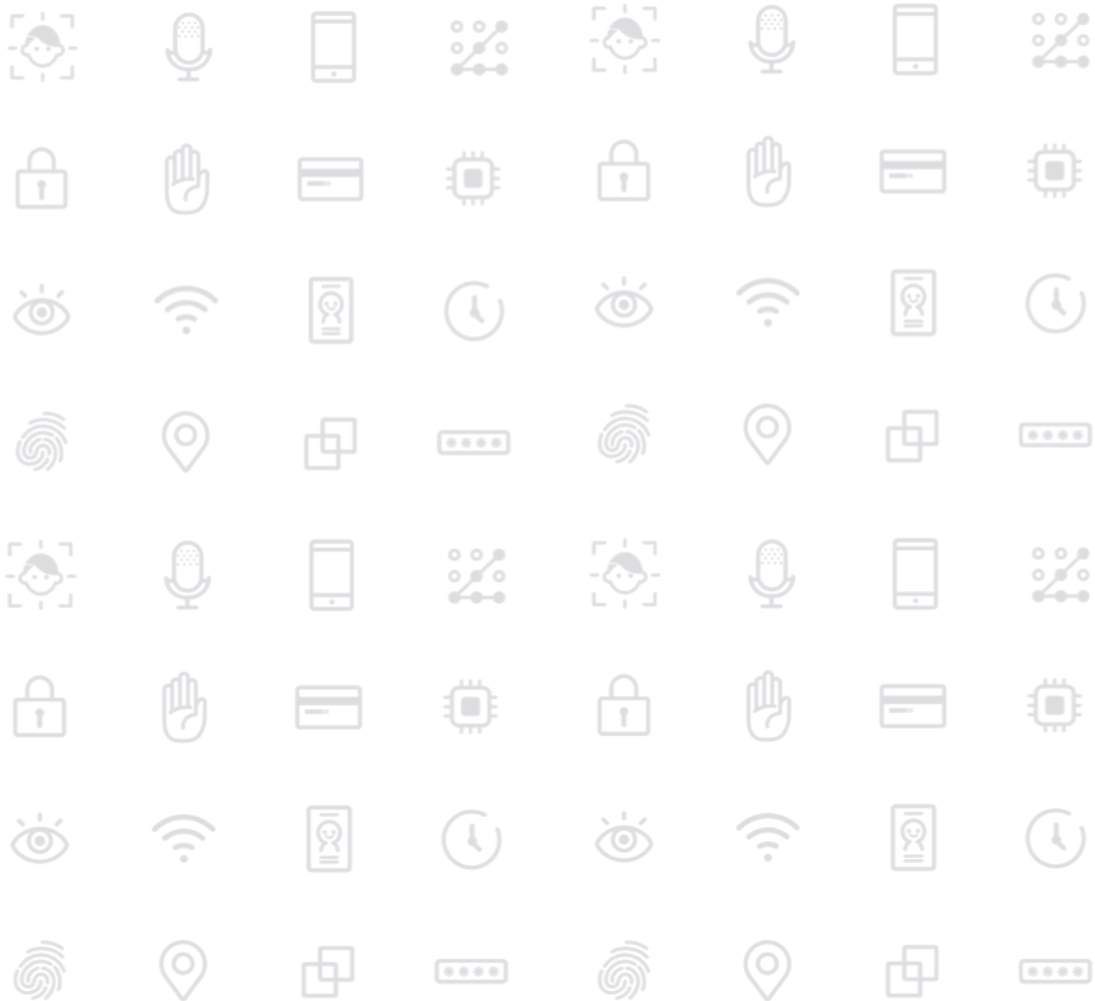


TOUCH-EN mVaccine CS v3.0

Android API 매뉴얼



본 문서의 저작권은 라온시큐어(주)에 있습니다.

전체 또는 일부를 무단 사용, 무단 복제 및 무단 배포하는 행위는 법적 제재를 받을 수 있습니다.

TOUCH-EN mVaccine

Copyright RaonSecure All rights reserved.

주 소: 서울특별시 영등포구 여의대로 108 (여의도동, 파크원 타워2 48F)

대표전화: 02-561-4545

홈페이지: www.raoncorp.com

개정 내역

문서버전	모듈버전	기본/수정사항	승인일자	변경자
V3.3.20	V3.9.8.2316	targetSDK 34 대응 broadcast receiver 내보내기 동작 지정 가이드 '3.1.4.2. 권한 등록' 오타 수정, POST_NOTIFICATIONS 권한 추가 '3.7. targetSdkVersion 34 이상' 가이드 추가 '1.17. Android resource linking failed 컴파일 에러 발생' FAQ 추가 '3.2.2.3. 컴포넌트 등록' 추가 동기식 루팅 검사 deprecated 처리 지원사양 변경 Android SDK 4.0.3 -> 6.0 제품구성 일본어 리소스 추가	2024-01-15	조영진
V3.3.19	v3.9.7.2316	'3.5.5. 탐지된 악성앱 리스트' 듀얼 엔진 타입 'malware_type =2' 추가. result code EXIST_VIRUS (=1001) 추가 3.5.7.4. 사용 예시 수정	2023-11-07	김민정, 조영진
V3.3.18	v3.9.4.2315	3.4.4. 백신 구동 이후 처리 END_INIT 추가 3.5.1. 실시간 검사 (ScanReceiver) 옵션 수정 Mini 모드 Thread 방식 API명 변경 (scan_mini -> scanMini)	2023-06-07	조영진
V3.3.17	v3.9.3.2314	AAB 배포 적용 절차 추가 적용 가이드 세분화 scan_package, backgroundScan, bg_rooting 옵션 default true 로 변경. mini 모드 Thread 방식 API 추가	2023-04-11	조영진
v3.3.16	v3.9.2.2313	AAR 적용 가이드 수정 및 추가 - 3.2.2.1. Application 속성 - 3.2.3. useLegacyPackaging 설정 3.1.6.2. gradle 설정 keepDebugSymbols 추가	2023-02-28	김민정 조영진
v3.3.15	-	오타 수정 및 문서 정돈	2023-01-30	박형준
v3.3.14	v3.9.1.2312	부록 FAQ 추가 - 1.16. 폭넓은 패키지(앱) 가시성 (QUERY_ALL_PACKAGES) 권한 심사가 거부되었을 경우 AAR 라이브러리 적용 방법 3.2.2.1. 권한 등록 추가 3.1.5. Proguard 난독화 예외 설정 최신화 3.4.3. 루팅 API 사용법 Rooting API 리턴 값 5 (앱 변조 시도가 감지됨) 추가	2023-01-25	김민정 조영진
v3.3.13	v3.8.9.2310	비동기 루팅 API 추가 부록 FAQ 추가	2022-11-11	조영진

		<ul style="list-style-type: none"> - '1.13. "Input dispatching timed out" 에러 발생' - '1.14. 32bit 단말에서만 악성앱 오탐' - '1.15. "SharedPreferences in credential encrypted storage are not available until after user is unlocked" 에러 발생' 		
v3.3.12	v.3.8.8.2310	getScanResult API 추가 '3.4.5. 악성코드, 보이스 피싱 분류 API'에서 '3.4.5. 탐지된 악성앱 리스트'로 변경 및 설명 추가	2022-09-14	조영진
v3.3.11	v.3.8.7.2310	Request Code POST_NOTI_NO 추가	2022-08-09	김민정
v3.3.10	v.3.8.7.2310	3.4.7 Notification 사용 설정 여부 확인 추가 [부록] 1.12 Android13 기기에서 Notification이 뜨지 않을 때 추가 [부록] 3.6. targetSdkVersion 33 이상 추가 useBlackAppCheck 옵션 설명 수정	2022-08-03	김민정 조영진
v3.3.9	v.3.8.3.2309	5.1. mini backgroundScan 옵션 내용 추가 [부록] 3.5. targetSdkVersion 31 이상 추가	2022-06-28	조영진
v3.3.8	v.3.8.2.2308	3.1.6. WorkManager 적용, 3.2.2. WorkManager 적용 가이드 추가 악성코드, 보이스 피싱 분류 API FULL모드 추가. 3.4.5.1.2. FULL모드 등록 가이드 추가. 3.4.5.1.1. MINI모드 등록 가이드 추가 및 수정. targetSDKVersion 31에서 AOS12 이상 ANR 발생 임시 대응 방안 제거 '3.1.4. AndroidManifest 설정' RemoveNotificationService 옵션 동작 및 가이드 변경	2022-04-07	조영진
v3.3.7	v.3.8.1.2308	mini, full, 실시간 검사 보이스 피싱 사용 옵션 추가 targetSDKVersion 31에서 AOS12 이상 ANR 발생 임시 대응 방안 작성	2022-02-24	조영진
v3.3.6	v.3.8.0.2308	부록 FAQ '1.11 앱 강제 종료 후 Notification이 사라지지 않을 때' 내용 수정 적용 가이드 '3.4.6. 백그라운드 앱 강제 종료 시 Notification 제거' 추가 RemoveNotificationService 옵션 추가	2022-02-15	조영진
v3.3.5	v.3.7.9.2307	'1.9 rooting_internal_check 옵션 2번 마운트검사 적용방법' 예외처리 방법 수정	2021-12-23	조영진
v3.3.4	v.3.7.8.2306	오타, 표기 오류, 패키지 폴더 등 수정	2021-12-03	장영재
v3.3.3		rootingexitapp 옵션 구현 가이드 추가 CommonUtil.isMvclsolService 옵션 추가 '앱 강제 종료 후 Notification이 사라지지 않을 때' FAQ 추가 rootingexitapp MINI, FULL, UIFULL 모드 동작 차이 가이드 추가	2021-11-29	조영진
v3.3.2	v.3.7.8.2305	악성코드, 보이스 피싱 분류 API	2021-09-29	조영진

v3.3.1	v3.7.7.2303	에뮬레이터 탐지 추가	2021-07-12	전용태
v3.3.0	v3.7.6.2302	구성 Library 수정 권한 추가	2021-01-25	전용태
v3.2.9	v3.7.4.2300	Flutter에서 적용방법 FAQ 추가	2020-11-03	전용태
v3.2.8	v3.7.3.2299	rooting_internal_check 소켓검사 제거	2020-09-28	전용태
v3.2.8	v3.7.3.2299	rooting_internal_check 마운트검사 추가	2020-09-08	전용태
v3.2.7	v3.7.1.2298	매뉴얼 리뉴얼	2020-08-27	전용태
v3.2.6	v3.7.0.2297	UI Full 스캔모드 추가 및 구성 요소 최신화	2020-08-04	전용태
v3.2.4	-	에러코드 수정 backgroundJobForLongTime / show_update 내용 추가	2020-06-05	전용태
v3.2.3	-	show_about 옵션 추가	2019-11-19	전용태
v3.2.2	v3.6.4.2293	Internal 체크용 CommonUtil.c() 추가	2019-11-19	전용태
v3.2.1	v3.6.3.2292	미지원 옵션 제거 (scan_heuristic)	2019-10-24	전용태
v3.2.1	v3.6.3.2292	show_badge 옵션 상세내용 추가	2019-09-25	전용태
v3.2.1	v3.6.3.2292	루팅API 반환 값 "2"에 처리 추가	2019-09-20	전용태
v3.2.1	v3.6.3.2292	권한 추가 (USE_FULL_SCREEN_INTENT) 서비스 추가 (DetectionResultSendService) 서비스 제거 (ScanService)	2019-09-16	전용태
v3.2.0	v3.6.2.2292	AndroidManifest Style 적용 수정	2019-08-14	전용태
v3.2.0	v3.6.2.2292	권한 등록 내용 추가	2019-08-08	전용태
v3.1.9	v3.6.2.2292	rooting_delay_time 옵션 추가	2019-07-01	전용태
v3.1.9	v3.6.2.2292	CodeReceiver 동적등록 사용방법 추가	2019-07-01	전용태
v3.1.8	v3.6.2.2292	무결성 검증 옵션 추가된 루팅 유틸 추가	2019-06-05	전용태
v3.1.7	-	64bit library 연동내용 추가	2019-06-05	전용태
v3.1.6	v3.6.1.2291	v_so 옵션 추가	2019-05-28	전용태
v3.1.5	v3.6.0.2290	Internal 루팅 검사 추가, 투명 스타일 추가	2019-04-22	전용태
v3.1.4	-	옵션 기본값 수정	2019-04-04	전용태
v3.1.6	-	에러 발생시 대응 방법 추가 (엔진 무결성 실패)	2019-03-08	전용태
v3.1.2	-	bg_rooting 옵션추가	2019-03-07	전용태
v3.1.1	-	targetSdkversion 28 이상 빌드 시, 패키지 삭제 권한 추가	2019-02-28	전용태
v3.1.0	-	API 추가 (show_badge)	2019-02-11	전용태
v3.0.9	-	Rooting API 내용 변경	2018-10-23	이남열
v3.0.8	-	AndroidManifest.xml 내용 변경	2018-07-12	전용태
v3.0.7	-	백그라운드 구동 탐지 설정(실시간검사) 내용 변경 백신구동 이후 처리 내용 추가	2018-05-24	이남열
v3.0.6	-	ScanReceiver 적용사항변경, CodeReceiver 부분 삭제	2018-02-01	이남열
v3.0.5	-	라이브러리 적용(libs) 변경	2017-12-06	이남열
v3.0.4	-	onInstallService 변경	2016-07-07	이남열
v3.0.3	-	난독화(Proguard) 적용 예외클래스 추가	2015-12-31	김연효

v3.0.2	-	루팅 스레드 API 제거 루팅 API 앱 명 출력 추가	2015-10-29	김연효
v3.0.1	-	루팅 결과 제외사항 반영 프로가드 난독화 예외처리 수정	2015-08-24	김연효
v3.0.0	-	정기릴리즈 수정사항 반영	2015-07-31	김연효
v2.0.0	-	정기릴리즈 수정사항 반영	2015-04-28	김연효
v1.0.3	-	루팅탐지 개선 추가	2015-01-06	김연효
v1.0.2	-	신규 릴리즈 추가/수정/개선 반영	2014-11-17	김연효
v1.0.1	-	샘플프로젝트 개선 반영	2014-08-01	김연효
v1.0.0	-	최초 작성	2014-02-10	김연효

