

# Perplexed Messengers from the Cloud:

## *Automated Security Analysis of Push-Messaging Integrations*

Yangyi Chen<sup>1\*</sup>, Tongxin Li<sup>2\*</sup>, XiaoFeng Wang<sup>1</sup>, ***Kai Chen***<sup>1, 3</sup> and Xinhui Han<sup>2</sup>

<sup>1</sup>Indiana University Bloomington

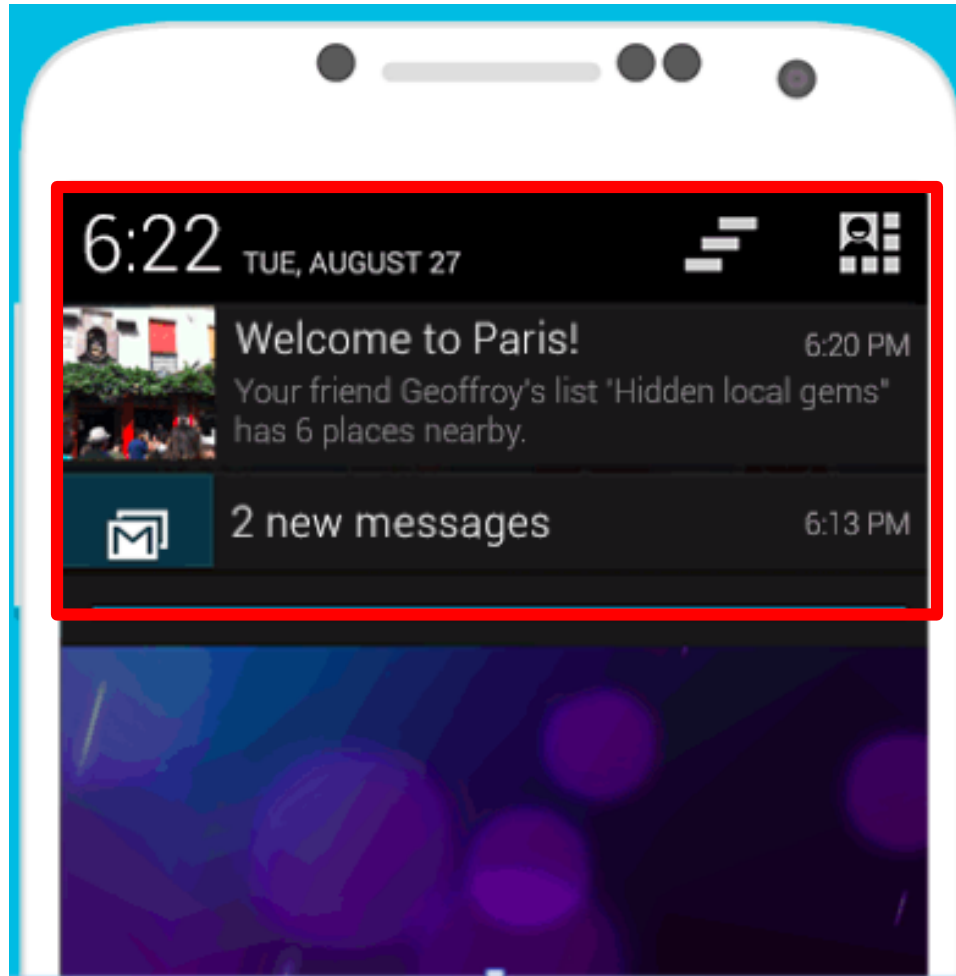
<sup>2</sup>Peking University

<sup>3</sup>Institute of Information Engineering, Chinese Academy of Sciences

---

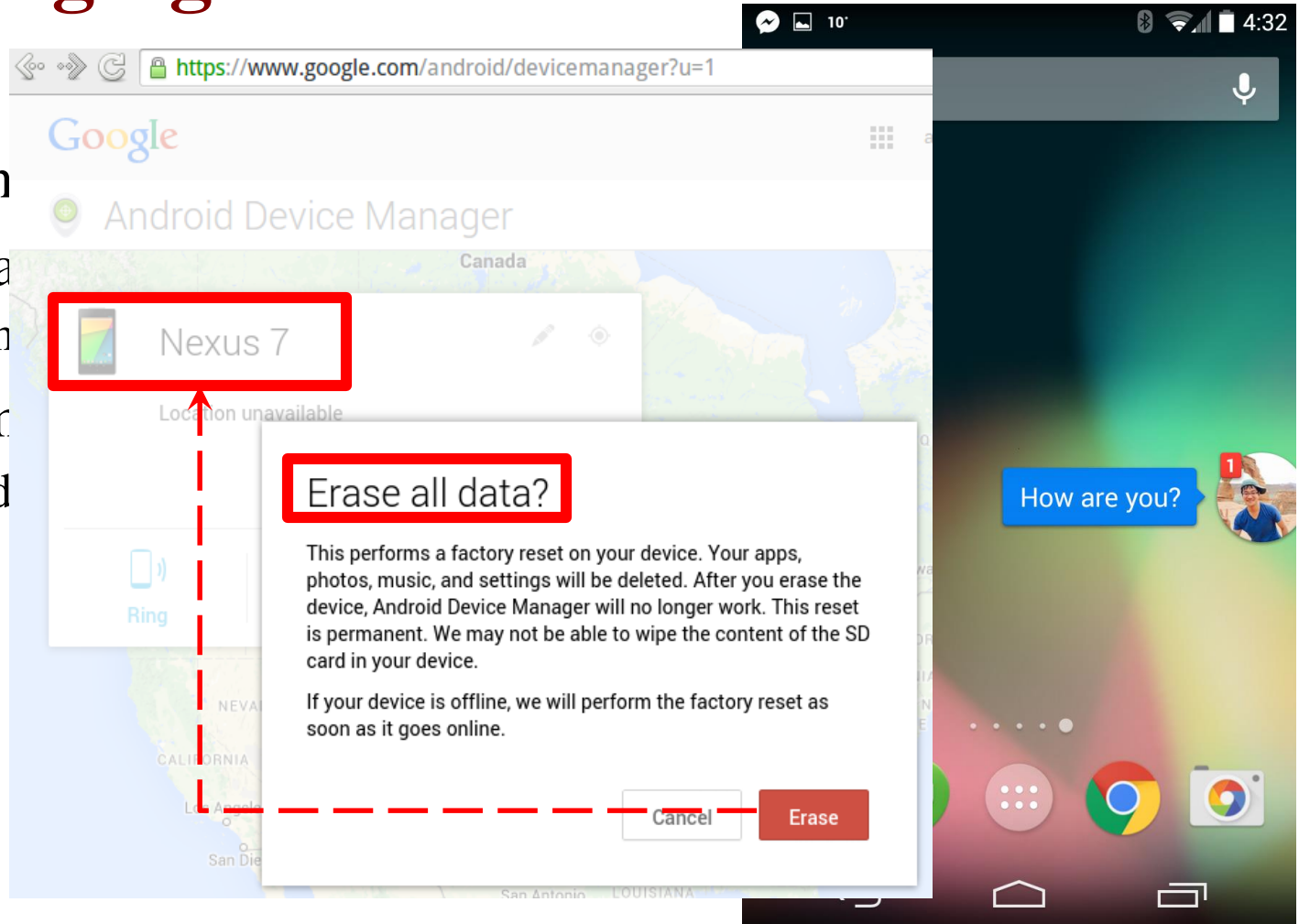
\*The names of the first two authors are in alphabetical order.

