Android app 함수 후킹

천재권

목차

버그헌팅 중급

버그헌팅 중급강의

normaltic

24년 실전형 사이버훈련장 - 버그헌팅 실습 훈련 (과정 개선)

중급 +228명 4.8

KISA 버그헌팅 버그바운티

버그헌팅 초급강의

김동규

24년 실전형 사이버훈련장 - 버그헌팅 실습 훈련 (과정 개선)

초급 +302명 4.74

KISA 버그헌팅 버그바운티 초급강의

버그헌팅 중급

안드로이드 앱 보안

04 어플리케이션 보안 이해

- 모바일 앱 해킹 개요
- 모바일 앱 취약점 분석 환경 세팅
- Android App 이해
- Android App 분석 실습 **2024 New**
- iOS App 이해
- Android App 취약점 (Insecure Deeplink, ...)
- iOS App 취약점 (Insecure Storage, ...)
- App Logic 조작 이해 (함수 후킹)
- Frida 이해
- Android App 함수 후킹
- iOS App 함수 후킹

05 보안 기술 우회 학습

- 보안 솔루션 이해 (Android)
- 보안 솔루션 Bypass (Android)
- 보안 솔루션 이해 (iOS)
- 보안 솔루션 Bypass (iOS)

06 취약점 분석 (Private Bounty 참여 및 실습)

08 Metasploit

- Metasploit 알아보기
- Metasploit 기본 사용법
- Metasploit Module 제작

09 1- Day Analyze

- 1-Day 취약점 (CVE) 분석 1
- 1-Day 취약점 (CVE) 분석 2
- 1-Day 취약점 (CVE) 분석 3
- 1-Day 취약점 (CVE) 분석 4
- 1-Day 취약점 (CVE) 분석 5
- 1-Day 취약점 (CVE) 분석 6
- 1-Day 취약점 (CVE) 분석 7

10 Chat GPT 버그헌팅

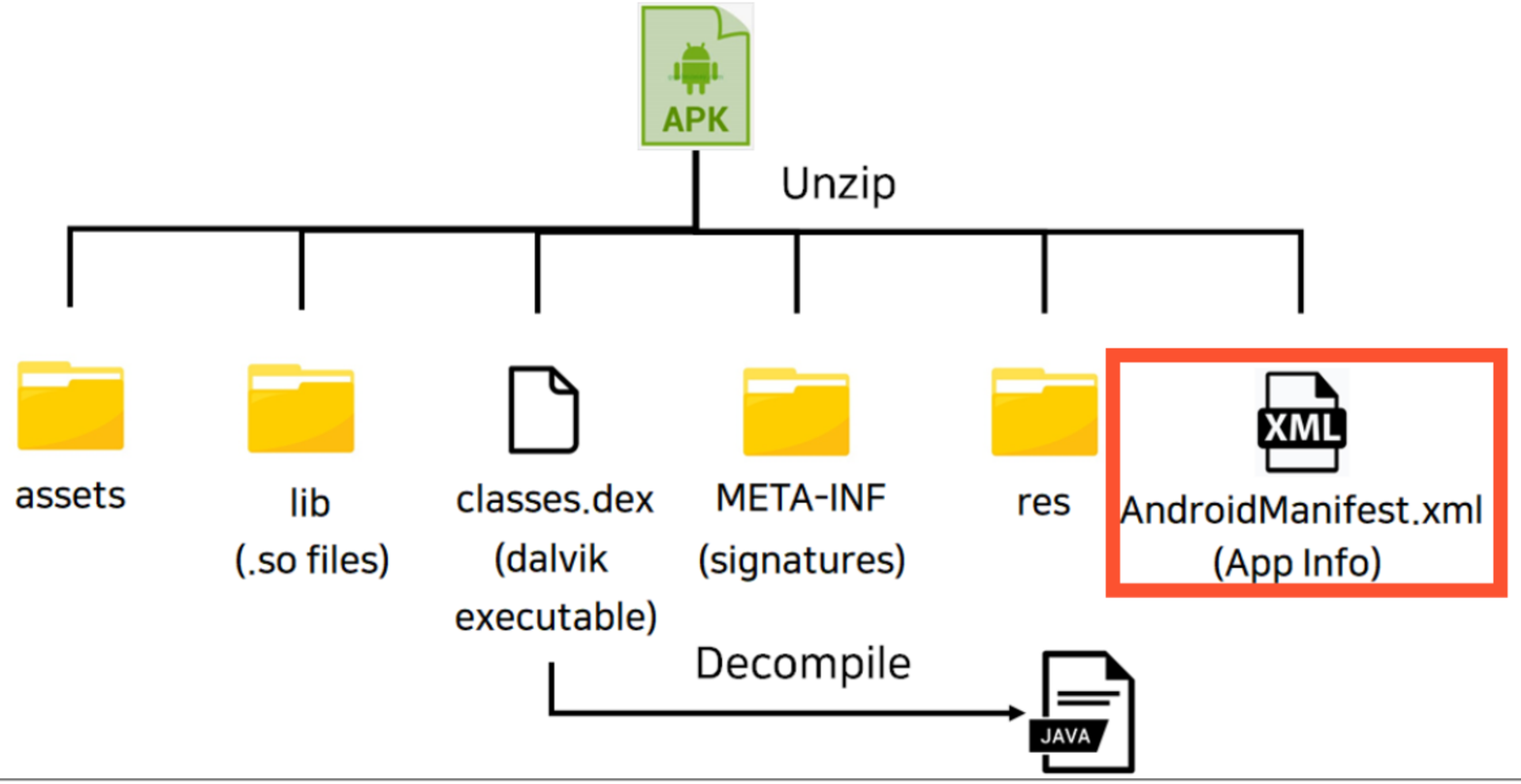
- ChatGPT를 활용한 버그헌팅 개념
- ChatGPT 버그헌팅 프롬프트

