Conclusion

- Definitions of Android App Vulnerabilities
- Vulnerabilities category
- Program analysis methods used for assesments
- Efforts made to turn research prototype into industrial product

Credits/Refs

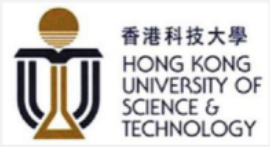
- FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps , PLDI'14
- A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps, CCS'14
- Precise Interprocedural Dataflow Analysis via Graph Reachability, POPL'95

Theory and Practice of Program Analysis in Mobile App Vulnerability Assesments

Flanker @ KEEN TEAM

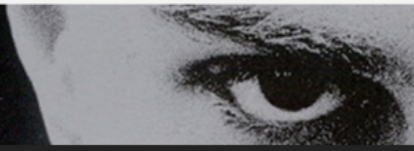
About us

About me



KEEN TEAM

- Program Analysis (Java)
- Reported multiple critical vulnerabilities in famous apps & *SRC

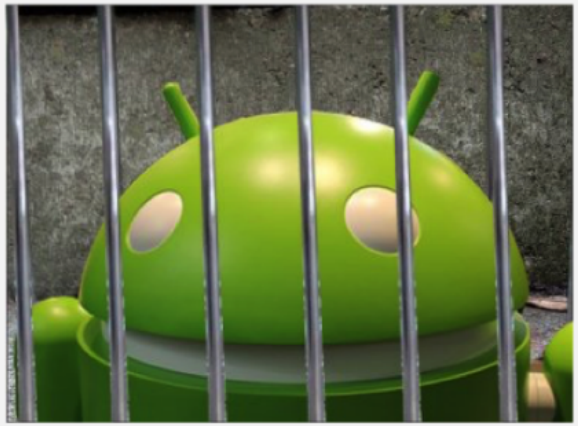


Abstract

- Give definition on Android Application Vulnerabilities
- Examples of typical vulnerabilities
- Introduce corresponding program analysis methods

Android Security Revisited

- Process isolation utilizing Linux UID/GID
- Resource & Data ACL based on Permission
- Signature & Certificate trust chain



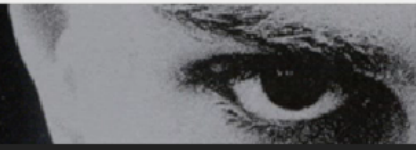
Application Vulnerability Definitions

- Attack purpose
  - Acquire permission/capability
  - Access/modify other app's data/state
- Attack through...
  - Remote (URL, webpage, etc.)
  - Local app
  - MITM
  - ADB
- Escape but don't break the sandbox

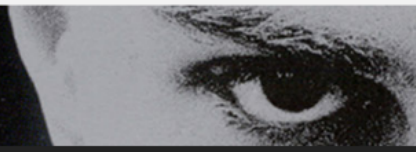
DEMO TIME!

No more sleeping!

KEEN TEAM

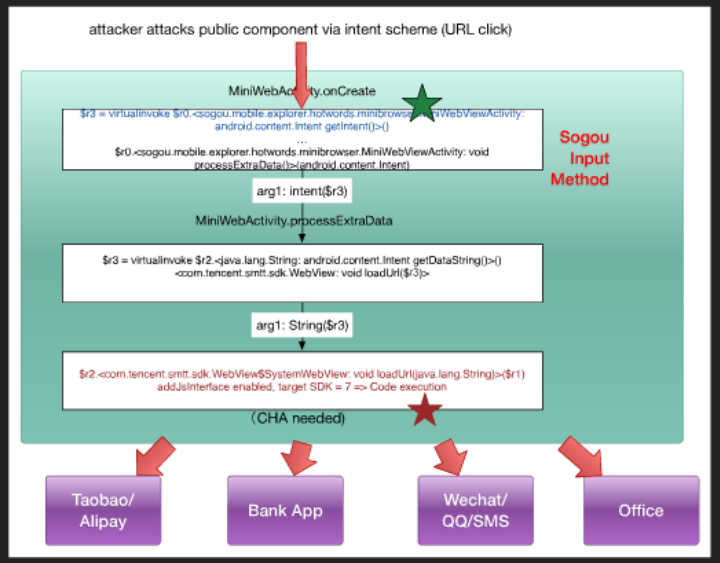


We'll talk about... universal vulns



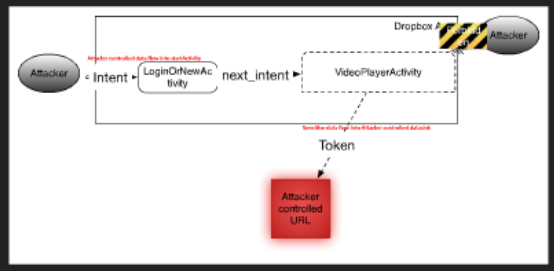
DataFlow Vulnerabilities

Attacker controlled data into sensitive API



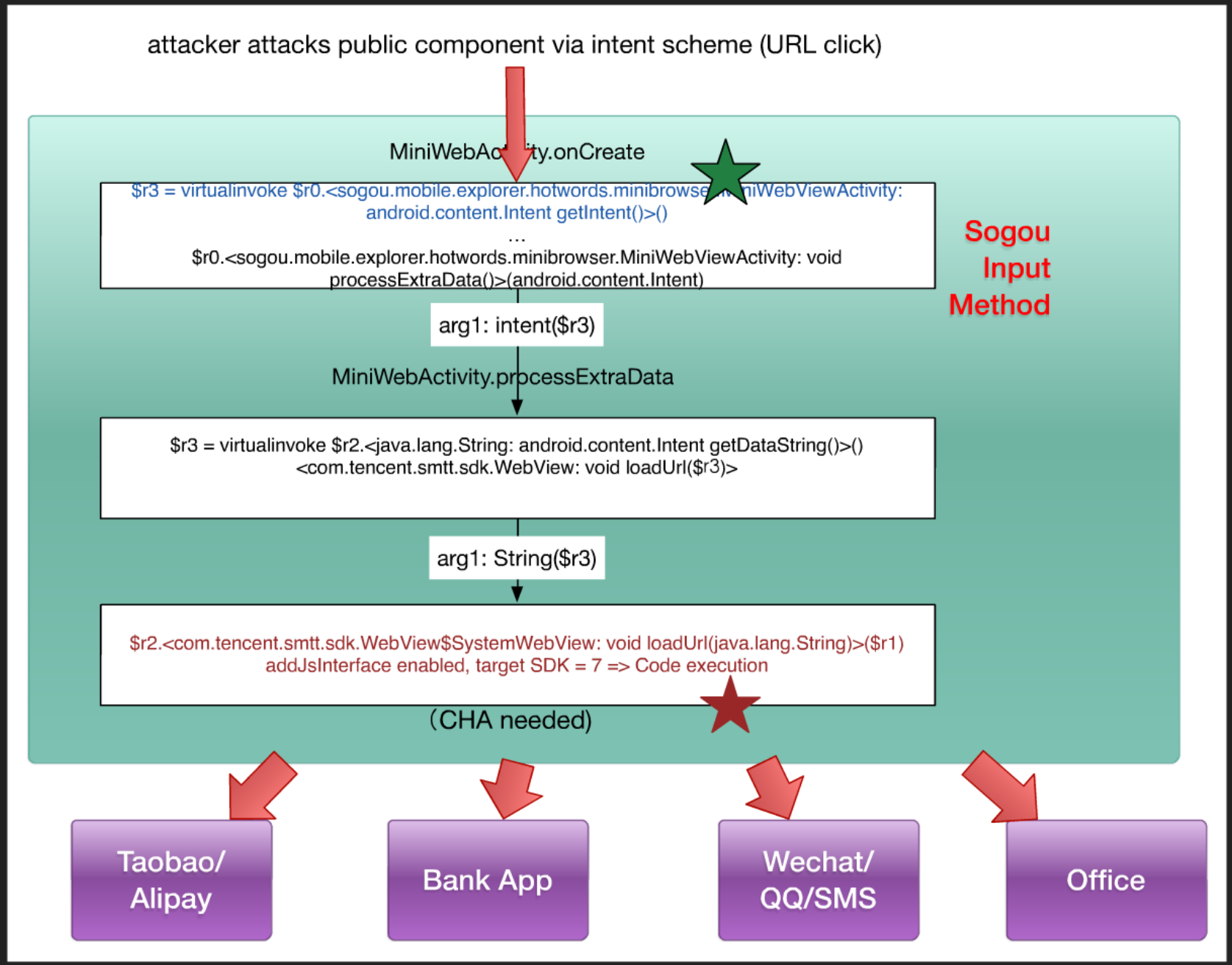
Sensitive data flowout to attacker-accessiable/controllable sink

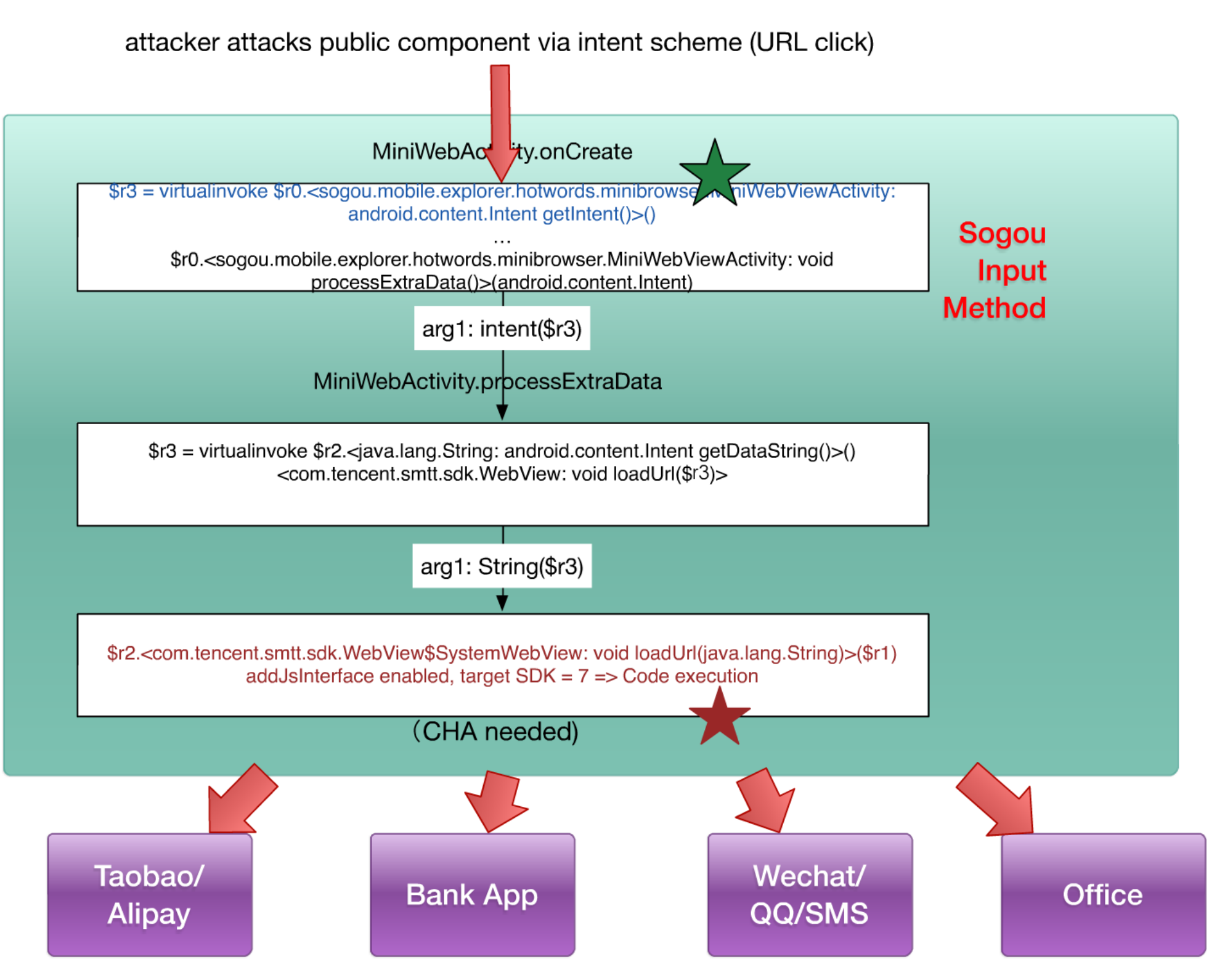
- Cleartext password flow (to insecure storage)
- Sensitive data flow (to controllable/insecure channel/storage)



