발간등록번호
11-1312000-000114-14

모바일 대민서비스 보안취약점 점검 가이드

행정안전부

KISA 한국인터넷진흥원

발간등록번호
11-1312000-000114-14

모바일 대민서비스 보안취약점 점검 가이드

행정안전부
KISA 한국인터넷진흥원

모바일 대민서비스
보안취약점 점검 가이드

제·개정이력 Revision History

순번	제·개정일	변경내용	발간팀	비고
1	2014.10.	**[제정]** • 모바일 대민서비스 보안취약점 점검 가이드	보안평가팀	
2	2015.12.	**[개정]** • 모바일 전자정부 서비스 관리지침의 개정에 따른 검증 기준 등 추가	보안평가인증팀	
3	2021.12.	**[개정]** • 진단 방법 등 최신화	전자정부보호팀	

목 차

Contents

[Part 1]

개 요

Part 1

개 요

'모바일 전자정부 서비스 관리 지침'에 따라 행정기관등[1]에서 개발 · 구축한 모바일 대민서비스에 대한 보안약점 · 취약점 등을 자체적으로 진단 · 제거 시에 참고할 수 있는 진단 절차 및 방법 등을 소개하고자 한다.

일반적으로 모바일 대민서비스는 모바일 웹, 반응형 웹, 모바일 앱 또는 하이브리드 앱 형태로 개발되는데, 모바일 웹과 반응형 웹은 일반 홈페이지와 유사하여 보안취약점 진단 방법[2]도 동일하다. 따라서 본 가이드에서는 모바일 웹과 반응형 웹에 대한 진단 방법은 설명하지 않고, 모바일 앱을 중심으로 보안취약점 진단 방법을 설명한다. 다만, 하이브리드 앱은 모바일 앱과 웹이 결합된 형태이므로 모바일 앱과 관련된 부분은 해당 가이드의 진단 방법을 참고하여 진단할 수 있다.

목적	• '모바일 전자정부 서비스 관리 지침'에 따라 모바일 대민서비스 개발 · 구축 시에 개발보안을 준수하여 앱(및 서버)을 개발했는지 여부를 확인하기 위한 보안약점 · 취약점 점검 기법 제시 • 소스코드 수준의 보안약점이 제거된 모바일 대민서비스를 보급함으로서 모바일 대민서비스의 안전성 및 신뢰성을 제고
대상	• 모바일 전자정부 대민서비스 개발자 등
범위	• 신규로 개발되거나 유지보수로 변경되는 모바일 앱(하이브리드 앱)의 진단 시 참조
구성	• (2장) 기능 보안취약점 진단 방법 • (3장) 소스코드 보안약점 진단 방법 • (4장) 앱 개발자를 위한 보안공통기반 제공 • (부록) 모바일 보안취약점 점검 도구

1) 행정기관등 : 전자정부법 제2조에서 정의하는 행정기관, 공공기관

2) 「주요정보통신기반시설 기술적 취약점 분석 · 평가 방법 상세가이드 08. Web(웹)」, KISA 홈페이지(http://www.kisa.or.kr)의 자료실 〉 관련법령 〉 안내서 · 해설서 참조

[Part 2]

기능 보안취약점 진단 방법

Part 2

기능 보안취약점 진단 방법

모바일 대민서비스 기능 보안취약점 기준은 『모바일 전자정부 서비스 관리 지침』의 모바일 서비스 앱 대상 보안취약점 점검기준(별표1)을 참고할 수 있다.

기능 보안취약점 점검기준					
번호	점검 항목	설명	비고	And roid	iOS
1	반복 설치 시 오류 발생	앱 반복 설치 시 주요 설정 파일 변경 등의 문제가 발생할 수 있는 취약점	설치	○	○
2	앱 설치 전후 비정상적인 파일 및 디렉토리 설치	앱 설치 전후로 비정상적인 파일 및 디렉토리가 생성될 수 있는 취약점	설치	○	○
3	불필요하거나 과도한 권한 설정	불필요하거나 과도한 권한 설정으로 앱 서비스 목적과 상이한 임의기능 동작이 가능한 취약점	설치	○	○
4	앱 삭제 후 안전성	앱 삭제 시 관련(설치된) 디렉토리 및 파일 이외 파일이 삭제될 수 있는 취약점	삭제	○	○
5	기능의 정상동작	각 기능이 오동작할 수 있는 취약점 *단, 인터페이스 관련 사항은 “모바일 애플리케이션 접근성 지침” 준수여부 등을 고려하여 판단할 수 있음	동작	○	○
6	임의기능 등 악성행위 기능 존재	앱 서비스 목적과 상이한 임의기능, 백그라운드에서 구동되는 악성행위 기능 (프로세스 등)이 존재할 수 있는 취약점	동작	○	○
7	정보 외부 유출	허가된 주소 이외의 주소로 정보 전송이 가능한 취약점	동작	○	○

번호	점검 항목	설명	비고	And roid	iOS
8	자원고갈	정상기능 오동작 또는 취약점을 이용하여 과도한 트래픽 사용 및 배터리를 고갈 시키는 취약점	동작	○	○
9	루팅 및 탈옥 기기에서의 정상 동작	루팅 및 탈옥된 기기에서 앱 설치·동작되어 보안메커니즘 우회 등이 발생할 수 있는 취약점	플랫폼	○	○
10	ID 값의 변경	안드로이드 플랫폼에 설치되는 앱에 부여되는 app_### 과 같은 일반 권한의 UID, GID 등 ID가 임의로 변경될 수 있는 취약점	플랫폼	○	-
11	동일키로 서명된 서로 다른 앱 간의 UID 공유	안드로이드 플랫폼에서 동일한 제작자가 개발한 앱의 경우 동일한 서명키(Key Sign)로 서로 다른 앱 간의 UID를 공유할 수 있는 취약점 *SharedUserID가 설정되지 않아야 함	플랫폼	○	-
12	인텐트 권한의 올바른 설정	기능 사용 요청 및 권한이 부적절한 취약점 *AndroidManifest.xml내에 intent-filter라는 항목에 요청 권한 이외에 과도한 권한을 부여해서는 안됨	플랫폼	○	-
13	인증 정보 생성 강도 적절성	사용자 ID, 패스워드 등의 인증 정보의 생성강도가 낮아 쉽게 추측할 수 있는 취약점	식별 및 인증	○	○
14	중요정보의 평문 저장 및 전송	중요정보(사용자 인증 정보, 사용자 개인 정보, 사용자 위치 정보)는 모바일기기에 평문으로 저장하거나 평문으로 전송되어 외부에 유출될 수 있는 취약점	암호	○	○
15	중요정보 저장 및 전송 시 취약한 암호알고리즘 적용	모바일기기 내 중요정보(사용자 인증 정보, 사용자 개인 정보, 사용자 위치 정보) 저장 또는 서버로 전송시 취약한 암호알고리즘 사용으로 중요정보가 유출될 수 있는 취약점 ※ "국가정보원 국가사이버안전센터"의 검증암호 알고리즘 참조	암호	○	○

번호	점검 항목	설명	비고	And roid	iOS
16	기타 중요 정보의 평문 저장 및 전송	IMEI 정보 등 기타 중요 정보가 모바일기기에 평문으로 저장되거나 평문으로 전송되어 외부에 유출될 수 있는 취약점	암호	○	○
17	기타 중요 정보 저장 및 전송 시 취약한 암호 알고리즘 적용	모바일기기에 IMEI 정보 등 중요정보를 저장 및 전송시 취약한 암호알고리즘 사용으로 중요정보가 유출될 수 있는 취약점	암호	○	○
18	파일 다운로드시 외부주소 및 파일 무결성 우회	설정 파일 또는 업데이트 다운로드시 외부주소 변조 및 파일 무결성 우회 가능 취약점	암호	○	○
19	개인정보 및 개인위치정보 수집 및 활용에 대한 동의	사용자의 동의없이 임의로 개인정보 및 개인위치정보를 수집하고 활용하여 발생할 수 있는 법적 문제 발생 취약점	수집 · 활용	○	○
20	난독화	역공학 기술에 의한 모바일 앱 소스코드 유출 및 보안메커니즘 우회 등이 발생할 수 있는 취약점	배포	○	-

본 장에서는 모바일 앱이 동적으로 실행되면서 발생 할 수 있는 기능상 보안취약점 진단 방법에 대하여 설명한다. 다만, 본 가이드에서 나타낸 진단 절차나 명령어 등은 점검 도구 등의 버전(또는 환경)에 따라 다소 달라질 수 있음을 참고하여야 한다.

제 1 절 설치 및 삭제

"설치 및 삭제" 관련 취약점은 모바일 앱의 설치 및 삭제, 일상적인 기능 이용에 있어 보안상의 문제점은 물론 기능상의 문제점과 설계 결함을 점검한다. 또한 앱을 삭제하였음에도 잔여 파일이 있어서 시스템 성능 저하는 물론 보안상의 위협을 불러일으키는지를 점검한다.

세부 보안취약점

1	반복 설치 시 오류 발생	3	불필요하거나 과도한 권한 설정
2	앱 설치 전후 비정상적인 파일 및 디렉토리 설치	4	앱 삭제 후 안전성

1. 반복 설치 시 오류 발생

가. 설명

- 프로그램 작성 시 개발자의 사소한 실수 등에 의해 주요 설정 파일이 변경되거나, 앱 설치 파일 일부가 삭제되지 않고 남아있는 등의 문제가 발생하면 해당 앱을 삭제하고 재설치하는 과정에서 문제가 발생할 수 있다.

나. 보안대책

- 앱 삭제 시 삭제되지 않고 남아있는 파일 및 디렉토리가 없도록 확실히 삭제되도록 수정한다.
- 앱 설치 또는 삭제 과정에서 시스템 설정을 변경시키는 경우 해당 설정 변경의 이유를 확인하고, 해당 앱 삭제 시 변경된 설정을 복원한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb[3]	Xcode[4], iTunes[5]
루팅 및 탈옥	필요	필요

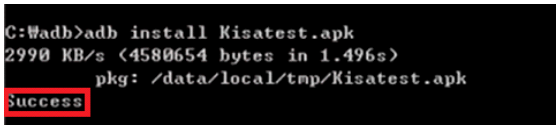
3) adb는 Android 운영체제 모바일 기기를 PC에서 다양하게 제어할 수 있도록 해주는 프로그램으로, 기기에서 다양한 명령어를 실행하는 데 사용할 수 있는 Unix 셸에 관한 액세스를 제공

4) 애플 iOS 운영체제 기반의 앱을 개발하고 분석할 수 있는 개발도구

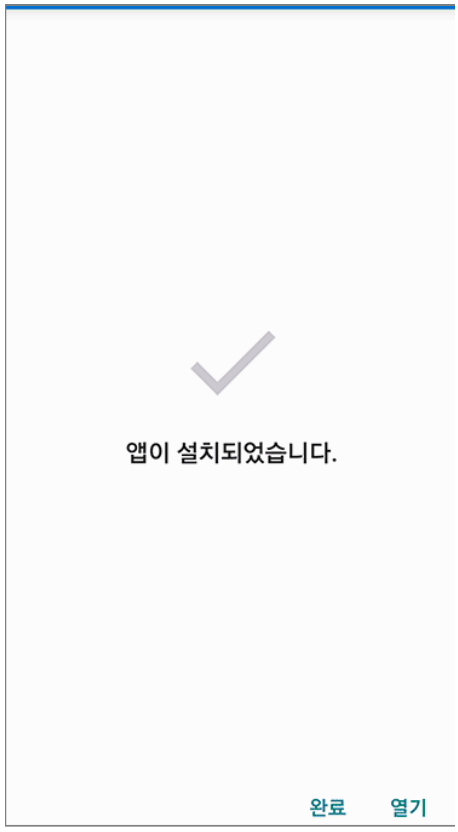
5) Mac이나 PC에서는 iTunes를 이용하여 모바일 기기와 연결하고 애플리케이션 설치 및 삭제 등 동기화 제공

① 점검 대상 앱을 모바일 기기에 설치한다.

- Android 플랫폼은 adb 프로그램을 이용해서 설치하거나, apk 파일을 이동식 드라이브로 인식된 Android 기기로 옮긴 후 수동으로 설치한다.

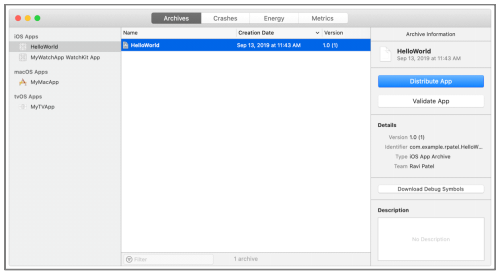


〈 adb(Android)를 이용한 앱 설치 〉



〈 모바일 기기에 수동 설치 〉

- iOS 플랫폼은 Xcode 프로그램 및 iTunes로 모바일 기기에 앱을 설치한다.



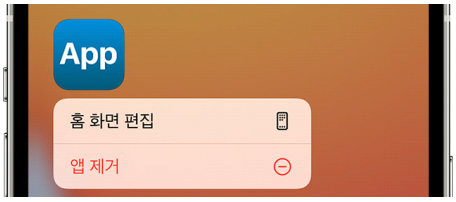
〈 Xcode(iOS)를 이용한 앱 설치 〉

〈 iTunes를 이용한 앱 설치 〉

② 설치된 앱에 대하여 삭제 작업을 수행한다.

- Android 플랫폼은 adb 프로그램을 이용해서 삭제하거나, Android 설정 내 애플리케이션에서 앱 선택 후 삭제버튼를 이용하여 삭제한다.
- iOS 플랫폼은 홈 화면에서 앱을 길게 터치하고 앱 제거를 탭하거나, 앱 보관함에 있는 앱을 길게 터치하고 앱 삭제한다.

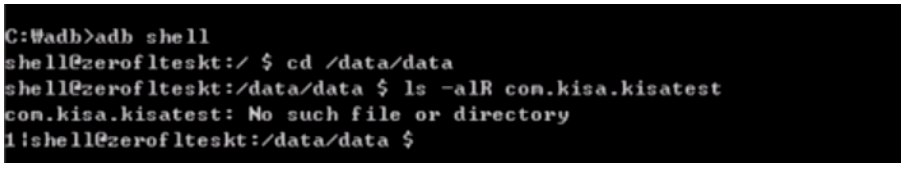
〈 Android adb로 앱 삭제 작업 〉



〈 iOS 설치된 앱 삭제 작업 〉

③ 앱 삭제 시 삭제되지 않고 잔존하는 파일 및 디렉토리가 존재하는지 확인한다.

- (명령어) ls -alR "앱 설치 경로"
  ※ Android 플랫폼은 /data/data 설치패키지 디렉토리 확인
  ※ iOS 플랫폼은 iOS 7 이하 - /var/mobile/Applications/앱_암호화_이름
  iOS 8 이상(앱 실행경로) - /var/containers/Bundle/Application/앱_암호화_이름/앱이름.app
  iOS 8 이상(앱 설치경로) - /var/mobile/Containers/Data/Application/앱_암호화_이름/



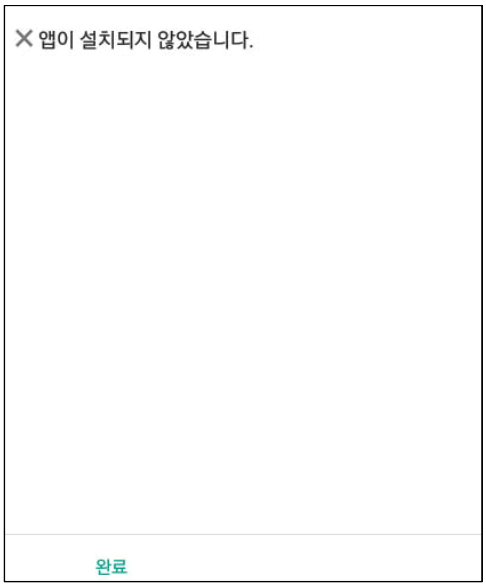
〈 앱 삭제 후 파일 및 디렉토리 완전 삭제 확인 〉

④ ①~③의 작업 과정을 3회 이상 반복하여 수행한다.

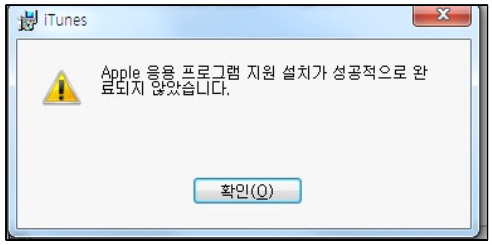
라. 진단기준

- 앱 설치, 삭제, 재설치 시도 중 에러가 발생하면 취약한 것으로 판단한다.

취약한 예시



〈 Android 플랫폼 〉



〈 iOS 플랫폼 〉

참 고 android:allowBackup

- Android 플랫폼의 경우 앱을 삭제하여도, 앱의 설정에 따라 Cloud로 백업을 하게 되는 경우가 있어, 재설치해도 SharedPreference 값을 복원하여 사용할 수 있고 중요정보가 제거되지 않는 경우가 있으므로 AndroidManifest에 allowBackup을 False로 설정여부 확인이 필요함
- 이 속성을 False로 설정하면 모든 어플리케이션 데이터가 adb를 이용하여 저장되는 전체 시스템 백업에 의해서도 어플리케이션의 백업 또는 복원이 수행되지 않음. (기본값 : True)

```
<application
    android:name=".Application"
    android:allowBackup="false"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:supportsRtl="true"
```

2. 앱 설치 전후 비정상적인 파일 및 디렉토리 설치

가. 설명

- 앱 설치 후 앱과 무관한 파일 또는 디렉토리가 생성되거나 설치된 파일 또는 디렉토리에 비정상적인 내용이 포함될 경우 악성코드 등으로 기기사용에 이상이 생길 수 있다.

나. 보안대책

- 설치되는 패키지 내부나 그 외 저장소에 의심되는 파일이나 디렉토리가 존재할 경우 해당 파일 및 디렉토리의 용도를 확인한 후 불필요할 경우 즉시 삭제하고 바이러스 검사 등을 수행한다.

  ※ Android 플랫폼은 /data/data 설치패키지 디렉토리 확인

  ※ iOS 플랫폼은 iOS 7 이하 - /var/mobile/Applications/앱_암호화_이름
  iOS 8 이상(앱 실행경로) - /var/containers/Bundle/Application/앱_암호화_이름/앱이름.app
  iOS 8 이상(앱 설치경로) - /var/mobile/Containers/Data/Application/앱_암호화_이름/

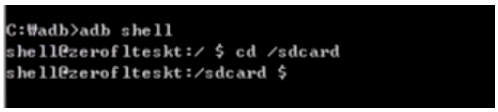
다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb	putty[6]
루팅 및 탈옥	필요	필요

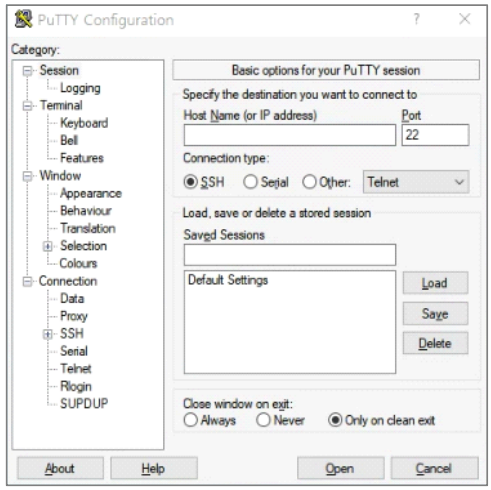
① 모바일 기기 접속을 위해 Android 플랫폼은 adb, iOS 플랫폼의 경우에는 ssh 접속 전용 프로그램인 putty를 이용하여 연결한다.

※ (명령어) adb shell

6) 가장 널리 사용되는 ssh 접속 프로그램으로, 직관적인 인터페이스로 사용이 간편하고 freeware라는 장점을 가지고 있다.



〈 adb(Android)를 이용한 연결 〉

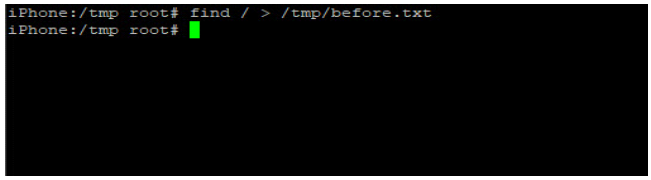


〈 putty(iOS)를 이용한 연결 〉

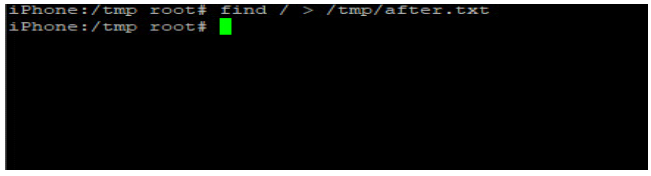
② 분석 대상 앱이 설치되었을 때 모바일 기기의 시스템 내부에 어떤 변화가 생기는지 비교 분석하기 위해 앱 설치 전후 시스템 내부 파일 목록을 작성한다.

※ (명령어) find / 〉 before.txt(설치 전)

※ (명령어) find / 〉 after.txt(설치 후)



〈 설치 전 시스템 파일 목록 생성 〉



〈 설치 후 시스템 파일 목록 생성 〉

③ 앱이 설치된 후 변경된 시스템의 파일들을 파악하기 위해 설치 전의 파일 목록과 설치 후 파일 목록을 diff[7] 명령어로 비교한다.

※ (명령어) diff /before.txt(설치 전 파일 목록) /after.txt(설치 후 파일 목록)

```
C:₩adb>adb shell su -c "diff /sdcard/before.txt /sdcard/after.txt" > diff.txt

C:₩adb>_
```

〈 diff 명령어를 이용한 파일 목록 비교 〉

④ 앱 설치경로 이외의 장소에 파일 및 디렉토리 생성 존재 유무를 확인한다.

라. 진단기준

- 앱 설치 전후로, 설치 전 파일과 설치 후 파일 목록을 비교하여 비정상적인 파일 및 디렉토리 존재 시 취약한 것으로 판단한다.
- Android 플랫폼은 /data/data 디렉토리 이외의 장소에 파일 생성 시 취약한 것으로 판단한다.
- iOS 플랫폼은 /var/mobile/Applications(iOS 7 이하) /var/mobile/Containers/Data/Application/(iOS 8 이상) 디렉토리 이외의 장소에 파일 생성 시 취약한 것으로 판단한다.

취약한 예시

```
root@kisatest:/ # diff /tmp/before.txt /tmp/after.txt
/data/data/com.kisatest/Shared_prefs
/data/data/com.kisatest/Shared_prefs/WebViewChromiumPrefx.xml
/sdcard/kisatest/secret.txt
/tmp/after.txt
```

안전한 예시

```
root@kisatest:/ # diff /tmp/before.txt /tmp/after.txt
/data/data/com.kisatest/Shared_prefs
/data/data/com.kisatest/Shared_prefs/WebViewChromiumPrefx.xml
/tmp/after.txt
```

7) 리눅스에서 두 개의 파일 간 차이점을 비교하는 명령어이다.

3. 불필요하거나 과도한 권한 설정

가. 설명

- 앱 개발 시 필요한 권한 및 기능 이외에 개발상의 편의를 위해 불필요하거나 과도한 권한을 부여할 때가 있다. 과도한 권한을 소유한 앱은 공격자가 이를 악용하여 이용할 가능성이 있다.

나. 보안대책

- 앱에 실제 사용하는 기능 및 권한이 AndroidManifest.xml[8] 파일에서 선언된 권한 간 차이점이 발견될 경우 해당 권한에 대한 요구사항을 분석하고, 불필요하거나 과도한 권한은 수정하거나 삭제한다.
- 안드로이드 6.0 미만 버전의 경우 개별 동의/선택 기능이 제공되지 않으므로 필수적으로 필요한 접근 권한만 설정해야 한다.
- 안드로이드 6.0 이상 버전과 iOS의 경우 선택적 권한 설정이 가능하므로 이용자가 접근 권한 동의 여부를 개별적으로 선택하도록 구현해야 한다.

  ※ iOS의 경우 “설정 – 개인 정보 보호”에서 기기에 저장된 정보를 접근하는 앱에 대해 제어할 수 있다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	apktool	-
루팅 및 탈옥	-	-

① Android 플랫폼의 경우 apk 파일에서 AndroidManifest.xml을 추출하여 접근 권한을 확인, iOS의 경우 기기에 설치한 후 앱 접근 권한을 확인한다.

※ (명령어) apktool.bat d "설치파일“

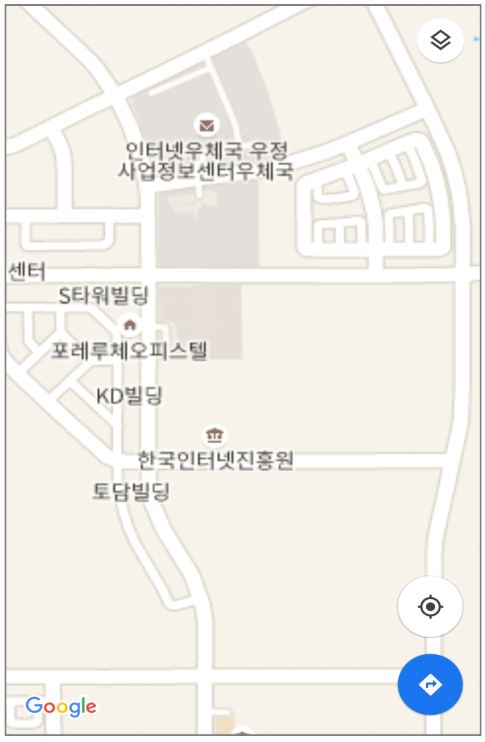
8) 안드로이드 애플리케이션에 대한 각종 정보를 기술한 애플리케이션 명세서를 말한다.

```
C:₩apktool>apktool d Kisatest.apk
: Using Apktool 2.0.0-RC4 on Kisatest.apk
: Loading resource table...
: Decoding AndroidManifest.xml with resources...
: Loading resource table from file: C:₩Users₩mobile003₩apktool₩framework₩1.apk
: Regular manifest package...
: Decoding file-resources...
: Decoding values */* XMLs...
: Baksmaling classes.dex...
```

〈 apktool을 이용한 apk 파일 압축 해제 〉

② 앱을 설치 및 실행하여 앱의 모든 기능을 동작시키면서 필요한 권한(예, SMS, GPS 등)을 확인한다.

〈 SMS 전송 실행 〉



〈 GPS 위치 정보 실행 〉

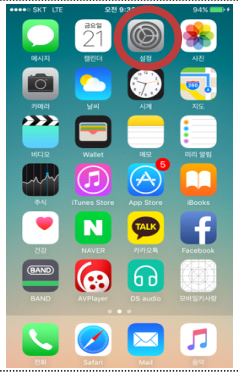
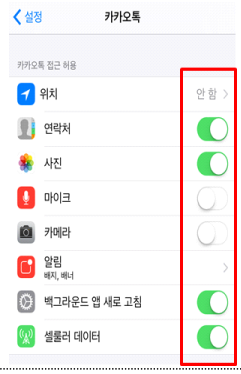
③ Android 플랫폼의 경우, AndroidManifest.xml 파일에서 사용되는 권한[9] (permission)을 확인한다.

