안드로이드 앱 분석

팀 기반의 설계 프로젝트

2019.03.27
박민재
souling4you@gmail.com

안드로이드 앱 분석

❖ 안드로이드 앱 빌드 과정

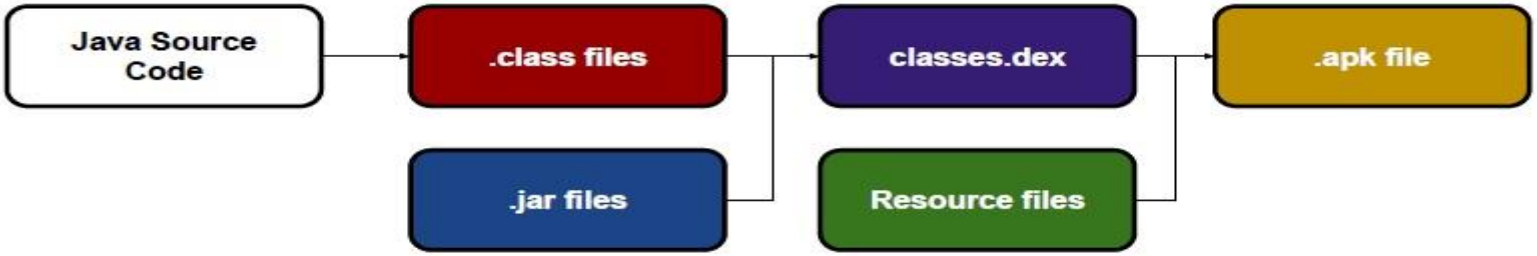


Fig. 1: Workflow of Android compilation.

출처(논문): Who Changed You Obfuscator Identificaton for Android

- 개발자는 Java를 이용하여 앱을 빌드
- Java source code는 Java compiler에 의해 .class 파일로 컴파일됨
- .class 파일은 표준 Java bytecode를 포함
- Java bytecode는 classes.dex로 컴파일됨
- classes.dex는 모든 Dalvik bytecode와 .jar file을 포함
- 일반적인 경우 classes.dex 파일은 하나이지만 앱의 크기가 클 경우 여러 개의 dex 파일로 분리됨
- 설치 시점에, Android runtime은 dex2oat 도구를 이용하여 classes.dex 파일을 컴파일
  - 출력 결과는 실행 가능한 코드이며, Dalvik bytecode보다 성능이 뛰어남
- .apk 파일은 dex 파일과 resource 파일을 포함

안드로이드 앱 분석

❖ **제임스 아서 고슬링 (James Arthur Gosling)**

- 캐나다 출신의 소프트웨어 개발자

❖ **Java**

- Write once, run anywhere.
- 썬 마이크로시스템즈에서 자바(Java) 언어를 발표했는데 오라클(Oracle)이 썬 마이크로시스템즈를 인수함에 따라 현재는 오라클에서 documentation을 제공.
- ① 이식성이 높은 언어
- ② 객체 지향 언어
- ③ 함수적 스타일 코딩을 지원 (ex. Lambda Expressions)
- ④ 메모리를 자동으로 관리
- ⑤ 다양한 운영체제에서 실행되는 프로그램을 개발
- ⑥ 멀티 스레드(Multi-Thread)를 쉽게 구현
- ⑦ 동적 로딩(Dynamic Loading)을 지원
- ⑧ 막강한 오픈소스 라이브러리가 풍부

출처(책): 이것이 자바다

안드로이드 앱 분석

❖ 자바 가상 기계(JVM)

- 운영체제는 자바 프로그램을 바로 실행할 수 없는데, 그 이유는 자바 프로그램은 완전한 기계어가 아닌, 중간 단계의 바이트 코드이기 때문에 이것을 해석하고 실행할 수 있는 가상의 운영체제가 필요하다. 이것이 자바 가상 기계(JVM: Java Virtual Machine)이다.

