移动平台隐私数据保护

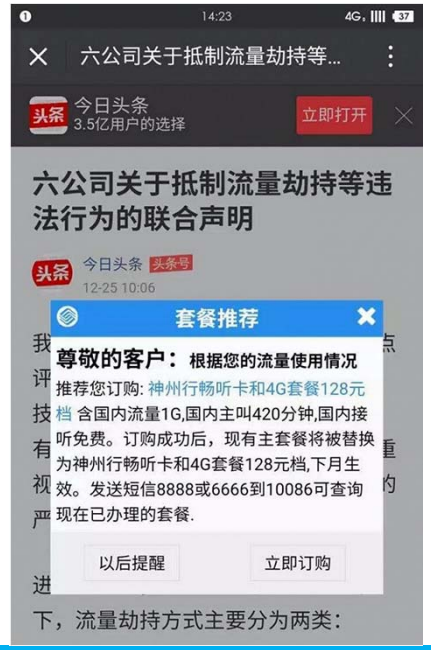
杨 珉

复旦大学

移动生态系统的安全态势

- 好莱坞明星艳照门：苹果iCloud云盘系统遭入侵
  - 2014年9月-10月，101位国际巨星
  - “Find My iPhone”服务器端的多个高危漏洞

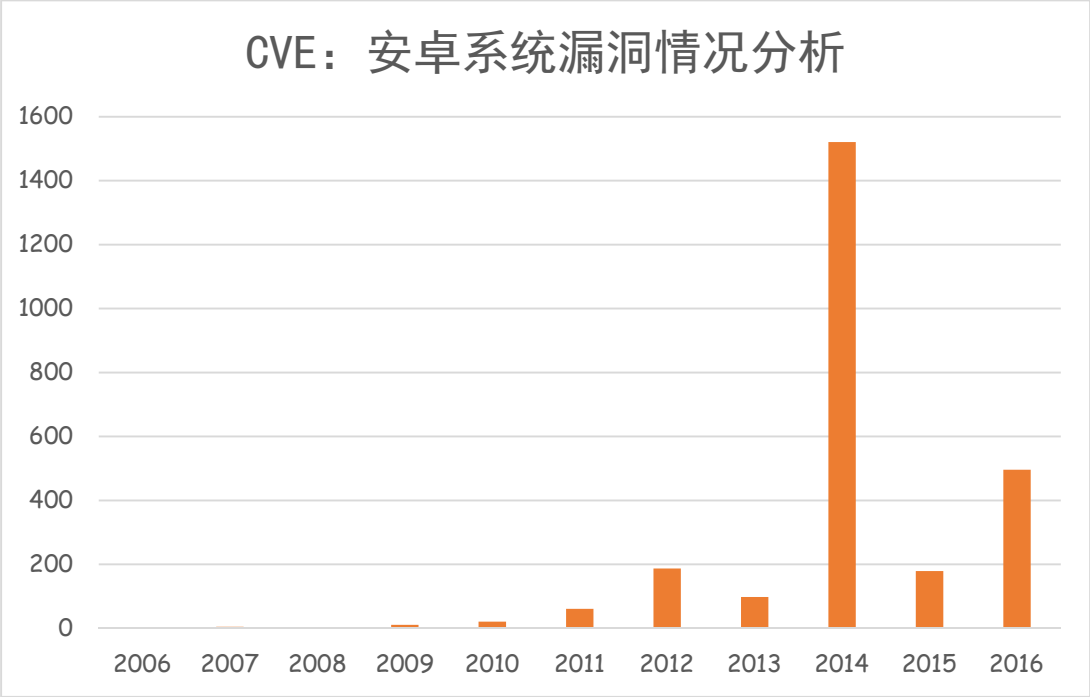
- 国产安卓手机成“肉鸡”：百度 WormHole 事件
  - 2015年11月，涉及所有国产安卓手机
  - 百度moplus SDK包含可以远程控制的“后门”
  - 移动恶意软件首次出现“蠕虫式”自主传播



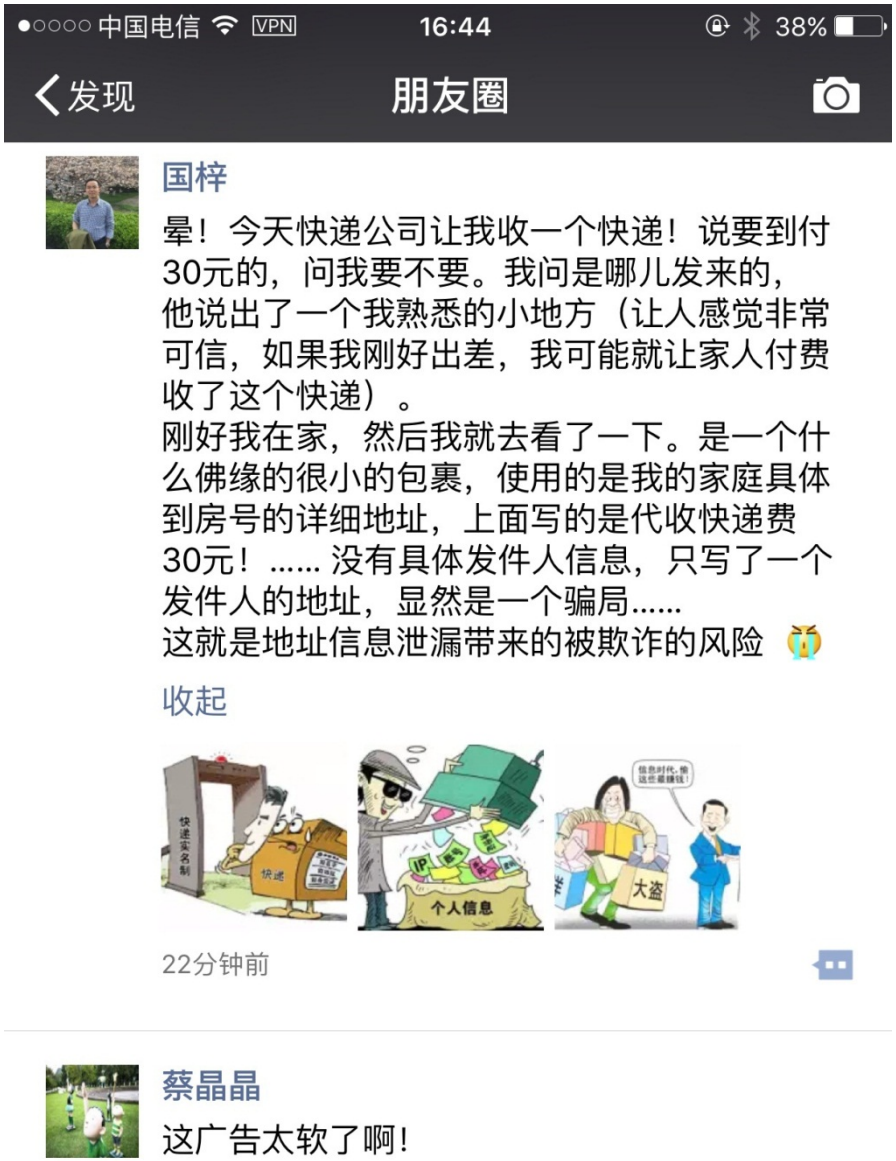
- 腾讯、新浪等六公司联合抵制移动流量注入
  - 2015年12月，三大运营商和黑色产业

移动生态系统的安全态势

- 大半个美国的网络服务瘫痪数小时
  - 2016年10月21日，数百万个网站无法访问
  - Mirai僵尸网络：大量IoT设备被远程控制
  - 对美国DNS服务商Dyn的DDoS攻击
- 移动系统漏洞的恶意利用导致一系列生态安全灾难



我的隐私数据随便拿？


●○○○○ 中国电信 VPN 16:44 38%
发现
朋友圈
国梓
晕！今天快递公司让我收一个快递！说要到付30元的，问我要不要。我问是哪儿发来的，他说出了一个我熟悉的小地方（让人感觉非常可信，如果我刚好出差，我可能就让家人付费收了这个快递）。
刚好我在家，然后我就去看了一下。是一个什么佛缘的很小的包裹，使用的是我的家庭具体到房号的详细地址，上面写的是代收快递费30元！..... 没有具体发件人信息，只写了一个发件人的地址，显然是一个骗局.....
这就是地址信息泄漏带来的被欺诈的风险
收起
快递实名制
快递
个人信息
大盗
22分钟前
蔡晶晶
这广告太软了啊！

