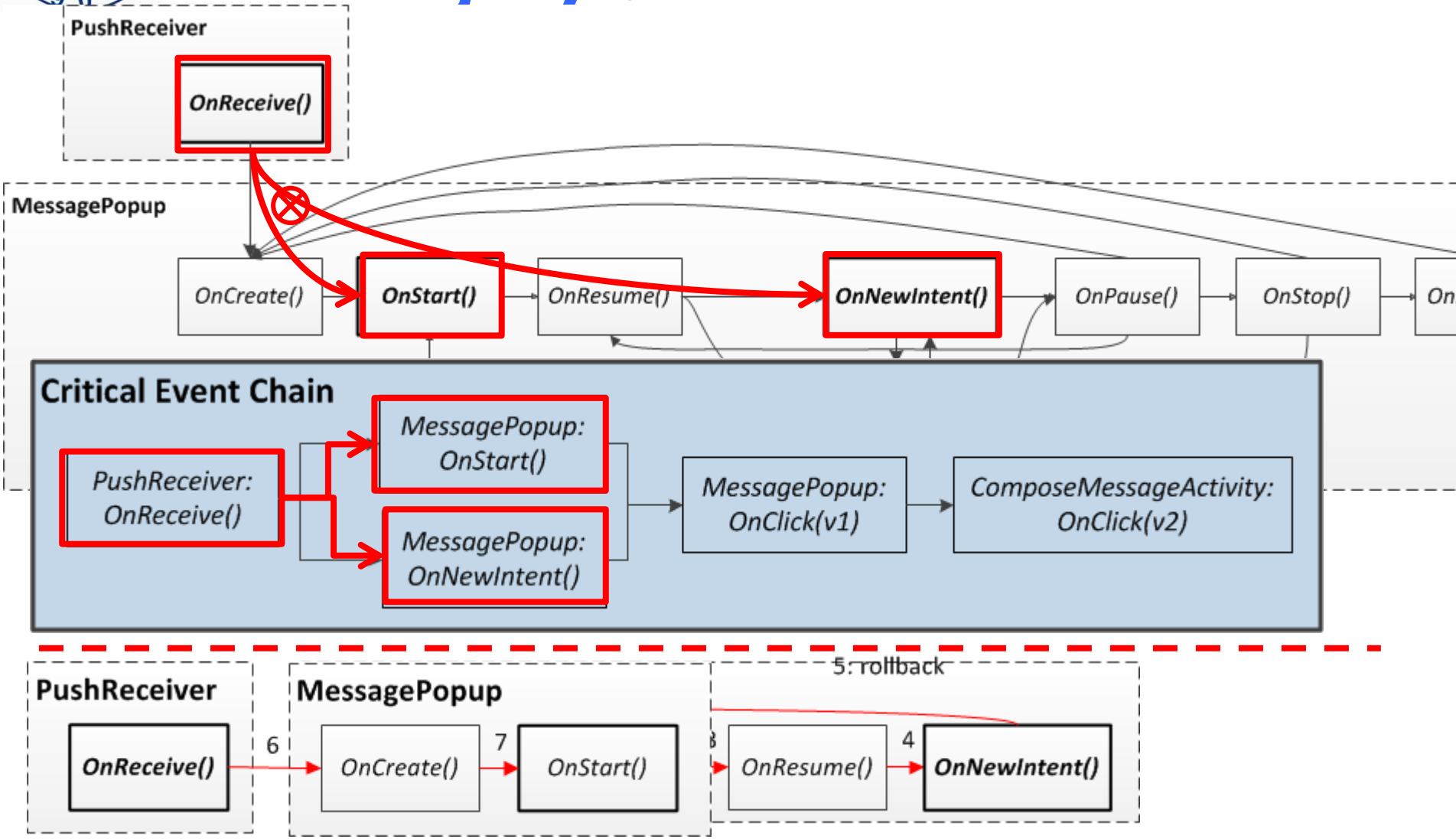
Critical Event: contains at least one instruction of the sensitive data transmission path