❖ 클래스

- 클래스에는 객체가 가져야 할 구성 멤버가 선언된다. 구성 멤버에는 필드(Field), 생성자(Constructor), 메소드(Method)가 있다.

```
public class ClassName {
        // 필드
        int fieldName;
        // 생성자
        ClassName() { }
        // 메소드
        void methodName() { }
}
```

출처(책): 이것이 자바다

안드로이드 앱 분석

```
public class ClassName {
        // 필드
        int fieldName;
        // 생성자
        ClassName() { }
        // 메소드
        void methodName() { }
}
```

- **필드**
  - 객체의 고유 데이터, 부품 객체, 상태 정보를 저장하는 곳
- **생성자**
  - new 연산자로 호출되는 특별한 중괄호 { } 블록
  - 객체 생성 시 초기화를 담당
- **메소드**
  - 객체의 동작에 해당하는 중괄호 { } 블록

출처(책): 이것이 자바다

안드로이드 앱 분석

❖ 메소드 오버로딩(Method Overloading)

- 파라미터의 타입, 개수, 순서 중 하나라도 다르면 메소드 이름이 동일하더라도 다른 메소드로 인식

```
public class Java {
    public String stringFromJava() {
        String hello = "Hello from Java";
        return hello;
    }
    public String calculateDec(int a) {
        char i;
        int sum=0;
        for(i=1; i<=a; i++) {
            sum += i;
        }
        String output = String.format("add from 1 to 20: %d",
sum);
        return output;
    }
    public String calculateDec2(double b) {
        String output2 = "?";
        if(b==0)
        {
            output2 = "zero";
        } else {
            output2 = "nonzero";
        }
        return output2;
    }
```

```
public class Java {
    public String Java() {
        String hello = "Hello from Java";
        return hello;
    }
    public String Java(int a) {
        char i;
        int sum=0;
        for(i=1; i<=a; i++) {
            sum += i;
        }
        String output = String.format("add from 1 to 20: %d",
sum);
        return output;
    }
    public String Java(double b) {
        String output2 = "?";
        if(b==0)
        {
            output2 = "zero";
        } else {
            output2 = "nonzero";
        }
        return output2;
    }
```

출처(책): 이것이 자바다

안드로이드 앱 분석

❖ 메소드 오버라이딩(Method Overriding)

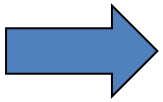
- 상속된 메소드의 내용이 자식 클래스에 맞지 않을 경우, 자식 클래스에서 동일한 메소드를 재정의하는 것

```
public class main {

	public static void main(String[] args) {
		good g1 = new good();
		notgood g2 = new notgood();

		g1.see();
		g2.see();
		maingood(g2);
	}

	public static void maingood(good g) {
		System.out.println("----------------------");
		System.out.println("notgood is typecast");
		g.see();
	}
}

class good {
	void see() {
		System.out.println("good");
	}
}

class notgood extends good {
	@Override
	void see() {
		System.out.println("notgood");
	}
}
```

출처(책): 이것이 자바다

안드로이드 앱 분석

```
public class main {

    public static void main(String[] args) {
        good g1 = new good();
        notgood g2 = new notgood();

        g1.see();
        g2.see();
        maingood(g2);
    }

    public static void maingood(good g) {
        System.out.println("--------------------");
        System.out.println("notgood is typecast");
        g.see();
    }
}

class good {
    void see() {
        System.out.println("good");
    }
}

class notgood extends good {
    @Override
    void see() {
        System.out.println("notgood");
    }
}
```



```
root@pmj:~# java main
good
notgood
--------------------
notgood is typecast
notgood
root@pmj:~#
```

안드로이드 앱 분석

❖ 리플렉션(Reflection)