※ Android 플랫폼은 apktool로 실행파일(.apk)의 압축을 풀어 Android Manifest.xml 파일을 얻을 수 있음

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="com.google.android.providers.gsf.permission.READ_GSERVICES"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
<uses-permission android:name="android.permission.BLUETOOTH"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.RECEIVE_MMS"/>
<uses-permission android:name="android.permission.FLASHLIGHT"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE"/>
<uses-permission android:name="android.permission.RECEIVE_SMS"/>
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.SET_TIME_ZONE"/>
<uses-permission android:name="android.permission.WRITE_CONTACTS"/>
<uses-permission android:name="android.permission.READ_CONTACTS"/>
<uses-permission android:name="android.permission.WRITE_CALENDAR"/>
<uses-permission android:name="android.permission.READ_CALENDAR"/>
<uses-permission android:name="android.permission.SET_WALLPAPER"/>
```

④ ②에서 확인된 실제 앱에서 사용되는 권한과 ③의 AndroidManifest.xml 파일 내에서 부여되고 있는 권한을 비교한다.

※ iOS 플랫폼은 설정 - 해당 앱 선택 - 접근 권한 내 기능 사용상 사용되지 않는 권한 설정 여부를 확인한다.

1단계	2단계	3단계
		

9) 안드로이드 권한 목록(Permission List)는 동 가이드의 [부록 2]를 참고

라. 진단기준

- Android 플랫폼의 경우 SMS, GPS 등 앱에서 실제 사용하는 기능 및 권한이 manifest 파일 내 uses-permission으로 부여하고 있는 권한과 일치하는지를 확인하여 manifest 파일 내에 불필요하거나 과도한 권한이 부여되어 있다면 취약한 것으로 판단하며, 실제 사용하는 기능 및 권한과 manifest 파일 내의 권한이 일치한다면 권한이 최소로 부여되어 있으므로 안전한 것으로 판단한다.
- iOS 플랫폼의 경우 실제 사용하는 기능과 설정 내 권한이 일치한다면 안전한 것으로 판단한다.

참 고 접근권한

앱 개발 시 접근 권한 최소화

- 앱 개발자는 OS사업자(구글 안드로이드 또는 애플 iOS 등)가 제공한 개발환경에 맞춰 앱 권한이 필요한 범위 내에서 사용되는지를 자체 검토하여 불필요한 앱 접근 권한이 설정되지 않도록 하며, 접근 권한이 과도하고 무분별하게 남용되지 않도록 해야 함

  ※ 방통위, 스마트폰 앱 접근 권한 개인정보보호 안내서)에 따르면, 앱 개발 시 접근 권한 범위를 서비스에 필요한 범위로 최소화하도록 규정

4. 앱 삭제 후 안전성

가. 설명

- 앱 삭제 후 앱과 무관한 파일 또는 디렉토리가 삭제되거나 앱과 관련된 디렉토리가 삭제된 후에도 남아있는 경우 모바일 기기 사용자의 정보를 탈취하거나 악의적인 행위가 발생할 수 있다.

나. 보안대책

- 앱 삭제 시 앱과 관련된 파일 및 디렉토리가 완전히 삭제되도록 한다.
  ※ Android 플랫폼은 /data/data 설치패키지
  Android 플랫폼에서 sd카드 사용 시 /sdcard 디렉토리 확인
  ※ iOS 플랫폼은 iOS 7 이하 - /var/mobile/Applications/앱_암호화_이름
  iOS 8 이상 - /var/mobile/Containers/Data/Application/앱_암호화_이름/

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb	putty
루팅 및 탈옥	필요	필요

① 앱을 설치한 후, 설치 디렉토리(/data/data) 내 디렉토리 및 파일 생성 여부를 확인한다.
※ (명령어) "ls -alR" 명령으로 해당 패키지 이하 모든 파일 검사

```
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 C3CC3EA9-C537-4B72-B455-61BAD7FFC855/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 C9A8E3B2-0201-4478-A507-6293172C2D64/
drwxr-xr-x  7 mobile mobile  224 Mar 18 10:31 CF026BB8-24B8-46AF-9B85-9D24356C7B04/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 D15DE54A-9E7F-40D5-B4E5-A09D58B5C7FE/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 D4F489C8-5681-4BC7-A730-EDD6D663E6FF/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 D8C5FFD1-F437-4FE4-A878-D1F376802982/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 DAC6471C-60CB-4A01-A397-26B95FA5506C/
drwxr-xr-x  8 mobile mobile  256 Sep  7 15:46 DBF09203-2382-4F3C-843A-72E59152BB15/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 E6552D73-E679-4AC3-A079-3D10DD44B571/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:16 E6F4BB05-FFFB-4EC3-B24F-48694485356E/
drwxr-xr-x  7 mobile mobile  224 Jul 12 07:59 E91EE853-B708-474F-A4F5-54CD9DEA40F1/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 EA8C01DD-B2EE-41CC-8B4B-EC19207449AF/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 ECEE8507-3AF1-423F-AF02-D3292C8E535B/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:16 ED4528E9-FEEE-4B3A-953D-35EBB49A8DC4/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 ED7F4D9A-C606-498B-AA95-B6E59EF93E22/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:17 F06BE6C5-B276-411F-A32A-6B573D020ACD/
drwxr-xr-x  7 mobile mobile  224 Mar 18 08:31 F3C18015-0795-4331-B6D5-64B9B13030D8/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:17 F4E87161-DF9E-47AC-8981-26352ABD710A/
```

〈 설치된 디렉토리 확인 〉

② 설치된 앱을 삭제한 후, 설치 디렉토리 내 파일 및 디렉토리 삭제 여부를 확인한다.
※ (명령어) "ls -alR 설치경로" 명령으로 해당 패키지 이하 모든 파일 검사(하위 경로 포함)

```
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:16 ED4528E9-FEEE-4B3A-953D-35EBB49A8DC4/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 ED7F4D9A-C606-498B-AA95-B6E59EF93E22/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:17 F06BE6C5-B276-411F-A32A-6B573D020ACD/
drwxr-xr-x  7 mobile mobile  224 Mar 18 08:31 F3C18015-0795-4331-B6D5-64B9B13030D8/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:17 F4E87161-DF9E-47AC-8981-26352ABD710A/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 F6E90089-1F3C-4C79-9760-6E6BEE20BEC1/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 F7A941C8-F06E-44C0-AC8F-C4E144174D7C/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 F7D841C0-C623-4BC6-A0C1-C3B5049E6D96/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 F83682E4-A50A-4F5F-8337-4B0B395C2740/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 F90702F0-8A14-4D1D-905E-FE2BFA2A9735/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 FD1D90DA-B7A8-4C4F-BDD6-B54CF5396F23/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:13 FDC58939-F8CC-41E6-94D7-EF0185D51BB3/
drwxr-xr-x  7 mobile mobile  224 Mar 16 10:14 FF62C6A9-028F-4F06-B253-D9C7B9345AD0/
iPhone:/var/mobile/Containers/Data/Application root# DBF09203-2382-4F3C-843A-72E59152BB15/
-sh: DBF09203-2382-4F3C-843A-72E59152BB15/: Is a directory
iPhone:/var/mobile/Containers/Data/Application root# ls -alR DBF09203-2382-4F3C-843A-72E59152BB15/
ls: cannot access 'DBF09203-2382-4F3C-843A-72E59152BB15/': No such file or directory
iPhone:/var/mobile/Containers/Data/Application root#
```

〈 삭제 후 디렉토리 완전 삭제 확인 〉

③ 디렉토리 및 파일이 완전하게 삭제되었는지 확인한다.
※ 디렉토리 및 파일이 완전하게 삭제되었을 시 “No such file or directory” 문구를 확인

라. 진단기준

- 앱 삭제 전후로, 아래와 같으면 취약한 것으로 판단한다.
  - (Android) /data/data 디렉토리나 내부에 잔존 파일이 존재
  - (Android) sd카드 사용 시 /sdcard 디렉토리 내부에 잔존 파일이 존재
  - (iOS) /var/mobile/Applications/(iOS 7 이하) /var/mobile/Containers/Data/Application/(iOS 8 이상) 디렉토리 내부에 잔존 파일이 존재
- 앱 설치 전후에 전체 시스템 내부 파일 목록을 작성 및 비교하여 변경 사항이 있다면 취약한 것으로 판단한다.

취약한 예시

```
root@kisatest:/mnt/sdcard/com.kisa.kisatest # ls
ls
secret.txt
test.png
root@kisatest:/mnt/sdcard/com.kisa.kisatest #
```

안전한 예시

```
KISAui-iPhone:/ root# diff /tmp/before.txt /tmp/after.txt
93708a93709
> /private/var/tmp/after.txt
KISAui-iPhone:/ root#
```

< 앱 설치 전/삭제 후 비교 >

```
KISAui-iPhone:/ root# ls -alR 7A70984E-6934-4cdb-A70C-651A40BF3364
ls: cannot access 7A70984E-6934-4cdb-A70C-651A40BF3364: No such file or directory
KISAui-iPhone:/ root#
```

< 삭제 후 디렉토리 확인 >

제 2 절 동작

"동작" 관련 보안취약점은 점검 대상 앱이 가지고 있는 전체 기능을 검사하여 원래의 기능대로 작동하고 있는지를 검사하며 각종 오탈자 및 잘못된 인터넷 링크 등 사용자가 직접적으로 불편을 느낄 수 있는 부분을 점검 및 수정하며, 카메라, 모뎀 등 하드웨어를 이용하기 위해 모바일 기기에서 필요한 권한이 적절하게 선언되어 있는지 점검한다.

세부 보안취약점

1	기능의 정상 동작	3	정보 외부 유출
2	임의기능 등 악성행위 기능 존재	4	자원고갈

1. 기능의 정상동작

가. 설명

- 모바일 앱의 구현된 기능들은 실행하였을 때, 앱의 각 메뉴는 설계서에 제시된 대로 정확하게 동작하여야 한다.
- 앱에서 제공하는 기능이 비정상적으로 동작하거나 사용자 입력값으로 정상 동작이 되지 않는 경우 정확한 보안취약점 점검이 어려우며, 오류정보 노출 등으로 공격자에게 중요정보가 노출될 수 있다.

나. 보안대책

- 기능 오동작, 미동작, 오탈자, 잘못된 링크 등 프로그램상의 오류가 발견될 경우 해당 내용을 수정한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	-	-
루팅 및 탈옥	-	-

① 앱 설치 후 실행하여, 앱의 각 페이지 및 포함된 기능들을 확인한다.

② 수행 도중 기능 정상 동작 여부를 확인하여 설계서에 따라 모바일 앱의 메뉴가 정확하게 동작하는지 확인한다. 단, 페이지가 많은 앱의 경우 시간이 많이 소요될 수 있다.

※ 기능 오동작 또는 미동작(로그인, SMS, GPS 등), 오탈자, 잘못된 링크 등을 점검한다.

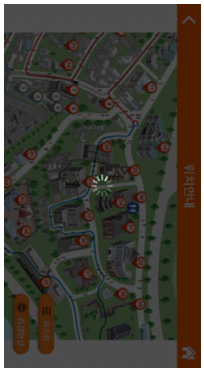
※ 사용자 또는 관리자의 권한에 따라 확인 가능한 기능이 제한될 경우 각 권한에 맞추어 전체 기능에 대한 전수검사를 수행한다.

라. 진단기준

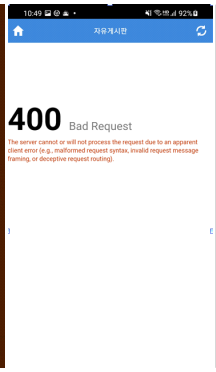
- 기능 오동작, 미동작, 오탈자, 잘못된 링크 등이 발견될 경우 취약한 것으로 판단한다.

취약한 예시

기능 오동작



기능 미동작



에러 페이지 출력

오탈자 표기

잘못된 링크

2. 임의기능 등 악성행위 기능 존재

가. 설명

- 모바일 앱에는 명시적으로 제공하는 기능만 동작해야 하며, 그 외 기능이 백그라운드(background)에서 구동되는 경우 악성 기능이 존재할 수 있다.

나. 보안대책

- 앱 설치 및 실행 후 알 수 없는 포트가 LISTEN 상태로 열려있는 경우 해당 포트의 통신이 악의적인지를 검토하고 포트 백도어(backdoor) 여부를 확인 후 보안취약점이 확인되면 해당 기능에 대하여 수정한다.
- 앱 설치 및 실행 후 생성되는 프로세스가 악의적인지와 프로세스의 권한이 적절하게 설정되었는지를 확인하여 보안취약점이 확인되면 해당 기능에 대하여 수정한다.
  ※ 생성되는 프로세스가 악의적인지의 판단기준은 생성된 프로세스 중에서 원래 정의되지 않은 프로세스가 있는지를 기준으로 함
- 앱 설치 및 실행에서 발생하는 패킷이 과도한 트래픽을 발생하거나, 악의적인 행위인지를 확인하여 보안취약점이 확인되면 해당 기능에 대하여 수정한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb, BurpSuite[10], Fiddler	putty, BurpSuite, Fiddler
루팅 및 탈옥	○	○

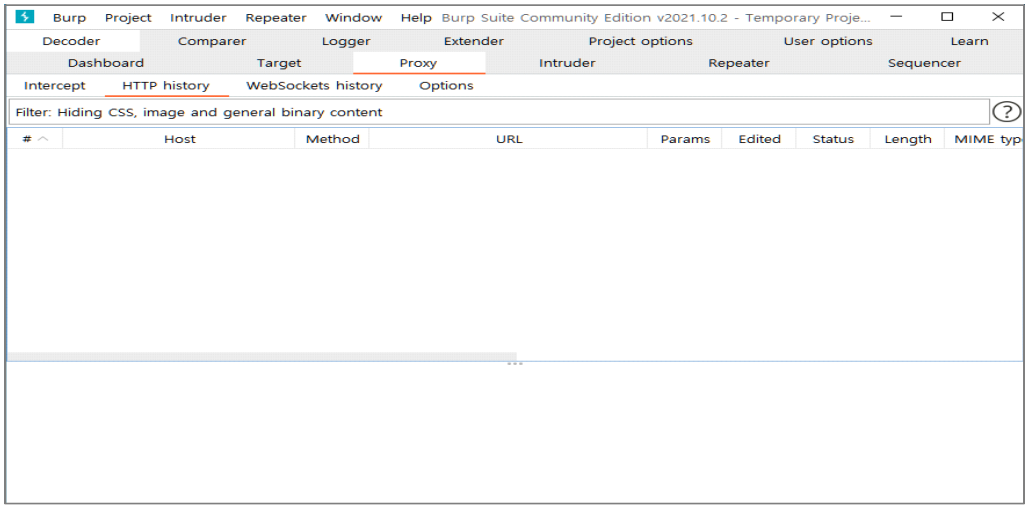
① 앱을 실행하여 앱의 모든 기능(예, 로그인, SMS 전송 등) 및 기능 요구 권한(AndroidManifest.xml 파일 내 permission 등)을 점검하며, 단말 내 설치된 백신 앱을 이용하여 악성코드 및 바이러스 여부를 확인한다.

※ 점검 대상 앱의 실제 사용되는 앱의 권한과 manifest 파일에서 부여하는 권한을 비교하여 사용되지 않는 권한이 부여되어 있는지를 점검한다. 불필요한 권한이 부여되어 있으면 악성 앱, 바이러스에 의해 권한이 악용될 수 있다.

② 프록시 도구(BurpSuite 등)를 이용하여 불필요한 패킷 전송 여부를 확인한다.

※ 점검 앱을 실행하여 각 기능을 실행하면서 프록시 도구에 전송되는 데이터를 살펴보고, 중요 정보(예, 개인정보 등)를 유출 시키는지 점검한다.

10) BurpSuite 는 웹 애플리케이션 보안을 테스트하고 분석하기 위한 Java 애플리케이션 입니다 .



〈 burpsuite 도구 실행 〉

③ ls 명령을 이용하여 불필요한 파일이 생성되는지 확인한다.

※ (명령어) ls -alR 패키지명

```
root@kisatest:/ # cd /data/data/com
root@kisatest:/data/data # ls -alR | grep 'com.kisa.kisatest'
drwxr-x--x u0_a312 u0_a312          2021-11-25 14:00 com.kisa.kisatest
root@kisatest:/data/data #
```

〈 ls 명령 실행 결과 〉

④ ps 명령어를 이용하여 패키지에서 사용하는 PID를 확인한다.

※ (명령어) ps | grep "패키지명"

```
root      9429  2     0       0     0023d6c8 00000000 S kworker/0:1
root      9432  2     0       0     0023d6c8 00000000 S kworker/3:1
root      9452  2     0       0     0023d6c8 00000000 S kworker/1:1
root      9453  2     0       0     0023d6c8 00000000 S kworker/u16:0
root      9454  2     0       0     0023d6c8 00000000 S kworker/u16:1
root      9464  2     0       0     0023d6c8 00000000 S kworker/0:1H
root      9465  2     0       0     0023d6c8 00000000 S kworker/2:0
root      9466  2     0       0     0023d6c8 00000000 S kworker/0:5
root      9479  2     0       0     0023d6c8 00000000 S kworker/5:1
root      9515  2     0       0     0023d6c8 00000000 S kworker/4:1
root      9526  2     0       0     0023d6c8 00000000 S kworker/1:3
system    9546  2983  3953452 97684 ffffffff 880bae78 S com.android.settings
root      9562  2     0       0     0023d6c8 00000000 S kworker/7:2
root      9563  2     0       0     0023d6c8 00000000 S kworker/u16:2
root      9566  2     0       0     0023c9d8 00000000 S kbase_event
u0_a322   9677  2983  1144576 209616 ffffffff 880bae78 S com.kisa.kisatest
```

〈 ps 명령 실행 결과 〉

⑤ netstat 명령을 이용하여 앱 설치 전과 앱 설치 및 실행 후를 비교하여 앱의 PID 값으로 오픈 포트(백도어 포트)를 확인한다.

※ (명령어) netstat -anp 〉 before_port.txt(설치전)

※ (명령어) netstat -anp 〉 after_port.txt(설치후)

※ (명령어) diff before_port.txt after_port.txt(설치 전후 비교)

```
z3q:/ # netstat -anp
Active Internet connections (established and servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Pro
tcp        0      0 0.0.0.0:24800           0.0.0.0:*               LISTEN      2672/ca
tcp        0      0 0.0.0.0:25000           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:5037          0.0.0.0:*               LISTEN      1806/ad
tcp        0      0 0.0.0.0:8470            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:18011           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8470          127.0.0.1:36216         ESTABLISHED -
tcp        0      0 127.0.0.1:36216         127.0.0.1:8470          ESTABLISHED 2175/sy
tcp        0      0 172.16.88.15:24800      172.16.88.2:2630        ESTABLISHED 2672/ca
tcp      130      0 172.16.89.15:45753      142.251.42.202:443      CLOSE_WAIT  2696/co
tcp        0      0 172.16.88.15:18011      172.16.88.2:2633        ESTABLISHED -
tcp        0      0 172.16.89.15:53745      172.217.175.106:443     TIME_WAIT   -
tcp6       0      0 :::18016                :::*                    LISTEN      2175/sy
tcp6       0      0 :::5555                 :::*                    LISTEN      1806/ad
tcp6       0      0 :::9527                 :::*                    LISTEN      2809/co
tcp6       0      0 :::18014                :::*                    LISTEN      2809/co
tcp6       0      0 ::ffff:172.16.89.:35421 ::ffff:216.58.220.1:443 ESTABLISHED 3990/co
taller
tcp6       0      0 ::ffff:172.16.89.:36111 ::ffff:142.250.207.:443 ESTABLISHED 2525/co
tcp6       0    182 ::ffff:172.16.88.1:5555 ::ffff:172.16.88.2:2673 ESTABLISHED 1806/ad
tcp6       0      0 ::ffff:172.16.89.:35419 ::ffff:216.58.220.1:443 ESTABLISHED 3460/co
tcp6       0      0 ::ffff:172.16.89.:60771 ::ffff:74.125.203.:5228 ESTABLISHED 2525/co
udp        0      0 172.16.89.15:68         172.16.89.2:67          ESTABLISHED 2175/sy
udp6       0      0 :::9528                 :::*                    2809/co
udp6       0      0 :::18012                :::*                    2175/sy
udp6       0      0 :::22469                :::*                    -
```

〈 netstat 명령 실행 결과 〉

⑥ ps 명령을 이용하여 앱 설치 전후 프로세스 비교하여 앱의 PID 값으로 불필요한 프로세스가 실행되는지를 확인한다.

※ (명령어) ps 〉 before_ps.txt(설치전)

※ (명령어) ps 〉 after_ps.txt(설치후)

※ (명령어) diff before_ps.txt after_pstxt(설치 전후 비교)

라. 진단기준

- 명시적인 기능 외 백그라운드로 실행되는 의심스러운 행위가 있으면 취약한 것으로 판단한다.
- 분석 대상 앱이 불필요한 패킷을 외부로 전송하거나 악의적인 파일, 프로세스 등을 생성할 경우 이는 모바일 기기에 잠재적인 위협을 가져올 수 있는 행위로 의심하여 취약한 것으로 판단한다.
- 앱 설치 파일이나 앱 설치 후 바이러스 및 악성코드 검사를 수행하여 악성코드 및 바이러스 감염 여부를 확인한다.

3. 정보 외부 유출

가. 설명

- 앱 서버 또는 업데이트 서버 등 허가된 주소(IP, URL 등) 이외의 주소로 정보 전송이 발생하는 경우, 해당 정보가 외부로 유출될 가능성이 존재한다.

나. 보안대책

- 전송되는 정보가 허가된 주소 이외의 다른 주소로 전송될 경우 관련 정보(사용되는 라이브러리, 전송되는 정보 등)를 확인하고 외부로 정보가 유출된다고 판별되면 관련 라이브러리나 컴포넌트에 대한 삭제 또는 수정한다.

다. 진단방법

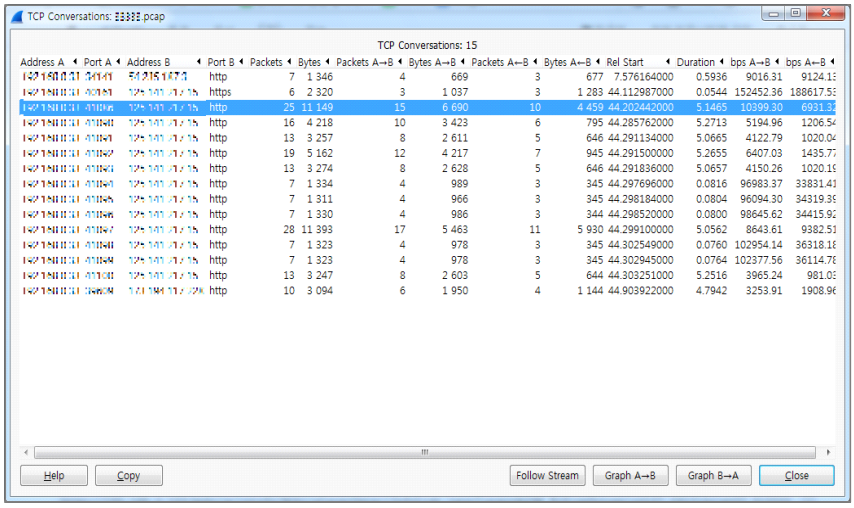
모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	BurpSuite, Fiddler, WireShark[11]	BurpSuite, Fiddler, WireShark
루팅 및 탈옥	-	-

① 앱을 설치한 후, 실행하여 앱의 모든 기능(로그인 등)을 수행한다.

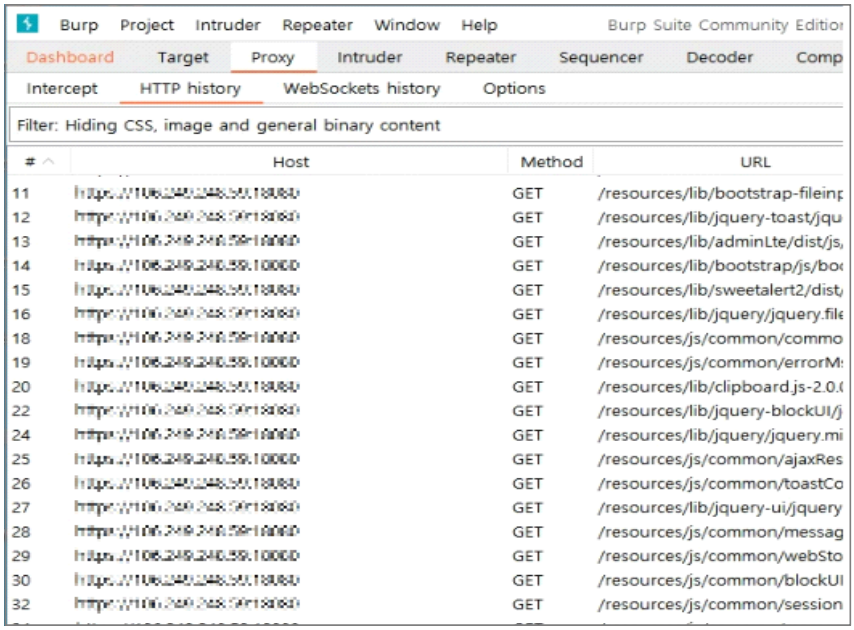
② 프록시 도구(BurpSuite 등) 및 패킷분석 도구(WireShark 등)를 이용하여 전송되는 정보를 캡처한 후 허가된 주소로 패킷이 전송되는지 확인한다.

※ 허가된 주소는 앱 서버 또는 업데이트 서버 주소 등을 의미하며, 명세서에서 해당 주소를 확인한다.

11) WireShark는 패킷 스니퍼 , 프로그램 사용 및 프로토콜 분석기 캡처 기능을 제공하는 네트워크 패킷 분석 프로그램입니다.



〈 패킷분석 도구(WireShark)로 연결된 IP 확인 〉



〈 프록시 도구(BurpSuite)로 연결된 IP 확인 〉

라. 진단기준

- 전송되는 정보가 허가된 주소 이외의 다른 주소로 전송되면 취약한 것으로 판단한다.

4. 자원고갈

가. 설명

- 개발자의 코딩 오류로 인한 비정상적인 자원 사용, 앱 사용 시 발생하는 반복 작업 및 동기화 등으로 과도한 트래픽이 발생하거나 배터리가 빠르게 소모될 수 있다.

나. 보안대책

- 배터리 · 트래픽 점검 앱을 이용하여 비정상적으로 많은 배터리 및 트래픽을 발생시키는지를 확인 후 수정한다.

  ※ 배터리 및 트래픽 소모 발생량에 대한 판단기준은 자체적으로 기준을 수립하여 판단한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	-	-
루팅 및 탈옥	-	-

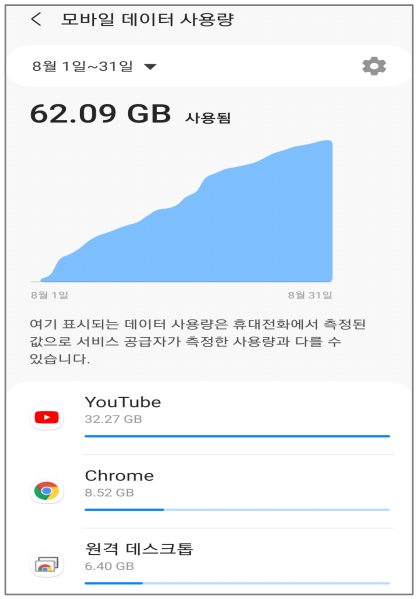
① 앱을 실행하여 사용되는 모든 기능 및 권한(SMS, GPS 등)을 확인한다.

② 모바일 기기에서 기본으로 제공되는 앱을 이용하여 배터리 사용량, 트래픽 사용량을 확인한다.

※ 배터리 확인 : (안드로이드) 설정 → 디바이스 관리 → 배터리
(아이폰) 설정 → 배터리

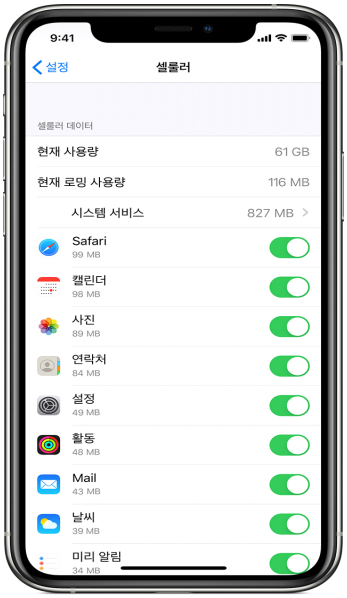
※ 트래픽 확인 : (안드로이드) 설정 → 연결 → 데이터 사용 → 모바일 데이터 사용량
(아이폰) 설정 - 셀룰러

〈 배터리 사용량 앱 예시- android 〉



〈 트래픽 사용량 앱 예시 - android 〉

〈 배터리 사용량 앱 예시- iOS 〉



〈 트래픽 사용량 앱 예시 - iOS 〉

라. 진단기준

- 앱 실행 시 배터리 사용량이 과도하게 증가하게 되면 취약한 것으로 판단한다.
- 앱 실행 시 트래픽 사용량이 과도하게 증가하게 되면 취약한 것으로 판단한다.

제 3 절 플랫폼

"플랫폼" 관련 보안취약점은 루팅[12] 및 탈옥[13] 기기에서 앱이 설치·동작되어 보안 메커니즘 우회 등이 발생할 수 있는 Android 및 iOS 플랫폼 기반의 취약점과 ID 값의 변경, UID 공유, 인텐트 권한 설정 등으로 발생이 가능한 Android 플랫폼 기반의 취약점을 의미한다.

세부 보안 취약점

1	루팅 및 탈옥 기기에서의 정상 동작	3	동일키로 서명된 서로 다른 앱 간의 UID 공유(Android)
2	ID 값의 변경(Android)	4	인텐트 권한의 올바른 설정 (Android)

1. 루팅 및 탈옥 기기에서의 정상 동작

가. 설명

- 이용자 단말기의 루팅(Android 기반 플랫폼) 및 탈옥(iOS 기반 플랫폼)으로 변조된 플랫폼(Android, iOS)은 해당 단말기에서 관리자 권한을 획득할 수 있으며, 플랫폼이 변조된 단말기에서는 악의적인 공격자가 애플리케이션 분석, 메모리 분석, 악성코드 감염, 중요 파일 및 중요정보 접근 등 다양한 공격을 수행할 수 있다.

나. 보안대책

- 앱 실행 시 수시로 플랫폼 변조 관련 파일을 탐지하도록 프로그램을 수정한다.
- 모바일 대국민 보안공통기반의 "단말 위·변조 방지" 기능을 사용하여 플랫폼 변조(루팅·탈옥) 여부를 검사하여 위·변조된 단말 접근을 차단한다.

12) 루팅은 모바일 기기에서 구동되는 안드로이드 운영체제상에서 최상위 권한(루트 권한)을 얻음으로 해당 기기의 생산자 또는 판매자 측에서 걸어 놓은 제약을 해제하는 행위이다

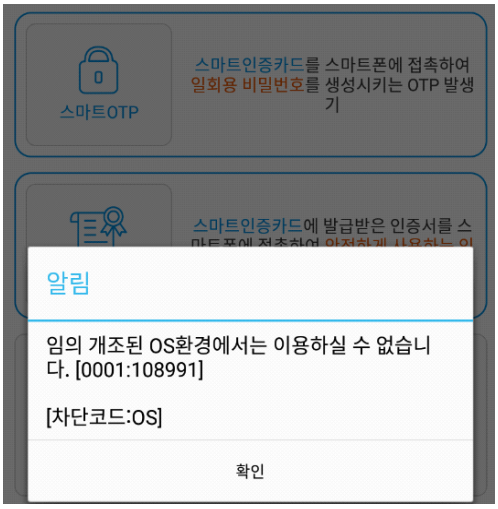
13) 탈옥은 iOS의 샌드박스 제한을 풀어 타 회사에서 사용하는 서명되지 않은 코드를 실행할 수 있게 하는 행위이다.

다. 진단방법

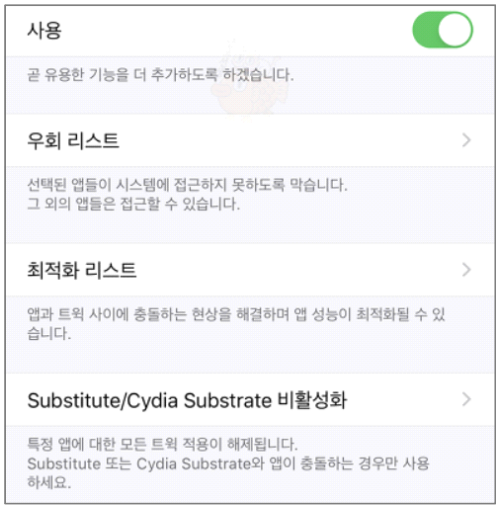
모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb	Xcode
루팅 및 탈옥	필요	필요

① 루팅 및 탈옥된 모바일 기기에서 앱을 설치하고 실행한다.

② 앱의 루팅, 탈옥 방지 기능의 동작 여부를 확인한다. 단, 루팅/탈옥 유틸리티 도구를 사용할 경우 임시로 루팅/탈옥을 하지 않은 것처럼 변경할 수 있으니, 프로세스 설치 여부 및 su 권한을 확인하여 루팅/탈옥 여부를 검사하는지 확인한다.



〈 루팅 및 탈옥 방지 기능 동작 〉



〈 탈옥 우회 유틸리티 도구 〉

라. 진단기준

- 앱이 루팅, 탈옥된 기기에서 작동할 경우 취약한 것으로 판단한다.

참 고 안드로이드 OS 변조 탐지 코드 예시