목차

1. 개요	9
1.1. 제품 소개	9
1.2. 목적	9
1.3. 사전 준비사항	9
1.4. 지원 사양	9
1.5. 용어 정리	9
2. 제품 구성	10
2.1. 제품 구성	10
2.2. 제품 흐름도(Flow)	10
2.2.1. FULL 모드 흐름도	10
2.2.2. MINI 모드 흐름도	11
2.2.3. 패턴 업데이트 흐름도	13
2.3. 주요 기능	13
3. 적용 가이드	14
3.1. JAR 라이브러리 적용 방법	14
3.1.1. Jar 라이브러리 적용	14
3.1.2. SO 라이브러리 적용	14
3.1.3. 리소스 적용	14
3.1.4. AndroidManifest 설정	14
3.1.5. build.gradle 설정	17
3.1.6. Proguard 난독화 예외 설정	18
3.1.7. targetSDKVersion 31 이상 WorkManager 필수 적용	19
3.2. AAR 라이브러리 적용 방법	20
3.2.1. AAR 라이브러리 적용	20
3.2.2. AndroidManifest 설정	20
3.2.3. build.gradle 설정	22
3.2.4. Proguard 난독화 예외 설정	23
3.2.5. targetSDKVersion 31 이상 WorkManager 필수 적용	24
3.3. AAB 배포 적용 절차	25
3.3.1. gradle.properties 설정(gradle 8.1 미만)	25
3.4. API 적용 절차	26
3.4.1. 사이트 라이선스 등록 구현	26
3.4.2. 초기화 메서드 구현	26
3.4.3. 검사하기 구현	26
3.4.4. 백신 구동 이후 처리	28
3.4.5. 종료 구현	31

3.5. 기타 적용 가이드	32
3.5.1. 실시간 검사 (ScanReceiver).....	32
3.5.2. 결과 수신 리시버 (CodeReceiver).....	34
3.5.3. 루팅 API 사용법.....	37
3.5.4. 에뮬레이터 탐지.....	38
3.5.5. 탐지된 악성앱 리스트	39
3.5.6. 백그라운드 앱 강제 종료 시 Notification 제거.....	41
3.5.7. Notification 사용 설정 여부 확인.....	42
4. API 목록	48
5. API 상세	48
5.1. mini().....	48
5.2. full()	53
5.3. uifull()	56
5.4. startActivityResult().....	58
5.5. onActivityResult()	59
5.6. onDestroy()	60
5.7. rootingCheck().....	61
6. 에러 코드	63
7. 오픈소스 라이선스.....	63
[부록].....	64
1. FAQ	64
2. 기능 동작 화면	77
3. targetSdkVersion에 따른 변경 사항.....	78

1. 개요

문서는 총 7개의 장으로 구성되며, 각 장의 내용은 다음과 같습니다.

제 1장에서는 본 제품을 소개하고 문서의 구성과 용어를 설명합니다.

제 2장에서는 본 제품의 구성과 흐름 그리고 주요 기능에 대해 설명합니다.

제 3장에서는 본 제품의 API를 사용하기 위한 개발 환경 설정 방법을 설명합니다.

제 4장에서는 개발 시 활용되는 API에 대해서 설명합니다.

제 5장에서는 본 제품에서 사용되는 API 목적과 사용법에 대해 설명합니다.

제 6장에서는 본 제품 사용 시 발생하는 에러코드들에 대해 설명합니다.

제 7장에서는 오픈소스 라이선스 관련 내용을 안내합니다.

부록에서는 제품 적용 및 이해에 도움되는 추가적인 내용들을 설명합니다.

1.1. 제품 소개

본 제품은 스마트폰 등의 모바일 기기 환경에서 악성코드를 사전 탐지 및 치료를 수행하여, 전자금융/거래 서비스 이용 간 보다 안전한 이용환경을 제공합니다.

1.2. 목적

본 설명서는 TouchEn mVaccine 라이브러리를 연동대상 앱에 적용하기 위한 가이드를 제공합니다. 가이드에 제공된 내용대로 연동하지 않을 시 연동대상 앱에서 문제발생에 소지가 있으므로 연동 전 반드시 본 문서 내용을 확인 후 연동하시기 바랍니다.

1.3. 사전 준비사항

본 제품은 라이선스 발급을 필요로 합니다.

1.4. 지원 사양

- Mobile(app)

구분	내용
운영체제	• Android 6.0 이상

1.5. 용어 정리

본 문서에서 사용되는 용어의 정의는 다음과 같습니다.

- TouchEn mVaccine 라이브러리
core_b2b2c_x.x.x.xxx.jar 형태의 JAR 파일.

2. 제품 구성

2.1. 제품 구성

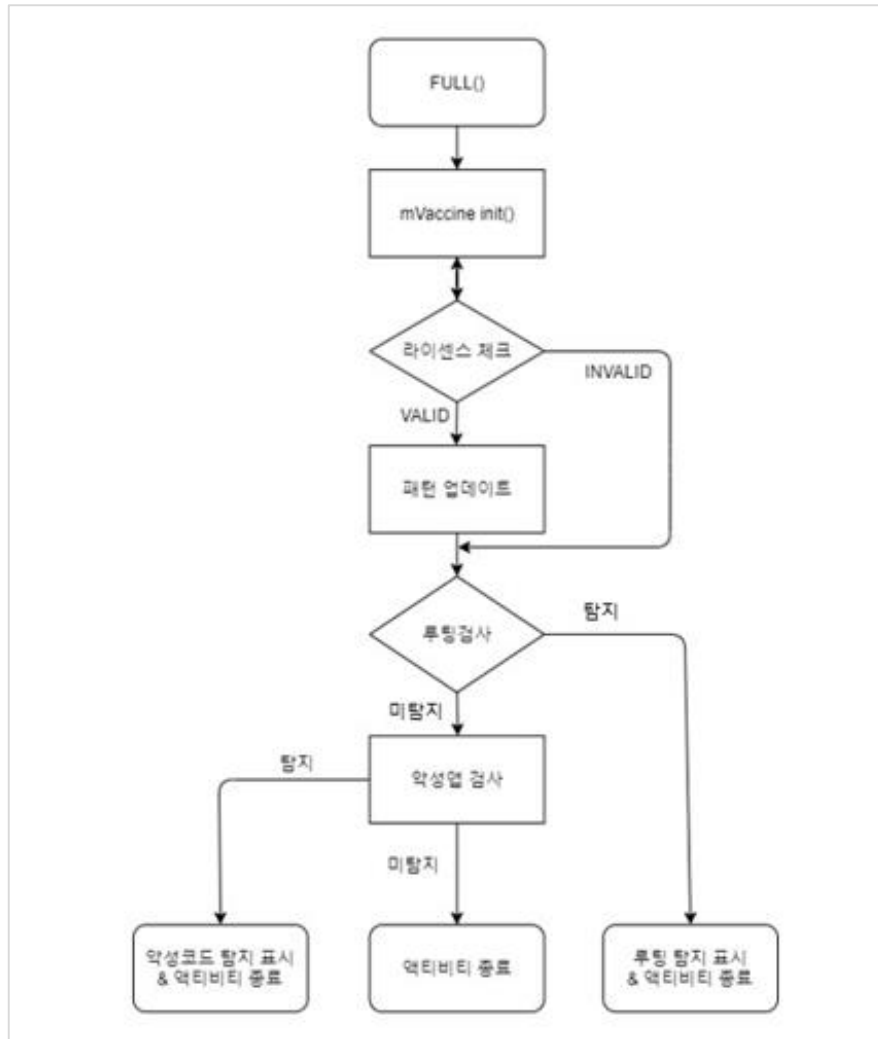
본 제품은 다음과 같이 구성되어 있습니다.

구분	폴더 및 파일명	설명
libs	Android/AAR/mvc_b2b2c_x.x.x.xxxx.aar	mVaccine AAR 라이브러리
	Android/AAR/RSLicenseSDK_x.x.x.jar	mVaccine 라이선스 라이브러리
	Android/JAR/01.libs/arm64-v8a	arm, x86 프로세스용 초기 구동 엔진 및 패턴 so
	Android/JAR/01.libs/armeabi	
	Android/JAR/01.libs/armeabi-v7a	
	Android/JAR/01.libs/x86	
	Android/JAR/01.libs/x86_64	
	Android/JAR/01.libs/core_b2b2c_x.x.x.xxxx.jar	mVaccine JAR 라이브러리
	Android/JAR/01.libs/RSLicenseSDK_x.x.x.jar	mVaccine 라이선스 라이브러리
	Android/JAR/02.res/drawable-xhdpi	mVaccine 이미지 파일
	Android/JAR/02.res/layout	mVaccine 레이아웃 파일
	Android/JAR/02.res/values	영문 리소스
	Android/JAR/02.res/values-ko	한글 리소스
	Android/JAR/02.res/values-ja	일본어 리소스
Doc	TouchEn mVaccine CS Android API 매뉴얼	제품 Android API 매뉴얼
	TouchEn mVaccine CS Android 원격감지 API 매뉴얼	원격감지 API 매뉴얼
Sample	Android Sample	AAR 라이브러리를 Kotlin 샘플 JAR 라이브러리를 Java 샘플
	Android Sample/Malware Sample(Android)	멀웨어 탐지용 샘플 APK ※ 자체, BD엔진 및 Supersu 루팅
	Android Sample/Remote Sample	원격 감지 Kotlin / Java 샘플

2.2. 제품 흐름도(Flow)

TouchEn mVaccine 제품의 처리 절차에 대한 플로우 차트를 안내합니다.

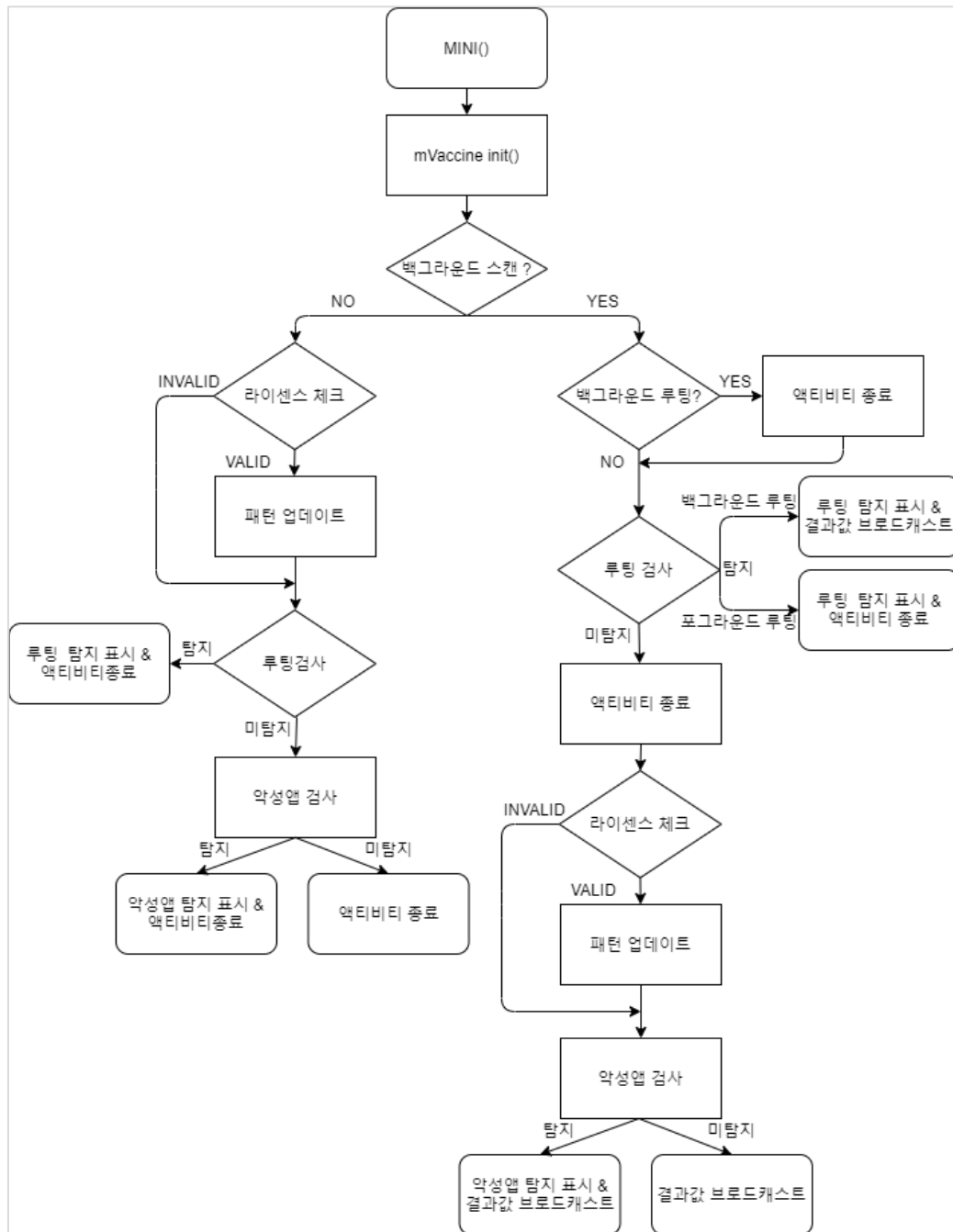
2.2.1. FULL 모드 흐름도



‘액티비티 종료’가 되면 Activity Result 값으로 검사 결과 값이 전달됩니다.

적용 앱은 Activity Result 값 (검사결과 값)을 바탕으로 다음 동작을 결정합니다.

2.2.2. MINI 모드 흐름도



backgroundScan 옵션이 true 일 경우 백그라운드 스캔이 실행됩니다.

