# Android static taint analysis 기법과 발전 방향

논문 230편을 통해 정리한 연구 동향

2022. 7. 4.

손지훈





### 목차



- Introduction
- Android static taint analysis 소개
- Flowdroid를 통해 알아보는 static taint analysis
- □ 논문 230편을 통해 정리한 연구 동향
- ☐ Future works

# Introduction

### Introduction



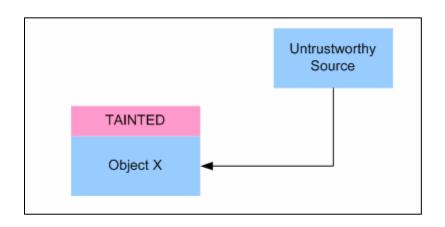
### **Android static taint analysis**

- Android application을 자동화 분석하는 하나의 기법
  - 분석해야 할 애플리케이션의 수는 점차 증가
  - 기존의 악성 애플리케이션에서 코드 일부만을 바꾸는 기법 → 자동 생성 가능
  - 분석가의 시간은 한정되어 있으므로 자동화 탐지가 필요함
- Static taint analysis 기법을 이용하면 악성 애플리케이션 자동 탐지 가능
  - Taint analysis 기법은 데이터의 흐름을 분석할 수 있는 기법 중 하나임
  - Static taint analysis 기법을 Android application에 적용하기 위해서 많은 어려움이 있음
- Flowdroid를 이용하면, 10분 내외의 시간에 1개의 애플리케이션 분석 가능
  - 애플리케이션 전체의 코드 구조를 모델링하고, 가능한 모든 데이터의 흐름을 파악
  - 사용자의 개인 정보를 유출하는지 혹은 해킹을 시도하는지 여부를 파악



### **Taint analysis**

▪ 신뢰할 수 없는(Untrustworthy) 데이터의 프로그램 내부 흐름을 분석하는 기법



```
main:
         call SOURCE
                         // create taint (eax)
         cmp eax, 20
         je if
         ine else
     if:
         mov ebx, eax
                         // taint propagation (eax -> ebx)
         push ebx
         call SINK
                         // source to sink
10
     else:
                         // delete taint
         xor eax, eax
```

#### ■ 용어 정리

- Taint: "오염"이라는 뜻으로 추적의 대상. tainted 데이터로부터 영향을 받은 데이터들은 tainted 된다.
- Source: taint 데이터가 생성되는 위치. 분석가가 추적하기를 원하는 데이터의 생성 지점. 외부 입력 값
- Sink: taint 데이터가 도착하는 위치. tained 데이터가 sink에 도착하는지 여부를 파악



### Taint analysis on Android application

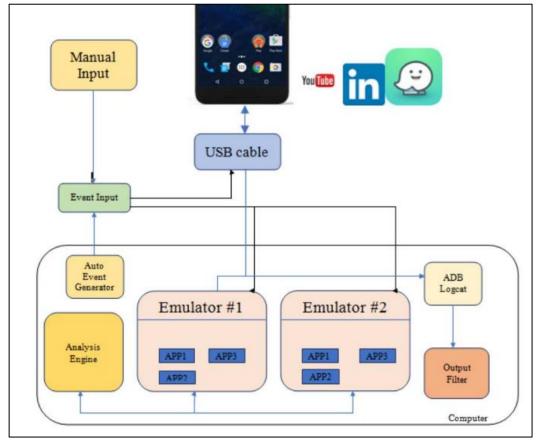
- 패러다임의 변화
  - Taint analysis 의 목적, "개인정보를 유출하는 악성 앱 탐지"
  - Source: 사용자의 개인정보를 반환하는 함수
  - Sink: 사용자 데이터를 외부로 유출할 수 있는 함수



### **Static vs Dynamic taint analysis**

#### Dynamic taint analysis

- 안드로이드 실행 환경(DVM, ART)을 일부 수정하거나 모니터링하는 방법으로 taint analysis 수행
- User input, Event를 자동으로 발생시키고,
   로그 데이터나 함수 실행 등을 추적함
- 장점
  - 실제 실행 환경과 거의 유사하게 동작하기 때문에 정확성이 높음
- 단점
  - 코드 실행 환경 구축이 까다로움
  - 시간이 오래 걸림
  - 코드 커버리지가 낮음





### **Static vs Dynamic taint analysis**

#### Static taint analysis

- 소스 코드(혹은 컴파일된 코드)를 input으로 받아서, 코드를 실행하지 않고 taint analysis 수행
- 코드 구조와 문법, 순서, 의미를 고려하여 taint가 전달되는 과정을 분석
- 장점
  - 코드 실행 환경 구축이 쉽다
  - 시간이 적게 걸린다
  - 코드 커버리지가 높다
- 단점
  - 애플리케이션 실행 환경을 알 수 없기 때문에 정확성이 떨어진다 (특히 False Positive가 높음)



### **Challenges to static taint analysis**

- 안드로이드 애플리케이션에 대해서는 static 기법이 더 많이 사용되었음
  - dynamic 기법의 단점 > static 기법의 단점
- 그러나 Android application에 대해 Static analysis를 적용하는 것은 쉽지 않음
  - Dalvik bytecode에 대한 분석의 어려움
  - Main entrypoint가 없음
  - Java reflection, Implicit flow, Inter-component communication(ICC) 등 처리가 어려움



### **Considerations 1 – Android specific features**

#### Android framework modeling

- Android system에서 제공하는 클래스, 함수들
- 1) Android framework 모델링, 2) Conservative approach

### Component lifecycle

- Android application은 4가지 컴포넌트(Activity, Service, Broadcast Receiver, Content Provider)로 구성됨
- 각각의 컴포넌트는 생성/시작/재개/정지/중지/종료의 lifecycle을 가짐
- Main entrypoint가 존재하지 않으며, 컴포넌트들의 lifecycle을 분석해야 실행 흐름을 파악할 수 있음
- 시스템/유저 이벤트(버튼 클릭, GPS 변경 등)에 따른 callback 함수를 분석해야 함



