Invetter: Locating Insecure Input Validations in Android Services

Lei Zhang[+], Zhemin Yang[+], Yuyu He[+], ZhenYu Zhang[+],

Zhiyun Qian[*], Geng Hong[+], Yuan Zhang[+] and Min Yang[+]

+ Fudan University * University of California Riverside

关于我

- 张磊
- lei_zhang14@fudan.edu.cn
- https://zzzxxxlll.github.io
  - 复旦大学博士生
  - 导师：杨珉
  - 研究方向：
    - 漏洞挖掘
  - 最关心的问题
    - 0 DAY你在哪儿
  - 目前战绩
    - … …

移动安全

区块链安全

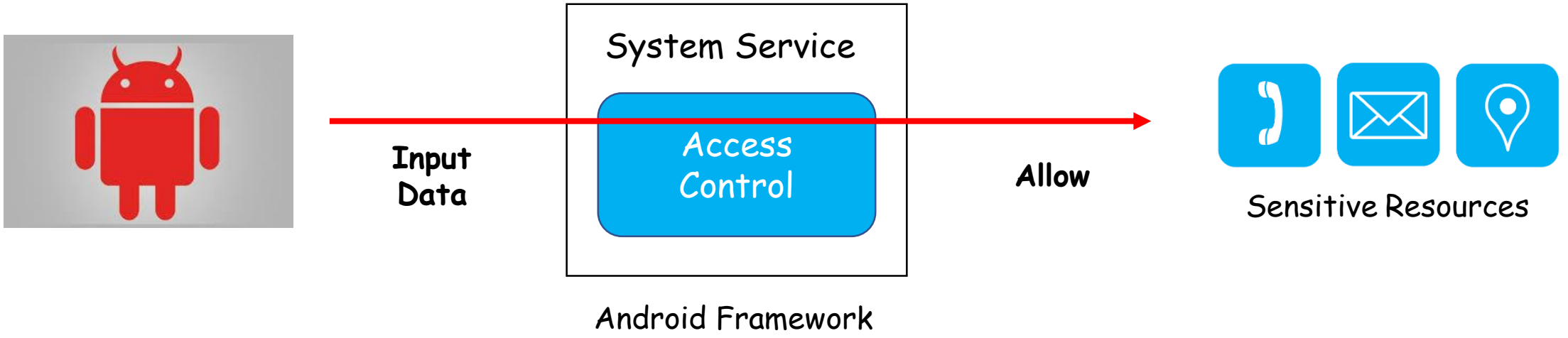
System Services of Android

- Perform sensitive operations
  - Window service, Location service, Notification service, etc.

Motivation

Android enforces a set of access controls to manage sensitive resources

Security flaws in the access controls make them vulnerable to illegal access



State of the art

- Permission validation is well studied
  - Kratos: Discovering Inconsistent Security Policy Enforcement in the Android Framework. NDSS (2016).
  - Harvesting Inconsistent Security Configurations in Customized Android Roms via Differential Analysis. USENIX (2016).
  - AceDroid: Normalizing Diverse Android Access Control Checks for Inconsistency Detection. NDSS(2018)
  - … …

However, an unstudied research area :

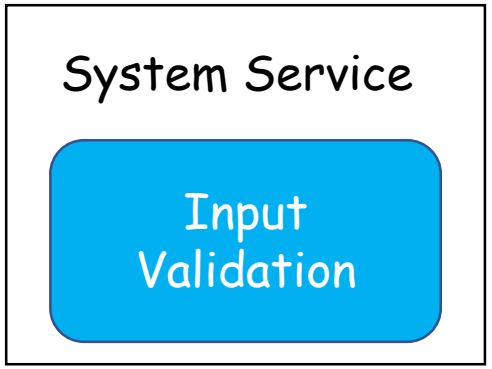
Input validation of Android services ?

Android Input Validation

```
if(inputData == "android"){
        //do sensitive operation
} else {
   throw new SecurityException("…");
}
```

```
if(isBlackListed(inputData)){
    throw new
        SecurityException(" …");
}
```

```
if(checkCaller(inputData)){
        return Mode_Allowed;
}
```



Android Framework

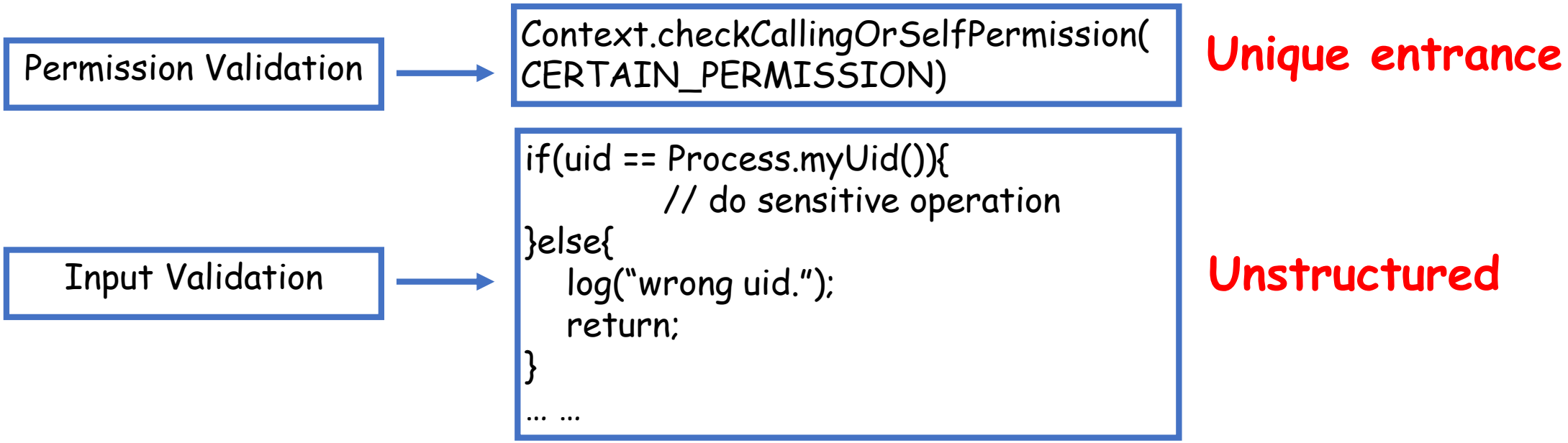
```
if(!checkPolicy(inputData)){
      recycle();
      return;
}
```

```
if(isSystemApp(inputData)){
    log("illegal access");
     return;
}
```