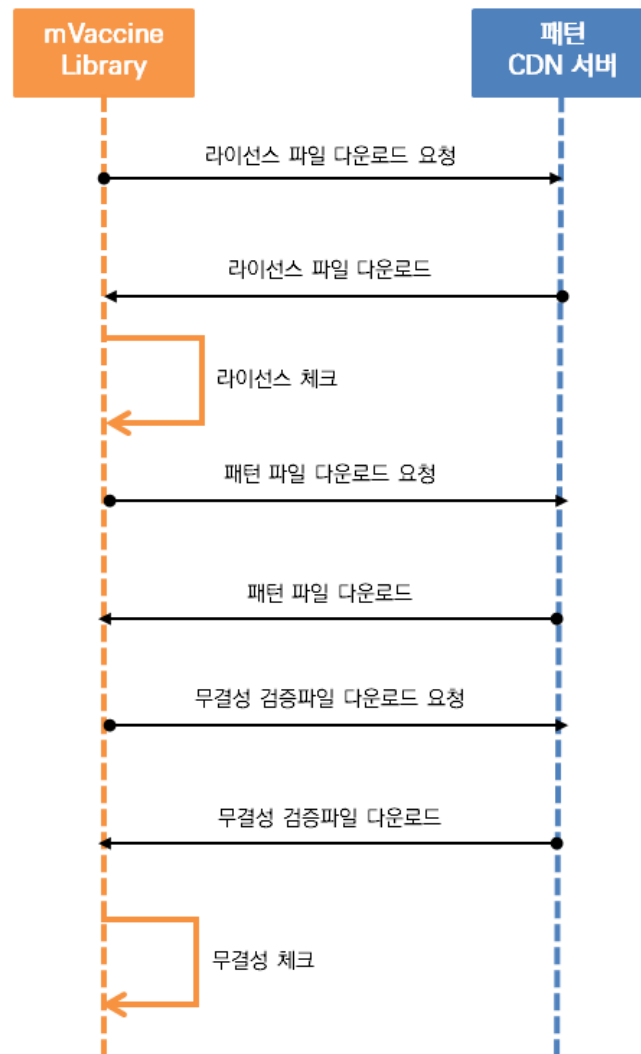
bg_rooting 옵션이 true 일 경우 백그라운드 루팅이 실행됩니다.

'액티비티 종료'가 되면 Activity Result 값으로 검사결과 값이 전달됩니다.

적용 앱은 Activity Result 값 (검사결과 값)을 바탕으로 다음 동작을 결정합니다.

액티비티 종료 후 검사결과 값에 대해서는 결과 수신 리시버(CodeReceiver)를 통해 받을 수 있습니다.

2.2.3. 패턴 업데이트 흐름도



라이선스 파일명: "사이트 아이디.dat"

라이선스 체크: 라이선스가 유효하지 않으면 패턴 다운로드 요청을 하지 않습니다.

2.3. 주요 기능

TouchEn mVaccine의 주요 기능은 다음과 같습니다.

구분	내용
악성 앱 탐지/치료	모바일 기기 전체 앱에 대한 검사를 수행하고 악성 앱을 탐지/치료합니다. 또한 서비스 중에도 지속적인 실시간 감시를 통해, 최초 구동 후 악성 앱이 설치 및 동작해도 탐지하여 치료합니다.
루팅 검사	모바일 기기의 루팅 여부를 탐지하고 결과 메시지를 보여줍니다.

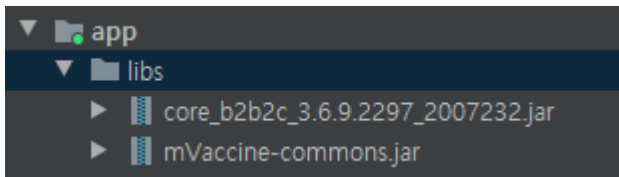
3. 적용 가이드

3.1. JAR 라이브러리 적용 방법

연동 대상 앱에서 mVaccine JAR 라이브러리를 사용할 경우 본 연동 가이드를 확인 후 연동 앱에 맞게 연동하시기 바랍니다.

3.1.1. Jar 라이브러리 적용

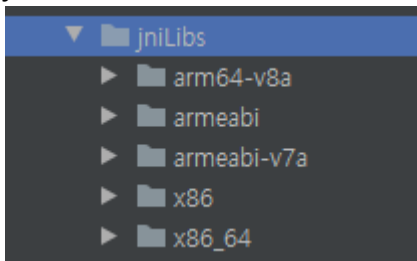
압축을 푼 폴더의 01.Moudle/Android/JAR/01.libs 폴더 안 JAR 라이브러리들을 연동 대상 앱 libs폴더에 복사합니다.



연동 대상 앱 프로젝트 내 build.gradle(app) 스크립트에 mVaccine 라이브러리 종속성을 추가합니다.

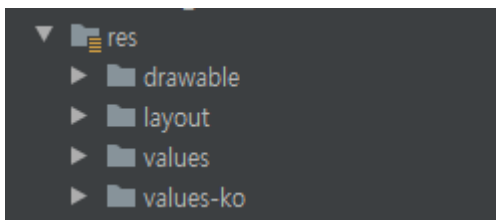
3.1.2. SO 라이브러리 적용

압축을 푼 폴더의 01.Moudle/Android/JAR/01.libs 폴더 안에 SO 라이브러리들을 연동 대상 앱 jniLibs폴더에 복사합니다.



3.1.3. 리소스 적용

압축을 푼 폴더의 01.Moudle/Android/JAR/02.res 폴더 안에 리소스 폴더들을 연동 대상 앱 res 폴더에 복사합니다.



3.1.4. AndroidManifest 설정

연동 대상 앱의 프로젝트 구성 파일 중 AndroidManifest.xml 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

3.1.4.1. Application 속성

```
<!-- mVaccine application 속성 설정 -->
<!-- APK에서 파일 시스템으로 네이티브 라이브러리를 추출하도록 설정 -->
<!-- 필수 속성은 아니지만 mVaccine 초기화 시간을 단축시켜줍니다. -->
<!-- 만약, AGP(Android Gradle Plugin) 4.2.0 이상을 사용한다면 AndroidManifest의 application
    속성 설정 대신 3.1.5.2. useLegacyPackaging 설정에 따라 설정하는 것을 권장합니다. --
>
<application android:extractNativeLibs="true"/>
```

3.1.4.2. 권한 등록

```
<!-- 패턴 업데이트 시 인터넷 권한 필요 -->
<uses-permission android:name="android.permission.INTERNET" />
<!-- mVaccine 패턴 업데이트 시 네트워크 정보 확인을 위한 권한 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<!-- targetSdk 28 이상에서 빌드할 경우 삭제 권한 필요 -->
<uses-permission android:name="android.permission.REQUEST_DELETE_PACKAGES" />
<!-- 악성코드 실시간 검사를 위해 사용되는 권한 -->
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<!-- Android 11 에서 설치된 패키지 명을 얻어올 때 필요 -->
<uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
<!-- android 13 이상 noti피케이션 표시를 위해 사용 -->
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
```

3.1.4.3. 컴포넌트 등록

```
<!-- mVaccine Activity 등록 -->
<!-- full()모드 구동을 위한 액티비티 -->
<activity
    android:name="com.TouchEn.mVaccine.b2b2c.activity.ScanActivity"
    android:theme="@style/Theme.mVaccine.NoTitleBar"
    android:screenOrientation="portrait"
    android:label="@string/app_name">
</activity>

<!-- uifull()모드 구동을 위한 액티비티 -->
<activity
    android:name="com.TouchEn.mVaccine.b2b2c.activity.UiScanActivity"
    android:theme="@style/Theme.mVaccine.NoTitleBar"
    android:screenOrientation="portrait"
    android:label="@string/app_name">
</activity>
```

<!-- mini()모드 구동을 위한 액티비티 -->

<activity

android:name="com.TouchEn.mVaccine.b2b2c.activity.BackgroundScanActivity"

android:label="@string/app_name"

android:configChanges="keyboard|keyboardHidden|orientation|screenLayout|screenSize"

android:theme="@style/Theme.mVaccine.Transparent" >

</activity>

<!-- 악성코드 탐지결과 리스트를 표시하기 위한 액티비티 -->

<activity

android:name="com.TouchEn.mVaccine.b2b2c.activity.ResultLogActivity"

android:label="@string/app_name"

android:screenOrientation="portrait"

android:exported="false"

android:theme="@style/Theme.mVaccine.NoTitleBar" >

</activity>

<!-- 악성코드 탐지결과 다이얼로그를 표시하기 위한 액티비티 -->

<activity

android:name="com.TouchEn.mVaccine.b2b2c.activity.BackgroundScan"

android:label="@string/app_name"

android:theme="@style/Theme.mVaccine.Transparent" >

</activity>

<!-- 실시간 검사를 위한 서비스 -->

<service **android:name="com.TouchEn.mVaccine.b2b2c.service.OnInstallService"**

android:process=":remote">

</service>

<!--백그라운드 구동시 악성코드 검사 결과 전달을 위한 서비스 -->

<service **android:name="com.TouchEn.mVaccine.b2b2c.service.DetectionResultSendService"**

android:process=":remote">

</service>

<!--루팅세부검사를 위한 서비스 (Internal Rooting Check 마운트검사(2번) 사용시 등록) -->

<service **android:name="com.TouchEn.mVaccine.b2b2c.service.MvclsolService"**

android:enabled="true"

android:isolatedProcess="true"


```

android:process=":mvclsolService">
</service>

<!--백그라운드에서 noti피케이션 제거 이슈 -->
<service android:name="com.TouchEn.mVaccine.b2b2c.service.RemoveNotificationService"
android:stopWithTask="false">
</service>

```

3.1.5. build.gradle 설정

연동 대상 앱의 프로젝트 구성 파일 중 app/build.gradle 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

3.1.5.1. doNotStrip 설정

AGP(Android Gradle Plugin) 4.2.0 이상

```

<!--build.gradle(app)의 packagingOptions에 옵션 추가-->
<!-- 네이티브 라이브러리 스트리핑 예외 처리-->
Android {
...
    PackagingOptions {
        .....
        JniLibs{
            keepDebugSymbols += "**/*.so"
        }
    }
}

```

AGP(Android Gradle Plugin) 4.2.0 미만

```

<!--build.gradle(app)의 packagingOptions에 옵션 추가-->
<!-- 네이티브 라이브러리 스트리핑 예외 처리-->
Android {
...
    PackagingOptions {
        .....
        doNotStrip "**/*.so"
    }
}

```

3.1.5.2. useLegacyPackaging 설정

AGP(Android Gradle Plugin) 4.2.0부터 AndroidManifest.xml의 <application> 속성으로 적용했던 extractNativeLibs 매니페스트 속성이 DSL 옵션 useLegacyPackaging으로 대체되었습니다. 매니페스트

파일의 extractNativeLivs 대신 앱의 build.gradle(app) 파일에서 useLegacyPackaging을 사용하여 네이티브 라이브러리 압축 동작을 구성해야 합니다.

```
<!--build.gradle(app)의 packagingOptions에 옵션 추가-->
<!-- APK에서 파일 시스템으로 네이티브 라이브러리를 추출하도록 설정 -->
Android {
    ...
    PackagingOptions {
        .....
        JniLibs{
            useLegacyPackaging = true
        }
    }
}
```

3.1.6. Proguard 난독화 예외 설정

Proguard 등의 난독화 적용 시 추가적인 설정을 통해 정상적인 구동을 제공합니다.

proguard-project.txt 파일에 아래와 같은 형식에 맞춰 예외처리 설정합니다.

Proguard 버전에 따라 오류 발생 소지가 있으니, 오류발생 시 로그 확인 및 전달 바랍니다.

3.1.6.1. proguard-rules.pro설정

```
-dontwarn org.apache.**
-dontwarn com.TouchEn.mVaccine.b2b2c.activity.**
-dontwarn com.secureland.smartmedic.core.**
-dontwarn com.bitdefender.antimalware.BDOfflineAVScanner
-dontwarn com.TouchEn.mVaccine.b2b2c.util.ProgressWheel

-keepclasseswithmembers class com.bitdefender.antimalware.BDOfflineAVScanner{
    private java.lang.String m_threatName;
    private int m_scanResult;
    private int m_threatType;
}

-keepclasseswithmembernames class * {
    native <methods>;
}

-keepclasseswithmembers class *{
    *** *Callback(...);
}

-keep class com.secureland.smartmedic.core.SmartMedicJni{
    *;
```

```

}
# 적용 앱의 패키지명.R$styleable 형식으로 사용합니다.
-keep class com.TouchEn.mVaccine.DEMO.R$styleable{
    *;
}

```

3.1.6.2. app/build.gradle설정

```

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

```

3.1.7. targetSDKVersion 31 이상 WorkManager 필수 적용

Android12(targetSDKVersion31) 이상 타겟팅 하는 앱부터 백그라운드에서 실행되는 동안 포그라운드 서비스 실행이 제한됩니다. mVaccine은 백그라운드 검사(MINI모드, 실시간 검사 등...)를 지원하기 때문에 Android12 이상 지원을 위해 WorkManager 등록을 통해 백그라운드에서 포그라운드 서비스 실행이 필요합니다. (<https://developer.android.google.cn/about/versions/12/foreground-services?hl=ko>)

3.1.7.1. 지원 사양

안드로이드12의 포그라운드 서비스 제한을 지원하기 위해서는 2.7.0 이상 사용이 필요합니다.

(<https://developer.android.com/jetpack/androidx/releases/work?hl=ko#2.7.0>)

구분	내용
WorkManager	• 2.7.0 이상
CompileSDK	• 31 이상
Gradle JDK	• 11 이상

3.1.7.2. gradle 설정

아래와 같이 app/build.gradle에 dependencies에 **implementation "androidx.work:work-runtime:2.7.0"** 추가해줍니다.

```

apply plugin: 'com.android.application'

...

dependencies {
    ...
}

```

```
        implementation "androidx.work:work-runtime:2.7.0"
    }
}
```

3.1.7.3. gradle.properties 설정

연동 대상 앱의 프로젝트 구성 파일 중 gradle.properties 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

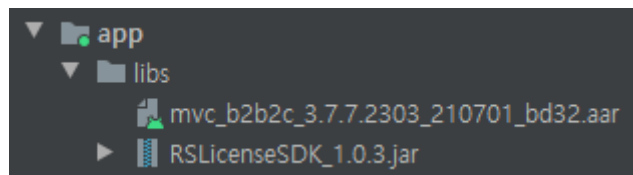
```
<!-- 3.2.5. targetSDKVersion 31 이상 WorkManager 필수 적용 -->
android.useAndroidX=true
android.enableJetifier=true
```

3.2. AAR 라이브러리 적용 방법

연동 대상 앱에서 mVaccine AAR 라이브러리를 사용할 경우 본 연동 가이드를 확인 후 연동 앱에 맞게 연동하시기 바랍니다.

3.2.1. AAR 라이브러리 적용

패키지 압축 파일 내 '01. Module/Android/AAR' 폴더의 모든 파일(aar, jar)을 연동 대상 앱 libs폴더에 복사합니다.