Rooting

안드로이드 OS에서 최고 관리자(Root) 권한을 획득하는 과정

- 단말기에서 실행되는 앱 process에 접근해 코드 흐름을 조작
- 앱 정보 열람, 수정
- 앱 분석할때 기본적으로 Root권한을 획득하고 시작

APK 구조


APK
Unzip
assets
lib
(.so files)
classes.dex
(dalvik
executable)
META-INF
(signatures)
res
XML
AndroidManifest.xml
(App Info)
Decompile
JAVA

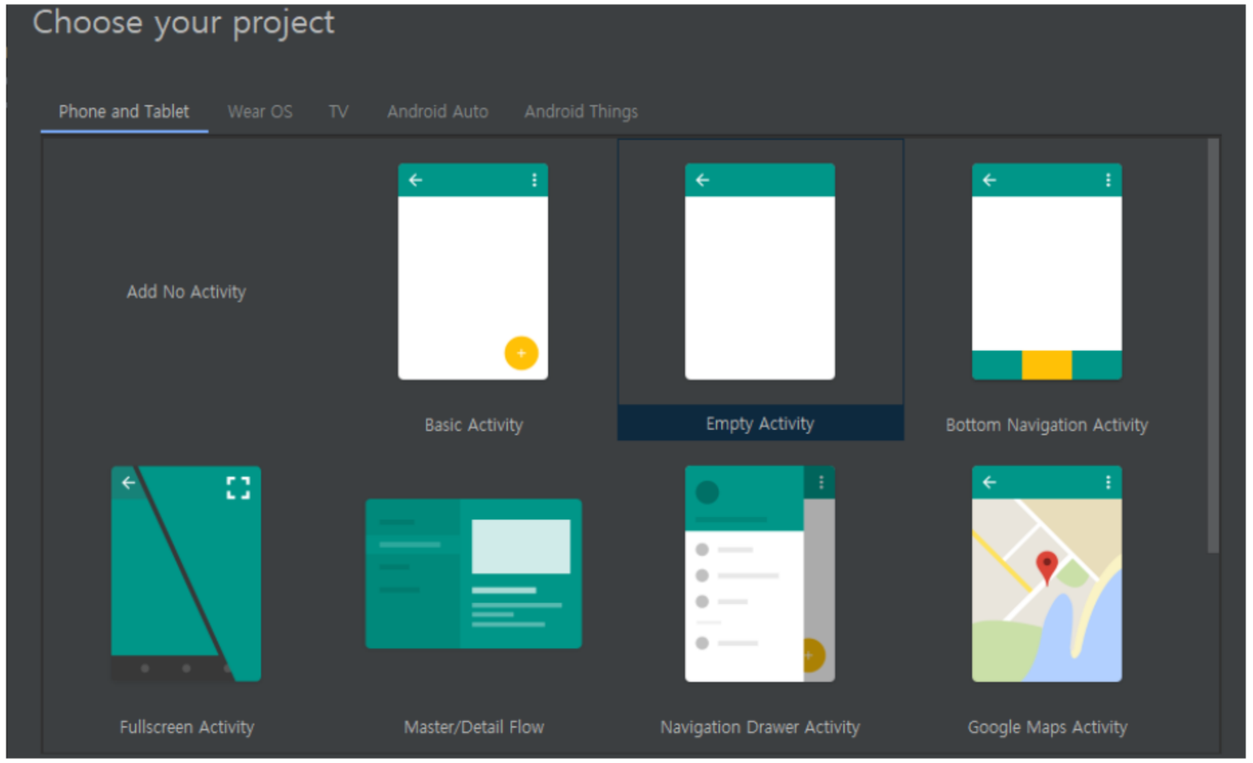
AndroidMainfest.xml

- 고유 패키지 이름
- 앱 버전 정보
- Activity

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:versionCode="1"
    android:versionName="1.0"
    package="owasp.mstg.uncrackable1">
    <uses-sdk
        android:minSdkVersion="19"