老法师也中招

- 吕述望 教授，博士生导师
  - 中科院信工所，信息安全国家重点实验室

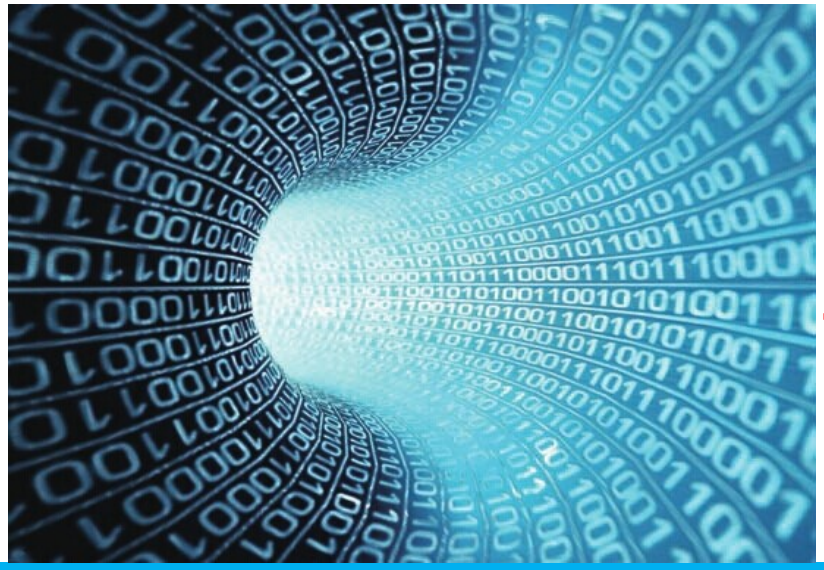
老吕，刚才你给我发过这个短信息吗？“严明，我是吕述望，这是我帮你拍的小视屏 df.tc/sgND9J”。我看着可疑，没有打开它。

Science: The End of Privacy

- 超学科的研究范畴
  - Unmasked
  - When your voice betrays you
  - Breach of trust
  - Risk of exposure
  - Could your pacemaker be hackable?
  - Trust me, I'm a medical researcher
  - Control use of data to protect privacy
  - Privacy and human behavior in the age of information
  - Balancing privacy versus accuracy in research protocols
  - What the "right to be forgotten" means for privacy in a digital age
  - ……

移动互联：泛在智能的平行空间

实体空间



网络空间

江湖：地下组织


漏洞挖掘者
僵尸网络集团
病毒
1 漏洞信息
5 肉鸡控制权
6．地下交易
——数百万肉鸡控制
感染终端
资金流向
入侵活动
病毒制造者
2、病毒
牧马人
钓鱼攻击
恶意软件
肉鸡中间商
网络设备中间商
终端玩家
盗窃
信用卡受害人
信用卡犯罪团伙
3. 后门/僵尸网络
4 肉鸡控制权
勒索
商业机密泄露公司
竞争对手
广告商
7 盗号木马
商业机密隐私
控制命令
帐号密码
计算能力
隐私资料
被盗窃后在黑市交易
被在黑市标价出租
被盗窃-勒索或泄露
商业机密被盗
隐私活动被盗窃
蠕虫
DDOS攻击
网络木马
钓鱼网站
更多受害者

移动系统安全

网络空间的不对称博弈

如影随形的高价值数据

系统安全关注现实威胁

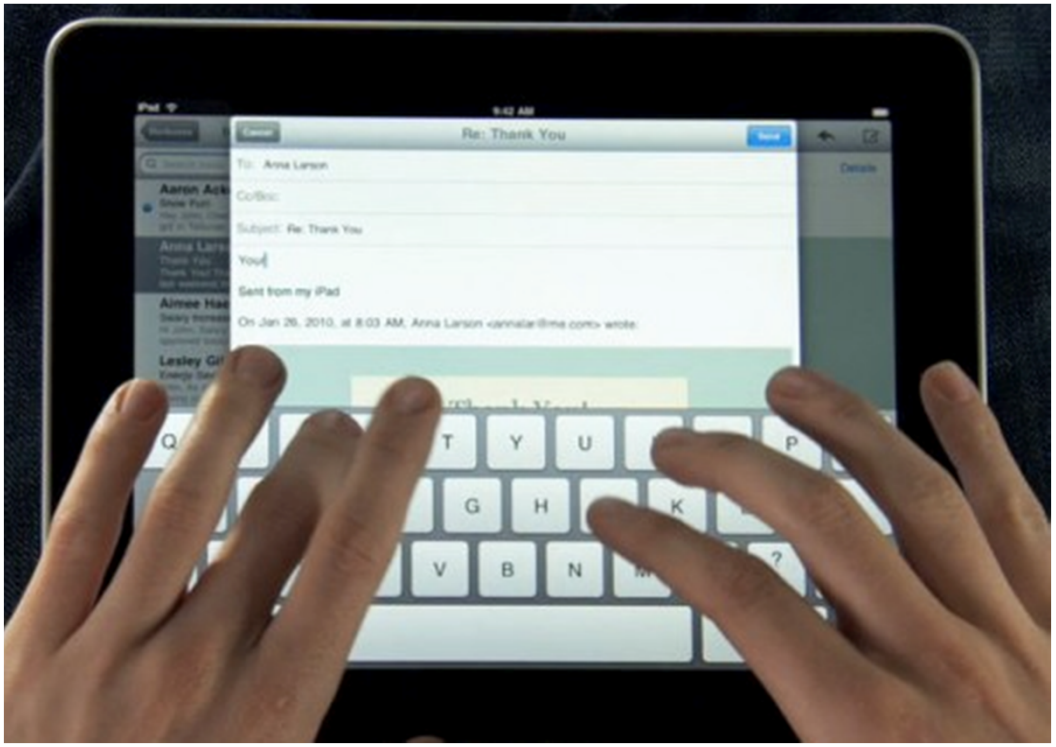
- 安全攻防
- 体系结构
- 操作系统
- 编译工具链
- 程序分析等

四大安全顶会为欧美学者垄断

移动系统安全是网络空间安全的重要环节，严重滞后于攻击技术发展。

移动平台隐私威胁新特点

- 移动平台上，用户输入的信息是隐私数据的主要来源



1. *UIPicker: User-Input Privacy Identification in Mobile Applications*, 25th USENIX SECURITY 2015
2. *Identifying User-Input Privacy in Mobile Applications at a Large Scale*, IEEE Trans. On Information Forensics & Security, 2016

针对用户输入隐私的攻击开始兴起

图 1 正常应用的启动界面

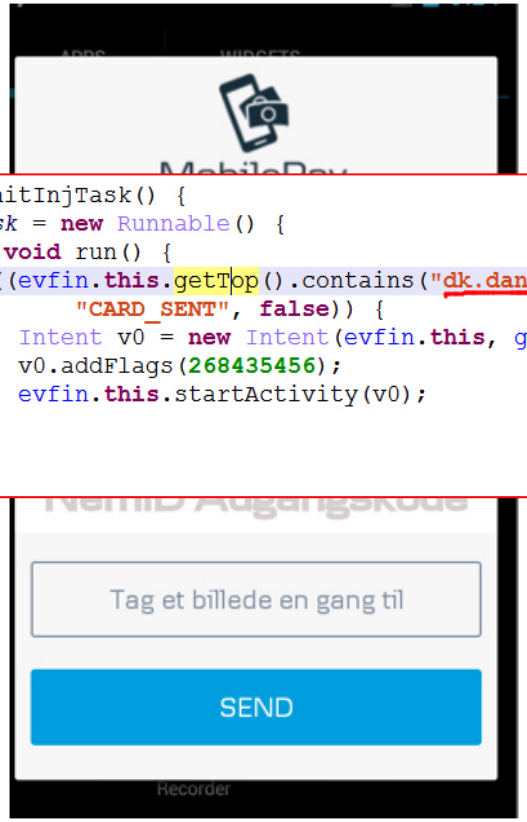
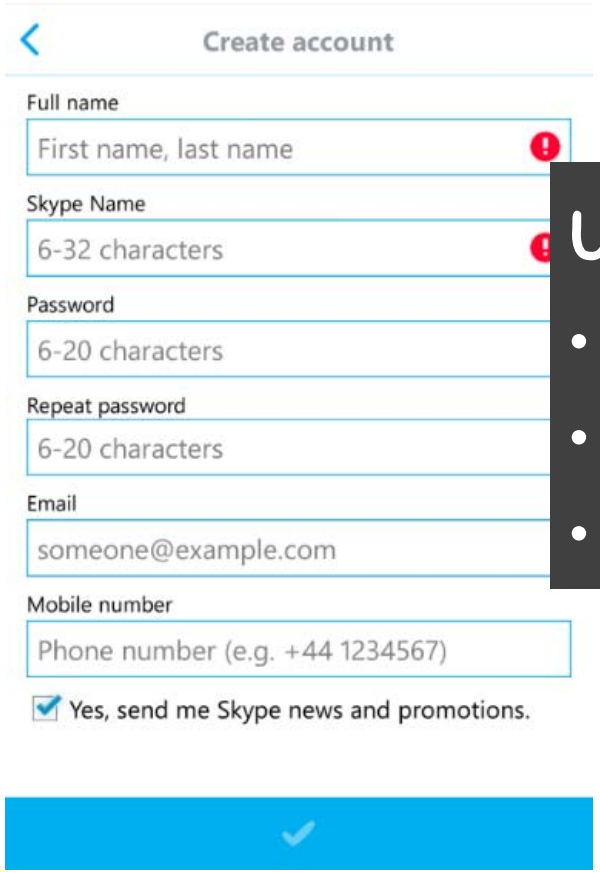