Unauthorized Origin Crossing on Mobile Platforms: Threats and Mitigation - CCS, MSR

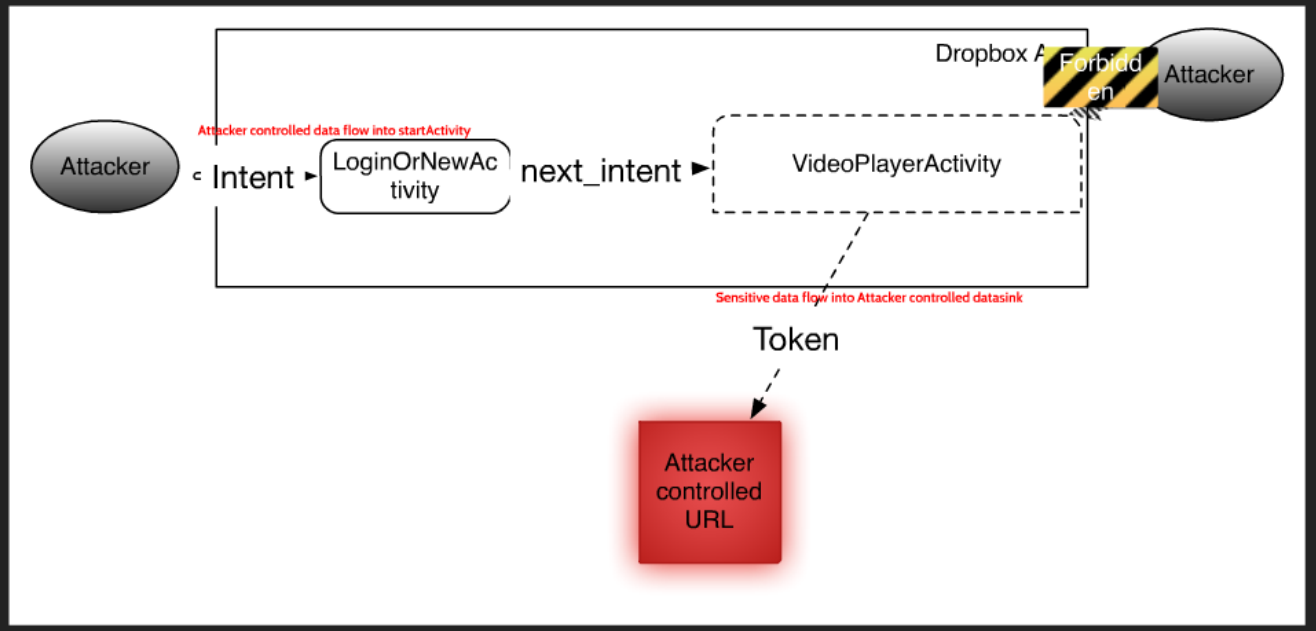
Attacker controlled data into sensitive API




attacker attacks public component via intent scheme (URL click)
MiniWebActivity.onCreate
$r3 = virtualinvoke $r0.<sogou.mobile.explorer.hotwords.minibrowser.MiniWebViewActivity: android.content.Intent getIntent()>()
...
$r0.<sogou.mobile.explorer.hotwords.minibrowser.MiniWebViewActivity: void processExtraData()>(android.content.Intent)
Sogou Input Method
arg1: intent($r3)
MiniWebActivity.processExtraData
$r3 = virtualinvoke $r2.<java.lang.String: android.content.Intent getDataString()>()
<com.tencent.smtt.sdk.WebView: void loadUrl($r3)>
arg1: String($r3)
$r2.<com.tencent.smtt.sdk.WebView$SystemWebView: void loadUrl(java.lang.String)>($r1)
addJsInterface enabled, target SDK = 7 => Code execution
(CHA needed)
Taobao/ Alipay
Bank App
Wechat/ QQ/SMS
Office

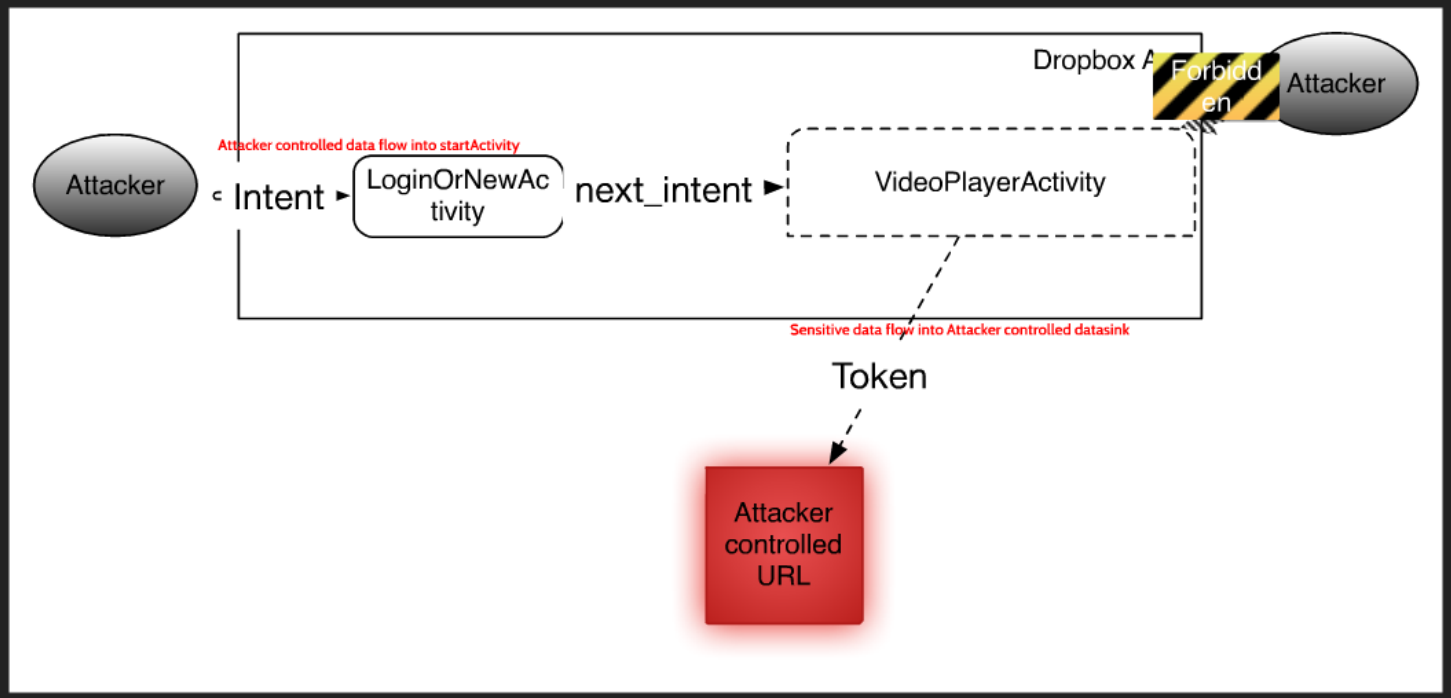
Sensitive data flowout to attacker-accessiable/controllable sink

- Cleartext password flow (to insecure storage)
- Sensitive data flow (to controllable/insecure channel/storage)