# What are Push Messages?



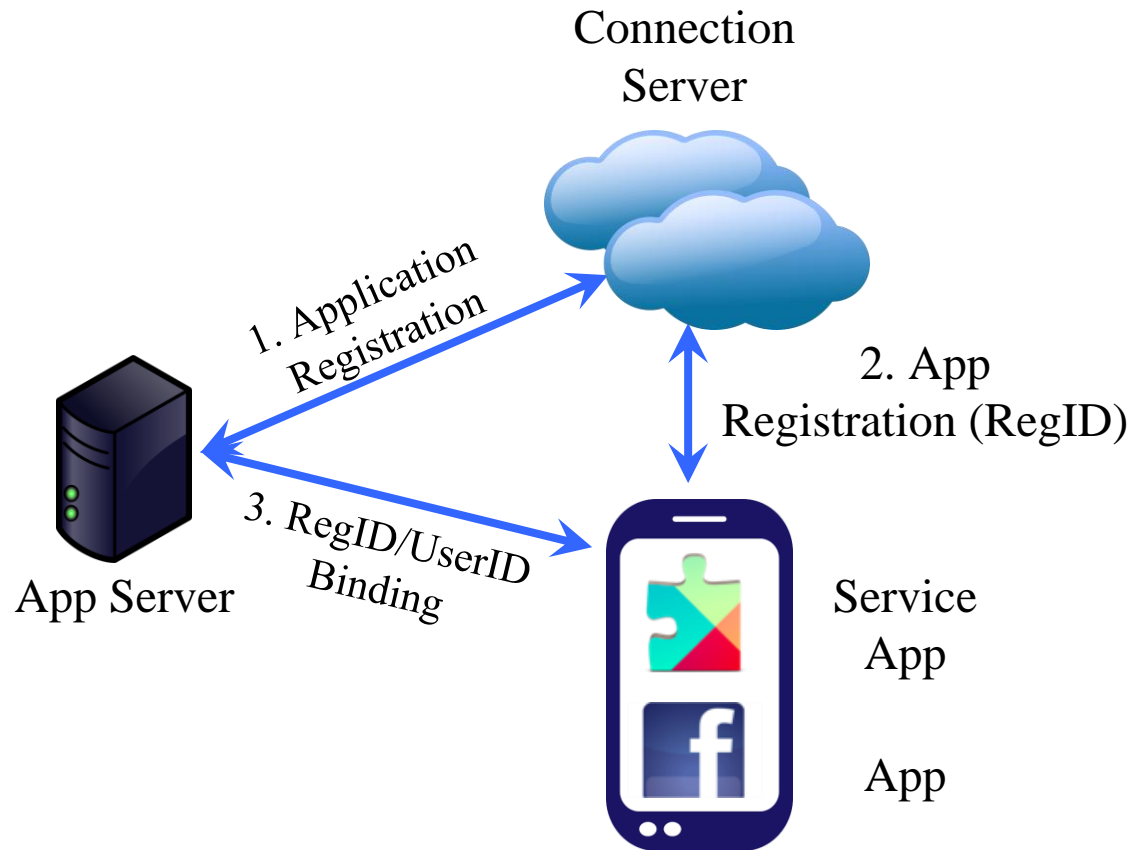
# Push-Messaging Services

- Major chan
- Push messa
  - Private m
- Push comm
  - Erasing d

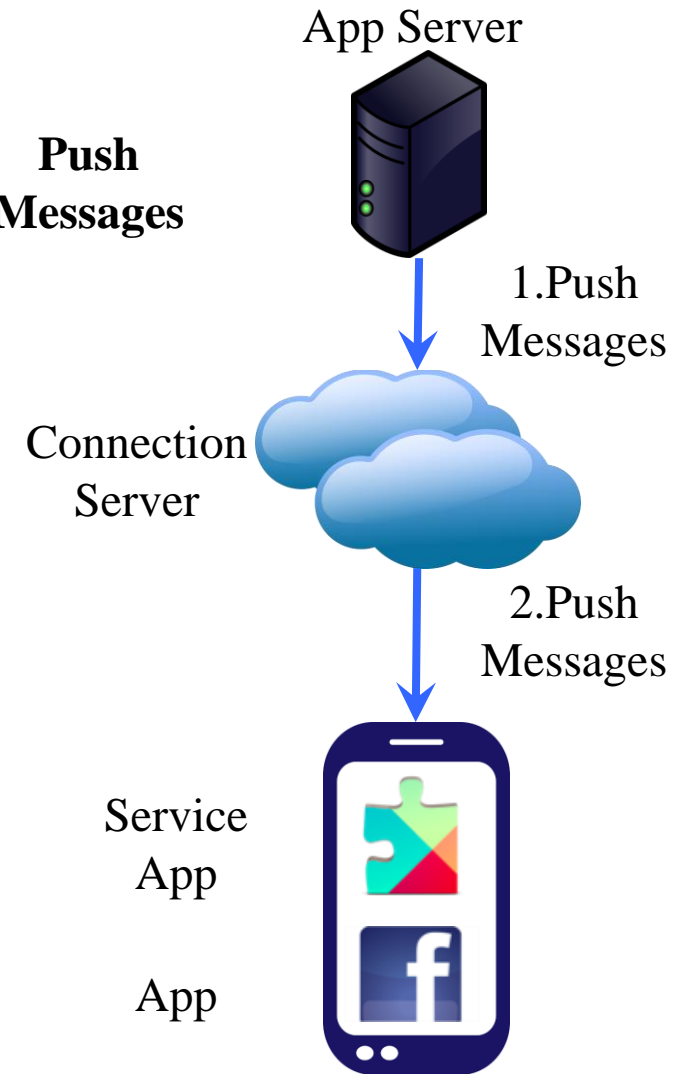


# How to Push Messages?

## Set Up



## Push Messages



# Different Push-Messaging Services

- Manufacturer Push-Messaging Services
    - Google Cloud Messaging(GCM), Amazon Device Messaging(ADM), etc.
  - Third-party Services (Chinese Market)
    - SDK instead of service app
    - Baidu Push, JPush, etc.
  - Syndication Services
    - Urban Airship, PushIO, Push Woosh, etc.
-

# Are Push-Messaging Services Secure?

- Publication from CCS 2014
    - **Manually** analyzed **4** services
      - GCM, ADM, UrbanAirship and one Chinese service
    - **Manually** discovered security risks in **63** apps
      - Steal/Inject push messages
  
  - Tip of iceberg?
-

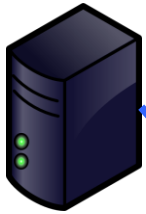
# What we do

- Identified a set of security principles and properties
  - Seminal
    - Evaluates the security qualities of the service's SDKs and their integrations within different apps
  - **Automatically** scanned
    - **30** services
    - **35,173** apps
  - Discovered **new** types of security risks
-

# Security Principles and Properties

## Principles:

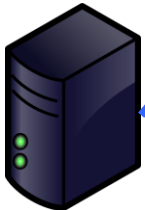
Authorized  
App Server



User



Malicious  
App Server



Attacker



## Properties:

- The app integrating the service should communicate with the service or the SDK through an authenticated secure channel.
- The app should check whether an incoming message belongs to the current user.
- CID (the authentication token for the third-party or syndicated service) should always be kept secret.