### **Considerations 1 – Android specific features**

#### Inter-component communication(ICC)

- 컴포넌트가 다른 컴포넌트와 소통할 때 Intent를 이용함
- Explicit Intent는 어떤 컴포넌트와 소통할지 명시해주는 것이지만,
   Implicit Intent는 소통할 컴포넌트를 Android system이 찾아주도록 요청하는 것

#### Native code

- C나 C++로 컴파일된 라이브러리를 Android 에서 사용할 수 있음(Android NDK, JNI)
- 문제는, Java 입장에서 C나 C++ 라이브러리를 전혀 해석할 수 없다는 점
- 마찬가지로 1) Native Function 모델링, 2) Conservative approach



#### **Considerations 2 – Sensitivities**

Flow sensitivity

```
Code Snippet

Sensitive Call-Graph

Insensitive Call-Graph

| public void flowSensitivity() {
| Animal a = new Human(); | a.walk(); | a = new Cat(); | Human.walk | Human.walk | Cat.walk |
```

Object sensitivity

```
public void objectSensitivity() {
     Contains c1 = new Contains();
                                                        objectSensitivity
                                                                                                objectSensitivity
     Contains c2 = new Contains();
     c1.setAnimal(new Human());
     c2.setAnimal(new Cat());
     c1.animal.walk();
   public class Contains {
                                                         Human, walk
                                                                                              Human, walk
                                                                                                           Cat.walk
     Animal animal;
     public void setAnimal(Animal a) {
       this.animal = a;
                                                     (e) Object Sensitty
13 }
```



#### **Other Considerations**

#### Implicit flows

■ Taint가 실행 흐름에 간접적으로 영향을 주어, 결과 값을 다르게 만드는 것

```
3  \rightaint:
4    print(True)
5  \rightarrow else:
6    print(False)
```

### Java-specific features

- Reflection: 중요 데이터 숨기거나, private 변수를 변경 가능함
- Exception: 상황에 따라 예외처리가 달라지기 때문에 처리하기 까다로움



#### Flowdroid 소개

- 2014년에 발표된 Android application static taint analysis 도구
  - Google Scholar 기준 2135회 인용, Github star 약 700개
  - 현재 존재하는 static analysis 도구 중에서 꽤(?) 뛰어난 성능
  - 현재까지 업데이트되고 있는 유일한 도구

Flowdroid: Precise context, flow, field, object-sensitive and lifecycle-aware taint analysis for android apps
SArzt, S Rasthofer, C Fritz, E Bodden, A Bartel... - Acm Sigplan ..., 2014 - dl.acm.org
... In this work we thus present FLOWDROID, a novel and ... Novel on-demand algorithms help
FLOWDROID maintain high ... Android test applications, FLOWDROID finds a very high fraction ...
☆ 저장 切 인용 2135회 인용 관련 학술자료 전체 28개의 버전 Web of Science: 794 ≫

r	develop 🕶 🥲 11 branches 🕒 8 ta	gs Go t	co file Code •
9	StevenArzt gnarf	3957c75 4 days ago	1,040 commits
	.settings	Merge branch 'develop' of https://github.com/MariaKober/FlowDroid in	16 days ago
	DroidBench @ 9898d71	improved the handling of hierarchy data from summaries	4 days ago
	soot-infoflow-android	style fix	6 days ago
	soot-infoflow-cmd	Merge branch 'develop' of https://github.com/MariaKober/FlowDroid in	16 days ago
	soot-infoflow-summaries	improved the handling of hierarchy data from summaries	4 days ago
	soot-infoflow	gnarf	4 days ago
	.gitignore	Merge branch 'develop' of https://github.com/timll/FlowDroid into dev	2 months ago
	.gitmodules	enabled DroidBench test cases	2 years ago
	.project	gitignore and .project added	16 days ago
	LICENSE	initial check-in of FlowDroid 2.5	5 years ago
	README.MD	Update README.MD	5 months ago
	pom.xml	new version	5 months ago



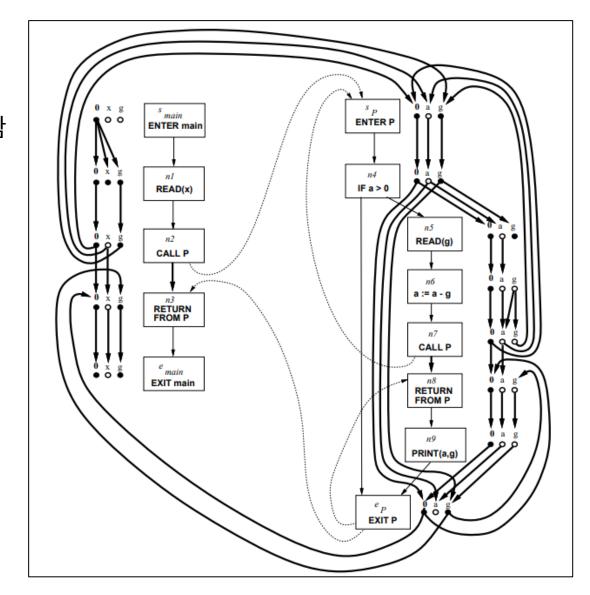
#### Flowdroid: Architecture

### Soot: APK to Callgraph

- Soot를 이용하여 Java bytecode 에서 Jimple로 변환함
  - Jimple: three-address Java intermediate representation
- Soot는 callgraph 생성을 지원함