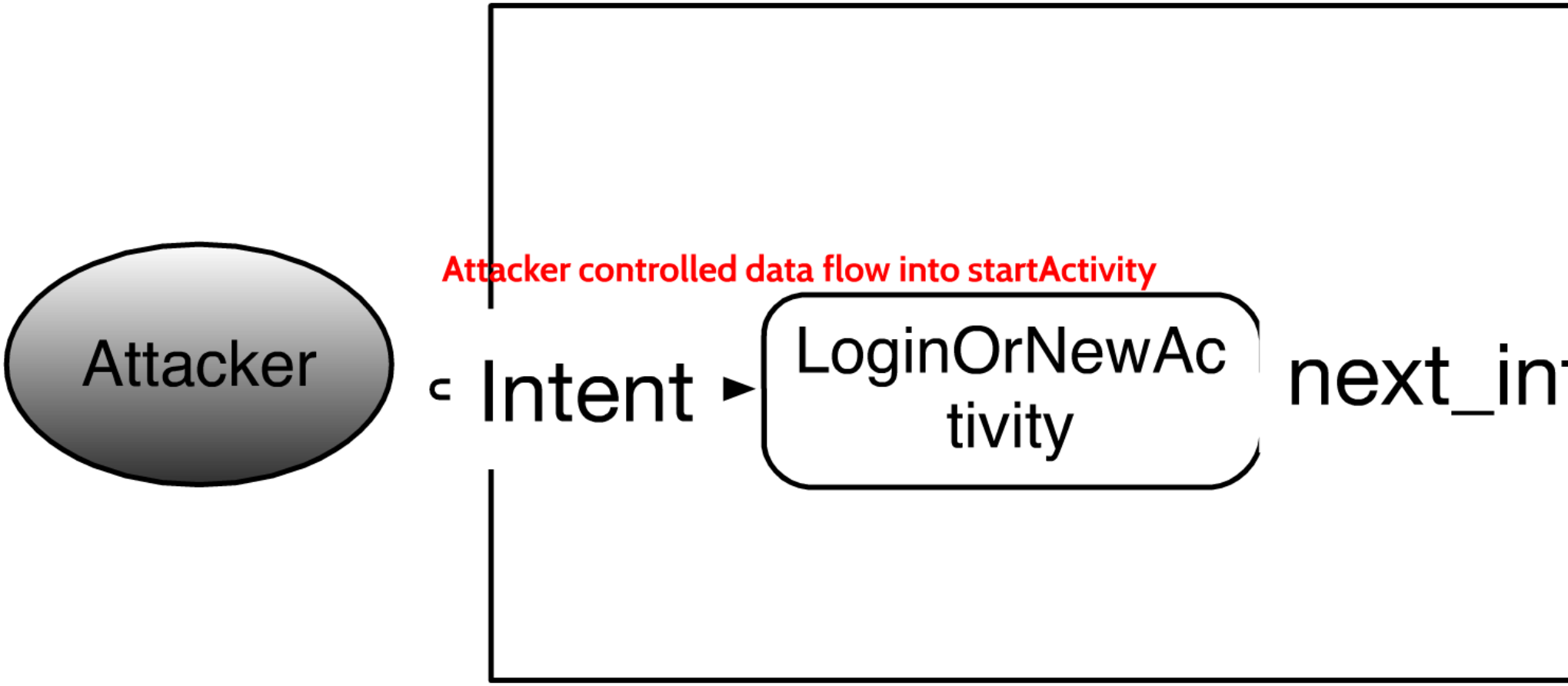


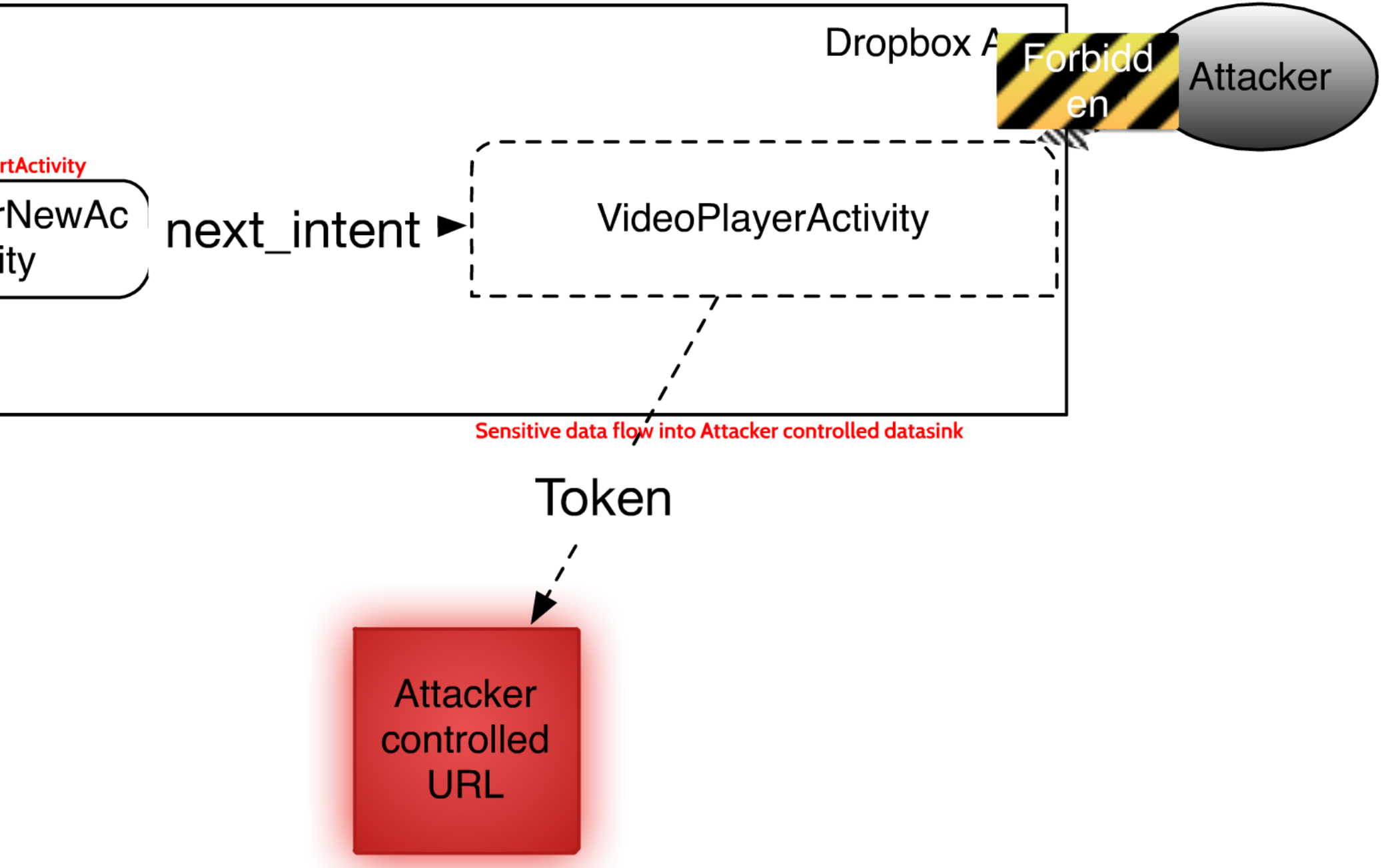
Unauthorized Origin Crossing on Mobile Platforms: Threats and Mitigation - CCS, MSR

- Cleartext password flow (to insecure storage)
- Sensitive data flow (to controllable/insecure channel/storage)



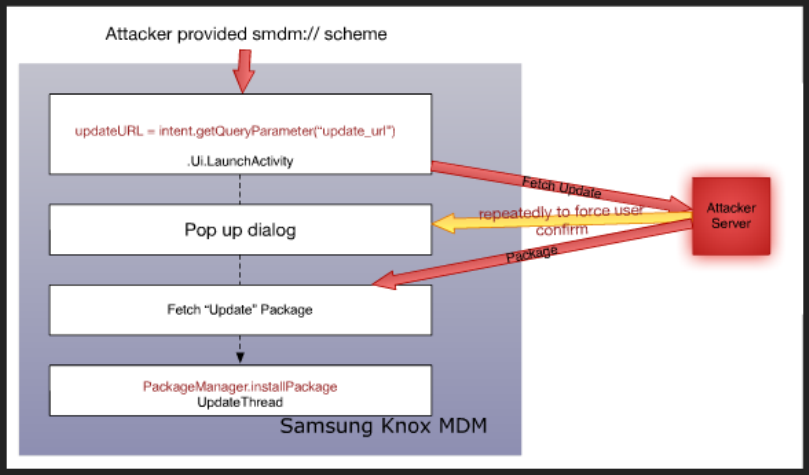
Unauthorized Origin Crossing on Mobile Platforms: Threats and Mitigation - CCS, MSR


Attacker controlled data flow into startActivity
Attacker
Intent
LoginOrNewAc
tivity
next_int


Dropbox A
Forbidd
en
Attacker
rtActivity
rNewAc
ity
next_intent
VideoPlayerActivity
Sensitive data flow into Attacker controlled datasink
Token
Attacker
controlled
URL