图 2 SlyRogue 病毒的仿冒界面

```
private void initInjTask() {
    this.injTask = new Runnable() {
        public void run() {
            if((evfin.this.getTop().contains("dk.danskebank.mobilepay")) && !evfin.settings.getBoolean(
                    "CARD_SENT", false)) {
                Intent v0 = new Intent(evfin.this, gmkxrny.class);
                v0.addFlags(268435456);
                evfin.this.startActivity(v0);
            }
        }
    };
```

来自AVL Team. 高能预警：丹麦“支付宝”MobilePay被盯上了！
http://blog.avlsec.com/2016/06/3339/mobilepay/

用户输入的隐私（UIP）数据



账号和密码信息

地址信息

资金账户信息

UIP数据特点

- UIP类数据格式各异
- UIP类数据获取接口非规整
- UIP类数据的获取形式多样

现有工作

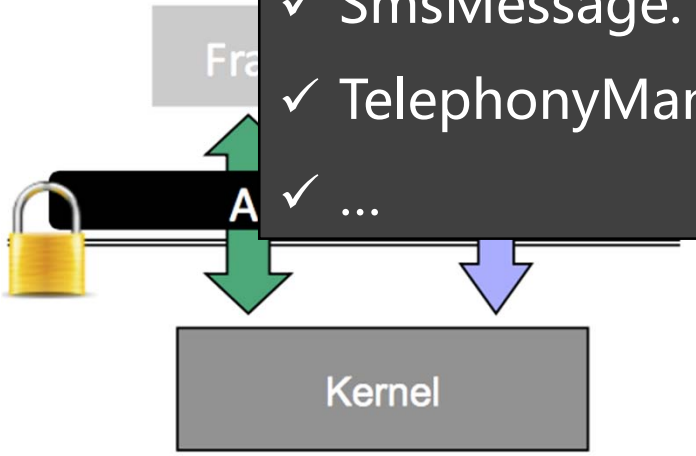
隐私泄露检测

- TaintDroid [OSDI 10]
- AppInt[illegible]
- VetDro[illegible]
- FlowDr[illegible]

```
c = taint_source()
...
a)
```

主要关注的敏感数据

- ✓ LocationManager. getLastKnownLocation()
- ✓ ContentResolver.query(CONTACT_URI)
- ✓ SmsMessage. getMessageBody ()
- ✓ TelephonyManager. getLine1Number()
- ✓ …



[illegible]12]

[illegible]]

- FlaskDroid [Security 13]
- ASM [Security 14]

现有工作

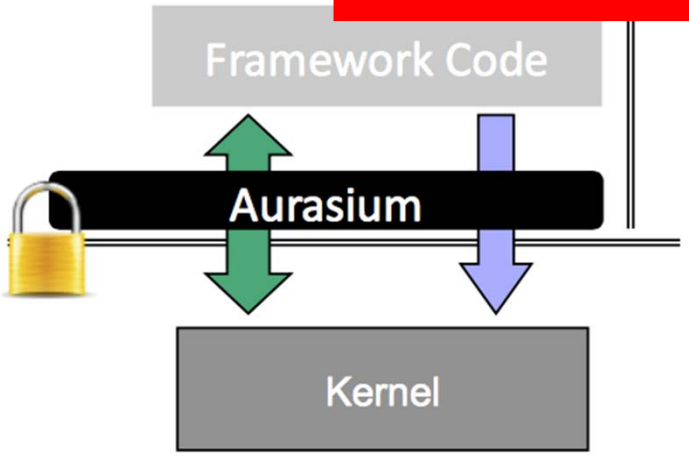
隐私泄露检测

- TaintDroid [OSDI 10]
- AppIntent [CCS 13]
- VetDroid [CCS 13]
- FlowDroid

```
c = taint_source()
...
a = b + c
...
_send(a)
```

现有工作：

无法适用于UIP隐私数据的保护



访问控制机制

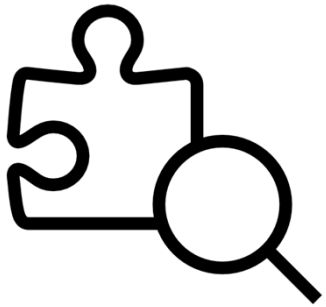
- Auriasium [Security 12]
- SEAndroid [NDSS 13]
- FlaskDroid [Security 13]
- ASM [Security 14]