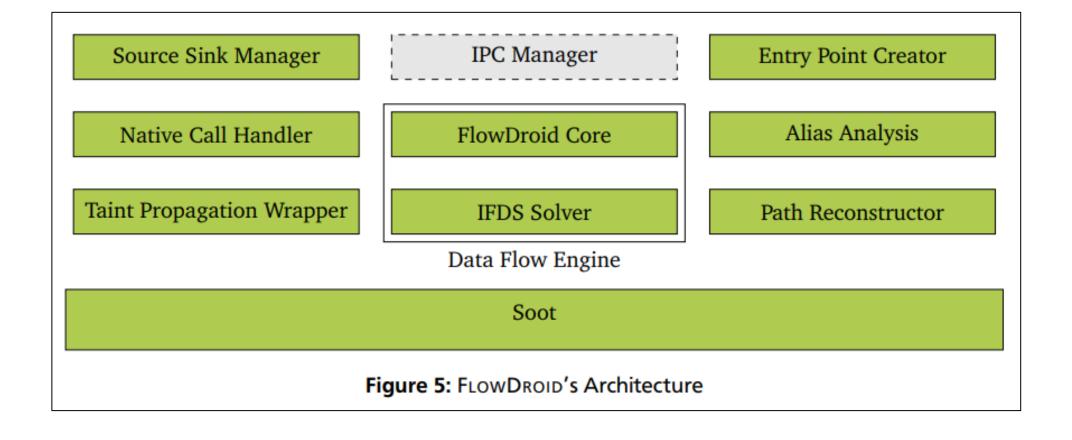
#### IFDS framework

- Interprocedural, finite, distributive, subset problems
- Callgraph를 기반으로 data-flow analysis를 poly-time 내에 풀어낼 수 있도록 하는 알고리즘





#### Flowdroid: Architecture

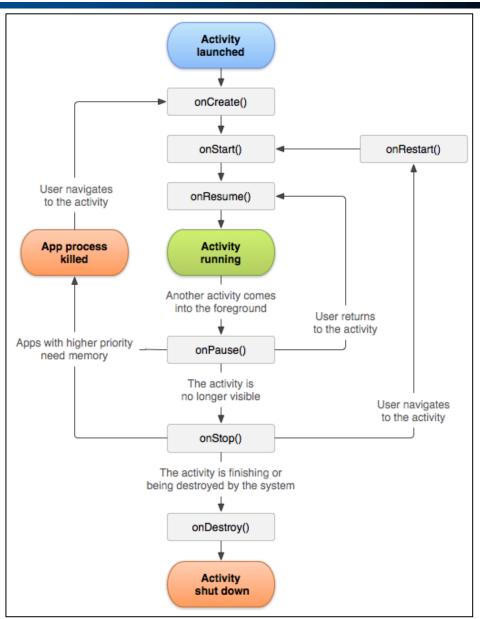




### Flowdroid: Lifecycle Modeling

- Component lifecycle을 고려한 모델링 기법
  - 컴포넌트들은 AndroidManifest.xml 에 등록되어 있음

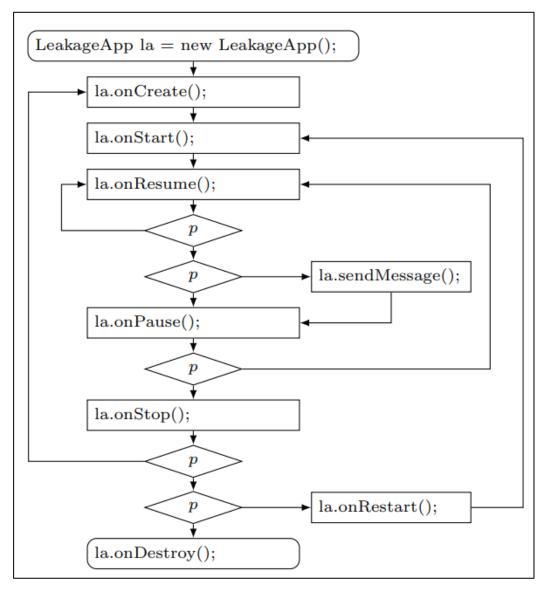
- lifecycle의 실행 순서는 알 수 없음
  - 유저가 다른 화면을 클릭한다면?
  - 유저가 오랫동안 앱을 사용하지 않은 상태로 방치한다면?
- 각각의 컴포넌트들을 오른쪽 그림과 같이 모델링





### Flowdroid: Lifecycle Modeling

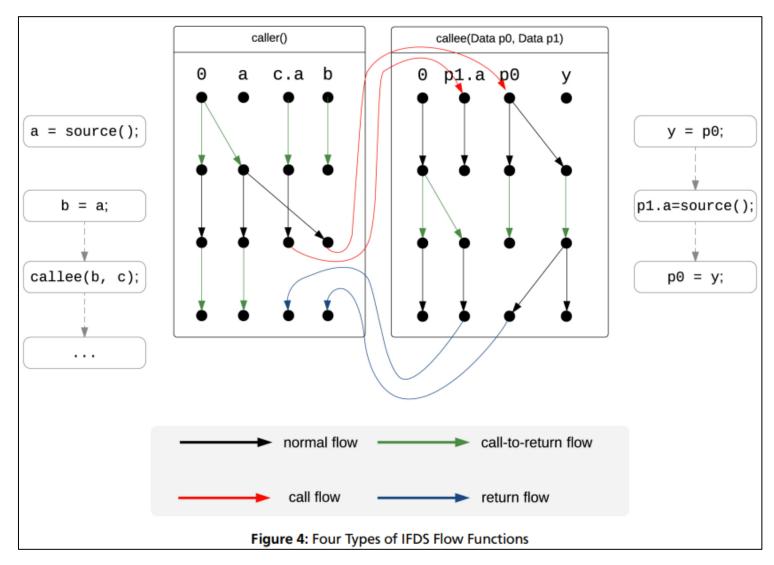
- Callback 함수는 '임의의 위치'에 등장하는 것으로 가정
  - 시스템&유저 이벤트에 따라 Callback 함수가 실행됨
  - ex) 유저의 버튼 클릭, GPS 변경 등
- 개별 Component의 실행 순서는 '임의의 순서'로 가정
  - 하나의 앱에는 여러 개의 컴포넌트가 존재
  - 무엇이 먼저 실행될지 알 수 없어, 가능한 모든 조합을 고려
- Lifecycle을 고려하여 dummy main 함수 생성
  - 개별 component들을 각각 모델링한 후에, 앱에 존재하는 모든 component들을 합쳐서 하나의 main 함수 생성
  - 추후 분석에서 Entry point로 사용됨