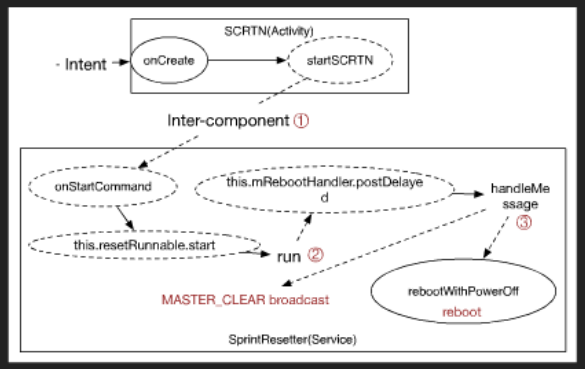
Capability Leak

Samsung Knox Remote code execution



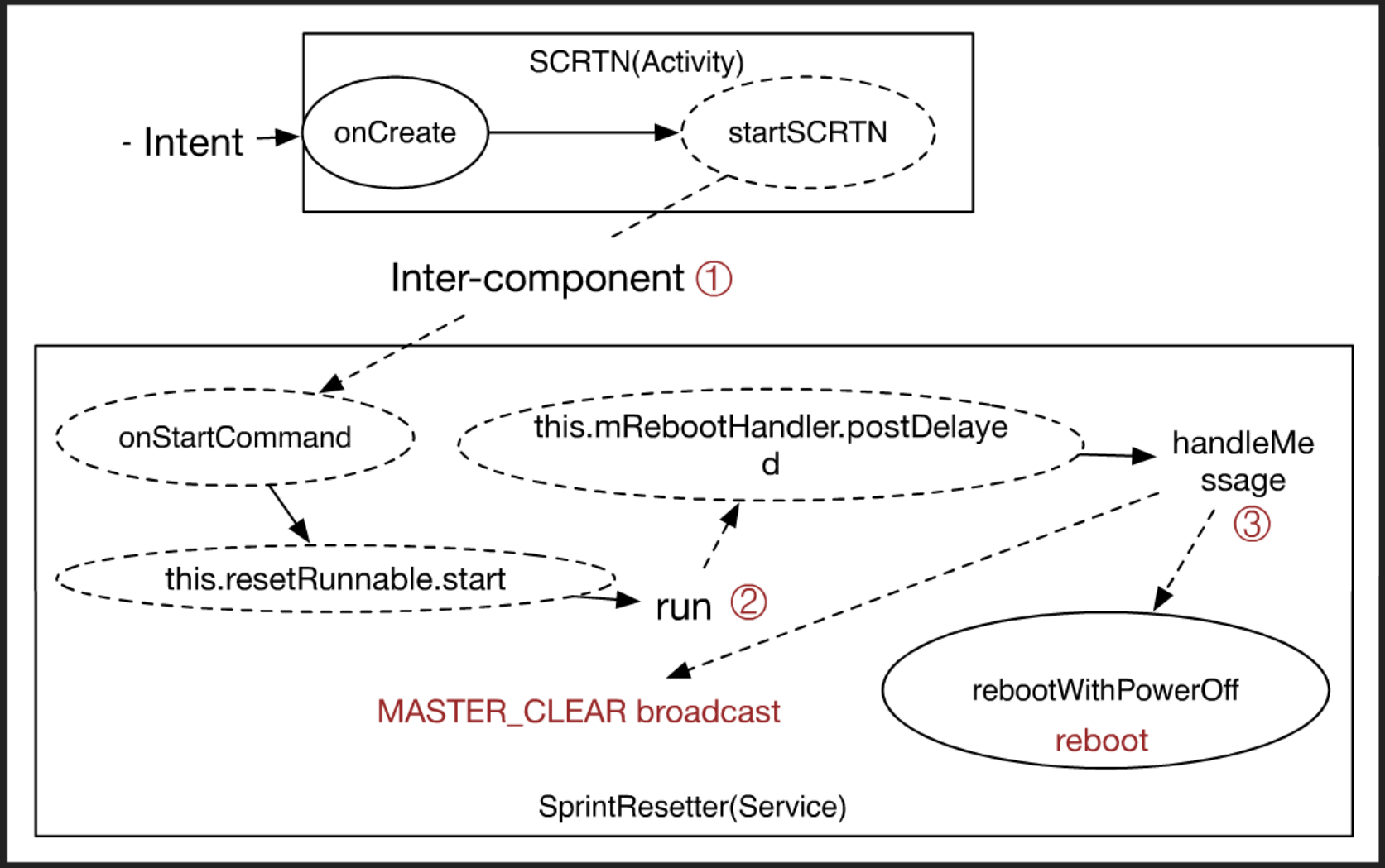
Quarks lab

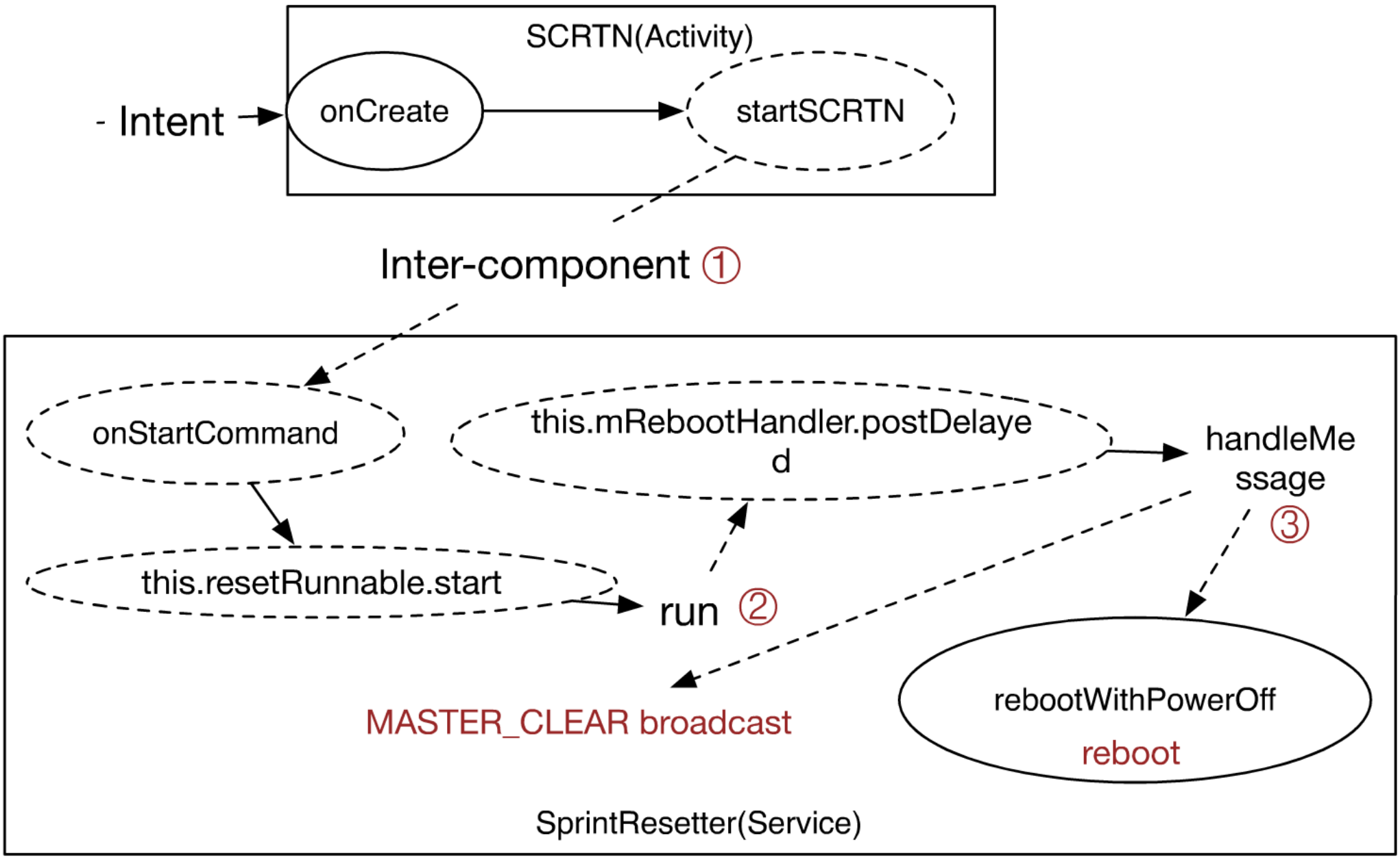
Nexus 5 Local DOS

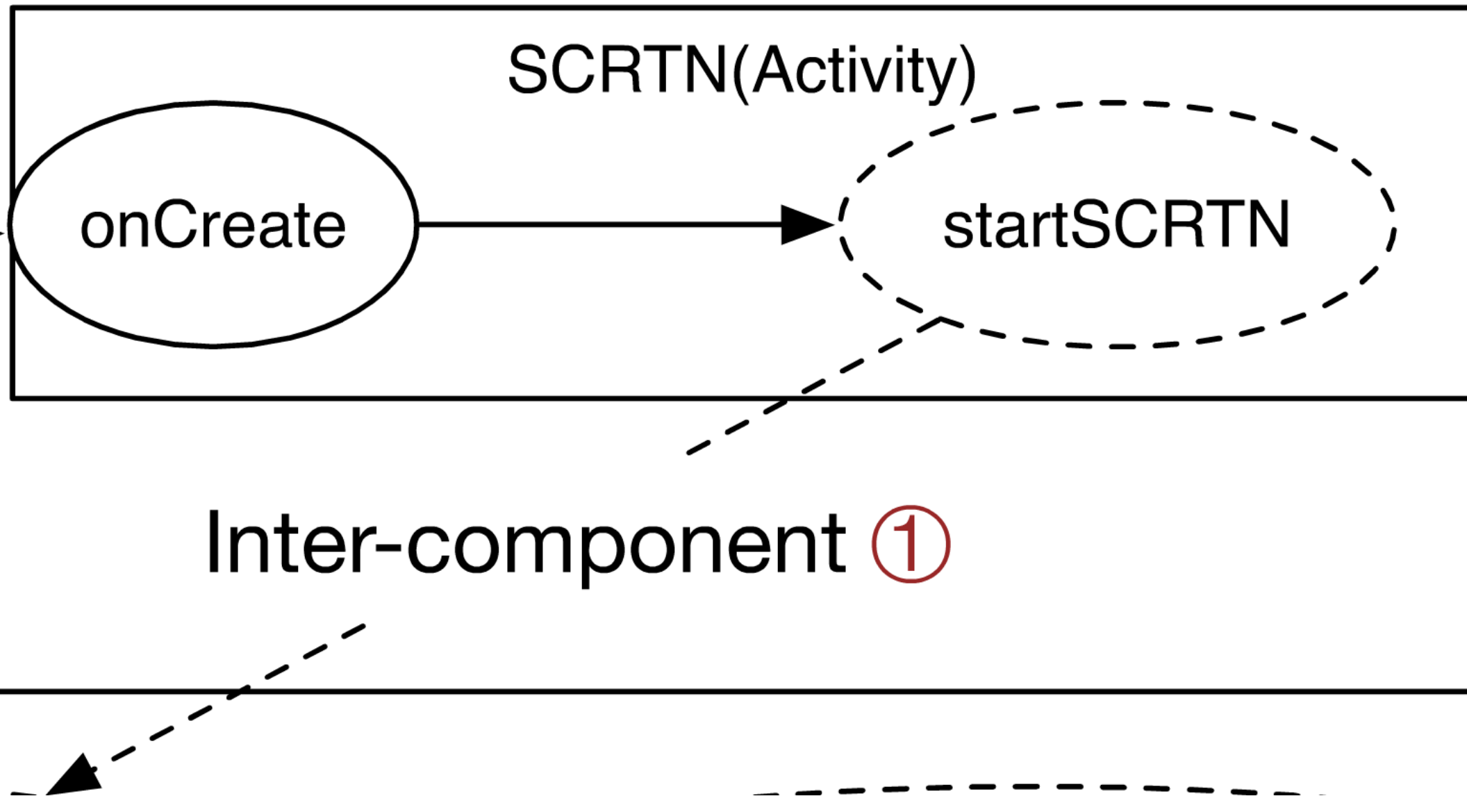


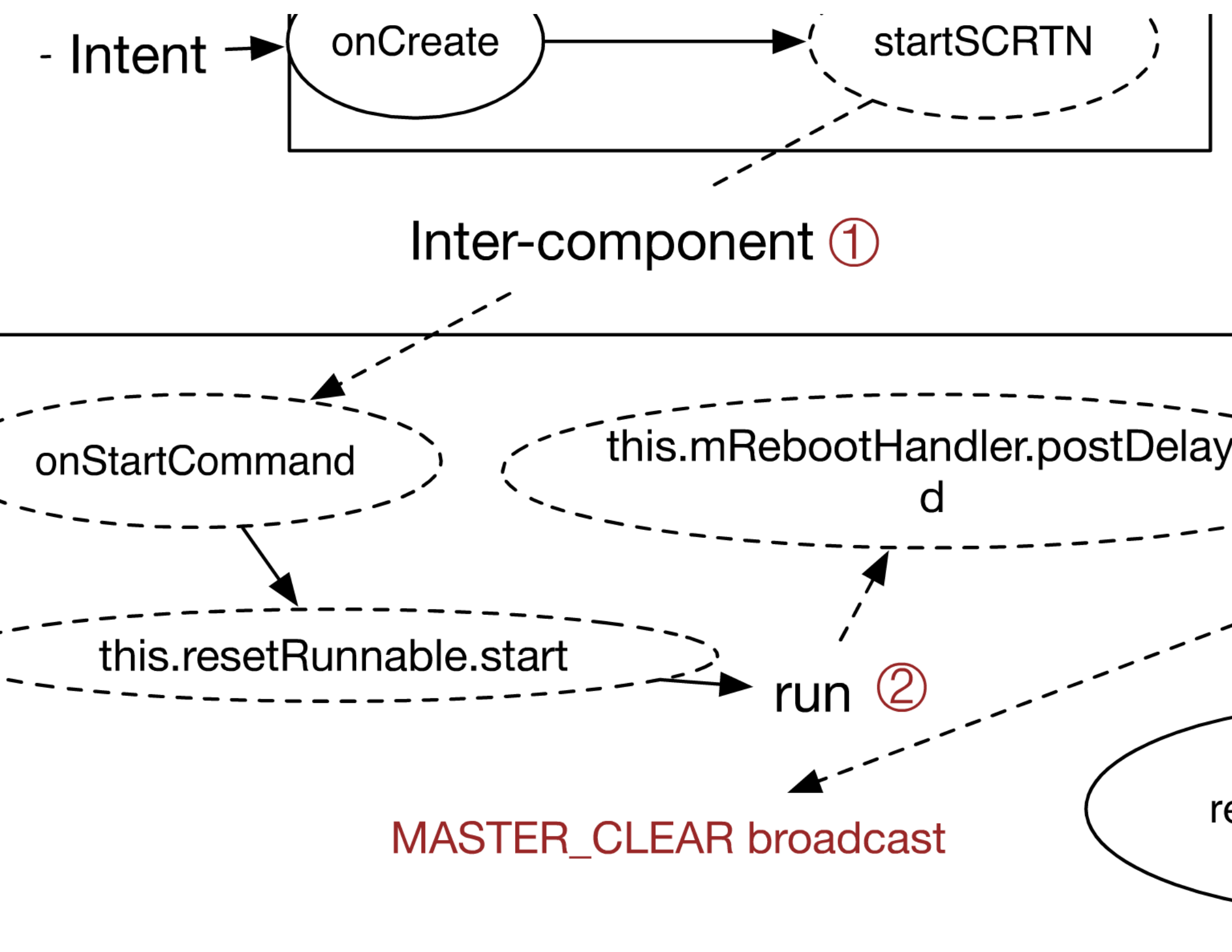
MWR labs

Nexus 5 Local DOS

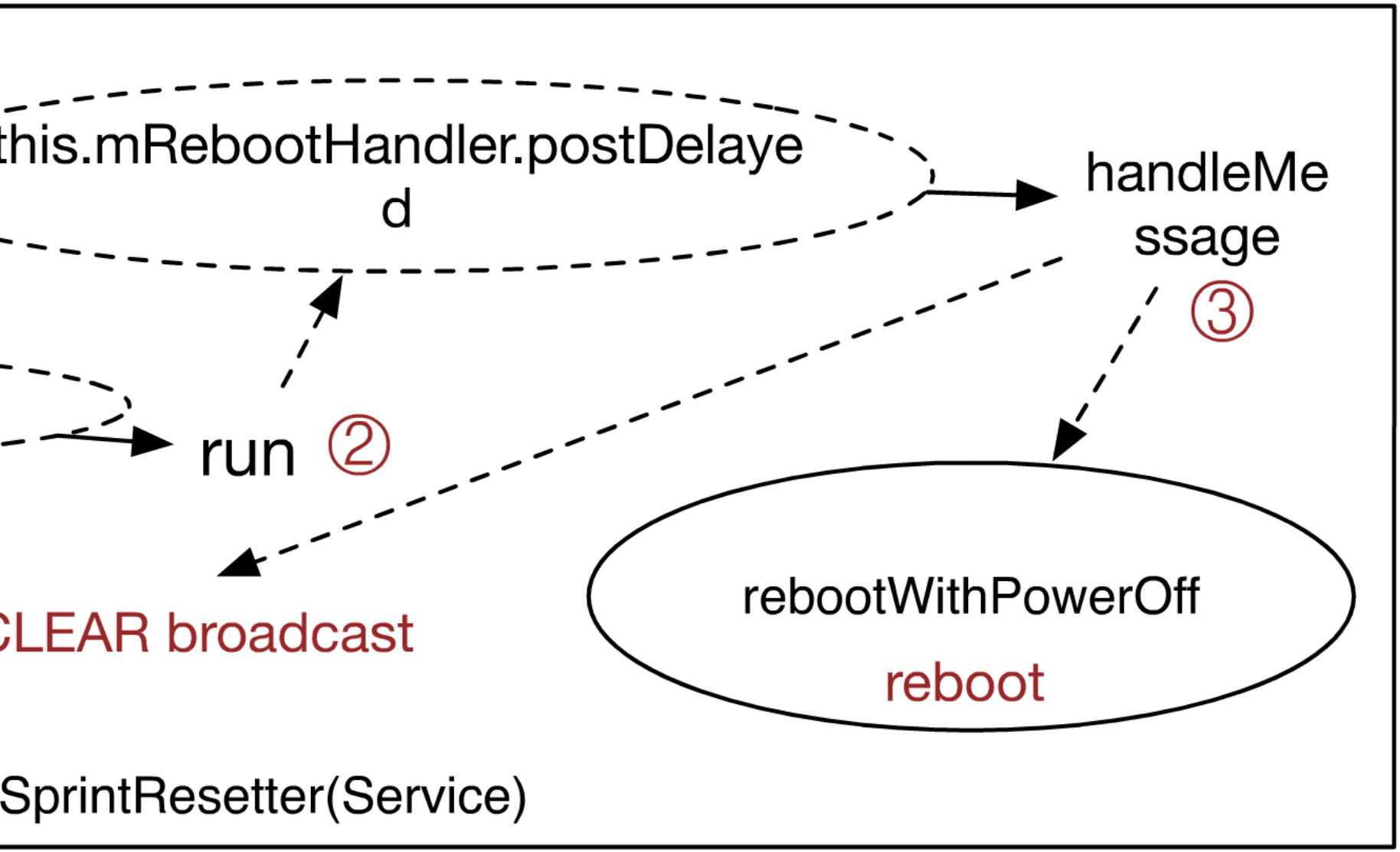



SCRTN(Activity)
- Intent
onCreate
startSCRTN
Inter-component ①
onStartCommand
this.mRebootHandler.postDelayed
handleMessage
③
this.resetRunnable.start
run ②
MASTER_CLEAR broadcast
rebootWithPowerOff
reboot
SprintResetter(Service)


SCRTN(Activity)
onCreate
startSCRTN
Inter-component ①

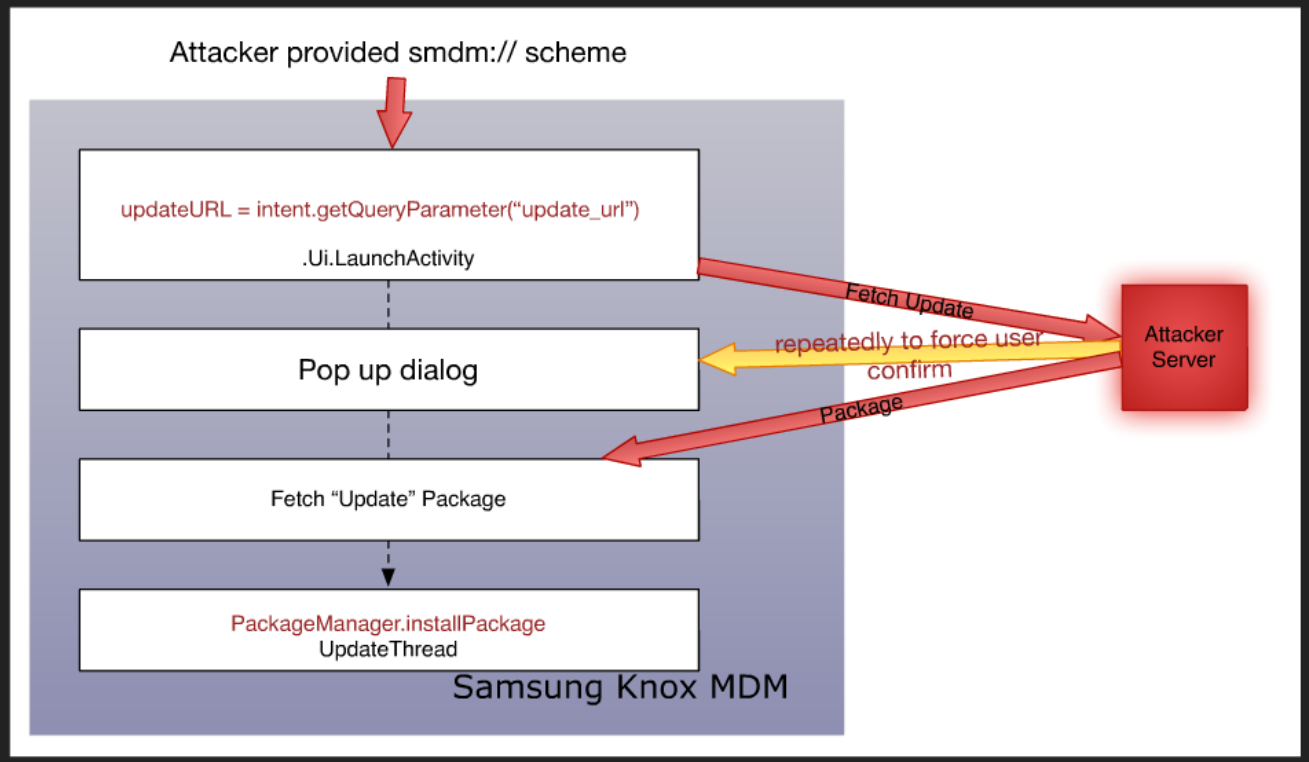

- Intent
onCreate
startSCRTN
Inter-component ①
onStartCommand
this.mRebootHandler.postDelay
d
this.resetRunnable.start
run ②
MASTER_CLEAR broadcast

mponent ①


this.mRebootHandler.postDelaye
d
handleMe
ssage
③
run ②
CLEAR broadcast
rebootWithPowerOff
reboot
SprintResetter(Service)

Capability Leak

Samsung Knox Remote code execution



Quarks lab

Other common vulns

- API misuse
  - HTTPS verification bypass
  - Insecure cypto
  - FileMode
  - ...
- Local Denial-of-service


10:24
错误反馈
抱歉，手机卫士运行出错了，为了帮助我们尽快改进和修复问题，我们希望您向我们发送以下信息：IMEI、机型和系统错误日志。其中IMEI和机型用于识别设备，错误日志用来记录出错时系统的运行状态，这些日志并不涉及您的个人信息和隐私
lpplpp0
以后再说
现在发送
中
En
English
?123

What's the industrial state-of-art?

xx lieshou
xx ju anquan
xx jingang