        android:targetSdkVersion="28"/>
    <application
        android:theme="@style/AppTheme"
        android:label="@string/app_name"
        android:icon="@mipmap/ic_launcher"
        android:allowBackup="true">
        <activity
            android:label="@string/app_name"
            android:name="sg.vantagepoint.uncrackable1.MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```

Activity

- 안드로이드 app의 화면
- 사용자와 상호작용을 하는 UI
- 하나의 클래스



Android

함수 후킹

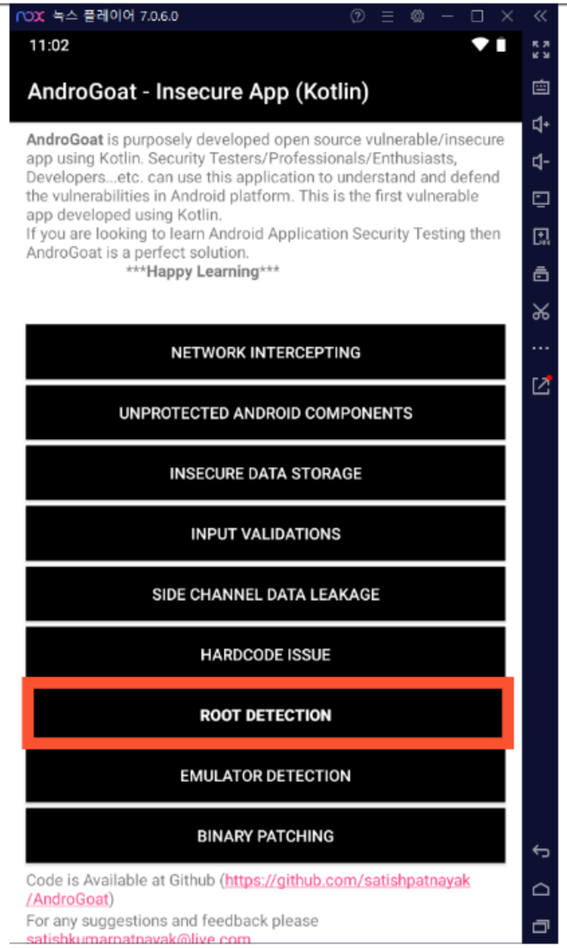
- 함수 실행 직후와 반환 직전에 임의의 코드를 삽입하여 실행 시키는 것
- 함수의 실행 흐름을 임의의 코드로 변경해주는 것