- Our work
  - The first study of the secure use of **sensitive input validations** in Android framework

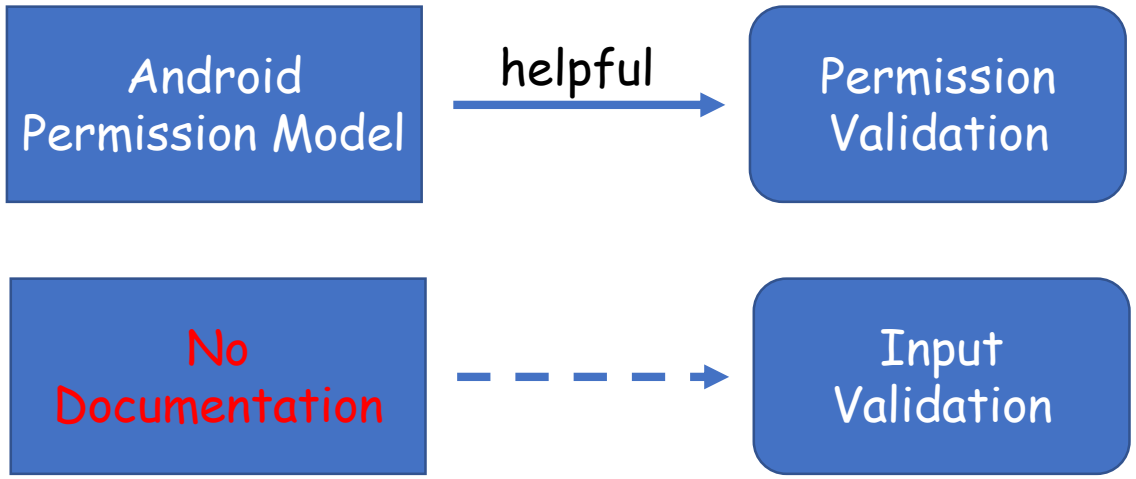
Locating Sensitive Input Validation is Difficult

- Unstructured
  - Any parameter may lead to a sensitive input validation
  - Various text strings in parameter



Locating Sensitive Input Validation is Difficult

- Ill-defined
  - No documentation
  - No system interface



Locating Sensitive Input Validation is Difficult

- Fragmented
  - Any system service (Java class)
  - Any public interface
  - Any branch statement

  - In Android 7.0,

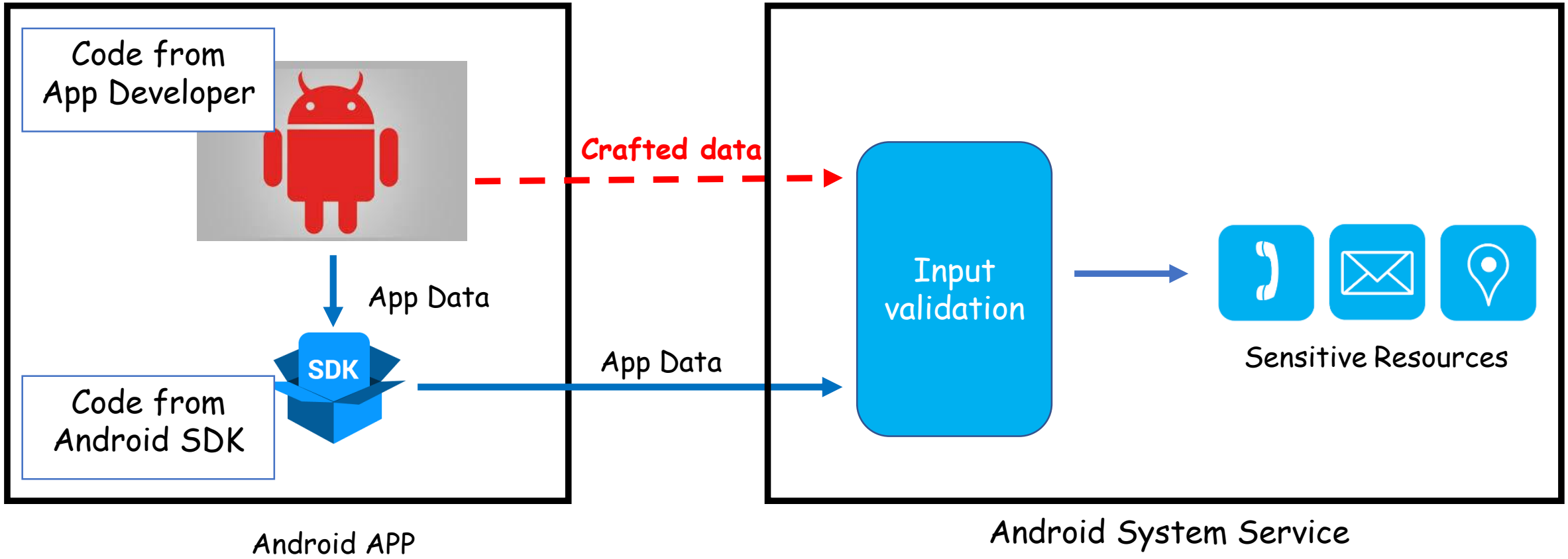
    6 Java classes for permission validation