Unfortunately, these are just simple "grep"
- detail plz refer to my xDef slide

Moses, lead these people out of Egypt

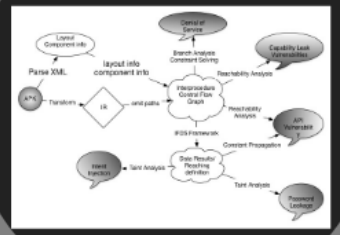
Why NOT program analysis?

- Java program analysis has been around for a long time
  - code-inspect, FindBug, etc.
- Dynamic analysis (fuzzing) is not a main option

API Misuse

- Interprocedure constant propagation (reaching definition)
- Class Hierachy Analysis

Finally



Data flow

- Taint analysis FlowIn
  - sources: intent, URI
  - sinks: sensitive API
- Taint anlysis FlowOut
  - sources: password, token, etc
  - sinks: insecure storage/channel

Capability leak

- Reachability analysis
- Query callgraph for sensitive APIs

Crash (DOS)

Most crashes caused by NPE or ClassCastException

Constraint solving is your friend!
plus taint analysis

Moses, lead these people out of Egypt

Why NOT program analysis?

- Java program analysis has been around for a long time
  - code-inspect, FindBug, etc.
- Dynamic analysis (fuzzing) is not a main option

API Misuse

- Interprocedure constant propagation (reaching definition)
- Class Hierachy Analysis

Data flow

- Taint analysis FlowIn
  - sources: intent, URI
  - sinks: sensitive API
- Taint anlysis FlowOut
  - sources: password, token, etc
  - sinks: insecure storage/channel

Capability leak

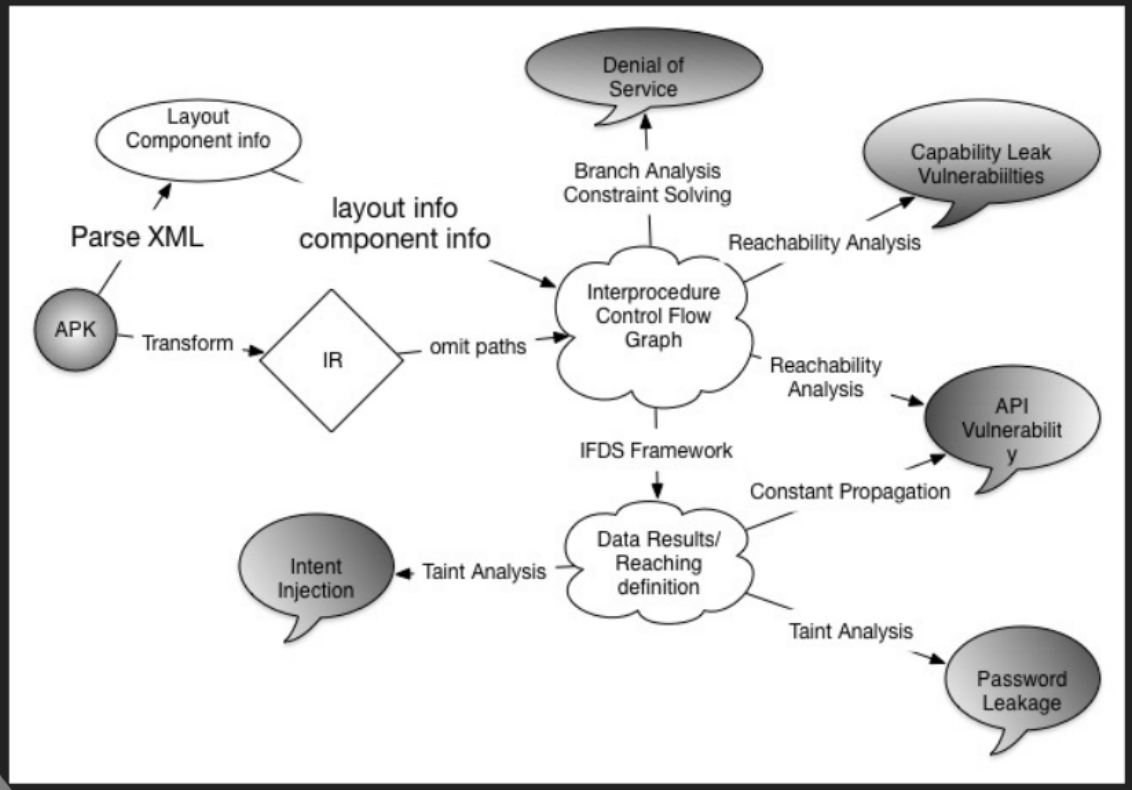
- Reachability analysis
- Query callgraph for sensitive APIs