함수 후킹

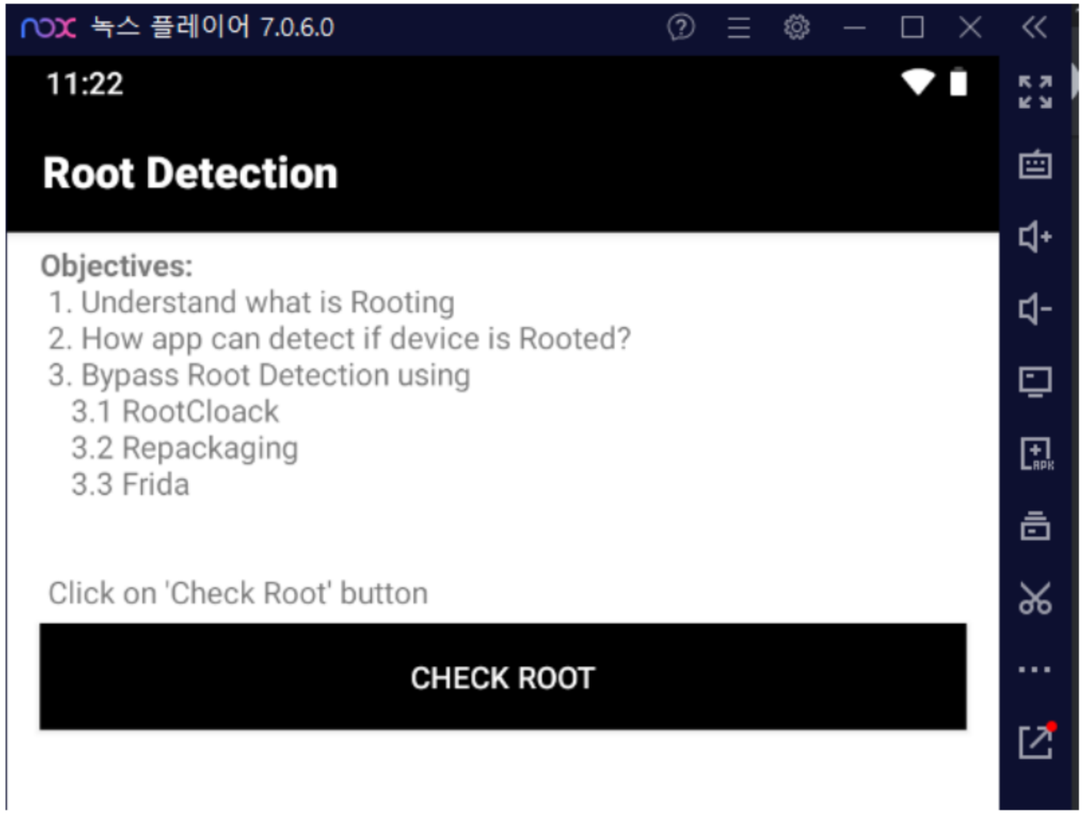
- Python 기반의 모듈
- Javascript를 사용해 함수 후킹 가능