### Flowdroid: Taint propagation

- Taint propagation rules
  - Taint 전달 규칙
  - Normal flow
  - Call flow
  - Return flow
  - Call-to-return flow





### Flowdroid: Taint propagation

#### Taint propagation rules

- (예시)Normal flow
  - $x. f^n = y. f^m$  (n, m은 경로의 길이를 의미)

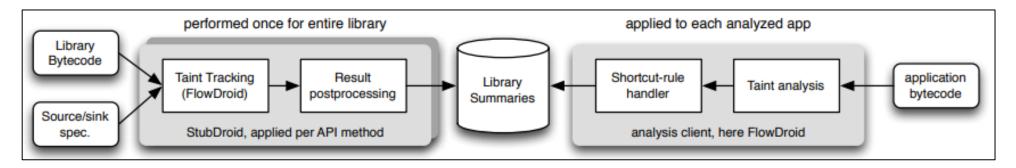
$$T \xrightarrow{s} \begin{cases} T \cup \{x.f^n.f^p\} & \forall p: y.f^m.f^p \in T \\ T \setminus \{x.f^n\} & y.f^m.f^* \notin T \land \neg arrayElem(x.f^n) \\ T & \text{otherwise} \end{cases}$$

- X is tainted, Y,Z are not tainted
- 1) Y = X , taint 전파
- 2) X = Y , taint 소멸
- 3) Y = Z , nothing



### Flowdroid: Taint propagation

- Stubdroid: Modeling the Android framework
  - 여태까지 알아본 방법으로 Android framework를 하나하나 분석할 수 있음
  - 그러나 각각의 앱마다 이를 수행하는 것은 비효율적이며 시간이 많이 듬
  - Flowdroid를 이용하여 데이터 흐름을 pre-compute 하는 방법



Stubdroid를 사용하면 Flowdroid의 실행 시간 단축이 가능함



### Flowdroid: Taint propagation

- Native 함수 처리: 실행 흐름 파악 불가
  - 2017년 기준 약 70%의 앱에서 적어도 1개 이상의 native 라이브러리 사용
  - C나 C++로 짜여진 라이브러리의 실행 흐름은 전혀 분석할 수가 없음!!!
  - Java 레벨에서 native 함수를 호출하는 JNI 코드가 존재함
- Modeling? Over-approximation?
  - Modeling
    - 일부 native 시스템 라이브러리 or 자주 쓰이는 라이브러리는 모델링 진행
    - System.arraycopy
  - Over-approximation
    - BLACKBOX 접근법
    - 함수의 파라미터 1개라도 tainted 되었다면, 나머지 파라미터와 return value를 모두 tainted 되었다고 간주함



### 연구 방법

Wohlin, C. (2014, May). Guidelines for snowballing in systematic literature studies and a replication in software engineering. In Proceedings of the 18th international conference on evaluation and assessment in software engineering (pp. 1-10).

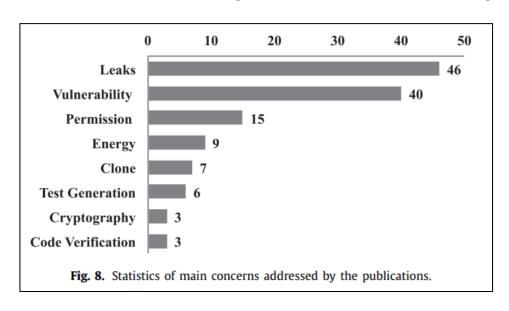
- Google Scholar 대상으로 크롤러 제작
- Systematic literature review 중 snowballing 기법에 따라 연구 진행

