

개인 과제 및 실습

2019.03.13
정재민
s17orlax@gmail.com

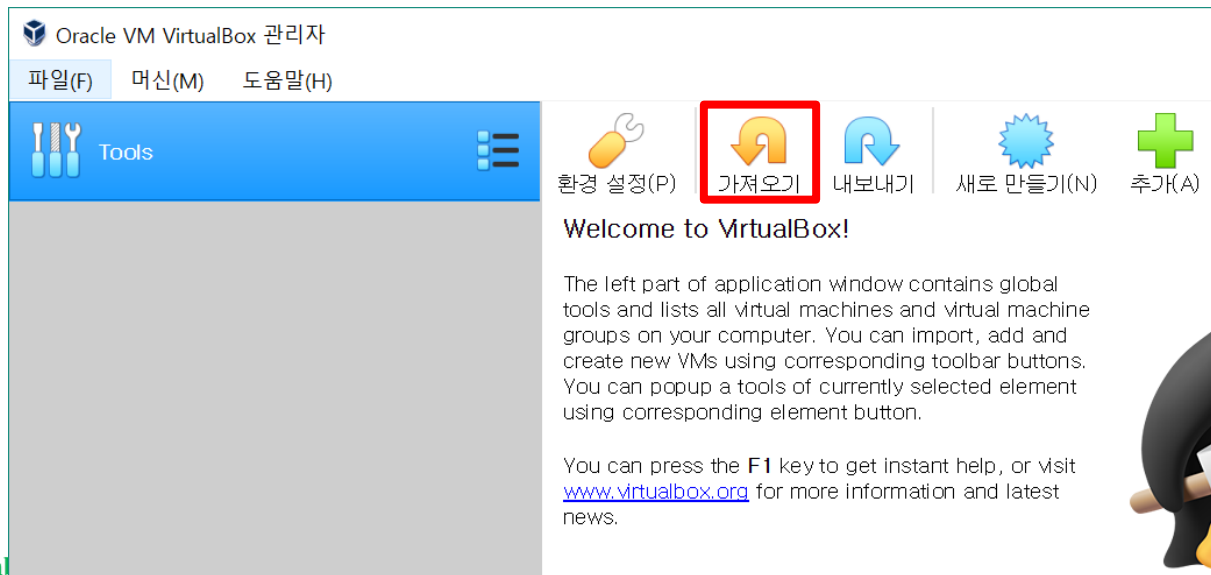
개인 과제 및 실습 - VM 설정

❖ SEED Labs (<http://www.cis.syr.edu/~wedu/seed/>)

- SEEDUbuntu-16.04 수정본에서 진행

❖ Setup

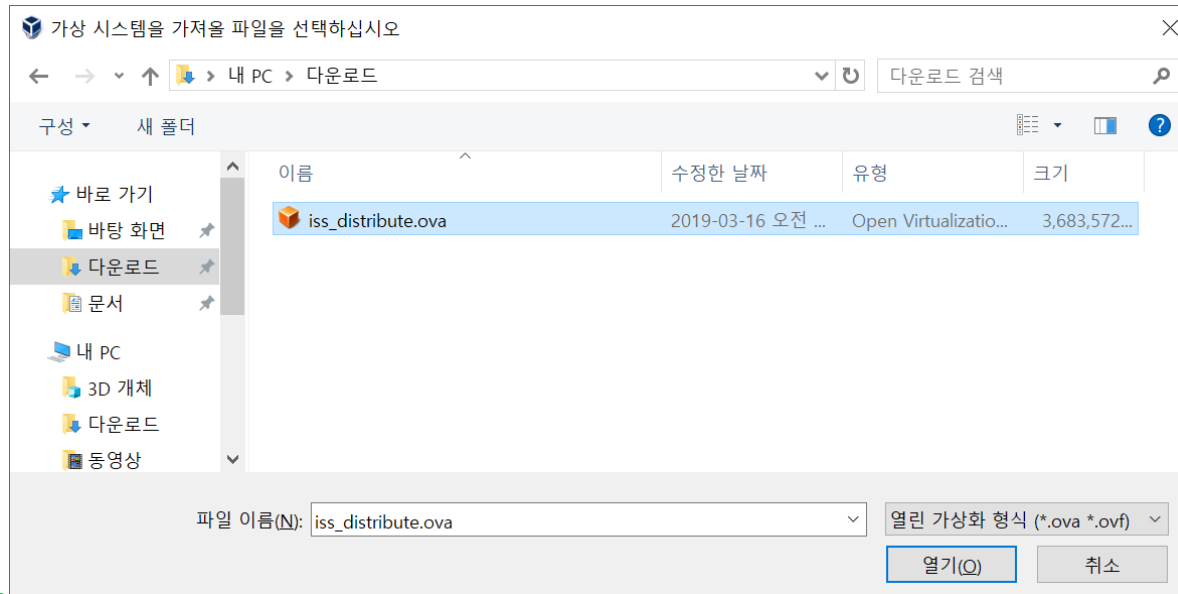
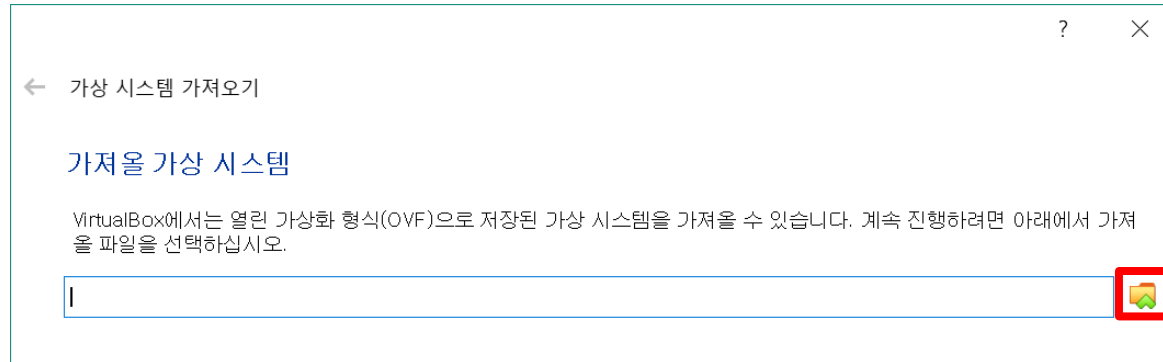
1. VM 다운로드 : 연구실 홈페이지에 링크 게재
2. 다운로드 받은 파일의 압축 해제
3. “VirtualBox” 설치 및 실행
4. “가져오기”



개인 과제 및 실습 - VM 설정

❖ Setup

5. 다운로드 받은 파일 선택 > “다음”



개인 과제 및 실습 - VM 설정

❖ Setup

6. “가져오기”

가상 시스템 설정

아래 목록은 가상 시스템 설명 파일에 나와 있는 가상 머신이며, 이를 VirtualBox로 가져왔을 때의 형태입니다. 보여져 있는 속성을 두 번 누르면 변경할 수도 있으며, 체크 상자를 사용해서 비활성화시킬 수도 있습니다.

가상 시스템 1

이름	iss_distribute
게스트 운영 체제 종류	Ubuntu (32-bit)
CPU	1
RAM	1024 MB
DVD	<input checked="" type="checkbox"/>
USB 컨트롤러	<input checked="" type="checkbox"/>
사운드 카드	<input checked="" type="checkbox"/> ICH AC97

You can modify the base folder which will host all the virtual machines. Home folders can also be individually (per virtual machine) modified.

C:\Users\Ws17or\VirtualBox VMs

MAC Address Policy: Include only NAT network adapter MAC addresses

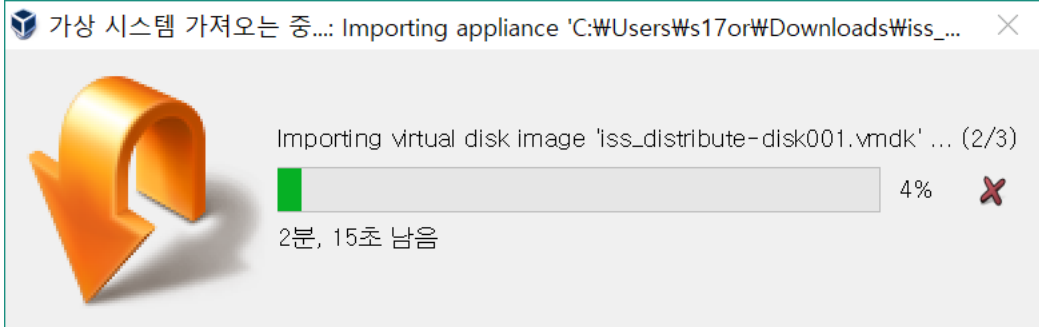
Additional Options: ☒ Import hard drives as VDI

가상 시스템이 서명되지 않았음

기본값 복원

가져오기

취소



개인 과제 및 실습 - VM 설정

❖ Setup

7. “시작”으로 VM 전원 ON

8. ID: seed / PW: dees



HW1

❖ 버퍼 오버플로우 취약점 실습

- 버퍼 오버플로우 취약점 실습 및 분석
- HW1_BOF 폴더

❖ 과제

1. BOF 취약점이 존재하는 소스코드와 실행파일 제공 (*bof.c, bof*)
2. 제공된 exploit 코드(*exploit*)를 이용하여 권한 상승
3. exploit 코드의 체계적인 분석
4. 소스코드 상에서 프로그램의 기능을 변경하지 않으면서 취약점을 보완할 수 있는 방안을 제시할 것