- 리플렉션은 자바 코드가 필드, 메소드, 로드하는 클래스의 생성자에 대한 정보를 찾고 이용하기 위한 것.

```
1  //Example (1): providing genericity
2  Class collectionClass;
3  Object collectionData;
4  public XmlToCollectionProcessor(Str s, Class c) {
5    collectionClass = c;
6    Class c1 = Class.forName("java.util.List");
7    if (c1 == c) {
8      this.collectionData = new ArrayList();
9    }
10   Class c2 = Class.forName("java.util.Set");
11   if (c2 == c){
12     this.collectionData = new HashSet();
13 }}
14
15 //Example (2): maintaining backward compatibility
16 try {
17   Class.forName("android.speech.tts.TextToSpeech");
18 } catch (Exception ex) {
19   //Deal with exception
20 }
21
22 //Example (3): accessing hidden/internal API
23 //android.os.ServiceManager is a hidden class.
24 Class c =
       Class.forName("android.os.ServiceManager");
25 Method m = c.getMethod("getService", new Class[]
       {String.class});
26 Object o = m.invoke($obj, new String[] {"phone"});
27 IBinder binder = (IBinder) o;
28 //ITelephony is an internal class.
29 //The original code is called through reflection.
30 ITelephony.Stub.asInterface(binder);
```

안드로이드 앱 분석

❖ **제네릭 타입(Generic type)**

- 타입을 파라미터로 가지는 클래스와 인터페이스

```
public class Box<T> {
    private T t;
    public T get() { return t; }
    public void set(T t) { this.t = t; }
}
Box<String> box = new Box<String>();
```

```
public class Box<String> {
    private String t;
    public String get() { return t; }
    public void set(String t) { this.t = t; }
}
```

안드로이드 앱 분석

❖ 안드로이드 4대 컴포넌트

- 1. 액티비티 (Activity)
  - 하나의 화면을 하나의 액티비티로 생각
  - 앱을 구성하는 화면을 액티비티로 구현하고 각각의 화면간에 이동하는 과정은 각각의 액티비티를 필요에 따라 열거나 닫거나 하는 과정
- 2. 서비스 (Service)
  - 백그라운드에서 실행되는 프로세스
- 3. 브로드캐스트 수신자(Broadcast Receiver)
  - 브로드캐스팅은 메시지를 여러 객체에서 전달하는 방법을 의미
  - 전달된 브로드캐스팅 메시지는 브로드캐스트 수신자라는 앱 구성 요소를 이용해서 받을 수 있음
- 4. 콘텐츠 제공자 (Content Provider)
  - 콘텐츠 제공자는 한 프로세스의 데이터에 다른 프로세스에서 실행 중인 코드를 연결하는 표준 인터페이스
  - 안드로이드 시스템에서 앱은 콘텐츠 제공자를 통해 제공하고자 설정한 공유 범위 내에서 네트워크, 데이터베이스, 파일시스템을 제공할 수 있고, 다른 앱은 콘텐츠 해결자(Content Resolver)를 통해서 관련 정보를 얻을 수 있음

출처(책): 안드로이드 애플리케이션 리버스 엔지니어링

안드로이드 앱 분석

❖ 안드로이드 앱 설정 파일

- AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.souli.ossecurity">

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

- package : 앱을 구분하기 위한 이름
- 동일한 디바이스에 동일한 패키지 이름을 갖는 앱은 설치될 수 없음

안드로이드 앱 분석

❖ 안드로이드 앱 설정 파일

- AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.souli.ossecurity">

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

- permission : 앱의 기능 상 필요한 권한을 명시
- Android 6.0 이상에서 dangerous permission은 실행 중에 user에게 요청하도록 프로그래밍 해야함

안드로이드 앱 분석

❖ 안드로이드 앱 설정 파일

- AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.souli.ossecurity">

    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

- application 속성 : 앱에 필요한 컴포넌트 등을 명시
- application 속성 안에 있는 android:name은 처음 시작하는 클래스명을 의미

