

# ANÁLISIS DE FLUJO DE INFORMACIÓN EN APLICACIONES ANDROID

**Lina Marcela Jiménez Becerra**

UNIVERSIDAD DE LOS ANDES  
DEPARTAMENTO DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN

Junio 9, 2015

# Descripción del Problema

## Manipulación de información del usuario

- El desarrollador Android no tiene cómo definir políticas de seguridad para regular el flujo de información de sus aplicaciones.
- Complejidad para prevenir fugas de información del usuario.

# Descripción del Problema

## Manipulación de información del usuario

- El desarrollador Android no tiene cómo definir políticas de seguridad para regular el flujo de información de sus aplicaciones.
- Complejidad para prevenir fugas de información del usuario.

## Reporte McAfee

- Aplicaciones Android invasivas de la privacidad del usuario.
- No toda aplicación invasiva contiene malware.
- De las aplicaciones que más vulneran la privacidad del usuario 35 % contienen malware.

# Descripción del Problema

## Limitaciones de la API

- Políticas de control de acceso de la API.
- Regular el acceso a recursos protegidos.
- No hacen seguimiento al flujo de información.

## Propuestas existentes

- Análisis estático y análisis dinámico.
- Análisis dinámico: actuales caminos de ejecución.
- Análisis estático: es posible incluir todos los caminos de ejecución.

# Descripción del Problema

## Propuestas existentes

Data-Flow con técnicas de análisis tainting.

- Se hace seguimiento a los datos marcados.
- No incluye todos los posibles caminos de ejecución.
- Ejemplo: FlowDroid

# Descripción del Problema

## Propuestas existentes

Flujo de información con técnicas Program Dependence Graphs(PDG).

- Los PDG proveen una representación del programa que se analiza.
- Análisis de flujos de información del programa de principio a fin.
- Incluye todos los posibles caminos de ejecución.
- Ejemplo: Joana.

# Descripción del Problema

## Propuestas existentes

Enfoque de las propuestas existentes:

- Identificar fugas de información en aplicativos ya implementados.
- FlowDroid: no incluye todos los posibles caminos de ejecución.
- Joana: no permite definir las políticas de seguridad a evaluar.

# Propuesta de solución

## El desarrollador requiere

- Definir políticas de seguridad desde la implementación de sus aplicativos.
- Una herramienta que verifique las políticas definidas.
- Garantizarle al usuario que la aplicación respeta determinadas políticas de seguridad.





# Características de Jif

- Lenguaje tipado de seguridad.
- Extensiones de seguridad al lenguaje java.
- Restricciones para uso de la información.
- Análisis de flujo de información mediante chequeo de etiquetas.

# Características sobresalientes de Jif

- Anotar propiedades de seguridad.
- Verificar las propiedades de seguridad.
- Cubrir todas las posibles ramas de ejecución en el análisis.
- Diseñado para aplicativos Java.

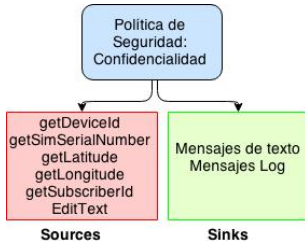
Flujos: explícitos - implícitos

```
int x,y;  
x = 1;  
y = 4 + x;
```

```
void foo(a){  
  int x;  
  if(a > 10)  
    x = 1;  
  else  
    x = 2;  
  printf(x);  
}
```



# Política de Seguridad



```
String imei = getDeviceId();  
sendTextMessage(imei);
```



# Anotaciones Propuestas

## DLM de Jif

**Principales**  
Autoridad

**Políticas**  
dueño: lista-lectores

**Etiquetas**

```
int code;  
int {Alice:} code;
```

### Autoridad máxima

El Principal *Alice* representa la máxima autoridad del programa.

### Política para anotar información con nivel de seguridad alto:

*{ Alice : }*

Sólo la autoridad máxima del programa podrá leer la información.

### Política para anotar información con nivel de seguridad bajo:

*{ }*

No se define un Principal la información podrá leerse por todos.

# Anotaciones a la API

## Flujo de información en la API

- La API posibilita el acceso de la app a sources y Sinks.
- Se generan flujos de información.
- Controlar flujos de información entre sources y sinks.

## Sources y sinks definidos en la API

- getDeviceld (método source) → TelephonyManager
- Mensajes de texto (sinks) → SmsManager



# Anotaciones a la API

## Flujo explícito

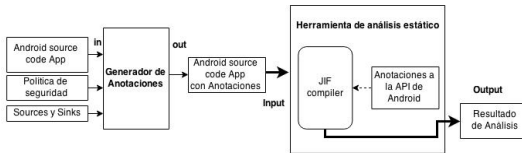
```
String {Alice:} imei = getDeviceId();  
String {} pub = imei;
```

## Flujo implícito

```
String {Alice:} passwd = EditText.getText();  
boolean {} passwdOk = false;  
if (passwd.equals("superSecure"))  
    passwdOk = true;  
if (passwdOk)  
    Log.i("INFO", "Password correcto");  
else  
    Log.i("INFO", "Password incorrecto");
```

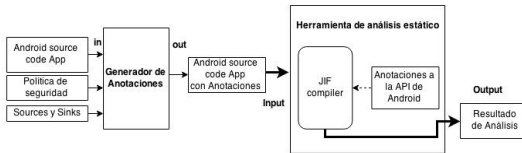
## Implementación

## Herramienta de Análisis Estático

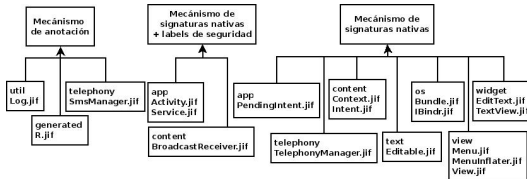


# Implementación

## Herramienta de Análisis Estático



## Anotaciones a la API



# Cuadro comparativo

Item	Prototipo	FlowDroid	JoDroid
Precisión	-	+	+
Recall	+	-	-
Costo en desempeño	-	-	+
Detección Flujos Implícitos	✓	X	✓
Detección automática de sources y sinks	X	✓	X
Soporte para análisis InterApp	X	✓	X

# Conclusiones

- Se dan los primeros pasos para el análisis de flujo de información de aplicaciones Android mediante Jif.

# Conclusiones

- Se dan los primeros pasos para el análisis de flujo de información de aplicaciones Android mediante Jif.
- El desarrollador obtiene las ventajas de bajo costo en desempeño.
- Análisis de flujos implícitos.

# Conclusiones

- Se dan los primeros pasos para el análisis de flujo de información de aplicaciones Android mediante Jif.
- El desarrollador obtiene las ventajas de bajo costo en desempeño.
- Análisis de flujos implícitos.
- Desempeño y completitud en el análisis.

# Conclusiones

- Se dan los primeros pasos para el análisis de flujo de información de aplicaciones Android mediante Jif.
- El desarrollador obtiene las ventajas de bajo costo en desempeño.
- Análisis de flujos implícitos.
- Desempeño y completitud en el análisis.
- Retos para el análisis de aplicaciones Android mediante el sistema de anotaciones de Jif.



# Trabajo Futuro

- Extensiones al esquema de anotación.
- Análisis de políticas de integridad.
- Mecanismos adicionales: declasificación y endorsement.

Preguntas ?