❖ 제출

- 과제 및 실습 수행 내용을 보고서로 제출
- 제출기간 : 3월 18일 ~ 4월 3일

HW2

❖ 암호화 관련 실습

- 간단한 암복호화 수행 및 MD5의 충돌 실습
- HW2_Crypto 폴더

❖ 과제

1. 치환 암호 기법으로 작성된 파일(*ciphertext.txt*)을 복호화
2. 복호화된 *ciphertext.txt*에서 확인할 수 있는 key로 암호화된 이미지를 복호화
3. MD5 hash의 충돌을 확인

❖ 제출

- 과제 및 실습 수행 내용을 보고서로 제출
- 단, 복호화한 이미지가 반드시 보고서에 포함되어야 함
- 제출기간 : 4월 22일 ~ 5월 8일

HW3

❖ SQL Injection 실습

- DB대상 공격을 수행하여 정보 수정 및 관리자 계정 획득
- HW3_SQLInjection 폴더

❖ 과제

1. PW를 모르는 상태로 관리자 계정 획득
2. 내 계정의 봉급 정보 수정
3. 타인의 정보 수정
4. SQL Injection 취약점의 개선 방안 분석

❖ 제출

- 과제 및 실습 수행 내용을 보고서로 제출
- 제출기간 : 5월 29일 ~ 6월 12일

안드로이드 악성 앱 분석 자동화

팀 기반의 설계 프로젝트

2019.03.13

박민재

souling4you@gmail.com

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 악성 앱 분석 자동화

■ 평가 기준

- 구현 : 안드로이드 악성 앱 분석 자동화 프로그램을 구현.
- 보고서 : 구현한 프로그램이 왜 악성 앱 분석에 적합한 프로그램이라고 생각하는지에 대하여 서술하되, 다음 내용을 포함해야 함.
 - 정상/악성 앱의 동작 방식, 정상/악성 앱을 판단하는 기준 (signature)
 - 악성 앱 분석 자동화 프로그램의 알고리즘 개요 및 설명
 - 실험 환경 및 실험 결과. 실험 결과에 대한 분석
 - 구현한 소스코드 및 실행코드 제출. 실행코드 수행 방법 설명.
 - 과제를 수행함에 있어서 어려운 점 또는 건의사항 기재

■ 제출 기한

- 2019년도 5월 1일 (수)

■ 배점 (총 40점)

- 구현 : 15점
- 보고서 : 15점
- 정상 앱, 악성 앱의 동작 방식 분석 결과를 제시 : 10점

안드로이드 악성 앱 분석 자동화

❖ 팀원 구성

- 자유롭게 3~4명씩 팀을 구성
- 팀이 구성된 팀은 조교에게 이메일로 구성원의 명단을 제출
- 팀이 구성되지 않은 학생은 조교가 임의로 팀을 구성
- 팀원과 상의를 통해 가장 효율적으로 과제를 수행할 수 있는 역할을 배분

❖ 데이터셋

- 정상 앱 100개 (F-Droid에서 수집)
- 악성 앱 100개 (AndroZoo에서 수집)
- 팀이 구성된 후에 메일을 조교에게 보내면, 조교가 팀 단위로 배포

❖ 주의사항

- 배포된 앱을 수업 과제 용도 이외의 목적으로 사용하지 말 것.
- 악성 앱 분석 시에, 분석 전용 환경 (예: 가상머신 또는 안드로이드 에뮬레이터)을 이용할 것.
- 실제 안드로이드 기기에 설치하지 않을 것.

안드로이드 악성 앱 분석 자동화

❖ 배포할 안드로이드 분석 자동화 프로그램의 사용 방법

- 가상머신에 Ubuntu-16.04.4-desktop-amd64를 설치
- feature_extractor.zip 파일을 우분투 환경으로 복사
- 다음과 같은 명령어를 수행하여 필요한 파일 설치

```
root@pmj-VirtualBox:~/feature_extractor# pwd
/root/feature_extractor
root@pmj-VirtualBox:~/feature_extractor# chmod 777 setup.sh
root@pmj-VirtualBox:~/feature_extractor# ./setup.sh
```

- 만약, setup.sh 파일이 제대로 실행되지 않을 경우 다음의 명령어를 직접 입력
 - sudo apt-get install python-pip
 - sudo apt install openjdk-8-jre-headless
 - sudo pip install --upgrade pip==9.0.1
 - sudo pip install androguard==3.2

■ 프로그램 실행

```
root@pmj-VirtualBox:~/feature_extractor# pwd
/root/feature_extractor
root@pmj-VirtualBox:~/feature_extractor# python feature_extractor.py ./example -All
```

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 앱 빌드 과정

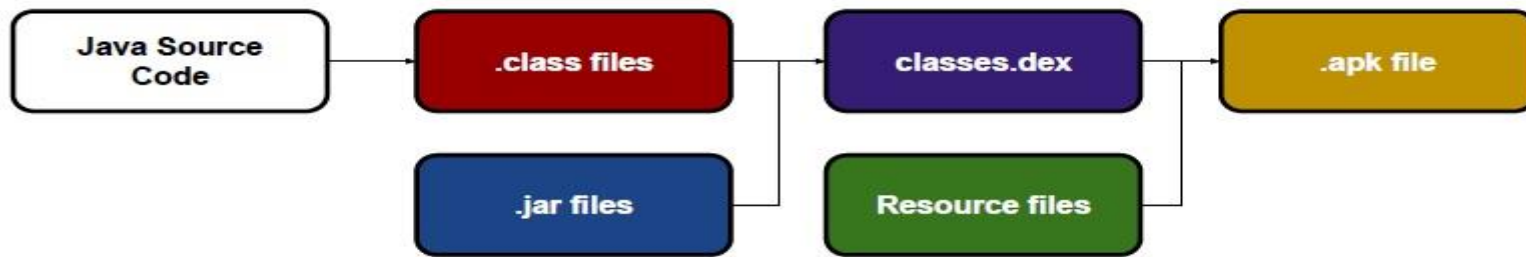


Fig. 1: Workflow of Android compilation.

출처(논문): Who Changed You Obfuscator Identification 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 악성 앱 분석 자동화

❖ 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 악성 앱 분석 자동화

❖ 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 악성 앱 분석 자동화

```
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 악성 앱 분석 자동화

❖ 메소드 오버로딩(Method Overloading)

- 파라미터의 타입, 개수, 순서 중 하나라도 다르다면 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;  
    }  
}
```

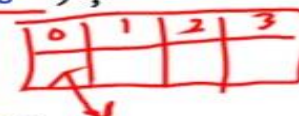
출처(책): 이것이 자바다

안드로이드 악성 앱 분석 자동화

❖ 리플렉션(Reflection)

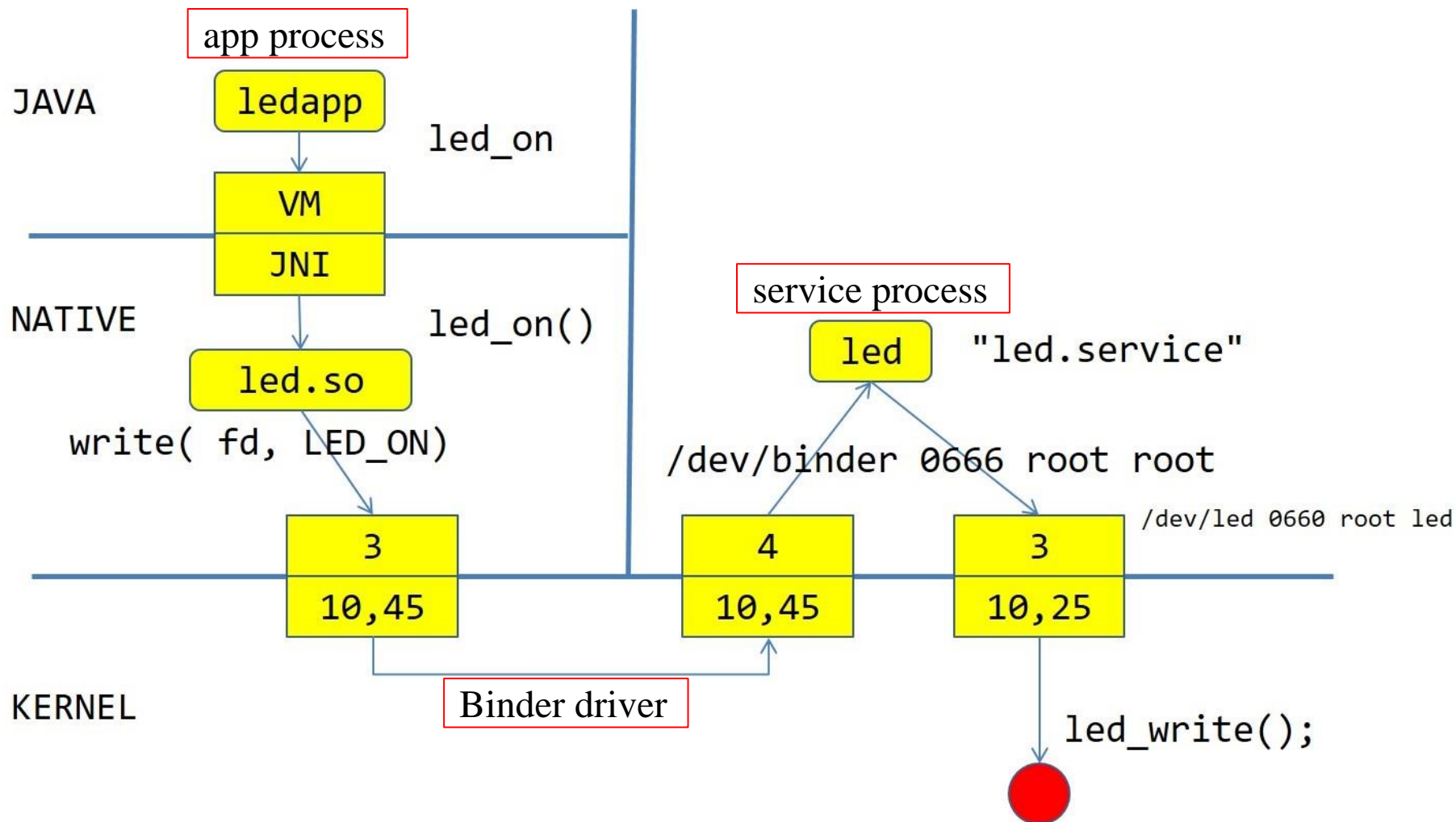
```
1 //Example (1): providing genericity
2 Class collectionClass;
3 Object collectionData;
4 public XmlToCollectionProcessor(Str s, Class c) {
5     collectionClass = c;
6     Class c1 = Class.forName("java.util.List");
7     if (c1 == c) {
8         this.collectionData = new ArrayList();
9     }
10    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 =
25     Class.forName("android.os.ServiceManager");
26 Method m = c.getMethod("getService", new Class[]
27     {String.class});
28 Object o = m.invoke($obj, new String[] {"phone"});
29 IBinder binder = (IBinder) o;
30 //ITelephony is an internal class.
31 //The original code is called through reflection.
32 ITelephony.Stub.asInterface(binder);
```