识别UIP数据是前提

如何识别UIP类数据是保护UIP类数据的前提

- 不是所有的用户输入都是同样敏感（支付密码 vs 搜索关键词）
- 识别APP中所有的UIP数据
- 并且识别接收UIP数据的控件

我们的工作



UIP数据识别：UIPicker

- 自动化的识别APP中所有的UIP控件
- 自动化标识控件的ID
- 支持海量软件的分析

UIP数据保护机制

- 防止UIP数据被中间人攻击
- 监测UIP数据是否经过HTTP传输
- 监测UIP数据是否经过不安全的HTTPs传输

如何识别UIP控件？

挑战

1、接受用户输入的界面很多

2、一个界面中存在多个接受用户输入的控件

3、接受用户输入的控件类型多样

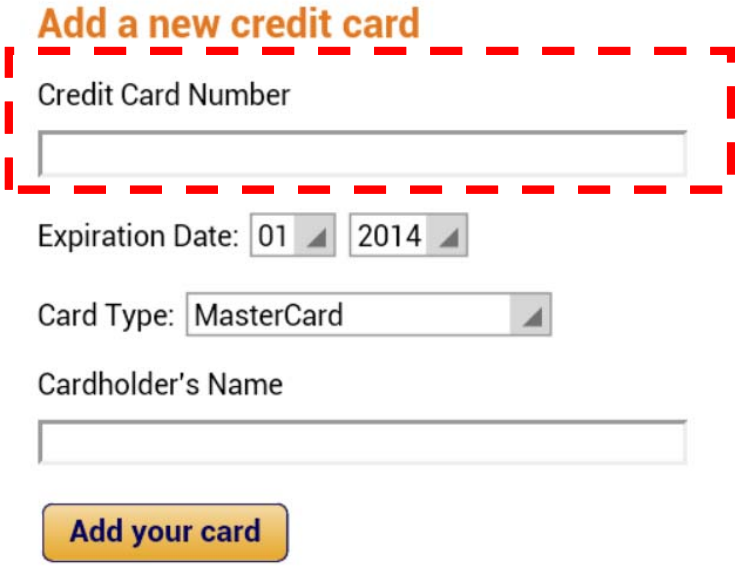
这种方案是否可行？

- 根据控件定义的属性，例如：“android:inputType=textPassword”
- 是否能够很好的解决问题？

我们的想法

UIP控件会包含一些隐私相关的元信息

- UI控件上会包含敏感数据的说明
- 开发者对UI控件的命名会标示一些敏感数据信息

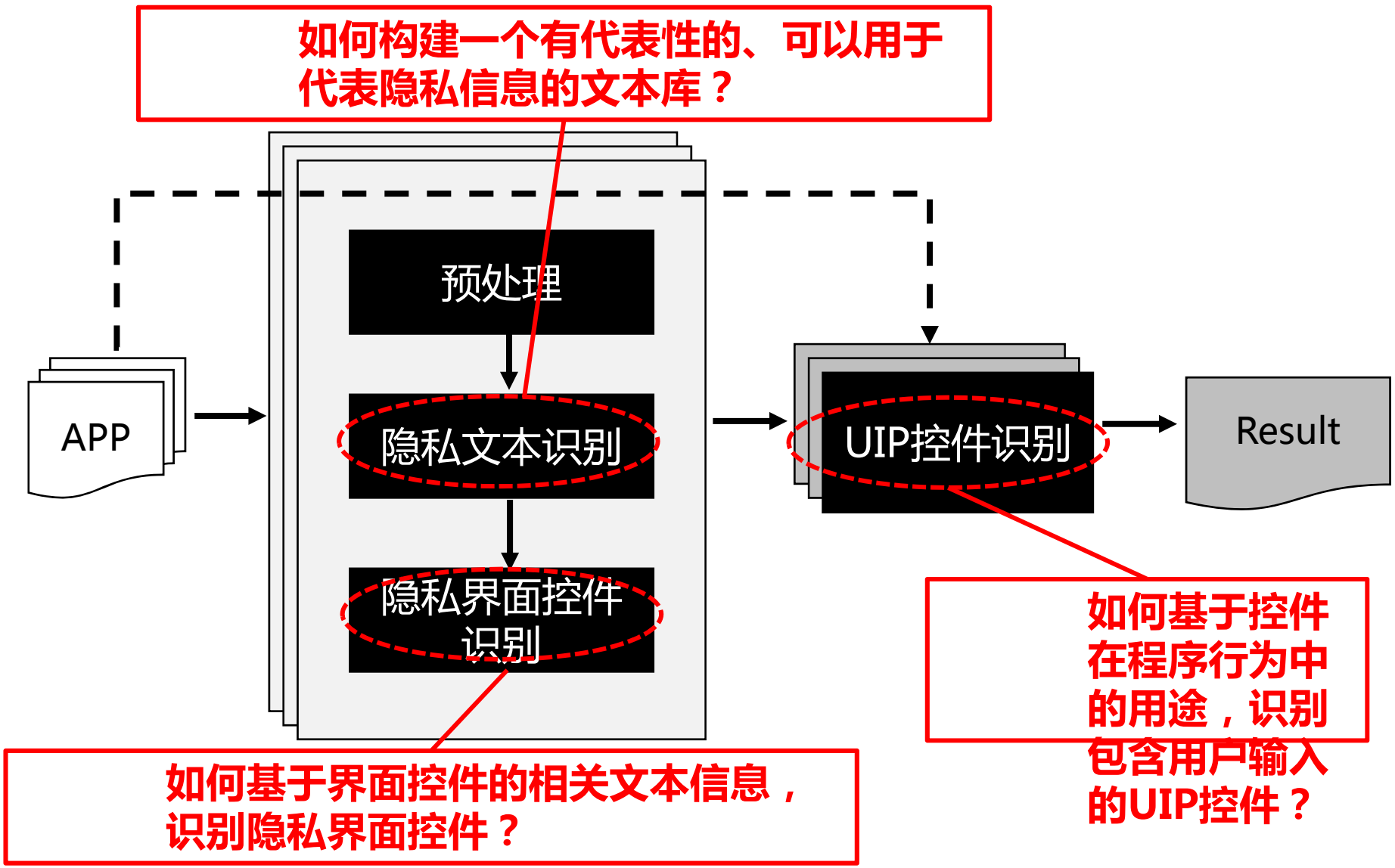


核心想法：

基于文本分析识别用户输入隐私控件

```
<TextView android:text="@string/opl_new_payment_credit_card_number" />
<EditText android:id="@id/opl_credit_card_number" android:inputType="number" />
```

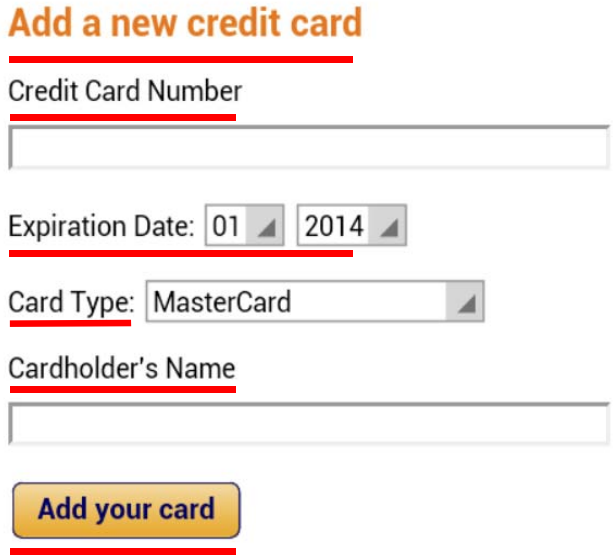
UIPicker架构


如何构建一个有代表性的、可以用于代表隐私信息的文本库？
预处理
APP
隐私文本识别
隐私界面控件识别
UIP控件识别
Result
如何基于控件在程序行为中的用途，识别包含用户输入的UIP控件？
如何基于界面控件的相关文本信息，识别隐私界面控件？

预处理

- 资源文件提取
  - 界面上出现的文本
  - 开发者定义控件的名称
- 分词
  - 基于分隔符和特殊字符
- 清洗
  - 非英语字符串、删除词
- 词根提取(Stemming)
  - 例如：将secured、security替换为secure

1 UI Texts



2 Layout Descriptions

@id/opl_credit_card_number
@string/opl_new_credit_card_expiration_date_month
@string/opl_new_credit_card_save_button

消除开发者习惯带来的隐私文本差异性

隐私文本识别-挑战

挑战：无法依赖人工定义一个很全的隐私文本集合

- 不同开发者会使用不同的词命名控件
- 不同的APP对于同样信息会有不同的文本表述
- **怎么找到尽可能多的能够描述一种用户隐私的文本？**

隐私文本识别-思路

我们的方法

- 人工定义一个初始的敏感文本集合（Initial Seeds）
- 基于初始集合，自动化的找到更多的类似的词

(1) Initial Seeds

(2) 较为完整的隐私文本集合

隐私文本识别-方法

我们的发现：隐私文本不是孤立出现的

- 包含隐私文本的页面会出现多个隐私文本
- 例如：登陆页面（用户名、密码等）、账号设置页面（姓名、邮箱等）

分析方法

1) 基于Initial Seeds标记含有（不含有）隐私文本的界面
2) 基于Chi-test聚类识别能够显著区分一个界面是否敏感的文本

UIP类型	Initial Seeds (7 words)	Chi-test words (228 words)
Login Crendentials & User Profile	username, password, email	mobil phone middl profile cellphon account nicknam firstnam lastnam person birth login confirm detail regist
Location	address, location	zip citi street postal locat countri
Financial	credit card, bank	secur month date pay year bill expir debit transact mm yy pin code