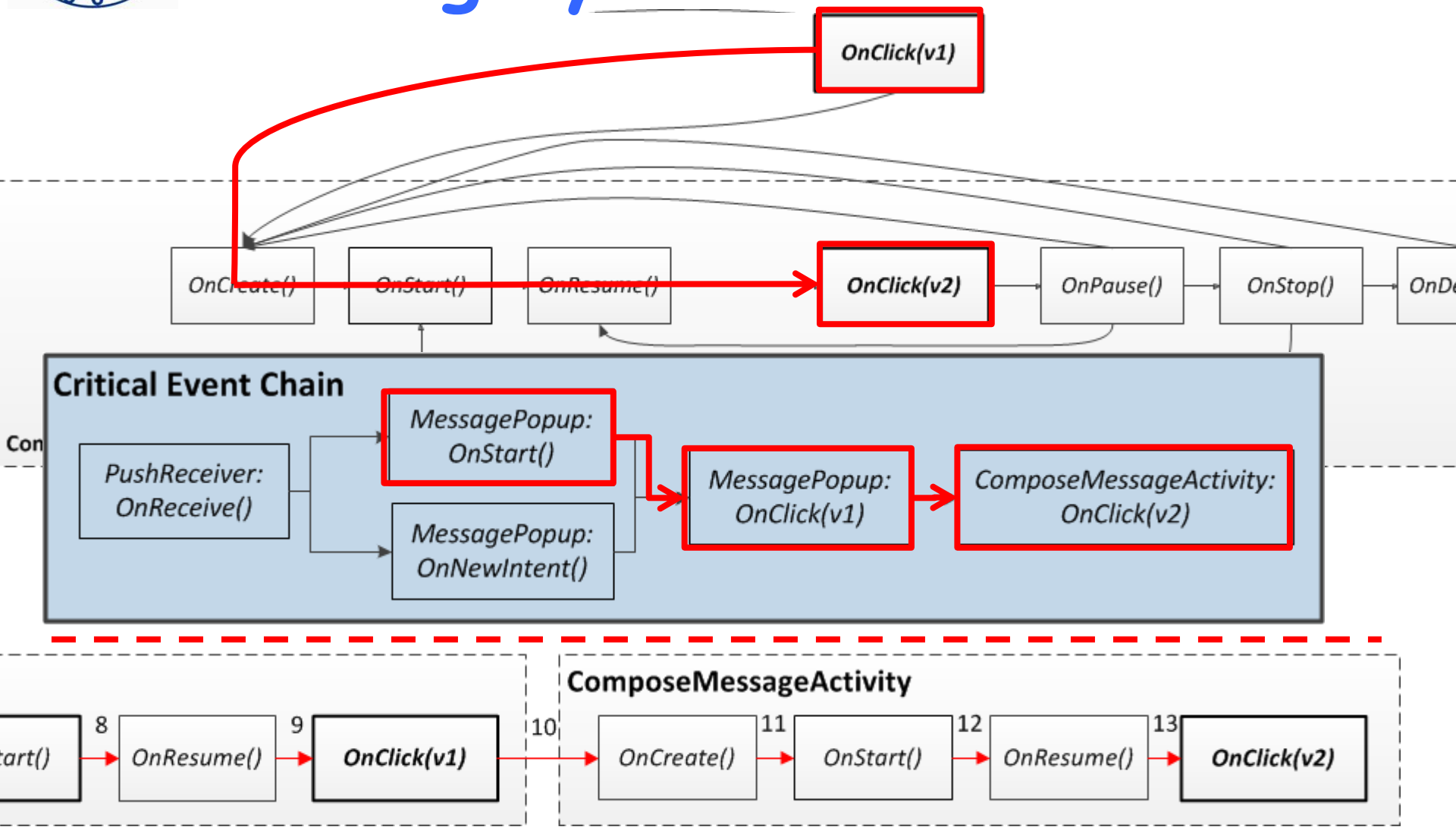
Event-space constraint graph



Guiding symbolic execution

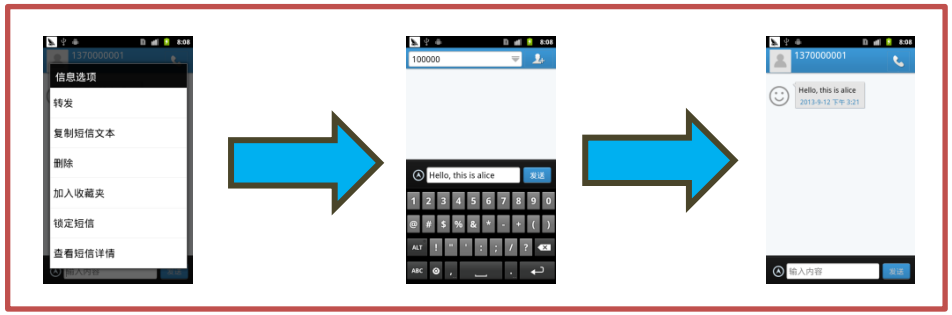


Guiding symbolic execution



AppIntent

Step2: Generate a controlled execution



- Given the app inputs
- Generate an execution which
  - Automatically trigger user interaction
  - Highlight activated views/sensitive data read and transmission
  - (details see our paper)

Outline

- Why we need AppIntent
- Demos
- Using AppIntent to analyze sensitive data transmission
- Evaluation Result

Evaluation

- Evaluation Platform
  - Implement on:
    - Soot for static analysis
    - JavaPathfinder for symbolic execution
    - Android InstrumentationTestRunner for controlled execution
  - Evaluate on:
    - Intel Xeon machine with 2 8-core 2.0GHz CPU
    - 32 GB memory
    - Debian Linux kernel version 2.6.32
    - Android version 2.3

Effectiveness of
Event-space constraint guided symbolic execution

Case	Origin (10 events) (hours)		Origin (20 events) (hours)		AppIntent (hours)	
Maps	5.43	✓	>120	✓	0.40	✓
youlu	0.97	✗	>120	✓	0.13	✓
Weixin	21.56	✗	>120	✗	1.33	✓

Effectiveness of AppIntent

Better coverage & precision than TaintDroid

Source	Unintended/ Intended Data Transmission	TaintDroid
Device ID	198/0	101
Phone Info	50/0	0
Location	46/4	11
Contacts	1/10	0
SMS	16/3	0
Total	219/17	125

Effectiveness of AppIntent

- Apps from Google Play

Source	Unintended/ Intended Data Transmission	TaintDroid
Device ID	24/0	37
Phone Info	0/0	19
Location	0/13	5
Contacts	1/9	3
SMS	1/7	0
Total	26/29	40

Usability of AppIntent

- Cases
  - 100 random cases reported
- Users
  - 3 Android experts
- Results
  - Decision made in less than one minute after the driven execution finishes
  - Result of 98 cases are the same as our judgment
  - 2 remaining cases are all about IMEI

Conclusion

- User intended/unintended sensitive data transmission
- AppIntent system to reveal user intension behind data transmission
- Event-space constraint guided symbolic execution
  - Event-space explosion problem
  - Guiding symbolic execution with event-space constraint graph
- Effectiveness and Usage
  - Effectively solve the search-space explosion problem
  - Effectively distinguish sensitive data transmissions
  - Easy to use