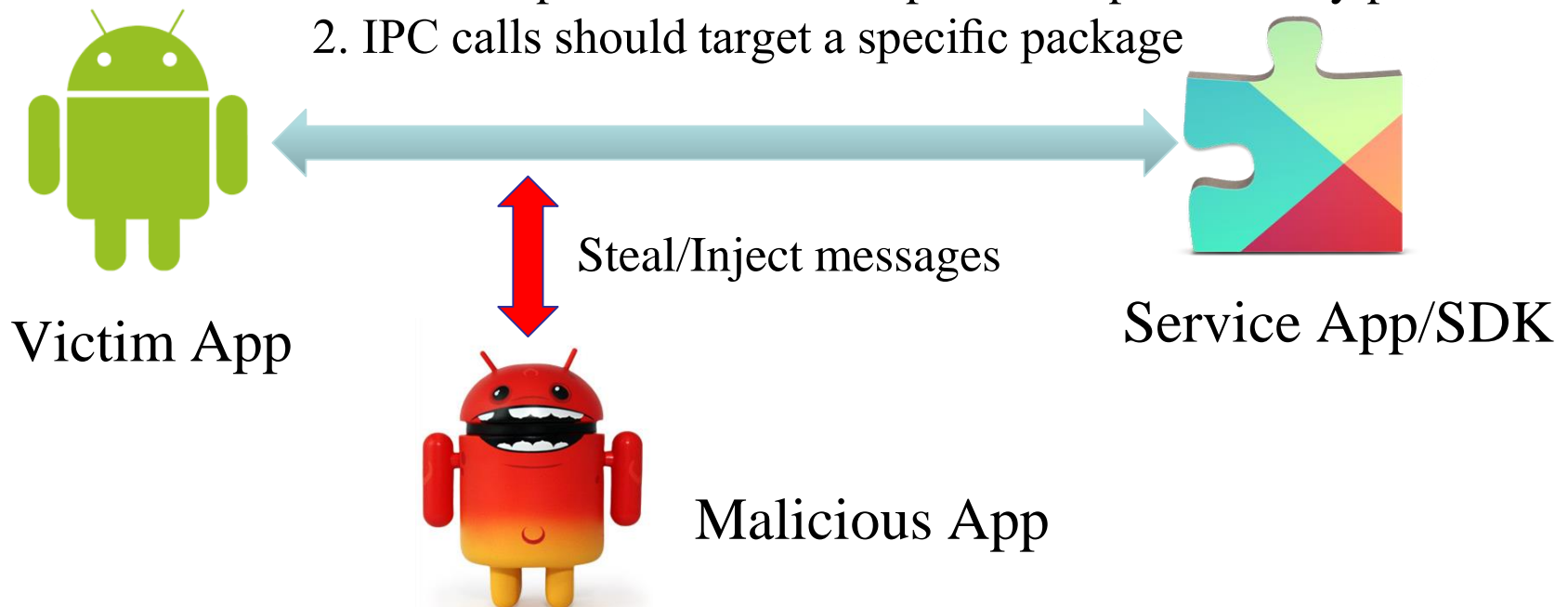


# Attacks

- Unprotected IPC channel
- From property: Secure channel

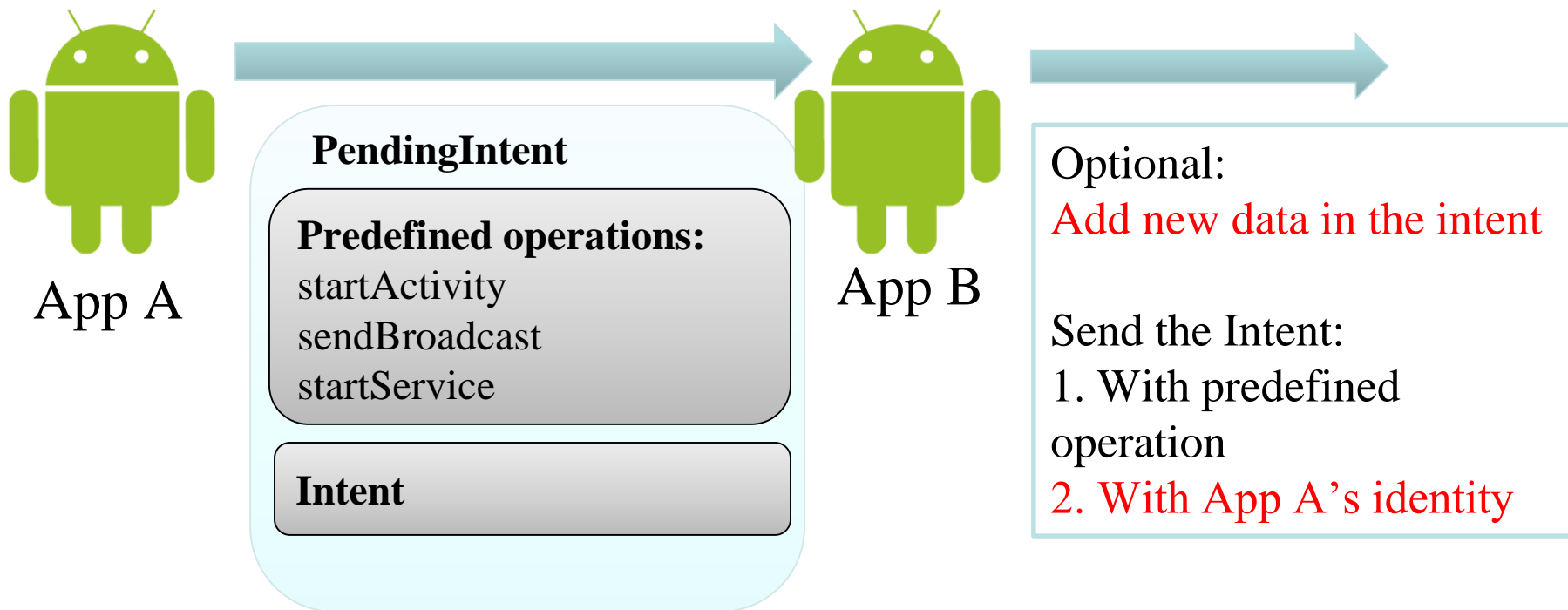
Protection:

1. IPC components should be private or protected by permissions
2. IPC calls should target a specific package



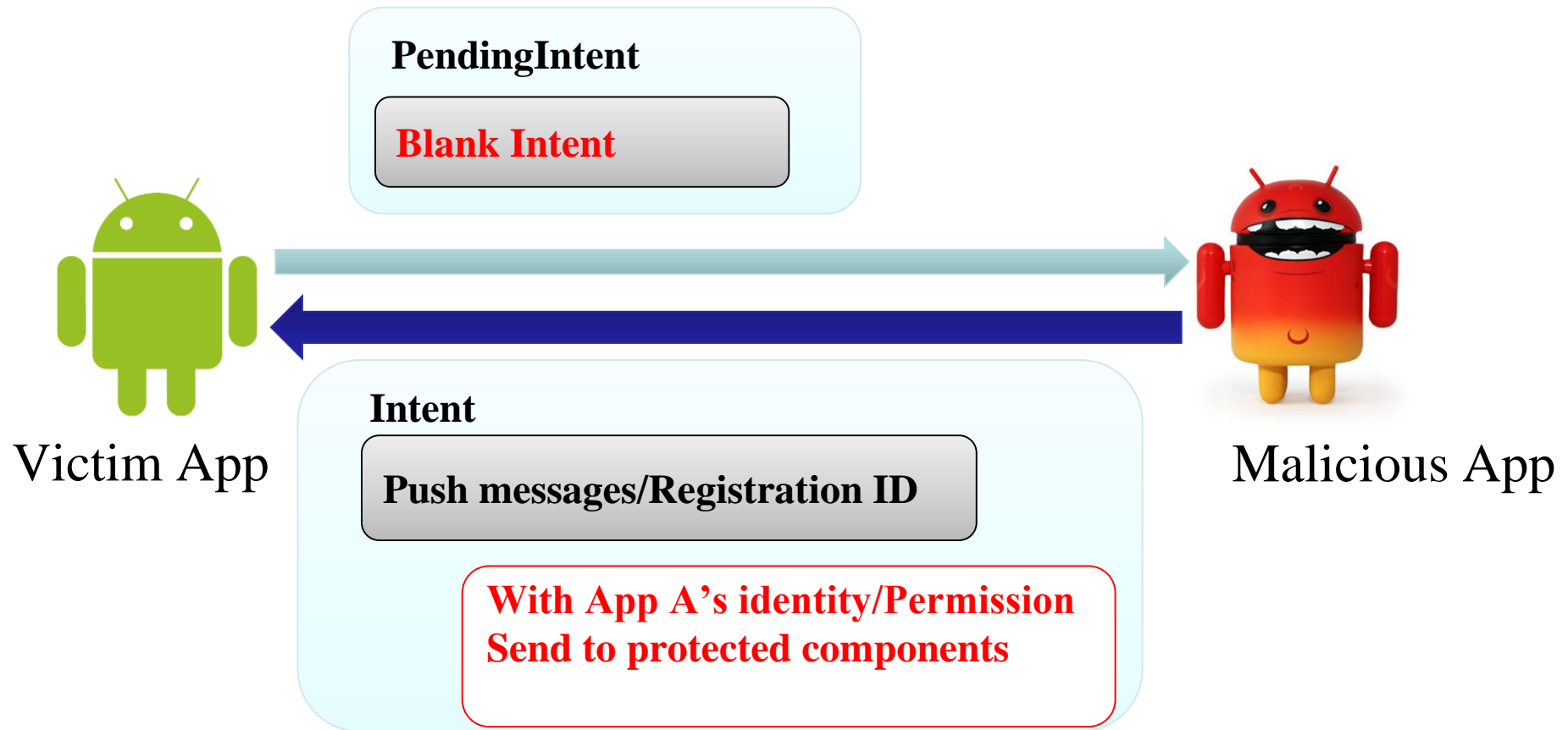
# Attacks

- PendingIntent Leakage
  - From property: Secure Channel
  - PendingIntent can break the secure IPC channel



# Attacks

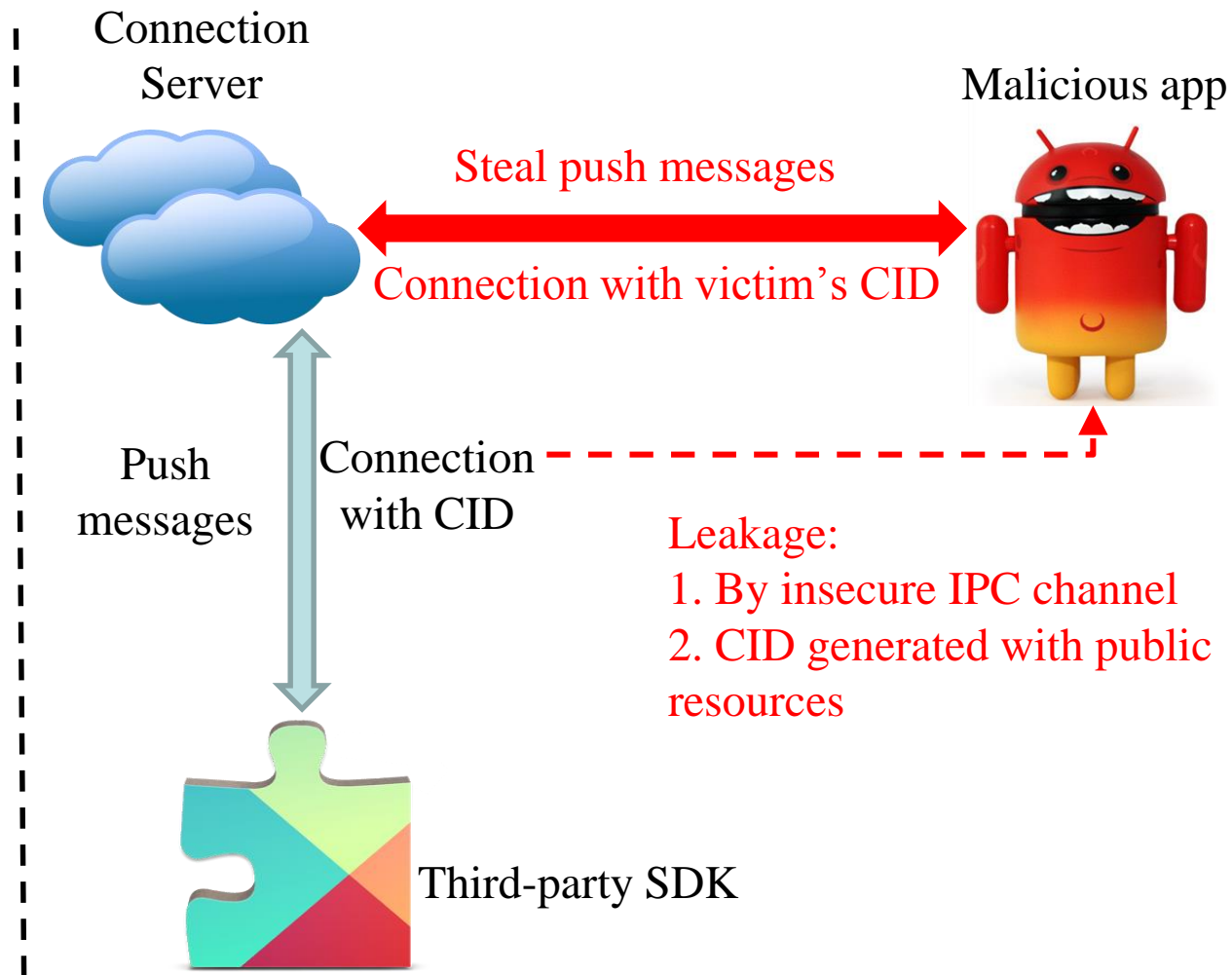
- PendingIntent Leakage



# Attacks

## CID leakage

- From property:
  - Keep CID secret
- CID:
  - Registration ID
  - Can be used to get push messages