연동 대상 앱 프로젝트 내 build.gradle(app) 스크립트에 mVaccine 라이브러리 종속성을 추가해 줍니다.

```
repositories { flatDir { dirs 'libs' } }

dependencies {

    implementation name: 'mvc_b2b2c_3.7.7.2303_210701_bd32', ext: 'aar'
    implementation files('libs/RSLicenseSDK_1.0.3.jar')
}
```

3.2.2. AndroidManifest 설정

연동 대상 앱의 프로젝트 구성 파일 중 AndroidManifest.xml 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

3.2.2.1. Application 속성

```
<!-- mVaccine application 속성 설정 -->
<!-- APK에서 파일 시스템으로 네이티브 라이브러리를 추출하도록 설정 -->
<!-- 필수 속성은 아니지만 mVaccine 초기화 시간을 단축시켜줍니다. -->
<!-- 만약, AGP(Android Gradle Plugin) 4.2.0 이상을 사용한다면 AndroidManifest의 application
    속성 설정 대신 3.2.3.2. useLegacyPackaging 설정에 따라 설정하는 것을 권장합니다. --
>
<application android:extractNativeLibs="true"/>
```

3.2.2.2. 권한 등록

```
<!-- Android 11 에서 설치된 패키지 명을 얻어올 때 필요 -->
<uses-permission android:name="android.permission.QUERY_ALL_PACKAGES"/>
```

3.2.2.3. 컴포넌트 등록

프로젝트가 targetSDK 34 미만일 경우 다음과 같이 mVaccine CS AAR의 서비스를 replace 해줍니다.

```
<!-- Android 운영체제에서 애플리케이션 스레드 종료처리 방지용 서비스 추가 -->
<service
    android:name="com.TouchEn.mVaccine.b2b2c.service.OnInstallService"
    android:process=":remote"
    tools:node="replace" />

<!-- Q 대응 헤드업 노티를 띄워주기 위해서 사용 -->
<service
    android:name="com.TouchEn.mVaccine.b2b2c.service.DetectionResultSendService"
    tools:node="replace"/>

<!-- 백그라운드에서 노티피케이션 제거 이슈 -->
<service android:name="com.TouchEn.mVaccine.b2b2c.service.RemoveNotificationService"
    android:stopWithTask="false"
    tools:node="replace" />

<!-- WorkManager ForegroundService 적용 -->
<service
    android:name="androidx.work.impl.foreground.SystemForegroundService"
    tools:node="replace" />
```

3.2.3. build.gradle 설정

연동 대상 앱의 프로젝트 구성 파일 중 app/build.gradle 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

3.2.3.1. doNotStrip 설정

AGP(Android Gradle Plugin) 4.2.0 이상

```
<!--build.gradle(app)의 packagingOptions에 옵션 추가-->
<!-- 네이티브 라이브러리 스트리핑 예외 처리-->
Android {
    ...
    PackagingOptions {
        .....
        JniLibs{
            keepDebugSymbols += "**/*.so"
        }
    }
}
```

AGP(Android Gradle Plugin) 4.2.0 미만

```
<!--build.gradle(app)의 packagingOptions에 옵션 추가-->
<!-- 네이티브 라이브러리 스트리핑 예외 처리-->
Android {
    ...
    PackagingOptions {
        .....
        doNotStrip "**/*.so"
    }
}
```

3.2.3.2. useLegacyPackaging 설정

AGP(Android Gradle Plugin) 4.2.0부터 AndroidManifest.xml의 <application> 속성으로 적용했던 extractNativeLibs 매니페스트 속성이 DSL 옵션 useLegacyPackaging으로 대체되었습니다. 매니페스트 파일의 extractNativeLivs 대신 앱의 build.gradle(app) 파일에서 useLegacyPackaging을 사용하여 네이티브 라이브러리 압축 동작을 구성해야 합니다.

```
<!--build.gradle(app)의 packagingOptions에 옵션 추가-->
<!-- APK에서 파일 시스템으로 네이티브 라이브러리를 추출하도록 설정 -->
Android {
    ...
    PackagingOptions {
        .....
    }
}
```

```
JniLibs{
    useLegacyPackaging = true
}
}
```

3.2.4. Proguard 난독화 예외 설정

Proguard 등의 난독화 적용 시 추가적인 설정을 통해 정상적인 구동을 제공합니다.

proguard-project.txt 파일에 아래와 같은 형식에 맞춰 예외처리 설정합니다.

Proguard 버전에 따라 오류 발생 소지가 있으니, 오류발생 시 로그 확인 및 전달 바랍니다.

3.2.4.1. proguard-rules.pro 설정

```
-dontwarn org.apache.**
-dontwarn com.TouchEn.mVaccine.b2b2c.activity.**
-dontwarn com.secureland.smartmedic.core.**
-dontwarn com.bitdefender.antimalware.BDOfflineAVScanner
-dontwarn com.TouchEn.mVaccine.b2b2c.util.ProgressWheel

-keepclasseswithmembers class com.bitdefender.antimalware.BDOfflineAVScanner{
    private java.lang.String m_threatName;
    private int m_scanResult;
    private int m_threatType;
}

-keepclasseswithmembernames class * {
    native <methods>;
}

-keepclasseswithmembers class *{
    *** *Callback(...);
}

-keep class com.secureland.smartmedic.core.SmartMedicJni{
    *;
}

# 적용 앱의 패키지명.R$styleable 형식으로 사용합니다.
-keep class com.TouchEn.mVaccine.DEMO.R$styleable{
    *;
}
```

3.2.4.2. app/build.gradle 설정

```
buildTypes {
```

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

3.2.5. targetSDKVersion 31 이상 WorkManager 필수 적용

Android12(targetSDKVersion31) 이상부터 백그라운드에서 실행되는 동안 포그라운드 서비스 실행이 제한됩니다. mVaccine은 백그라운드 검사(MINI모드, 실시간 검사 등...)를 지원하기 때문에 Android12 이상 지원을 위해 WorkManager 등록을 통해 백그라운드에서 포그라운드 서비스 실행이 필요합니다. <https://developer.android.google.cn/about/versions/12/foreground-services?hl=ko>

3.2.5.1. 지원 사양

안드로이드12의 포그라운드 서비스 제한을 지원하기 위해서는 2.7.0 이상 사용이 필요합니다.

<https://developer.android.com/jetpack/androidx/releases/work?hl=ko#2.7.0>

구분	내용
WorkManager	• 2.7.0 이상
CompileSDK	• 31 이상
Gradle JDK	• 11 이상

3.2.5.2. gradle 설정

아래와 같이 build.gradle에 dependencies에 implementation "androidx.work:work-runtime:2.7.0" 추가 해줍니다.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 31
    buildToolsVersion '30.0.2'
    defaultConfig {
        applicationId "com.TouchEn.mVaccine.DEMO"
        minSdkVersion 15
        targetSdkVersion 31
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
```



```

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

    packagingOptions {
        /** Android Gradle Plugin 4.2 이상 */
        jniLibs {
            keepDebugSymbols += "**/*.so"
        }
        /** Android Gradle Plugin 4.2 미만 */
        // doNotStrip "**/*.so"
    }

    productFlavors {
    }
}

dependencies {
    implementation 'androidx.appcompat:appcompat:1.0.0'
    implementation "androidx.work:work-runtime:2.8.0"
    implementation fileTree(dir: 'libs', include: ['*.aar', '*.jar'], exclude: [])
}

```

3.2.5.3. gradle.properties 설정

연동 대상 앱의 프로젝트 구성 파일 중 gradle.properties 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

```

<!-- 3.2.5. targetSDKVersion 31 이상 WorkManager 필수 적용 -->
android.useAndroidX=true
android.enableJetifier=true

```

3.3. AAB 배포 적용 절차

3.3.1. gradle.properties 설정(gradle 8.1 미만)

연동 대상 앱의 프로젝트 구성 파일 중 gradle.properties 파일에 mVaccine 구동에 필요 한 설정을 추가합니다.

<!-- '3.1.4.1. / 3.2.2.1. Application 속성'에서 android:extractNativeLibs="true" 사용시 설정 -->
android.bundle.enableUncompressedNativeLibs = false

3.4. API 적용 절차

3.4.1. 사이트 라이선스 등록 구현

mVaccine이 구동 될 연동앱 MainActivity의 onCreate() 함수 부분에 아래와 같이 구현합니다.
 발급받은 사이트 ID, KEY 값을 MainActivity의 onCreate() 부분에 구현합니다.

```
public void onCreate(Bundle savedInstanceState) {
    /* mVaccine, 사이트 ID, 라이선스key 설정 */
    com.secureland.smartmedic.core.Constants.site_id="지급받은 사이트 ID";
    com.secureland.smartmedic.core.Constants.license_key="지급받은
    라이선스 KEY";
}
```

Information

사이트 인증을 위해 지급받은 사이트 ID, KEY 값을 정확하게 입력해야 합니다.
 ID, KEY 값이 맞지 않을 경우 mVaccine 구동이 정상적으로 되지 않습니다.

3.4.2. 초기화 메서드 구현

백신 엔진을 초기화 하는 코드입니다. 패턴/엔진의 업데이트, 검사하기의 기능을 초기화 합니다.
 MainActivity의 onCreate() 부분에 초기화 메소드를 구현합니다.

```
public void onCreate(Bundle savedInstanceState) {
    /* mVaccine, 백신 엔진 초기화 */
    try {
        SmartMedic.init(this);
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

3.4.3. 검사하기 구현

mVaccine 구동을 구현합니다. 4장 API 상세 및 MainActivity_sample.java 샘플소스를 참고하여

연동바랍니다. 아래 권장 옵션 조합 Case로 구현하시길 바랍니다.

3.4.3.1. mini 모드

mVaccine 검사 시 검사진행 UI가 없이 진행되는 모드입니다. (부록2. 기능 동작 화면 참조)

백신 검사 진행 UI가 생략된 UI 간소화 모드 백신 구동입니다.

화면 중앙 백신 구동 중 이미지가 보여집니다.

Activity 사용하는 mini 모드

```
public void mini() {
    Intent i = new Intent(this, BackgroundScanActivity.class);
    i.putExtra("useBlackAppCheck", true);
    i.putExtra("scan_rooting", true);
    i.putExtra("scan_package", true);
    i.putExtra("useDualEngine", true);
    i.putExtra("backgroundScan", true); // mini 전용
    i.putExtra("rootingexitapp", true);
    i.putExtra("show_update", false);
    i.putExtra("show_license", false);
    i.putExtra("show_notify", false); // mini 전용
    i.putExtra("show_scan_ui", true); // mini 전용

    this.startActivityForResult(i, REQUEST_CODE);
}
```

Thread 사용하는 mini 모드

```
public void mini() {
    Bundle bundle = new Bundle();

    bundle.putBoolean("useBlackAppCheck", true);
    bundle.putBoolean("scan_rooting", true);
    bundle.putBoolean("scan_package", true);
    bundle.putBoolean("useDualEngine", true);
    bundle.putBoolean("rootingexitapp", true);
    bundle.putBoolean("show_update", false);
    bundle.putBoolean("show_license", false);
    bundle.putBoolean("show_notify", false); // mini 전용
    bundle.putBoolean("show_scan_ui", true); // mini 전용

    CommonUtil.scanMini(getApplicationContext(), bundle);
}
```

3.4.3.2. full 모드

mVaccine 검사 시 검사 진행 UI가 사용자에게 제공되는 모드입니다. (부록2. 기능 동작 화면 참조)

백신 검사 진행 UI가 제공되는 모드의 백신 구동입니다.

```
public void full() {
    Intent i = new Intent(this, ScanActivity.class);
    i.putExtra("useBlackAppCheck", true);
    i.putExtra("scan_rooting", true);
    i.putExtra("scan_package", true);
    i.putExtra("useDualEngine", true);
    i.putExtra("rootingexitapp", true);
    i.putExtra("show_license", false);
    i.putExtra("show_toast", false);
    i.putExtra("show_update", false);
    this.startActivityForResult(i, REQUEST_CODE);
}
```

3.4.3.3. ui full 모드

mVaccine 검사 시 full 모드 보다 상세한 검사 진행 UI가 사용자에게 제공되는 모드이며, 상세한 백신 검사 진행 UI가 제공되는 모드의 백신 구동입니다. (부록 내 기능동작 화면 참조)

```
public void uiFull() {
    Intent i = new Intent(this, UiScanActivity.class);
    i.putExtra("useBlackAppCheck", true);
    i.putExtra("scan_rooting", true);
    i.putExtra("scan_package", true);
    i.putExtra("useDualEngine", true);
    i.putExtra("rootingexitapp", true);
    i.putExtra("show_license", false);
    i.putExtra("show_toast", false);
    i.putExtra("show_update", false);
    this.startActivityForResult(i, REQUEST_CODE);
}
```

3.4.4. 백신 구동 이후 처리

MainActivity에 onActivityResult() 함수를 아래와 같이 구현하여 백신 구동 이후 연동 로직을 작성하여, 이후 연동 앱 구동을 수행합니다.

3.4.4.1. Request Code 정의

mVaccine 구동 시 RequestCode 로 정의된 값을 정의합니다.

```
/* mVaccine, RequestCode 정의 */
public static final int REQUEST_CODE = 777;
private final static int MESSAGE_ID = 12345;
private final static int MESSAGE_ID1 = 123456;
```

속성	설명	값
REQUESTCODE	제품의 RequestCode	777

EMPTY_VIRUS	바이러스가 탐지되지 않은 경우	1000
EXIST_VIRUS	바이러스가 탐지 된 경우 (MINI모드 전용)	1001
EXIST_VIRUS_CASE1	바이러스 탐지 후 사용자가 모든 바이러스 삭제	1010
EXIST_VIRUS_CASE2	바이러스 탐지 후 사용자가 바이러스를 부분 치료한 경우	1100
ROOTING_EXIT_APP	루팅시 앱 종료	1200
ROOTING_YES_OR_NO	루팅 (예/아니오)	1210
ROOTING_YES	루팅시 경고 창 뜬 후 “예” 버튼 누를 시 검사 진행	1220
V_DB_FAIL	DB파일 무결성 검증 실패	1240
SCAN_STOP	백신검사 중 검사중지 버튼을 클릭하여 임의 중단한 경우	1230
POST_NOTI_NO	알림 허용 권한이 없는 경우	1300
END_INIT	초기화 완료	1400
MESSAGE_ID	검사진행 Notification	12345
MESSAGE_ID1	검사결과 Notification	123456

3.4.4.2. 백신구동 이후 결과값 처리 – Activity Result로 받을 때

Code 값 결과 받는 방법

```
Activity호출 시 this.startActivityForResult("호출할 Activity로 보낼 인텐트값", 백신의 코드);
//인텐트 보내고 결과값을 얻어오는 경우
```

백신 구동 이후 로직 연동

```
public void onActivityResult(int requestCode, int resultCode, Intent intent){
    if ( requestCode == 정의된 코드값 ){
        if(resultCode== "정의된 코드값"){
            //코드값에 따른 수행메소드
        }
    }
}
```

샘플 코드

```
@Override
public void onActivityResult(int requestCode, int resultCode, Intent intent) {
    super.onActivityResult(requestCode, resultCode, intent);
    Log.d("mvc", "resultCode :" + resultCode);
    if (requestCode == 777) {
        switch(resultCode){
            case com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP:
```

```

        case com.secureland.smartmedic.core.Constants.ROOTING_YES_OR_NO:
        case com.secureland.smartmedic.core.Constants.V_DB_FAIL:
            this.finish();
            break;
        case com.secureland.smartmedic.core.Constants.EMPTY_VIRUS:
            break;
        case com.secureland.smartmedic.core.Constants.EXIST_VIRUS:
            break;
        case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1:
            break;
        case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE2:
            break;
    }
}
}

```

3.4.4.3. 백신구동 이후 결과값 처리 - 브로드캐스트 리시버로 받을 때

백신 검사 결과값은 브로드캐스트 리시버로도 받아 볼 수 있습니다.