안드로이드 앱 분석

❖ 안드로이드 Sdk 버전

- 앱 개발 시에 최소 Sdk 버전과 대상 Sdk 버전을 설정할 수 있음
- Apktool로 앱을 디스어셈블하면 "apktool.yml"이라는 파일이 생성되는데, 그 파일을 열면 sdkInfo 정보를 확인할 수 있음

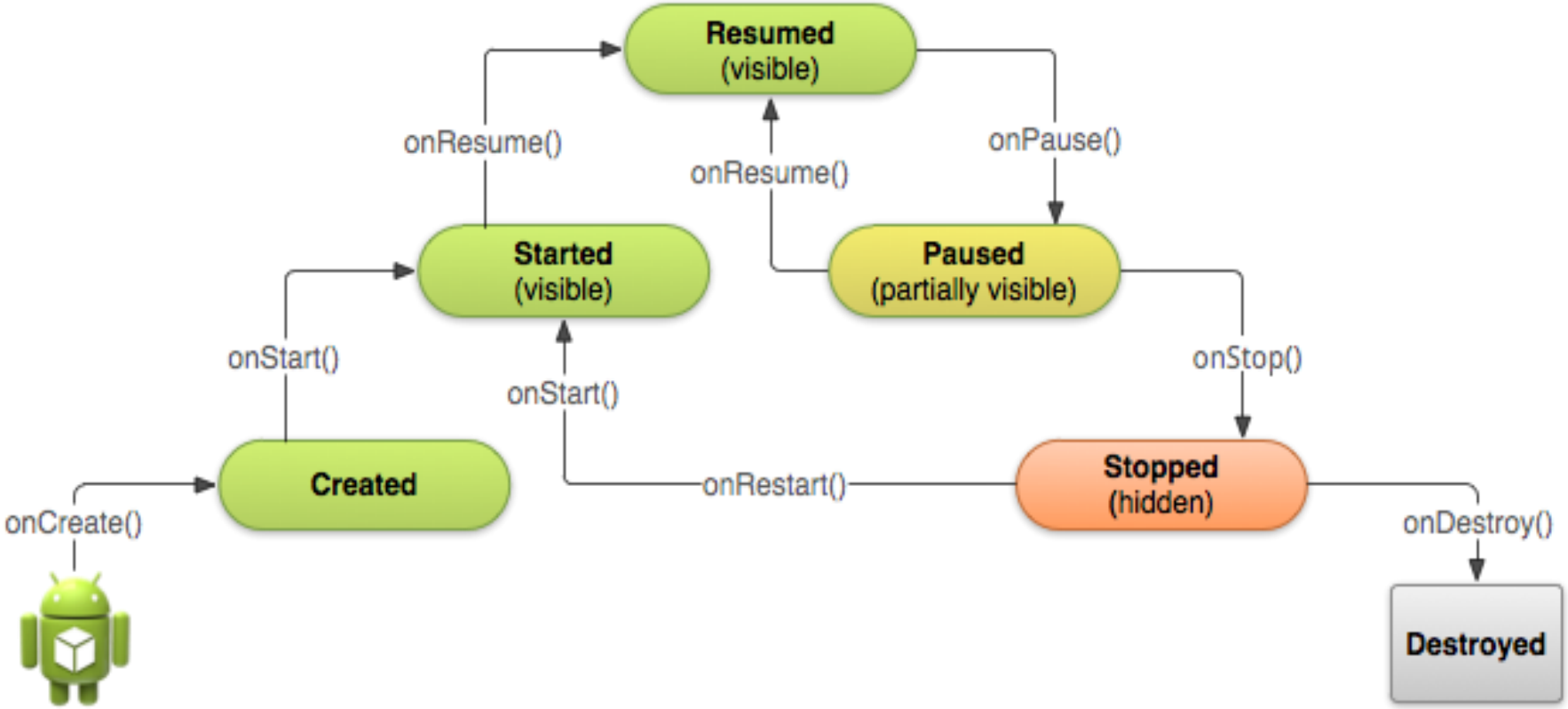
```
sdkInfo:
  minSdkVersion: '4'
  targetSdkVersion: '15'
```