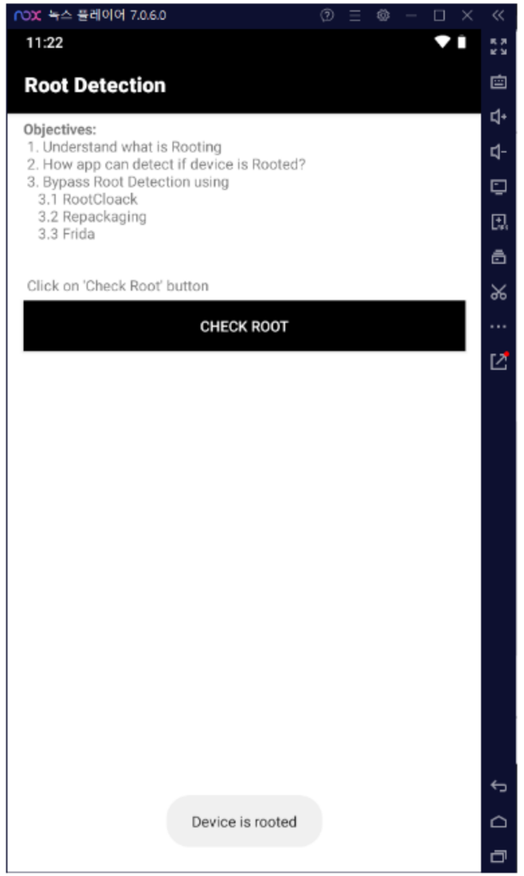
02 AndroGoat 루팅 탐지 기능 우회

02 AndroGoat 루팅 탐지 기능 우회



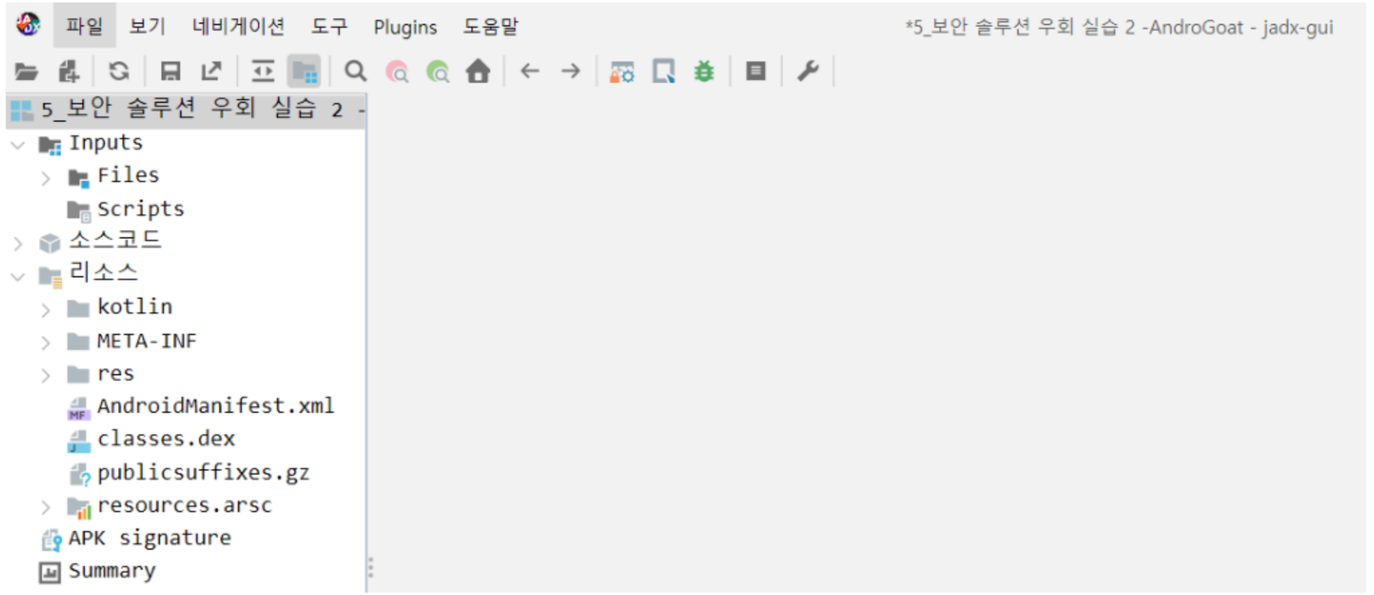
02 AndroGoat 루팅 탐지 기능 우회





jadx로 디컴파일

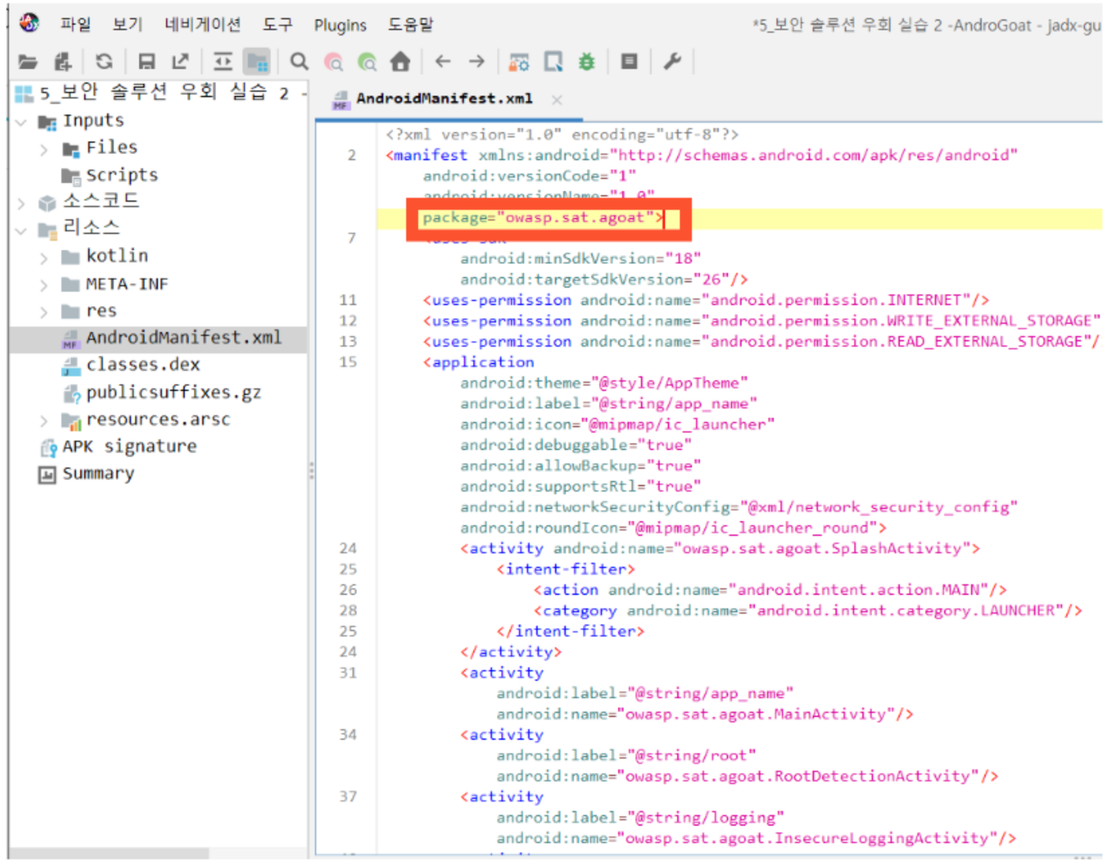
- 바이트코드를 Java코드로 변환 시켜줌



02 AndroGoat 루팅 탐지 기능 우회

AndroidManifest.xml

- 고유 패키지 이름 확인
- Python 코드에 패키지 이름 입력



02 AndroGoat 루팅 탐지 기능 우회

```python
import frida, sys
import argparse

def on_message(message, data):
    if message['type'] == 'send':
        print(message['payload'])
    elif message['type'] == 'error':
        print(message['stack'])

def get_script(script_name):
    with open("./" + script_name, 'r') as f:
        script = f.read()
    return script

help_script="JS File Inject"
parser = argparse.ArgumentParser(description=help_script)
parser.add_argument('--script', required=True, help='JS File to Inject')
args = parser.parse_args()

device = frida.get_usb_device(1)
p1 = device.spawn(["owasp.sat.agoat"])
process_session = device.attach(p1)

script = process_session.create_script(get_script(args.script))
script.on('message', on_message)
script.load()

device.resume(p1)

sys.stdin.read()
```

inject.py JS bypass.js ×

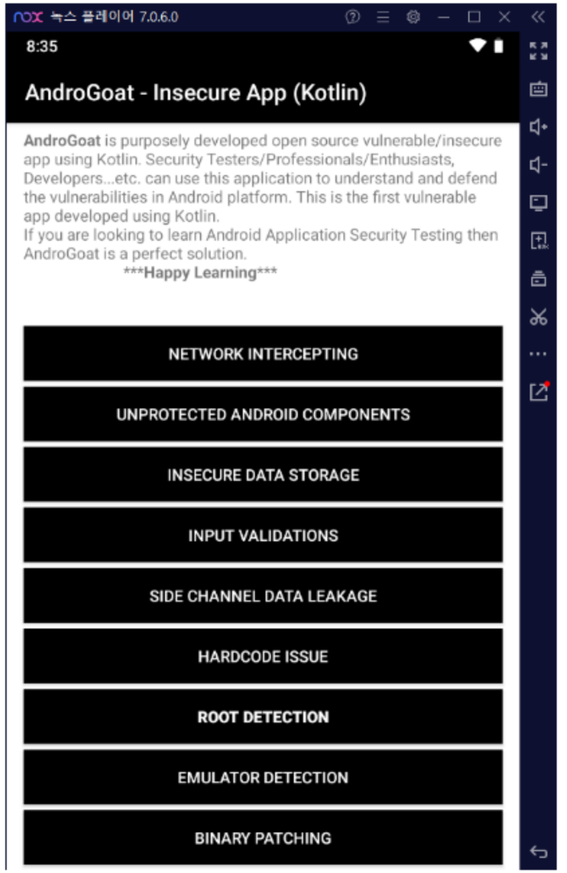
AndroGoat > JS bypass.js

```
1  console.log('[+] JS Load!');
```