결과 값 받을 리시버 등록 시 다음과 같은 인텐트 필터를 적용합니다.

```

<intent-filter>
    <action android:name="com.TouchEn.mVaccine.b2b2c.FIRE" />
</intent-filter>

```

리시버에 결과 수신 코드를 적용합니다

```

@Override
public void onReceive(Context context, Intent intent) {

    int i = intent.getIntExtra("result", 0);

    switch(i){
        case com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP:
        case com.secureland.smartmedic.core.Constants.ROOTING_YES_OR_NO:
        case com.secureland.smartmedic.core.Constants.V_DB_FAIL:
            break;
        case com.secureland.smartmedic.core.Constants.EMPTY_VIRUS:
            break;
        case com.secureland.smartmedic.core.Constants.EXIST_VIRUS:
            break;
    }
}

```

```

        case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1:
            break;
        case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE2:
            break;
    }
}

```

3.4.5. 종료 구현

MainActivity에 초기화 onDestroy() 함수를 아래와 같이 구현하여 백신을 종료합니다.

Information

CommonUtil.isRun = false; 옵션은 사용하면 안됩니다. (코드에 있을 시 반드시 삭제필요)

```

@Override
protected void onDestroy() {
    super.onDestroy();
    //mVaccine Notification 제거
    Util.cancelMvcNotification(getApplicationContext());

    //실시간 탐지, CodeReceiver 해제
    try {
        unregisterReceiver(scanReceiver);
    } catch (IllegalArgumentException e) {
        //scanReceiver not registered
    }
    try {
        unregisterReceiver(codeReceiver);
    } catch (IllegalArgumentException e) {
        //codeReceiver not registered
    }

    // 종료 toast 메시지 표시
    Toast.makeText(this, "TouchEn_mVaccine을  
종료합니다.", Toast.LENGTH_SHORT).show();

    // mVaccine 종료
}

```

```

        this.finish();
    }

```

3.5. 기타 적용 가이드

3.5.1. 실시간 검사 (ScanReceiver)

앱이 설치되거나 업데이트 될 때 실시간으로 검사하고 결과를 표시합니다.

Information

targetSdkversion 26 이상부터는 정적 리시버 등록이 제한되어 ScanReceiver를 동적으로 등록해서 사용해야 합니다.

3.5.1.1. 동작 흐름도



3.5.1.2. 동적 등록 가이드

앱을 시작할 때 실시간 검사 리시버인 ScanReceiver를 동적으로 등록합니다.

```

@Override
public void onCreate(Bundle savedInstanceState) {

    /*----- 실시간검사 동적 등록 방법 -----*/
    IntentFilter intentFilter = new IntentFilter();
    intentFilter.addAction("android.intent.action.PACKAGE_ADDED");
    intentFilter.addAction("android.intent.action.PACKAGE_INSTALL");
    intentFilter.addAction("android.intent.action.PACKAGE_CHANGED");

```



```
intentFilter.addAction("android.intent.action.PACKAGE_REPLACED");
intentFilter.addDataScheme("package");
ScanReceiver scanReceiver = new ScanReceiver();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    registerReceiver(scanReceiver, intentFilter,
RECEIVER_NOT_EXPORTED);
} else {
    registerReceiver(scanReceiver, intentFilter);
}
```

앱이 종료될 때 등록한 리시버를 해제합니다.

```
try {
    unregisterReceiver(scanReceiver);
} catch (IllegalArgumentException e) {
    //scanReceiver not registered
}...
```

3.5.1.3. 실시간 검사 옵션

속성	설명
setScanPhishing	true: 보이스 피싱 악성앱 검사
	false: 보이스 피싱 악성앱 검사 안 함
	default : true
setShowBadge	true: 아이콘에 뱃지 표시
	false: 미 출력
	default : false
setNotifyClearable	true: Swipe를 통해 noti피케이션 삭제 가능
	false: noti피케이션 삭제 불가능
	default : false
setUseDualEngine	true: BitDefender 엔진 검사
	false: 검사 안 함
	default : false

```
scanReceiver.setScanOption(
    new ScanOption.Builder()
        .setScanPhishing(true)
        .setShowBadge(false)
        .setNotifyClearable(true)
```

```

        .setUseDualEngine(true)

        .build()

    );

    registerReceiver(scanReceiver, intentFilter);

```

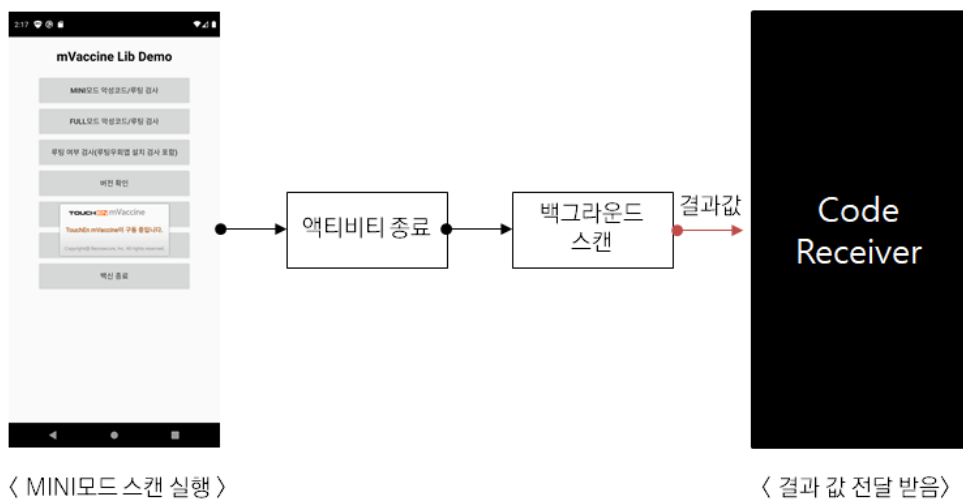
3.5.2. 결과 수신 리시버 (CodeReceiver)

스캔이 백그라운드에서 동작되는 경우 해당 리시버로 결과 코드가 전송됩니다.

Information

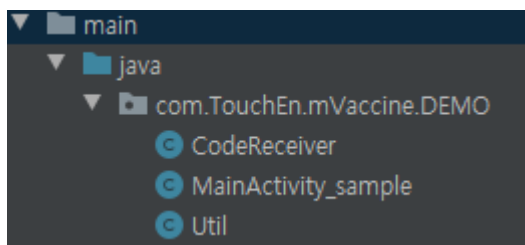
해당 리시버를 사용하기 위해서는 정적/동적 등록 중 하나의 방법으로 등록해야 합니다.
Thread 사용 mini 모드 사용시 검사 결과를 받기 위해 필수적으로 사용해야 합니다.

3.5.2.1. 동작 흐름도



3.5.2.2. 정적 등록 가이드

프로젝트에 CodeReceiver를 구현합니다. (mVaccine Sample Project 참고)



```
public class CodeReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        int i = intent.getIntExtra( name: "result", defaultValue: 0);
        Log.e( tag: "CodeReceiver", msg: "result = " + i);
        switch (i){
            case com.secureland.smartmedic.core.Constants.EMPTY_VIRUS: //1000
                Log.e( tag: "CodeReceiver", msg: "com.secureland.smartmedic.core.Constants.EMPTY_VIRUS");
                break;

            case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1: //1010
                Log.e( tag: "CodeReceiver", msg: "com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1");
                break;

            case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE2: //1100
                Log.e( tag: "CodeReceiver", msg: "com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE2");
                break;

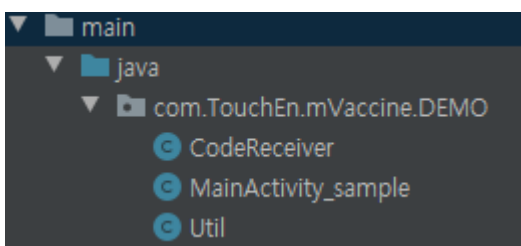
            case com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP: //1200
            case com.secureland.smartmedic.core.Constants.ROOTING_YES_OR_NO: //1210
            case com.secureland.smartmedic.core.Constants.V_DB_FAIL: //1240
                break;
        }
    }
}
```

AndroidManifest.xml에 CodeReceiver를 등록합니다.

```
<!-- mVaccine Activity 등록 -->
<!-- full()모드 구동을 위한 액티비티 -->
<receiver
    android:name=".CodeReceiver"
    android:exported="false">
    <intent-filter>
        <action android:name="com.TouchEn.mVaccine.b2b2c.FIRE" />
    </intent-filter>
</receiver>
```

3.5.2.3. 동적 등록 가이드

CodeReceiver를 구현합니다. (mVaccine Sample Project 참고)



```
public class CodeReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        int i = intent.getIntExtra( name: "result", defaultValue: 0);
        Log.e( tag: "CodeReceiver", msg: "result = " + i);
        switch (i){
            case com.secureland.smartmedic.core.Constants.EMPTY_VIRUS: //1000
                Log.e( tag: "CodeReceiver", msg: "com.secureland.smartmedic.core.Constants.EMPTY_VIRUS");
                break;

            case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1: //1010
                Log.e( tag: "CodeReceiver", msg: "com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1");
                break;

            case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE2: //1100
                Log.e( tag: "CodeReceiver", msg: "com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE2");
                break;

            case com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP: //1200
            case com.secureland.smartmedic.core.Constants.ROOTING_YES_OR_NO: //1210
            case com.secureland.smartmedic.core.Constants.V_DB_FAIL: //1240
                break;

        }
    }
}
```

앱을 시작할 때 결과 수신 리시버인 CodeReceiver를 동적으로 등록합니다.

인텐트 필터명: 패키지명 + ".mVaccine.FIRE"

```
IntentFilter intentFilter2 = new IntentFilter();
intentFilter2.addAction(getPackageName()+".mVaccine.FIRE");
codeReceiver = new CodeReceiver();
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    registerReceiver(codeReceiver, intentFilter2,
RECEIVER_NOT_EXPORTED);
} else {
    registerReceiver(codeReceiver, intentFilter2);
}
```

앱이 종료될 때 등록한 리시버를 해제합니다.

```
try {
    unregisterReceiver(codeReceiver);
} catch (IllegalArgumentException e) {
    //codeReceiver not registered
}
```

3.5.3. 루팅 API 사용법

제품 옵션 적용 이외에 별도 Rooting 관련 API를 제공합니다.

제품 구동 간 옵션에 의한 백신 검사 중 루팅 체크 이외에 각각의 요소에서 루팅 체크를 할 수 있도록 별도로 API를 제공합니다.

```
import com.TouchEn.mVaccine.b2b2c.util.CommonUtil; // import CommonUtil 클래스

/* ===== 1. 기본 루팅 API ===== */
// 결과값에 따른 리스너 구현 필요
CommonUtil.CListener cListener = new CommonUtil.CListener() {
    @Override
    public void onFinish(String result, String[] message) {
        // 구현 부분
        // 메인 스레드가 아니므로 UI작업이 필요하다면 Looper 에서 작업 필요
    }
};
CommonUtil.c(this, true, message, 0, cListener);
// 위 함수와 동일하게 사용가능 (함수명 유추를 막기 위해 c 권장)
CommonUtil.checkRooting(this, true, message, 0, cListener);

/* ===== 2. 동기 기본 루팅 API ===== */
/* ===== v3.9.8.2316 - deprecated. ===== */
CommonUtil.checkRooting(this,true);
// 암호화된 루팅 API (함수명 유추를 막기 위해 c 권장)
CommonUtil.c(this,true);

/* ===== 3. 앱 네임 리턴 형 루팅 API ===== */
/* ===== v3.9.8.2316 - deprecated. ===== */
String message[] = new String[2];
CommonUtil.checkRooting(this, true, message);
message[0]; //(루팅관련 앱 탐지시 앱네임 리턴값)
// 암호화된 루팅 API (함수명 유추를 막기 위해 c 권장)
CommonUtil.c(this,true,message);

/* ===== 4. internal 루팅 API ===== */
/* ===== v3.9.8.2316 - deprecated. ===== */
```

```
CommonUtil.checkRooting(this, true, message, 0);-
// 위 함수와 동일하게 사용가능 (함수명 유추를 막기 위해 c 권장)
CommonUtil.c(this,true,message,0);-

/* ===== 5. 무결성 검증 옵션 추가된 루팅 API ===== */
/* ===== v3.9.8.2316 - deprecated. ===== */
CommonUtil.checkRootingVso(this, true, message, true);-
```

Rooting API 파라미터는 표와 같습니다.

속성	설명
첫번째	호출하는 액티비티 자신
두번째	True시 불법 앱 검사/false시 불법 앱 검사 안 함
세번째	루팅 관련 앱 탐지 시 앱 이름을 리턴 받기 위한String 배열
네번째	Internal 루팅 검사 시 사용. 0: 기본검사, 1: 파일상세검사, 2: 마운트 검사, 3: 모두 검사 (확장성을 위해 Integer 자료형을 사용) (2번 마운트 검사 사용시 FAQ 참고)
다섯번째	비동기로 결과를 받아오기 위한 리스너 CLitener

Rooting API 리턴 값은 표와 같습니다.