    vs

    173 Java classes for input validation

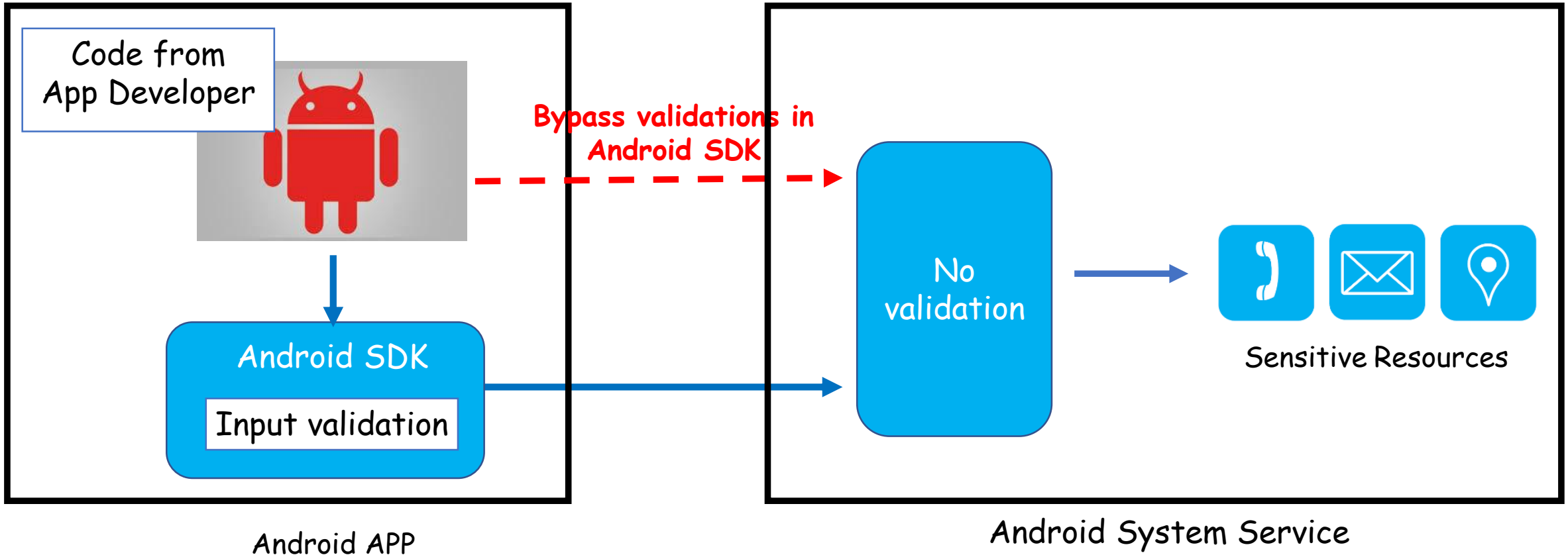
Input Validation can be Vulnerable

- Confusions about system security model
  - Incorrectly trusting app supplied data



Input Validation can be Vulnerable

- Confusions about system security model
  - Incorrectly trusting code in the app process



Input Validation can be Vulnerable

- Weakened validations in customization

```
if( wouldeToggleZenMode( ringerMode)&&
checkCallerIsSystemOrSamePackage(call
er)&& checkAccessPolicy( caller))
 {
    throw new SecurityException(...)
 }

setRingerMode( ringerMode, caller ...)
.....
```

AOSP image
(version 6.0)