```
1   WebView wv = (WebView) findViewById(R.id.webView1);
2   wv.getSettings().setJavaScriptEnabled(true);
3   wv.getSettings().setPluginsEnabled(true);
4   wv.setHorizontalScrollBarEnabled(false);
5   wv.setVerticalScrollBarEnabled(false);
6   wv.setOnTouchListener(new View.OnTouchListener() {
7   @Override
8   public boolean onTouch(View arg0, MotionEvent arg1) {
9   switch (arg1.getAction()){
10                  case MotionEvent.ACTION_DOWN :
11                  [루팅 방지 로직 삽입 or 루팅 방지 함수 호출]
12                  break;
13
14  return false;
15  }
16  });
```

참 고 iOS 탈옥 폰 탐지 항목

구분	탐지항목
파일 탐지	/etc/fstab /etc/master.passwd /usr/sbin/sshd /etc/apt /bin/bash /Application/Cydia.app
애플리케이션 탐지	Cydia SBSetting Winterboard FakeCarrier
시스템 탐지	특정 포트 사용 여부(22) 특정 프로세스 동작 여부(sshd) Shell 권환 확인(root)
URL	cydia://package/com.example.package

참 고 iOS 탈옥 시 생성 파일

/private/var/stash
/private/var/lib/apt
/private/var/tmp/cydia.log
/private/var/lib/cydia
/private/var/mobile/Library/SBSettings/Themes
/Library/MobileSubstrate/MobileSubstrate.dylib
/Library/MobileSubstrate/DynamicLibraries/Veency.plist
/Library/MobileSubstrate/DynamicLibraries/LiveClock.plist
/System/Library/LaunchDaemons/com.ikey.bbot.plist
/System/Library/LaunchDaemons/com.saurik.cydia.Startup.plist
/var/cache/apt
/var/lib/apt
/var/lib/cydia
/var/log/syslog
/var/tmp/cydia.log
/bin/bash
/bin/sh
/usr/sbin/sshd
/usr/libexec/ssh-keysign
/usr/sbin/sshd
/usr/bin/sshd
/usr/libexec/sftp-server
/etc/ssh/sshd_config
/etc/apt
/Applications/Cydia.app
/Applications/RockApp.app
/Applications/Icy.app
/Applications/WinterBoard.app
/Applications/SBSettings.app
/Applications/MxTube.app
/Applications/IntelliScreen.app
/Applications/FakeCarrier.app
/Applications/blackra1n.app

2. ID 값의 변경(Android)

가. 설명

- Android는 서로 다른 앱의 데이터에 접근하지 못하도록 만들기 위해서 다른 UID(User ID)를 할당하며, 특정 파일에 대한 접근 권한을 얻기 위해 그 권한과 관련된 GID(Group ID)를 할당한다.
- UID와 GID가 변경되어 중요한 데이터 또는 권한에 접근할 수 있게 되었을 경우 시스템 내 중요정보가 탈취될 수 있다.

나. 보안대책

- 앱의 설치 권한과 실행 권한이 같은지를 확인한 후, 해당 권한이 같지 않다면 해당 앱의 권한 상승 기능을 제거하고 최소한의 권한으로 실행될 수 있도록 프로그램을 수정한다.

다. 진단방법

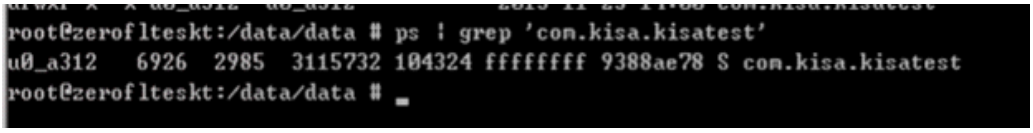
모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb	-
루팅 및 탈옥	필요	-

① 앱을 설치한 후, /data/data 디렉토리 내 앱 설치 파일의 권한(UID, GID)을 확인한다.
※ (명령어) "ls -al 패키지명" 명령을 이용해 UID, GID 파일 권한 확인

```
C:₩adb>adb shell
shell@zerofIteskt:/ $ su
root@zerofIteskt:/ # cd /data/data
root@zerofIteskt:/data/data # ls -al | grep 'com.kisa.kisatest'
drwxr-x--x u0_a312  u0_a312           2015-11-23 14:00 com.kisa.kisatest
root@zerofIteskt:/data/data #
```

〈 /data/data 디렉토리 내 앱 설치 파일 권한 확인 〉

② 모바일 기기에서 앱을 실행한 후, ps 명령을 이용한 앱의 실행 권한(UID, GID)을 확인한다.
※ (명령어) "ps | grep 패키지명" 명령을 이용해 UID, GID 실행 권한 확인

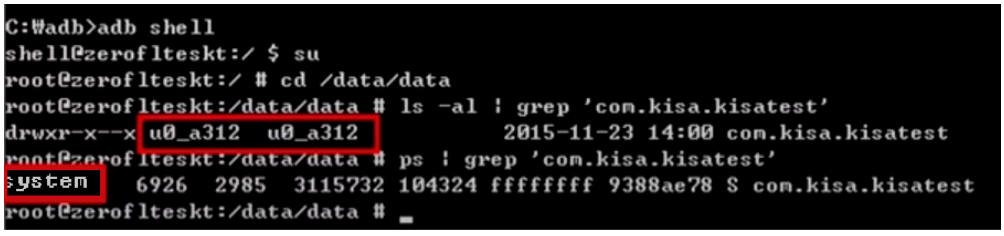


〈 ps 명령을 이용한 실행 권한 확인 〉

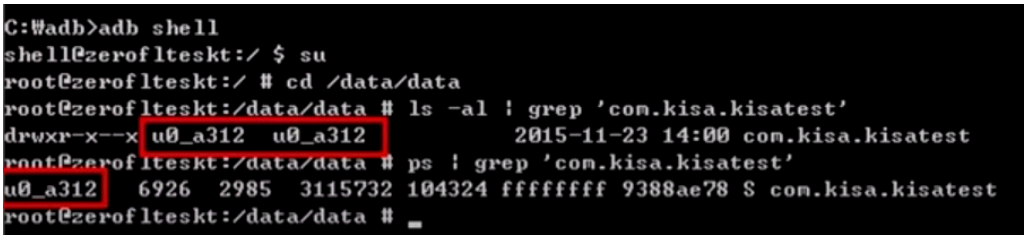
라. 진단기준

- 설치된 앱의 설치 권한과 실행 권한이 다를 경우(설치 UID와 실행 UID가 다를 경우) 취약한 것으로 판단한다.

취약한 예시



안전한 예시



3. 동일키로 서명된 서로 다른 앱 간의 UID 공유(Android)

가. 설명

- Android 시스템에서 개발자는 동일 키로 서명된(Key Sign) 앱 제작이 가능하며, 동일 키로 서명된 앱 간 SharedUserId[14]를 이용하여 UID를 공유할 수 있다.
- 이 경우 서로의 데이터에 접근할 수 있으며, 같은 프로세스에서 실행될 수 있다.
  ※ Android API 29버전 이상에서는 SharedUserId 사용이 중단됨

나. 보안대책

- AndroidManifest.xml 파일 내에 SharedUserId가 설정된 경우 해당 부분을 제거하는 것을 권장한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	apktool	-
루팅 및 탈옥	필요	-

① 앱을 apktool 툴을 이용해 압축 해제한다.
※ (명령어) apktool.bat d xxx.apk

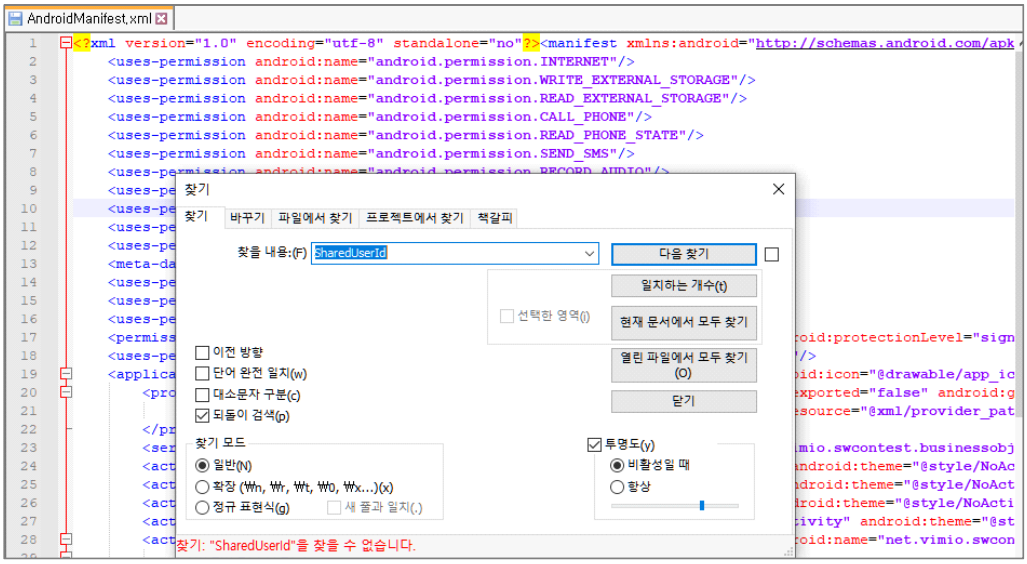
```
C:₩apktool>apktool d Kisatest.apk
: Using Apktool 2.0.0-RC4 on Kisatest.apk
: Loading resource table...
: Decoding AndroidManifest.xml with resources...
: Loading resource table from file: C:₩Users₩mobile003₩apktool₩framework₩1.apk
: Regular manifest package...
: Decoding file-resources...
: Decoding values */* XMLs...
: Baksmaling classes.dex...
```

〈 apktool을 이용한 apk 파일 압축 해제 〉

14) Android는 애플리케이션마다 리눅스 User ID를 할당하는데 sharedUserId는 서로 다른 애플리케이션에서 같은 User ID를 공유할 수 있음

② AndroidManifest.xml 파일 내 sharedUserId 존재 여부를 확인한다.

※ AndroidManifest.xml 파일에서 SharedUserId 문자열 검색



〈 AndroidManifest.xml 파일 내 sharedUserId 존재 여부 확인 〉

라. 진단기준

- AndroidManifest.xml 파일 내에 SharedUserId가 설정된 경우 취약한 것으로 판단한다.

취약한 예시

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8" standalone="no"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android" andoid:sharedUserId="kisa.user" package="com.kisa.kisat
3      <uses-permission android:name="android.permission.INTERNET"/>
4      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
6      <uses-permission android:name="android.permission.CALL_PHONE"/>
7      <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
8      <uses-permission android:name="android.permission.SEND_SMS"/>
```

안전한 예시

AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8" standalone="no"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.kisa.kisatest" platformBuildVersionCode="2
3      <uses-permission android:name="android.permission.INTERNET"/>
4      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
5      <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE"/>
6      <uses-permission android:name="android.permission.CALL_PHONE"/>
7      <uses-permission android:name="android.permission.READ_PHONE_STATE"/>
8      <uses-permission android:name="android.permission.SEND_SMS"/>
9      <uses-permission android:name="android.permission.RECORD_AUDIO"/>
```

4. 인텐트 권한의 올바른 설정(Android)

가. 설명

- Android에서 앱 액티비티, 서비스, Broadcast Receiver는 인텐트로 활성화된다. 인텐트 필터는 Activity나 서비스가 할 수 있는 작업과 Receiver가 처리할 수 있는 Broadcast 유형을 선언한다.
- 이 경우 intent-filter 내 과도한 권한이 부여될 경우 의도하지 않은 인텐트 처리가 발생할 수 있다.

나. 보안대책

- 해당 앱의 실제 동작에 필요한 intent 권한만 설정되도록 수정한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	apktool	-
루팅 및 탈옥	-	-

① apktool을 이용하여 분석 대상 앱의 apk파일 압축을 해제한다.

② AndroidManifest.xml 파일 내 intent-filter 항목을 검사하여, 필터링하는 intent를 앱이 사용하는지 확인한다.

※ intent-filter 내에 필터링하는 항목은 action, data, category 등의 항목이 존재

```
<activity android:configChanges="keyboardHidden|orientation|screenSize" android:name="net.
    <intent-filter>
        <action android:name="android.intent.action.VIEW"/>
        <category android:name="android.intent.category.DEFAULT"/>
        <category android:name="android.intent.category.BROWSABLE"/>
        <data android:scheme="http"/>
        <data android:scheme="https"/>
        <data android:host="youtube.com"/>
        <data android:host="m.youtube.com"/>
        <data android:host="www.youtube.com"/>
        <data android:host="youtu.be"/>
    </intent-filter>
```

〈 AndroidManifest.xml의 intent-filter 〉

③ 정의된 인텐트를 직접 adb로 실행하여 기능이 동작하는지를 확인한다.

※ adb shell am start "intent" 명령어로 activity를 실행할 수 있음

```
//Action만 갖고 있는 Intent 실행
$ adb shell am start -a android.intent.action.MY_ACTION
//Action, Category를 갖고 있는 Intent 실행
$ adb shell am start -a android.intent.action.MY_ACTION -c
android.intent.category.MY_CATEGORY
//Action과 Data를 갖고 있는 Intent 실행
$ adb shell am start -a android.intent.action.MAIN -d https://google.com
//Package가 설정된 Intent 실행
$ adb shell am start -a android.intent.action.MAIN -p com.example.myapp
//Package, Component가 설정된 Intent 실행
$ adb shell am start -a android.intent.action.MAIN -n
com.example.myapp/com.example.myapp.MainActivity
```

```
c:\adb> adb shell am start -a android.intent.action.VIEW -d "https://www.google.com"
Starting: Intent { act=android.intent.action.VIEW dat=https://www.google.com/... }

c:\adb>
```

〈 adb를 이용한 intent 실행 〉

라. 진단기준

- AndroidManifest.xml 내부의 intent-filter 항목에 과도하게 많은 필터링 정의가 되어 있다면 취약한 것으로 판단한다.

취약한 예시

```
        </intent-filter>
    </service>
    <receiver android:name="
        <intent-filter>
            <action android:name="android.intent.action.PACKAGE_INSTALL"/>
            <action android:name="android.intent.action.PACKAGE_ADDED"/>
            <action android:name="android.intent.action.PACKAGE_REPLACED"/>
            <action android:name="android.intent.action.SCREEN_ON"/>
            <action android:name="android.intent.action.SCREEN_OFF"/>
            <action android:name="android.intent.action.PHONE_STATE"/>
            <action android:name="ACTION.RESTART.PersistentService"/>
            <action android:name="com.tomatosystem.exmobilelib.impl.service.on"/>
            <action android:name="com.tomatosystem.exmobilelib.impl.service.off"/>
            <action android:name="com.tomatosystem.exmobilelib.impl.service.save"/>
            <action android:name="com.tomatosystem.exmobilelib.impl.service.load"/>
            <action android:name="com.tomatosystem.exmobilelib.impl.service.addAutoLogin"/>
            <action android:name="com.tomatosystem.exmobilelib.impl.service.removeAutoLogin"/>
            <category android:name="android.intent.category.DEFAULT"/>
            <data android:scheme="package"/>
        </intent-filter>
    </receiver>
```

안전한 예시

```
        <activity
            android:name='

android:configChanges="orientation|keyboard|keyboardHidden|locale|touchscree
n|navigation|screenLayout|uiMode|screenSize"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.VIEW" />

                <category android:name="android.intent.category.DEFAULT" />
                <category android:name="android.intent.category.BROWSABLE" />

                <data
                    android:
            </intent-filter>
        </activity>
```

참 고 intent-filter 속성

android:icon

필터에 설명된 기능이 컴포넌트에 있다는 것을 사용자에게 설명할 때 상위 활동, 서비스, broadcast receiver를 나타내는 아이콘이다. 이 속성은 이미지 정의가 포함된 Drawable 리소스의 참조로 설정해야 한다. 기본값은 상위 컴포넌트의 icon 속성에서 설정된 아이콘이며, 상위 항목이 아이콘을 지정하지 않으면 기본값은 <application> 요소에서 설정된 아이콘이 됩니다.

android:label

상위 컴포넌트를 나타내는, 사용자가 읽을 수 있는 라벨이다. 필터에 설명된 기능이 컴포넌트에 있다는 것을 사용자에게 설명할 때 사용된다.
기본값은 상위 컴포넌트에서 설정한 라벨이며, 상위 항목에서 라벨이 지정되지 않으면 기본값은 <application> 요소의 label 속성에서 설정된 라벨이 된다.

android:priority

필터에 설명된 유형의 인텐트를 처리하는 것과 관련하여 상위 컴포넌트에 부여해야 하는 우선순위이다. 이 속성은 액티비티와 broadcast receiver를 처리하며 액티비티가 필터와 일치하는 인텐트에 대한 우선순위 정도를 정한다. 우선순위가 서로 다른 여러 액티비티가 인텐트를 처리할 수 있는 경우 Android에서는 우선순위 값이 더 높은 액티비티만 인텐트의 잠재적 타겟으로 간주한다.
또한 broadcast 메시지를 수신하기 위해 broadcast receiver를 실행하는 순서를 제어하며, 우선순위가 높은 수신기가 먼저 호출된다. 순서는 동기 메시지에만 적용되며 비동기 메시지와 관련해서는 무시된다.
※ broadcast를 수신하는 특정 순서를 꼭 적용해야 하는 경우나 Android에서 특정 액티비티를 다른 액티비티에 비해 먼저 처리하도록 강제하려는 경우에만 이 속성을 사용한다.

android:order

여러 필터가 일치할 때 필터가 처리되는 순서이며, order는 priority와는 다르다.
※ Android API 28 이후 도입

제 4 절 식별 · 인증 및 암호

"식별 · 인증 및 암호" 보안취약점은 사용자 ID, 비밀번호 등의 인증 정보 생성 강도가 낮아 쉽게 추측되거나, 중요정보(사용자 인증 정보, 사용자 개인 정보, 사용자 위치 정보) 또는 기타 중요 정보(IMEI, IMSI 정보 등)가 모바일 기기에 평문으로 저장되거나 평문으로 전송되어 외부에 유출될 수 있는 취약점을 의미한다.

세부 보안취약점

1	인증 정보 생성 강도 적절성	4	기타 중요정보의 평문 저장 및 전송
2	중요정보의 평문 저장 및 전송	5	기타 중요정보 저장 및 전송 시 취약한 암호 알고리즘 적용
3	중요정보 저장 및 전송 시 취약한 암호 알고리즘 적용	6	파일 다운로드시 외부주소 변조 및 파일 무결성 우회

1. 인증 정보 생성 강도 적절성

가. 설명

- 사용자 비밀번호 등 인증 정보에 안전한 비밀번호 조합규칙(비밀번호의 길이, 허용 문자, 조합 등)이 적용되지 않으면 무차별 대입 공격에 인증 정보가 누출될 수 있다.
- 또한, 비밀번호 재발급(아이디/비밀번호 찾기 등)이 취약한 경우에도 공격자가 불법적으로 다른 사용자의 비밀번호를 획득, 변경, 복구할 수 있다.

나. 보안대책

- 비밀번호는 한국인터넷진흥원의 『패스워드 선택 및 이용 안내서』의 안전한 패스워드를 적용한다.

  ※ 방법1. 두 종류 이상의 문자 구성과 8자리 이상의 길이로 구성된 문자열
  (문자 종류는 알파벳 대문자와 소문자, 특수문자, 숫자의 4가지임)

  ※ 방법2. 10자리 이상의 길이로 구성된 문자열
  (숫자로만 구성할 경우 취약할 수 있음)

- 비밀번호 입력정보가 사용자에게 노출되지 않도록 마스킹(예, “*”) 처리를 수행한다.
- 사용자 비밀번호를 발급해주거나 확인해줄 때 화면에 바로 출력해주는 것이 아니라 인증된 사용자 메일이나 SMS로 전송해주어야 한다.
- 비밀번호 재발급 검증 실패에 대한 임계값을 설정하여 일정 횟수 이상 실패한 경우 다른 방식으로 비밀번호 찾기 기능을 제공하여야 한다.
- 검증 후 기존의 비밀번호가 아닌 임시 비밀번호를 발급하도록 설계하고, 발급받은 즉시 새로운 비밀번호로 재설정하도록 하여야 함

다. 진단방법

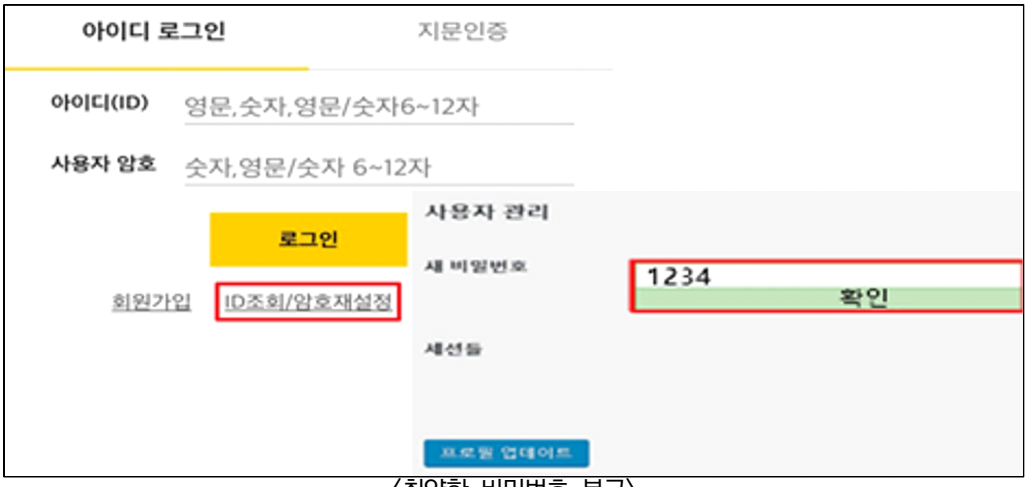
모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	-	-
루팅 및 탈옥	-	-

① 앱을 실행하여 사용자 ID/비밀번호를 입력할 수 있는 회원가입 메뉴 등의 존재 여부를 확인한다.

② 간단한 비밀번호(예 : 1234)로 회원가입 등이 가능한지 확인한다.

③ 재설정 또는 비밀번호 찾기 기능으로 임시로 발급되는 비밀번호 몇 개를 획득하여 사용자의 연락처, 주소, 메일 주소, 일정 패턴을 임시 발급 비밀번호로 이용하고 있는지 확인하고, 임시 발급된 비밀번호를 인증된 사용자 메일이나 SMS로 전송하는지 확인

〈 단순한 ID/비밀번호로 회원가입 〉



〈취약한 비밀번호 복구〉

라. 진단기준

- 안전하지 않은 비밀번호 생성규칙으로 비밀번호 생성·변경할 수 있으면 취약한 것으로 판단한다.
- 비밀번호 재설정 시 일정 패턴으로 재설정되고 화면에 바로 출력되면 취약한 것으로 판단한다.

취약한 예시

안전한 예시

2. 중요정보의 평문 저장 및 전송

가. 설명

- 중요 정보(인증 정보, 개인 정보, 위치 정보)가 모바일 기기에 평문으로 저장되거나 평문으로 전송되는 경우 공격자에게 중요정보가 노출될 수 있다.

나. 보안대책

- 중요정보가 모바일 기기에 평문으로 저장될 경우 해당 저장 기능에 112비트[15] 이상의 보안 강도를 갖는 안전한 암호화 기능을 적용한다.
- 로그인 시 서버로 전송되는 인증 정보는 TLS 통신 방법[16]을 이용하는 등 암호화하여 전송한다.
- 모바일 대국민 보안공통기반의 “E2E 암호화” 기능을 활용하여 전송되는 인증 정보를 암호화하여 전송한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	**iOS 플랫폼**
점검 도구	adb, SQLite[17], BurpSuite, Fiddler, WireShark	putty, BurpSuite, Fiddler, WireShark
루팅 및 탈옥	필요	필요

① 앱을 설치 및 실행한 후 로그인, 사용자 위치 확인 등 중요정보가 입력되어 저장되거나 사용되는 기능을 실행한다.

② grep 명령어로 중요정보 문자열을 검색하여 중요정보가 평문으로 저장되는지 확인한다.

※ (명령어) “grep -rinoE ‘검색어(예, password)’ *” 명령을 이용해 평문으로 저장되었는지 확인

※ Android 플랫폼의 경우, SQLite DB를 사용하여 모바일 기기 내 DB를 쉽게 분석할 수 있음

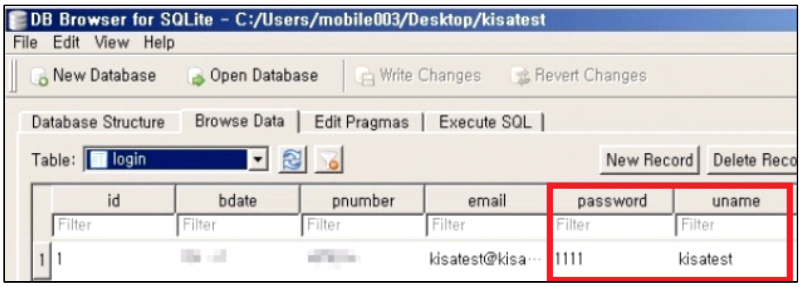
15) 보안강도 112비트에 대한 암호알고리즘 안전성 유지기간은 2011년부터 2030년까지로 권장하고 있다. (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018)

16) SSL 프로토콜은 패킷 탈취 등 보안취약점을 발생할 수 있으므로 보안이 강화된 TLS v1.2 이상을 사용하기를 권장

17) 서버가 없이 구축이 가능한 SQL 데이터베이스 엔진으로 모바일 앱(안드로이드)에서 DB 분석을 위해 사용한다.

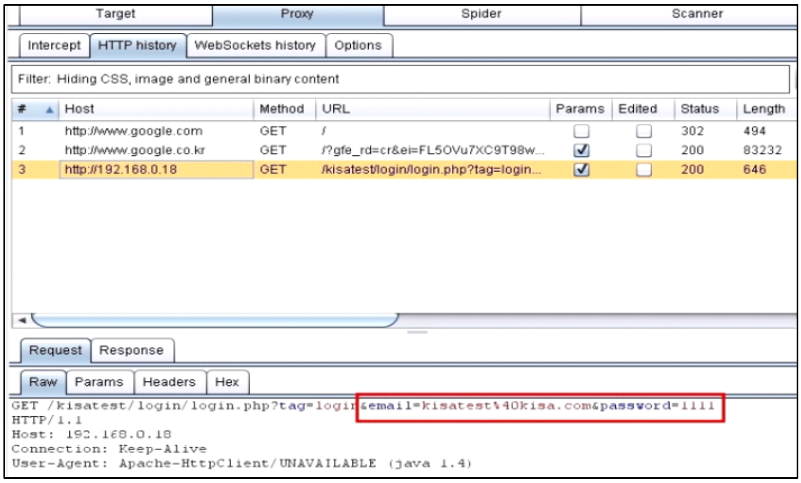
```
c:\adb> adb shell
shell@kisatest:/ $ su
su
root@kisatest:/ # cd /data/data/com.kisa.kisatest
root@kisatest:/ data/data/com.kisa.kisatest # grep -rinoE 'kisapass' *
Binary file databases/kisatest/kisatest matches
Binary file databases/kisatest/kisatest-journal matches
grep: lib: No such file or directory
2:root@kisatest:/data/data/com.kisa.kisatest #
```

〈 grep 명령으로 'kisapass' 문자열 검색 예시 〉



〈검색 결과로 나온 파일을 SQLite 확인 수행〉

③ 프록시 도구(예, BurpSuite 등) 및 패킷분석 도구(예, WireShark 등)를 사용하여 캡처된 패킷 내 인증 정보 등 중요정보가 평문으로 전송되는지 확인한다.



〈 인증 정보 평문 전송 확인 〉

라. 진단기준

- 모바일 기기 내부 파일에서 중요정보를 검색했을 때, 평문으로 중요정보 검색이 가능할 경우 취약한 것으로 판단한다.
- 로그인 등 서버로 전송되는 중요정보가 평문으로 전송된다면 취약한 것으로 판단한다.
- 중요정보 전송 시 HTTPS로 통신할 경우 안전하다고 판단한다.

취약한 예시