속성	설명
0	루팅 되어 있지 않음
1	루팅 됨 (루팅시 생성되는 디렉토리 경로 패턴 탐지)
2, 3	불법 앱(루팅 관련 우회 앱) 활동(프로세스)이 감지됨
4	불법 앱이(루팅 관련 우회 앱) 설치되어 있음 두번째 파라미터 true 시에만 검출
5	앱 변조 시도가 감지됨
6	무결성 검증 실패 시

3.5.4. 에뮬레이터 탐지

제품 옵션 적용 이외에 별도 에뮬레이터탐지 API를 제공합니다.

```
import com.TouchEn.mVaccine.b2b2c.util.CommonUtil; // import CommonUtil 클래스

// v3.7.7.2303 - deprecated.
CommonUtil.isEmulator(getApplicationContext());-

// 2. 비동기형 에뮬레이터 탐지 API
```

// 패턴업데이트가 변경될 경우 2번째 파라미터 updateUrl 값을 수정합니다.

```
CommonUtil.isEmulator(getApplicationContext(), updateUrl: null, new CommonUtil.CkEmListener() {
    @Override
    public void OnFinished(boolean isem) {
        // isem : 탐지 결과값
        if(isem)
            Toast.makeText(getApplicationContext(), text: "에뮬레이터로 탐지되었습니다.", Toast.LENGTH_SHORT).show();
    }
});
```

// 암호화된 에뮬레이터 탐지 API (함수명 유추를 막기 위해 isem 권장합니다.)

```
CommonUtil.isem(getApplicationContext(), updateUrl: null, new CommonUtil.CkEmListener() {
    @Override
    public void OnFinished(boolean isem) {
        // isem : 탐지 결과값
        if(isem)
            Toast.makeText(getApplicationContext(), text: "에뮬레이터로 탐지되었습니다.", Toast.LENGTH_SHORT).show();
    }
});
```

API 리턴 값은 표와 같습니다.

속성	설명
true	에뮬레이터 기기로 판단
false	에뮬레이터 기기가 아닌 것으로 판단

3.5.5. 탐지된 악성앱 리스트

탐지된 악성 앱에 대한 정보를 ArrayList<Map>으로 리턴 해 줍니다.

MINI모드, FULL모드에 리스너 등록을 통한 방식, API를 호출하는 방식 세 가지를 제공합니다.

리턴 Map keys는 표와 같습니다.

속성	설명
virusName	고유 번호
name	악성 앱 이름
path	악성 앱 경로
_id	인덱스
scanType	PA : 패키지 RT : 루팅
packageName	패키지 명
check	사용자 체크박스 체크 여부
malware_type	0 : 악성 앱 1 : 보이스피싱 앱 2 : DualEngine 악성 앱

3.5.5.1. MINI모드 리스너 등록

MINI모드 검사가 끝남과 동시에 탐지된 악성앱 리스트를 받기 위해 리스너를 등록하는 방식입니다. BackgroundScanActivity 실행 전에 BackgroundScanActivity.SetScanListener(ScanResultListener);를 통해 리스너 등록을 해줍니다.

```
BackgroundScanActivity.SetScanListener(new ScanResultListener() {
    @Override
    public void onScanComplete(ArrayList<Map> arrayList) {
        Map<String,String> info = new HashMap<String,String>();
        String pkgName="";
        String malType="";
        for (int i = 0; i < arrayList.size(); i++) {
            info = (Map)arrayList.get(i);
            pkgName = (String)info.get("packageName");
            malType = (String)info.get("malware_type");
            Log.d("", "PackageName : " + pkgName + "\n Type : " + malType);
        }
    }
});
```

3.5.5.2. FULL모드 리스너 등록

FULL모드 검사가 끝남과 동시에 탐지된 악성앱 리스트를 받기 위해 리스너를 등록하는 방식입니다. ScanActivity 실행 전에 ScanActivity.SetScanListener(ScanResultListener);를 통해 리스너 등록을 해줍니다.

```
ScanActivity.SetScanListener(new ScanResultListener() {
    @Override
    public void onScanComplete(ArrayList<Map> arrayList) {
        Map<String,String> info = new HashMap<String,String>();
        String pkgName="";
        String malType="";
        for (int i = 0; i < arrayList.size(); i++) {
            info = (Map)arrayList.get(i);
            pkgName = (String)info.get("packageName");
            malType = (String)info.get("malware_type");
            Log.d("", "PackageName : " + pkgName + "\n Type : " + malType);
        }
    }
});
```


3.5.5.3. getScanResult API

검사 완료 후 탐지된 악성앱 리스트를 원하는 시점에 받기 위해 API를 호출하는 방식입니다.
해당 API 호출 시점에 탐지된 악성앱 리스트를 리턴 합니다.

```
import com.TouchEn.mVaccine.b2b2c.util.CommonUtil; // import CommonUtil 클래스

ArrayList<Map> arrayList;
Map<String,String> info = new HashMap<String,String>();

// API 호출
arrayList = CommonUtil.getScanResult(this,true);
String pkgName="";
String malType="";
for (int i = 0; i < arrayList.size(); i++) {
    info = (Map)arrayList.get(i);
    pkgName = (String)info.get("packageName");
    malType = (String)info.get("malware_type");
    Log.d("", "PackageName : " + pkgName + "\n Type : " + malType);
}
```

getScanResult API 파라미터

속성	설명
Context	Context
ScanPhishing	보이스 피싱 탐지 사용 여부

3.5.6. 백그라운드 앱 강제 종료 시 Notification 제거

구 버전 Android OS에서는 앱을 강제종료 하게 되면 onDestroy가 호출 안되는 경우가 있습니다.
이를 해결하기 위해서는 Service를 이용하여 Task가 종료되었을 때를 확인해 Notification을 제거해 줍니다.

Android 9.0 이상에서 메모리 관리를 위해 백그라운드에 있는 앱을 절전모드로 유지하게 되어있습니다.
앱 절전 모드로 전환 시 Notification 통제가 불가능해지면 Notification을 제거해 줍니다.

3.5.6.1. 등록 가이드

1. Service 를 AndroidManifest 에 등록합니다. 반드시 stopWithTask 속성을 false 로 설정해 줍니다.

```
...
<service
    android:name="com.TouchEn.mVaccine.b2b2c.service.RemoveNotificationService"
    android:stopWithTask="false" />
...
```

2. 초기 Activity의 onCreate에서 서비스를 실행시켜 줍니다.

```
...
public void onCreate(Bundle savedInstanceState) {
    this.startService(new Intent(this, RemoveNotificationService.class));
}
...
```

3.5.7. Notification 사용 설정 여부 확인

Android13부터 알림(Notification) 런타임 권한 사항이 변경되어 사용자로부터 알림 권한을 승인 받아 Notification 기능을 제공할 수 있습니다.

이에 따라, 알림 허용 권한이 없는 경우에 Notification 이 뜨지 않는 현상이 발생합니다.

Android 13(API 수준 33)으로 빌드 된 앱은 AndroidManifest.xml에 POST_NOTIFICATIONS 권한을 추가하여 알림 권한을 원하는 시점에 요청할 수 있습니다.

하지만, Android 12L(API 수준 32) 이하로 빌드 된 앱은 POST_NOTIFICATIONS과는 관계없이 Notification이 최초로 notify 되는 시점에 알림 허용에 대한 권한 요청을 하게 되며 사용자가 알림 권한에 대해 허용하지 않았을 경우 사용자가 앱을 제거한 후 재설치 하거나 Android13으로 빌드 된 앱으로 업데이트 하지 않는 이상 권한 요청 메시지가 다시 표시되지 않습니다.

이에 대해 mVaccine은 Notification 사용 설정 여부를 확인할 수 있는 두 가지 방법을 제공합니다.

1. 알림 사용이 불가능한 경우에 code 반환
2. 알림 사용 가능 여부 체크 API 제공

두 가지 방법을 이용해 Notification 사용이 불가능 함을 전달하여, 앱이 알림 허용 권한 문제에 대해 대응할 수 있도록 지원합니다.

3.5.7.1. POST_NOTIFICATIONS 권한 추가

Android 13(API 수준 33)에서는 앱에서 예외 없는 알림을 보내기 위해서는 새로운 런타임 권한 POST_NOTIFICATIONS을 선언해야 합니다.

AndroidManifest.xml 에 다음과 같이 POST_NOTIFICATIONS 권한을 추가합니다.

```
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
```

3.5.7.2. 알림 사용이 불가능한 경우에 code 반환

mVaccine에서는 Notification을 사용하는 기능 실행 시, NOTI가 뜨는 시점에 알림 허용 권한 체크를 하여 알림 사용 설정이 되어있지 않을 경우에 POST_NOTI_NO(1300)를 CodeReceiver를 통해 반환합니다. (Android 13 이상만 알림 사용 설정을 확인합니다.)

```
public class CodeReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {
        Uri data = intent.getData();
        String type = intent.getStringExtra("type");
        int i = intent.getIntExtra("result", 0);

        Log.e("CodeReceiver", "result = " + i);

        switch (i)
        {
            .....
            .....
            case com.secureland.smartmedic.core.Constants.POST_NOTI_NO: //1300
                // 노티가 떴는지 않은 경우
                Log.e("CodeReceiver",
                    "com.secureland.smartmedic.core.Constants.POST_NOTI_NO");
                break;
        }
    }
}
```

반환된 POST_NOTI_NO를 통해 앱은 알림 사용이 불가능한 상황임을 인지하여 대응해야 합니다. (CODE 결과에 따른 동작은 mVaccine이 아닌 결과를 받은 앱에서 결정합니다.)

3.5.7.3. 알림 사용 가능 여부 체크 API 제공

mVaccine에서는 NOTI가 뜨는 시점에 알림 허용 권한 체크를 하는 것 대신 미리 알림 사용이 가능한지 체크할 수 있도록 API를 제공합니다.

앱은 CommonUtil.checkNotiPermission 함수를 사용하여, mVaccine 기능을 실행시키지 않아도 알림 사용

가능 여부를 확인할 수 있습니다.

```
if(!CommonUtil.checkNotiPermission(this.getApplicationContext())){
    // 알림에 대한 권한이 없는 경우
}
```

3.5.7.4. 사용 예시

mVaccine 에서 제공하는 알림 사용 여부 체크 결과를 통해 앱은 그 다음 동작을 결정해야 합니다.

여기서 mVaccine 은 악성앱 mini 모드 검사 시, 알림 사용 가능 여부를 확인하는 사용 예시를 보여줍니다.

Information

제공하는 알림 사용 여부 체크 결과에 따른 동작은 mVaccine에서 제공하는 영역이 아닌 앱에 따라 상이하게 구현되는 부분이므로 **제시된 예시는 구현 참고용으로만 사용**해야 합니다.

다음 예시는 mVaccine 에서 제공하는 데모를 기준으로 작성되었습니다.

[동작 방식 요약] Notification을 사용하는 기능 동작 전에 CommonUtil.checkNotiPermission를 이용해 알림 사용 가능 여부를 체크하고 허용 권한을 요청하거나 사용자에게 권한이 없음을 알립니다.

1. BackgroundScan 을 이용하는 mini 모드 동작 전에 CommonUtil.checkNotiPermission 을 이용해 알림 사용 여부를 체크하고 targetSDK 버전에 따라 알림 권한을 요청하거나 사용자에게 권한이 없음을 dialog 를 통해 알립니다..

※ getNotiPermissionSample 함수는 알림 권한 검사 이후 동작에 대한 예시로 제공하는 것으로 필수적으로 적용해야하는 함수가 아닙니다

//노티 권한이 없을 시, mini 실행 제어 방식

```
if(!CommonUtil.checkNotiPermission(this.getApplicationContext())){
    if( Build.VERSION.SDK_INT >=33 ) {
        ActivityCompat.requestPermissions(this, new
            String[]{"android.permission.POST_NOTIFICATIONS"},12233);
    } else {
        getNotiPermissionSample(); //사용자에게 권한이 없음을 dialog 를 통해
        알리는 함수 샘플
    }
}
```

```

        return;
    }
    mini(); // 백신 액티비티에서 약성코드,루팅 탐지

```

<getNotiPermissionSample 함수>

- 앱이 targetSDK 33 이상으로 빌드 된 경우: 이전에 최초 알림 권한 요청에 대해 거절한 적이 있는 경우 알림 권한을 다시 요청합니다.
만약 두 번 이상 알림 권한 요청에 대해 거절하여 더 이상 OS 를 통한 알림 권한 요청을 할 수 없는 경우 알림 권한을 수동으로 설정하도록 요청하는 dialog 가 뜨도록 합니다.
사용자는 알림 권한 수동 설정 요청 dialog 을 통해 알림 권한 창으로 이동하여 직접 알림을 허용할 수 있고, 수동 설정에 대해서도 거절할 경우 toast 메시지를 통해 사용자에게 알림 권한이 없어 제공되는 기능에 제한이 있음을 명시하도록 예시를 구성하였습니다.

앱이 targetSDK 33 미만으로 빌드 된 경우: OS 를 통한 알림 권한 요청을 할 수 없으므로 알림 권한을 수동으로 설정하도록 요청하는 dialog 가 뜨도록 합니다.
사용자는 알림 권한 수동 설정 요청 dialog 을 통해 알림 권한 창으로 이동하여 직접 알림을 허용할 수 있고, 수동 설정에 대해서도 거절할 경우 toast 메시지를 통해 사용자에게 알림 권한이 없어 제공되는 기능에 제한이 있음을 명시하도록 예시를 구성하였습니다.

```

public void getNotiPermissionSample(){
    //TARGET33 은 POST_NOTIFICATIONS 권한을 통한 요청 먼저 진행
    if (Build.VERSION.SDK_INT >= 33) {
        if (ActivityCompat.shouldShowRequestPermissionRationale(MainActivity_sample.this,
            "android.permission.POST_NOTIFICATIONS")) {
            //알림 권한 요청에 한 번 거절한 경우 재요청
            ActivityCompat.requestPermissions(this, new
            String[]{"android.permission.POST_NOTIFICATIONS"}, 12233);
            return;
        }
    }

    //알림 권한 요청 두 번 이상 거절한 경우 수동으로 권한 변경 요청
    AlertDialog.Builder builder = new AlertDialog.Builder(this);
    builder.setTitle("권한 설정");
    builder.setMessage("알림 권한이 없습니다. \n 권한 거절로 인해 일부 기능이 제한됩니다.");
    builder.setPositiveButton("권한 설정하러 가기", new DialogInterface.OnClickListener() {

```

```

@Override
public void onClick(DialogInterface dialogInterface, int i) {
    try {
        Intent intent = null;
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            intent = new Intent(Settings.ACTION_APP_NOTIFICATION_SETTINGS);
            intent.putExtra(Settings.EXTRA_APP_PACKAGE, getPackageName());
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        } else {
            intent = new Intent(Settings.ACTION_APPLICATION_DETAILS_SETTINGS);
            intent.addCategory(Intent.CATEGORY_DEFAULT);
            intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            intent.setData(Uri.parse("package:" + String.valueOf(getPackageName())));
        }
        startActivity(intent);
    } catch (Exception e) {
        e.printStackTrace();
        Intent intent = new Intent(Settings.ACTION_MANAGE_APPLICATIONS_SETTINGS);
        intent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
        startActivity(intent);
    }
}

});
builder.setNegativeButton("취소", new DialogInterface.OnClickListener() {
    @Override
    public void onClick(DialogInterface dialogInterface, int i) {
        Toast.makeText(getApplicationContext(), "알림 권한 설정이 취소되었습니다. 일부 기능이 제한됩니다.", Toast.LENGTH_SHORT).show();
    }
});
builder.create().show();
}

```

2. 그리고 권한이 없는 경우 mVaccine 의 기능이 동작하지 않도록 합니다.
 (해당 데모에서는 mVaccine 의 기능을 mini 모드로 작성하였습니다.)