Customization

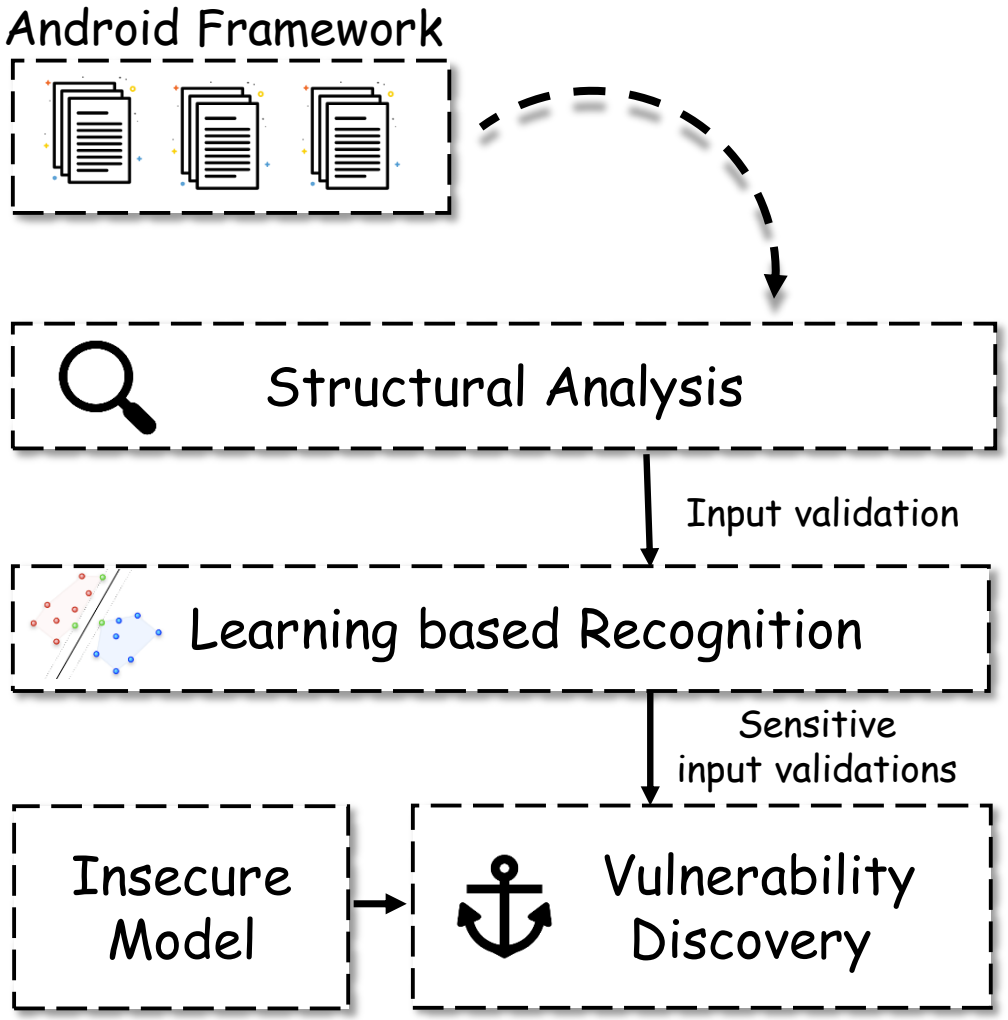
NO CHECK

setRingerMode(ringerMode, caller ...)
.....

3rd party image

Methodology Design

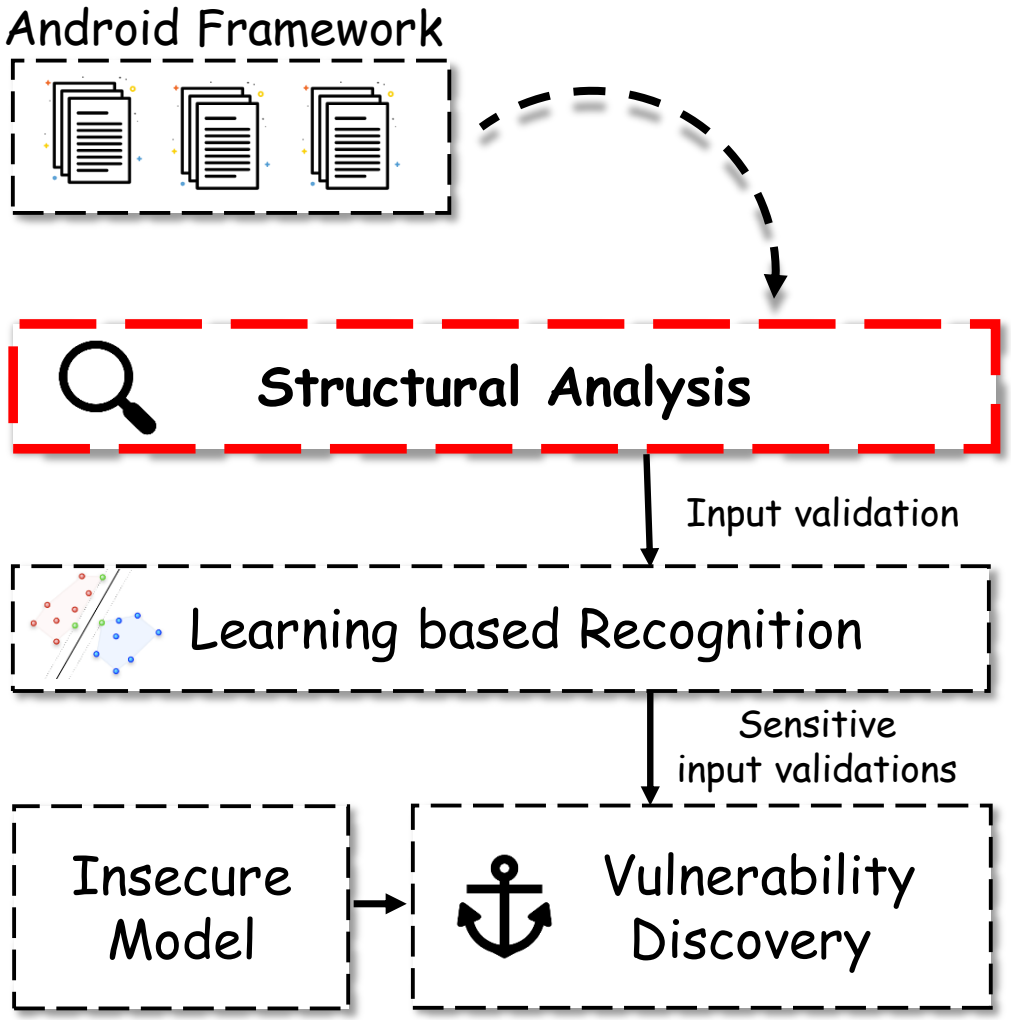
```
## SystemServiceExample.java
public void onTransact(input ... ){
      if( ! isSystemPackageName(input.packageName) )
                throw new SecurityException("Not a system app");
      if( ! isValideMode(input.type){
                log("illegal type: "+input.type);
                return;
      }
      //sensitive operation
      //lots of code snippets
       ... ...
      if( ! checkData(input.data){
                warn("wrong data : "+input.data);
                return;
      }
      ... ...
}
```



Methodology Design

```
## SystemServiceExample.java
public void onTransact(input ... ){
        if( ! isSystemPackageName(input.packageName) )
                throw new SecurityException("Not a system app");
        if( ! isValideMode(input.type){
                log("illegal type: "+input.type);
                return;
        }
        //sensitive operation
        //lots of code snippets
        ... ...
        if( ! checkData(input.data){
                warn("wrong data : "+input.data);
                return;
        }
        ... ...
}
```

input validation



Structural Analysis

- Recognize input validation
  - Key Insight : Input validation terminates its normal execution when the validation fails
  - Termination branch