보름



안드로이드 악성 앱 분석 자동화

❖ 안드로이드 OS



출처(강의): 아임구루 김정인대표님

기기의 led를 on시키는 device driver

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 4대 컴포넌트

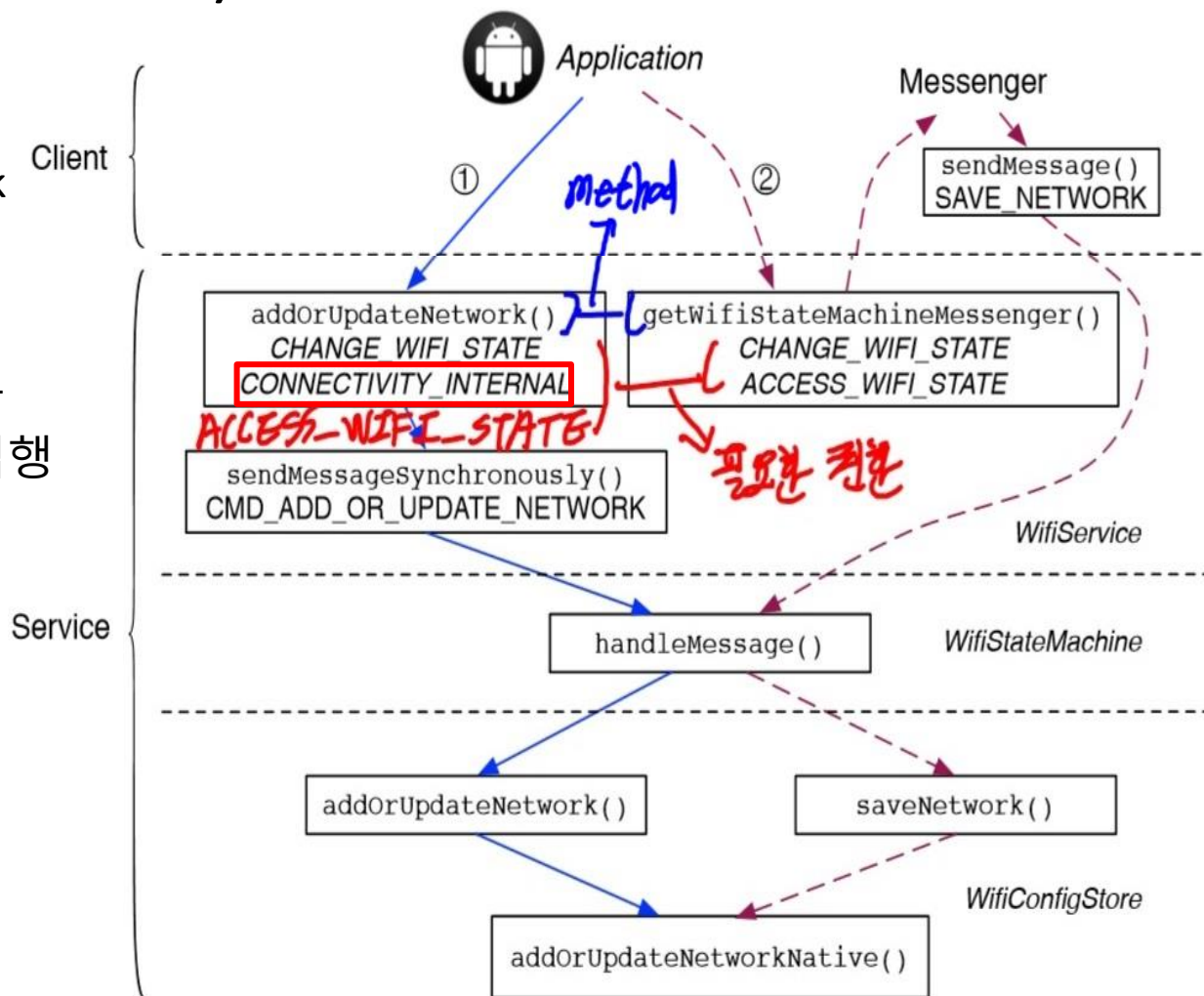
- 1. 액티비티 (Activity)
 - 하나의 화면을 하나의 액티비티로 생각
 - 앱을 구성하는 화면을 액티비티로 구현하고 각각의 화면간에 이동하는 과정은 각각의 액티비티를 필요에 따라 열거나 닫거나 하는 과정
- 2. 서비스 (Service)
 - 백그라운드에서 실행되는 프로세스
- 3. 브로드캐스트 수신자(Broadcast Receiver)
 - 브로드캐스팅은 메시지를 여러 객체에서 전달하는 방법을 의미
 - 전달된 브로드캐스팅 메시지는 브로드캐스트 수신자라는 앱 구성 요소를 이용해서 받을 수 있음
- 4. 콘텐츠 제공자 (Content 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 악성 앱 분석 자동화

❖ 안드로이드 퍼미션(Permission)

- permissions check
- UID check
- package name check
- thread status check
- permissions check는 기본적인 보안 정책 집행



출처(논문): Kratos Discovering Inconsistent Security Policy Enforcement in the Android Framework

안드로이드 악성 앱 분석 자동화

❖ 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 악성 앱 분석 자동화

❖ 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 악성 앱 분석 자동화

❖ 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 악성 앱 분석 자동화

❖ 안드로이드 데몬 or 프로세스

■ init process

- 최초 부팅시 load되는 데몬으로써 pid 1 을 갖는다.
- init.rc 를 parsing하여 action/command를 수행하고 service를 기동시킨다.
- am.ExecuteOneCommand()로 command를 수행
- class_start로 service를 실행
- 자식 프로세스가 죽었을 경우 init 프로세스로 signal이 전달되고, init은 handle_signal을 호출하여 자식 프로세스를 처리한다.

■ property

- property란 전체 프로세스가 볼 수 있는 시스템 전역 변수이다.
- 보는 기능 : 모든 프로세스가 가능, 세팅 기능 : init만 가능
- property에는 ctl 과 trigger가 있다.
- ctl은 stop, start, restart로 서비스를 제어 가능하고, trigger는 변경했을 시 자동으로 코드를 실행시킨다.

■ ueventd

- 하드웨어가 추가되었을 때, 디바이스 특수 파일을 생성하거나 제거하는 데몬
- ueventd.rc를 parsing하여 /dev 디렉토리 밑에 존재하는 ueventd 파일에 add라고 써서, 커널 레벨에 존재하는 디바이스들을 유저 레벨에 파일로 생성한다.
- /dev는 메모리에 존재하기 때문에 하위 파일들은 시스템이 종료되면 삭제된다.

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 데몬 or 프로세스

■ servicemanager

- 모든 service들은 servicemanager에 등록해야만 유저 앱에서 사용할 수 있다.
- servicemanager는 service 리스트를 제공하고 안드로이드 기기에서 service list 명령어로 확인할 수 있다.
- service check <서비스명>을 이용해서 service가 존재하는지 확인할 수 있다.
- servicemanager는 context manager에 등록된다.