//노티 권한이 없을 시, mini 실행 제어 방식

```

if(!CommonUtil.checkNotiPermission(this.getAppContext())){
    if( Build.VERSION.SDK_INT >=33 ) {
        ActivityCompat.requestPermissions(this, new
        String[]{"android.permission.POST_NOTIFICATIONS"},12233);
    } else {
        getNotiPermissionSample(); //사용자에게 권한이 없음을 dialog 를 통해
알리는 함수 샘플
    }
    return;
}
mini(); // 백신 액티비티에서 악성코드,루팅 탐지

```

3. TargetSDK33으로 빌드 된 앱의 경우에는 ActivityCompat.requestPermissions를 통해 권한을 허용 받았을 때와 허용 받지 못했을 때에 대한 동작도 정의할 수 있습니다.

(해당 데모에서는 알림 권한이 없을 경우 발생할 수 있는 기능 제약에 대해 설명하고, 권한을 재요청 하도록 구성하였습니다.)

@Override

```

public void onRequestPermissionsResult(int requestCode, @NonNull String[] permissions,
@NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);

    if(grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED){
        //permission was granted
    }else{
        //permission was denied
        AlertDialog.Builder builder = new AlertDialog.Builder(this);
        builder.setTitle("알림 권한 요청");
        builder.setMessage("알림 권한이 없는 경우, \n 백신 기능 사용에 제약이
발생합니다. \n 알림 권한을 허용해주시길 부탁드립니다.");
        builder.setPositiveButton("확인", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialogInterface, int i) {
                //권한 재요청 샘플
                getNotiPermissionSample();
            }
        });
    }
}

```

```

        builder.create().show();
    }
}
}

```

4. API 목록

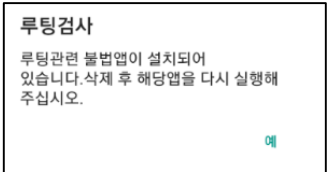
mVaccine 샘플 프로젝트에 적용되어 있는 메서드입니다.

함수	내용
mini()	UI 간소화 모드로 악성코드 스캔
full()	UI 모드로 악성코드 스캔
uiFull()	상세한 UI 모드로 악성코드 스캔
startActivityResult()	백신 실행 결과 Code 값 체크
onActivityResult()	백신 실행 결과 처리를 위한 액티비티 메서드
onDestroy()	백신 종료
rootingCheck()	루팅 체크

5. API 상세

mVaccine 샘플 프로젝트에 적용되어 있는 메서드에 대한 상세 설명입니다.

5.1. mini()

구분	내용	
목적	mVaccine 구동 - UI간소화 모드	
사용법	mini()	
파라미터	N/A	
결과	성공 - mVaccine UI간소화 모드 구동 실패 - NULL	
옵션	intent 옵션에 따른 구동 설정이 가능 (샘플코드 옵션을 권장함)	
	속성	설명
	useBlackAppCheck 	true: 루팅 관련 우회 앱 검사 (루팅 앱으로 탐지 시 rootingexitapp, rootingyes, rootingyesorno 옵션 무시) false: 검사 안 함 default : true

	scan_rooting	true: 단말 루팅 여부 검사
		false: 검사 안 함
		default : true
	scan_package	true: 악성코드 앱 검사
		false: 검사 안 함
		default : true
	useDualEngine	true: BitDefender 엔진 검사
		false: 검사 안 함
		default : false
	backgroundScan (mini 전용)	true: 악성코드 백그라운드 검사
		false: 검사완료 시 까지 UI유지
		default : true
	rootingexitapp	true: 루팅 단말일 경우 앱 종료 (onActivityResult에 구현 필요 *비고 확인)
		false: default 값으로 적용
		default: rootingYesOrNo 옵션
	rootingyesorno	true: 경고 메시지 후 악성 앱 스캔 종료/취소
		false: default 값으로 적용
		default: rootingYesOrNo 옵션
	rootingyes	true: 경고 메시지 후 악성 앱 스캔 계속 진행
		false: default 값으로 적용
		default: rootingYesOrNo 옵션
	show_update	true: 패턴/엔진 업데이트 시 토스트 보임
		false: 안보임
		default : true
	show_license	true: 라이선스 업데이트 UI를 출력
		false: 안보임
		default : true

<p>show_notify (mini 전용)</p>  TouchEn mVaccine TouchEn mVaccine TouchEn mVaccine이 구동 중입니다.	<p>true: 검사 간 noti피케이션 표시</p> <p>false: 미 표시</p> <p>default : true</p>
<p>notifyAutoClear (mini 전용)</p>	<p>true: 검사 후 noti피케이션 자동 종료</p> <p>false: 미 종료</p> <p>default : false</p>
<p>notifyClearable (mini 전용)</p>	<p>true: Swipe를 통해 noti피케이션 삭제 가능</p> <p>false: noti피케이션 삭제 불가능</p> <p>default : false</p>
<p>show_toast</p>	<p>true: 토스트 출력</p> <p>false: 미 출력</p> <p>default : true</p>
<p>show_warning</p> <p>검사 설정이 되지 않았습니다.</p>	<p>true: NW연결/검사설정 실패 경고 토스트 출력</p> <p>false: 미 출력</p> <p>default : true</p>
<p>show_about (mini 전용)</p> 	<p>true: noti 클릭 시 연동 앱 / 버전 정보 화면 표시</p> <p>false: 미 출력</p> <p>default : true</p>
<p>show_scan_ui (mini 전용)</p> 	<p>true: 검사 중 이미지 출력</p> <p>false: 미 출력</p> <p>default : true</p>
<p>showBlackAppName</p> 	<p>true: 루팅 앱 탐지 시 앱명 출력</p> <p>false: 루팅 앱 탐지 시 앱명 미 출력</p>

		default : true
	show_badge (mini 전용)	true: 아이콘에 뱃지 표시
		false: 미 출력
		default : false
	bg_rooting (mini 전용)	true: 루팅 검사를 백그라운드 처리
		false: 루팅 검사 포그라운드 처리
		default : true
	rooting_internal_check	1 (Constants.RT_INTERNAL_FILE) : 파일 상세 검사
		2 (Constants.RT_INTERNAL_MNT) : 마운트 검사
		3 (Constants.RT_INTERNAL_ALL) : 모두 검사
		0: 기본 검사
		default: 0
	v_so	true: 엔진/DB so 무결성 검사 수행
		false: 엔진/DB so 무결성 검사 미수행
		default : true
	rooting_delay_time	N초: 루팅 탐지 N초 후 앱 자동 종료
		default: 사용 안 함
	scan_heuristic <small>(구글의 구동중인 앱 정보 미지원으로 인한 지원종료)</small>	true: 휴리스틱 검사
		false: 검사안함
		default : false
	scan_phishing	true: 보이스 피싱 악성앱 검사
		false: 보이스 피싱 악성앱 검사 안 함
		default : true
예제	// Activity 사용 API public void mini() { Intent i = new Intent(this , BackgroundScanActivity. class); i.putExtra("useBlackAppCheck", true); i.putExtra("scan_rooting", true); i.putExtra("scan_package", true); i.putExtra("useDualEngine", true); i.putExtra("backgroundScan", true); i.putExtra("rootingexitapp", true); i.putExtra("show_update", false); i.putExtra("show_license", false); i.putExtra("show_notify", false); }	

	<pre> i.putExtra("show_toast", false); i.putExtra("show_warning", false); i.putExtra("show_scan_ui", true); i.putExtra("show_badge", false); i.putExtra("bg_rooting", true); i.putExtra("show_about ", true); i.putExtra("scan_phishing", true); this.startActivityForResult(i, REQUEST_CODE); } </pre>
	<pre> // Thread 사용 API public void mini() { Bundle bundle = new Bundle(); bundle.putBoolean("useBlackAppCheck", true); bundle.putBoolean("scan_rooting", true); bundle.putBoolean("scan_package", true); bundle.putBoolean("useDualEngine", true); bundle.putBoolean("rootingexitapp", true); bundle.putBoolean("show_update", false); bundle.putBoolean("show_license", false); bundle.putBoolean("show_notify", false); bundle.putBoolean("show_scan_ui", true); CommonUtil.scanMini(getApplicationContext(), bundle); } </pre>
관련 함수	-
비고	<p>- rooting_internal_check: 확장성을 위해 int 자료형을 사용합니다. 2번 '마운트검사'에 대한 사용 방법은 [부록]FAQ를 참고 바랍니다.</p> <p>- rooting_delay_time: 루팅 탐지 N 초 후 앱 종료. 다른 루팅 옵션보다 우선적으로 적용됩니다. OS 환경에 따라 앱이 종료 후 재실행 될 수 있습니다.</p> <p>- rootingexitapp: 라이브러리에서 앱으로 onActivityResult를 통해 종료시키라는 결과값을 전달. 적용 시 onActivityResult에 아래와 같이 종료 로직 구현 필요합니다.</p> <div style="background-color: #f4a460; padding: 5px; text-align: center;">예제</div> <pre> public void onActivityResult(int requestCode, int resultCode, Intent intent) { super.onActivityResult(requestCode, resultCode, intent); if (requestCode == 777) { if (resultCode == com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP) this.finish(); } } </pre> <p>- show_badge: notifyClearable 옵션이 true일 경우 모든 경우에서 정상 동작하지만,</p>

	<p>notifyClearable 옵션이 false일 경우 OS, 런처 정책에 따라 동작사항이 결정됩니다.</p> <ul style="list-style-type: none"> - show_update: 'backgroundScan' 옵션이 false일 경우 동작합니다. 해당 옵션이 'true'일 경우, 최신 버전인 경우만 체크하여 토스트로 표시합니다. - show_license: scan_rooting, backgroundScan 이 false일 경우 동작하지 않습니다. - backgroundScan: scan_rooting이 true일 경우 동작하지 않습니다(false). <p>useDualEngine이 true일 경우 강제로 동작합니다(true)</p>
--	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

5.2. full()

구분	내용	
목적	mVaccine 구동 - UI 모드	
사용법	full()	
파라미터	N/A	
결과	<p>성공 - mVaccine UI 모드 구동</p> <p>실패 - NULL</p>	
옵션	intent 옵션에 따른 구동 설정이 가능 (mini 모드를 우선 권장)	
	속성	설명
	useBlackAppCheck 	true: 루팅 관련 우회 앱 검사 (루팅 앱으로 탐지 시 rootingexitapp, rootingyes, rootingyesorno 옵션 무시)
		false: 검사 안 함
		default : true
	scan_rooting	true: 단말 루팅 여부 검사
		false: 검사 안 함
		default : true
	scan_package	true: 악성코드 앱 검사
		false: 검사 안 함
		default : true
	useDualEngine	true: BitDefender 엔진 검사
		false: 검사 안 함
		default : false
	dualEngineBackground (full 전용)	true: BitDefender 엔진 백그라운드 검사
		false: 검사 완료 시까지 UI유지
		default : true

backgroundJobForLongTime (full 전용)	true: 악성코드 검사 간 5초 경과 90% 미만 검사 시 백그라운드 구동 여부 띄움
	false: 백그라운드 구동 여부 안 띄움
	default : false
useStopDialog (full 전용)	true: 검사 중지 버튼 표시
	false: 검사 중지 버튼 미 표시
	default : true
rootingexitapp	true: 루팅 단말일 경우 앱 종료 (onActivityResult에 구현 필요 *비고 확인)
	false: default 값으로 적용
	default: rootingYesOrNo 옵션
rootingyesorno	true: 경고 메시지 후 악성 앱 스캔 종료/취소
	false: default 값으로 적용
	default: rootingYesOrNo 옵션
rootingyes	true: 경고 메시지 후 악성 앱 스캔 계속 진행
	false: default 값으로 적용
	default: rootingYesOrNo 옵션
show_update	true: 패턴/엔진 업데이트 시 토스트 보임
	false: 안보임
	default : true
show_license	true: 라이선스 업데이트 UI를 출력
	false: 안보임
	default : true
show_toast	true: 토스트 출력
	false: 미 출력
	default : true
show_warning	true: 경고 토스트 출력
	false: 미 출력
	default : true
showBlackAppName	true: 루팅 앱 탐지 시 앱명 출력

	<div> <div>루팅검사</div> <div>'SuperSU' 루팅관련 불법앱이 설치되어 있습니다. 삭제 후 해당 앱을 다시 실행해 주십시오.</div> <div>예</div> </div>	false: 루팅 앱 탐지 시 앱명 미 출력 default : true
		1 (Constants.RT_INTERNAL_FILE) : 파일 상세 검사 2 (Constants.RT_INTERNAL_MNT) : 마운트 검사 3 (Constants.RT_INTERNAL_ALL) : 모두 검사 0: 기본검사 default : 0
	rooting_internal_check	
	v_so	true: 엔진/DB so 무결성 검사 수행 false: 엔진/DB so 무결성 검사 미수행 default : true
	rooting_delay_time	N초: 루팅 탐지 N초 후 앱 자동 종료 default: 사용 안 함
	scan_heuristic (구글의 구동중인 앱 정보 미지원으로 인한 지원종료)	true: 휴리스틱 검사 false: 검사안함 default : false
	scan_phishing	true: 보이스 피싱 악성앱 검사 false: 보이스 피싱 악성앱 검사 안 함 default : true
예제	<pre> public void full() { Intent i = new Intent(this, ScanActivity.class); i.putExtra("useBlackAppCheck", true); i.putExtra("scan_rooting", true); i.putExtra("scan_package", true); i.putExtra("useDualEngine", true); i.putExtra("dualEngineBackground", true); i.putExtra("backgroundJobForLongTime", true); i.putExtra("rootingexitapp", true); i.putExtra("show_update", false); i.putExtra("show_license", false); i.putExtra("show_rooting", false); i.putExtra("show_toast", false); i.putExtra("show_warning", false); i.putExtra("scan_phishing", false); this.startActivityForResult(i, REQUEST_CODE); } </pre>	
관련 함수	-	
비고	- rooting_internal_check: 확장성을 위해 Integer 자료형을 사용합니다. 2번 '마운트검사'에 대한 사용 방법은 [부록]FAQ를 참고 바랍니다. - rooting_delay_time: 루팅 탐지 N 초 후 앱 종료. 다른 루팅 옵션보다 우선적으로 적용됩니다. OS 환경에 따라 앱이 종료 후 재실행 될 수 있습니다.	