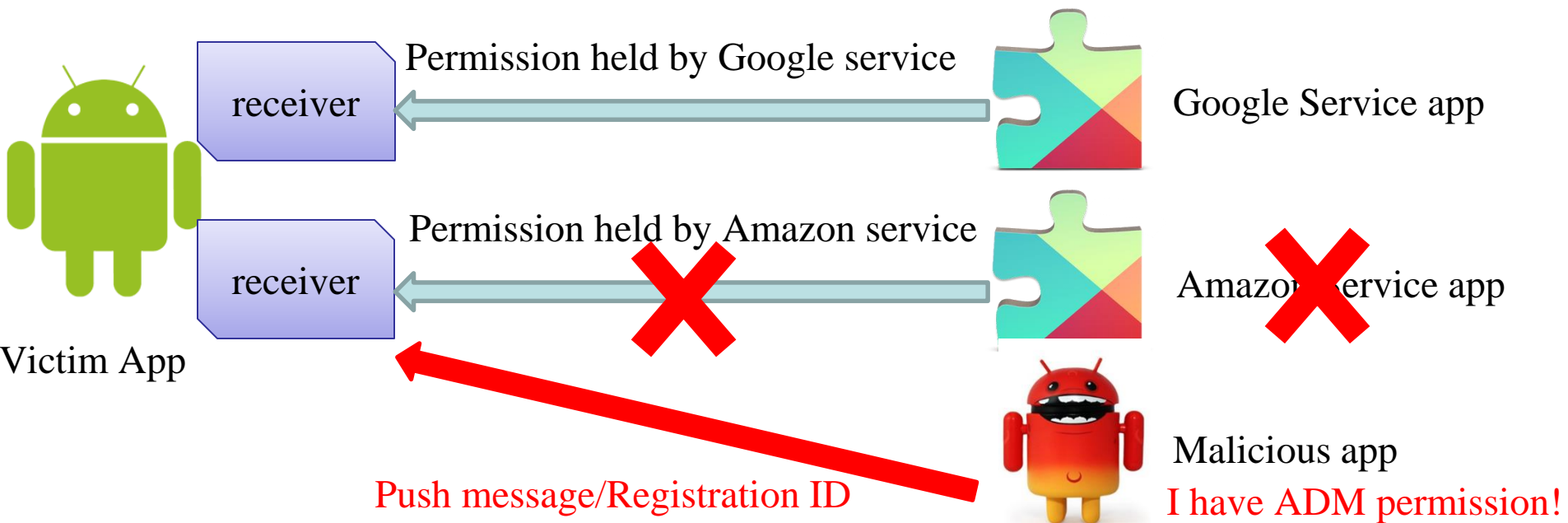


# New Attacks

- Service Confusion Problem
    - From property:
      - Secure channel
      - Receivers protected by permissions
    - Question:
      - Who has the permission?
        - Google's service(GCM), Amazon's service(ADM)
      - Any exception?
-

# New Attacks

- Service Confusion Problem
  - When apps/SDKs integrate multiple push-messaging services



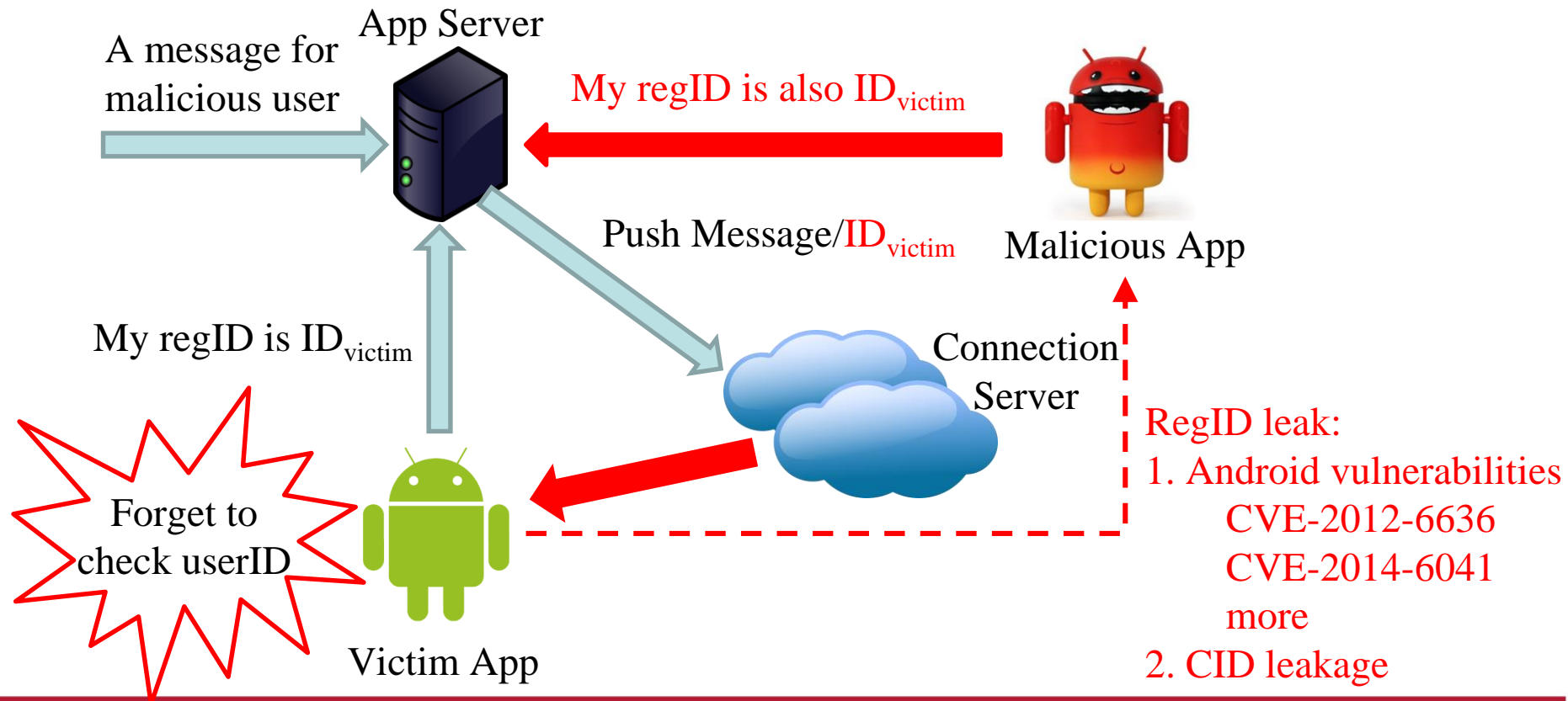
# New Attacks

- Service Confusion Problem
  - Impersonate push messaging service
    - Steal messages
    - Send fake messages
    - Facebook, Skype, UrbanAirship