출처(웹사이트): https://mindster.in/blog/evolution-android

안드로이드 앱 분석

❖ 안드로이드 액티비티 수명 주기

- main() 메서드로 앱이 실행되는 다른 프로그래밍 패러다임과는 달리, 안드로이드 시스템은 Activity 인스턴스 수명 주기의 특정 단계에 부합하는 특정 콜백 메서드를 호출하여 해당 인스턴스 내 코드를 실행한다



- https://developer.android.com/training/basics/activity-lifecycle/starting?hl=ko

안드로이드 앱 분석

❖ 안드로이드 디바이스 드라이버

- Binder
  - 최대 크기가 512바이트인 패킷까지만 전송할 수 있다. 따라서 소용량 데이터 전송에 사용한다.
  - string을 쓸 때는 Java의 표준에 맞춰서 length, unicode(2바이트)의 형태로 맞춰준다.
  - 원격지에 있는 함수를 호출한다. 그러기 위해선 객체를 보내주어야 하는데 이를 보통 객체 직렬화(object serialization)라고 하지만, 안드로이드에서는 flatten 이라는 용어를 쓴다.

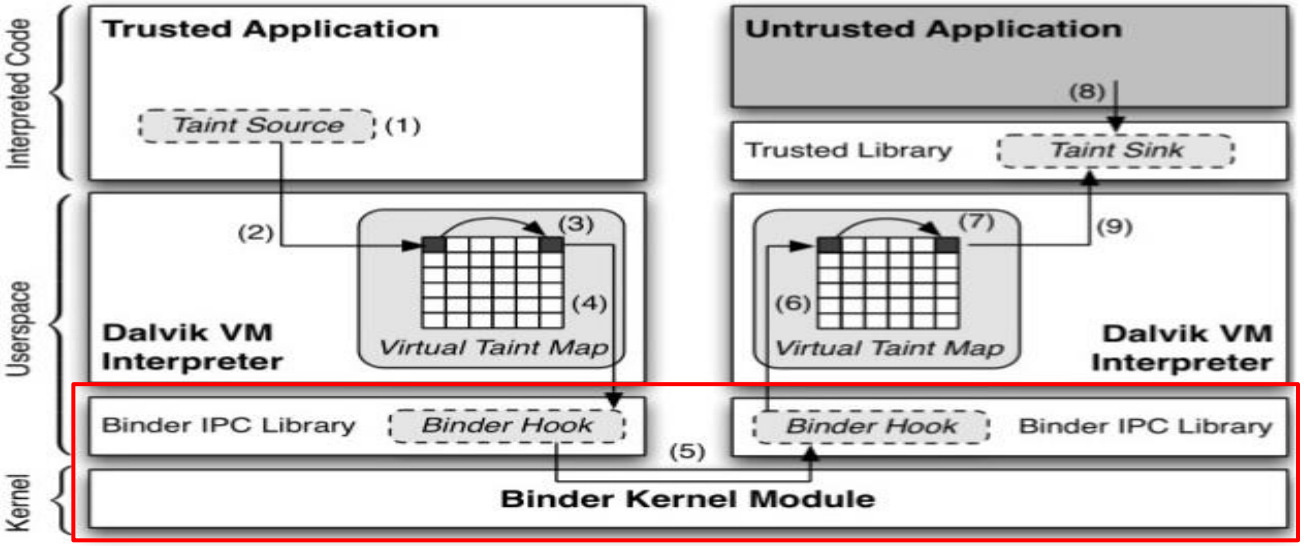
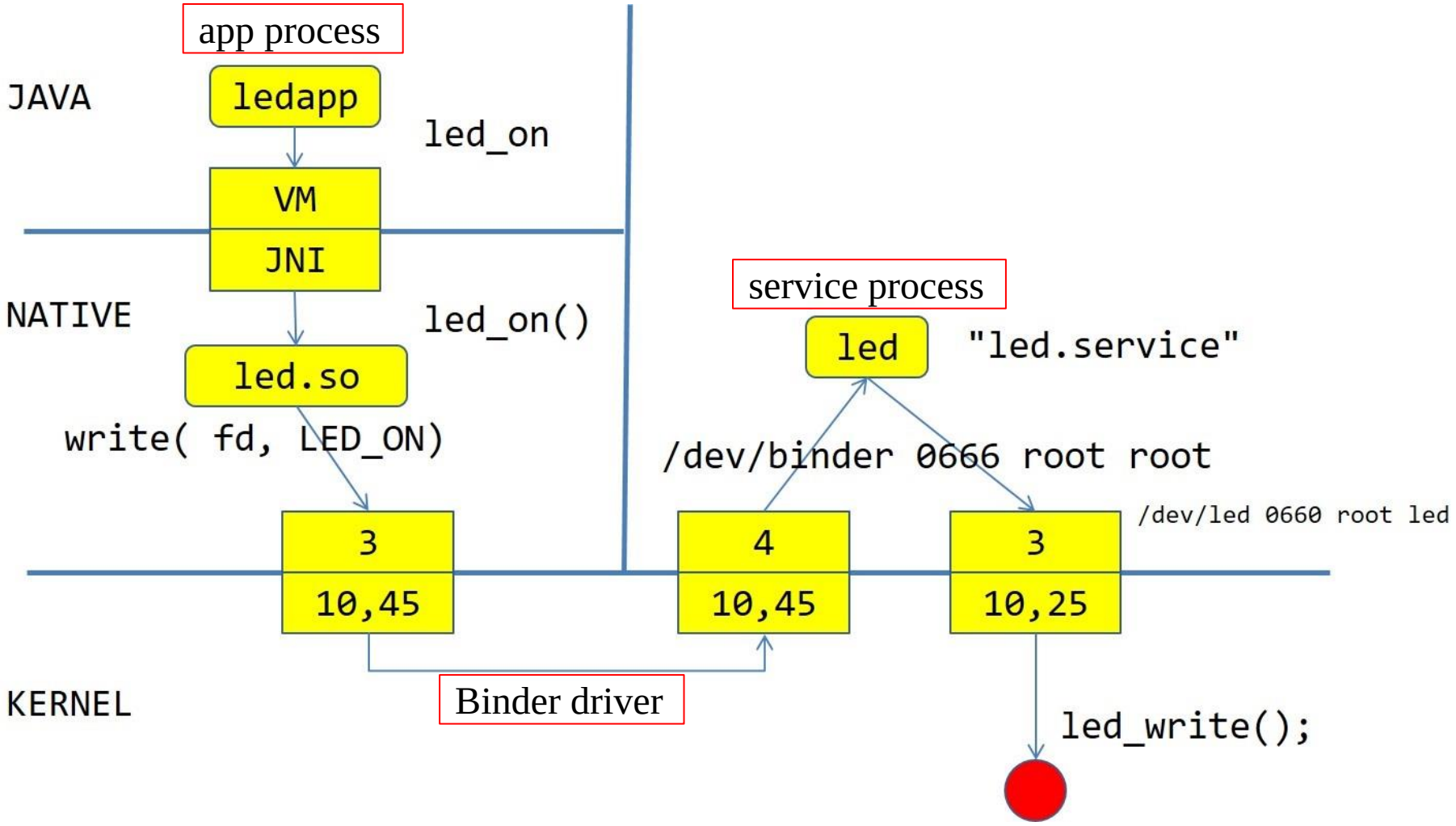


Figure 2: TaintDroid architecture within Android.