1 목록	연도 이를	l용수 저자	출판사	수절 여부	라벨	មាន
88 P138	2020 Static Flow Analysis for Hybrid and Native Android Applications	1 C Rizzo	pure.rovalhollowav.ac.uk	TRUE		이 논문의 박사학위 논문임. 나중에 나올 거야!! BabelView: Evaluating the Impact of Code Injection Attacks in Mobile Webviews
84 P142	2020 Investigating performance issues in mobile apps	1 T Das	iris.assi.it	TRUE		static을 이용하긴 하나. Eclipse plugin이라 화이트박스임.
es P144	2020 Characterizing the evolution of statically-detectable performance issues of Android apps	3 T Das, M Di Penta, I Malavolta	Springer	TRUE		static를 이용하긴 하나, Eclipse plugin이라 화이트박스임.
te P149	2020 VerHealth: Vetting Medical Voice Applications through Policy Enforcement	4 FH Shezan, H Hu, G Wang, Y Tian	dl.acm.org	TRUE		android 대상이 아님, alexa 대상
P154	2020 Source-Codeless Testing for Android Apps	C Escobar-Velasquez	pro.eeei.erolaxeeei	TRUE		1, APK만을 이용하여 애플리케이션에 대한 테스팅을 진행하기 위한 연구임, 2, 컨퍼런스 논문이라 evaluation 관련은 나와있지 않을
87						3. 애플리케이션에 대한 자동화된 테스트 연구. 테스트 항목은 다양할(버그디렉션, 에너지, 퍼포먼스, 보안 등)
88 P164	2020 A Systematic Literature Review and Quality Analysis of Javascript Malware Detection	2 MF Sohan, A Basalamah	pro.eeei.erolaxeeei	TRUE		android 대상이 아님
te P165	2020 Detecting and explaining self-admitted technical debts with attention-based neural networks	1 X Wang, J Liu, L Li, X Chen, X Liu, H Wu	dl.acm.org	TRUE		android 대상이 아님
P167	2020 Static asynchronous component misuse detection for Android applications	1 L Pan, B Cui, H Liu, J Yan, S Wang, J Yan	dl.acm.org	TRUE		1. AsyncChecker를 개발. Androlic이라는 도구를 이용해서 만들었다고 함(ㅋㅋ이 도구가 동일저자 도구임. 도구 열심히 만들어놓고 눈문 재랑으로 편하게 쓰기!) 2.AsyncBench라는 벤
. 107	Edeb diatic asymptotic transfer of Android applications	T E Tall, D Col, IT Elo, 5 Tall, 6 Wally, 5 Tall	<u>ulacili.old</u>	IIIOL		나 스타 5개) 3. Asynchronous programming을 하기 위한 AsyncTask 라는 도구가 많이 쓰이는데, 여기에서 발생할 수 있는 취약점.
P179	2020 MADFU: An Improved Malicious Application Detection Method Based on Features Uncertainty	3 H Yuan, Y Tang	mdpi.com	TRUE		1. detection model based on features uncertainty (IMAPEU). input(permission)과 output(label)간의 관계를 logistic regression를 통해 분석하였다. 또한 Markov chain Monte Carlo (
	EVEN MADE OF ALL IMPROVED MELICIONS Application Detector Metitod based on Fedures Office Itality	3 ITTOMIT, I TAILING	mod.com	IIIOL		의 uncertainty를 해결하였다. 2. SIGPID와 Chi-Square에 비해 탐지도가 높음. 3. malware detection
e2 P187	2020 Detection of Ransomware Based on Recurrent Neural Network (RNN)	S Kanumuri, VT Kantipudi, AVA Mary…	Springer	TRUE		The distriction of the state of
88 P188	2020 Study on Program Partitioning and Data Protection in Computation Offloading	E Lee, 8 Pak	koreascience.or.kr	TRUE		한국는문!! static analysis 사용하지 않음.
84 P231	2020 코드 오프로딩 환경에서 프로그램 분할과데이터 보호에 대한 연구	이윤영, 박수희	ktsde.kips.or.kr	TRUE		ENCERT Static analysis Are or Are Grant General Static analysis Are or Are Grant General Grant Gener
				TRUE		
86 P245	2020 Comparing assembler procedures by analyzing sequences of opcodes	N Pejić, M Cvetanović, Z Radivojević	scindeks.ceon.rs	TRUE		Andreid static analysis (IRT) OR
88 P43	2020 Maddroid: Characterizing and detecting devious ad contents for android apps	26 T Liu, H Wang, L Li, X Luo, F Dong, Y Guo	dl.acm.org			Android static analysis 사용하기 않음.
97 P50	2020 A review of android malware detection approaches based on machine learning	48 K Liu, S Xu, G Xu, M Zhang, D Sun, H Liu	ieeexplore.ieee.org	TRUE		서비이 트로 12 이용. 양설 배함]
P59	2020 From needs to actions to secure apps? the effect of requirements and developer practices on app	13 C Weir, B Hermann, S Fahl	usenix.ora	TRUE		1. 인동서 검증의 정확한 사용을 확인하기 위해서 MalloDroid 이용. OPAL 프레임워크를 이용해서 앱 내부에 존재하는 https://linearchites.upa. 암호관련 취약점을 탐지하기 위해서는
	security					유출 탐지금 위해서는 FlowDroid로 이용한 source-target 간의 분석을 하였음.
						2. MalloDroid, CogniCrypt, FlowDroid를 실제 앱에 적용됐을 때의 실패가 얼마나 있었는지가 잘 나옴. 실제 앱은 인터뷰 대상자들이 보내준 앱이기 때문에 벤치마크 등은 비교 불가능.
80						3. SSL Security, Crypto- graphic API Misuse, and Privacy Leaks. 취약점 탐지
88 P60	2020 A survey of Android malware detection with deep neural models	35 J Qiu, J Zhang, W Luo, L Pan, S Nepal···	dl.acm.org	TRUE		서베이 는문. 3. 딥러닝!! DL-based Android malware detection and classification
100 P64	2020 Scalable analysis of interaction threats in IoT systems	14 M Alhanahnah, C Stevens, H Bagheri	dl.acm.org	TRUE		android 대상이 아님
101 P80	2020 A Framework for producing effective and efficient secure code through malware analysis	6 AK Pandey, A Tripathi, M Alenezi…	researchgate.net	TRUE		android 대상이 아닐
102 P84	2020 Androzooopen: Collecting large-scale open source android apps for the research community	8 P Liu, L Li, Y Zhao, X Sun, J Grundy	dl.acm.org	TRUE		"Androzoo는 안드로이드 데이터셋임. 아주 유명하고 유용한 것으로 보임 Android static analysis 사용하지 않음."
P86	2020 Borrowing your enemy's arrows: the case of code reuse in android via direct inter-app code	8 J Gao, L Li, P Kong, TF Bissyandė, J Klein	dl.acm.org	TRUE		□ 1. DICIDer를 개발. Soot 와 FlowDroid에 기반하고 있을. Soot 중에서 더 정확한 SPARK 알고리즘을 사용하였음. DICI 관련 API를 스캔하고 ~~~ 하는 것은 독자적으로 개발한 것으로 보