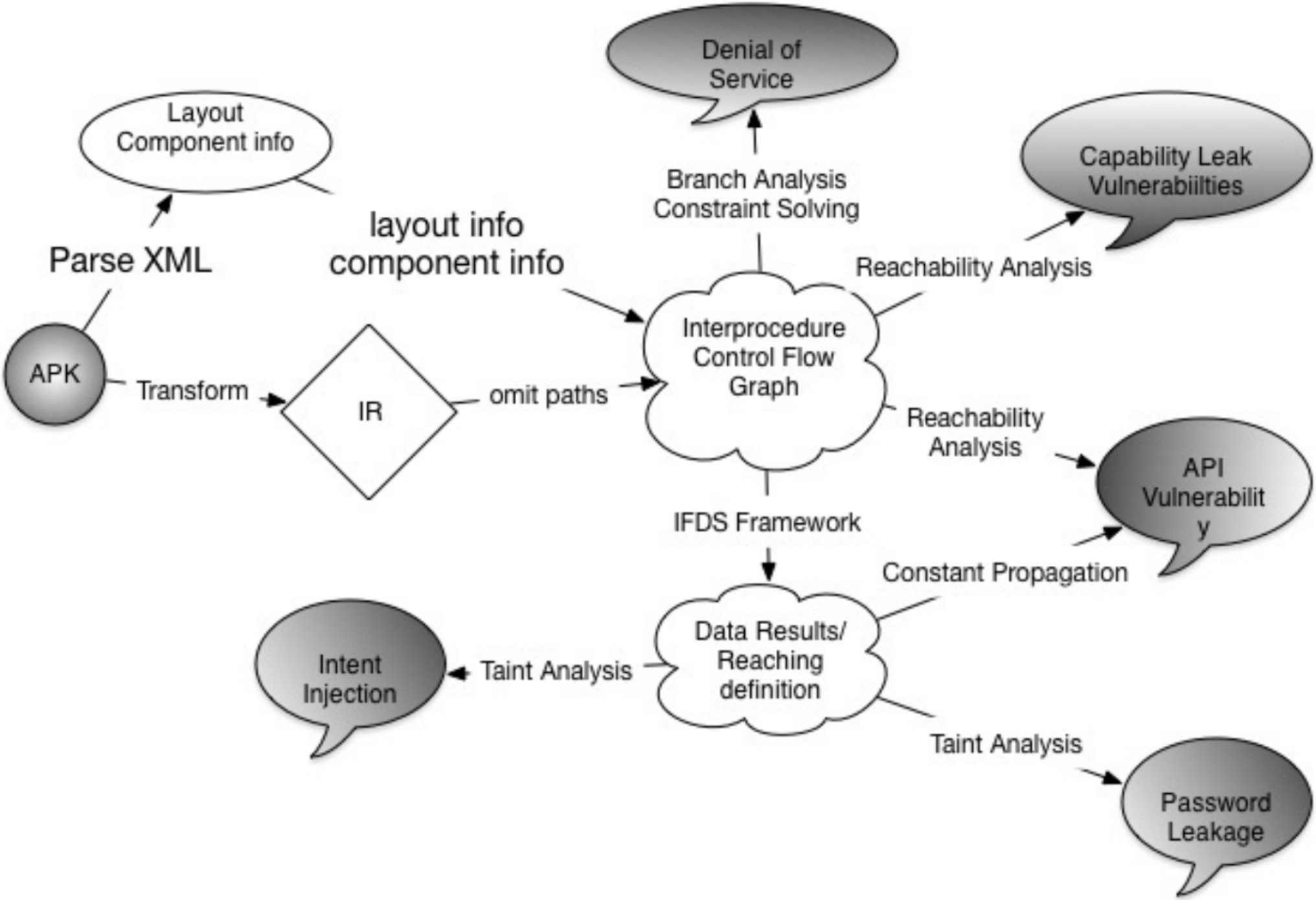
Crash (DOS)

Most crashes caused by NPE or ClassCastException

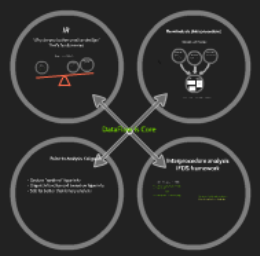
Constraint solving is your friend!
plus taint analysis

Finally


Denial of Service
Layout Component info
Branch Analysis Constraint Solving
Capability Leak Vulnerabiilties
Parse XML
layout info component info
Reachability Analysis
Interprocedure Control Flow Graph
APK
Transform
IR
omit paths
Reachability Analysis
API Vulnerabilit y
IFDS Framework
Constant Propagation
Data Results/ Reaching definition
Intent Injection
Taint Analysis
Taint Analysis
Password Leakage


Denial of Service
Layout Component info
Branch Analysis Constraint Solving
Capability Leak Vulnerabiilties
Parse XML
layout info component info
Reachability Analysis
Interprocedure Control Flow Graph
APK
Transform
IR
omit paths
Reachability Analysis
API Vulnerabilit y
IFDS Framework
Constant Propagation
Data Results/ Reaching definition
Intent Injection
Taint Analysis
Taint Analysis
Password Leakage

It's time for some concepts and *real* practice!



IR

Why do you bother smali or dex2jar?
That's for dummies

Jimple vs smali&dex2jar

FlowAnalysis (Intraprocedure)

Concepts and Example

DataFlow is Core

Point-to Analysis, Callgraph

- Deduce "runtime" type info
- Dispatch function call based on type info
- Still far better than binary analysis

Interprocedure analysis
IFDS framework

Given Interprocedural CFG

Precise Interprocedural Dataflow Analysis via Graph Reachability

POPL

Thomas Reps,Susan Horwitz, Mooly Sagiv

IR

Why do you bother smali or dex2jar?
That's for dummies

Jimple vs smali&dex2jar

What's for dummies

Jimple vs smali&dex2jar

Jimple

- 15 insts
- constant propagation done on the fly
- handful APIs
- Direct transformation via dexpler

Even better! Callgraph/ControlFlowGraph boosted

smali/dalvik bytecode

- › 200 instructions
- typeless
- no constant information

dex2jar & java bytecode & java source

- full of bugs, easy to crash
- AST is far from enough
- bytecode › 200 insts

dex2jar & java bytecode & java source

- full of bugs, easy to crash
- AST is far from enough
- bytecode > 200 insts

smali/dalvik bytecode

- › 200 instructions
- typeless
- no constant information

```
.method private testString(String)V
          .registers 3
          .param p1, "string"
          .prologue
00000000  sget-object            v0, System->out:PrintStream
00000004  invoke-virtual         PrintStream->println(String)V, v0, p1
0000000A  return-void
.end method
```

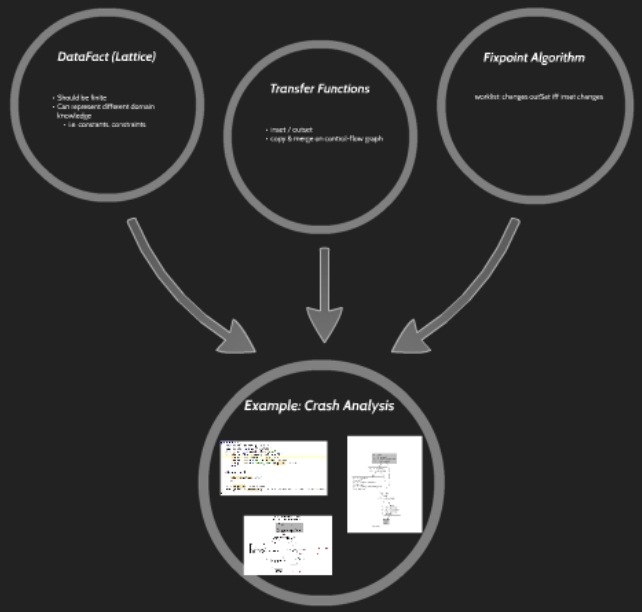
Jimple

- 15 insts
- constant propagation done on the fly
- handful APIs
- Direct transformation via dexpler

Even better! Callgraph/ControlFlowGraph boosted

FlowAnalysis (Intraprocedure)

Concepts and Example



A Einarsson: A Survivor's Guide to Java Program analysis with Soot

Concepts and Example

DataFact (Lattice)

- Should be finite
- Can represent different domain knowledge
  - i.e. constants, constraints

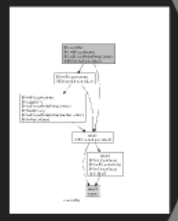
Transfer Functions

- inset / outset
- copy & merge on control-flow graph

Fixpoint Algorithm

worklist: changes outSet iff inset changes

Example: Crash Analysis



DataFact (Lattice)

- Should be finite
- Can represent different domain knowledge
    - i.e. constants, constraints

Transfer Functions

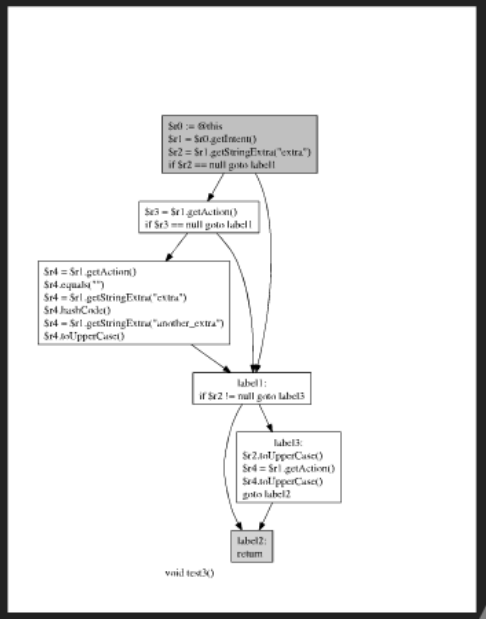
- inset / outset
- copy & merge on control-flow graph