Thanks

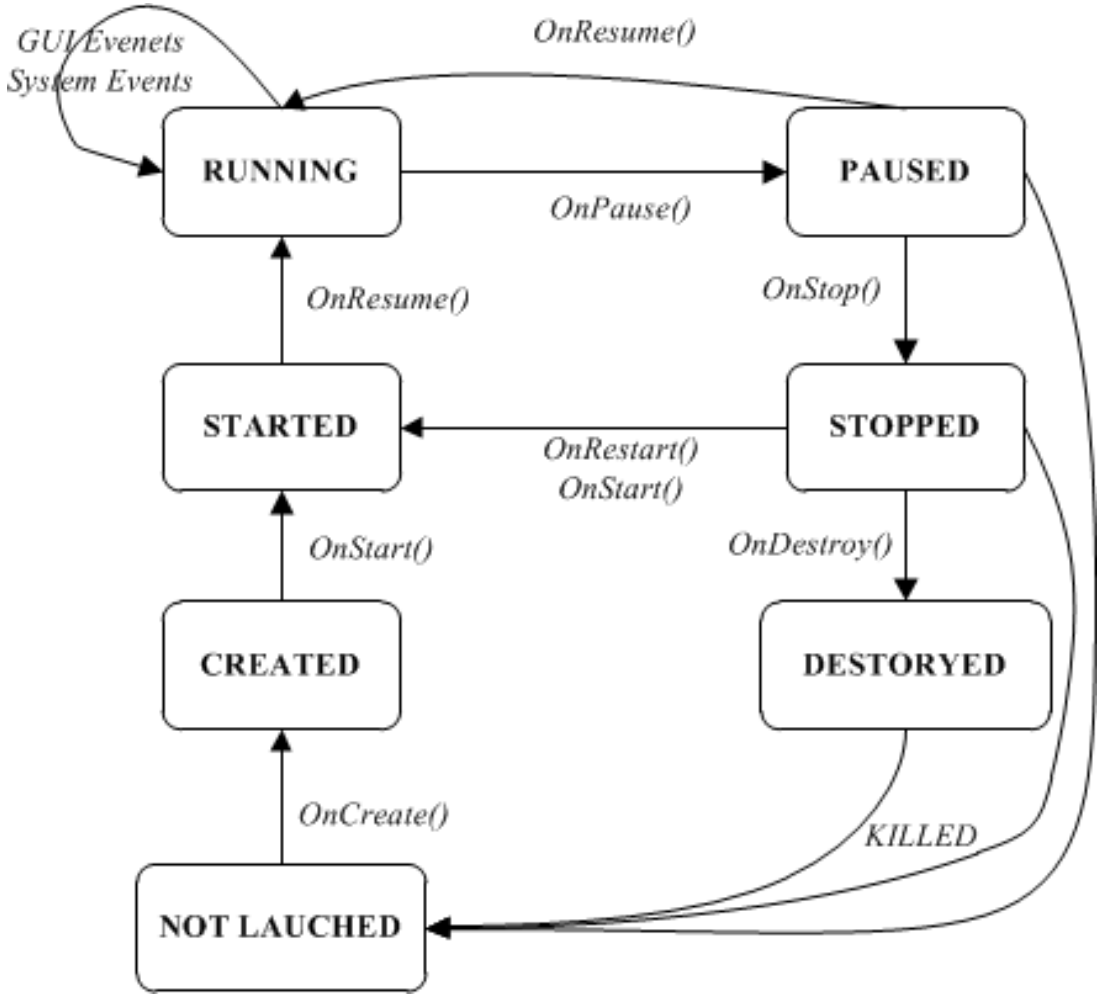
AppIntent

Analyzing Sensitive Data Transmission in Android for Privacy Leakage Detection

Questions?