➢ Throw exception	✓ Throw new SecurityException(…)
➢ Return constant	✓ Return Mode_Allowed
➢ Log and return	✓ Log.warn(…)
➢ Recycle and Return	✓ Recycle()

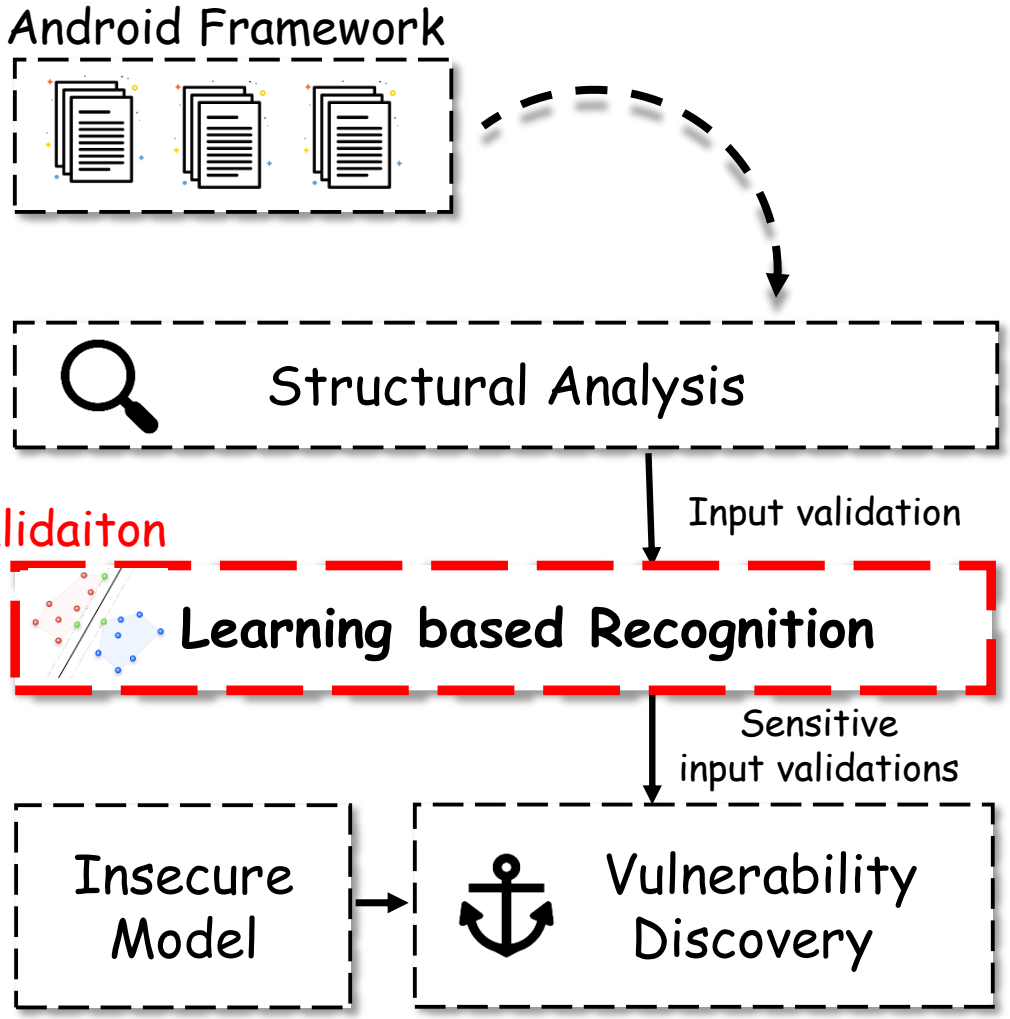
Methodology Design

```
## SystemServiceExample.java
public void onTransact(input ... ){
    if( ! isSystemPackageName(input.packageName) )
            throw new SecurityException("Not a system app");
    if( ! isValideMode(input.type){
            log("illegal type: "+input.type);
            return;
    }
    //sensitive operation
    //lots of code snippets
    ... ...
    if( ! checkData(input.data){
            warn("wrong data : "+input.data);
            return;
    }
    ... ...
}
```