```
c:\adb> adb shell
shell@kisatest:/ $ su
su
root@kisatest:/ # cd /data/data/com.kisa.kisatest
root@kisatest:/ data/data/com.kisa.kisatest # grep -rinoE 'kisapass' *
Binary file databases/kisatest/kisatest matches
Binary file databases/kisatest/kisatest-journal matches
grep: lib: No such file or directory
2:root@kisatest:/data/data/com.kisa.kisatest #
```

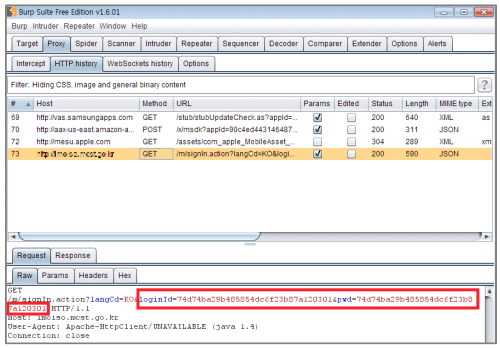
중요정보 평문 저장

중요정보 평문 전송

안전한 예시

```
c:\adb> adb shell
shell@kisatest:/ $ su
su
root@kisatest:/ # cd /data/data/com.kisa.kisatest
root@kisatest:/ data/data/com.kisa.kisatest # grep -rinoE 'kisapass' *
grep: lib: No such file or directory
1:root@kisatest:/data/data/com.kisa.kisatest #
```

중요정보 평문 저장



중요정보 평문 전송

3. 중요정보 저장 및 전송 시 취약한 암호 알고리즘 적용

가. 설명

- 모바일 기기 내 중요정보(인증 정보, 개인 정보, 위치 정보 등) 저장 또는 서버로 전송 시 취약한 암호 알고리즘 사용하면 공격자에게 중요정보가 노출될 수 있다.
  ※ "국가정보원 국가사이버안전센터"의 검증대상 암호 알고리즘 및 "한국인터넷진흥원"의 『암호 알고리즘 및 키 길이 이용 안내서』 참조

나. 보안대책

- 검증된 암호화 알고리즘을 사용하여야 하며, 보안강도 112비트 이상의 키 길이(대칭키 암호 알고리즘[18], 일방향 해시함수[19] 알고리즘은 112비트 이상, 공개키 암호 알고리즘[20]은 2048비트 이상)와 목적에 맞는 알고리즘 선택 및 유효기간을 설정하여 사용한다.
- 비밀번호 저장의 경우 일방향 해시함수에 솔트(salt)를 추가하여 사용한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb, SQLite, BurpSuite, Fiddler, WireShark	putty, BurpSuite, Fiddler, WireShark
루팅 및 탈옥	필요	필요

① 앱을 설치 및 실행한 후 중요정보가 사용되는 로그인, 사용자 위치 정보 사용 등 중요정보 입력 및 사용 기능을 실행한다.

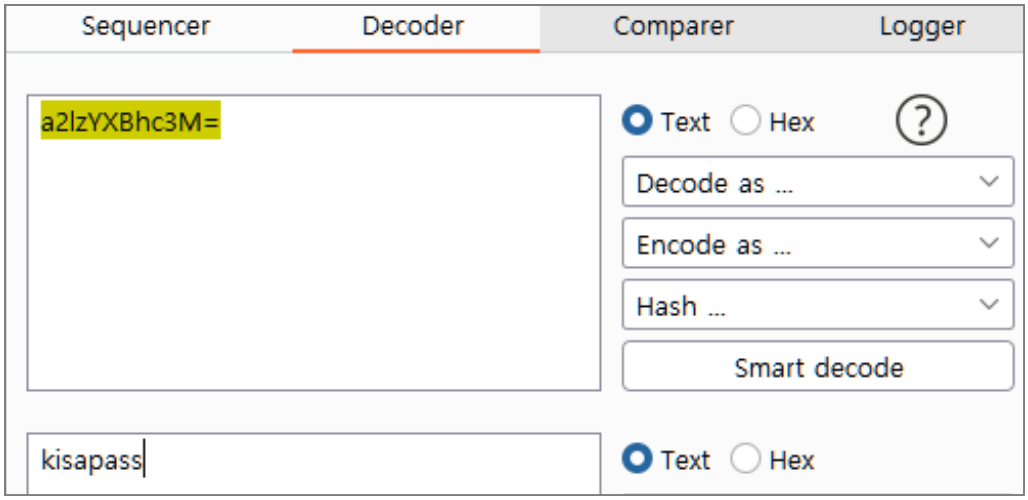
② grep 명령을 사용해 중요정보 저장 여부를 검사하고, 프록시 도구(예, BurpSuite 등) 및 패킷분석 도구(예, WireShark 등)로 캡처된 패킷의 정보를 분석한다.
※ 중요정보의 암호화의 길이가 짧거나 의심되는 경우 확인한다.
※ (명령어) "grep -rinoE '검색어' *"명령어로 중요정보가 저장되었는지 확인

③ 취약한 일방향 해시함수나 단순 인코딩이 되었는지 확인한다.

18) 대칭키 암호 알고리즘 : AES, SEED, ARIA, HIGHT 등 (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018.12)
19) 해시함수 : HAS, SHA, SHA3, LSH 등 (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018.12)
20) 공개키 암호 알고리즘 : RSAES 등 (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018.12))

```
c:\adb> adb shell
shell@kisatest:/ $ su
su
root@kisatest:/ # cd /data/data/com.kisa.kisatest
root@kisatest:/ data/data/com.kisa.kisatest # grep -rinoE 'a2lzYXBhc3M=' *
Binary file databases/kisatest/kisatest matches
grep: lib: No such file or directory
1:root@kisatest:/data/data/com.kisa.kisatest #
```

〈 중요정보 Base64 인코딩 저장 확인 〉



〈 Base64로 디코딩하여 중요정보 여부 확인 〉

라. 진단기준

- 로그인에 사용한 ID, 비밀번호를 검색했을 때 평문으로 검색이 가능할 경우 취약한 것으로 판단한다.
- 소스코드를 분석하여 사용되는 암호화 방식을 확인하고 대칭키 암호 알고리즘, 해시함수를 사용할 경우 112비트 미만의 키 길이를 갖는 약한 암호화를 사용할 경우 취약한 것으로 판단한다.
- 소스코드를 분석히야 사용되는 암호화 방식을 확인하고 공개키 암호 알고리즘의 경우 2048비트 미만의 키 길이를 갖는 약한 암호화를 사용할 경우 취약한 것으로 판단한다.

4. 기타 중요정보의 평문 저장 및 전송

가. 설명

- 기타 중요정보(IMEI[21], IMSI[22] 등)가 모바일 기기에 평문으로 저장되거나 평문으로 전송되는 경우 공격자에게 중요정보가 노출될 수 있다.

나. 보안대책

- 기타 중요정보가 평문으로 저장될 경우 해당 저장 기능에 112비트 이상의 보안강도를 갖는 안전한 암호화 기능을 적용한다.
- 기타 중요정보가 서버로 전송되는 경우 TLS 통신 방법을 이용하는 등 암호화하여 전송한다.
- 모바일 대국민 보안공통기반의 "E2E 암호화" 기능을 활용하여 전송되는 기타 중요정보를 암호화하여 전송한다.
- 앱 실행 시 기타 중요정보에 대한 임시파일 생성을 금지하며, 불가피할 경우 휘발성 메모리에 암호화하여 임시 저장하되 앱 종료 시 즉시 삭제하게 한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb, SQLite, BurpSuite, Fiddler, WireShark	putty, BurpSuite, Fiddler, WireShark
루팅 및 탈옥	필요	필요

① 앱을 실행하여 앱의 모든 기능을 동작시킨다.

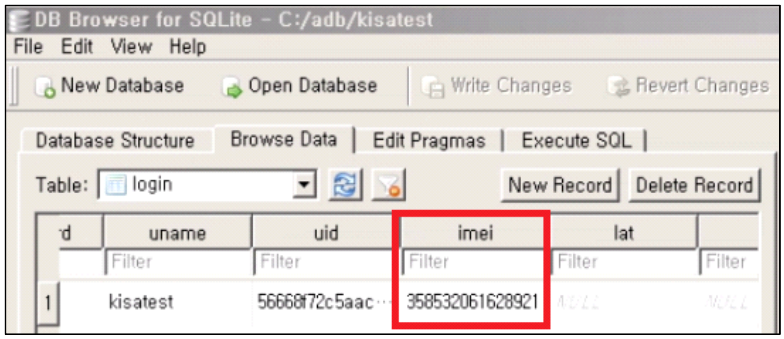
② grep 명령을 사용하여 기타 중요정보 문자열을 검색하여 평문으로 저장되는지 확인한다.

21) IMEI는 모바일 단말기 고유번호(Internaional Mobile Equipment Identity)로서 해당 일련번호는 핸드폰 분실시 해당 휴대폰을 정지시키기 위한 용도 혹은 개통하려 할 때 도난 여부 확인 등에 사용한다.

22) IMSI는 범용개인식별모듈(USIM) 카드의 고유번호(International Mobile Subscriber Identity)로서 공격자가 해당 식별번호를 탈취할 경우, 탈취한 IMSI를 자신의 IMSI로 위장하여 인증할 경우 공격자는 피해자의 모바일 기기에서 전송되는 전화 및 데이터를 모두 감청할 수 있게 된다.

```
C:₩adb>adb shell
shell@zerofIteskt:/ $ su
root@zerofIteskt:/ # cd /data/data/com.kisa.kisatest
rep -rionE '358532061628921' *
Binary file databases/kisatest matches
Binary file databases/kisatest-journal matches
grep: lib: No such file or directory
2|root@zerofIteskt:/data/data/com.kisa.kisatest # cd databases
```

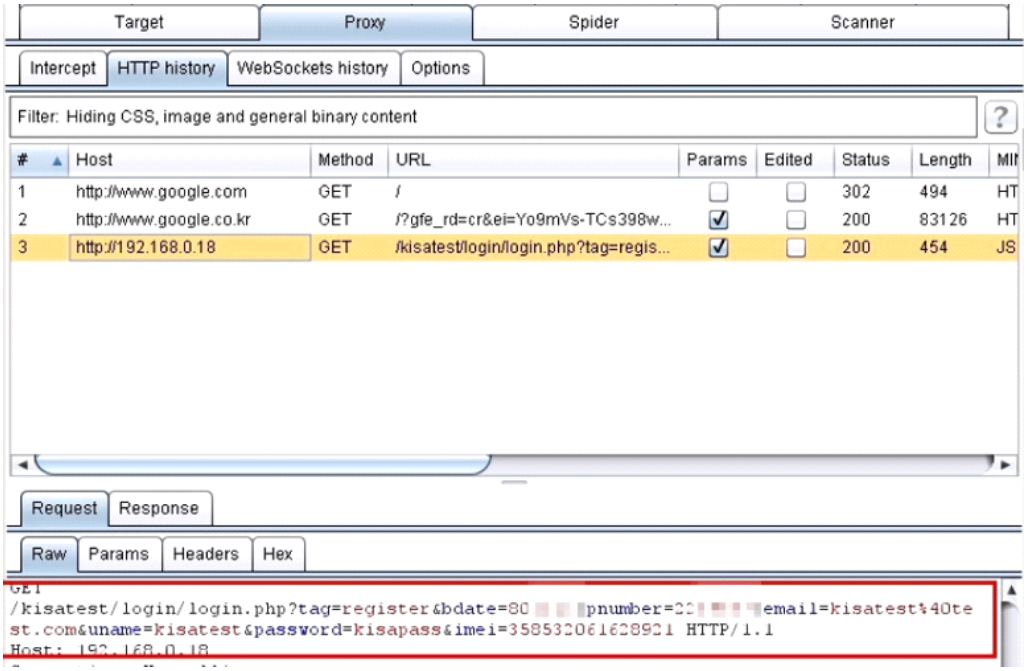
〈 grep 명령으로 IMEI 문자열 검색 〉



〈 IMEI 평문 저장 〉

③ 프록시 도구(예, BurpSuite) 및 패킷분석 도구(예, WireShark)로 캡처된 기타 중요정보 관련 패킷을 분석한다.

※ HTTPS 프로토콜을 사용 시 프록시 도구인 BurpSuite를 사용하여 확인이 어려울 수도 있으므로 패킷분석 도구인 WireShark를 사용하여 네트워크 트래픽으로 확인할 수 있음



〈 IMEI 평문 전송 〉

라. 진단기준

- 기타 중요정보(IMEI, IMSI)가 평문으로 검색이 가능할 경우 취약한 것으로 판단한다.
- 기타 중요정보(IMEI, IMSI)가 평문으로 전송이 가능할 경우 취약한 것으로 판단한다.
- HTTPS 프로토콜을 사용하여 기타 중요정보를 전송하는 경우에는 안전한 것으로 판단한다.

5. 기타 중요 정보 저장 및 전송 시 취약한 암호 알고리즘 적용

가. 설명

- 모바일 기기에 IMEI, IMSI 정보 등 기타 중요정보를 저장 및 전송 시 취약한 암호알고리즘 사용하면 공격자에게 기타 중요정보가 유출될 수 있다.

  ※ "국가정보원 국가사이버안전센터"의 검증대상 암호알고리즘 및 "한국인터넷진흥원"의 『암호 알고리즘 및 키 길이 이용 안내서』 참조

나. 보안대책

- 검증된 암호화 알고리즘을 사용하여야 하며, 보안강도 112비트 이상의 키 길이(대칭키 암호 알고리즘[23], 일방향 해시함수[24] 알고리즘은 112비트 이상, 공개키 암호 알고리즘[25]은 2048비트 이상)와 목적에 맞는 알고리즘 선택 및 유효기간을 설정하여 사용한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	adb, SQLite, BurpSuite, Fiddler, WireShark	putty, BurpSuite, Fiddler, WireShark
루팅 및 탈옥	○	○

① 앱을 설치 및 실행한 후 기타 중요정보가 사용되는 로그인, 사용자 위치정보 사용 등 기타 중요정보 입력 및 사용 기능을 실행한다.

② grep 명령을 사용해 기타 중요정보 저장 여부를 검사하고, 프록시 도구(예, BurpSuite 등) 및 패킷분석 도구(예, WireShark 등)로 캡처된 패킷의 정보를 분석한다.

※ 기타 중요정보의 암호화의 길이가 짧거나 의심되는 경우 확인한다.

※ (명령어) "grep -rinoE '검색어' *"명령어로 기타 중요정보가 저장되었는지 확인

③ 취약한 일방향 해시함수나 단순 인코딩이 되었는지 확인한다.

---

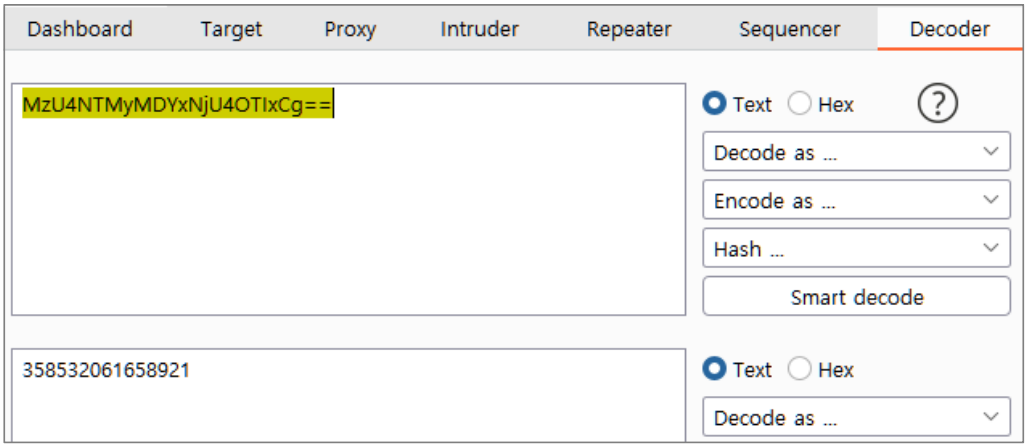
23) 대칭키 암호 알고리즘 : AES, SEED, ARIA, HIGHT 등 (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018.12)

24) 해시함수 : HAS, SHA, SHA3, LSH 등 (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018.12)

25) 공개키 암호 알고리즘 : RSAES 등 (KISA, 암호 알고리즘 및 키 길이 이용 안내서, 2018.12))

```
root@kisatest:/ # cd /data/data/com.kisa.kisatest
root@kisatest:/ data/data/com.kisa.kisatest # grep -rinoE 'MzU4NTMyMDYxNjU4OTIxCg==' *
Binary file databases/kisatest/kisatest matches
grep: lib: No such file or directory
1:root@kisatest:/data/data/com.kisa.kisatest #
```

〈 IMEI 단순 인코딩 확인 〉



〈 IMEI 단순 인코딩 확인 〉

④ 기타 중요정보(IMEI, IMSI) 저장 및 전송 시 취약한 암호화 알고리즘 사용 여부를 확인한다.

라. 진단기준

- 기타 중요정보(IMEI, IMSI) 검색했을 때 평문으로 검색이 가능할 경우 취약한 것으로 판단한다.
- 소스코드를 분석하여 사용되는 암호화 방식을 확인하고 대칭키 암호 알고리즘, 해시함수를 사용할 경우 112비트 미만의 키 길이를 갖는 약한 암호화를 사용할 경우 취약한 것으로 판단한다.
- 소스코드를 분석하여 사용되는 암호화 방식을 확인하고 공개키 암호 알고리즘의 경우 2048비트 미만의 키 길이를 갖는 약한 암호화를 사용할 경우 취약한 것으로 판단한다.

6. 파일 다운로드 시 외부주소 변조 및 파일 무결성 우회

가. 설명

- 파일 또는 업데이트 다운로드 시 외부주소 변조 및 파일 무결성 검사 등이 적절히 수행되지 않으면 공격자가 파일 다운로드 기능을 이용하여 시스템 중요 파일 및 정보를 탈취할 수 있다.

나. 보안대책

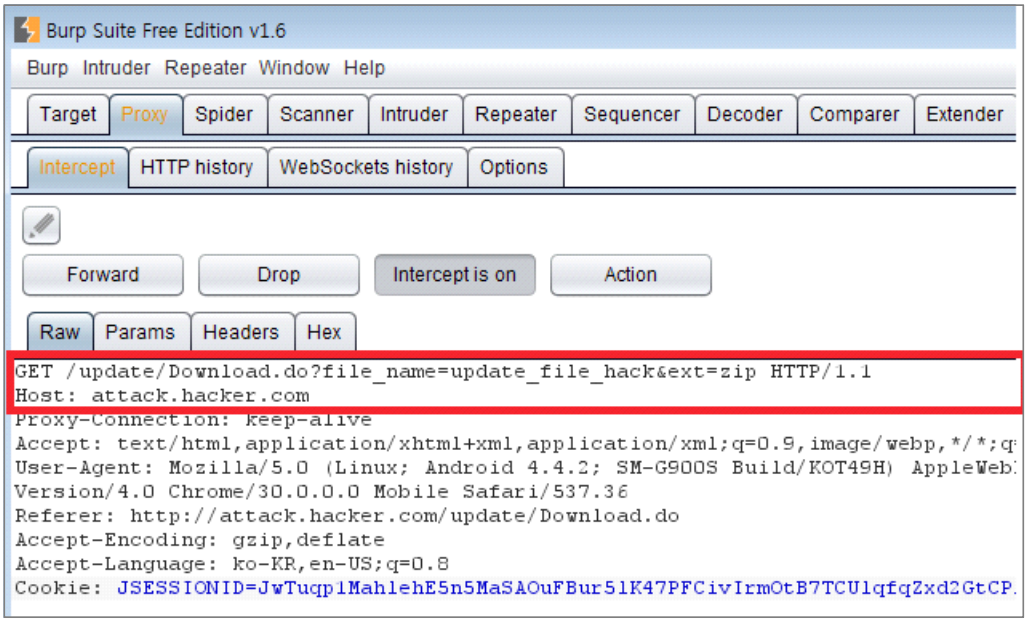
- 설정 파일 또는 업데이트는 다운로드 시 서버주소가 외부로 공개되거나, 외부로 공개된 서버의 파일이 무결성 검사 없이 다운로드 되지 않도록 한다.
  ※ 동적으로 추가되는 jar 파일, 펌웨어 파일, 추가적인 apk 파일, 앱이 사용하려는 리소스나 설정 파일(config, message text 등)에 대해 확인한다.
- 외부주소 또는 파일에 비정상적인 변경 사항이 생길 시에는 오류 알람 또는 로그 등으로 사용자에게 알리고, 파일의 무결성을 검사하여 앱이 동작하지 않도록 한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	**iOS 플랫폼**
점검 도구	adb, SQLite, BurpSuite, Fiddler, WireShark	putty, BurpSuite, Fiddler, WireShark
루팅 및 탈옥	필요	필요

① 앱에 파일 다운로드 기능이 구현되어있을 경우, 앱 설정 파일 또는 실행 파일에 포함된 서버 주소를 확인한다.

② 프록시 도구(BurpSuite 등)를 사용하여 기본 설정된 서버주소 변경 및 삭제를 수행하며, 이때 무결성 오류 점검을 수행하는지를 확인한다.
※ 서버주소 변경 및 삭제 시 무결성 오류 발생 알람 또는 로그 등을 확인



〈 다운로드 서버주소 변경 예시 〉

③ 다운로드 되는 파일에 대한 무결성 검사를 하는지 확인한다.

라. 진단기준

- 앱 설정 파일 또는 실행 파일에 포함된 서버주소에 허가되지 않은 다른 주소가 포함되어 있으면 취약한 것으로 판단한다.
- 기본 설정된 서버주소 변경 및 삭제 시 무결성 오류 발생 알람 또는 로그를 확인할 수 있으면 안전한 것으로 판단한다.
- 기본 설정된 서버주소 변경 및 삭제 시 앱의 기능 등이 정상적으로 동작할 경우 취약한 것으로 판단한다.

제 5 절 수집 · 활용 및 배포

"수집 · 활용 및 배포" 보안취약점은 사용자 동의 없이 임의로 개인 정보 및 개인 위치 정보를 수집하고 활용하여 법적 문제가 발생하거나, 역공학 기술에 의한 모바일 앱 소스 코드 유출 및 보안 메커니즘 우회 등이 발생할 수 있는 취약점을 의미한다.

세부 보안취약점

1	개인정보 및 개인위치정보 수집 및 활용에 대한 동의	2	난독화

1. 개인정보 및 개인위치정보 수집 및 활용에 대한 동의

가. 설명

- 개인정보(성명, 주민등록번호 등) 및 개인위치정보(GPS 정보)를 무분별하게 수집하고 활용하면 개인정보 오 · 남용으로 인한 정보주체의 사생활 침해 등이 발생할 수 있다.

나. 보안대책

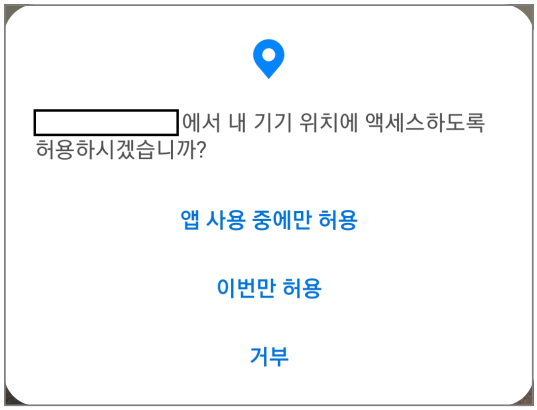
- 개인정보 및 개인위치정보 수집 · 활용에 대한 안내 및 동의를 받아야 한다.
- 개인정보 및 개인위치정보 수집에 관한 동의를 받는 경우, 정보주체가 동의 여부를 실질적으로 선택할 수 있도록 수집목적과 항목 등을 구체적이고 명확하게 안내해야 한다.
- 앱에서 사용하는 권한에 대해 서비스 제공을 위해 반드시 필요한 필수적 접근 권한과 선택적 접근 권한을 구분하여 접근 권한이 필요한 항목 및 그 이유 등을 이용자에게 알기 쉽게 알려야 한다.
  ※ 개인정보 수집과 동의절차는 행정안전부의 『개인정보 수집 최소화 가이드라인』 참조
- 앱의 접근 권한에 대해서도 동의를 받을 수 있는 기능을 제공하고, 이용자가 접근 권한에 대한 사후적 철회가 가능하도록 해야 한다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	-	-
루팅 및 탈옥	-	-

① 앱을 설치 및 실행한 후, 앱이 개인정보 및 개인위치정보를 사용하는 기능이 있는지 확인한다.

② 개인정보 및 개인위치정보를 사용하는 기능이 존재할 경우, 사용자에게 개인정보 및 개인위치정보 사용에 대한 동의를 구하는지 확인한다.

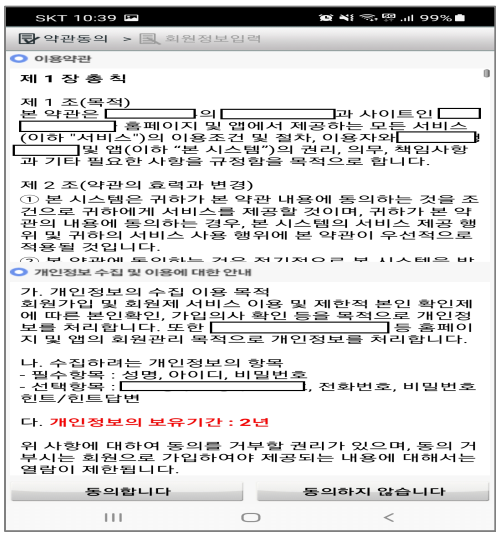


〈 개인위치정보 사용에 대한 동의 요청(예시) 〉

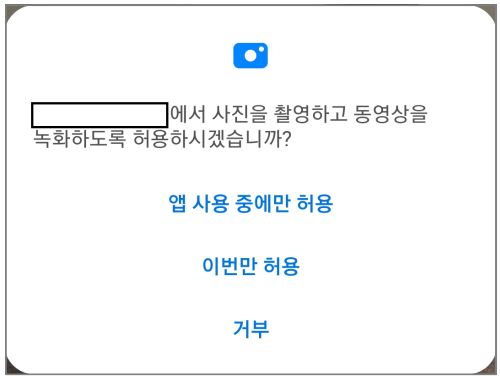
라. 진단기준

- 사용자에게 사전 동의를 구하지 않고 개인정보 및 개인위치정보를 사용하는 기능이 동작하면 취약한 것으로 판단한다.

안전한 예시



〈 개인정보 수집 동의 〉



〈 접근권한 수집 동의 〉

참 고 접근권한 동의 및 고지

사용자 동의 관련 참고사항

- 개인정보(성명, 주민등록번호 등) 및 개인위치정보(GPS) 등 사생활(Privacy)와 관련된 정보를 수집하고 활용하는 권한을 설정할 때에는 사전에 앱 사용자에게 허용 여부를 알리고 [26]동의를 얻어야 함
- 사용자 스마트폰에 대한 접근 권한(위치, 카메라 등)이 필요한 경우, 앱 서비스의 본질적 기능 수행에 반드시 필수적인 권한 항목과 그렇지 않은 선택적 권한 항목을 구분*하고, 접근권한 별 세부 항목과 이유를 앱 사용자가 명확히 인지하도록 알리고** 사용자로부터 동의를 받아야 함

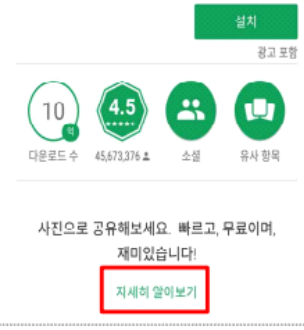
*필수 권한 항목과 선택 권한 항목 구분(예시)	
필수 권한 항목	**선택 권한 항목**
앱 서비스의 본질적 기능을 수행하기 위해 반드시 필요한 권한 항목	앱 개발자 또는 사업자의 필요에 의해 수집하는 권한 항목

**접근 권한 별 목적 안내(예시)	
접근권한	**권한이 필요한 이유(예시)**
위 치	SNS 위치 공유, 주변 맛집 찾기
카메라	SNS 서비스 사진 업로드, 사진 전송, 사진촬영 즉시 업로드
연락처	SNS 친구추가, 연락처 공유 기능

< 앱 설치 시 접근권한 고지에 대한 예시 >

1단계

[안드로이드폰]



[아이폰]

2단계

[필수적 접근권한]

항목	이유
저장 권한	음반 이미지, 곡 재생파일 임시저장
전화 권한	재생 중 전화 상태 체크, 휴대폰 번호로 로그인
SMS 권한	수신한 SMS 인증번호 자동 입력

[선택적 접근권한]

항목	이유
위치 권한	위치기반 상품 추천 서비스 제공

※ 선택적 접근권한의 허용에는 동의하지 않아도 서비스의 이용이 가능합니다.

26) 동의를 얻는 방법은 가이드의 “제5절 수집·활용 및 배포” - “1. 개인정보 및 개인위치정보 수집 및 활용에 대한 동의“ 참고

2. 난독화(Android)

가. 설명

- 안드로이드 플랫폼 기반의 모바일 앱의 경우, 디컴파일 도구[27](예, apktool) 이용하면 실행 파일(.apk)을 소스코드로 쉽게 변환시킬 수 있어 앱 구조 및 소스코드를 쉽게 파악할 수 있고, 소스코드가 원본 수준으로 노출되므로 공격자가 앱 위·변조에 활용할 수 있다.

나. 보안대책

- 난독화 기능이 우수한 상용도구를 이용하는 것을 권고하나, 그렇게 하지 못할 땐 최소한 구글에서 제공하는 오픈소스 난독화도구(ProGuard[28]) 등을 활용하여 난독화를 적용하여야 한다.

  ※ 다만, 오픈소스 난독화 도구의 경우는 상용도구 수준의 기능 및 보안성을 담보하지는 않는다.

다. 진단방법

모바일 플랫폼 별 진단 환경		
	Android 플랫폼	iOS 플랫폼
점검 도구	apktool, dex2jar, JD-GUI	-
루팅 및 탈옥	-	-

① apk 파일을 압축 해제한다.

※ 압축 해제를 위해 파일명.apk 에서 파일명.zip 으로 확장자를 변경 후 압축을 해제

② 압축 해제한 apk 파일에서 classes.dex 파일을 추출한다.

③ 디컴파일 도구인 dex2jar를 이용해 classes.dex파일을 디컴파일 수행한다.

※ dex2jar : apk파일에서 추출한 classes.dex 파일을 jar 파일로 변환시킬 수 있는 디컴파일 도구

※ d2j-dex2jar classes.dex 디컴파일 명령어를 실행하여 JAR 파일인 class_dex2jar.jar 파일 추출

27) 디컴파일 도구를 이용하여 디컴파일 후 smali 코드를 변조하여 보안 로직을 우회하거나 악의적인 행위 등을 할 수 있다

28) 구글 난독화 도구(ProGuard) : https://www.guardsquare.com/proguard

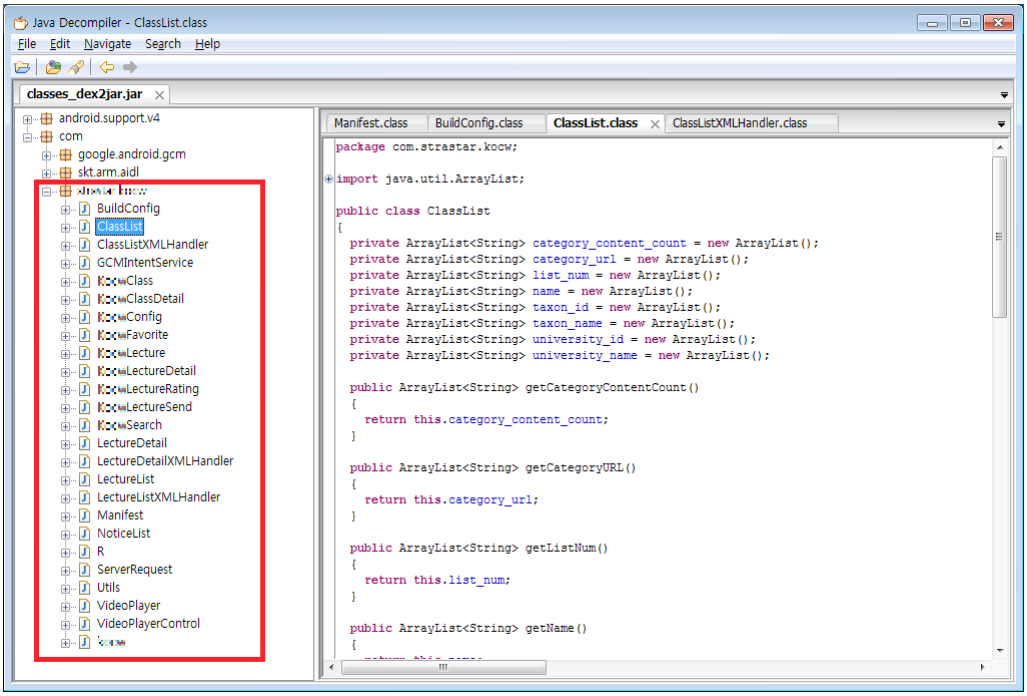
```
D:\app\dex2jar> d2j-dex2jar kisatest.apk
dex2jar kisatest.apk -> .\kisatest-dex2jar.jar