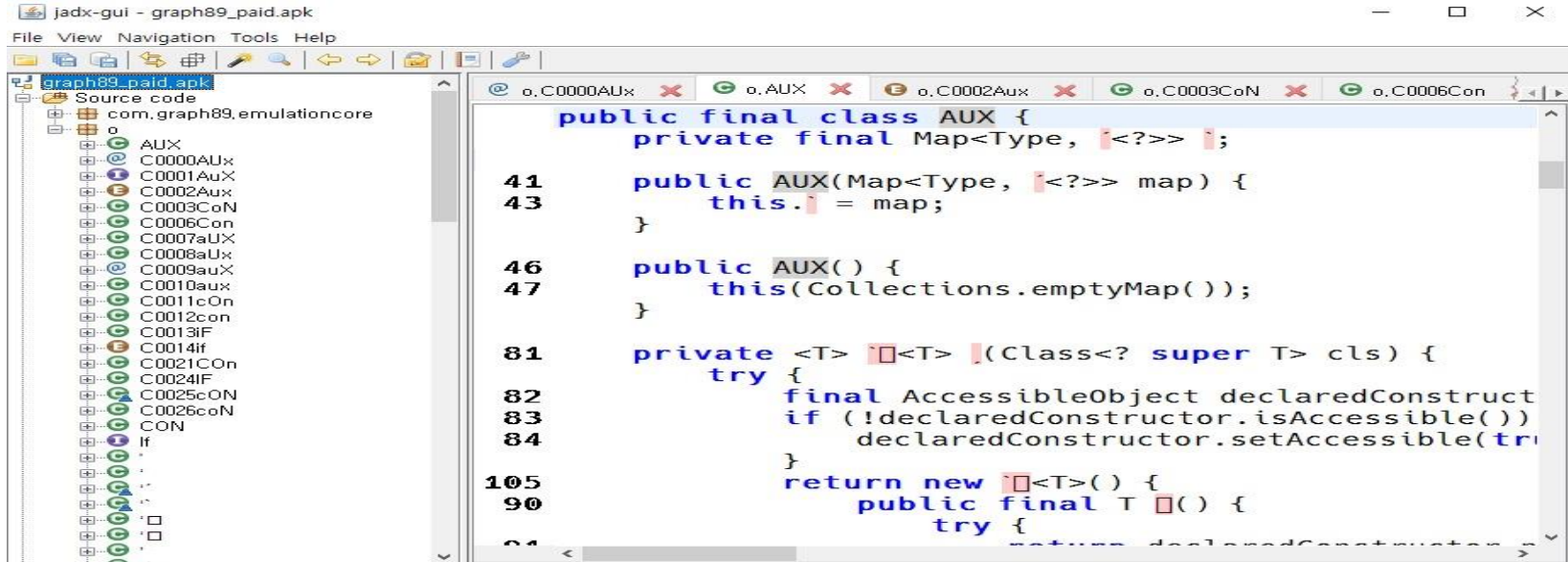
■ zygote


- pid 2
- Java에서 사용하는 class와 resource를 VM상에 미리 load 시켜놓는다.
- init.zygote32.rc 에서 service로 zygote를 띄워준다.
- 프로세스를 fork하고 자신의 memory layout을 복사하여 만든 memory에 SystemServer(java service를 한 번에 동작시키는 서버)를 올려서 동작시킨다. 그 이후에 zygote는 polling 상태로 기다리고 있다가 요청이 오면 사용자 앱의 main을 찾아서 invoke_main을 호출하여 새로 생성된 프로세스(app.apk)를 실행시킨다.

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 앱 분석 도구

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



- 
 - 달빅 바이트코드를 Smali 코드로 변환하는 도구
 - <https://ibotpeaches.github.io/Apktool/install/>

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 악성 앱 분석 자동화 목적

- 안드로이드에는 방대한 양의 애플리케이션이 존재하고, 안드로이드를 대상으로 한 악성 앱이 점차 증가하는 추세
 - 하지만, 이러한 앱을 분석가가 일일이 수동으로 분석하기에는 어려움.
 - 앱에 존재하는 정적인 정보를 이용하여 악성 앱 분석을 자동화하고, 자동화 도구에서 분석하지 못하는 앱만 수동으로 분석한다면, 분석 비용이 낮아질 수 있음.
 - 정적 정보(Static feature) : 앱을 실행하지 않고도 추출할 수 있는 정보
 - 동적 정보(Dynamic feature) : 앱을 실행하면서 추출할 수 있는 정보
- 실제 악성 앱을 대상으로 분석 자동화 프로그램을 구현해봄으로써 Anti-virus 프로그램의 동작 방식을 간접적으로 이해.
 - Anti-virus 프로그램 개발자가 어떠한 관점에서 Anti-virus 프로그램을 작성하려고 할지, 이해할 수 있음

안드로이드 악성 앱 분석 자동화

❖ 안드로이드 악성 앱 탐지 기법

- Permission 기반의 악성 앱 탐지 기법
 - 논문 : Significant Permission Identification for Machine-Learning-Based Android Malware Detection, IEEE TRANSACTIONS ON INDUSTRIAL INFORMATICS, VOL. 14, NO.7, JULY 2018
 - 중요한 Permission을 추출하여 악성 앱을 탐지하기 위한 정보로 사용
 - 머신러닝 기반 분류 알고리즘을 이용
- API 기반의 악성 앱 탐지 기법
 - 논문 : Fine-grained Android Malware Detection based on Deep Learning, 2018 IEEE Conference on Communications and Network Security (CNS): IEEE CNS 2018 - Posters
 - API Call, Permission을 추출하여 Deep Neural Network를 이용하여 학습시키고 이 모델을 탐지에 사용
 - 논문 : Large-scale Malware Automatic Detection Based On Multiclass Features and Machine Learning, CSAE '18, October 22-24, 2018, Hohhot, China
 - API Call, Permission, 4 Components을 추출하여 머신러닝 모델을 이용하여 학습시키고 이 모델을 탐지에 사용
- 다양한 특징 정보를 이용한 악성 앱 탐지 기법
 - 논문 : A Multimodal Deep Learning Method for Android Malware Detection Using Various Features, IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY, VOL. 14, NO. 3, MARCH 2019
 - String, Method opcode, Method API, Shared library function opcode, Permission, Component, Environmental feature

Python, Androguard, Regular Expression

2019.03.13

박민재

souling4you@gmail.com

Python

❖ 귀도 반 로섬(Guido van Rossum, 1956년 1월 31일 ~)

- 네덜란드 출신의 컴퓨터 프로그래머
- 파이썬의 자비로운 종신 독재자(BDFL, Benevolent Dictator for Life)

❖ Python

- 귀도 반 로섬이 1989년 크리스마스 주에 연구실이 닫혀있어서 심심한 김에 만든 프로그래밍 언어

❖ "Life is too short, You need python."

- And the bachelor's course is too short, You need python.



❖ 리스트(list)

- 리스트의 각 요소는 어떤 객체도 될 수 있다.
- 리스트는 변경 가능(mutable)하기 때문에 항목을 할당하고, 자유롭게 수정하거나 삭제할 수 있다.

```
>>> empty_list = []  
>>> empty_list  
[]  
>>> weekdays = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']  
>>> weekdays  
['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday']
```


❖ 튜플(tuple)

- 튜플의 각 요소는 어떤 객체도 될 수 있다.
- 튜플은 불변(immutable)하기 때문에 할당하고 나서, 이를 바꿀 수 없다.

```
>>> empty_tuple = ()
>>> empty_tuple
()
>>> one_marx = 'Groucho',
>>> one_marx
('Groucho',)
>>> type(one_marx)
<class 'tuple'>
>>> marx_tuple = 'Groucho', 'Chico', 'Harpo'
>>> marx_tuple
('Groucho', 'Chico', 'Harpo')
>>> type(marx_tuple)
<class 'tuple'>
>>> marx_tuple = ('Groucho', 'Chico', 'Harpo')
>>> marx_tuple
('Groucho', 'Chico', 'Harpo')
>>> type(marx_tuple)
<class 'tuple'>
```

❖ 딕셔너리(dictionary)

- 항목의 순서를 따지지 않으며, 0 또는 1과 같은 오프셋으로 항목을 선택할 수 없다. 대신 값(value)에 상응하는 고유한 키(key)를 지정한다.
- 딕셔너리는 변경 가능하므로 키-값 요소를 추가, 삭제, 수정할 수 있다.

```
>>> empty_dict = {}
>>> empty_dict
{}
>>> type(empty_dict)
<class 'dict'>
>>> my_dict = {
...     "year": "2018",
...     "month": "12",
...     "day": "13"
... }
>>> my_dict
{'year': '2018', 'month': '12', 'day': '13'}
>>> my_dict["year"]
'2018'
>>> my_dict["month"]
'12'
>>> my_dict["day"]
'13'
```

❖ 셋(set)

- 셋은 수학에서 집합이다.
- 각 키는 유일해야 한다.
- 어떤 것이 존재하는지 여부만 판단하기 위해서는 셋을 사용한다.

```
>>> empty_set = set()
>>> empty_set
set()
>>> type(empty_set)
<class 'set'>
>>> even_numbers = {0, 0, 2, 2, 4, 4, 6, 6, 8, 8}
>>> even_numbers
{0, 2, 4, 6, 8}
>>> odd_numbers = {1, 1, 3, 3, 5, 5, 7, 7, 9, 9}
>>> odd_numbers
{1, 3, 5, 7, 9}
```

❖ print 함수

```
root@pmj:~# python3
Python 3.6.5 (default, May  3 2018, 10:08:28)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello, Python")
Hello, Python
>>> █
```

❖ type 함수

- 대상 타입을 출력

```
>>> type("Hello, Python")
<class 'str'>
```

- "Hello, Python"은 str 클래스의 객체이다.