Known sensitive input validation

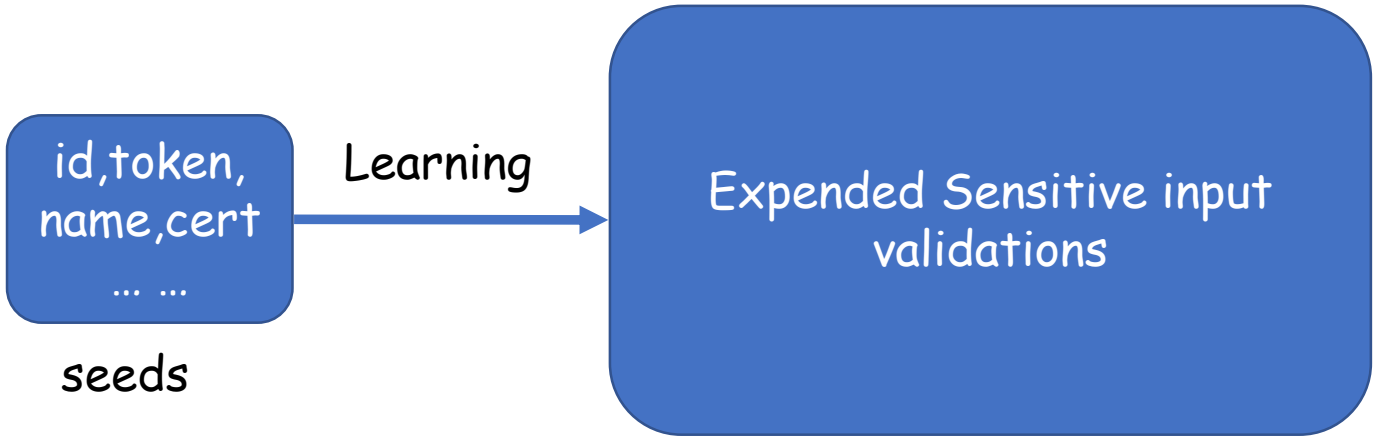
Adjacent input validaiton

Non-adjacent input validation will be ignored



Learning Based Recognition

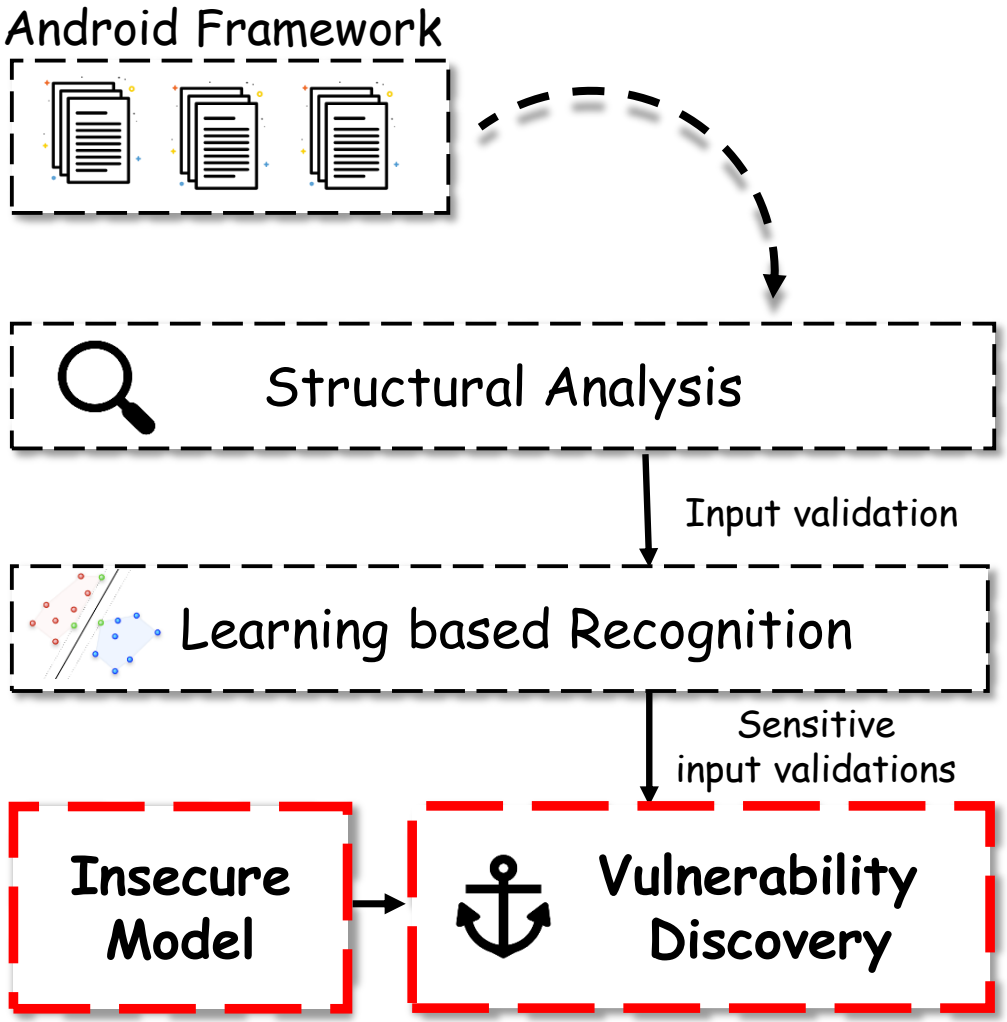
- Recognize sensitive input validation
  - Key Insight : Sensitive input validations are likely to be verified together
  - Association rule mining
    - If one input validation is adjacent to a sensitive input validation, it is sensitive with a high possibility
    - **Known** sensitive input validations -> **unknown** sensitive input validations



Methodology Design

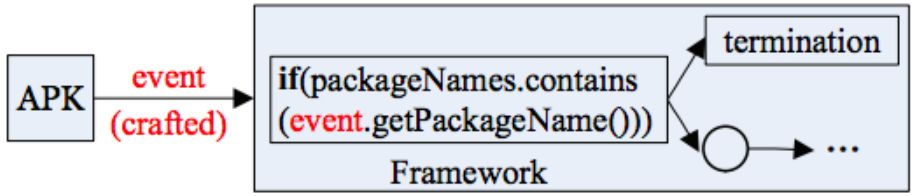
```
## SystemServiceExample.java
public void onTransact(input ... ){
    if( ! isSystemPackageName(input.packageName) )
            throw new SecurityException("Not a sy
    if( ! isValideMode(input.type){
            log("illegal type: "+input.type);
            return;
    }
    //sensitive operation
    //lots of code snippets
     ... ...
    if( ! checkData(input.data){
            warn("wrong data : "+input.data);
            return;
    }
    ... ...
}
```