# New Attacks

- User Confusion Problem
  - Push Messages are for the app, not for the user

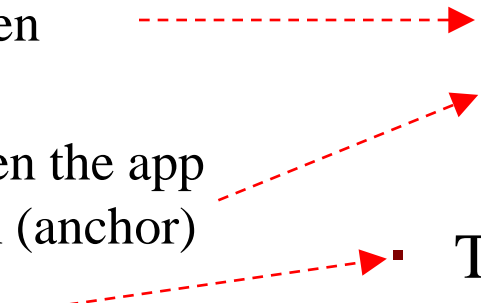




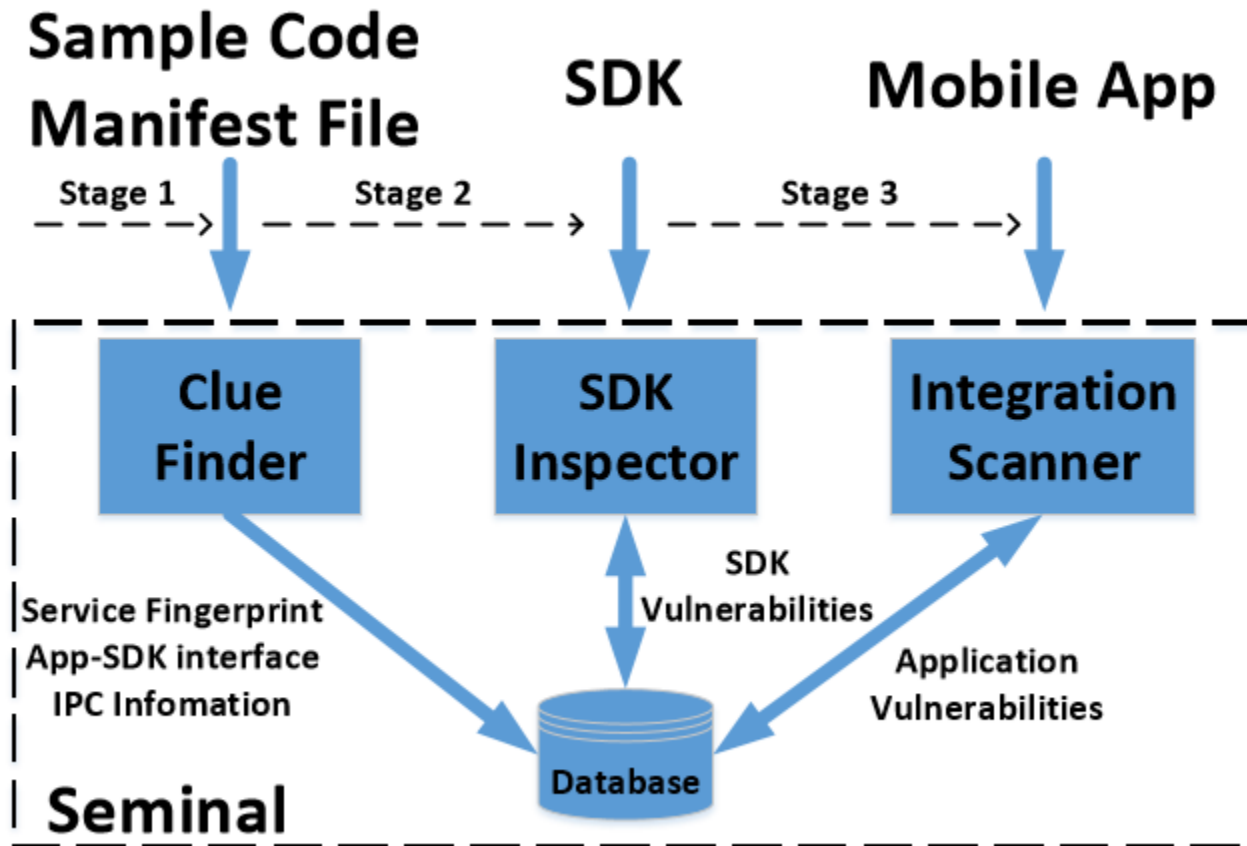
**In this demo, we demonstrate  
that a malicious app without  
any permission can inject  
messages to Facebook.**

---

# Challenges for Analysis

- It is less clear
    - What service has been integrated there
    - The interface between the app and the service SDK (anchor)
  - Large-scale analysis
  - Analyze demo code
    - Get fingerprints
    - Find anchor
  - Three stage analysis
    - Clue Finder
      - Analyze demo code
    - SDK Inspector
      - Analyze SDK
    - Integration Scanner
      - Analyze app
- 

# Seminal Architecture



# Clue Finding

- Analyze sample code
  - A few hundred lines with a manifest file
  - Look for service type, fingerprint, anchor

<b>Inputs</b>	Manifest File, Sample Code, Manufacturer Actions
<b>Service Type</b>	<i>Syndication</i> : service (not in sample code)+manufacturer action <i>Third-party</i> : otherwise
<b>Fingerprint</b>	Service name that is defined in SDK (not in sample code)
<b>Anchor</b>	Check sample code for: BroadcastReceiver, registerReceiver, Service (onHandleIntent) or Callback function

# Sample Code

## Manifest File Snippets:

```
<receiver android:name="com.pushio.manager.PushIOBroadcastReceiver"  
    android:permission="com.google.android.c2dm.permission.SEND">  
    <intent-filter>  
        <action android:name="com.google.android.c2dm.intent.RECEIVE" />  
        <action android:name="com.google.android.c2dm.intent.REGISTRATION" />  
        <category android:name="com.pushio.basic" />  
    </intent-filter>  
</receiver>  
...  
<service android:name="com.pushio.manager.PushIOGCMIntentService" />
```

Service Type

Fingerprint

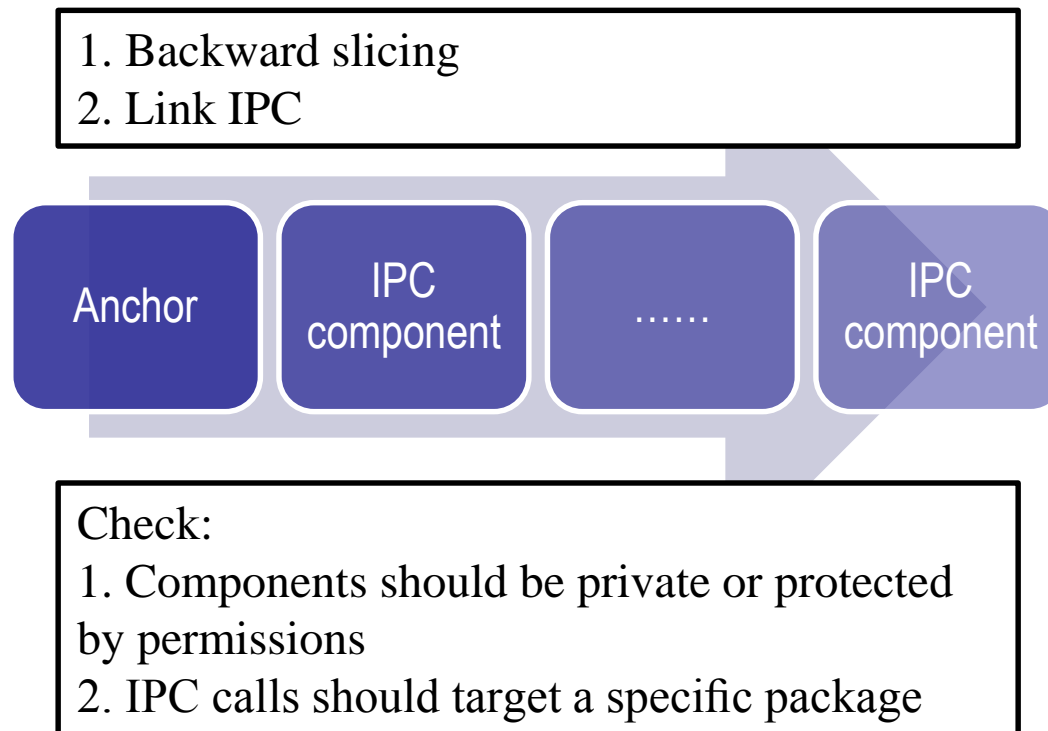
## Sample Code Snippets:

```
public class PushSettings extends Activity {  
    private BroadcastReceiver mBroadcastReceiver;  
    ...  
    @Override  
    public void onResume() {  
        super.onResume();  
        mBroadcastReceiver = new BroadcastReceiver() {  
            @Override  
            public void onReceive(Context context, Intent intent) {...}  
            registerReceiver(mBroadcastReceiver,  
                new IntentFilter("com.pushio.basic.PUSHIOPUSH")); } ...  
    }  
}
```