❖ dir 함수

- 대상의 속성을 출력

```
>>> dir("Hello, Python")
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
 '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__gt__', '__hash__', '__init__',
 '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__', '__ne__',
 '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setattr__',
 '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count', 'encode',
 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isdecimal',
 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper',
 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'replace', 'rfind', 'rindex', 'rjust',
 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title',
 'translate', 'upper', 'zfill']
>>> a = "Hello, Python"
>>> a.__contains__
<method-wrapper '__contains__' of str object at 0x7f9c207a5670>
>>>
```

- `__contains__` 는 특별 메서드
- 특별 메서드는 일반적으로 파이썬 인터프리터가 호출하기 위한 것

❖ generator 함수

- generator 함수는 함수 본체를 포함하는 generator 객체를 생성
- next()를 제너레이터 객체에 호출하면 함수 본체에 있는 다음 yield로 진행하며, next()는 함수 본체가 중단된 곳에서 생성된 값을 평가한다. 마지막으로, 함수 본체가 반환될 때 이 함수를 포함하고 있는 제너레이터 객체는 Iterator 프로토콜에 따라 StopIteration 예외를 발생시킨다.

```
>>> def gen_123():
...     yield 1
...     yield 2
...     yield 3
...
>>> type(gen_123)
<class 'function'>
>>> type(gen_123())
<class 'generator'>
>>> for i in gen_123():
...     print(i)
...
1
2
3
```

❖ generator 함수

- generator 함수는 함수 본체를 포함하는 generator 객체를 생성
- next()를 제너레이터 객체에 호출하면 함수 본체에 있는 다음 yield로 진행하며, next()는 함수 본체가 중단된 곳에서 생성된 값을 평가한다. 마지막으로, 함수 본체가 반환될 때 이 함수를 포함하고 있는 제너레이터 객체는 Iterator 프로토콜에 따라 StopIteration 예외를 발생시킨다.

```
>>> g = gen_123()
>>> next(g)
1
>>> next(g)
2
>>> next(g)
3
>>> next(g)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```


❖ generator 함수

```
>>> def gen_123():
...     yield 1
...     yield 2
...     yield 3
...
>>> g = gen_123()
>>> i = [1, 2, 3, 4]
>>> for tmp_i in i:
...     try:
...         next(g)
...     except StopIteration as e:
...         print("이미 제너레이터 객체 요소의 끝에 도달했습니다.")
...
1
2
3
이미 제너레이터 객체 요소의 끝에 도달했습니다.
>>>
```


❖ 코루틴(Coroutine)

- 제너레이터의 호출자는 send()를 이용해서 제너레이터 함수 내부의 yield 표현식의 값이 될 데이터를 전송할 수 있다. 이렇게 제너레이터가 호출자에 데이터를 생성해주고 호출자로부터 데이터를 받으면서 호출자와 협업하는 프로시저인 코루틴이 된다.

```
>>> def simple_coroutine():
...     print("-> coroutine started")
...     x = yield
...     print("-> coroutine received:", x)
...
>>> my_coro = simple_coroutine()
>>> my_coro
<generator object simple_coroutine at 0x7fc3cd761308>
>>> next(my_coro)  코루틴을 대기 상태로 만들
-> coroutine started
>>> my_coro.send(42) send() 메서드를 통해 코루틴에 42를 보냄
-> coroutine received: 42
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
StopIteration
```

코루틴도 제너레이터의 끝에 도달하면
StopIteration을 반환함

❖ 데커레이터(Decorator)

- 데커레이터는 다른 함수를 인수로 받는 콜러블(데커레이트된 함수)이다.
 - 콜러블: 호출할 수 있는 객체 (사용자 정의 함수, 내장 함수, 내장 메서드, 메서드, 클래스, 클래스 객체, 제너레이터 함수)
- 데커레이터는 데커레이트된 함수에 어떤 처리를 수행하고, 함수를 반환하거나 함수를 다른 함수나 콜러블 객체로 대체한다.

```
>>> def deco(func): deco()가 inner() 함수 객체를 반환
...     def inner():
...         print('running inner()')
...     return inner
...
>>> @deco
... def target(): target()을 deco로 데커레이트함
...     print('running target()')
...
>>> target() 데커레이트된 target()을 호출하면 실제로는 inner()를 실행
running inner()
>>> target
<function deco.<locals>.inner at 0x7fb7b148b730>
```

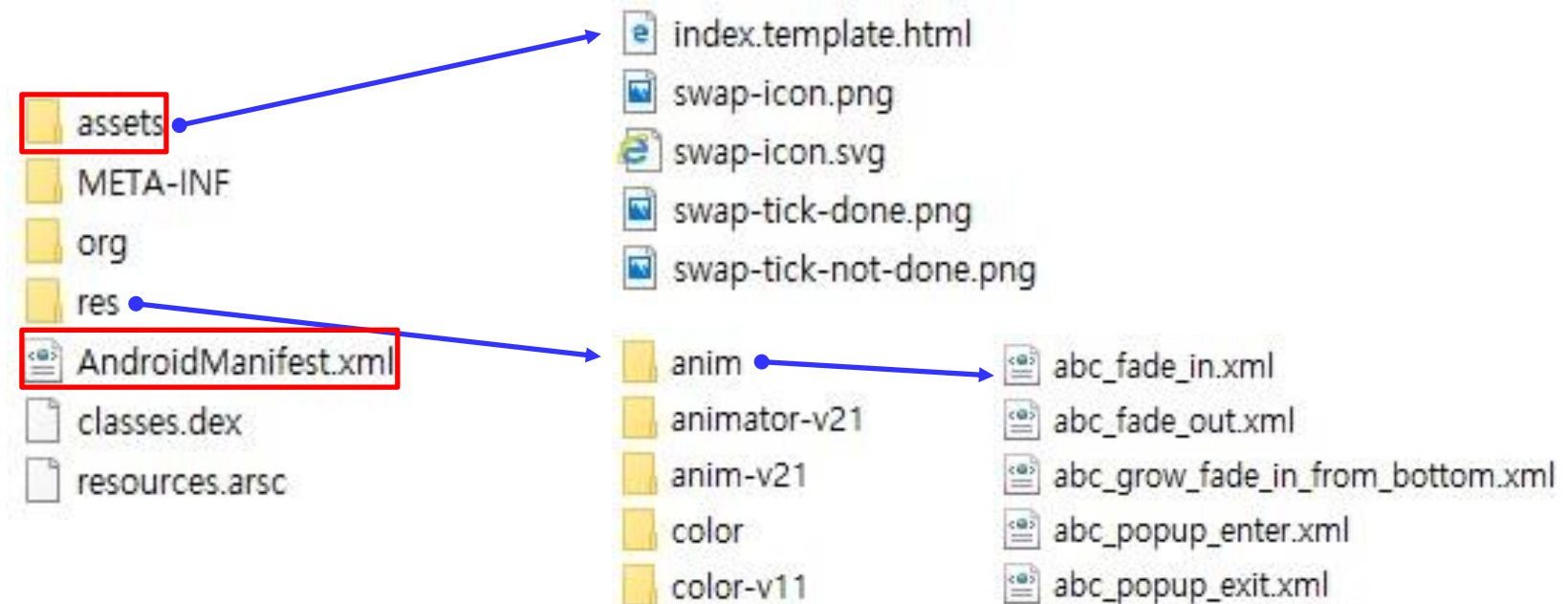
조사해보면 target이 inner()를 가리키고 있음을 알 수 있다.

The android platform

출처(논문): Stealth attacks An extended insight into the obfuscation effects on Android malware

❖ 안드로이드 앱

- 기본적으로 압축된 archive이며 .apk 파일 확장자를 가진다.
- AndroidManifest.xml : entry-points, permissions, actions
- classes.dex : Dalvik byte code
- assets : external resources (audio files, images)
- res : .xml files (visual structure of the user interface)

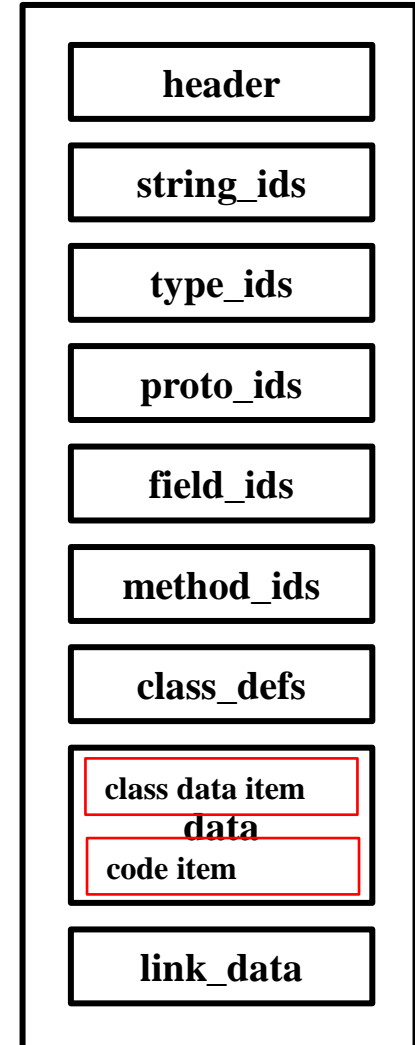


The android platform

출처(논문): Stealth attacks An extended insight into the obfuscation effects on Android malware

❖ Dex file structure

- Header
 - 매직 넘버, 파일 체크섬 넘버, .dex file에 대한 20bytes의 SHA-1 hash
 - 헤더와 파일의 크기, 엔디안 태그, Data 영역에 대한 특정한 주소
 - 다른 IDs 영역의 크기와 위치
- String IDs
 - 관련된 문자열이 저장되는 data section를 가리키는 주소
- Type IDs
 - 일치하는 string type에 대한 참조를 포함하는 String IDs 관련 주소
- Proto IDs
 - 메소드 리턴타입과 파라미터를 어떻게 찾을 수 있는지에 대한 주소
- Field IDs
 - 클래스 필드에 대한 정보를 찾아오기 위한 참조
- Method IDs
 - 메소드에 대한 정보를 찾아오기 위한 참조
- Classes Defs
 - 모든 클래스 파라미터를 정의
- Data (class data item과 code item으로 구성됨)
 - class data item : size와 offset에 관련된 모든 정보
 - code item : 각각의 클래스와 각각의 메소드에 대한 바이트코드



Androguard

❖ Androguard

- 안드로이드 파일과 함께 동작하는 완벽한 파이썬 도구(Full Python Tool)
- Androguard 3.2 버전을 기준으로 Apk를 입력으로 받아 Dex 파일에 있는 code_item을 파싱하는 코드(code_item.py)를 작성하여 테스트
- 명령어

```
root@pmj:~/seuresw_180819# python codeitem.py
```

- 구현 코드

```
from androguard.core.bytecodes.apk import APK
from androguard.core.bytecodes.dvm import DalvikVMFormat
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

a = APK("example/frozenbubble.apk")
d = DalvikVMFormat(a)

code_item = d.get_codes_item()

code = code_item.show()
```

대상 Apk 파일의 경로를 파라미터로 넘김

Androguard

❖ Androguard

■ document