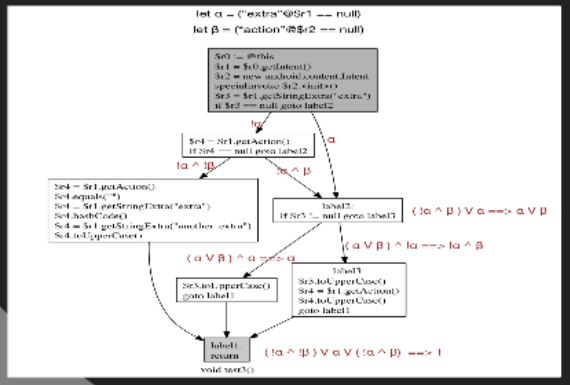
Fixpoint Algorithm

worklist: changes outSet iff inset changes

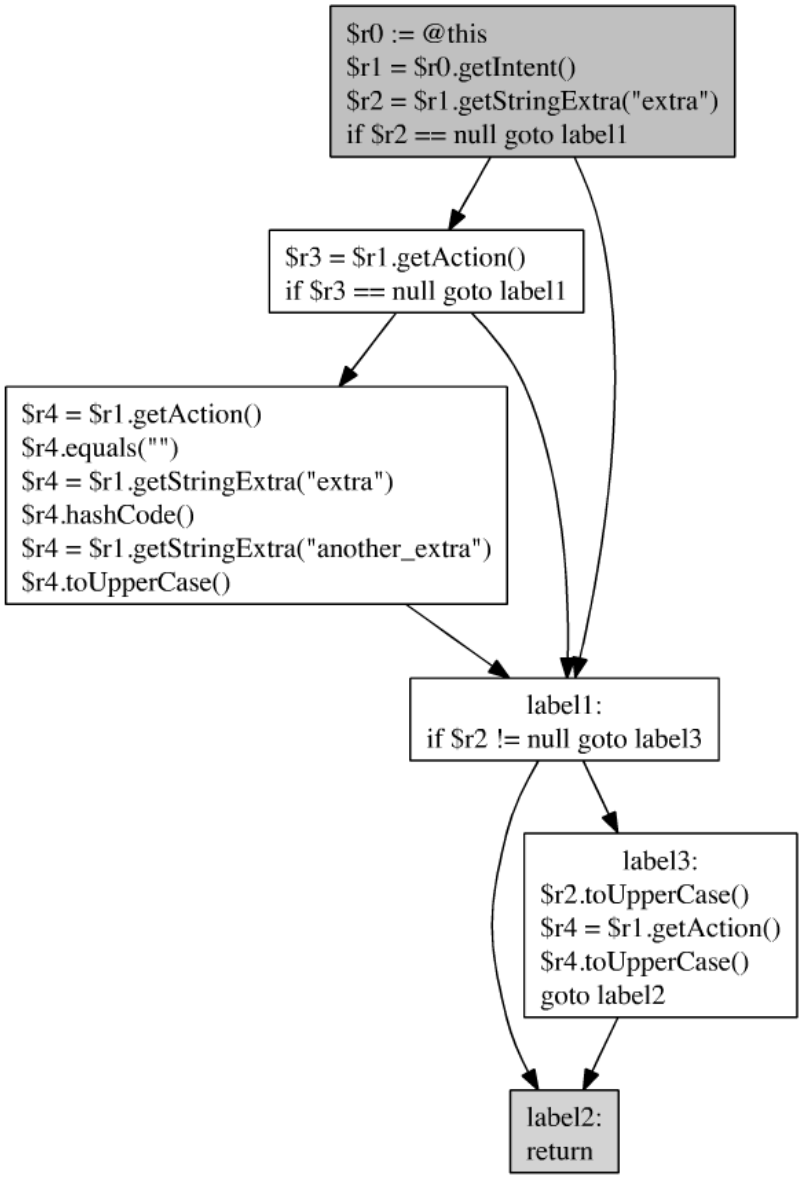
Example: Crash Analysis

```
public void test3() {
    Intent intent = getIntent(); //SOURCE A
    Intent intent1 = new Intent(); //SOURCE B
    String string = intent.getStringExtra("extra");
    if (string != null && intent.getAction() != null) {
        intent.getAction().equals(""); //WON'T CRASH
        intent.getStringExtra("extra").hashCode();//NOT CRASH
        intent.getStringExtra("another_extra").toUpperCase();//CRASH
        return;
    }
    if(string == null)
    {
        string.toUpperCase(); //CRASH
        return;
    }
    string.toUpperCase(); //WON'T CRASH
    intent.getAction().toUpperCase(); // WILL CRASH BECAUSE CONSTRAINT SAYS ['action']@A == null
}
```





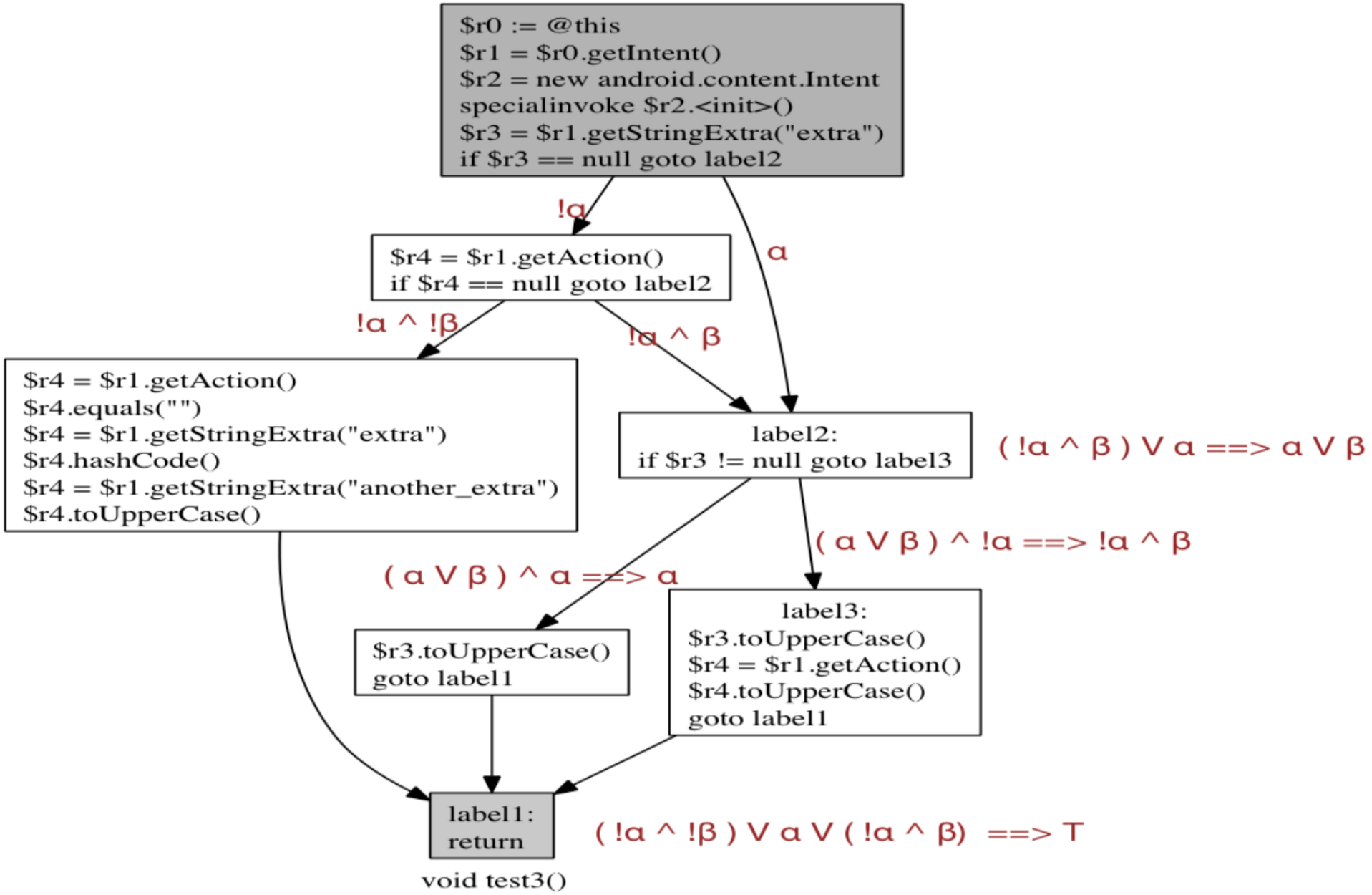
```java
public void test3() {
    Intent intent = getIntent(); //SOURCE A
    Intent intent1 = new Intent(); //SOURCE B
    String string = intent.getStringExtra("extra");
    if (string != null && intent.getAction() != null) {
        intent.getAction().equals(""); //WON'T CRASH
        intent.getStringExtra("extra").hashCode();//NOT CRASH
        intent.getStringExtra("another_extra").toUpperCase();//CRASH
        return;
    }
    if(string == null)
    {
        string.toUpperCase(); //CRASH
        return;
    }
    string.toUpperCase(); //WON'T CRASH
    intent.getAction().toUpperCase(); // WILL CRASH BECAUSE CONSTRAINT SAYS ['action']@A == null
}
```


$r0 := @this
$r1 = $r0.getIntent()
$r2 = $r1.getStringExtra("extra")
if $r2 == null goto label1
$r3 = $r1.getAction()
if $r3 == null goto label1
$r4 = $r1.getAction()
$r4.equals("")
$r4 = $r1.getStringExtra("extra")
$r4.hashCode()
$r4 = $r1.getStringExtra("another_extra")
$r4.toUpperCase()
label1:
if $r2 != null goto label3
label3:
$r2.toUpperCase()
$r4 = $r1.getAction()
$r4.toUpperCase()
goto label2
label2:
return

void test3()

let α = (“extra”@$r1 == null)

let β = (“action”@$r2 == null)



void test3()

Point-to Analysis, Callgraph

- Deduce "runtime" type info
- Dispatch function call based on type info
- Still far better than binary analysis

Interprocedure analysis IFDS framework

Given Interprocedural CFG

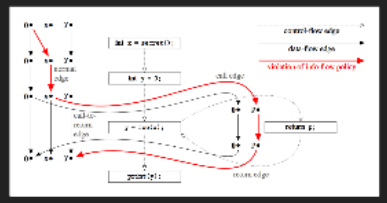
Precise Interprocedural Dataflow Analysis via Graph Reachability