D:\app/dex2jar>
```

〈dex2jar 실행〉

④ class_dex2jar.jar 파일을 디컴파일 확인 도구인 JD-GUI로 코드를 확인한다.

※ JD-GUI : 추출된 JAR파일로부터 자바 소스코드 확인을 할 수 있는 자바 디컴파일 도구

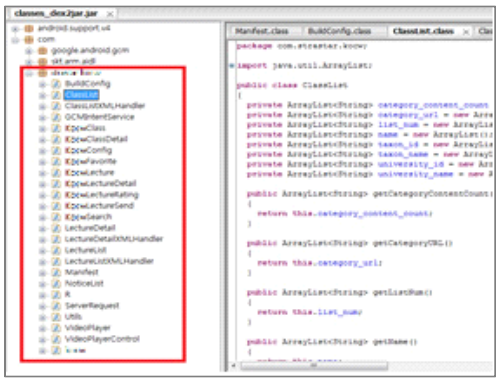


〈 "JD-GUI"을 이용한 난독화 여부 진단 〉

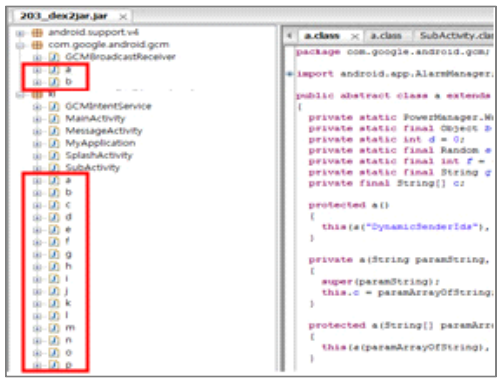
라. 진단기준

- 디컴파일 도구를 이용하여 앱의 클래스명, 필드명, 메소드명 등의 식별자 정보가 원래의 소스코드를 유추할 수 있는 경우 취약한 것으로 판단한다.

취약한 예시



안전한 예시



참 고 난독화 도구

오픈소스 난독화도구(ProGuard) 적용 방법

1. Android Studio 내 main module 내에 있는 build.gradle에 설정 값 추가
2. build.gradle파일 내 minifyEnabled를 true로 설정

```
buildTypes {
        debug {
            minifyEnabled true
        }
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
    }
```

참 고 난독화 방식

1. 배치 난독화 : 식별자 이름, 소스 코드 형식을 바꾸거나 주석을 제거하는 등의 기법을 사용하여 난독화 하는 기법으로, 실제 소스 코드가 컴파일되어 바이너리 파일로 변환될 경우 코드보호가 불가능한 난독화 기법이다.
2. 데이터 난독화 : 프로그램 내부의 자료 구조를 바꾸거나 자료를 암호화하는 방식으로 메모리 및 네트워크 자원 감시로 인한 정보 누출을 막는 난독화 기법으로 배열 분할, 배열 병합, 인코딩 변경, 변수 승격 기법 등이 있다.
3. 제어 난독화 : 원본 프로그램의 제어 흐름을 변경하여 프로그램의 역난독화나 역공학에 어려움을 주기 위한 방법으로 불분명 술어, 루프 조건 변경, 루프 조건 확장, 루프 펴기, 제어 흐름 평탄화, 재배치, 부가 오퍼랜드 삽입, 인라인/아웃라인 기법 등이 있다.
4. 방지 난독화 : 디스어셈블러나 디버거가 정상적으로 작동하기 힘들게 변환하는 방법을 말하며, 정적 디스어셈블리 방지 난독화나, 디버거 실행 감지 시 프로그램의 종료 등의 방법이 있다.

```
public class LOCKIN
{
int bt;
int cmd;
}
```

<소스코드 원본>

```
public class c
{
int a;
int b;
}
```

<분석 가능 난독화>

<분석 불가능 난독화>

참 고 앱 위 · 변조 관련 참고사항

앱 위 · 변조 관련 참고사항

Android 앱은 이식성이 높은 자바 언어로 구현되어, 역공학이 쉬워 앱 위변조를 이용한 악성코드 삽입과 리패키징이 쉽다. 악성코드를 삽입한 리패키징된 앱은 중요정보 탈취 및 여러 악성행위에 이용된다.

앱 위 · 변조 진단 방법

(진단 기준) 리패키징을 이용하여 앱의 이름, 아이콘, 내부 소스코드 등을 변경한 후에 앱이 실행되거나 재배포가 가능하면 취약한 것으로 판단

앱 위변조 여부 진단 방법

1. apktool를 이용해 apk 파일 디컴파일 수행
   ※ 명령어 : apktool d 분석대상앱.apk

2. 디컴파일 된 파일 변조 수행(앱 이름, 실행아이콘 등)
   ※ (예) /res/Values 폴더에 존재하는 strings.xml 파일 내의 app_name 부분의 앱 이름 수정
   ※ (예) /res/drawable 폴더에 존재하는 실행아이콘 이미지를 ic_launcher.png를 다른 이미지로 변경
3. apktool을 이용해 리패키징을 수행
   ※ 명령어 : apktool b 분석대상앱 폴더
4. 서명 프로그램(keytool, jar signer 등)을 이용해 빌드된 apk 파일에 대하여 서명(코드사인)
   ※ keytool 키 생성 프로그램, jarsigner 서명 프로그램
5. 서명된 apk 파일을 모바일 기기에 설치하여 정상적으로 실행되는지를 확인
   ※ 서명된 apk 파일을 adb 툴을 이용하여 설치하여 동작 여부 확인

앱 위변조 방지솔루션 도입 시 체크사항

- 역 분석하여 프로그램 변조 가능여부
- APP 관련 데이터(String 등)에 대한 암호화 여부
- 무결성 검증 로직에 대한 변조 가능여부
- OS변조 탐지 가능여부
- 위변조 APP 발생시 모니터링 가능여부

[Part 3]

소스코드 보안약점 진단 방법

Part 3

소스코드 보안약점 진단 방법

모바일 앱 소스코드 보안약점 기준은 『모바일 전자정부 서비스 관리 지침』의 모바일 앱 보안약점 점검기준(별표 3) 및 모바일 시큐어코딩 가이드[29]와 소프트웨어 보안약점 진단 가이드[30]를 참고할 수 있다.

모바일 앱 소스코드 보안약점 기준		
번호	**점검 항목**	**설명**
1	SQL 삽입	검증되지 않은 외부 입력값이 SQL 쿼리문 생성에 사용되어 악의적인 쿼리가 실행될 수 있는 보안약점
2	경로 조작 및 자원 삽입	검증되지 않은 외부 입력값이 시스템 자원 접근경로 또는 자원제어에 사용되어 공격자가 입력값을 조작해 공격할 수 있는 보안약점
3	크로스사이트 스크립트	검증되지 않은 외부 입력값에 의해 사용자 브라우저에서 악의적인 스크립트가 실행될 수 있는 보안약점
4	운영체제 명령어 삽입	검증되지 않은 외부 입력값이 운영체제 명령문 생성에 사용되어 악의적인 명령어가 실행될 수 있는 보안약점
5	오버플로우(정수형, 메모리 버퍼)	정수값 및 메모리 버퍼의 경계값이 범위를 넘어서는 경우, 프로그램이 예기치 않게 동작될 수 있는 보안약점
6	취약한 암호화 알고리즘 사용	중요정보(금융정보, 개인정보, 인증정보 등)를 열람(또는 변경) 할 수 있게 하는 보안약점
7	중요정보 평문 저장	중요정보(비밀번호, 개인정보 등)를 암호화하여 저장하지 않아 정보가 노출될 수 있는 보안약점

29) 한국인터넷진흥원, “Android-JAVA 시큐어 코딩 가이드”, 한국인터넷진흥원 홈페이지(https://www.kisa.or.kr)의 자료실 〉 기술안내서가이드

30) 행정안전부, “소프트웨어 보안약점 진단가이드”, 행자부 홈페이지(https://www.mois.go.kr/)의 정책자료 〉 참고자료 및 간행물

모바일 앱 소스코드 보안약점 기준		
번호	**점검 항목**	**설명**
8	중요정보 평문 전송	중요정보(비밀번호, 개인정보 등) 전송시 암호화하지 않거나 안전한 통신채널을 이용하지 않아 정보가 노출될 수 있는 보안약점
9	하드코드된 비밀번호	소스코드 내에 비밀번호가 하드코딩 되어 소스코드 유출시 노출 우려 및 키 변경이 용이하지 않는 보안약점
10	충분하지 않은 키 길이 사용	데이터의 기밀성, 무결성 보장을 위해 사용되는 키의 길이가 충분하지 않아 기밀정보 누출, 무결성이 깨지는 보안약점
11	적절하지 않은 난수값 사용	예측 가능한 난수사용으로 공격자로 하여금 다음 숫자 등을 예상하여 시스템 공격이 가능한 보안약점
12	하드코드된 암호화 키	소스코드 내에 암호화키가 하드코딩 되어 소스코드 유출시 노출 우려 및 키 변경이 용이하지 않는 보안약점
13	주석문안에 포함된 시스템 주요정보	소스코드내의 주석문에 인증정보 등 시스템 주요정보가 포함되어 소스코드 유출시 노출될 수 있는 보안약점
14	경쟁조건: 검사시점과 사용시점(TOCTOU)	멀티 프로세스 상에서 자원을 검사하는 시점과 사용하는 시점이 달라서 발생하는 보안약점
15	오류메시지 및 시스템 데이터 정보노출	사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되는 보안약점
16	오류상황 대응 부재	시스템에서 발생하는 오류 상황을 처리하지 않아 프로그램 실행정지 등 의도하지 않은 상황이 발생할 수 있는 보안약점
17	부적절한 예외 처리	예외에 대한 부적절한 처리로 인해 의도하지 않은 상황이 발생될 수 있는 보안약점
18	Null Pointer 역참조	Null로 설정된 변수의 주소값을 참조했을 때 발생하는 보안약점
19	부적절한 자원 해제	사용된 자원을 적절히 해제하지 않으면 자원 누수 등이 발생하고, 자원이 부족하여 새로운 입력을 처리할 수 없게 되는 보안약점
20	취약한 API	사용 취약하다고 알려진 함수를 사용함으로써 예기치 않는 보안위협에 노출될 수 있는 보안약점
21	안드로이드 애플리케이션 컴포넌트의 부적절한 접근 허용	안드로이드 애플리케이션 컴포넌트 설정으로 부적절한 접근이 허용되어 외부의 애플리케이션에 의해 의도치 않게 실행될 수 있는 보안약점

모바일 앱 소스코드 보안약점 기준		
번호	점검 항목	설명
22	민감한 정보 전송을 위한 암시적 intent 사용	암시적 intent를 사용하여 민감한 정보가 전송시 도청 및 악성행위 삽입이 가능한 보안약점
23	접근제어 없이 내 · 외부저장소 사용	내 · 외부저장소(SD카드 등)에 중요한 정보를 암호화하여 저장하지 않아 정보가 노출되거나 수정이 가능한 보안약점
24	안드로이드의 권한 검사 우회	권한이 전혀 없는 호출 프로그램이 응용 프로그램의 권한을 사용하게 되어 권한 검사를 우회할 수 있는 보안약점
25	클래스 로딩 하이재킹	프로그램의 클래스 로드시 검색되는 디렉토리의 이름이 변경되어 클래스 경로를 공격자가 명시적으로 제어할 수 있는 보안약점
26	소스코드 난독화 미적용	역공학 기술에 의한 소스코드 유출 및 보안메커니즘 우회 등이 발생할 수 있는 보안약점

본 장에서는 모바일 앱 소스코드에서 발생할 수 있는 주요 보안약점 중심의 진단방법을 제시하여 소스코드 보안약점 진단방법을 설명한다.

제 1 절 입력데이터 검증 및 표현

1. 개요

프로그램 입력값에 대한 검증 누락 또는 부적절한 검증, 데이터의 잘못된 형식지정으로 인해 발생할 수 있는 보안약점을 제거하여야 한다.

2. 세부 보안약점

입력데이터 검증 및 표현은 “SQL 삽입”, “크로스사이트 스크립트” 등 5개의 보안약점으로 소스코드 보안약점 진단방법은 소프트웨어 보안약점 진단가이드를 참조한다.

세부 보안 약점

1	SQL 삽입	4	운영체제 명령어 삽입
2	경로 조작 및 자원 삽입	5	오버플로우(정수형, 메모리 버퍼)
3	크로스사이트 스크립트		

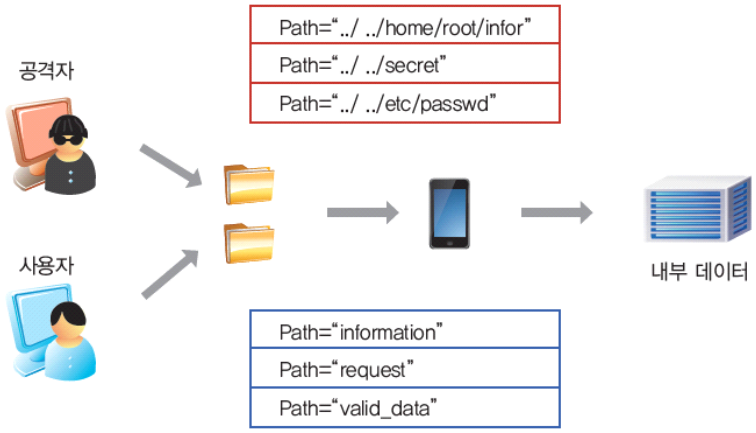
3. 주요 진단사례

입력데이터 검증 및 표현에서 많이 발견되는 소스코드 보안약점은 “경로 조작 및 자원 삽입”, “오버플로우(정수형, 메모리 버퍼)”로 진단 방법은 다음과 같다.

가. 경로조작 및 자원삽입

1) 개요

외부 입력값이 검증없이 디렉토리 경로 문자열 생성이나 시스템 자원(Resource) 접근에 사용되는 경우 발생할 수 있는 보안약점이다. 공격자는 외부 입력값을 조작하여 시스템이 보호하는 자원에 임의로 접근하거나 수정할 수 있으며, 잘못된 입력값으로 인해 시스템 자원간 충돌이 발생할 수 있다. 또한 의도하지 않은 접근 제한 영역에 문자열 구성이 가능해질 수가 있다.



〈 경로를 조작하여 권한 없는 정보에 접근 〉

2) 보안대책

- 외부의 입력을 자원(파일, 소켓의 포트 등)의 식별자로 사용하는 경우, 적절한 검증을 거치도록 하거나, 사전에 적합한 리스트에서 선택되도록 한다.
- 외부의 입력이 파일명인 경우에는 경로순회(Directory Traversal) 공격의 위험이 있는 문자(",/,\,&,.. 등)을 제거하는 필터를 이용한다.

안전한 코드의 예

```
1: . . .
2: public void onCreate(Bundle savedInstanceState) {
3:     super.onCreate(savedInstanceState);
4:     setContentView(R.layout.main);
5:     EditText input = (EditText)findViewById(R.id.Text);
6:     String name = input.getText().toString();
7:     String dir = "/usr/local/tmp";
8:     if ( name != null && !"".equals(name) ) {
9:         name = name.replaceAll("/", "");
10:        name = name.replaceAll("\\\\", "");
11:        name = name.replaceAll("\\.", "");
12:        name = name.replaceAll("&", "");
13:        name = name + "-report";
14:    }
15:    File file = new File(dir + name);
16:    if (file != null) file.delete();
17: . . .
18: }
```

3) 진단방법

- 외부의 입력(①)이 삭제할 파일의 경로설정에 사용(②)되는지 확인한다.
- 외부 입력값에 대한 필터링 또는 검증절차가 있는지 확인한다.
  - 공격자에 의해 name의 값으로 ../../../rootFile.txt와 같은 값을 전달하면, 의도하지 않았던 파일이 삭제되어 시스템에 악영향을 끼칠 수 있다.(③)

안전하지 않은 코드의 예 Android-JAVA

```
1: . . .
2: public void onCreate(Bundle savedInstanceState) {
3:     super.onCreate(savedInstanceState);
4:     setContentView(R.layout.main);
5:     EditText input = (EditText)findViewById(R.id.Text);
6:     String name = input.getText().toString(); ------------- ①
7:     File file = new File("usr/local/tmp/" + name); ------------- ②
8:     file.delete(); ------------- ③
9: . . .
10: }
```

4) 진단기준

- 외부의 입력값에 대한 필터링 또는 검증절차가 없다면 취약한 것으로 판단한다.
- 파일명, 소켓의 포트 등 자원을 사용하는지 확인하고, 해당 자원을 외부에서 직접 접근이 가능하다면 취약한 것으로 판단한다.

제 2 절 보안기능

1. 개요

보안기능(인증, 접근제어, 기밀성, 암호화, 권한 관리 등)을 부적절하게 구현 시 발생할 수 있는 보안약점을 제거하여야 한다.

2. 세부 보안약점

보안기능은 “취약한 암호화 알고리즘 사용”, “하드코드된 비밀번호” 등 8개의 보안약점으로 소스코드 보안약점 진단방법은 소프트웨어 보안약점 진단가이드를 참조한다.

세부 보안 약점

1	취약한 암호화 알고리즘 사용	5	충분하지 않은 키 길이 사용
2	중요정보 평문 저장	6	적절하지 않은 난수값 사용
3	중요정보 평문 전송	7	하드코드된 암호화 키
4	하드코드된 비밀번호	8	주석문안에 포함된 시스템 주요정보

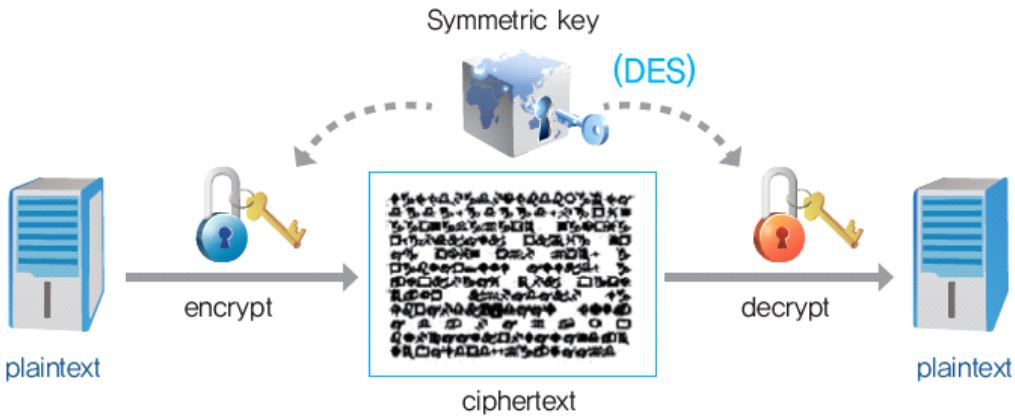
3. 주요 진단사례

보안기능에서 많이 발견되는 소스코드 보안약점은 “취약한 암호화 알고리즘 사용”, “적절하지 않은 난수값 사용”으로 진단 방법은 다음과 같다.

가. 취약한 암호화 알고리즘 사용

1) 개요

보안적으로 취약하거나 위험한 암호화 알고리즘을 사용해서는 안된다. 표준 암호화 알고리즘을 사용하지 않는 경우 공격자가 알고리즘을 분석하여 무력화시킬 수 있는 가능성을 높일 수도 있다. 몇몇 오래된 암호화 알고리즘의 경우는 컴퓨터의 성능이 향상됨에 따라 취약해지기도 해서, 예전에는 해독하는데 몇 년이 걸리던 알고리즘이 며칠이나 몇 시간 내에 해독되기도 한다. RC2, RC4, RC5, RC6, MD4, MD5, SHA1, DES, 2DES 알고리즘이 여기에 해당된다.



〈 취약한 암호화 알고리즘 사용 〉

2) 보안대책

자신만의 암호화 알고리즘을 개발하는 것은 위험하며, 학계 및 업계에서 이미 검증된 표준화된 알고리즘을 사용한다. 보안에 취약한 DES, RC5 등의 암호 알고리즘을 대신하여, AES, SEED 등의 안전한 암호알고리즘을 사용한다.

안전한 코드의 예 Android-JAVA

```
1. // 안전한 AES 알고리즘을 최소 128바이트 길이 키로 이용
2. ……
3.     public byte[] encrypt(byte[] msg, Key k) {
4.         byte[] rslt = null;
5.
6.         try {
7.         // 낮은 보안수준의 DES 알고리즘을 높은 보안수준의 AES 알고리즘으로 대체한다.
8.             Cipher c = Cipher.getInstance("AES/CBC/PKCS5Padding");
9.             c.init(Cipher.ENCRYPT_MODE, k);
10.            rslt = c.update(msg);
11.        } catch (InvalidKeyException e) {
12.            ……
13.        }
14.        return rslt;
15.    }
16. }
```

3) 진단방법

- 보안에 취약한 암호 알고리즘(DES 등)을 사용하는지 확인한다.(①)

※ 자체 구현한 암호화 관련 알고리즘을 사용할 경우에는 취약한 것으로 판단한다. 만약, 암호전문가가 존재할 경우 해당 암호함수 알고리즘의 적절성을 판단하여 보증하는 것은 가능하다.

안전하지 않은 코드의 예 Android-JAVA