```
class androguard.core.bytecodes.dvm.DalvikVMFormat (buff, decompiler=None, config=None, using_api=None)
```

Bases: androguard.core.bytecode._Bytecode

This class can parse a classes.dex file of an Android application (APK).

Parameters

- **buff** (*string*) – a string which represents the classes.dex file
- **decompiler** (*object*) – associate a decompiler object to display the java source code

Example DalvikVMFormat(read(“classes.dex”))

```
get_codes_item()
```

This function returns the code item

Return type *CodeItem* object

Androguard

❖ Androguard

■ 실행 결과

```
root@pmj:~/securerw_180819# python codeitem.py
No handlers could be found for logger "androguard.xml"
CODE_ITEM
020002000100000065bd0000060000005b010f007010040200000e0004000300010000008bd700000b0000006e10d50003000a00390005006e10f600020012000f
000200020001000000e7bc0000060000005b0110007010040200000e00040003000100000095c800000b0000006e10d50003000a00390005006e10f60002001200
0f00020002000100000089bd0000060000005b0111007010040200000e0004000300010000009dd900000b0000006e10d50003000a00390005006e10f600020012
000f000200020001000000b6bd0000060000005b0112007010040200000e000400030001000000f3dc00000b0000006e10d50003000a00390005006e10f6000200
12000f00020002000100000080bd0000060000005b0113007010040200000e00090002000700000066d900001c00000013006a007110850100005470130071108c
0100000c001a011d046e20c1031000547013001211122212031204120512067707830100000e0002000200010000009bbd0000060000005b011400701004020000
0e000900020007000000aadb00002800000071008b0100000a0813006d007110850100005470140071108c0100000c001a011d046e20c10310003d080e00547014
001211122212131204120512067707830100000e0054701400711208a0180000e000200020001000000c8bd0000060000005b0115007010040200000e0009000200
0700000087e000003800000013006f007110920100005470150071108c0100000c001a011d046e20c10310007100800100000a0013016b0033100e005470150012
21122212131234120512067707830100000e007100800100000a0013016c0033100d00547015001221122212131214120512067707830100000e00020002000100
0000dabd0000060000005b0116007010040200000e00090002000700000092e200003800000013006e007110920100005470160071108c0100000c001a011d046e
20c10310007100800100000a0013016b0033100e00547016001211122212131234120512067707830100000e007100800100000a0013016c0033100d0054701600
1211122212131214120512067707830100000e000200020001000000e3bd0000060000005b0117007010040200000e000400030001000000bde200000b0000006e
10d50003000a00390005006e10f600020012000f000200020001000000febd0000060000005b0118007010040200000e000300030001000000e1e3000004000000
72103e0001000e0002000200010000004abd0000060000005b0119007010040200000e000400020002000000e8d50000170000001300690071108d010000542019
0071108c0100000c001a011d046e20c103100054201900124171208e0110000e000200020001000000d1bd0000060000005b011a007010040200000e0004000300
01000000b4e000000b0000006e10d50003000a00390005006e10f600020012000f00020002000100000010be0000060000005b011b007010040200000e00030003
0000000000b5e70000010000000e000200020001000000f9bc0000060000005b011c007010040200000e00050003000100000032ce00001a00000012016e10d500
```

Regular Expression

❖ Regular Expression (정규 표현식)

- 텍스트나 문자열의 검색과 치환을 위해 사용
- Language에 국한하지 않고 사용 가능
- 디컴파일한 코드에서 Constant String을 파싱하여 개수를 카운트하는 코드

```
String_array = re.findall(r"\".*?\\"", source)
for s in String_array:
#     print(s)
    if s:
        stringnum += 1
```

- 위 코드에서 사용된 정규 표현식

표현식	설명
\	메타 문자를 문자 그대로 사용하겠다는 의미
.	임의의 한 문자
*	앞 문자가 없을 수도 무한정 많을 수도 있음
?	앞 문자가 없거나 하나 있음 (탐욕적 탐색을 막음)

Regular Expression

❖ Regular Expression (정규 표현식)

- 앞에 본 정규표현식에서 ?가 없으면 한 라인에 두 개 이상의 문자열이 있을 경우, 파싱을 잘못 수행함
- ex) DecryptionMethod("Constant", "String") 라는 라인이 있을 때, "Constanant", "String"이라고 파싱하게 됨
- ?가 있다면,
"Constant"와 "String"을 각각 파싱함

Regular Expression

❖ Regular Expression (정규 표현식)

- Java method descriptor에서 파라미터의 타입이 문자열인 경우를 파싱하는 코드

```
regex = re.compile('\s*Ljava/lang/String;\s*')

# filter methods which contains parameter Ljava/lang/String
for method in methods:
    print(method.get_method().get_descriptor())
    res = regex.search(method.get_method().get_descriptor())
```

- 위 코드 에서 사용된 정규표현식

표현식	설명
\	메타 문자를 문자 그대로 사용하겠다는 의미
[]	문자의 집합이나 범위를 나타내며 두 문자 사이는 - 기호로 범위를 나타냄
*	모든 문자
\s	공백 문자
\S	공백 문자가 아닌 나머지 문자

Regular Expression

❖ Regular Expression (정규 표현식)