출처(논문): TaintDroid An Information-Flow Tracking System for Realtime Privacy

안드로이드 앱 분석

❖ 안드로이드 OS

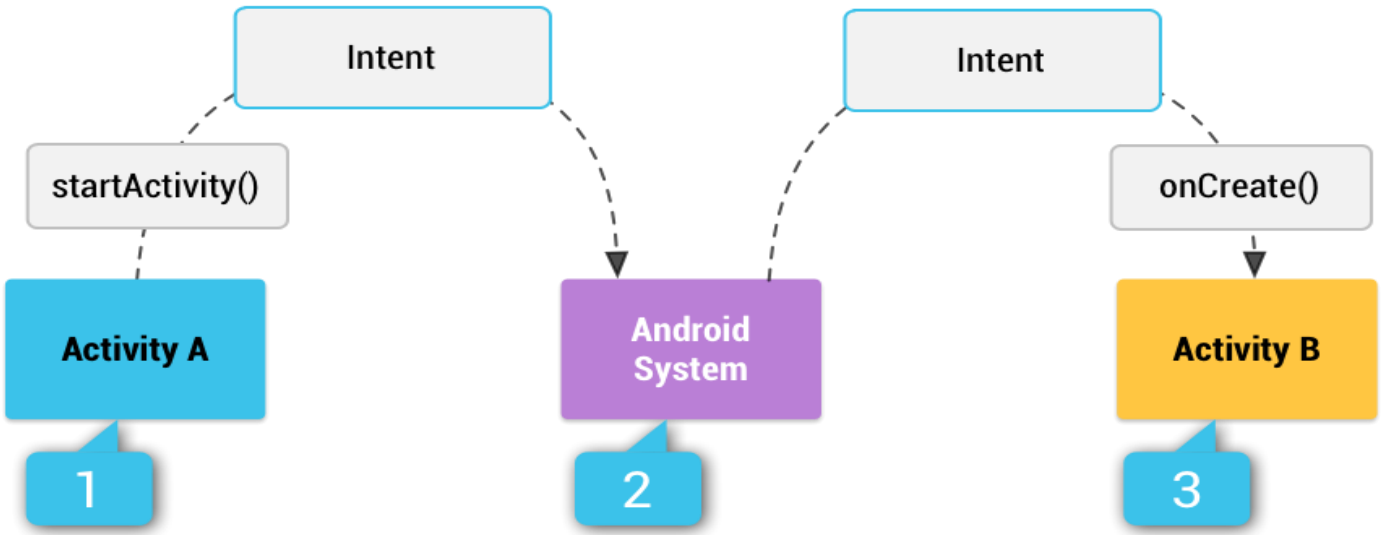


출처(강의): 아임구루 김정인대표님

안드로이드 앱 분석

❖ 안드로이드 인텐트 및 인텐트 필터

- 인텐트(Intent)
  - 일종의 매시지 객체, 이것을 사용해 다른 앱 구성 요소로부터 작업을 요청할 수 있다.
  - 명시적 인텐트 : 시작할 구성 요소를 이름으로 지정(완전히 정규화된 클래스 이름)
  - 암시적 인텐트 : 특성 구성 요소의 이름을 대지 않지만, 그 대신 수행할 일반적인 작업을 선언하여 또 다른 앱의 구성 요소가 이를 처리할 수 있도록 해준다.
- 인텐트 필터(Intent filter)
  - 앱의 매니페스트 파일에 들어 있는 표현으로, 해당 구성 요소가 수신하고자 하는 인텐트의 유형을 나타낸 것