```
1. // 안전하지 않은 DES 암호화 알고리즘 사용
2. ……
3.     public byte[] encrypt(byte[] msg, Key k) {
4.         byte[] rslt = null;
5.
6.         try {
7.         // DES등의 낮은 보안수준의 알고리즘을 사용하는 것은 안전하지 않다.
8.             Cipher c = Cipher.getInstance("DES"); ------------- ①
9.             c.init(Cipher.ENCRYPT_MODE, k);
10.            rslt = c.update(msg);
11.        } catch (InvalidKeyException e) {
12.        ……
13.        }
14.        return rslt;
15.        }
16.    }
```

4) 진단기준

- 취약한 암호 알고리즘을 사용하거나 자체 구현한 암호화 관련 알고리즘을 사용할 경우에는 취약한 것으로 판단한다.
- Base64 인코딩의 경우는 암호화가 아닌 인코딩이므로 암호화를 위해 Base64 인코딩을 사용하였다면 취약한 것으로 판단한다.

제 3 절 시간 및 상태

1. 개요

동시 또는 거의 동시 수행을 지원하는 병렬 시스템이나 하나 이상의 프로세스가 동작되는 환경에서 시간 및 상태를 부적절하게 관리하여 발생할 수 있는 보안약점을 제거하여야 한다.

2. 세부 보안약점

시간 및 상태는 “경쟁조건: 검사시점과 사용시점(TOCTOU)” 등 1개의 보안약점으로 소스코드 보안약점 진단방법은 소프트웨어 보안약점 진단가이드를 참조한다.

세부 보안 약점

1	경쟁조건: 검사시점과 사용시점(TOCTOU)

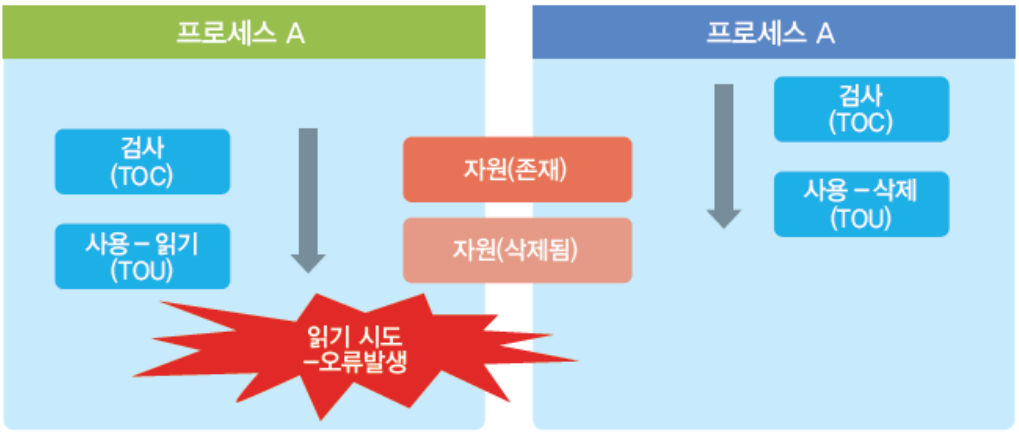
3. 주요 진단사례

시간 및 상태에서 많이 발견되는 소스코드 보안약점은 “경쟁조건: 검사시점과 사용시점(TOCTOU)”으로 진단 방법은 다음과 같다.

가. 경쟁조건: 검사시점과 사용시점(TOCTOU)

1) 개요

병렬 실행 환경의 응용프로그램에서는 자원을 사용하기 전에 자원의 상태를 검사한다. 그러나 자원을 사용하는 시점에 자원의 상태가 변하는 경우가 있다. 이것으로 인해 프로그램에 여러 가지 문제, 즉 교착 상태, 경쟁 조건 및 기타 동기화 오류 등이 발생할 수 있다.



〈 경쟁조건 : 검사시점과 사용지점(TOCTOU) 〉

2) 보안대책

- 공유자원(예: 파일)을 여러 스레드가 접근하여 사용할 경우, 동기화 구문(예: synchronized)을 사용하여 한 번에 하나의 스레드만 접근 가능하도록 한다.

안전한 코드의 예 Android-JAVA

```
1. // 공유자원(파일 등)을 여러 스레드가 접근하여 사용할 경우, 동기화 구문을
2. // 이용하여 한 번에 하나의 스레드만 접근 가능하도록 변경한다.
3. public class SA367 extends Activity {
4.     public void onCreate(Bundle savedInstanceState) {
5.     super.onCreate(savedInstanceState);
6.
7.     FileAccessThread fileAccess = new FileAccessThread();
8.     Thread first = new Thread(fileAccess);
9.     Thread second = new Thread(fileAccess);
10.    Thread third = new Thread(fileAccess);
11.    Thread fourth = new Thread(fileAccess);
12.    first.start();
13.    second.start();
14.    third.start();
15.    fourth.start();
16.    }
17. }
```

```
18.
19. class FileAccessThread implements Runnable {
20.     public synchronized void run() {
21.         try {
22.             File f = new File("Test.txt");
23.             if (f.exists()) { // 만약 파일이 존재하면 파일 내용을 읽음
24.                 Thread.sleep(100); // 시간이 소요되는 작업을 가정함
25.                 BufferedReader br = new BufferedReader(new FileReader(f));
26.                 System.out.println(br.readLine());
27.                 br.close(); // 파일 내용을 모두 읽은 후 삭제
28.                 f.delete();
29.             }
30.         } catch (IOException e) { // 예외처리
31.             System.err.println("IOException occured");
32.         }
33.     }
34. }
```

3) 진단방법

- 소스코드 상에 공유자원(파일, 폴더 등)을 스레드가 사용하는지 확인한다.(①)
- 파일의 존재를 확인하는 부분과 실제로 파일을 사용하는 부분을 실행하는 과정에서 시간차가 발생하는 경우, 파일에 대한 삭제가 발생하여 프로그램이 예상하지 못하는 형태로 수행될 수 있다. 또한 시간차를 이용하여 파일을 변경하는 등의 공격에 취약할 수 있다.(②)

안전하지 않은 코드의 예 Android-JAVA

```
1. // 파일의 존재를 확인하는 부분과 실제로 파일을 사용하는 부분을 실행하는
2. // 과정에서 시간차가 발생하는 경우, 파일에 대한 삭제가 발생하여 프로그램이
3. // 예상하지 못하는 형태로 수행될 수 있다.
4. // 또한 시간차를 이용하여 파일을 변경하는 등의 공격에 취약할 수 있다.
5.
6. public class UA367 extends Activity {
7.     public void onCreate(Bundle savedInstanceState) {
```

```
8.          super.onCreate(savedInstanceState);
9.          FileAccessThread fileAccessThread = new FileAccessThread();
10.         FileDeleteThread fileDeleteThread = new FileDeleteThread();
11.         fileAccessThread.start();
12.         fileDeleteThread.start();
13.     }
14. }
15.
16. class FileAccessThread extends Thread { ------------- ①
17.     public void run() {
18.         try {
19.             File f = new File("Test_367.txt"); ------------- ②
20.             if (f.exists()) { // 만약 파일이 존재하면 파일내용을 읽음
21.                 BufferedReader br = new BufferedReader(new FileReader(f));
22.                 br.close();
23.             }
24.         } catch(FileNotFoundException e) {
25.             System.out.println("Exception Occurred") ; //예외처리
26.         } catch(IOException e) {
27.             System.out.println("Exception Occurred") ; //예외처리
28.     }
29. }
30.
31. class FileDeleteThread extends Thread { ------------- ①
32.     public void run() {
33.         try {
34.             File f = new File("Test_367.txt"); ------------- ②
35.             if (f.exists()) { // 만약 파일이 존재하면 파일을 삭제함
36.                 f.delete();
37.             }
38.         } catch(FileNotFoundException e) {
39.             System.out.println("Exception Occurred") ; //예외처리
40.         } catch(IOException e) {
41.             System.out.println("Exception Occurred") ; //예외처리
42.         }
43.     }
44. }
```

4) 진단기준

- 하나의 공유자원(예, 파일)을 동시에 접근할 가능성이 존재한다면 취약한 것으로 판단한다.
- 동기화 구문(예: synchronized) 등을 사용하거나 공유자원 엑세스를 관리하는 Pool 형태의 자체 모듈을 제작할 경우에는 안전하다고 판단한다.

제 4 절 에러처리

1. 개요

에러를 처리하지 않거나, 불충분하게 처리하여 에러 정보에 중요정보(시스템 등)가 포함될 때 발생할 수 있는 보안약점을 제거하여야 한다.

2. 세부 보안약점

에러처리는 “오류메시지 및 시스템 데이터 정보노출”, “오류상황 대응 부재” 등 3개의 보안약점으로 소스코드 보안약점 진단방법은 소프트웨어 보안약점 진단가이드를 참조한다.

세부 보안 약점

1	오류메시지 및 시스템 데이터 정보노출	3	부적절한 예외 처리
2	오류상황 대응 부재		

3. 주요 진단사례

에러처리에서 많이 발견되는 소스코드 보안약점은 “오류메시지 및 시스템 데이터 정보노출”, “오류상황 대응 부재”로 진단 방법은 다음과 같다.

가. 오류 메시지 및 시스템 데이터 정보노출

1) 개요

사용자가 볼 수 있는 오류 메시지나 스택 정보에 시스템 내부 데이터나 디버깅 관련 정보가 공개되지 않아야 한다.

〈 오류 메시지를 기반한 내부 정보 노출 〉

2) 보안대책

- 최종 사용자에게 배포되는 SW에서는 공격에 활용될 수 있는 내부 구조나 민감한 정보를 오류 메시지로 출력하지 말아야 한다.

안전한 코드의 예 Android-JAVA

```
1. // 예외 이름이나 스택 트레이스를 출력하지 않는다.
2. ……
3. public void onCreate(Bundle savedInstanceState) {
4.     super.onCreate(savedInstanceState);
5.     try{
6.         throw new IOException();
7.     }
8.     catch (IOException e) {
9.         Toast.makeText(getBaseContext(), "에러 발생", Toast.LENGTH_LONG).show();
10.    }
11. }
```

- 또한, 예외상황이 발생할 경우 시스템 메시지 등의 정보를 화면에 출력하도록 하는 경우가 많다. 가급적이면 공격의 빌미가 될 수 있는 오류와 관련된 상세한 정보는 최종사용자에게 노출하지 않는다.

안전한 코드의 예 Android-JAVA

```
1. // 가급적이면 공격의 빌미가 되는 오류와 관련된 상세한 정보는 최종 사용자에게
2. // 노출하지 않도록 한다.
3. ……
4. public void onCreate(Bundle savedInstanceState) {
5.     super.onCreate(savedInstanceState);
6.     setContentView(R.layout.main);
7.     try { ioFunction(); }
8.         catch (IOException e) {
9.             // end user가 볼 수 있는 오류 메시지 정보를 생성하지 않아야 한다.
10.            Toast.makeText(getBaseContext(), "IOException Occured", Toast.LENGTH_LONG).show();
11.        }
12.    }
13. private void ioFunction() throws IOException { …… }
14. ……
```

3) 진단방법

- 예외 이름이나 오류추적 정보를 출력하면 프로그램 내부 정보가 유출될 수 있으므로 오류메시지를 출력하는 경우 해당오류에 시스템 환경, 정보, 데이터 등 민감한 정보가 포함되어 있는지 확인한다.(①)

안전하지 않은 코드의 예 Android-JAVA

```
1. // 예외 이름이나 스택 트레이스를 출력하면 프로그램 내부 정보가 유출된다.
2. ……
3. public void onCreate(Bundle savedInstanceState) {
4.     super.onCreate(savedInstanceState);
5.     try{ throw new IOException(); }
6.     catch (IOException e) { e.printStackTrace(); } ---------- ①
7. }
```

- 또한, 예외 발생시 getMessage() 등으로 오류와 관련된 시스템 에러정보 등 민감한 정보가 유출될 수 있는지 확인한다.(①)

안전하지 않은 코드의 예 Android-JAVA

```
1. // 예외 발생시 getMessage()로 오류와 관련된 시스템 에러정보 등
2. // 민감한 정보가 유출될 수 있다.
3. ……
4. public void onCreate(Bundle savedInstanceState) {
5.     super.onCreate(savedInstanceState);
6.     setContentView(R.layout.main);
7.     try { ioFunction(); }
8.     catch (IOException e) {
9.         // 예외 발생시 e.getMessage()로 오류 메시지 정보가 유출된다.
10.        Toast.makeText(getBaseContext(), e.getMessage(), Toast.LENGTH_LONG).show(); -- ①
11.    }
12. }
13. private void ioFunction() throws IOException { …… }
14. ……
```

4) 진단기준

- 오류 메시지에 환경 및 사용자 관련 데이터 등의 내부 정보가 유출될 경우 취약한 것으로 판단하다.
- getMessage() 등의 함수를 이용해 시스템 내부 데이터나 디버깅 정보를 공개하면 취약한 것으로 판단한다.

제 5 절 코드오류

1. 개요

타입변환 오류, 자원(메모리 등)의 부적절한 반환 등과 같이 개발자가 범할 수 있는 코딩 오류로 인해 유발되는 보안약점을 제거하여야 한다.

2. 세부 보안약점

코드오류는 “Null Pointer 역참조”, “부적절한 자원 해제” 등 2개의 보안약점으로 소스코드 보안약점 진단방법은 소프트웨어 보안약점 진단가이드를 참조한다.

세부 보안 약점

1	Null Pointer 역참조	2	부적절한 자원 해제

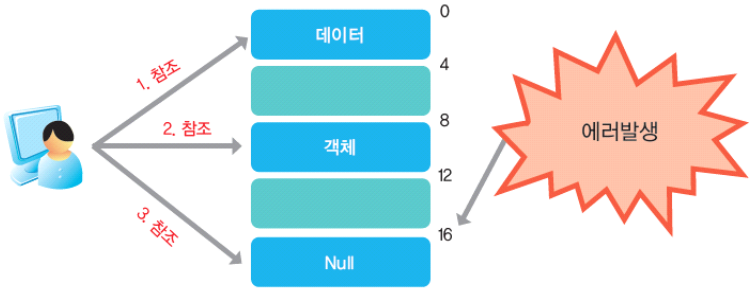
3. 주요 진단사례

코드오류에서 많이 발견되는 소스코드 보안약점은 “Null Pointer 역참조”, “부적절한 자원 해제”로 진단 방법은 다음과 같다.

가. Null Pointer 역참조

1) 개요

Null Pointer 역참조는 '일반적으로 '그 객체는 NULL이 될 수 없다'라고 하는 가정을 위반했을 때 발생한다. 공격자가 의도적으로 NULL 포인터 역참조를 실행하는 경우, 그 결과 발생하는 예외 사항을 이용하여 추후의 공격을 계획하는 데 사용될 수 있다.



〈 Null Pointer 역참조 〉

2) 보안대책

- 널(null) 값이 될 수 있는 레퍼런스(reference)는 참조하기 전에 널(null) 값 여부를 검사하여 안전한 경우에만 사용해야 한다.

안전한 코드의 예 Android-JAVA

```
1. //cmd가 널인지 우선 검사 후에 사용.
2. public void onCreate(Bundle savedInstanceState) {
3.     uper.onCreate(savedInstanceState);
4.     String cmd = System.getProperty("cmd");
5.     EditText text = (EditText) findViewById("R.id.text");
6.     if(cmd != null) cmd = cmd.trim();
7.     String name = text.setText(cmd);
8. ……
```

3) 진단방법

- 참조되는 객체가 널 값이 될 수 있는지 확인한다.(①) 프로그래머는 "cmd" 프로퍼티가 항상 정의되어 있다고 가정하지만(②), 공격자가 프로그램의 환경을 제어해 "cmd" 프로퍼티가 정의되지 않게 하면, cmd는 널이 되어 trim() 메소드를 호출 할 때 Null Pointer 예외가 발생하게 된다.

안전하지 않은 코드의 예 Android-JAVA

```
1. // "cmd" 속성이 항상 정의되어 있다고 가정하고 있지만, 공격자가 "cmd" 속성을
2. // 조작하면, cmd은 null이 되고 trim() 메소드 호출 시 널 포인터 예외가 발생
3. ……
4. public void onCreate(Bundle savedInstanceState) {
5.     super.onCreate(savedInstanceState);
6.     String cmd = System.getProperty("cmd"); ------------ ②
7.     EditText text = (EditText) findViewById("R.id.text");
8.     cmd = cmd.trim(); ---------------- ①
9.     String name = text.setText(cmd);
10. ……
```

4) 진단기준

- 표현된 객체가 널 값이 될 수 있는지 확인하여 널 값을 체크하여 예외 처리를 한 경우 안전하다고 판단한다.
- 널 체크를 하지 않은 경우에는 취약한 것으로 판단한다.

제 6 절 API 오용

1. 개요

의도된 사용에 반하는 방법으로 API를 사용하거나, 보안에 취약한 API를 사용하여 발생할 수 있는 보안약점을 제거하여야 한다.

2. 세부 보안약점

API 오용은 “취약한 API 사용” 등 1개의 보안약점으로 소스코드 보안약점 진단방법은 소프트웨어 보안약점 진단가이드를 참조한다.

세부 보안 약점

1	취약한 API 사용

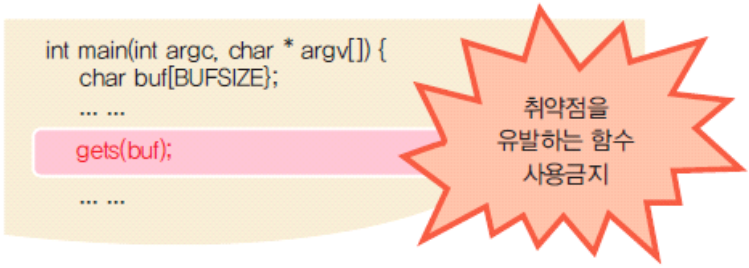
3. 주요 진단사례

API 오용에서 많이 발견되는 소스코드 보안약점은 “취약한 API 사용”으로 진단 방법은 다음과 같다.

가. 취약한 API 사용

1) 개요

특정 라이브러리 함수들은 보안 취약성을 전혀 고려하지 않고 개발되어서, 사용 자체가 취약성이 될 수 있다. 예를 들면, gets() strcpy() 함수는 입력 크기 제한 사항을 점검하지 않기 때문에, 입력 버퍼를 넘치게 할 수 있다. 즉, 이러한 함수들은 사용 자체만으로 보안취약성이 야기된다.



〈 취약한 API 사용 〉

2) 보안대책

- gets, sprintf, strcat, strcpy, vsprintf 등 이미 위험하다고 알려진 라이브러리 함수를 사용하지 말아야 한다.
- 아래의 예제코드에서는 버퍼 오버플로우 공격에 취약한 strcpy() 함수 대신 바운더리를 체크하는 strlcpy() 함수를 사용한다.

안전한 코드의 예 Objective C

```
1. - (void)goodCode: (char*)tempString {
2.     #define BUFSIZE 10
3.     char *dest = NULL;
4.     dest = (char *)malloc(BUFSIZE);
5.     strlcpy(dest, tempString, BUFSIZE);
6.     NSLog(@"dest : %s", dest);
7. }
```

3) 진단방법

- 취약한 API를 사용하는지 확인하여 취약한 API인 strcpy() 함수 사용을 확인한다.(①)

안전하지 않은 코드의 예 Objective C

```
1. char buf1[100];
2. char buf2[200];
3. scanf("%s", buf2);
4. strcpy(buf1, buf2); ---------- ①
```

- 취약한 API 함수사용이 없는 경우에는 사용하는 API의 안전한 설정여부를 확인하여 buf 문자열이 10바이트로 제한됨에도 불구하고,(①) 입력버퍼의 바운더리 체크를 하지 않는 strcpy() 함수를 사용하여 버퍼 오버플로우가 발생할 수 있는지 확인한다.(②)

안전하지 않은 코드의 예 Objective C

```
1. - (void)badCode: (char*)tempString {
2.     char buf[10]; ------------- ①
3.     strcpy(buf, tempString); ------------- ②
4.     NSLog(@"buf : %s", buf);
5. }
```

4) 진단기준

- gets, sprintf, strcat, strcpy, vsprintf 등 이미 위험하다고 알려진 라이브러리 함수를 사용할 때에는 취약한 것으로 판단한다.
- 취약한 API를 사용하지 않아도 API의 설정여부를 확인하여 취약점이 발생 가능한지를 확인한다.

제 7 절 모바일 환경 특화

1. 개요

공개영역(CWE, CWE/SANS Top25, OWASP Mobile Top 10 등) 관련 모바일 환경에 특화된 보안약점이 발견되지 않도록 제거하여야 한다.

2. 세부 보안약점

모바일 환경 특화는 “안드로이드 애플리케이션 컴포넌트의 부적절한 접근 허용” 등 6개의 보안약점으로 소스코드 보안약점 진단방법은 아래의 주요 진단사례를 참조한다.

세부 보안 약점

1	안드로이드 애플리케이션 컴포넌트의 부적절한 접근 허용	4	안드로이드의 권한 검사 우회
2	민감한 정보 전송을 위한 암시적 intent 사용	5	클래스 로딩 하이재킹
3	접근제어 없이 내 · 외부저장소 사용	6	소스코드 난독화 미적용

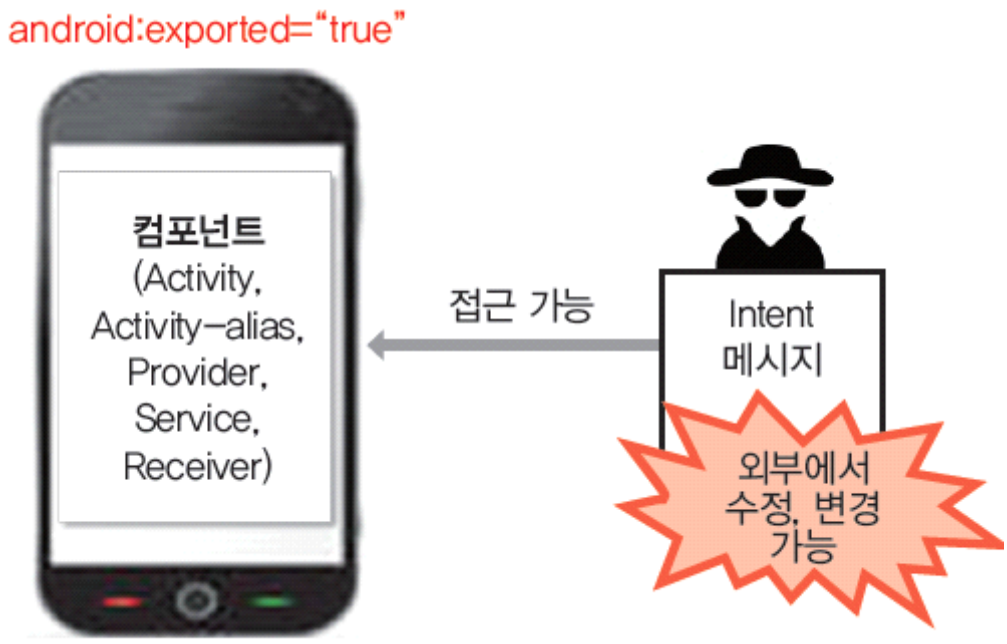
3. 주요 진단사례

모바일 환경 특화에서 많이 발견되는 소스코드 보안약점은 진단 방법은 다음과 같다.

가. 안드로이드 애플리케이션 컴포넌트의 부적절한 접근 허용(Android)

1) 개요

안드로이드 애플리케이션 컴포넌트(Component)를 부적절하게 설정할 경우, 외부 애플리케이션에서 컴포넌트가 의도치 않게 실행될 수 있다. Manifest의 exported 속성으로 외부 애플리케이션에서 실행 가능 여부를 설정할 수 있으며, 실행이 가능한 컴포넌트는 Activity, Activity-alias, Content Provider, Service, Broadcast Receiver 이다. export가 가능하도록 설정된 Activity로 악성행위를 수행할 수 있으며, Contents Provider에 민감한 데이터가 저장되어 있다면 데이터를 조회하거나 변경할 수 있다.



〈 외부 애플리케이션에서 컴포넌트로 접근 가능 〉

2) 보안대책

- 컴포넌트에 대한 속성을 android:exported="false"로 설정하여 외부 애플리케이션에 컴포넌트에 대한 접근권한을 외부에 제공하지 않는 것이 바람직하다.

  ※ exported 값이 "false" 일 경우에는 같은 애플리케이션이거나 또는 같은 user ID를 가진 애플리케이션만 해당 컴포넌트로의 호출이 가능

안전한 코드의 예 Android-JAVA

```
1. <?xml version="1.0" encoding="utf-8"?>
2. <manifest xmlns:android="http://schems.android.com/apk/res/android"
3.     package="com.example.android.sample"
4.     android:versionCode="1" android:versionName="1.0">
5.     <application android:icon="@drawable/icon" android:label="@string/label">
6.         <service android:name=".syncadapter.SyncService" android:exported="false">
7.         .....
8.     </application>
9. </manifest>
```

3) 진단방법

- 애플리케이션에서 사용하는 안드로이드 컴포넌트들을 확인하고, Android Manifest.xml 파일에서 android:exported의 속성 값을 확인한다.(①)

안전하지 않은 코드의 예 Android-JAVA

```
1. 〈?xml version="1.0" encoding="utf-8"?〉
2. 〈manifest xmlns:android="http://schems.android.com/apk/res/android"
3.     package="com.example.android.sample"
4.     android:versionCode="1" android:versionName="1.0"〉
5.     〈application android:icon="@drawable/icon" android:label="@string/label"〉
6.         〈service android:name=".syncadapter.SyncService" android:exported="true"〉①
7. .         ........
8.     〈/application〉
9. 〈/manifest〉
```

4) 진단기준

- 컴포넌트에 대한 속성을 android:exported="true"일 경우 취약한 것으로 판단한다.

참고 컴포넌트

exported 초기 설정 관련 참고사항

exported 초기 설정값은 intent-filter의 유무로 결정. intent-filter가 존재하지 않는 경우 다른 애플리케이션에서 존재를 알 수 없어 "false"로 처리된다. 반대로 intent-filter가 한 개라도 존재하는 경우에는 외부에서 존재를 알 수 있으므로, 자동으로 "true"와 같은 효과 발생
※ android:exported 의 기본값은 false로 설정된다.

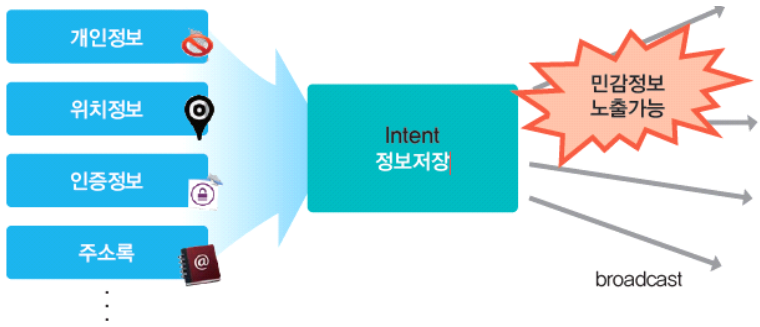
MAIN/LAUNCHER Activity 관련 참고사항

MAIN/LAUNCHER Activity에는 exported="false" 속성을 설정하면 에러(SecurityException)가 발생하기 때문에 MAIN/LAUNCHER Activity는 exported="true"로 설정

나. 민감한 정보 전송을 위한 암시적 intent 사용(Android)

1) 개요

애플리케이션은 다른 애플리케이션에 메시지를 전달하기 위해 intent에 정보를 저장하여 브로드캐스트를 수행할 수 있다. 메시지를 전달받는 애플리케이션은 브로드캐스트 리시버(Broadcast Receiver)를 설정하여 intent의 정보를 얻을 수 있다. 하지만, 중요정보(개인정보, 민감정보 등)의 내용이 담긴 intent를 브로드캐스트로 전달하면 외부 애플리케이션에서 중요정보에 대해 도청 공격이 가능하다. 또한, 악의적인 브로드캐스트를 전달하게 되면 서비스 거부 공격(DoS, Denial of Service attack)도 가능하다.



〈민감정보가 담긴 intent의 브로드캐스트〉

2) 보안대책

- 중요정보를 intent를 사용하여 브로드캐스트로 전송하지 말아야 한다.
- 다른 애플리케이션에게 민감한 정보가 포함된 intent를 전송하지 않는 LocalBroadcastManager 클래스를 사용한다.
  ※ androidx.localbroadcastmanager는 버전 1.1.0-alpha01에서 지원 중단됨
- LocalBroadcastManager 클래스가 지원되지 않을 경우 LiveData 반응형 스트림을 사용한다.

안전한 코드의 예 Android-JAVA

```
1. Intent intent = newIntent("my-sensitive-event");
2. intent.putExtra("event",this is a test event");
3. LocalBroadcastManager.getInstance(this).sendBroadcast(intent);
4. // 민감한 정보를 LocalBroadcastManager를 사용하여 전송
```

안전한 코드의 예 Android-JAVA

```
public class NameActivity extends AppCompatActivity {

    private NameViewModel model;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Other code to setup the activity...