```
(Landroid/net/Uri; Ljava/lang/String;)Landroid/content/Intent;  
(F F F F)V  
( )V  
(F F F F)V  
(F F)V  
(F F)V  
( )V  
(I I Ljava/lang/CharSequence;)Landroid/text/SpannableStringBuilder;  
(I)C  
(I I Ljava/lang/CharSequence; I I)Landroid/text/SpannableStringBuilder;  
(Ljava/lang/CharSequence;)V  
(Landroid/graphics/Bitmap; I I)V  
( )I  
(S B I I I)Ljava/lang/String;  
( )V  
( )V  
(Landroid/graphics/Bitmap; Ljava/lang/String; Ljava/lang/String;)Ljava/lang/String;
```

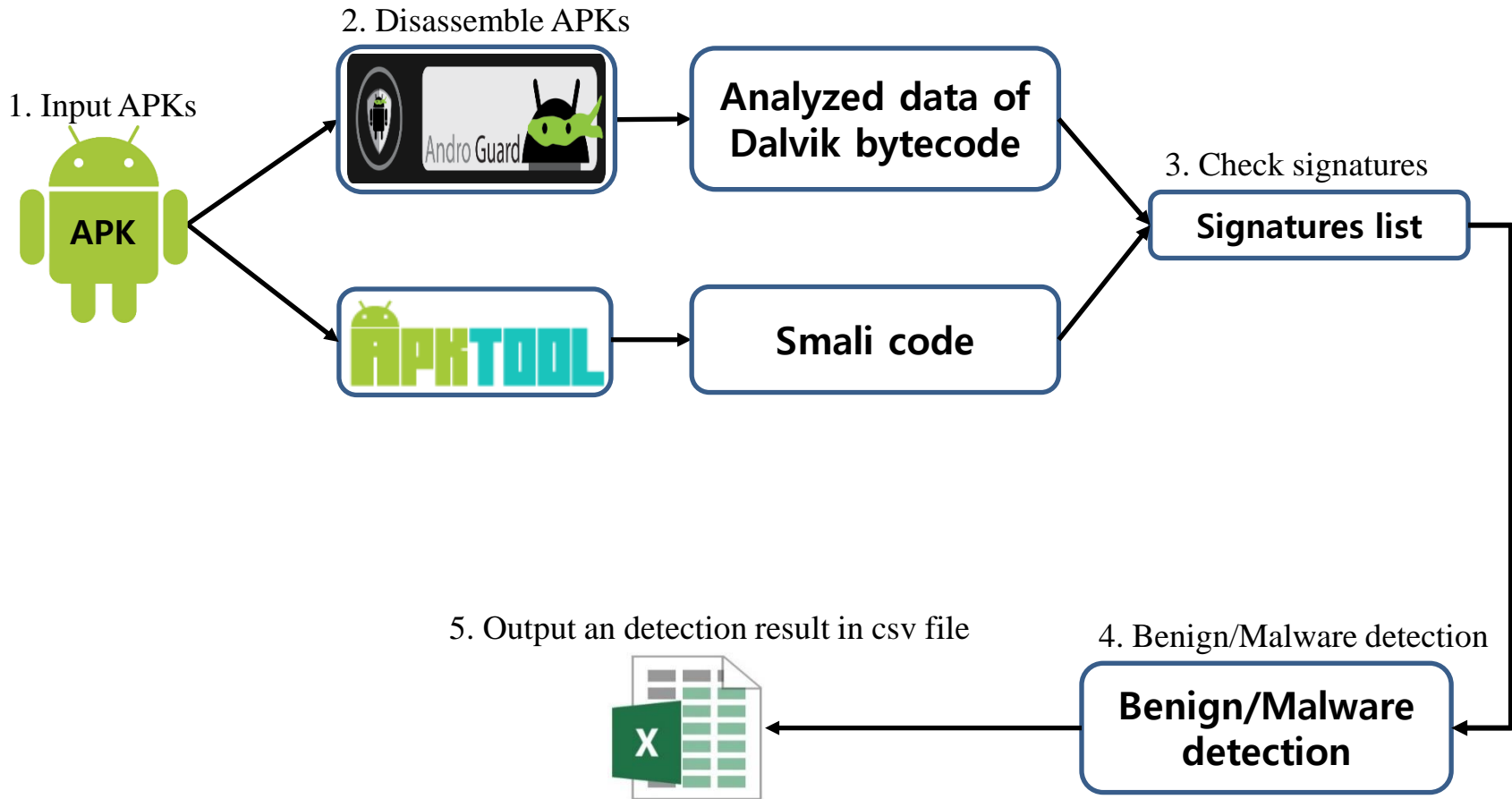
공백 문자가 앞에만 있고, 뒤에는 없음

공백 문자가 앞, 뒤로 있음

- ❖ 대상 Java method descriptor의 파라미터 타입이 문자열일 때, 몇 번째 파라미터로 문자열이 넘어가는지 분석하는 입장에서는 알 수 없기 때문에, 모든 가능한 경우를 다 파싱하기 위하여 작성된 정규표현식임을 알 수 있다.

Thank You !

❖ 안드로이드 앱 분석 프로그램의 개요도



❖ Virus Total

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



바이러스토탈은 **의심스런 파일과 URL을 분석**하고 바이러스, 웜, 트로얀과 모든 종류의 악성 코드를 쉽고, 빠르게 탐지할 수 있는 편리한 무료 서비스입니다.

📁 파일

🌐 URL

🔍 검색

선택 파일 없음

파일 선택

최대 파일 크기: 128MB

'검사 시작!' 버튼을 클릭함으로써, 저희의 **서비스 약관**에 동의하는 것이며,
바이러스토탈이 이 파일을 보안 커뮤니티와 공유하는 것을 허용함을 뜻합니다.
자세한 내용은 **개인정보 보호정책**을 참조하십시오.

검사 시작!

부록

❖ 안드로이드 에뮬레이터

- Android Emulator는 기기를 시뮬레이션하여 이를 개발용 컴퓨터에 표시
- 에뮬레이터를 사용하면 하드웨어 기기를 사용하지 않고서도 Android 앱의 프로토타입을 만들고 개발, 테스트
- 에뮬레이터는 Android 전화기, 태블릿, Android Wear 및 Android TV 기기 등을 지원
- 연결된 하드웨어 기기를 사용할 때보다 빠르게 정보를 전송할 수 있어 개발 절차를 한층 빠르게 함

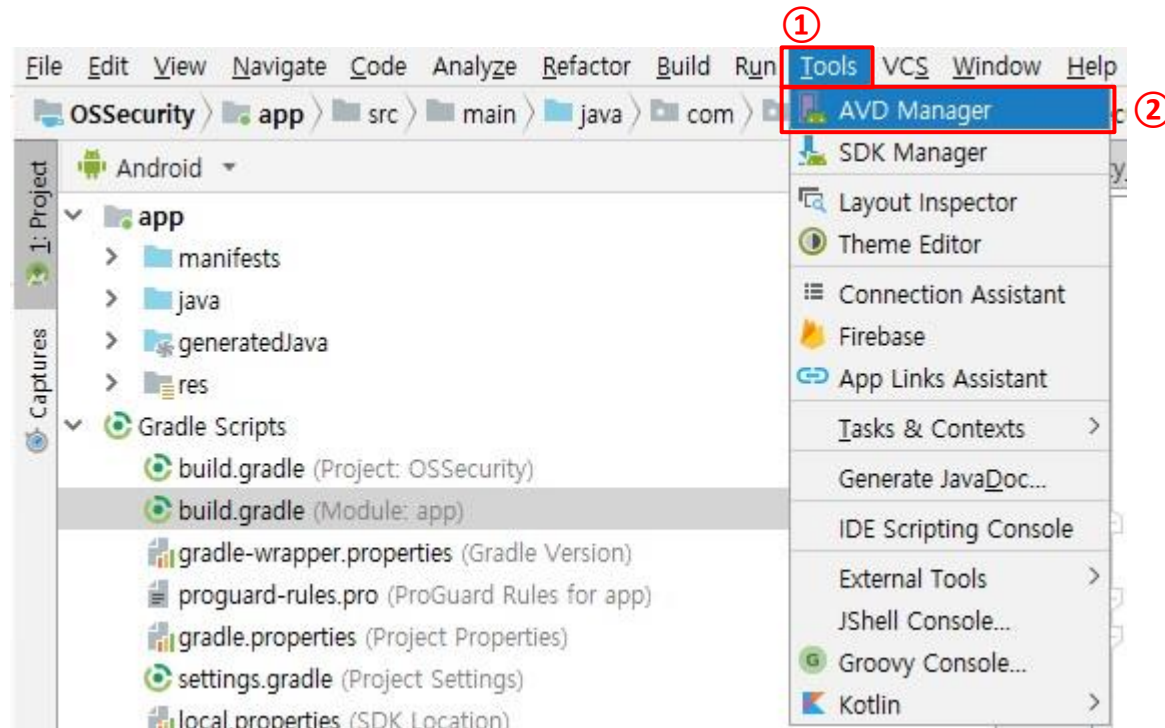


출처(사이트): <https://developer.android.com/studio/run/emulator?hl=ko>

부록

❖ 안드로이드 에뮬레이터 생성

- 안드로이드 스튜디오 > Tools > AVD Manager







부록

❖ 안드로이드 에뮬레이터 생성

- Android Virtual Device Manager > Create Virtual Device...

Android Virtual Device Manager

Your Virtual Devices
Android Studio

Type	Name	Play Store	Resolution	API	Target	CPU/ABI	Size on Disk	Actions
	Android Accelerated x86 Nougat		Unknown Resolution	25	Android 7.1.1 (Google APIs)	x86	1.0 GB	 Download ▼
	Nexus SX API 19		1080 × 1920: 420dpi	19	Android 4.4 (Google APIs)	arm	5.7 GB	▶ ✎ ▼
	Nexus SX API 25	▶	1080 × 1920: 420dpi	25	Android 7.1.1 (Google Play)	x86	5.7 GB	▶ ✎ ▼

③

+ Create Virtual Device...

부록

❖ 안드로이드 에뮬레이터 생성

- Virtual Device Configuration > Next

Virtual Device Configuration

Select Hardware
Android Studio

Choose a device definition

Category	Name	Play Store	Size	Resolution	Density
TV	Nexus 6P		5.7"	1440x2560	560dpi
Phone	Nexus 6		5.96"	1440x2560	560dpi
	Nexus 5X	▶	5.2"	1080x1920	420dpi
Wear OS	Nexus 5	▶	4.95"	1080x1920	xxhdpi
Tablet	Nexus 4		4.7"	768x1280	xhdpi
	Galaxy Nexus		4.65"	720x1280	xhdpi
	5.4" FWVGA		5.4"	480x854	mdpi
	5.1" WVGA		5.1"	480x800	mdpi
	4.7" WXGA		4.7"	720x1280	xhdpi

New Hardware Profile Import Hardware Profiles

Nexus 5X

1080px
5.2"
1920px

Size: large
Ratio: long
Density: 420dpi

Clone Device...

4

Previous Next Cancel Finish

부록

❖ 안드로이드 에뮬레이터 생성

- Virtual Device Configuration > Other Images > Next
- Target에서 **안드로이드 앱의 API Level에 적합한 항목을 선택**할 것

Virtual Device Configuration



Select a system image

Recommended x86 Images **Other Images**

Release Name	API Level	ABI	Target
Nougat Download	24	armeabi-v7a	Android 7.0 (Google APIs)
Nougat Download	24	arm64-v8a	Android 7.0
Nougat Download	24	armeabi-v7a	Android 7.0
Marshmallow Download	23	armeabi-v7a	Android 6.0 (Google APIs)
Marshmallow Download	23	armeabi-v7a	Android 6.0
Lollipop	22	armeabi-v7a	Android 5.1 (Google APIs)
Lollipop	22	armeabi-v7a	Android 5.1
Lollipop	21	armeabi-v7a	Android 5.0 (Google APIs)
Lollipop	21	armeabi-v7a	Android 5.0
KitKat	19	armeabi-v7a	Android 4.4 (Google APIs)
KitKat	19	armeabi-v7a	Android 4.4
Jelly Bean Download	15	armeabi	Android 4.3 (Google APIs)

KitKat



API Level

19

Android

4.4

Google Inc.

System Image

armeabi-v7a

Recommendation

Consider using an x86 system image on an x86 host for better emulation performance.

Questions on API level?

See the [API level distribution chart](#)

⑤

Previous

Next

Cancel

Finish

부록

❖ 안드로이드 에뮬레이터 생성

- 최신 안드로이드 스튜디오 버전(3.2)에서 armeabi 계열 에뮬레이터가 제대로 작동하지 않는 issue가 있으므로 **x86 Images**를 사용할 것을 권장 !

Virtual Device Configuration



Select a system image

Recommended **x86 Images** Other Images

Release Name	API Level	ABI	Target
Nougat Download	24	armeabi-v7a	Android 7.0 (Google APIs)
Nougat Download	24	arm64-v8a	Android 7.0
Nougat Download	24	armeabi-v7a	Android 7.0
Marshmallow Download	23	armeabi-v7a	Android 6.0 (Google APIs)
Marshmallow Download	23	armeabi-v7a	Android 6.0
Lollipop	22	armeabi-v7a	Android 5.1 (Google APIs)
Lollipop	22	armeabi-v7a	Android 5.1
Lollipop	21	armeabi-v7a	Android 5.0 (Google APIs)
Lollipop	21	armeabi-v7a	Android 5.0
KitKat	19	armeabi-v7a	Android 4.4 (Google APIs)
KitKat	19	armeabi-v7a	Android 4.4
Jelly Bean Download	15	armeabi	Android 4.3 (Google APIs)

KitKat



API Level

19

Android

4.4

Google Inc.

System Image

armeabi-v7a

Recommendation

Consider using an x86 system image on an x86 host for better emulation performance.

Questions on API level?

See the [API level distribution chart](#)

Previous

⑤

Next

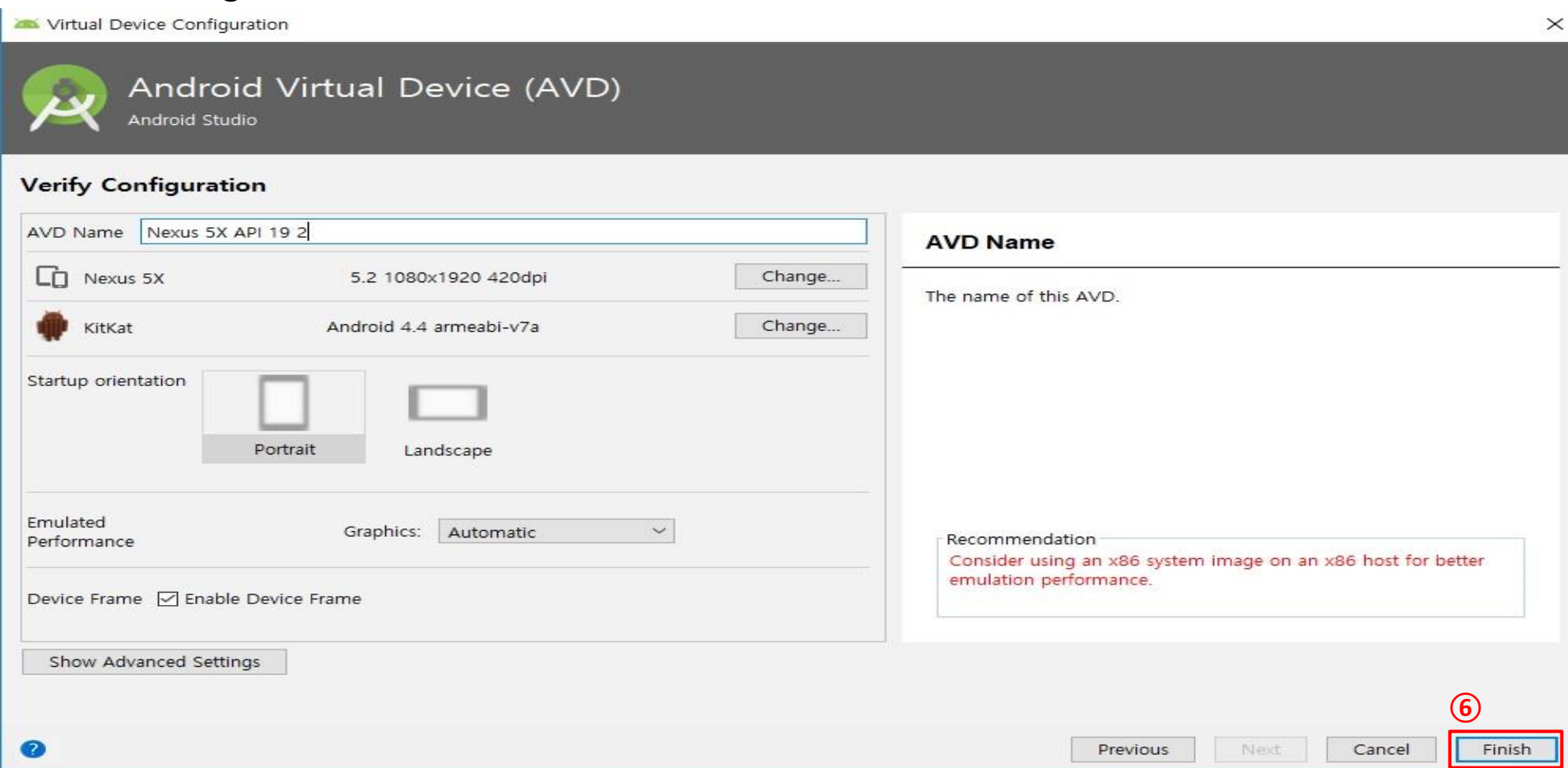
Cancel

Finish

부록

❖ 안드로이드 에뮬레이터 생성

- Virtual Device Configuration > Other Images > Next
- Target에서 **안드로이드 앱의 API Level에 적합한 항목을 선택**할 것



❖ ADB (Android Debug Bridge)

- 안드로이드 스튜디오 설치 시에 SDK Manager에 내장되어 배포됨
- adb 명령어
 - adb devices : 현재 데스크탑(PC)에 연결된 디바이스를 조회

cmd 명령 프롬프트