108	invocation					에 대한 첫 논문이라 비교대상 없음. 3. Direct inter-app code invocation을 탐지하는 도구를 개발. functionalities (either entire Java classes, methods or object fields) implemented
P88	2020 A longitudinal study of application structure and behaviors in android	11 H Cai, BG Ryder	pro, eeei, erolaxeeei	TRUE		1. hybrid approach. Static code analysis(Soot + Flowdroid + GATOR)를 이용. callback control-flow 분석을 위해서는 GATOR를 이용하였고, 컴포넌트 타입에 대한 정보(?)를 얻기
104						2개가 Soot 기반이니까 Soot도 이용, 2. 비교없을, 3. 안드로이드 앱 구조와 행동에 대해 이해하기 위해서 《ICC에 대한 동계나, 기타 동계는 이 논문 참고하면 종을 듯.
P89	2020 {FIRMSCOPE}: Automatic uncovering of privilege-escalation vulnerabilities in pre-installed apps in	10 M Elsabagh, R Johnson, A Stavrou, C Zuo···	usenix.org	TRUE		1. 현재의 static analysis가 가지고 있는 문제들을 자기만의 방법으로 해결할, C1 ~ C5까지, 아주 종은 논문인듯!!!
106	android firmware					2. FlowDroid, Amanddroid, DroidSafe보다 앞섬, DroidBench, DIALDroid-Bench benchmark도 이용함. 3. 펌웨어에 미리 설치된 특권 앱들에 대해서 분석하기 위해서
108 P91	2020 A self-configuring and adaptive privacy-aware permission system for Android apps	5 GL Scoccia, M Autili, P Inverardi	pro.eeei.erolgxeeei	TRUE		Android static analysis 사용하지 않을.
107 P94	2020 A systematic mapping study on software quality control techniques for assessing privacy in	7 D8 Guamán, JM Del Alamo, JC Caiza	pro.eeei.erolgxeeei	TRUE		
108 P99	2020 Recovering Android Bad Smells from Android Applications	7 G Rasool, A Ali	Springer	TRUE		1. JavaParser(오픈소스)골 이용하고, 2. 25개의 패턴을 탐지하고 실제 앱을 통해 검증할, 비교는 불가능해보이지만 다른 눈문에 비해 패턴이 많다고 할, 3. bad smells를 탐지하고, 제기
109 P102	2020 Automated repair of resource leaks in android applications	3 BN Bhatt, CA Furia	arxiv.org	FALSE		
110 P106	2020 Enabling Mutant Generation for Open-and Closed-Source Android Apps	5 C Escobar-Velásquez	ieeexplore.ieee.org	FALSE		
111 P131	2020 Salsa: static analysis of serialization features	2 JCS Santos, RA Jones, M Mirakhorli	dl.acm.org	FALSE		No Android
112 P132	2020 A Taxonomy for Security Flaws in Event-Based Systems	2 YK Lee, D Kim	mdpi.com	FALSE		open access repository
118 P145	2020 Tool Support for Green Android Development: A Systematic Mapping Study.	2   Fatima, H Anwar, D Pfahl, U Qamar	scitepress.org	FALSE		서베이 는표(systematic mapping study)
114 P163	2020 Detection of Vulnerabilities Related to Permissions Requests for Android Apps Using Machine	D Bassolê, Y Traorê, G Koala, F Tchakountê		FALSE		Mail of the Colonians and Colo
116 P175	2020 SeMA: Extending and Analyzing Storyboards to Develop Secure Android Apps	J Mitra, VP Ranganath, T Amtoft, M Higgins		FALSE		open access repository
			IET IET	FALSE		joben access leptository 1. FlowDroid를 요구사항에 맞게 변경하여, privacy leakage에 사용할 수 있도록 하였다. 기본적으로 hybrid approach인데, static analysis로는 leakage path를 구성하여 privacy proted
-176	2020 Container-Based Privacy Preserving Scheme for Android Applications	H Cui, S Shao, S Niu, W Zhang, Y Yuan	IEI	FALSE		
118 118 D104	2020 Octo Conferencial Marketonia Conf	A Latinahus and E Ustranama	dia and and	ENIOE		다고함, 2, 어떤 앱을 이용했는지 잘 모르겟네요~ 3, privacy leakage detection
117 P184	2020 Safe Configurable Multitenant SaaS	A Leijonhufvud, F Håkansson	diva-portal.org	FALSE		android (III of II)
118 P189	2020 Risk Assessment, Threat Modeling and Security Testing in SDLC	1 AHA Kamal, CCY Yen, GJ Hui, PS Ling	arxiv.org	FALSE		open access XIX
P193	2020 Towards Better Static Analysis Security Testing Methodologies	B Aloraini	uwspace.uwaterloo.ca	FALSE		1. Buffer Error Finder (BEFinder)를 개발 input validation과 representation과 관련된 buffer error가 널리 퍼져있다. 불신뢰하는 인포를 결정하는 'validator'를 찾아보는 도구임. Source
118		10.0				source와 sink가 실현불가능한 경로를 가진 경우에는 제거해주었다. 2. 현재 존재하는 분석도구가 1.에서의 취약점 담지 못함 3. 통합 분석 프레임워크에서 recall. Input validation? buff
	2020 Empirical testing for establishing benchmarks: process review and comparison between java, kotlin	J8 Espitia Acero	repositorio.uniandes.edu.c			접속불가
	2020 Characterizing the evolution of statically-detectable performance issues of Android apps	D Teerath, M Ivano	search.proquest.com	FALSE		peer review
122 C27	2020 A systematic literature review of android malware detection using static analysis	40 Y Pan, X Ge, C Fang, Y Fan	pro.eeei.erolaxeeei		Start Set	서베이 는문 1. 없음. 2. 없음. 3. 악성코드
C29	2020 EstiDroid: Estimate API Calls of Android Applications Using Static Analysis Technology	4 W Fan, D Zhang, Y Chen, F Wu, Y Liu	pro.eeei.erolaxeeei	0	Start Set	1. Soot, Jimple 이용, 분석과정 알고리즘은 독자적으로 개발하였음. 2. 동적 분석 도구인 DroidInjector 을 세팅해서 API 호출의 정확도로 측정함. 3. API 호출을 측정, 평가, 기존의 분석
128						다면, 이 는문은 앱의 실행 호름에서 각각의 API가 등장하는 빈도(수)에 대한 정보를 준다.
124 C30	2020 DroidDet: effective and robust detection of android malware using static analysis along with rotation	4 W Fan, D Zhang, Y Chen, F Wu, Y Liu	pro.eeei.erolaxeeei	0	Start Set	1. 머신러닝 기법 종. Rotation Forest (RF) 기법을 이용하여 분석하였음. 2. 가장 최신의 SVM 모델을 이용한 악성앱 탐지 논문에 비해 황상을 이루었음. 3. malware 탐지
124 630	2020 DibioDet. effective and robust detection of android manware using static analysis along with rotation	#TW Fan, Dizhang, TiChen, Fiwu, YiLlu	DIO. BBB BIOIDXBBB.	U	Oldit Off	h, 마면너는 기본 술, notation roles (nr) 기본들 이름이어 도둑이었을, 3, 가장 최연국 6VM 고본을 이름한 작성을 들시 도쿄에 마에 당성을 이루었는, 3, Malwais 당시