Use input as caller identity

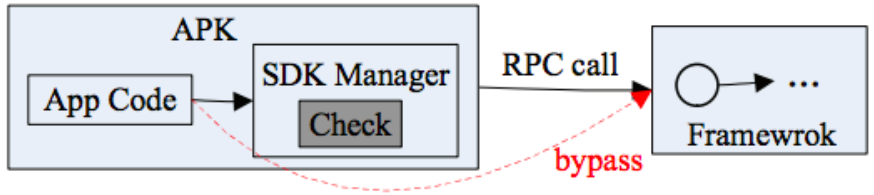


Vulnerability Discovery

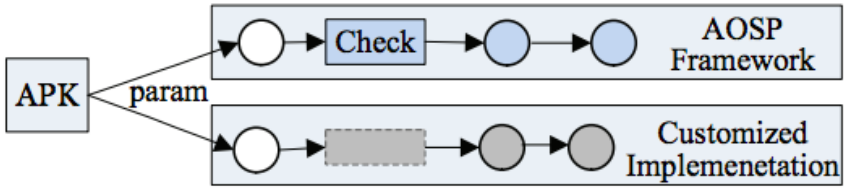
- Insecure model
  - Rule a : Incorrectly trusting app-supplied data
    - App Input -> caller identity
  - Rule b : Incorrectly trusting code in the app process
    - Input validation in Android SDK and not in Android Service
  - Rule c : Weaken validations in customized images
    - Inconsistent validations



(a) Incorrectly trusting app-supplied data



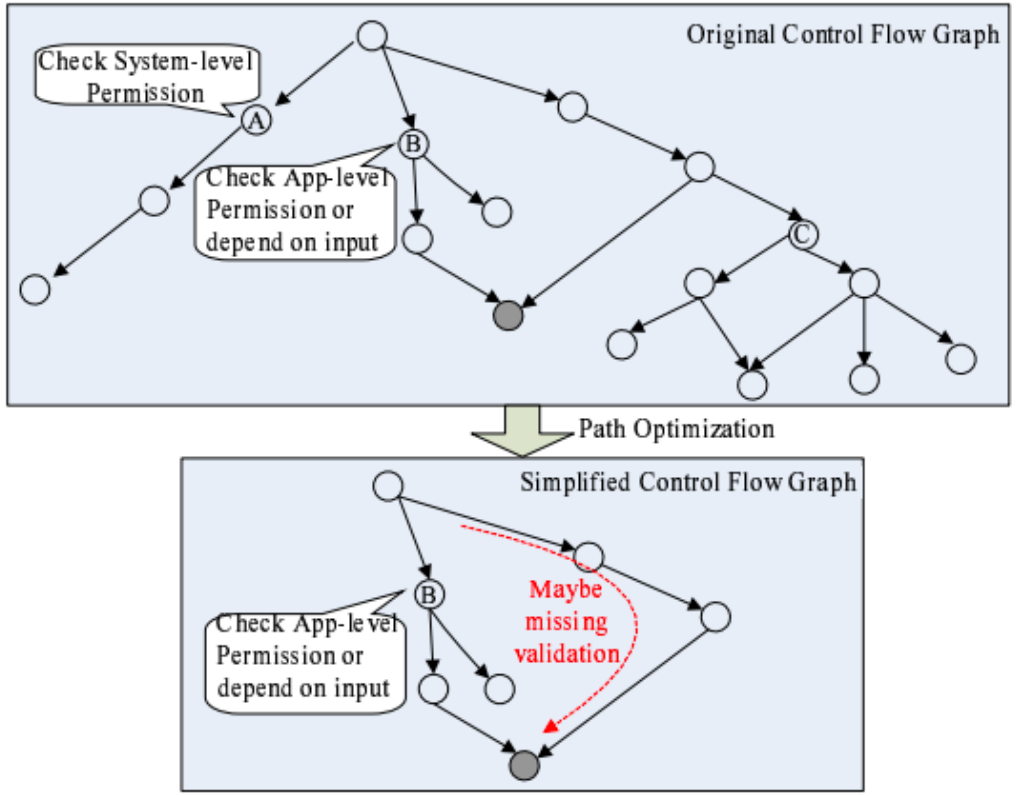
(b) Incorrectly trusting code in the app process



(c) Weakened sensitive input validation in customized images

Vulnerability Discovery

- Static Analysis
  - Implemented on Soot, with about 12,000 lines of Java code
  - Java byte-code level
  - Optimization for Path sensitive analysis
    - Reduce irrelevant nodes
    - Reduce privileged nodes



  - Github :
    - https://github.com/zzzxxxlll/Invetter-os

Evaluation

- Overall
  - Statistics about analysis target (on average)
    - 100+ system services
    - 15,000+ java classes
    - 200,000+ if statements
  - CentOS 7 server, with four 8-core 2.0GHz CPUs and 192 GB memory
- Data set
  - 4 AOSP images
    - AOSP 7.1, 7.0, 6.0, 5.0, (AOSP 8.0)
  - 4 customized images
    - Xiaomi Note2 (6.0), Huawei P9(7.0), Huawei Mate9(7.0), Samsung S6(5.0), (Xiaomi Mix2(8.0), Huawei P10(8.0))