```
C:\Users\souli>adb devices
List of devices attached
emulator-5554    device
```

- adb shell : 연결된 디바이스의 shell로 접속

cmd 명령 프롬프트 - adb shell

```
C:\Users\souli>adb shell
root@generic:/ #
```

❖ ADB (Android Debug Bridge)

- adb shell 명령어
 - **id** : 사용자 id와 그룹 id를 조회
 - 에뮬레이터는 기본적으로 루팅된 상태이기 때문에 uid와 gid가 0으로 나옴

```
C:\ 명령 프롬프트 - adb shell

C:\Users\souli>adb shell
root@generic:/ # id
uid=0(root) gid=0(root) context=u:r:shell:s0
root@generic:/ #
```

- **cat /proc/meminfo** : 메모리 정보를 조회

```
C:\ 명령 프롬프트 - adb shell

root@generic:/ # cat /proc/meminfo
MemTotal:      1554240 kB
MemFree:       618184 kB
Buffers:       39408 kB
Cached:        516084 kB
SwapCached:    0 kB
```

❖ ADB (Android Debug Bridge)

- adb shell 명령어
 - **getprop** : property를 조회

cmd 명령 프롬프트 - adb shell

```
root@generic:/ # getprop
[ARGH]: [ARGH]
[dalvik.vm.heapsize]: [384m]
[dalvik.vm.stack-trace-file]: [/data/anr/traces.txt]
[debug.force_rtl]: [0]
[dev.bootcomplete]: [1]
[gsm.current.phone-type]: [1]
[gsm.defaultpdcontext.active]: [false]
```














- **ps** : 데몬 or 프로세스를 조회

cmd 명령 프롬프트 - adb shell

```
1|root@generic:/ # ps | grep init
root      1      0      644      500      c00ed2bc 0001a098 S /init
root@generic:/ # ps | grep ueventd
root      458    1      576      260      c00ed2bc 0001a098 S /sbin/ueventd
root@generic:/ # ps | grep servicemanager
system    648    1      1000     176      c029ad18 b6fa041c S /system/bin/servicemanager
root@generic:/ # ps | grep zygote
root      655    1      675256  40276      ffffffff b6ec5568 S zygote
```


❖ JDK (Java Development Kit)

- <https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>
- 개발 환경에 맞는 버전을 설치

Java SE Development Kit 8u192		
You must accept the Oracle Binary Code License Agreement for Java SE to download this software.		
Thank you for accepting the Oracle Binary Code License Agreement for Java SE; you may now download this software.		
Product / File Description	File Size	Download
Linux ARM 32 Hard Float ABI	73 MB	 jdk-8u192-linux-arm32-vfp-hflt.tar.gz
Linux ARM 64 Hard Float ABI	69.92 MB	 jdk-8u192-linux-arm64-vfp-hflt.tar.gz
Linux x86	170.9 MB	 jdk-8u192-linux-i586.rpm
Linux x86	185.7 MB	 jdk-8u192-linux-i586.tar.gz
Linux x64	167.99 MB	 jdk-8u192-linux-x64.rpm
Linux x64	182.87 MB	 jdk-8u192-linux-x64.tar.gz
Mac OS X x64	245.92 MB	 jdk-8u192-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	133.2 MB	 jdk-8u192-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	94.33 MB	 jdk-8u192-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	134.01 MB	 jdk-8u192-solaris-x64.tar.Z
Solaris x64	92.17 MB	 jdk-8u192-solaris-x64.tar.gz
Windows x86	197.57 MB	 jdk-8u192-windows-i586.exe
Windows x64	207.42 MB	 jdk-8u192-windows-x64.exe

❖ 안드로이드 스튜디오(Android Studio)

- <https://developer.android.com/studio/?hl=ko#downloads>
- 개발 환경에 맞는 버전을 설치

Android Studio downloads

Platform	Android Studio package	Size	SHA-256 checksum
Windows (64-bit)	android-studio-ide-181.5056338-windows.exe Recommended	927 MB	6ee509f3391757fe87cc5c1e4970a0228fc1ad6ca34a8b31c0a28926179353a9
	android-studio-ide-181.5056338-windows.zip No .exe installer	1001 MB	21aebb3a7fab4931b830ec40d836d6945eabb4f32acf1b52fae148d33599fd7c
Windows (32-bit)	android-studio-ide-181.5056338-windows32.zip No .exe installer	1000 MB	3a61a587c90e358ab15d076d0306550564ad4cc5a8aa1dd0c22c6afd092e976a
Mac	android-studio-ide-181.5056338-mac.dmg	989 MB	b8d2b7add6a7c776d16a8e48bd35c3e2bba18b4717131d7b9a00fa416ebe4480
Linux	android-studio-ide-181.5056338-linux.zip	1007 MB	b9ec0d44f2feaafe1e3fbd1ed696bf325f9e05cfb6c1ace84dbf87ae249efa84

❖ 참고 사이트

- <https://scholar.google.co.kr>
 - PPT에 적혀있는 논문 이름을 검색하면 해당 논문을 찾을 수 있음
- <https://developer.android.com/reference/packages>
 - API level 19~26까지 사용 가능
- <https://developer.android.com/studio/command-line/adb>
 - ADB (Android Debug Bridge) 명령어
- <https://docs.oracle.com/javase/8/docs/api/overview-summary.html>
 - java 버전 8을 기준으로 참고
- <https://stackoverflow.com/>
 - 개발 시에 참고