        // Get the ViewModel.
        model = new ViewModelProvider(this).get(NameViewModel.class);

        // Create the observer which updates the UI.
        final Observer<String> nameObserver = new Observer<String>() {
            @Override
            public void onChanged(@Nullable final String newName) {
                // Update the UI, in this case, a TextView.
                nameTextView.setText(newName);
            }
        };

        // Observe the LiveData, passing in this activity as the LifecycleOwner and
the observer.
        model.getCurrentName().observe(this, nameObserver);
    }
}
```

3) 진단방법

- intent에 중요정보를 사용하는지 확인한다.(①)
- 보안에 취약한 권한이 설정되어 있는지 확인한다.(②)
- 중요 정보가 저장된 intent를 브로드캐스트로 전송하는지 확인한다.(③)

안전하지 않은 코드의 예 Android-JAVA

```
1. Intent intent = new Intent();
2. intent.setAction("com.example.CreateUser");
3. intent.putExtra("Username",uname_string); ①
4. intent.putExtra("Password", pw_string); ①
5. intent.addFlag(FLAG_GRANT_READ_URI_PERMISSION); ②
6. SendBroadcast(intent); ③
```

4) 진단기준

- Intent에 중요정보를 담아 브로드캐스트로 전송할 경우 취약한 것으로 판단한다.
- intent의 일부 권한인 FLAG_GRANT_READ_URI_PERMISSION, FLAG_GRANT_WRITE_URI_PERMISSION과 같은 권한으로 부여하면 취약한 것으로 판단한다.

참 고 intent 권한

취약한 intent 권한 참고사항

- FLAG_GRANT_READ_URI_PERMISSION : 지정된 URI 읽기에 대한 권한(Permission)
- FLAG_GRANT_WRITE_URI_PERMISSION : 지정된 URI 쓰기에 대한 권한(Permission)

다. 접근제어 없이 내 · 외부저장소 사용(Android)

1) 개요

안드로이드 내 · 외부 저장소(SD카드 등)에서 파일 저장시 다른 애플리케이션에서 접근할 수 있도록 인자값으로 MODE_WORLD_READABLE, MODE_WORLD_WRITABLE 등을 사용할 경우, 해당 파일에 대해 정보가 노출되거나 수정이 가능할 수 있다.

〈접근제어 없이 내 · 외부저장소 사용〉

2) 보안대책

- 다른 애플리케이션에서 민감한 데이터 파일에 접근하지 못하도록 속성을 MODE_PRIVATE으로 설정한다.

안전한 코드의 예 Android-JAVA

```
1. privateString filename = “secret.txt”
2. privateString string = “sensitive data such as credit card number”
3. FileOutputStream fos = null;
4.
5.     try {
6.         fos = openFileOutput(filename, Context.MODE_PRIVATE);
7.         fos.write(string.getByte());
8.         fos.close();
```

3) 진단방법

- 외부 애플리케이션에서 접근이 가능하도록 안드로이드 내 · 외부 저장소에 파일을 저장하는지 확인한다.(①)

안전하지 않은 코드의 예 Android-JAVA

```
1. privateString filename = “secret.txt”
2. privateString string = “sensitive data such as credit card number”
3. FileOutputStream fos = null;
4.
5.     try {
6.         fos = openFileOutput(filename, Context.MODE_WORLD_WRITEABLE);①
7.         fos.write(string.getByte());
8.         fos.close();
```

4) 진단기준

- 안드로이드 내·외부 저장소에 중요한 파일 생성 시 접근제어 모드가 “MODE_WORLD_READABLE” 또는 “MODE_WORLD_WRITEABLE” 로 설정한 경우 취약한 것으로 판단한다.
- 외부 애플리케이션에서 접근이 불가능한 “MODE_PRIVATE”로 설정한 경우 안전한 것으로 판단한다.

참 고 접근제어 모드

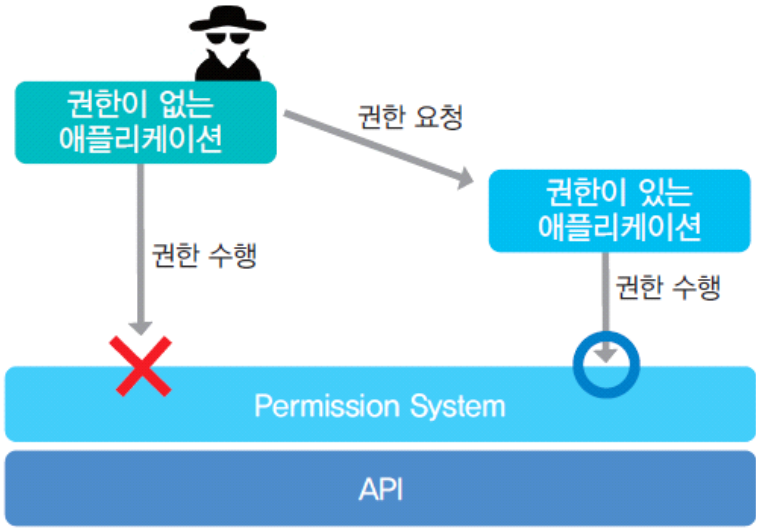
안드로이드 플랫폼의 파일생성 접근제어 모드 참고사항

- MODE_WORLD_READABLE : 다른 앱들이 읽을 수 있도록 허용하는 모드
- MODE_WORLD_WRITEABLE : 다른 앱들이 쓸 수 있도록 허용하는 모드
- MODE_PRIVATE : 생성한 APP에서만 저장된 파일을 사용

라. 안드로이드의 권한 검사 우회(Android)

1) 개요

필요한 권한이 없거나 권한이 전혀 없는 호출 애플리케이션이 다른 애플리케이션의 권한을 사용하여 권한 검사를 우회할 수 있다. 이로 인해, 적절한 권한이 없는 악성 애플리케이션에 접근 권한을 부여할 수 있으며, confused deputy 공격[31]이 발생할 수 있다.



〈 권한이 없는 애플리케이션이 권한 검사를 우회하여 권한을 수행 〉

2) 보안대책

- 호출하는 애플리케이션이 올바른 사용 권한을 갖고 있는지 확인하기 위해 Context.checkCallingPermission() 또는 Context.enforceCallingPermission() 함수를 사용한다.

안전한 코드의 예 Android-JAVA

```
1. Context c = MainActivity.this;
```

31) confused deputy 공격은 서브젝트가 클라이언트의 권한을 사용해야 하는 경우에 실수로 자기 자신의 권한으로 일을 수행했을 경우를 일컫는다.

```
2. //현재 activity에서 context를 가져옵니다.
3.     if(c.checkCallingPermission(permission.INTERNET) ==
PackageManager.PERMISSION_DENIED){
4.     Toast.makeText(getBaseContext(),
5.     "INTERNET PERMISSION_DENIED",
6.     Toast.LENGTH_SHORT ).show();
7.     }else if(c.checkCallingPermission(permission.INTERNET) ==
PackageManager.PERMISSION_GRANTED){
8.         Toast.makeText(getBaseContext(),
9.         "INTERNET PERMISSION_GRANTED",
10.        Toast.LENGTH_SHORT ).show();
11.    }
```

3) 진단방법

- 애플리케이션 권한검사 우회가 가능한 checkCallingOrSelfPermission(), checkCallingOrselfUriPermission() 함수를 사용하고 있는지 확인한다.(①)

안전하지 않은 코드의 예 Android-JAVA

```
1. private boolean checkPermission() {
2.     if (getContext().checkCallingOrSelfPermission("com.test.testpermission") ==
       PackageManager.PERMISSION_GRANTED) {①
3.         return true;
4.     }
5.     return false;
6. }
```

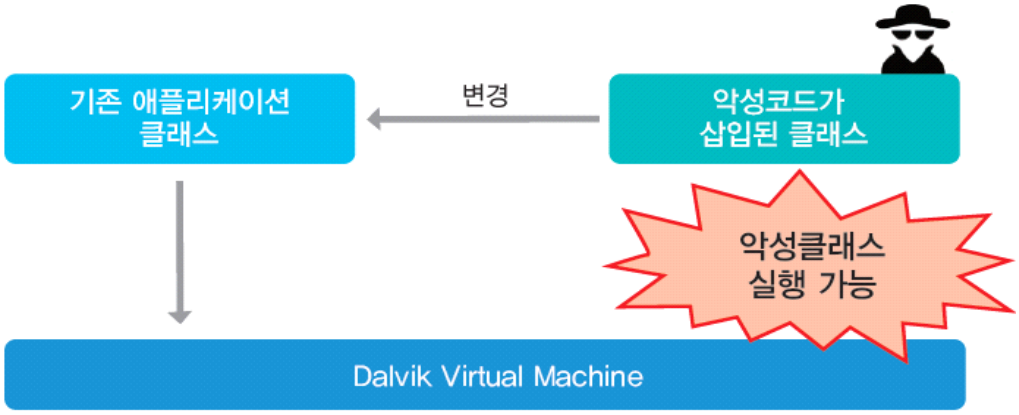
4) 진단기준

- checkCallingOrSelfPermission() 또는 checkCallingOrselfUriPermission() 함수는 악성 애플리케이션이 해당 애플리케이션의 권한을 사용하여 권한검사를 우회할 수 있으므로 취약한 것으로 판단한다.

마. 클래스 로딩 하이재킹(Android)

1) 개요

신뢰할 수 없는 소스 또는 환경에서 클래스를 로드하면 애플리케이션이 공격자 대신 악의적인 명령을 실행할 수 있다. 즉, 애플리케이션이 클래스를 로드하기 위해 검색하는 디렉터리의 이름을 공격자가 변경하여 자신이 제어권을 가진 디렉터리를 가리키도록 하는 등의 악성행위가 가능하다.



〈클래스를 조작하여 악성코드 실행〉

2) 보안대책

- 절대경로를 사용하여 로드 가능한 클래스를 명시적으로 제어한다.
- 외부 저장소에 있는 클래스를 로드하지 않는다.
- 로드하는 클래스가 의도된 클래스인지 여부를 확인한다.

안전한 코드의 예 Android-JAVA

```
1. for (File file : files) {
2.     final File tmpDir = new File("data/local/tmp/optdexjars/" + optDexFolder + "/");
3.     tmpDir.mkdir();
4.     final DexClassLoader classloader = new DexClassLoader(
5.         file.getAbsolutePath(), tmpDir.getAbsolutePath(), "data/local/tmp/natives/", ClassLoader.
       getSystemClassLoader());
6.         Class<?> classToLoad = (Class<?>) classloader .loadClass("com.registry.Registry");
7.         Field classesField = classToLoad.getDeclaredField("_classes");
8.         ArrayList<Class<?>> classes = (ArrayList<Class<?>>) classesField.get(null);
9.     }
```

3) 진단방법

- 애플리케이션에서 클래스가 로드된 부분을 확인하고, 상대경로를 활용하여 클래스가 로드되는지 확인한다.(①)
- 외부 저장소에 있는 클래스가 로드되는지 여부를 확인한다.(②)
- 로드되는 클래스가 원래 의도한 클래스인지에 대한 검사 여부를 확인한다.

안전하지 않은 코드의 예 Android-JAVA

```
1. ...
2. DexClassLoader dexclassLoader = new DexClassLoader(mypath, optimizeDexOutputPath.
   getAbsolutePath(), null, getClassLoader();①
3. // 상대경로로 로드할 클래스를 확인한다.
4. ...
```

안전하지 않은 코드의 예 Android-JAVA

```
1. ...
2. DexClassLoader dexClassLoader = new DexClassLoader(absolutePath, Environment.
   getExternalStorageDirectory(), null, getClassLoader());②
3. // 외부저장소에서 클래스를 로드한다.
4. ...
```

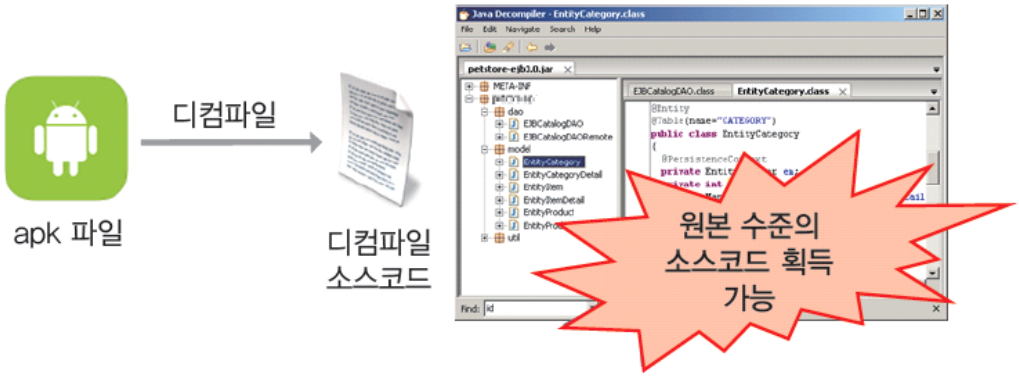
4) 진단기준

- 상대경로를 사용하여 클래스가 로드되거나, 의도하지 않은 클래스가 로드되는 경우 취약한 것으로 판단한다.
- 외부저장소에서 클래스를 로드되는 경우 취약한 것으로 판단한다.

바. 소스코드 난독화 미적용(Android)

1) 개요

안드로이드 플랫폼을 기반으로 개발된 모바일 애플리케이션의 경우, 디컴파일시 안드로이드 실행파일을 소스코드 쉽게 변환시킬 수 있어 애플리케이션 구조에 대한 분석이 가능하다. 소스코드 난독화 기술을 적용하지 않을 시에는 원본수준의 소스코드가 외부로 노출될 수 있으며, 애플리케이션의 역공학을 이용하여 소스코드를 확보 후, 위/변조하여 중요정보 탈취 및 추가적 범죄에 이용될 우려가 높다.



〈 apk파일을 디컴파일하여 소스코드 확인 〉

2) 보안대책

- 난독화 기능이 우수한 상용도구를 이용하는 것을 권고하나, 그러지 못할 경우 최소한 구글에서 제공하는 오픈소스 난독화도구(ProGuard) 등을 적용하여야 한다.
- ProGuard 설정 시 Android Studio 내 main module 내에 있는 build.gradle 파일 내 minifyEnabled를 true로 설정한다.

안전한 코드의 예 Android-JAVA

```
buildTypes {
        debug {
            minifyEnabled true
        }
        release {
            minifyEnabled true
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
  }
```

3) 진단방법

- ProGuard 설정 시 Android Studio 내 main module 내에 있는 build.gradle 파일 내 minifyEnabled를 설정 값을 확인한다.

안전하지 않은 코드의 예 Android-JAVA

```
buildTypes {
        debug {
            minifyEnabled false
        }
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),
'proguard-rules.pro'
        }
  }
```

4) 진단기준

- 난독화 기능이 우수한 상용도구를 적용할 경우 안전한 것으로 판단한다.
- 난독화 기능이 우수한 상용도구를 이용하는 않거나 ProGuard 설정이 되어있지 않는 경우 취약한 것으로 판단한다.

[Part 4]

앱 개발자를 위한 보안공통기반 제공

Part 4

앱 개발자를 위한 보안공통기반 제공

행정안전부는 행정기관 및 공공기관 등이 대국민 모바일 서비스 구축 시 앱 위·변조 방지, 화면캡처 방지, 문서변환 등 모바일 관련 보안 요소를 공동으로 활용할 수 있는 시스템인 모바일 공용 프레임워크를 구축·운영하고 있으며 "모바일 대국민 보안공통기반"의 주요 기능은 아래와 같다.

모바일 대국민 보안공통기반 주요 기능

번호	주요기능	모바일 대국민 보안공통기반기능 설명	대응 보안위협
1	접속 기능	• 대국민 공통기반과 각 기관 대국민 모바일 서비스 서버간 서비스 연계 - 서비스 요청을 하는 단말 또는 기관 모바일 서비스 서버의 1차 접속 포인트 - 단말 또는 모바일 서비스 서버로부터의 서비스 요청에 대한 응답 수행	-
2	인증 기능	• 공통기반으로 접속하는 모바일 서비스 서버의 정당성을 전자서명으로 검증 기능	사용자 인증
3	문서변환	• 문서 자료의 유출 방지를 위해 원본 문서를 이미지 등의 형태로 변환하여 단말에서 열람할 수 있는 기능	자료 유출 방지
4	화면 캡쳐 방지	• 대국민 모바일 서비스에서 제공되는 각종 정보를 사용자가 모바일 단말에서 캡처가 불가능 하도록 통제하는 기능	화면 캡쳐 방지
5	앱 위·변조 방지	• 대국민 모바일 서비스 정보를 사전에 등록하여 위·변조여부 검증 기능	위·변조 앱 사용 방지
6	단말 위·변조 방지	• 사용자 단말의 탈옥, 루팅 등의 여부를 확인하여 위·변조 여부에 따라 서비스 접근 거부 기능	위·변조 단말 사용 방지

모바일 대국민 보안공통기반 주요 기능			
번호	주요기능	모바일 대국민 보안공통기반기능 설명	대응 보안위협
7	E2E 암호화	• 사용자 단말과 공통기반, 각 기관 대국민 모바일 서비스 서버와 공통기반 서버간의 E2E 암호화 통신 기능 제공	전송데이터 유출
8	가상키보드	• 모바일 기기의 키패드로부터 입력되는 주요 정보 보안 강화 – 모바일 기기에서 비밀번호 등 중요 정보 입력 시 모바일 기기 키패드의 위치를 무작위로 변경하여 입력 정보에 대한 보안 강화 – 메모리 해킹 및 키로깅(key logging) 방지 기능	입력정보 유출 방지
9	푸시(PUSH)	• 모바일 기기로 메시지 전송을 위한 푸쉬(Push) 시스템 도입	–

모바일 대민서비스 구축 · 운영시 “모바일 대국민 보안공통기반”을 활용하여 보안취약점을 제거할 수 있으며, 아래의 표는 모바일 대민서비스 보안취약점 제거를 위한 보안공통기반 활용 방안을 나타낸다.

보안취약점 제거를 위한 모바일 대국민 보안공통기반 활용 방안			
번호	보안공통기반 주요기능	모바일 앱 보안취약점	비고
1	접속 기능	–	–
2	인증 기능	–	–
3	문서변환	–	–
4	화면 캡쳐 방지	–	–
5	앱 위 · 변조 방지	–	–
6	단말 위 · 변조 방지	• 루팅 및 탈옥 기기에서의 정상동작	제3절 플랫폼
7	E2E 암호화	• 중요정보의 평문 저장 및 전송 • 기타 중요정보의 평문 저장 및 전송	제4절 식별 · 활용 및 배포
8	가상키보드	–	–
9	푸시(PUSH)	–	–

본 가이드 제2장 제3절의 “루팅 및 탈옥 기기에서의 정상 동작” 보안취약점은 “모바일 대국민 보안공통기반”에서 제공하는 “단말/앱 위·변조 방지” 기능을 활용하여 모바일 단말 플랫폼 변조(루팅 및 탈옥) 및 서비스(앱)에 대한 무결성 여부를 점검할 수 있다.
또한, 제2장 제4절의 “중요정보의 평문 저장 및 전송” 및 “기타 중요정보의 평문 저장 및 전송” 등 보안취약점은 “모바일 대국민 보안공통기반”에서 제공하는 “E2E 암호화” 기능을 활용하여 전송되는 중요정보 및 기타 중요정보의 인증 정보를 암호화하여 전송할 수 있다.
앱 개발자는 해당 보안공통기반의 제공 기능 관련 자세한 사항은 “모바일 대민서비스 구축 가이드”를 참고한다.

부 록

부록 1 모바일 보안취약점 점검 도구

번호	점검 도구	설명 / 다운로드 URL
1	adb	Android 운영체제 모바일 기기를 PC에서 다양하게 제어할 수 있도록 해주는 프로그램으로, 특히 모바일 기기의 쉘을 PC에서 제어할 수 있다는 점이 가장 뛰어난 장점 https://developer.android.com/studio/releases/platform-tools
2	apktool	내부 코드들에 대한 악성 행위 여부를 검사할 수 있는 패키지 분석 도구 중 하나로, apk 파일의 압축을 해제하기 위해 사용 https://ibotpeaches.github.io/Apktool/
3	BurpSuite	취약점 분석용 proxy 프로그램으로 자주 사용되는 도구이며, BurpSuite를 실행시키기 위해서는 JVM(Java Vritual Machine) 설치가 필요 https://portswigger.net/burp/download.html
5	dex2jar	apk 파일의 압축을 해제했을 때 생성되는 .dex 파일을 jar 파일로 변환해주는 도구 https://sourceforge.net/projects/dex2jar/
6	JD-GUI	jar 파일 안에 들어있는 .class 파일은 jd-gui와 같은 프로그램을 이용하면 더욱 가독성 높은 결과물을 얻을 수 있음 http://java-decompiler.github.io/
7	putty	가장 널리 사용되는 ssh 접속 프로그램으로, 직관적인 인터페이스로 사용이 간편하고 freeware라는 장점을 가지고 있음 https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html
8	ProGuard	자바 클래스 파일을 난독화하여 Android 앱의 역공학을 어렵게 만드는 도구로 식별자를 변환하는 기법을 적용할 수 있음 https://developer.android.com/studio/build/shrink-code?hl=ko
9	SQLite	서버가 없이 구축이 가능한 SQL 데이터베이스 엔진으로 모바일 앱(안드로이드)에서 DB 분석을 위해 사용 https://www.sqlite.org/download.html
10	WireShark	널리 사용되는 네트워크 분석 프로그램으로 네트워크상에서 캡쳐한 데이터에 대한 정보를 제공하는 도구 https://www.wireshark.org/download.html
11	Xcode	애플 iOS 운영체제 기반의 앱을 개발하고 분석할 수 있는 개발도구 https://developer.apple.com/kr/xcode/resources/

부록 2 안드로이드 접근권한 목록(Permission List)

번호	분류	접근권한 명	접근권한 설명
1	계정	ACCOUNT_MANAGER	계정 관리 권한
2		GET_ACCOUNTS	어카운트 획득 권한
3		GET_ACCOUNTS_PRIVILEGED	어카운트 획득 권한
4	기기 및 앱 기록	READ_LOGS	로그 읽어오기 권한
5	네트워크 통신	ACCESS_NETWORK_STATE	네트워크 상태 접근 권한
6		CHANGE_NETWORK_STATE	통신 상태 변경 권한
7		BLUETOOTH	블루투스 권한
8		BLUETOOTH_ADMIN	블루투스 어드민 권한
9		BLUETOOTH_ADVERTISE	가까운 블루투스 장치 광고 권한
10		BLUETOOTH_CONNECT	페이링된 블루투스 장치 연결 권한
11		BLUETOOTH_PRIVILEGED	사용자 작업 없이 블루투스 장치 페이링 연결 권한
12		BLUETOOTH_SCAN	가까운 블루투스 장치 검색
13		INTERNET	인터넷 권한
14		NFC	NFC 수행 권한
15		NFC_PREFERRED_PAYMENT_INFO	NFC 결제 권한
16		NFC_TRANSACTION_EVENT	NFC 트랜잭션 이벤트 권한
17	문자	RECEIVE_MMS	수신 권한 권한
18		BROADCAST_SMS	SMS에 대한 브로드캐스트 권한
19		RECEIVE_SMS	SMS 수신 권한
20		SEND_SMS	SMS 송신 권한
21		READ_SMS	SMS 읽기 권한
22	배터리소모	VIBRATE	진동 권한
23	시스템 도구	ACCESS_WIFI_STATE	WiFi 상태 접근 권한
24		CHANGE_WIFI_STATE	WiFi 상태 변경 권한
25		CHANGE_WIFI_MULTICAST_ST	WiFi 멀티캐스트 상태 변경 권한

번호	분류	접근권한 명	접근권한 설명
		ATE	
26		READ_SYNC_SETTINGS	동기설정 읽어오기 권한
27		READ_SYNC_STATS	동기상태 읽어오기 권한
28		WRITE_SYNC_SETTINGS	동기설정 쓰기 권한
29	오디오/ 녹음	MODIFY_AUDIO_SETTINGS	오디오설정 편집 권한
30		RECORD_AUDIO	오디오 녹음 권한
31		CAPTURE_AUDIO_OUTPUT	오디오 출력 캡쳐 권한
32		ADD_VOICEMAIL	음성 메일 권한
33	웨어러블센서	BODY_SENSORS	바디 센서 접근 권한
34		ACTIVITY_RECOGNITION	신체 활동 인식
35	위치	ACCESS_FINE_LOCATION	정확한 위치 권한(GPS)
36		CONTROL_LOCATION_UPDATES	위치정보 갱신 권한
37		ACCESS_COARSE_LOCATION	대략적인 위치 권한
38		ACCESS_LOCATION_EXTRA_COMMANDS	로케이션 옵션 커맨드 액세스 권한
39		ACCESS_BACKGROUND_LOCATION	앱이 백그라운드에서 위치 허용 권한
40		ACCESS_MEDIA_LOCATION	사용자 공유 컬렉션에 유지되는 위치 권한
41		INSTALL_LOCATION_PROVIDER	위치 제공자를 위치 관리자에 설치 권한
42		LOCATION_HARDWARE	하드웨어 위치 기능 권한
43	주소록	READ_CONTACTS	주소록 읽어오기 권한
44		WRITE_CONTACTS	주소록 쓰기 권한
45	카메라	CAMERA	카메라 권한
46	캘린더	READ_CALENDAR	캘린더 읽어오기 권한
47		WRITE_CALENDAR	캘린더 쓰기 권한
48	통화	PROCESS_OUTGOING_CALLS	전화 발신처리 접근 권한
49		CALL_PHONE	통화 권한
50		CALL_PRIVILEGED	통화(긴급전화포함) 권한

번호	분류	접근권한 명	접근권한 설명
51		READ_CALL_LOG	통화 로그 읽기 권한
52		WRITE_CALL_LOG	통화 로그 쓰기 권한
53		ANSWER_PHONE_CALLS	앱이 수신 전화에 허용 권한
54		CALL_COMPANION_APP	InCallService API 사용 시 호출 도우미 활성화 권한
55		MANAGE_ONGOING_CALLS	통화 세부 정보를 쿼리하고 진행 중인 통화를 관리 권한
56		MANAGE_OWN_CALLS	ConnectionService API로 자체 호출 권한
57		MODIFY_PHONE_STATE	통화상태 편집 권한
58		READ_PHONE_STATE	통화상태 읽어오기
59		READ_PRECISE_PHONE_STATE	통화상태 읽기 전용 권한
60		READ_PHONE_NUMBERS	장치 전화번호 읽기 권한
61		ACCEPT_HANDOVER	통화 앱이 다른 앱에서 계속 통화하는 권한
62	파일저장	MOUNT_FORMAT_FILESYSTEMS	외부저장소 포맷 권한
63		MOUNT_UNMOUNT_FILESYSTEMS	외부저장소 언마운트 권한
64		READ_EXTERNAL_STORAGE	외부저장소 읽기 권한
65		WRITE_EXTERNAL_STORAGE	외부저장소 쓰기 권한
66		MANAGE_EXTERNAL_STORAGE	외부저장소 관리 권한
67	시스템 권한	ACCESS_CHECKIN_PROPERTIES	체크인 데이터베이스의 속성테이블로 액세스 권한
68		ACCESS_BLOBS_ACROSS_USERS	사용자간 데이터 Blob 권한
69		ACCESS_NOTIFICATION_POLICY	알림 정책 권한
70		BATTERY_STATS	배터리 상태 권한
71		REQUEST_IGNORE_BATTERY_	Settings

번호	분류	접근권한 명	접근권한 설명
		OPTIMIZATIONS	ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS 사용 권한
72		BIND_ACCESSIBILITY_SERVICE	시스템 바인딩 AccessibilityService 필요 권한
73		BIND_APPWIDGET	AppWidget 필요 권한
74		BIND_AUTOFILL_SERVICE	AutofillService 필요 권한
75		BIND_CALL_REDIRECTION_SERVICE	CallRedirectionService 필요 권한
76		BIND_CARRIER_MESSAGING_CLIENT_SERVICE	CarrierMessagingClientService 하위 클래스 보호 권한
77		BIND_CARRIER_MESSAGING_SERVICE	CarrierMessagingClientService 하위 클래스 보호 권한(API 23 이상 지원하지 않음)
78		BIND_CARRIER_SERVICES	이동통신사 앱 서비스 바인딩 권한
79		BIND_CHOOSER_TARGET_SERVICE	(API 30 이상 지원하지 않음)
80		BIND_COMPANION_DEVICE_SERVICE	CompanionDeviceServices 필요 권한
81		BIND_CONDITION_PROVIDER_SERVICE	ConditionProviderService 필요 권한
82		BIND_CONTROLS	타사 제어 요청 권한
83		BIND_DEVICE_ADMIN	장치 관리 수신기 권한
84		BIND_DREAM_SERVICE	DreamService
85		BIND_INCALL_SERVICE	InCallService 필요 권한
86		BIND_INPUT_METHOD	InputMethodService 필요 권한
87		BIND_MIDI_DEVICE_SERVICE	MidiDeviceService 필요 권한
88		BIND_NFC_SERVICE	HostApduService 또는 OffHostApduService 필요 권한
89		BIND_NOTIFICATION_LISTENER_SERVICE	NotificationListenerService 필요 권한
90		BIND_PRINT_SERVICE	PrintService 필요 권한

번호	분류	접근권한 명	접근권한 설명
91		BIND_QUICK_ACCESS_WALLET_SERVICE	QuickAccessWalletService 필요 권한
92		BIND_QUICK_SETTINGS_TILE	빠른 설정 필요 권한
93		BIND_REMOTEVIEWS	RemoteViewsService 필요 권한
94		BIND_SCREENING_SERVICE	CallScreeningService 필요 권한
95		BIND_TELECOM_CONNECTION_SERVICE	ConnectionService 필요 권한
96		BIND_TEXT_SERVICE	TextService 필요 권한
97		BIND_TV_INPUT	TvInputService 필요 권한
98		BIND_VISUAL_VOICEMAIL_SERVICE	VisualVoicemailService 필요 권한
99		BIND_VOICE_INTERACTION	VoiceInteractionService 필요 권한
100		BIND_VPN_SERVICE	VpnService 필요 권한
101		BIND_VR_LISTENER_SERVICE	VrListenerService 필요 권한
102		BIND_WALLPAPER	WalterService 필요 권한
103		BROADCAST_PACKAGE_REMOVED	제거 권한된 패키지에 대한 브로드캐스트 권한
104		BROADCAST_STICKY	인텐트브로드 캐스트 권한
105		BROADCAST_WAP_PUSH	WAP PUSH 수신 권한
106		CHANGE_COMPONENT_ENABLED_STATE	컴포넌트의 실효성 변경 권한
107		CHANGE_CONFIGURATION	컨피그 변경 권한
108		CLEAR_APP_CACHE	어플리케이션 캐시 클리어 권한
109		DELETE_CACHE_FILES	캐시파일 제거 권한
110		DELETE_PACKAGES	패키지 제거 권한
111		REQUEST_DELETE_PACKAGES	패키지 제거 요청 권한
112		DIAGNOSTIC	진단 리소스 읽고 쓰기 권한
113		DISABLE_KEYGUARD	키가드 끄기 권한
114		DUMP	캐시 덤프 권한
115		EXPAND_STATUS_BAR	상태 표시줄 확장 권한

번호	분류	접근권한 명	접근권한 설명
116		STATUS_BAR	상태 표시줄 지정 권한
117		FACTORY_TEST	팩토리 테스트 권한
118		FOREGROUND_SERVICE	ServicestartForeground 필요 권한
119		GET_PACKAGE_SIZE	패키지 획득 권한
120		GET_TASKS	태스크 획득 권한(API 21부터 지원하지 않음)
121		GLOBAL_SEARCH	글로벌 검색 필요 권한
122		HIDE_OVERLAY_WINDOWS	시스템 오버레이 필요 권한
123		HIGH_SAMPLING_RATE_SENSORS	센서 데이터 속도 권한
124		INSTALL_PACKAGES	패키지 인스톨 권한
125		INSTALL_SHORTCUT	앱 런처 바로가기 권한
126		INSTANT_APP_FOREGROUND_SERVICE	빠른 실행 앱이 포그라운드 서비스 사용 권한
127		INTERACT_ACROSS_PROFILES	동일 프로필 그룹간 허용 권한
128		KILL_BACKGROUND_PROCESSES	백그라운드 프로세스 중지 권한
129		LAUNCH_MULTI_PANE_SETTINGS_DEEP_LINK	활동 표시 권한
130		LOADER_USAGE_STATS	엑세스 로그 읽기 권한
131		REQUEST_INSTALL_PACKAGES	패키지 인스톨 요청 권한
132		MASTER_CLEAR	마스터 클리어 권한
133		MANAGE_DOCUMENTS	문서 관리 권한
134		MANAGE_MEDIA	미디어 관리 권한
135		MEDIA_CONTENT_CONTROL	콘텐츠 관리 권한
136		PERSISTENT_ACTIVITY	액티비티 지속 권한(API 15 이상 지원하지 않음)
137		PACKAGE_USAGE_STATS	패키지 사용 통계 권한
138		QUERY_ALL_PACKAGES	기기의 모든 일반 쿼리 권한
139		READ_INPUT_STATE	입력상태 읽어오기 권한(API 16

번호	분류	접근권한 명	접근권한 설명
			이상 지원하지 않음)
140		READ_VOICEMAIL	음성 메일 읽기 권한
141		REBOOT	리부트 권한
142		RECEIVE_BOOT_COMPLETED	boot 완료 권한
143		RECEIVE_WAP_PUSH	WAP PUSH 메시지 수신 권한
144		REORDER_TASKS	태스크 리오더 권한
145		RESTART_PACKAGES	패키지 리스타트 권한(API 15 이상 지원하지 않음)
146		REQUEST_COMPANION_PROFILE_WATCH	시계 장치 연결 권한
147		REQUEST_COMPANION_RUN_IN_BACKGROUND	Companion 앱이 백그라운드 허용 권한
148		REQUEST_COMPANION_START_FOREGROUND_SERVICES_FROM_BACKGROUND	Companion 앱이 백그라운드에서 포그라운드 허용 권한
149		REQUEST_COMPANION_USE_DATA_IN_BACKGROUND	Companion 앱이 백그라운드 데이터 사용 권한
150		REQUEST_OBSERVE_COMPANION_DEVICE_PRESENCE	Companion 앱 알림 권한
151		REQUEST_PASSWORD_COMPLEXITY	화면 잠금 복잡성 요구 권한
152		SCHEDULE_EXACT_ALARM	알람 API 사용 권한
153		SEND_RESPOND_VIA_MESSAGE	통화 시 다른 음용프로그램 요청 권한
154		SET_ALARM	알람 설정 권한
155		SET_ALWAYS_FINISH	액티비티 전체 종료 권한
156		SET_ANIMATION_SCALE	스케일 애니메이션 지정 권한
157		SET_DEBUG_APP	디버그 어플리케이션 지정 권한
158		SET_PREFERRED_APPLICATIONS	자주 사용하는 어플리케이션 지정 권한(API 15 이상 지원하지 않음)
159		SET_PROCESS_LIMIT	제한처리 지정 권한
160		SET_TIME	시간 설정 권한

번호	분류	접근권한 명	접근권한 설명
161		SET_TIME_ZONE	타임존 지정 권한
162		SET_WALLPAPER	배경화면 지정 권한
163		SET_WALLPAPER_HINTS	배경화면 힌트 지정 권한
164		SIGNAL_PERSISTENT_PROCESSES	영구 프로세스 신호 요청 권한
165		START_FOREGROUND_SERVICES_FROM_BACKGROUND	백그라운드에서 포그라운드 시작 권한
166		START_VIEW_PERMISSION_USAGE	앱의 권한 사용 화면 권한
167		SYSTEM_ALERT_WINDOW	알림 윈도우 권한
168		TRANSMIT_IR	IR 송신기 권한
169		UPDATE_DEVICE_STATS	기기 통계 업데이트 권한
170		UPDATE_PACKAGES_WITHOUT_USER_ACTION	앱 업데이트에 사용자 작업 권한
171		USE_FINGERPRINT	지문 사용 권한(API 28 이상 지원하지 않음)
172		USE_FULL_SCREEN_INTENT	알림 전체 화면 인텐트 권한
173		USE_ICC_AUTH_WITH_DEVICE_IDENTIFIER	ICC 기반 인증 권한
174		USE_SIP	SIP 서비스 권한
175		UWB_RANGING	초광대역 지정 권한
176		USE_BIOMETRIC	생체인식 사용 권한
177		WAKE_LOCK	알람 권한
178		WRITE_APN_SETTINGS	APN 설정쓰기 권한
179		WRITE_GSERVICES	G서비스 쓰기 권한
180		WRITE_SETTINGS	설정 쓰기 권한
181		WRITE_SECURE_SETTINGS	보안 시스템 쓰기 권한
182		WRITE_VOICEMAIL	음성메일 쓰기 권한

※ 안드로이드 개발자 홈페이지(http://developer.android.com/reference/android/Manifest.permission.html) 참조

참고 1 기능보안취약점 진단 환경 구축

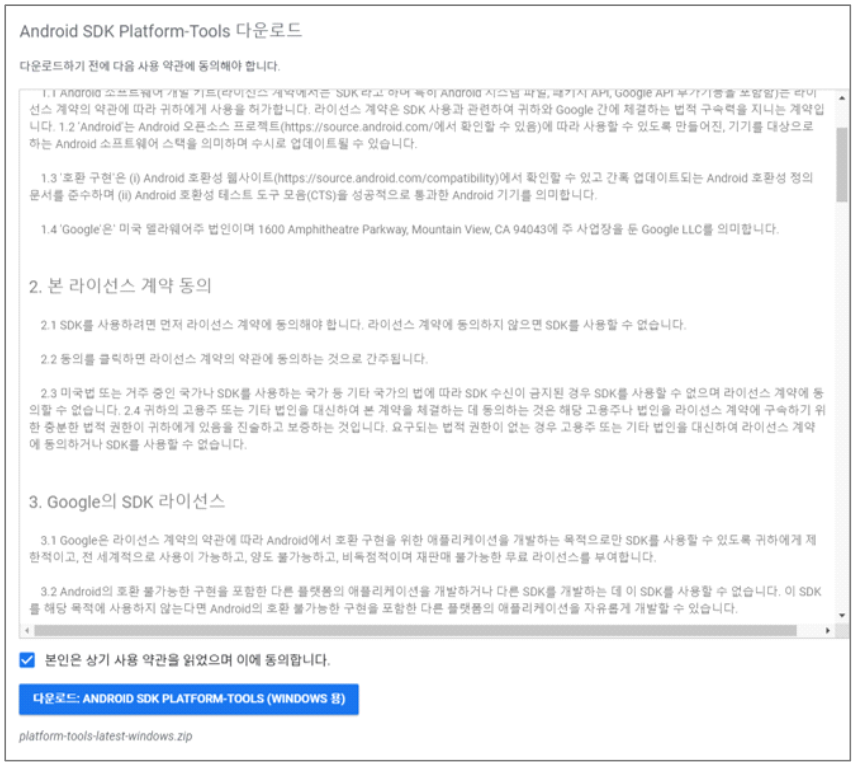
1. adb 연결

가. 설명

- Android ADB (Android Debug Bridge)는 PC와 스마트 폰 간에 통신을 할 수 있는 명령어도 도구이다. 안드로이드 개발자에게는 apk 설치, log 출력 등에서 adb를 사용한다.

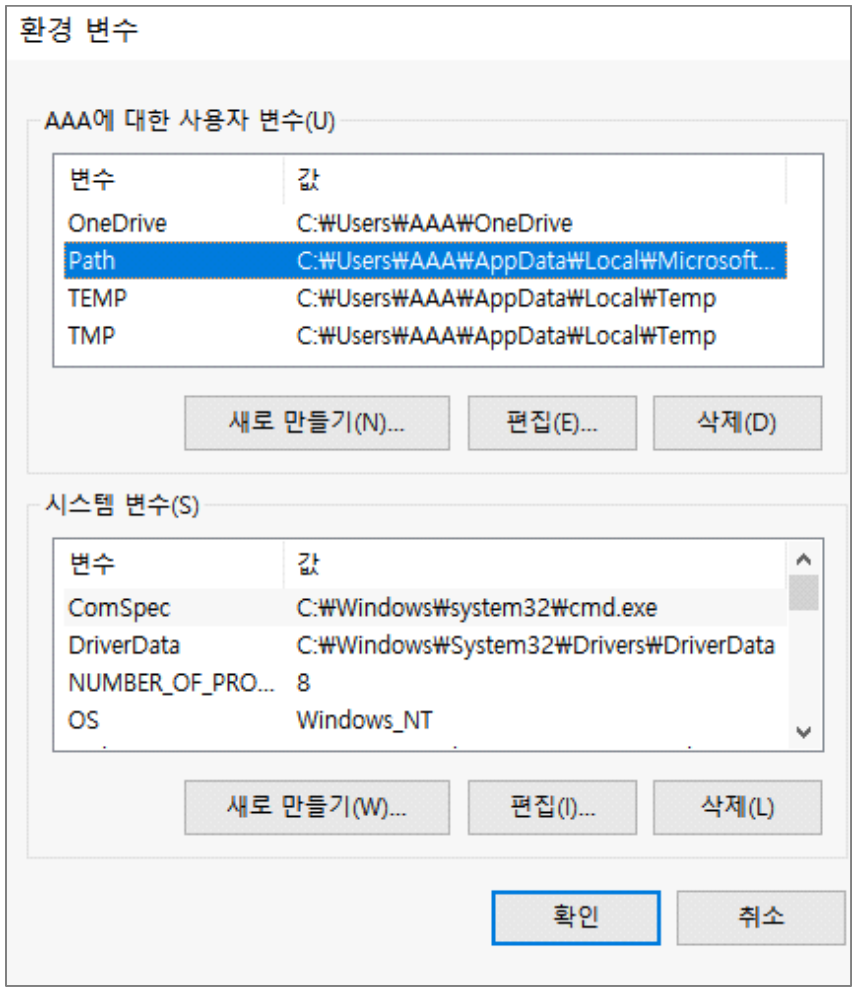
나. 환경 구축

- 단말기 제조사 USB 드라이버를 설치 (각 제조사 홈페이지에서 USB 통합 Driver를 제공) - 필수
- PC에서 ADB Tool 다운로드: 최신 버전 Android SdK platform tool 설치
  ※ https://developer.android.com/studio/releases/platform-tools?hl=ko



〈 Androd SDK Platform-Tool 다운로드 〉

- PC 환경 변수에 ADB 설치 Path 추가
  ※ 제어판-시스템 및 보안-시스템-고급 시스템 설정 - 시스템 속성 - 고급 - 환경 변수 선택
  ※ path 항목에 platform-tools 설치 경로 추가

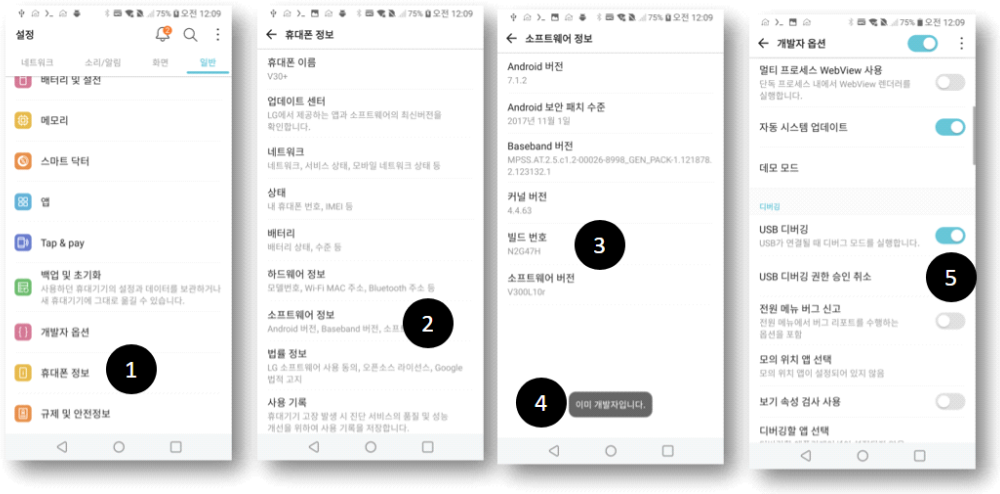


〈 환경변수 설정 〉

다. ADB 디바이스 USB 연결

- 단말기의 개발자 메뉴에서 'USB 디버깅' 연결 활성화
  ※ 개발자 메뉴 활성화: 시스템 세팅 - 휴대폰 정보 - 소프트웨어 정보- 빌드 번호를 3번 누르기

※ 개발자 메뉴에서 "USB 디버깅 옵션" 활성화



〈 스마트폰에서 USB 디버깅 연결 활성화 〉

- USB 연결 후 adb 정상 동작 확인
  ※ 명령 프롬프트 창에서 $ adb devices 입력