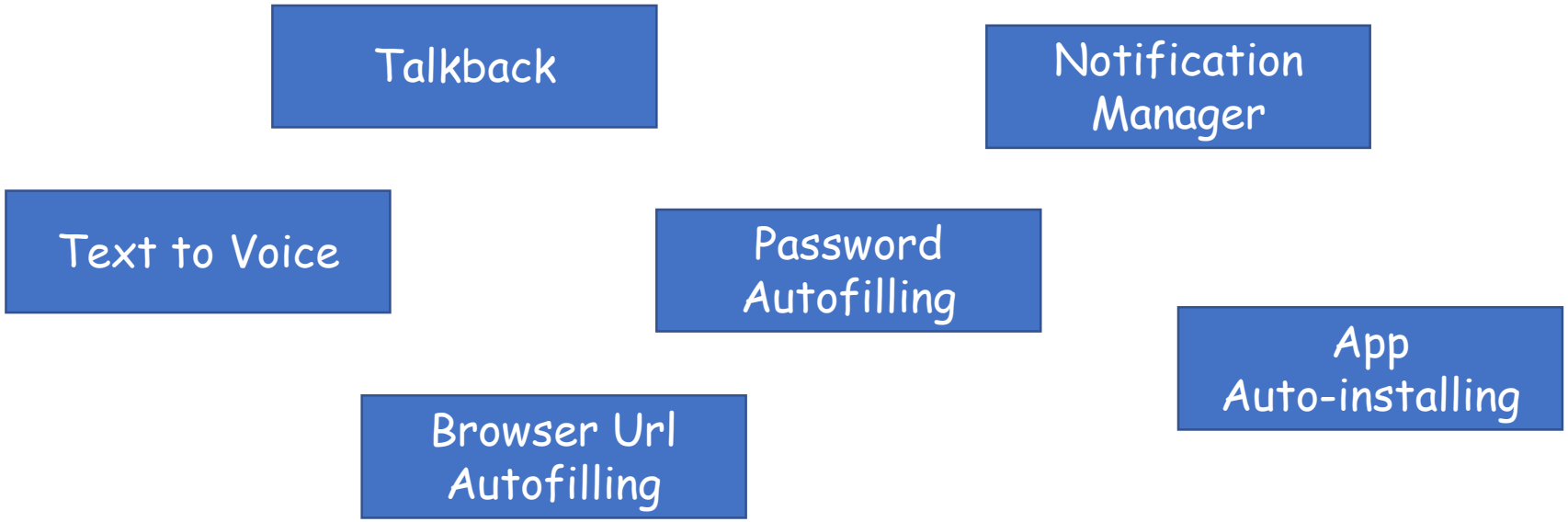
Result

- Performance
  - 11.8 hours for 8 images
  - 85 minutes on average
- Precision
  - 103 possible insecure input validations
  - TP = 86
  - Exploitable = 20
- Real World Attacks
  - Privilege escalation = 17
  - Log overflow = 1
  - Privacy leakage = 2

  - Demos are illustrated in the online video :
    - https://youtu.be/erLY_OMi4kQ.

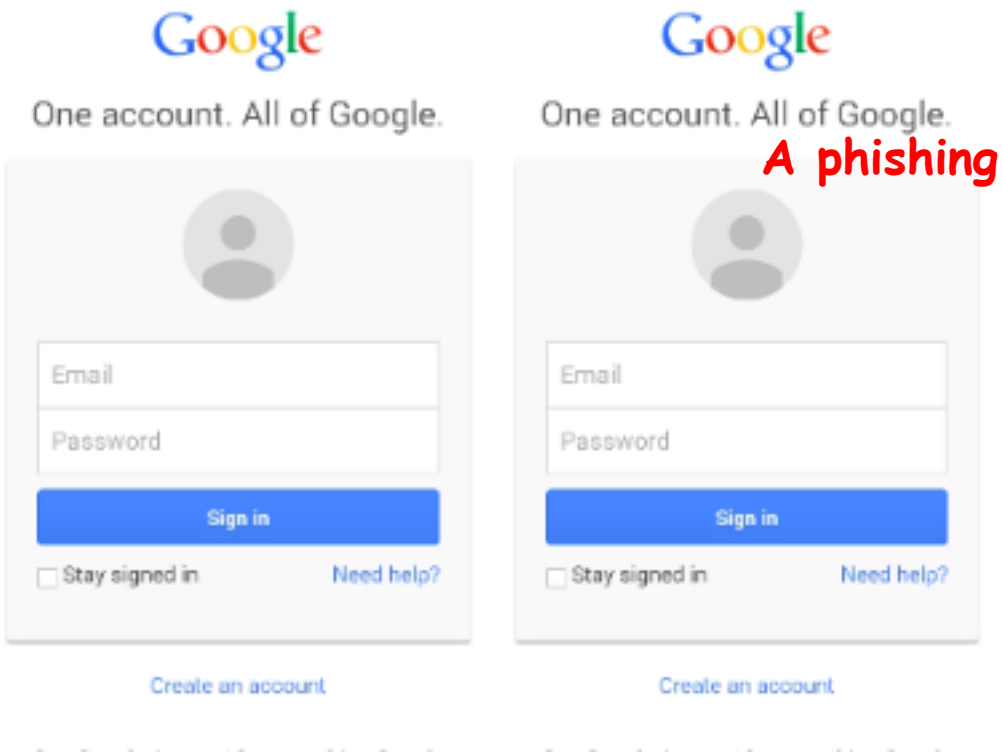
Case Study 1

- Sending arbitrary accessibility event
  - AccessibilityManagerService
  - Use app input to verify caller identity
  - Any app can craft an event with a faked package name



Case Study 2

- Stealthy phishing attack
  - WindowManagerService
  - Inconsistent validations in two execution paths
  - Allow a malware to create crafted Toast message with arbitrary scope of view space



A phishing window

Conclusion

- Invetter
  - A novel technique for identifying insecure input validations
  - Extend scope for finding more authorization vulnerabilities in Android Framework

- Evaluation on 8 Android images in the wild
  - 20 Input-based vulnerabilities in Android system services
  - Weakened access controls in Android customization

超越

再前进一步就是超越

快到碗里来！

- **复旦大学系统与软件安全实验室**
- 导师：杨珉 （教授，博导，国家973项目首席科学家，长江学者）
  - m_yang@fudan.edu.cn
- 实验室研究方向：
  - 移动安全
  - 区块链安全
  - 网络安全
  - 二进制安全
- 实验室战队：
  - CTF战队：白泽

Thanks !

Q&A

- lei_zhang14@fudan.edu.cn
- m_yang@fudan.edu.cn
- yangzhemin@fudan.edu.cn