隐私界面控件识别

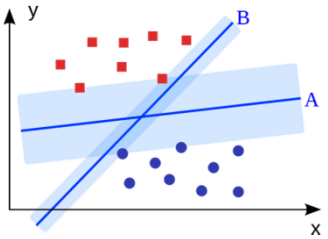
如何判断一个界面控件是否接受UIP数据

- 基于隐私文本判断一个UIP控件是否敏感

识别方法

1) 基于支持向量机（SVM）进行自动化分类

- 控件所属的隐私文本 → 特征值
- 涵盖控件左右邻居控件



隐私界面控件识别

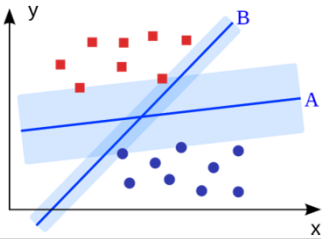
如何判断一个界面控件是否接受UIP数据

- 基于隐私文本判断一个UIP控件是否敏感

识别方法

1) 基于支持向量机（SVM）进行自动化分类

- 控件所属的隐私文本 → 特征值
- 涵盖控件相邻控件



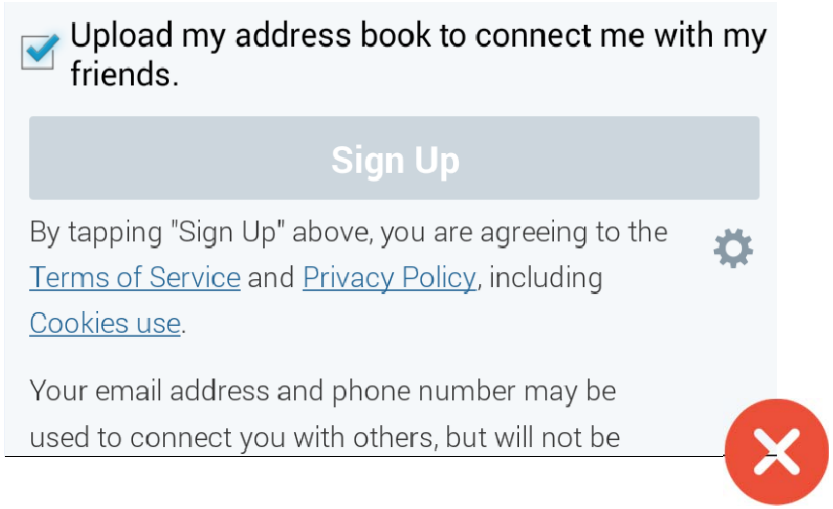
2) 如何获取训练集合

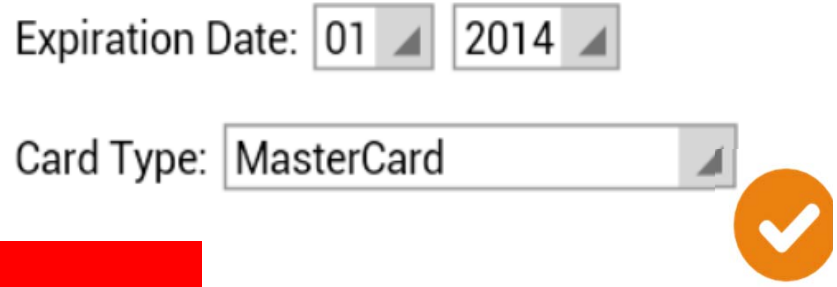
```
<EditText android:id= "@id/password_edittext"
android:inputType="textPassword" />
```

UIP类型	训练集合识别方法
登陆凭证 & 账户信息	控件属性：textEmailAddress, textPersonName, textPassword, textVisiblePassword, password/email/phoneNumber
地理位置	控件属性：textPostalAddress
支付类信息	人工标记

UIP控件识别

- **问题：不是每一个隐私界面控件都处理用户输入的信息**
  - 如何识别一个隐私界面控件内的数据是否来自用户的输入
- **接受用户输入类型：**
  - 主动输入：EditText
  - 被动输入：下拉菜单
  - 定制化控件：com.alipay.InputBox





思路：基于程序访问含有隐私文本控件的行为

UIP控件识别

1、找到访问含有敏感文本信息的控件

```
Activity.setContentView(R.layout.add_credit_card.xml)

Void OnCreate(Bundle bundle){
    InputBox IB = findViewById(2131231511);
    submitBtn = findViewById(2131623982);
    submitBtn.setOnClickListener(new addCardListener());
}
```

2、识别用户交互行为

```
addCardListener.onClick(View v) {
        ......
        creditCard = IB.getText();
```

UIP控件行为特征：

控件的文本信息会在某一个事件处理函数中获取

运行时UIP保护机制

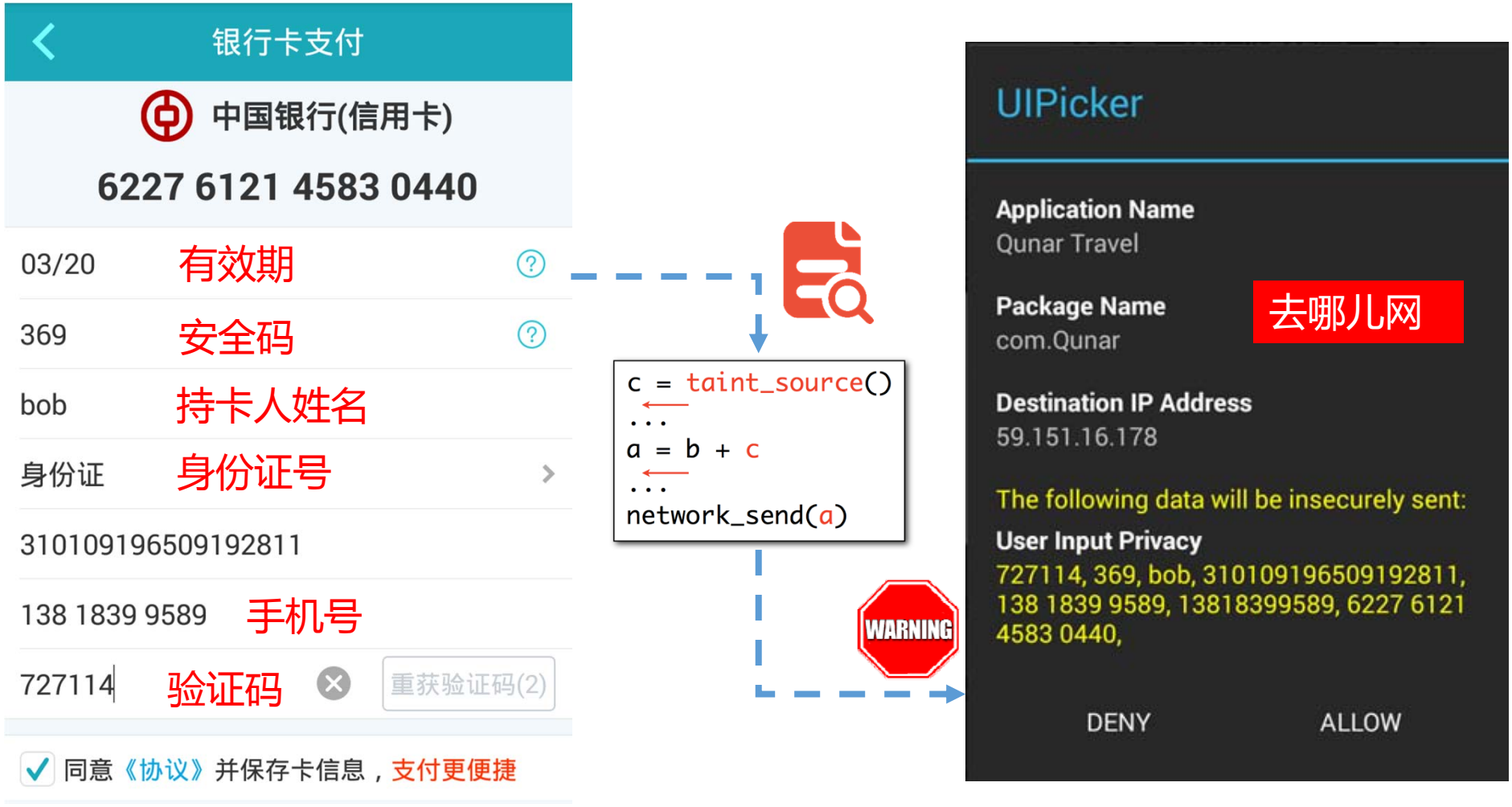
TaintDroid系统[OSDI 2010]

- 基于动态污点分析技术跟踪敏感数据的传播
- 仅支持对通过系统API访问的隐私数据的保护

UIP数据保护系统

1) 基于UIP控件标记隐私数据
2) 基于TaintDroid跟踪UIP数据的传播
3) 监测网络发送UIP数据的行为
   - 是否使用HTTP发送
   - 是否使用不安全的HTTPs发送

运行时UIP保护机制


银行卡支付
中国银行(信用卡)
6227 6121 4583 0440
03/20
有效期
369
安全码
bob
持卡人姓名
身份证
身份证号
310109196509192811
138 1839 9589
手机号
727114
验证码
重获验证码(2)
同意《协议》并保存卡信息，支付更便捷
c = taint_source()
...
a = b + c
...
network_send(a)
WARNING
UIPicker
Application Name
Qunar Travel
Package Name
com.Qunar
去哪儿网
Destination IP Address
59.151.16.178
The following data will be insecurely sent:
User Input Privacy
727114, 369, bob, 310109196509192811, 138 1839 9589, 13818399589, 6227 6121 4583 0440,
DENY
ALLOW