POPL

Thomas Reps,Susan Horwitz, Mooly Sagiv

Inter-procedural, finite, distributive subset problem

Define custom transfer functions at graph edges



Given Interprocedural CFG

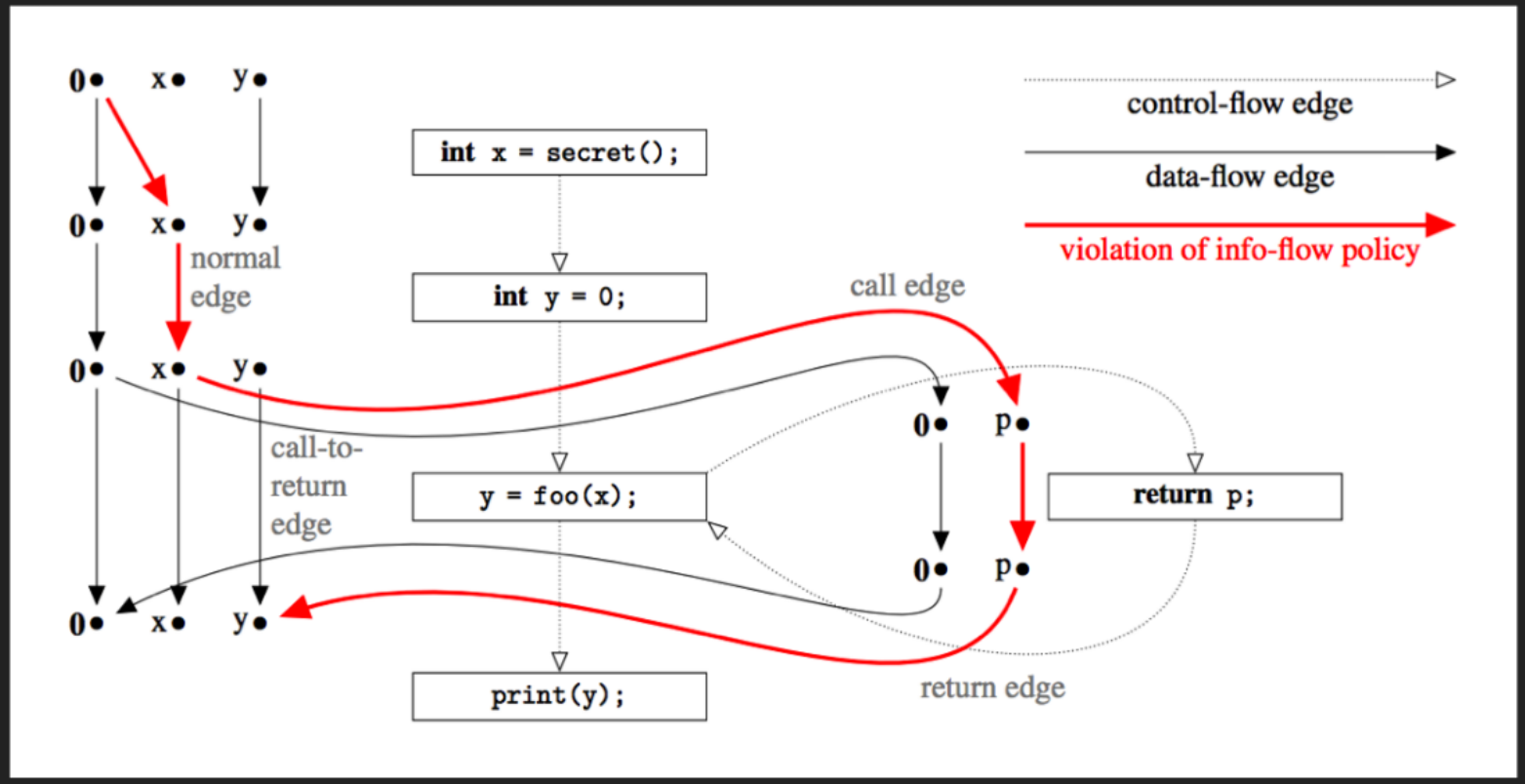
Precise Interprocedural Dataflow Analysis via Graph Reachability

POPL

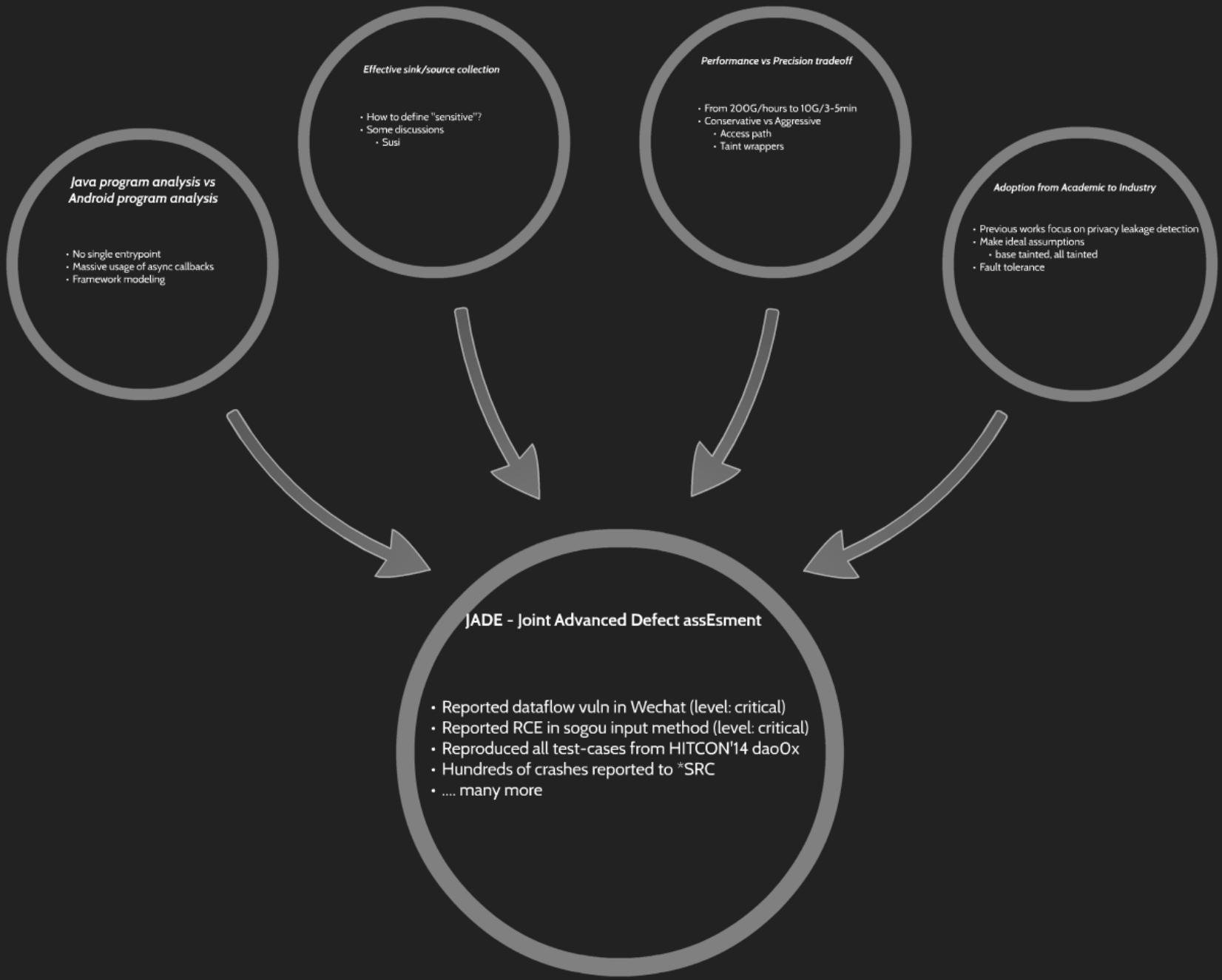
Thomas Reps,Susan Horwitz, Mooly Sagiv

Inter-procedural, finite, distributive subset problem

Define custom transfer functions at graph edges



Additional work



Java program analysis vs Android program analysis

- No single entrypoint
- Massive usage of async callbacks
- Framework modeling

Effective sink/source collection

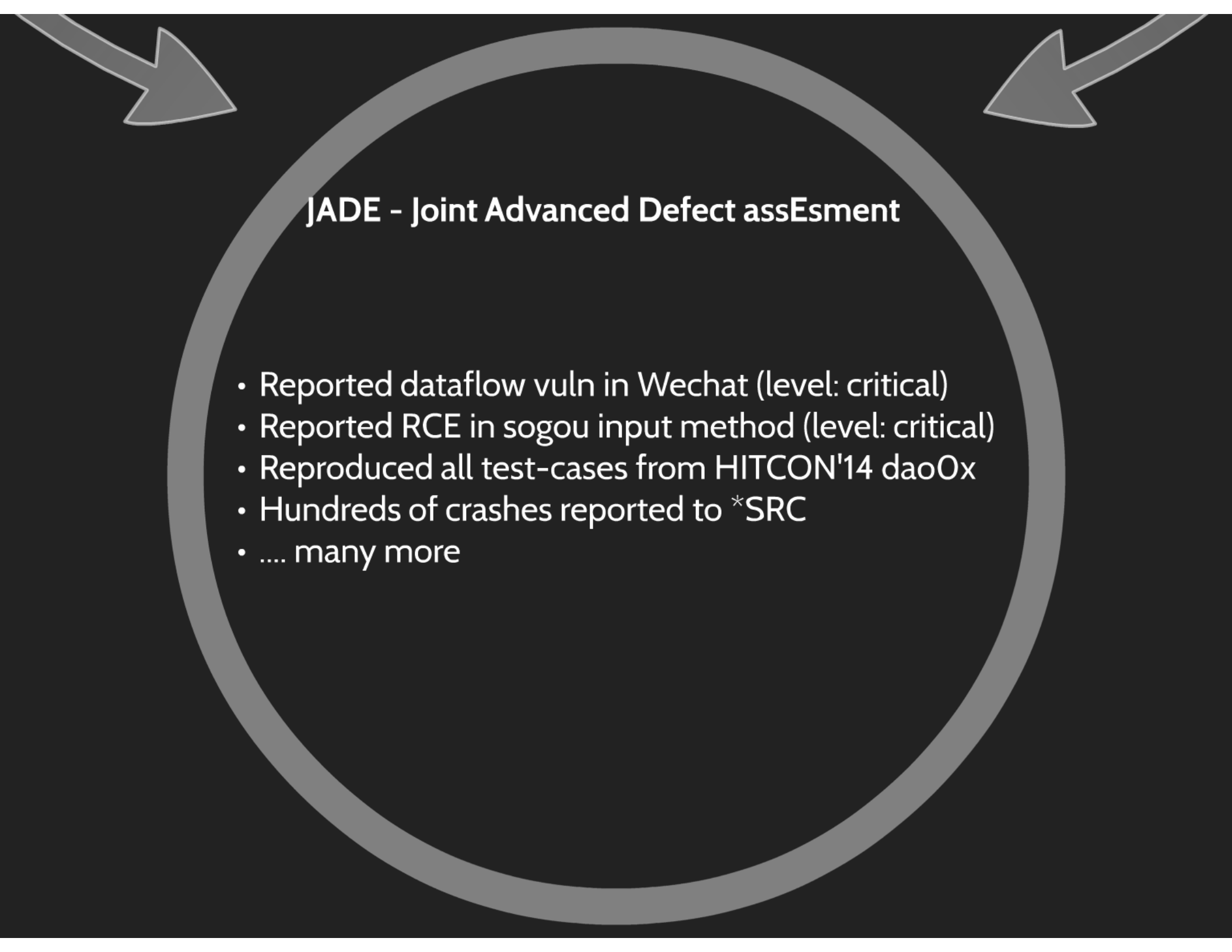
- How to define "sensitive"?
- Some discussions
  - Susi

Performance vs Precision tradeoff

- From 200G/hours to 10G/3-5min
- Conservative vs Aggressive
  - Access path
  - Taint wrappers

Adoption from Academic to Industry

- Previous works focus on privacy leakage detection
- Make ideal assumptions
  - base tainted, all tainted
- Fault tolerance


JADE - Joint Advanced Defect assEsment
• Reported dataflow vuln in Wechat (level: critical)
• Reported RCE in sogou input method (level: critical)
• Reproduced all test-cases from HITCON'14 dao0x
• Hundreds of crashes reported to *SRC
• .... many more

Conclusion

- Definitions of Android App Vulnerabilities
- Vulnerabilities category
- Program analysis methods used for assesments
- Efforts made to turn research prototype into industrial product

Credits/Refs

- FlowDroid: Precise Context, Flow, Field, Object-sensitive and Lifecycle-aware Taint Analysis for Android Apps , PLDI'14
- A Precise and General Inter-component Data Flow Analysis Framework for Security Vetting of Android Apps, CCS'14
- Precise Interprocedural Dataflow Analysis via Graph Reachability, POPL'95

Thanks!

Weibo: Flanker_KEEN

KEEN TEAM