owasp.sat.agoat

02 AndroGoat 루팅 탐지 기능 우회

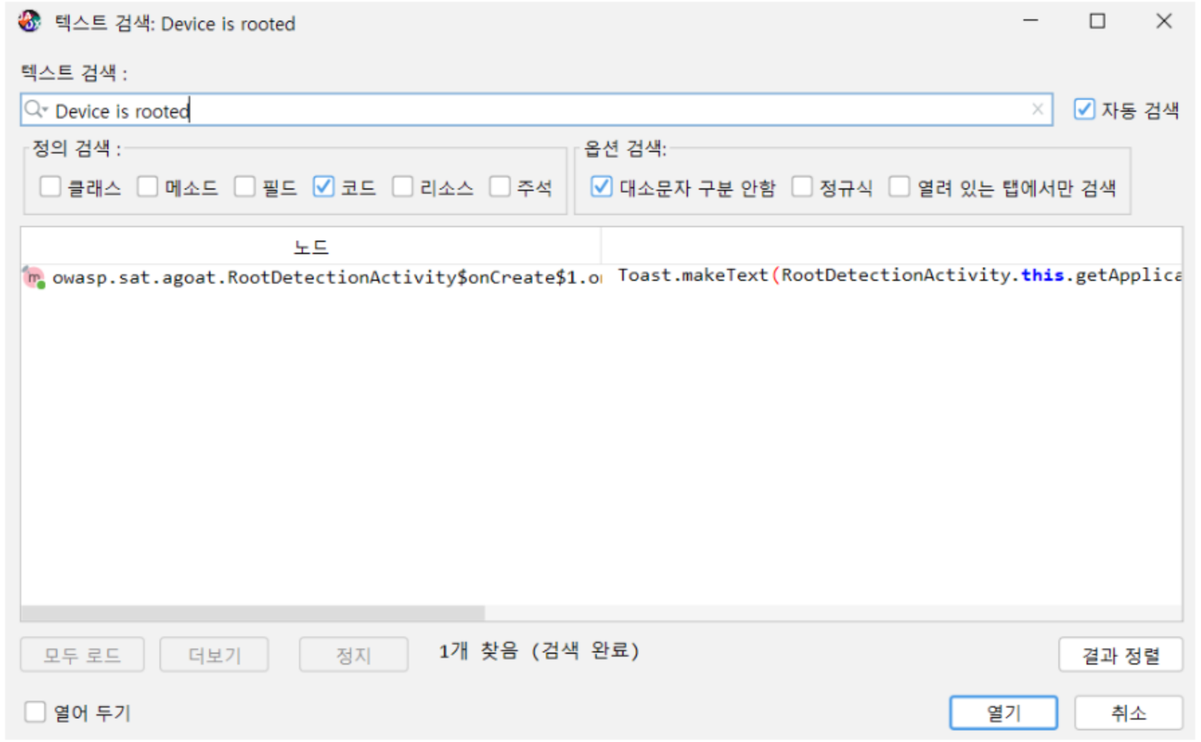
```
C:₩Users₩wornj₩OneDrive₩바탕 화면₩mobile₩AndroGoat>python inject.py --script bypass.js
[+] JS Load!
```



02 AndroGoat 루팅 탐지 기능 우회

루팅 탐지 로직을 찾기 위해 텍스트검색

- Device is rooted



isRooted 메서드 확인

```
public final void onClick(View it) {
    if (RootDetectionActivity.this.isRooted()) {
        Toast.makeText(RootDetectionActivity.this.getApplicationContext(), "Device is rooted", 1).show();
    } else {
        Toast.makeText(RootDetectionActivity.this.getApplicationContext(), "Device is not rooted", 1).show();
    }
}
```

02 AndroGoat 루팅 탐지 기능 우회

이 메서드를 후킹해서 루팅탐지를 우회

```
public final boolean isRooted() {
    String[] file = {"/system/app/Superuser/Superuser.apk", "/system/app/Superuser.apk", "/sbin/su", "/system/bin/su",
    boolean result = false;
    for (String files : file) {
        File f = new File(files);
        result = f.exists();
        if (result) {
            break;
        }
    }
    return result;
}
```