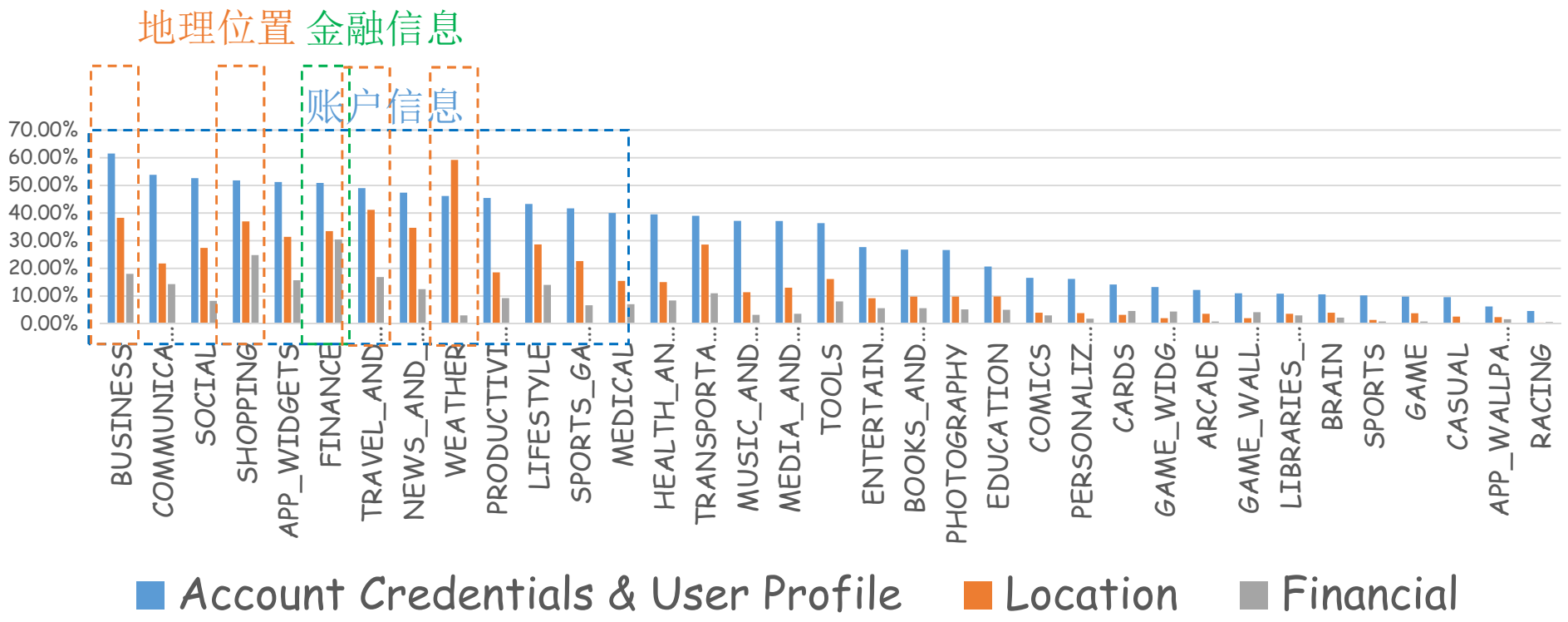
实验分析

数据集

- 来自Google应用商城爬取的17,425 个应用
- 共35个类别，每一个类别500个应用

UIP数据在APP中的分布

- 在17,425个APP中，35.46%的APP含有用户输入隐私
- 在35个类别中，有9个类别过半的APP含有用户输入的隐私



识别到了多少UIP数据

与采用控件属性相比

textEmailAddress, textPersonName, textPassword, textVisiblePassword, password/email/phoneNumber

textPostalAddress

Category	InputType	UIPicker	Incremental
Account Credentials & User Profile	24,021	46,227	26,087
Location	941	14,311	13,370
Financial	-	6,353	-
Total	24,962	71,224	46,262

15x

2x

识别到了多少UIP数据

一些静态UI控件也包含输入类的隐私

Type	# Elements	% in UIP Data
TextView	10,582	14.86%
Customized	5,075	7.13%
Spinner	1,962	2.75%
Others	784	1.10%
Total	18,403	25.84%

精确度

- 验证集合：随机从10个类别中各选取20个应用，共200个应用

UIP类型	True Positive	False Positive	False Negative
人工验证结果	975	67	107

- 结果：
  - 精度度：93.6%
  - 召回率：90.1%

$$Precision = \frac{TP}{TP+FP} \qquad Recall = \frac{TP}{TP+FN}$$

移动平台软件行为管控机制

- 软件行为管控
  - **目的**：安全性
  - 管理程序的执行、控制程序的行为

1. *FineDroid: Enforcing Permissions with System-wide Application Execution Context*. 11th EAI International Conference on Security and Privacy in Communication Networks (SecureComm 15), 2015
2. *Rethinking Permission Enforcement Mechanism on Mobile Systems*，IEEE Trans. on Information Forensics and Security, 2016

移动平台软件行为管控机制

- 传统行为管控机制
  - **针对某个(些)程序**，从程序外部控制其能操作的资源
  - 例如：SELinux、iptables
- 规则样例
  - 允许init类型对unlabeled类型的filesystem进行mount的操作

```
allow init unlabeled:filesystem mount;
```

  - 允许user_t对bin_t类型的file进行除了write setattr ioctl相关的操作

```
allow user_t bin_t:file ~{write setattr ioctl};
```

  - 在WIFI下丢弃uid为10042的数据包

```
iptables -A OUTPUT -o wlan+ -m owner --uid-owner 10042 -j DROP
```

移动平台软件行为管控机制

- 移动应用软件新特征
  - 重交互：移动应用程序处于复杂的交互执行环境中
  - 模块化：移动应用程序内部含有多个利益体的代码

我们看到的问题

- **程序交互中的攻击**
  - 处于交互中的应用程序是否值得相信？
- 一些案例
  - 案例一: 系统预制应用中存在众多权限泄露漏洞，可导致系统重启、程序崩溃、强制关机 [CCS 2013]
  - 案例二: 应用中的数据存储接口被利用，导致用户私密信息泄露、核心业务数据被修改 [NDSS 2013]
  - 案例三: WebView中的 URL 注入攻击

我们看到的问题

- **嵌入代码中的攻击**
  - 打包在同一个应用程序的代码是否都可以相信？
- 程序内部的SDK可以做什么
  - 可以获得APP的哪些敏感数据？
    - by Pluto [NDSS 2016]
    - 通过分析医疗、教育、健康、生活类应用的私有目录下数据
    - 发现APP中的SDK可以读取到用户的性别、电话、年龄、疾病等信息
  - 是否可以攻击程序逻辑？
    - Hook程序代码
    - Hook框架层代码

反思

- **现有软件构造流程是否是一种倒退？**
  - 从软件工程的角度来看
    - 基于组件开发、不重复造轮子、提高复用率
  - 从项目开发的角度来看
    - Time-to-release、保障上线时间、快速出原型
  - 从产品理念的角度来看
    - 业务聚焦、领域细分、单点突破
  - 生产力决定生成关系
    - 移动平台降低了开发者门槛、竞争白热化
    - 移动平台为了**更好的连接** => 移动平台产品的连接
    - 代表着一种更强的生产力：具有各自利益的主体进行协作

反思

- **为何现有行为管控机制无法保护移动APP的安全？**
  - 现有安卓行为管控机制
    - 基于UID隔离各个应用程序
    - 基于UID控制每一个程序可以使用的权限/资源
  - 问题分析
    - **应用程序交互不敏感（Inter-application In-sensitive）**

```
String url = intent.getStringExtra("load_url");
webview.loadUrl(url);
```

怎么判断url来自于开发者还是外部攻击者?

    - **应用程序内部不敏感（Intra-application In-sensitive）**

```
open("/data/data/com.XX/shared_prefs/..")
```

怎么判断访问请求来自于APP还是SDK?