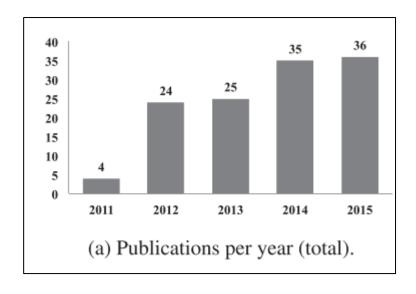


### ~2015년: 연구 분야의 탄생

### 연구의 목적은 Privacy leaks, Vulnerability



#### • 연구 분야의 성장





### ~2015년: 연구 분야의 탄생

■ Android static analysis 중에서 taint analysis가 가장 많은 비율을 차지

Techniques	Publications	Percentage <sup>a</sup>
Abstract interpretation	Anadroid [69] Cortesi et al. [122], Hopper [79], Julia [4], Lu et al. [89], Mann et al. [93], Rocha et al. [156], Scandal [114],	6.5%
Taint analysis	AAPL [61], Amandroid [68], Anadroid [69], AndroidLeaks [80], Apparecium [83], AppAudit [85], AppCaulk [86], AppContext [88], Apposcopy [92], AppSealer [94], Bastani et al. [99], Brox [103], Capper [105], CHEX [8], Cortesi et al. [122], CredMiner [126], DescribeMe [129], Dflow+DroidInfer [132], DidFail [36], DroidChecker [140], DroidJust [56], DroidSafe [58], FlowDroid [6], FUSE [70], Gallingani et al. [71], HelDroid [77], IccTA [7], Lotrack [153], Mann et al. [93], MobSafe [96], PermissionFlow [104], SEFA [115], Sufatrio et al. [121], TASMAN [123], TrustDroid [125], Uranine [127], W2AlScanner [131], WeChecker [134]	30.6%
Symbolic execution	ACTEve [63], Applntent [90], ClickRelease [113], Gallingani et al. [71], Jensen et al. [84], SIG-Droid [52], TASMAN [123], W2AIScanner [131]	6.5%
Program slicing	AndroidLeaks [80], Apparecium [83], AppCaulk [86], AppSealer [94], AQUA [142] Brox [103], Capper [105], CredMiner [126], CryptoLint [54], eLens [66], Hopper [79], MobSafe [96], Poeplau et al. [106], Rocha et al. [156], SAAF [111],	12.1%
Code instrumentation	ACTEve [63] Androguard2 [74], android-app-analysis-tool [78], AppCaulk [86], AppSealer [94], AsyncDroid [143], Bastani et al. [99], Brahmastra [146], Capper [105], Cassandra [53], CMA [116], DidFail [36], DroidSafe [58], I-ARM-Droid [152], IccTA [7], Nyx [43], ORBIT [154], Rocha et al. [156], SIF [157], SmartDroid [118], Sufatrio et al. [121], Uranine [127], vLens [44],	18.5%
Type/Model checking	Choi et al. [147], Covert [124] Dflow+DroidInfer [132], DroidAlarm [138], IFT [82], Lu et al. [89], Mann et al. [93], SADroid [112],	6.5%

Droidsafe(NDSS 2015, 473회 인용), Flowdroid(Acm Sigplan Notices 2014, 2135회 인용),
 Amandroid(ACM CCS 2014, 469회 인용), Iccta(IEEE ICSE, 636회 인용),
 Androidleaks(International Conference on Trust and Trustworthy Computing 2012, 794회 인용)



### ~2018년: 연구 발전 climax

- 대표적인 도구들이 자리를 잡기 시작하였음
  - Flowdroid
    - Iccta 와 통합하여 ICC 기능 강화
    - Stubdroid를 이용하여 정확성 및 속도 향상
    - Implicit flow 지원
    - 2014년 이후로 현재까지 유지/보수 진행 중
  - Amandroid
    - ICC 에서의 정확도와 알고리즘 일부를 개선
    - 2014년 이후 성능 개선하여 2018년 논문 재게재, 이후 개발 중단
  - Droidsafe
    - 2016년까지 이후 지원 중단

In NDSS (Vol. 15, No. 201, p. 110).



### ~2018년: 연구 발전 climax

- 대표적인 도구들의 성능을 비교하려는 연구들이 등장함
  - Analyzing the analyzers: Flowdroid/iccta, amandroid, and droidsafe (ACM ISSTA, 2018)
  - Do android taint analysis tools keep their promises? (ACM ESEC/FSE, 2018)
  - Discovering flaws in security-focused static analysis tools for android using systematic mutation (USENIX Security 18)
  - A qualitative analysis of Android taint-analysis results (IEEE/ACS ASE, 2019)



### ~2018년: 연구 발전 climax

- Static analysis의 challenge들을 해결하려는 연구들이 등장함
  - Native 함수
    - Jn-saf: Precise and efficient ndk/jni-aware inter-language static analysis framework for security vetting of android applications with native code (ACM CCS, 2018)
  - Reflection
    - Droidra: Taming reflection to support whole-program analysis of android apps (ACM ISSTA, 2016)
  - ICC(inter-component communication)
    - Effective {Inter-Component} Communication Mapping in Android: An Essential Step Towards Holistic Security Analysis (USENIX Security 2013)
    - Composite Constant Propagation: Application to Android Inter-Component Communication Analysis (IEEE/ACM ICSE, 2015)
    - Iccta: Detecting inter-component privacy leaks in android apps (IEEE/ACM ICSE, 2015)

DFRC - Korea-based Digital Forensics Think Tank



### 2019년~: 다양한 분야로의 발전

### Taint analysis 기법의 발전

- An efficient approach for taint analysis of android applications (Computer & Security, 2021)
- Raicc: Revealing atypical inter-component communication in android apps (IEEE/ACM ICSE, 2021)

#### ■ 취약점 발견

- Poster: On Vulnerability Evolution in Android Apps (IEEE/ACM ICSE, 2018)
- FIRMSCOPE: Automatic uncovering of privilege-escalation vulnerabilities in pre-installed apps in android firmware (USENIX Security, 2020)
- Borrowing your enemy's arrows: the case of code reuse in android via direct inter-app code invocation (ACM ESEC/FSE, 2020)