02 AndroGoat 루팅 탐지 기능 우회

Java.perform

- Frida에서 Java 환경의 코드를 조작하기 위해 사용되는 메서드

Java.use

- 앱의 특정 클래스를 불러올 수 있는 메서드

implementation

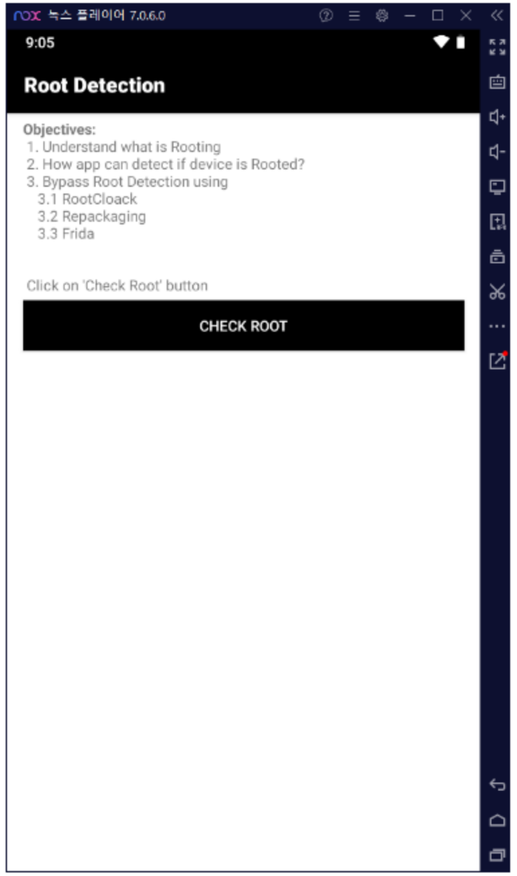
- 함수를 새롭게 재정의

ndroGoat > JS bypass.js > ...

```
1   console.log('[+] JS Load!');
2
3   Java.perform(function () {
4     var root_check_class = Java.use('owasp.sat.agoat.RootDetectionActivity');
5
6     var root_check_method = root_check_class.isRooted;
7
8     root_check_method.implementation = function () {
9       send('[-] Root Check Method Called!');
10      var ret = this.isRooted();
11      send('[-] Root Check Method result before : ' + ret);
12      ret = false;
13      send('[-] Root Check Method result after : ' + ret);
14      return ret;
15    };
16  });
```

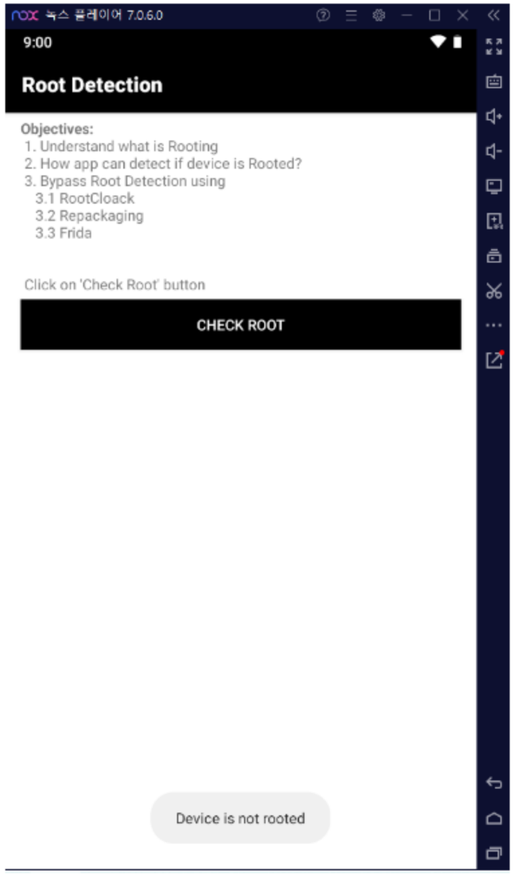
02 AndroGoat 루팅 탐지 기능 우회

```
C:₩Users₩wornj₩OneDrive₩바탕 화면₩mobile₩AndroGoat>python inject.py --script bypass.js
[+] JS Load!
```



02 AndroGoat 루팅 탐지 기능 우회

```
C:₩Users₩wornj₩OneDrive₩바탕 화면₩mobile₩AndroGoat>python inject.py --script bypass.js
[+] JS Load!
[-] Root Check Method Called!
[-] Root Check Method result before : true
[-] Root Check Method result after : false
```



Q & A