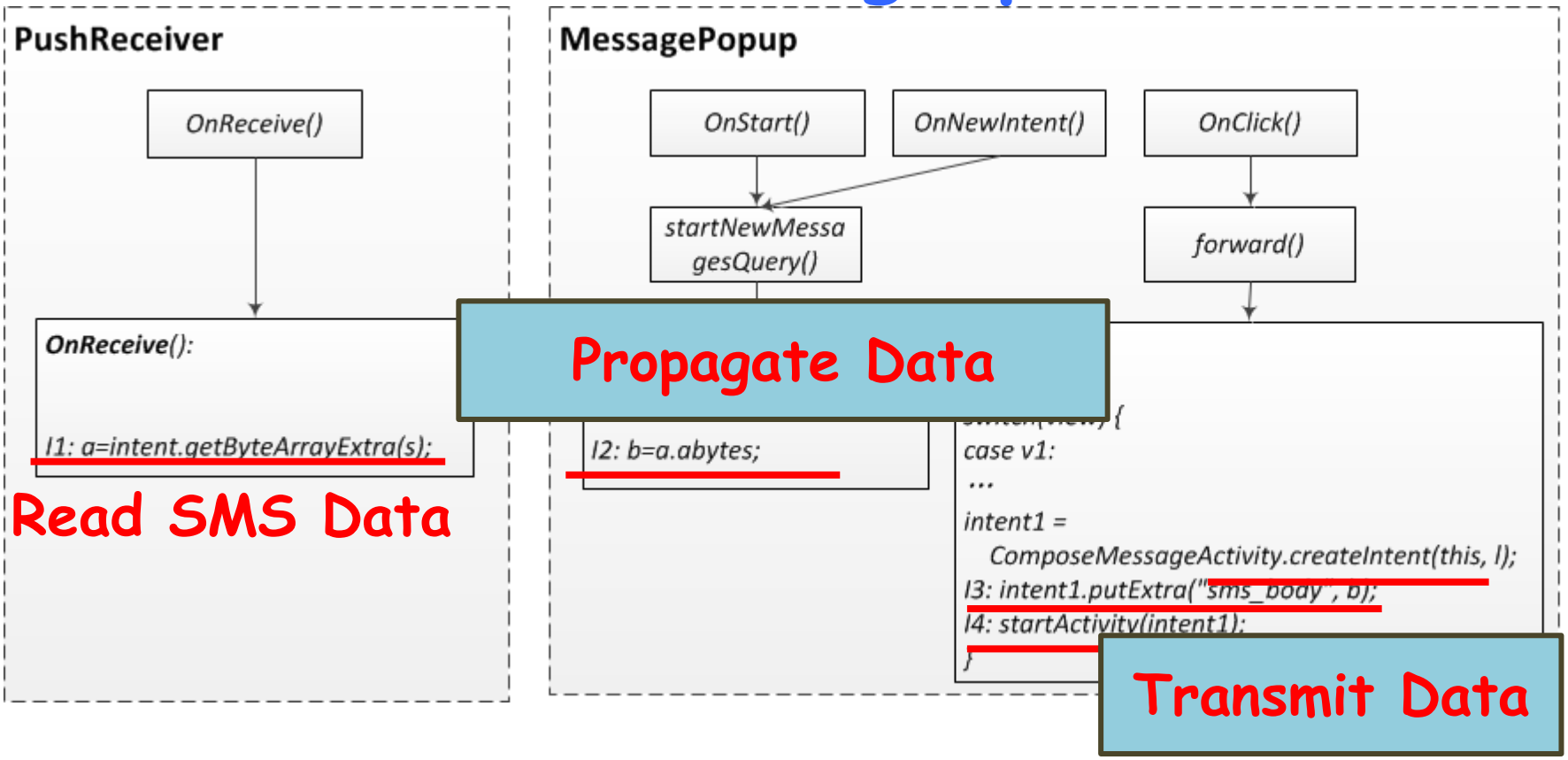
Backup slides

Background: Android execution model



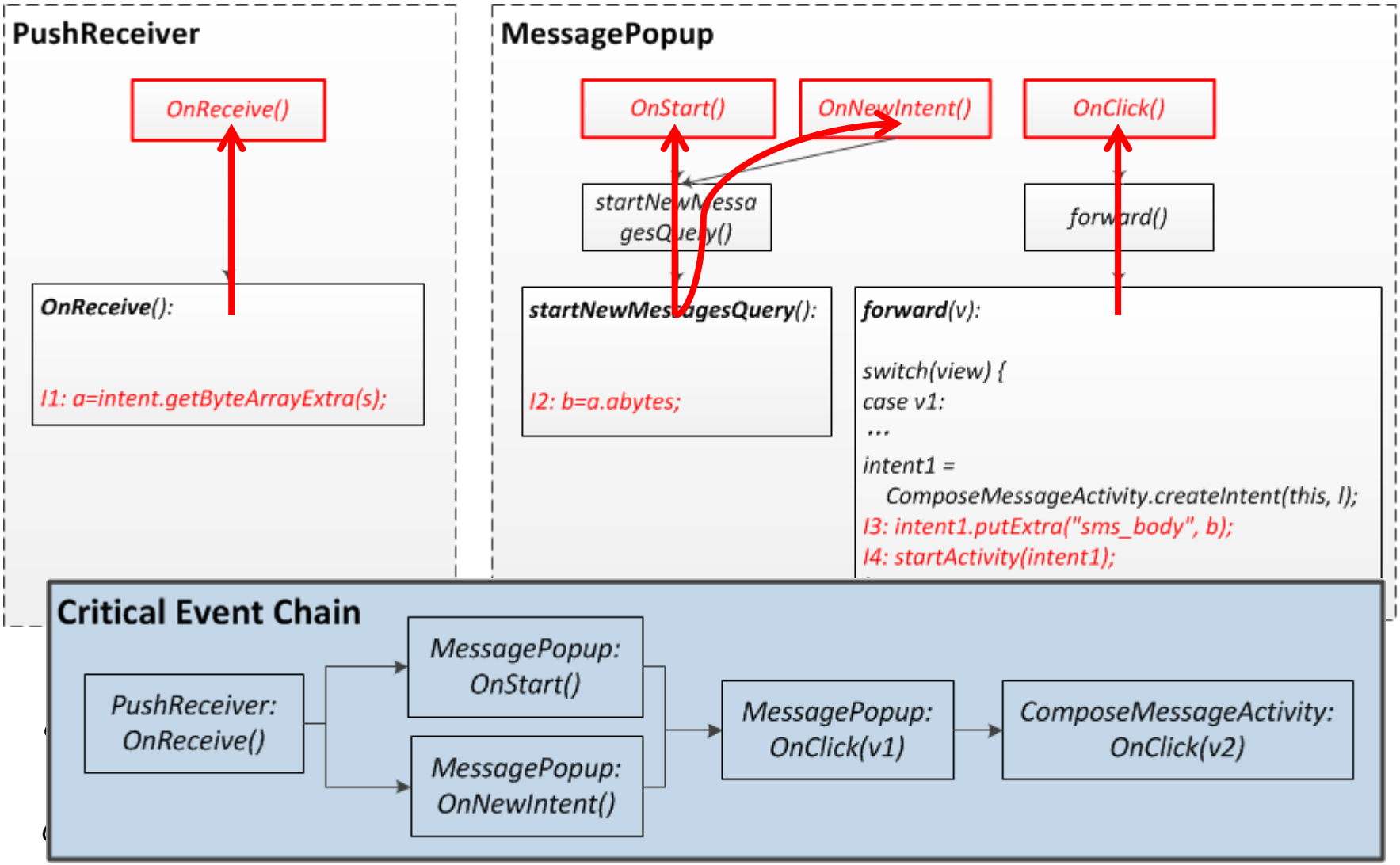
- Callbacks of lifecycle states
  - onStart, onPause…
- GUI/system events
  - Click button, location change, etc.

Construct event-space constraint graph



Sensitive data transmission path: {i1,i2,i3,i4,i5,i6}

Construct event-space constraint graph



Construct event-space constraint graph