Anchor

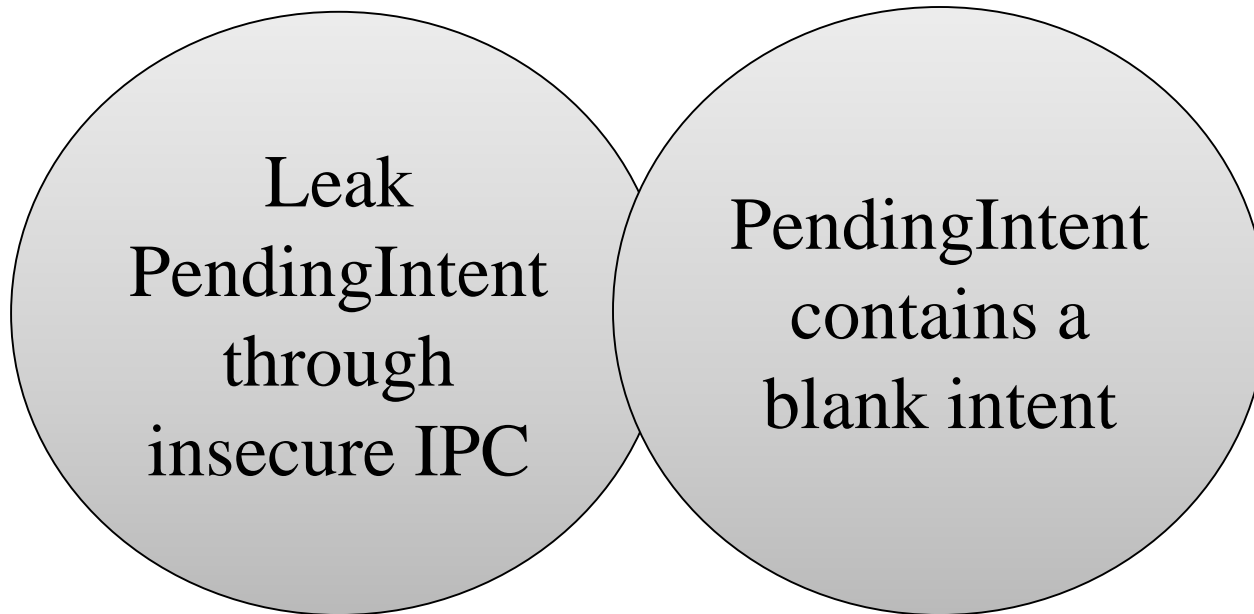
# SDK Analysis

- Insecure IPC



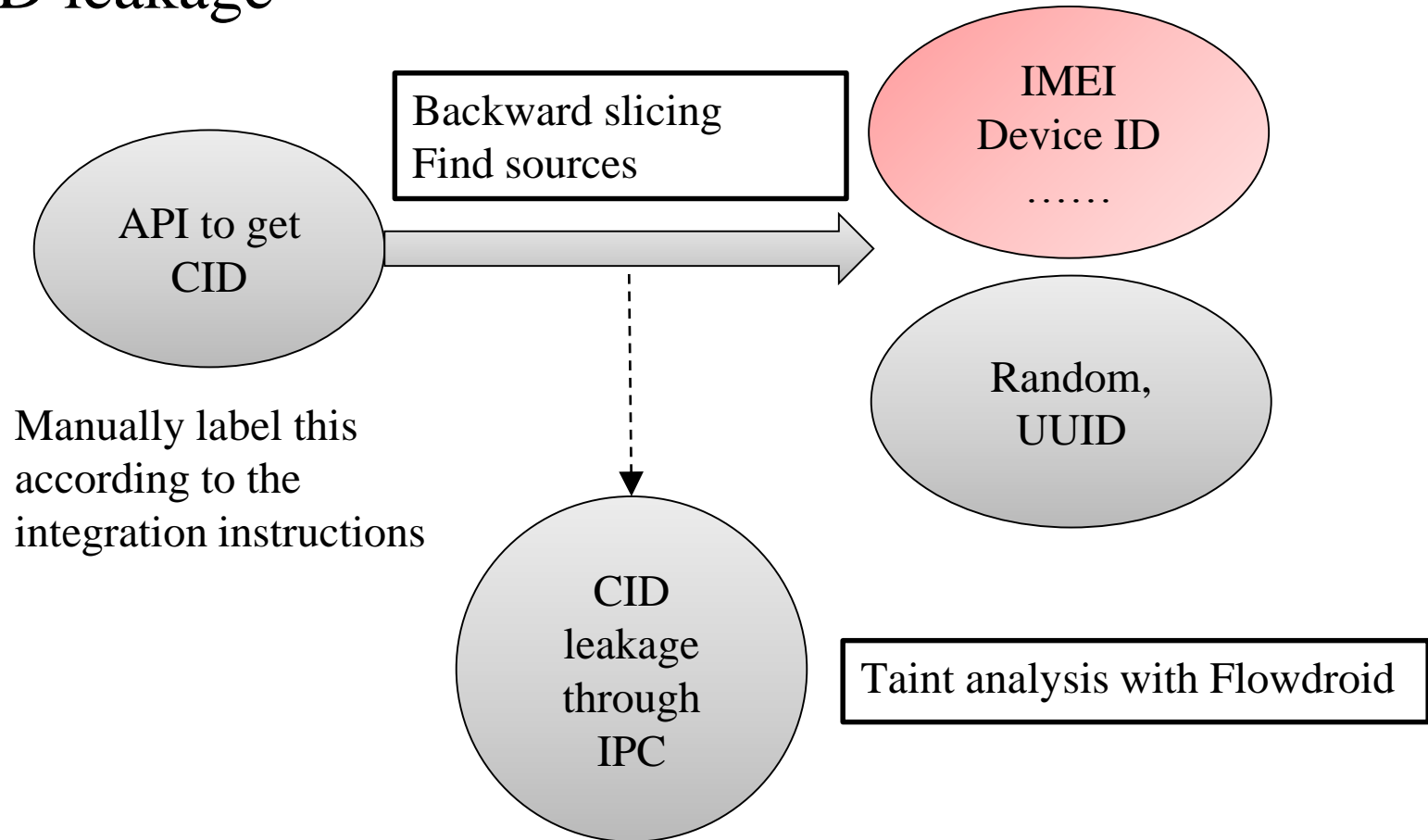
# SDK Analysis

- PendingIntent leakage



# SDK Analysis

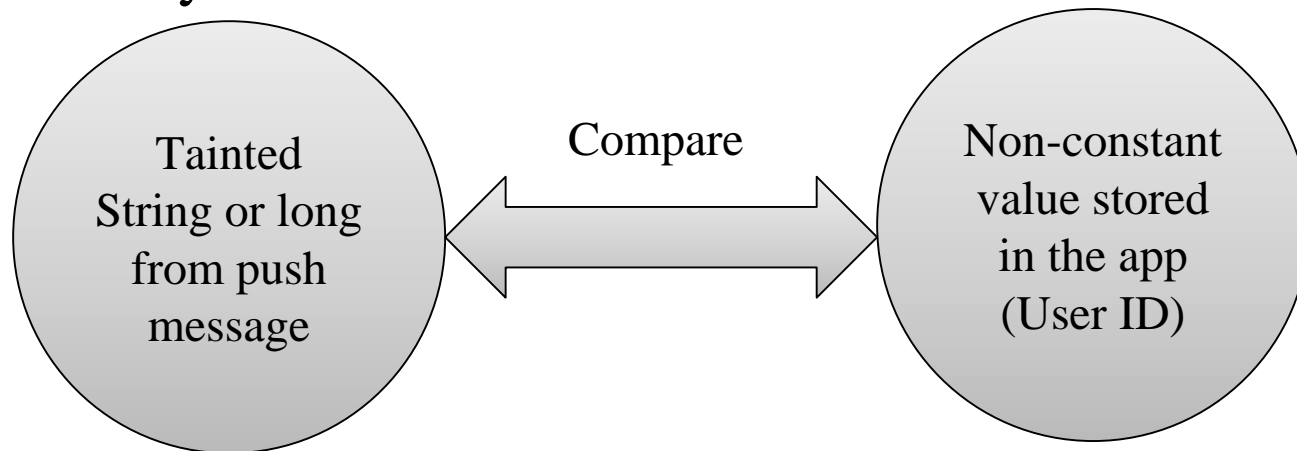
- CID leakage





# Integration-Specific App Checking

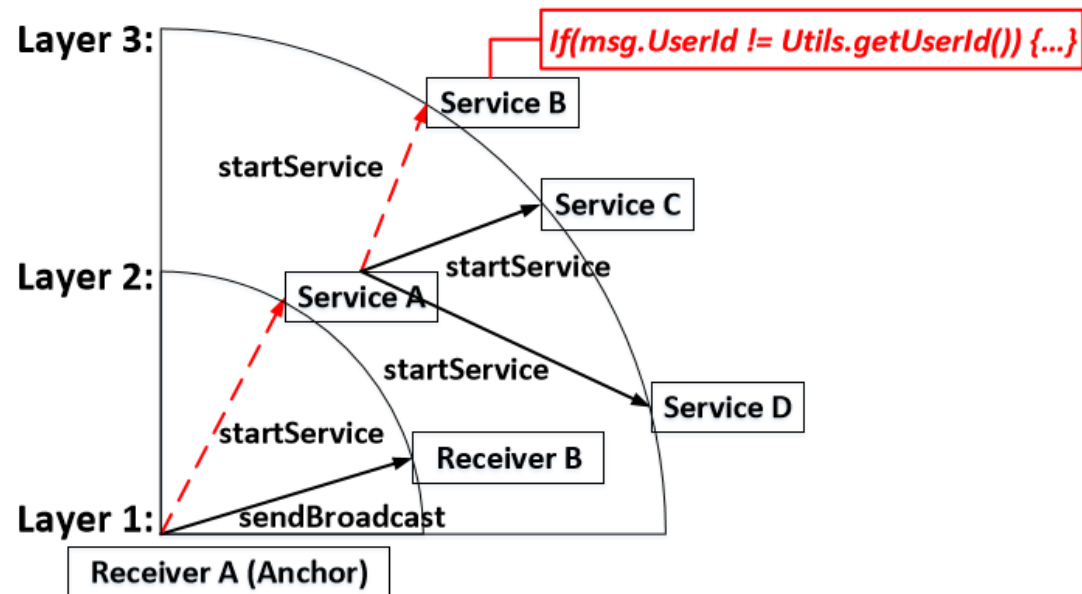
- User ID filtering (User Confusion)
  - Taint analysis



```
if (localVineSingleNotification.recipientUserId != this.mAppController.getActiveId())  
    SLog.e("This message is intended for someone else {}.\"",  
        Long.valueOf(localVineSingleNotification.recipientUserId));
```

# Integration-Specific App Checking

- User ID filtering (User Confusion)
  - A breadth-first, layered analysis
  - Taint analysis over partial CFG
  - Heuristic: 3 layers



# Measurement and Discovery

- SDK Risks
    - We discovered **17 (>50%)** security-critical flaws within **30** SDK
    - Always completed the analysis within 10 minutes
  
  - App Risks
    - Scanned **35,173** apps using push messaging
    - Find **26,069** potential problems in **17,668 (>50%)** apps
    - On average 108 seconds were spent on each app
-

Service	Type	Weaknesses
UrbanAirship	Syndication	Service Confusion
PushIO	Syndication	Insecure Broadcast Channel/CID Exposure/Service Confusion
Push Woosh	Syndication	Insecure Broadcast Channel/CID Exposure