一些相关工作

- 孤立的解决问题
  - IPC Inspection [USENIX Security 2011]
    - 基于进程调用栈自省技术限制程序在交互状态下能够使用的权限集合
  - TrustDroid, XManDroid [NDSS 2012]
    - 基于预定义规则阻止有风险的程序交互
  - Compac [CODASPY 2014]
    - 跟踪程序内部不同package之间的行为交互，限制不同package能够使用的权限
  - FlexDroid [NDSS 2016]
    - 基于跨进程的程序内栈自省技术识别程序内部执行状态，限制程序SDK能够使用的权限

我们的想法：通用化的精细行为管控

- 构造适用于移动平台的新型软件行为管控模型
  - 之前的做法：将行为能力绑定在特定的程序(UID)上
    - 普通API：所有APP都可以调用，例如动态加载、数据库访问
    - 敏感API：APP被授予相应权限，例如访问网络、发送短信
    - 二元组：*<app, behavior>*
  - **我们的思路**
    - **三元组：*<app, context, behavior>***
    - 描述一个APP在处于什么样的上下文中可以做什么
  - Context是什么?
    - 地理位置？设备充电中？设备连接WIFI？
    - android.content.Context？

我们的想法：通用化的精细行为管控

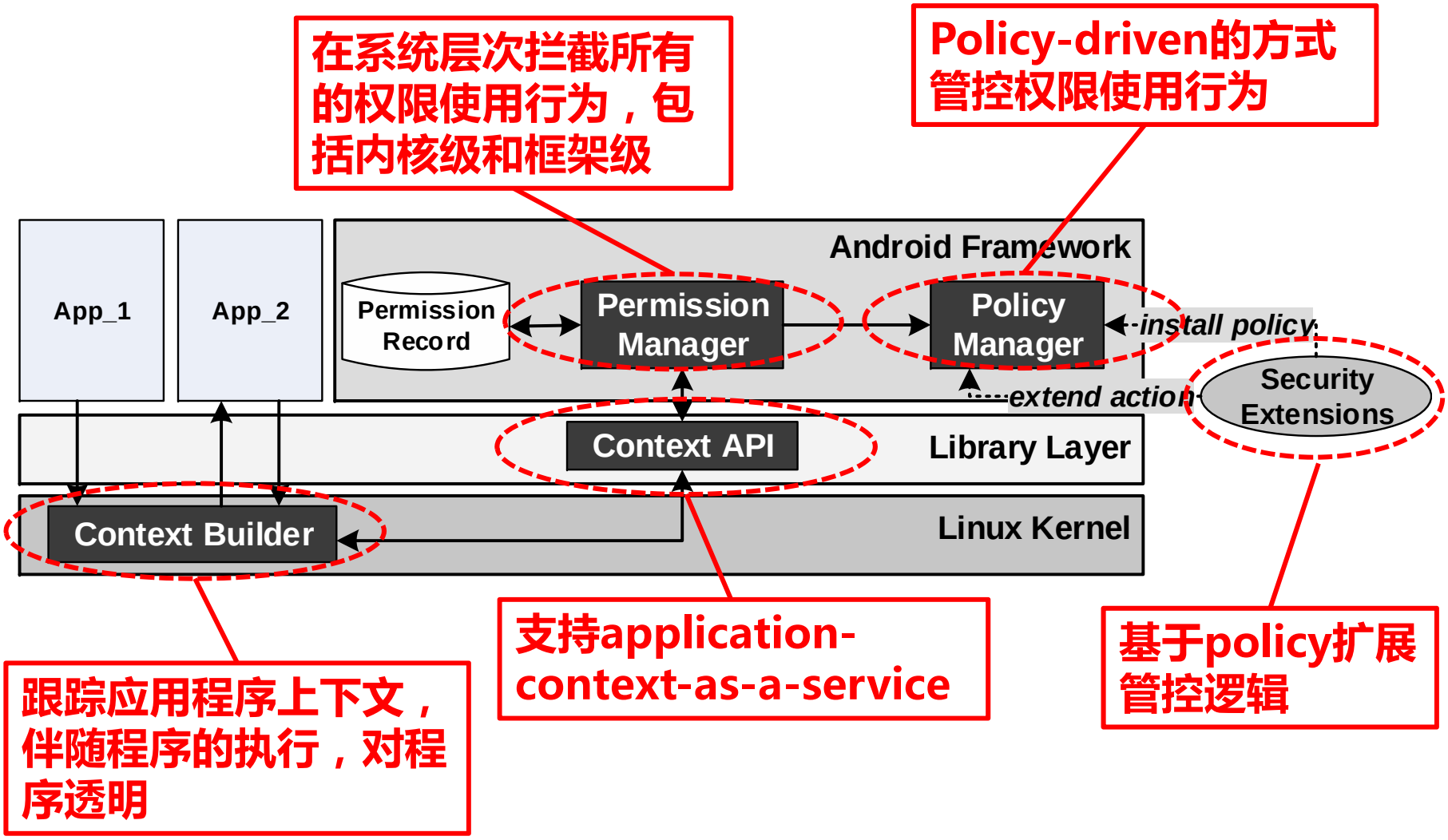
- System-wide Application Execution Context
  - 应用程序交互上下文
    - 刻画程序处于的复杂外部执行流
  - 应用程序内部上下文
    - 刻画程序内部精确的执行流

- Application-context-as-a-Service
  - 系统内建的功能，跟踪程序执行上下文
  - 将程序执行上下文作为一种系统服务
    - 应用程序根据自己的上下文审查函数的调用
    - 系统根据程序的上下文审查API的调用

FineDroid系统设计

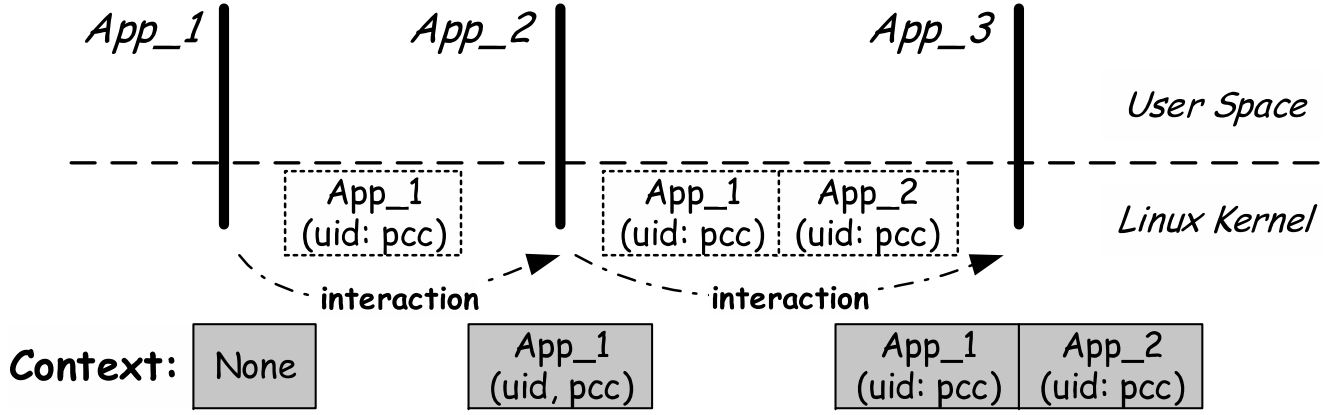
- **以权限为例**，对程序权限使用行为进行管控
  - 系统根据应用程序上下文，管控程序对权限的使用
  - 在进程内管控不同上下文下的权限使用行为
- Application-context-as-a-Service
  - 专有API用于获取当前执行流所处于的程序上下文
  - **目标**：跟踪程序交互，识别程序内部执行流
- Policy-driven的方式管控程序权限使用行为
  - Out-of-the-box，行为管控逻辑独立于APP的实现
  - Easy-to-update，规则即时下发，易扩展

系统架构


在系统层次拦截所有的权限使用行为，包括内核级和框架级
Policy-driven的方式管控权限使用行为
App_1
App_2
Android Framework
Permission Record
Permission Manager
Policy Manager
install policy
extend action
Security Extensions
Context API
Library Layer
Context Builder
Linux Kernel
跟踪应用程序上下文，伴随程序的执行，对程序透明
支持application-context-as-a-service
基于policy扩展管控逻辑

Context Builder模块

- 应用程序**交互**上下文
  - 基于Binder的进程交互机制，支持对调用者进程的跟踪
  - 无法回溯整个调用链上的所有进程
- 扩展Binder协议
  - 维护完整的进程间交互信息
  - 每一个进程需要记录的信息：UID+PCC



Context Builder模块

- 应用程序上下文**传播**
  - Binder级的跟踪只能维护进程间*直接调用*时的上下文
  - 如何在***间接的程序交互***下维护应用程序上下文？
- 有哪些间接的执行流？
  - 组件交互
    - 组件交互都通过AMS路由，跨程序的组件无法直接调用
  - 程序内线程交互
    - 程序内线程交互可能不通过Binder进行
  - 事件交互
    - Callback执行的上下文与Callback注册时的上下文不同

Context API模块

- 获取当前执行流处于的应用程序上下文信息
  - 调用链上进程的基本信息
    - 存储于内核之中，由Binder驱动维护
    - static final native int[] getUidChain()
    - static final native int[] getPccChain()
  - 程序内部上下文详细信息，包含所有stack上函数信息
    - static final PccInfo getPccSig(int uid, int pcc)
    - Stack上的每一个函数的签名信息抽象为一个四元组
    - *<class_name, method_name, method_sig, callsite_offset>*

Permission Manager模块

- 目标：在所有的权限使用点进行拦截，汇总到统一的一个点进行管控
- 两种权限
  - 框架级权限
    - 依赖于PackageManagerService，根据程序安装时用户授予的权限进行管控
    - **处理方法**：调用Policy Manager模块路由权限决策
  - 内核级权限
    - 依赖于Linux内核的UID/GID机制进行保护，例如网络、文件系统
    - **处理方法**：识别特定的UID/GID，通过进程间交互将最终决策交给Policy Manager

Policy Manager模块

- 输入：app, app_context, permission
- 输出：deny/grant/...
- 规则引擎

policy := <*action*> <*app*> <*permission*> <*context*>
action := *grant* | *deny* | ...