```
C:₩Users₩Administrator>adb devices
List of devices attached
05157df51be5111f        device
```

〈 adb devices 명령어 실행 〉

라. Wi-Fi를 이용한 디바이스 연결

- 먼저 단말기와 연결할 PC를 동일한 Wi-Fi에 접속한다.
- USB 케이블로 단말기와 PC를 연결한다.
- "adb tcpip 포트번호" 명령어로 단말기가 TCP/IP 연결을 수신대기 할 수 있도록 설정한다.

```
C:₩>adb tcpip 5555
restarting in TCP mode port: 5555
```

〈 adb tcpip 명령어 실행 〉

- 단말기와 PC에 연결된 USB 케이블을 제거한다.
- 단말기에서 Wi-Fi IP 주소를 확인한다.

※ 단말기의 설정 - 연결 - Wi-Fi - 현재 연결되어 있는 Wi-Fi IP 주소 확인

〈 단말기 내 IP 주소 확인 〉

- 확인한 IP주소를 이용하여 adb 연결 수행
  ※ "adb connect 단말기 IP 주소" 명령어 실행

```
C:₩>adb connect 192.168.219.100:5555
connected to 192.168.219.100:5555
```

〈 adb connect 명령어 실행 〉

- 정상적인 연결 확인
  ※ "adb devices" 명령어 실행

```
C:₩>adb devices
List of devices attached
192.168.219.100:5555    device
```

〈 adb devices 명령어 실행 〉

2. 단말기 프록시 환경 설정

가. 설명

- 프록시 서버는 클라이언트와 서버 사이에서 데이터를 중계하는 역할을 하는 서버를 말한다. 모바일 앱 취약점 진단시 단말기의 통신 과정을 보다 쉽게 확인하기 위해 단말기 내 프록시 환경 설정을 설명한다.

나. 안드로이드 환경 설정

- 환경설정 → Wi-Fi 활성화 → 네트워크 설정 → 프록시 수동 설정
- 프록시 설정

  ※ 프록시 호스트 이름 : Wi-Fi에 연결된 PC 아이피주소

  ※ 프록시 포트 : 프록시 툴의 포트번호

〈 안드로이드 프록시 설정 〉

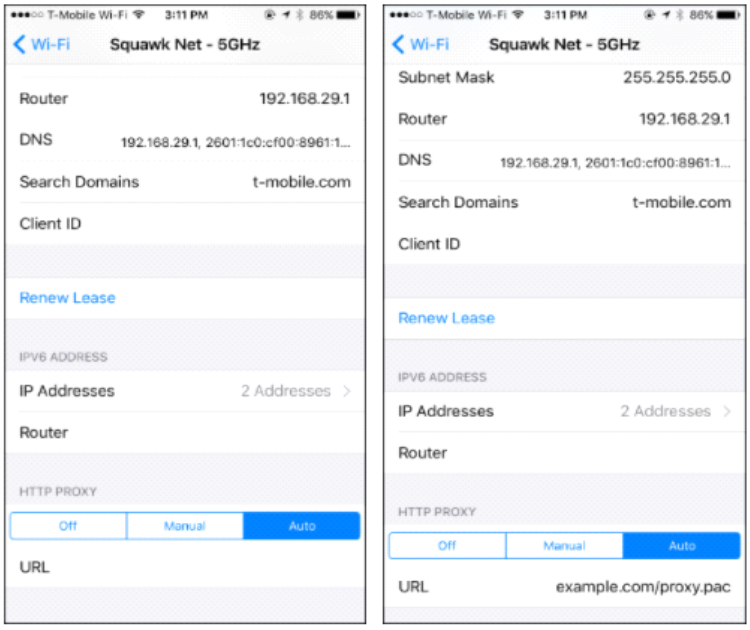
- 프록시 인증서 설정

  ※ 주소창에 프록시 아이피:포트(ex:192.168.43.27:8888)를 입력하면 Fiddler Echo Service 문구가 나타난다. 하단의 FiddlerRoot certificate 링크 클릭하여 인증서를 가상 단말기에 설치한다.

〈 프록시 툴 인증서 설정 〉

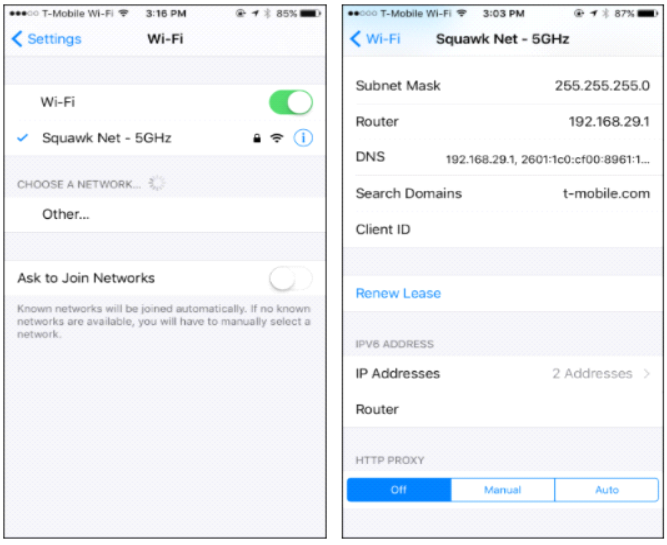
다. 아이폰 환경 설정

- 환경설정 → Wi-Fi 활성화 → 프록시 설정 액세스 → Wi-Fi 네트워크 선택 → HTTP 프록시 옵션 자동 선택



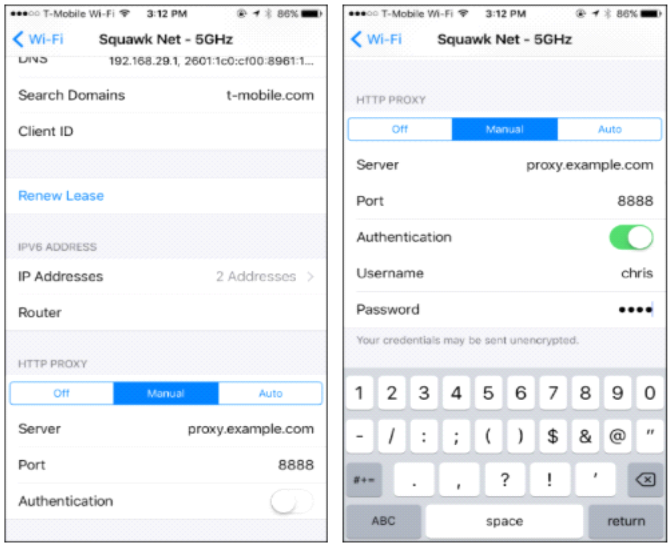
〈 아이폰 프록시 환경 설정 〉

- 자동 프록시 구성 스크립트 사용
  ※ HTTP PROXY에서 AUTO 선택
  ※ URL 상자에 프록시 자동 구성 스크립트 주소 입력



〈 아이폰 프록시 자동 구성 스크립트 환경 설정 〉

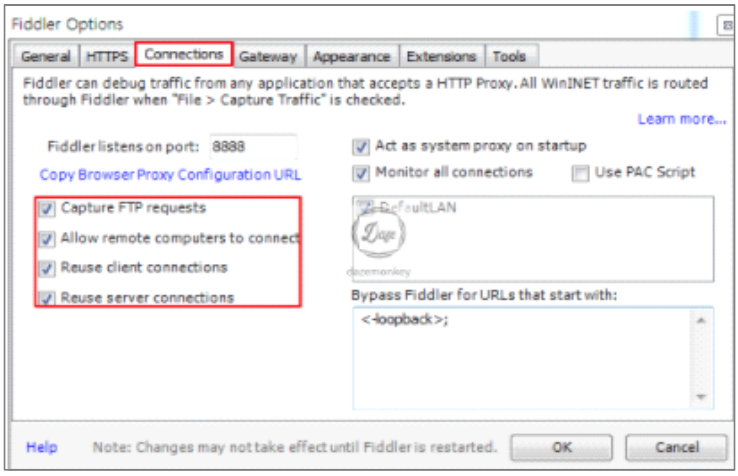
- 프록시 서버의 주소와 포트를 수동 지정
  ※ Manual을 선택하여 프록시 서버의 주소와 포트번호를 입력
  ※ 사용자 이름과 비밀번호가 필요한 경우 인증 옵션을 활성화
  ※ 사용자 이름 및 암호를 입력



〈 아이폰 프록시 수동 지정 환경 설정 〉

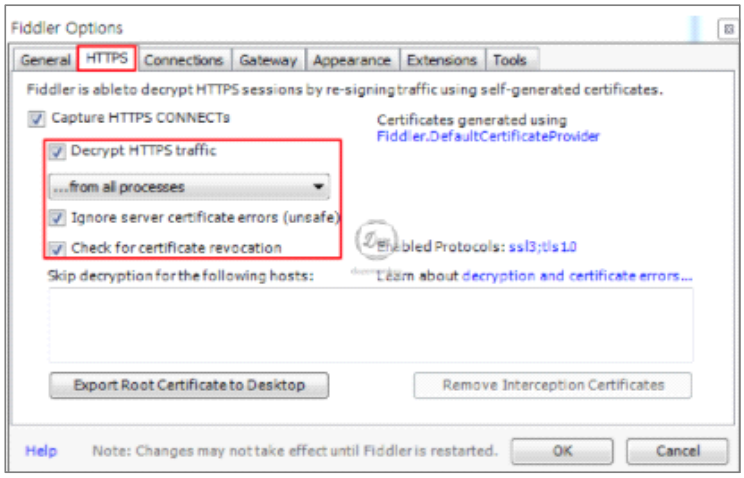
라. 프록시 툴 환경 구성

- 상단의 Tools - Option - Connections 탭의 4개 항목 체크
  - ※ Capture FTP requests
  - ※ Allow remote computers to connect
  - ※ Reuse client connections
  - ※ Reuse server connections



〈 Fiddler connection 환경 설정 〉

- HTTPS 탭의 3개 항목 체크
  - ※ Decrypt HTTPS traffic
  - ※ Ignore server certificate errors (unsafe)
  - ※ Check for certificate revocation



〈 Fiddler HTTPS 환경 설정 〉

3. 앱 플레이어 진단 환경 설정(Android)

가. 설명

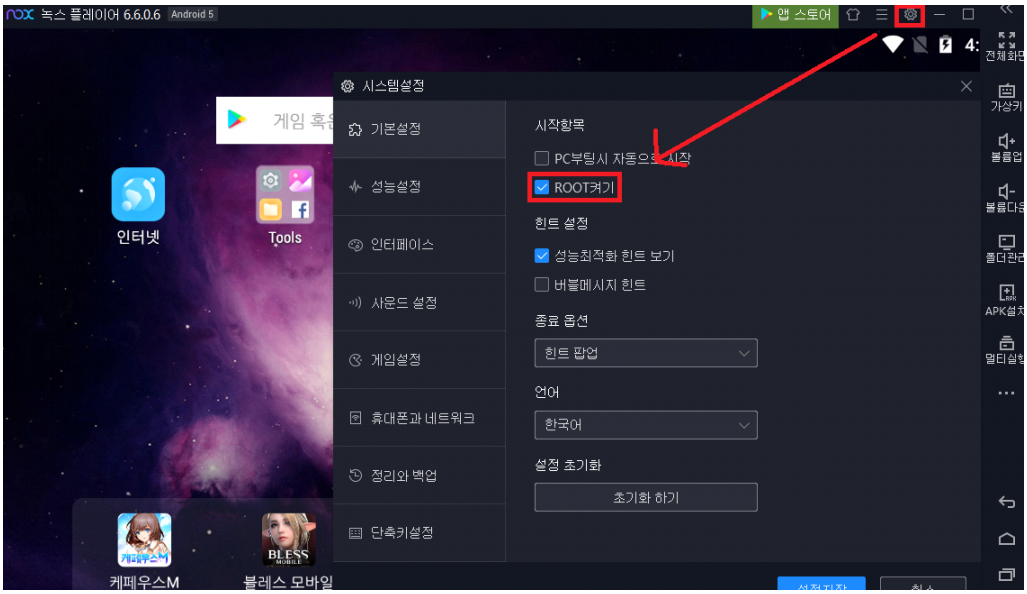
- 앱 플레이어는 PC에서 Android 애플리케이션을 실행할 수 있는 가상 앱 플레이어 프로그램이다. PC에서 대부분 모바일 게임을 실행하기 위해 만들어졌지만, 취약점 진단을 위해 많이 활용된다. 모바일 기기가 없거나 루팅된 단말기가 없는 경우 활용도가 높다. 앱 플레이어는 NOX, 지니모션 등이 존재한다.

나. 설치 환경

- 앱 플레이어 설치
  ※ NOX 앱 플레이어 다운로드 (https://kr.bignox.com/)
- ADB 환경 구축
  ※ 앱 플레이어 내 ADB 접속 툴 이용 가능

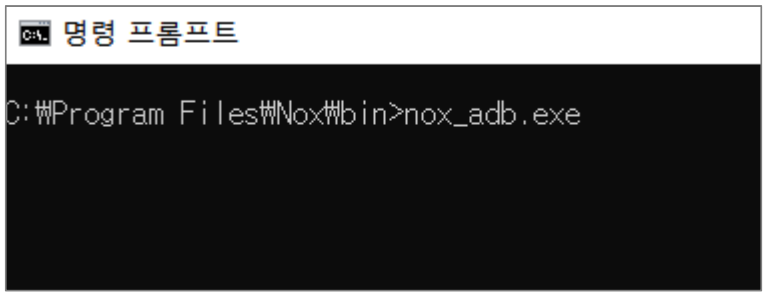
다. 앱 플레이어 환경 설정

- 안드로이드 내부 접속을 위한 Root 권한 획득
  ※ 설정 - 기본설정 - ROOT 켜기



〈 앱 플레이어 ROOT 환경 설정 〉

- 앱 플레이어 내 adb 연결
  ※ C:\Program Files (x86)\Nox\bin 내 nox_adb.exe 실행



〈 앱 플레이어 ROOT 환경 설정 〉

모바일 대민서비스 보안취약점 점검 가이드

인 쇄 2021년 12월
발 행 2021년 12월

발행처 행정안전부
세종특별자치시 한누리대로 411(어진동)
Tel : 02-2100-3399

한국인터넷진흥원
전라남도 나주시 진흥길 9
Tel : 061-820-2748

디자인/인쇄_예원디자인 www.yewondnp.com

비 매 품

모바일 대민서비스 보안취약점 점검 가이드

행정안전부

KISA 한국인터넷진흥원