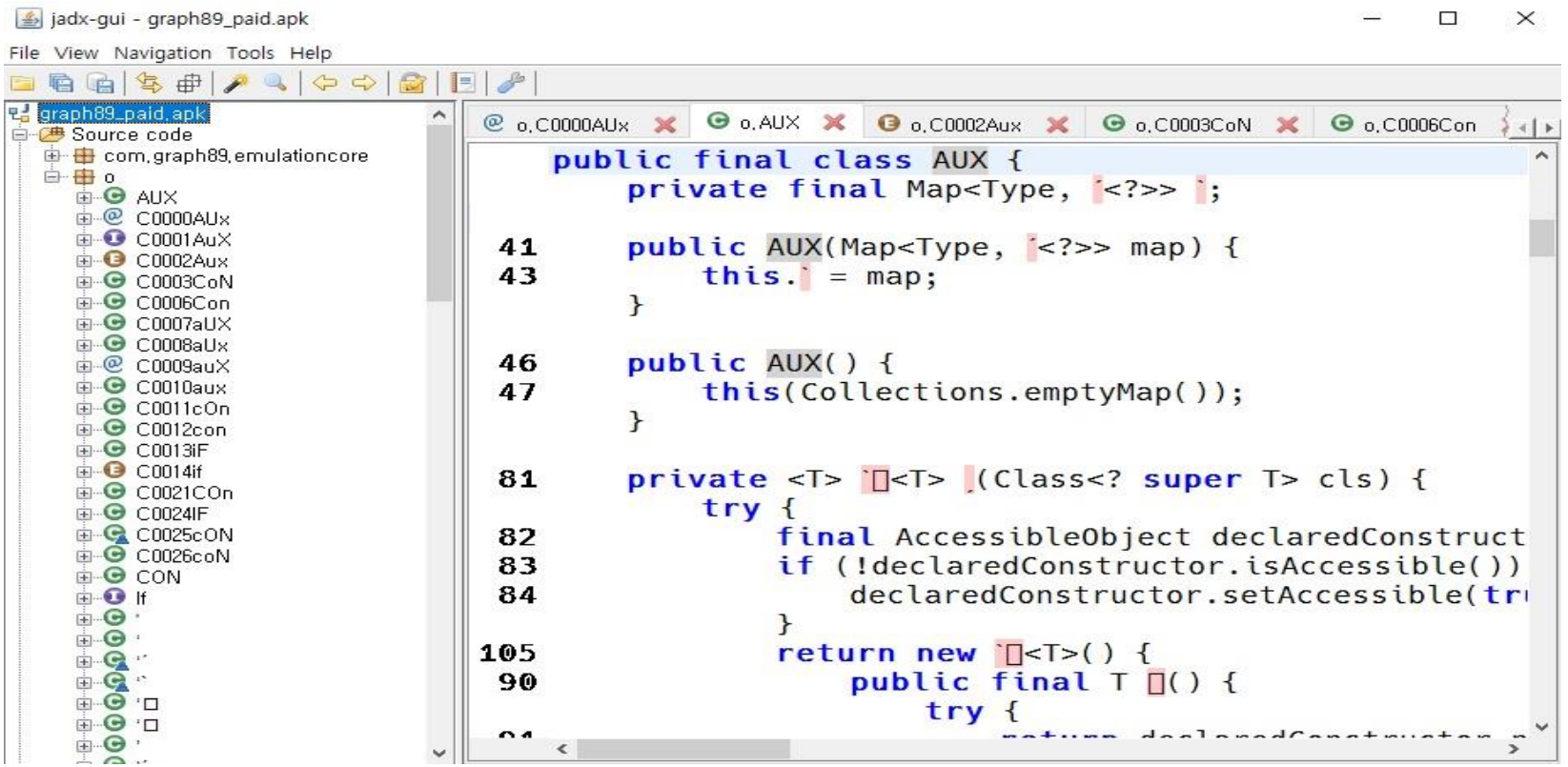


- https://developer.android.com/guide/components/intents-filters?hl=ko

안드로이드 앱 분석

❖ 안드로이드 앱 정적 분석

- Jadx
  - 디컴파일 도구
  - https://github.com/skylot/jadx
  - Gui와 Cli 버전 모두 존재



안드로이드 앱 분석

❖ Virus Total

- https://www.virustotal.com/ko/

Thank You !