  - 基于XML表述和存储
  - 规则匹配：选择一条最能准确表述当前应用程序上下文的规则，通过cache提高匹配效率
  - 扩展性：action可以扩展，规则可以运行时增删改

Policy Manager模块

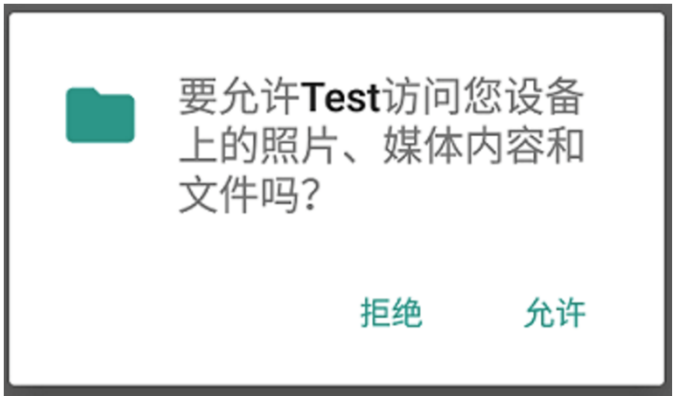
- 一个例子

```
<policy action="deny" app="com.android.mms" permission="SEND_SMS" >
  <uid-selector selector="strictcontains" >
    <uid-context uid="^com.android.mms" pcc="*" />
    <uid-context uid="com.android.mms" />
      <pcc-selector selector="contains" >
        <method-sig className="com.android.mms.transaction.SmsReceiver"
                    methodName="beginStartingService" />
      </pcc-selector>
    </uid-context>
  </uid-selector>
</policy>
```

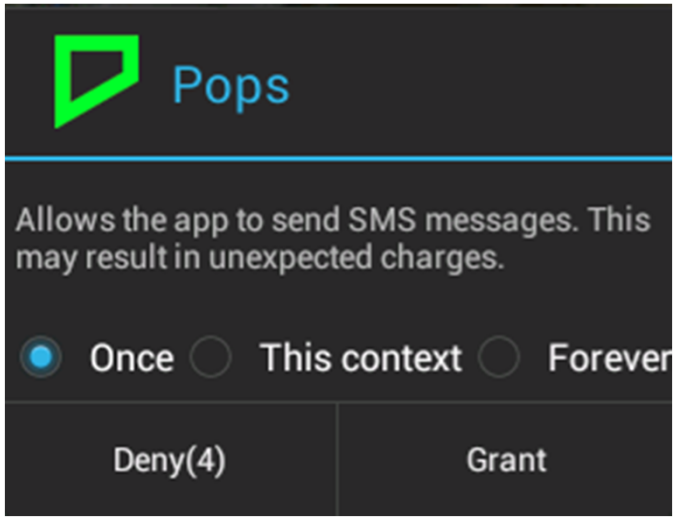
XML Tag	用途
uid-selector	表述一个程序交互链的特征
uid-context	表述一个程序上下文的特征
pcc-selector	表述一个程序内部函数调用链的特征
method-sig	表述一个程序内部函数调用的特征

Security Extensions模块

- 案例一：APP权限管理
  - 权限动态管控



  - 权限即时管控



Security Extensions模块

- 案例二：权限泄露漏洞修复
  - 权限泄露漏洞
    - APP_A存在一个供外部调用的**接口X**
    - APP_A的敏感行为能力通过**X**被APP_B使用，导致能力泄露

权限泄露漏洞的一个例子

```
public class SmsReceiverService extends Service {
    public int onStartCommand(Intent intent, int flags, int startId) {
        ... ...
      Message msg = mServiceHandler.obtainMessage();
      msg.arg1 = startId;
      msg.obj = intent;
      mServiceHandler.sendMessage(msg);
        ... ...
    }

  private final class ServiceHandler extends Handler {
    public void handleMessage(Message msg) {
        Intent intent = (Intent)msg.obj;
        if (intent != null) {
            String action = intent.getAction();
                ... ...
            } else if (SMS_RECEIVED_ACTION.equals(action)) {
                handleSmsReceived(intent, error);
            }
                ... ...
        }
    }
  }
  private void handleSmsReceived(Intent intent, int error) {
    SmsMessage[] msgs = Intents.getMessagesFromIntent(intent);
    String format = intent.getStringExtra("format");
    Uri messageUri = insertMessage(this, msgs, error, format);
    if (messageUri != null) {
        long threadId = MessagingNotification.getSmsThreadId(this, messageUri);
        MessagingNotification.blockingUpdateNewMessageIndicator(this, threadId, false);
    }
  }
}
```

AOSP中Mms程序的WRITE_SMS权限泄露漏洞

基于规则的漏洞修复

- 如何基于FineDroid修复权限泄露漏洞？
  - 基于程序间交互上下文，可以有效区分一个当前的执行流是来自于**内部程序**还是**外部程序**
  - 基于程序内部上下文，可以精确识别当前执行流**是否处于漏洞被利用的路径**上
  - 例子：修复SmsReceiver中的 SEND_SMS 权限泄露

```
<policy action="deny" app="com.android.mms" permission="SEND_SMS" >
  <uid-selector selector="strictcontains" >
    <uid-context uid="^com.android.mms" pcc="*" />
    <uid-context uid="com.android.mms" />
      <pcc-selector selector="contains" >
        <method-sig className="com.android.mms.transaction.SmsReceiver"
                    methodName="beginStartingService" />
      </pcc-selector>
    </uid-context>
  </uid-selector>
</policy>
```

修复规则的自动生成

- 基于漏洞检测工具，自动生成修复规则
  - CHEX [CCS 2012]
  - 对检测日志分析，提取函数调用路径和组件信息

应用名	泄露路径数	生成规则数
com.gmail.traveldevel.android.vlc.app-131	2	2
com.froogloid.kring.google.zxing.client.android-67	24	24
de.cellular.tagesschau-5	361	361
com.akbur.mathsworkout-92	2	2
com.appspot.swisscodemonkeys.paintfx-4	2	2
com.androidfu.torrents-26	1	1
com.espn.score_center-141	6	6
com.espn.score_center-142	6	6
fr.pb.tvflash-9	2	2
hu.tagsoft.ttorrent.lite-15	8	8
总计	414	414

Application-context-as-a-service

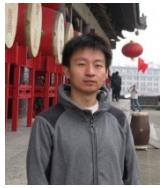
- 应用程序上下文的应用场景
  - 当前Component是被外部程序调用还是内部程序调用？
  - 我自身的一些私有接口会不会被外部程序调用？
  - 我集成的SDK会不会发短信？会不会读取我的私有文件？
  - 我集成的SDK会不会搜集我的用户的信息？
  - 我集成的SDK有没有可能攻击我？
  - 等

InForSec：网络安全国际学术论坛

www.inforsec.org

联合发起人


杨珉


Tao Wei


段海新


Tao Wan

- 连接国际学术圈、对接工业界的学术社区
- 技术委员会：30+ 位顶尖华人学者
- 线下/线上融合的学术交流活动

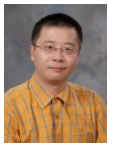

Kang Li


Yan Chen


Hao Chen


Xiaofeng Wang


Wenyuan Xu


Guofei Gu


Yang Liu


Tielei Wang


Long Lu


Zhenkai Liang


Ninghui Li


Peng Ning


Peng Liu


Kui Ren

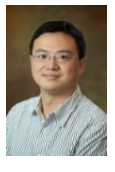

Zhiqiang Lin


苏璞睿

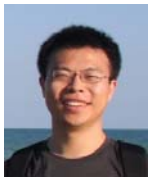

Xiangyu Zhang


Dongyan Xu


武成岗


Heng Yin


Xvxian Jiang


陈恺


Wenliang Du


Zhiyun Qian

敬请各位专家和同行指导！