### 2019년~: 다양한 분야로의 발전

### Energy Leak, Wake lock Detection

- Detecting energy bugs in android apps using static analysis (ICFEM, 2017)
- Sentinel: generating GUI tests for Android sensor leaks (IEEE/ACM AST, 2018)
- Wake Lock Leak Detection in Android Apps Using Multi-Layer Perceptron (Electronics, 2021)

### Application 내부의 feature를 파악

- 안드로이드 악성코드 분류를 위한 Flow Analysis 기반의 API 그룹화 및 빈도 분석 기법 (정보보호학회논문지, 2019)
- Deepintent: Deep icon-behavior learning for detecting intention-behavior discrepancy in mobile apps (ACM CCS, 2019)
- RedDroid: Android application redundancy customization based on static analysis (IEEE ISSRE, 2018)

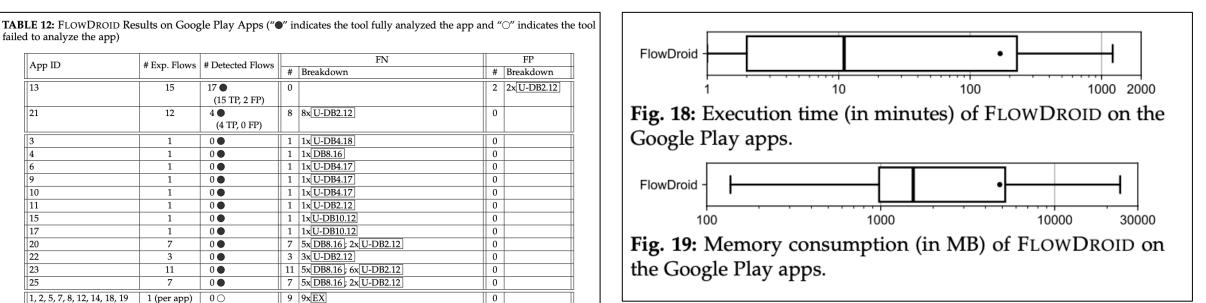
DFRC - Korea-based Digital Forensics Think Tank



### Static taint analysis의 한계

- Native 라이브러리의 비중 증가
  - 하이브리드 앱, 프로그레시브 웹 앱(PWA)의 등장
  - Flutter, React Native 등 크래스 플랫폼 프레임워크의 등장
- 정확도 및 시간에서의 문제점

App ID	# Erm Elevis	# Detected Flows	FN			FP	
App 1D	# Exp. Flows	# Detected Flows	#	Breakdown	#	Breakdown	
13	15	17 •	0		2	2x U-DB2.12	
		(15 TP, 2 FP)					
21	12	4 •	8	8xU-DB2.12	0		
		(4 TP, 0 FP)					
3	1	0 •	1	1x[U-DB4.18]	0		
4	1	0 •	1	1x DB8.16	0		
6	1	0 •	1	1x[U-DB4.17]	0		
9	1	0 •	1	1x[U-DB4.17]	0		
10	1	0 •	1	1x[U-DB4.17]	0		
11	1	0 •	1	1x[U-DB2.12]	0		
15	1	0 •	1	1xU-DB10.12	0		
17	1	0 •	1	1xU-DB10.12	0		
20	7	0 •	7	5x DB8.16; 2x U-DB2.12	0		
22	3	0 •	3	3x[U-DB2.12]	0		
23	11	0 •	11	5x[DB8.16]; 6x[U-DB2.12]	0		
25	7	0 •	7	5x[DB8.16]; 2x[U-DB2.12]	0		
1, 2, 5, 7, 8, 12, 14, 18, 19	1 (per app)	0 🔾	9	9xEX	0		
16	3	0 🔾	3	3xEX	0		
24	7	0 🔾	7	7xOM	0		



Zhang, J., Wang, Y., Qiu, L., & Rubin, J. (2021). Analyzing android taint analysis tools: FlowDroid, Amandroid, and DroidSafe. IEEE Transactions on Software Engineering. 35



### Static taint analysis의 활용

- 머신러닝에 활용하기 위한 feature 추출
  - 기존 도구들 활용
  - 최근에 활발하게 연구되고 있음
- Energy leak Detection
- Software debloating, Application redundancy



### **Apply to Digital Forensics?**

- Source와 Sink의 개념을 응용
  - Source: 수집하고자 하는 증거(이름, 전화번호, 이메일, IMEI, Mac address ...)
  - Sink: 데이터의 저장 위치(로컬 데이터베이스, 원격지 URL ...)

#### ■ 선행 연구

- Automated forensic analysis of mobile applications on Android Devices (Digital Investigation, 2018)
- EviHunter: identifying digital evidence in the permanent storage of android devices via static analysis (ACM CCS, 2018)
- Android Encryption Database Forensic Analysis Based on Static Analysis (ACM CSAE, 2020)
- LogExtractor: Extracting Digital Evidence from Android Log Messages via String and Taint Analysis
   (DFRWS USA 2021)

DFRC - Korea-based Digital Forensics Think Tank



### **Apply to Digital Forensics?**

- Static taint analysis를 통해 데이터 저장 경로, 증거 종류를 파악
  - 자주 쓰이는 애플리케이션에 대해서 미리 데이터베이스 구축
  - 사용자 디스크 이미지에 대해서, 사전에 알고 있는 경로로부터 데이터 추출
    - 분석가: 시간 단축, 분석의 정확도 증가 // 피의자: 불필요한 개인정보 유출 방지

App Evidence Database									
App Package Name	Evidence File Path	Evidence Type							
com.app1	/data/data/com.app1/files/a.txt	location, time							
com.app1	/data/data/com.app1/databases/m.sqlite	visited URL							
com.app2	/data/data/com.app2/shared_prefs/b.xml	text input							
com.app2	/data/data/com.app2/cache/crash.bin	time							
File System Image of Suspect's Device Matcher Report									





### **Digital Forensic Research Center**

Graduate School of Information Security, Korea Univ.

hunjison@korea.ac.kr

Questions?