Pushapps	Syndication	CID Exposure
Baidu	Third-Party	Insecure Broadcast Channel
Getui	Third-Party	Insecure Broadcast Channel
Xiaomi	Third-Party	Insecure Broadcast Channel
XG Push	Third-Party	Insecure Broadcast Channel/CID Exposure
Bmob	Third-Party	Insecure Broadcast Channel
Yunba	Third-Party	Insecure Broadcast Channel
Zhiyou	Third-Party	Insecure Broadcast Channel
Mpush	Third-Party	Insecure Broadcast Channel/CID Exposure
LeanCloud	Third-Party	Insecure Broadcast Channel
Umeng	Third-Party	Insecure Broadcast Channel/CID Exposure(risk)
JPush	Third-Party	Insecure Broadcast Channel
ShengdaPush	Third-Party	CID Exposure
Huawei	Third-Party	Insecure Broadcast Channel

# Vulnerable Popular Apps

App	Downloads	Vulnerability Type	Sample Contents at Risk
Facebook	500M+	Service Confusion	Messages
Skype	100M+	Service Confusion	Messages
Pinterest	10M+	User Confusion	Messages
Yelp	10M+	User Confusion	Messages
Linkedin	10M+	PendingIntent	Invitation, Messages
eBay	50M+	PendingIntent	Shipment, Messages

# Conclusion

- Identified a set of security principles and properties
  - Seminal
    - Automatic analysis tool for push messaging services and apps integrating them
    - Sample code based, three stages
  - Large-scale scan
    - 30 SDKs and more than 30,000 apps
    - many risks discovered
  - New attacks
    - Service confusion, User confusion and so on
-

**Thanks!**

---