- rootingexitapp: 라이브러리에서 앱으로 onActivityResult를 통해 종료시키라는 결과값을 전달. 적용시 onActivityResult에 아래와 같이 종료 로직 구현 필요합니다.

예제

```
public void onActivityResult(int requestCode, int resultCode,
    Intent intent) {
    super.onActivityResult(requestCode, resultCode,
    intent);
    if (requestCode == 777) {
        if (resultCode ==
        com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP)
            this.finish();
    }
}
```

- backgroundJobForLongTime 옵션은 useStopDiaog = true상태에서만 동작합니다.

5.3. uifull()

구분	내용																	
목적	mVaccine 구동 - UI FULL 모드																	
사용법	uiFull()																	
파라미터	N/A																	
결과	성공 - mVaccine UI FULL 모드 구동 실패 - NULL																	
옵션	intent 옵션에 따른 구동 설정이 가능 (mini 모드를 우선 권장)																	
	<table> <tr> <th>속성</th><th>설명</th></tr> <tr> <td rowspan="3"> useBlackAppCheck  </td><td>true: 루팅 관련 우회 앱 검사 (루팅 앱으로 탐지 시 rootingexitapp, rootingyes, rootingyesorno 옵션 무시)</td></tr> <tr> <td>false: 검사 안 함</td></tr> <tr> <td>default : true</td></tr> <tr> <td rowspan="3"> scan_rooting </td><td>true: 단말 루팅 여부 검사</td></tr> <tr> <td>false: 검사 안 함</td></tr> <tr> <td>default : true</td></tr> <tr> <td rowspan="3"> scan_package </td><td>true: 악성코드 앱 검사</td></tr> <tr> <td>false: 검사 안 함</td></tr> <tr> <td>default : true</td></tr> <tr> <td rowspan="3"> useDualEngine </td><td>true: BitDefender 엔진 검사</td></tr> <tr> <td>false: 검사 안 함</td></tr> <tr> <td>default : false</td></tr> </table>	속성	설명	useBlackAppCheck 	true: 루팅 관련 우회 앱 검사 (루팅 앱으로 탐지 시 rootingexitapp, rootingyes, rootingyesorno 옵션 무시)	false: 검사 안 함	default : true	scan_rooting	true: 단말 루팅 여부 검사	false: 검사 안 함	default : true	scan_package	true: 악성코드 앱 검사	false: 검사 안 함	default : true	useDualEngine	true: BitDefender 엔진 검사	false: 검사 안 함
속성	설명																	
useBlackAppCheck 	true: 루팅 관련 우회 앱 검사 (루팅 앱으로 탐지 시 rootingexitapp, rootingyes, rootingyesorno 옵션 무시)																	
	false: 검사 안 함																	
	default : true																	
scan_rooting	true: 단말 루팅 여부 검사																	
	false: 검사 안 함																	
	default : true																	
scan_package	true: 악성코드 앱 검사																	
	false: 검사 안 함																	
	default : true																	
useDualEngine	true: BitDefender 엔진 검사																	
	false: 검사 안 함																	
	default : false																	

rootingexitapp	<div> <div>루팅검사</div> <div>루팅폰에서는 서비스를 이용하실 수 없습니다.</div> <div>예</div> </div>	true: 루팅 단말일 경우 앱 종료 (onActivityResult에 구현 필요 *비고 확인) false: default 값으로 적용 default: rootingYesOrNo 옵션
rootingyesorno	<div> <div>루팅검사</div> <div>루팅폰입니다. 검사를 중지하실려면 예를 검사를 계속 진행하실려면 아니오를 누르세요</div> <div>아니요 예</div> </div>	true: 경고 메시지 후 악성 앱 스캔 종료/취소 false: default 값으로 적용 default: rootingYesOrNo 옵션
rootingyes	<div> <div>루팅검사</div> <div>루팅폰입니다. 스마트폰 보안에 영향을 주는 구조변경(탈옥,루팅)을 한폰에서 발생한 문제는 자사에서는 책임지지 않습니다.</div> <div>예</div> </div>	true: 경고 메시지 후 악성 앱 스캔 계속 진행 false: default 값으로 적용 default: rootingYesOrNo 옵션
show_update	<div> <div>최신 패턴을 사용 중입니다.</div> </div>	true: 패턴/엔진 업데이트 시 토스트 보임 false: 안보임 default : true
show_license	<div> <div>라이센스를 체크하는 중입니다.</div> </div>	true: 라이선스 업데이트 UI를 출력 false: 안보임 default : true
show_toast		true: 토스트 출력 false: 미 출력 default : true
show_warning	<div> <div>검사 설정이 되지 않았습니다.</div> </div>	true: 경고 토스트 출력 false: 미 출력 default : true
showBlackAppName	<div> <div>루팅검사</div> <div>'SuperSU' 루팅관련 불법앱이 설치되어 있습니다.삭제 후 해당앱을 다시 실행해 주십시오.</div> <div>예</div> </div>	true: 루팅 앱 탐지 시 앱명 출력 false: 루팅 앱 탐지 시 앱명 미 출력 default : true
rooting_internal_check		1 (Constants.RT_INTERNAL_FILE) : 파일 상세 검사 2 (Constants.RT_INTERNAL_MNT) : 마운트 검사 3 (Constants.RT_INTERNAL_ALL) : 모두 검사 0: 기본검사 default : 0
v_so		true: 엔진/DB so 무결성 검사 수행

		false: 엔진/DB so 무결성 검사 미수행
		default : true
	rooting_delay_time	N초: 루팅 탐지 N초 후 앱 자동 종료
		default: 사용 안 함
	scan_heuristic (구글의 구동중인 앱 정보 미지원으로 인한 지원종료)	true: 휴리스틱 검사 false: 검사안함 default : false
예제	<pre> public void uiFull() { Intent i = new Intent(this, UiScanActivity.class); i.putExtra("useBlackAppCheck", true); i.putExtra("scan_rooting", true); i.putExtra("scan_package", true); i.putExtra("useDualEngine", true); i.putExtra("rootingexitapp", true); i.putExtra("show_update", false); i.putExtra("show_license", false); i.putExtra("show_rooting", false); i.putExtra("show_toast", false); i.putExtra("show_warning", false); this.startActivityForResult(i, REQUEST_CODE); } </pre>	
관련 함수	-	
비고	<p>- rooting_internal_check: 확장성을 위해 Integer 자료형을 사용합니다. 2번 '마운트검사'에 대한 사용 방법은 [부록]FAQ를 참고 바랍니다.</p> <p>- rooting_delay_time: 루팅 탐지 N 초 후 앱 종료. 다른 루팅 옵션보다 우선적으로 적용됩니다. OS 환경에 따라 앱이 종료 후 재실행 될 수 있습니다.</p> <p>- rootingexitapp: 라이브러리에서 앱으로 onActivityResult를 통해 종료시키라는 결과값을 전달. 적용시 onActivityResult에 아래와 같이 종료 로직 구현 필요합니다.</p> <div style="background-color: #f4a460; padding: 5px; text-align: center;">예제</div> <pre> public void onActivityResult(int requestCode, int resultCode, Intent intent) { super.onActivityResult(requestCode, resultCode, intent); if (requestCode == 777) { if (resultCode == com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP) this.finish(); } } </pre>	

5.4. startActivityForResult()

구분	내용
목적	백신탐검 결과 Code 값 얻기 인텐트로 액티비티를 실행시키고 결과값을 얻어오는 경우 사용

사용법	startActivityResult(i, REQUEST_CODE)
파라미터	i: 호출할 Activity가 포함된 인텐트 REQUEST_CODE: 백신상태 코드
결과	성공 - 백신구동 결과 값 실패 - NULL
옵션	N/A
예제	this.startActivityResult(i, REQUEST_CODE);
관련 함수	-
비고	-

5.5. onActivityResult()

구분	내용
목적	백신구동 결과에 따른 이후 처리 구현
사용법	onActivityResult(int requestCode, int resultCode, Intent intent)
파라미터	requestCode: 백신상태 코드 intent: 인텐트
결과	성공 - 백신상태 코드에 대한 분기처리 실패 - NULL
옵션	N/A
예제	<pre> @Override public void onActivityResult(int requestCode, int resultCode, Intent intent) { super.onActivityResult(requestCode, resultCode, intent); Log.d("mvc", "resultCode :" + resultCode); if (requestCode == 777) { switch(resultCode){ case com.secureland.smartmedic.core.Constants.ROOTING_EXIT_APP: case com.secureland.smartmedic.core.Constants.ROOTING_YES_OR_NO: case com.secureland.smartmedic.core.Constants.V_DB_FAIL: this.finish(); break; case com.secureland.smartmedic.core.Constants.EMPTY_VIRUS: break; case com.secureland.smartmedic.core.Constants.EXIST_VIRUS: break; case com.secureland.smartmedic.core.Constants.EXIST_VIRUS_CASE1: break; } } } </pre>

	<pre> case com.secureland.smartmedic.core.Constants. <i>EXIST_VIRUS_CASE2</i>: break; } } } </pre>
관련 함수	-
비고	-

5.6. onDestroy()

구분	내용
목적	mVaccine 종료
사용법	onDestroy()
파라미터	i: 호출할 Activity가 포함된 인텐트 REQUEST_CODE: 백신상태 코드
결과	성공 - 백신종료 실패 - NULL
옵션	N/A
예제	<pre> @Override protected void onDestroy() { super.onDestroy(); //mVaccine Notification 제거 Util.cancelMvcNotification(getApplicationContext()); //실시간 탐지, CodeReceiver 해제 try { unregisterReceiver(scanReceiver); } catch (IllegalArgumentException e) { //scanReceiver not registered } try { unregisterReceiver(codeReceiver); } catch (IllegalArgumentException e) { //codeReceiver not registered } // 종료 toast 메시지 표시 Toast.makeText(<i>this</i>, "<i>TouchEn_mVaccine</i>을 </pre>

	<pre> 종료합니다.Toast.LENGTH_SHORT).show(); // mVaccine 종료 this.finish(); } </pre>
관련 함수	-
비고	-

5.7. rootingCheck()

구분	내용
목적	단말기 루팅 여부 체크 및 루팅 관련 앱 체크 및 이름 확인
사용법	CommonUtil.checkRooting(Context, boolean, String[])
파라미터	context: 앱 컨텍스트 boolean: 루팅 관련 앱 true 체크/false 미 체크 String[]: 루팅 관련 앱 이름 int: internal 루팅 검사 시 사용 (0: 기본검사 / 1: 파일상세검사 / 2: 마운트 검사 / 3: 전체 검사) CListener : 검사 결과를 받기 위한 리스너
결과	성공 <ul style="list-style-type: none"> - 0: 루팅 되어 있지 않음 - 1: 루팅 됨 (루팅 탐지 방식에 따라 결과 값 다름) - 2, 3: 불법 앱의 활동(프로세스)가 감지됨 - 4: 불법 앱이 설치되어 있음 - 5: 앱 변조 시도가 감지됨 - 6: 백신 모듈 무결성 검증 실패 실패 <ul style="list-style-type: none"> - NULL
옵션	N/A
예제	<pre> public void rootingCheck(int internalCheckNum) { String message[] = new String[2]; CommonUtil.CListener cListener = new CommonUtil.CListener() { @Override public void onFinish(String result, String[] message) { showRootingCheckResult(result, message[0]); } } </pre>

```

};

CommonUtil.c(this, true, message, internalCheckNum, cListener);
}

private void showRootingCheckResult(String strIsRooting, String blackAppName)
{

    Handler handler = new Handler(Looper.getMainLooper());
    handler.post(new Runnable() {

        @Override
        public void run() {

            if (strIsRooting.equals("1")) {
                Log.d("mVaccine", "Rooting OK !!");
                Toast.makeText(getApplicationContext(), "루팅 단말 입니다.",
Toast.LENGTH_SHORT).show();
            }else if (strIsRooting.equals("2") || strIsRooting.equals("3")) {
                Log.d("mVaccine", "BlackApp Installed");
                Toast.makeText(getApplicationContext(), "루팅 관련 앱 활동이
탐지되었습니다.", Toast.LENGTH_SHORT).show();
            }else if (strIsRooting.equals("4")) {
                Log.d("mVaccine", "BlackApp Installed");
                Toast.makeText(getApplicationContext(), blackAppName+"루팅
우회 앱이 설치되어 있습니다.", Toast.LENGTH_SHORT).show();
            }else if (strIsRooting.equals("6")) {
                Log.d("mVaccine", "verify failed");
                Toast.makeText(getApplicationContext(), "무결성 검증에 실패
하였습니다..", Toast.LENGTH_SHORT).show();
            }else {
                Log.d("mVaccine", "No Rooting !!");
                Toast.makeText(getApplicationContext(), "루팅 단말이
아닙니다.", Toast.LENGTH_SHORT).show();
            }
        }
    });
}

```

관련 함수	-
비고	C0 함수 대체가능 (난독화명) / 권장

6. 에러 코드

에러	내용	
사이트 아이디/ 라이선스 키 미 입력 에러	에러 메시지	로그태그 SmartMedic에 key site, key 가 공백으로 나옴
	원인	스캔 시작 전 사이트 아이디 미 입력 시 발생
	표시 위치	로그캣
	해결 방법	사이트 아이디 정상 입력
라이선스 기간 종료	에러 메시지	"license expired" 토스트 메시지 발생
	원인	라이선스 만료일 초과
	표시 위치	스마트폰 내부에서 토스트 메시지로 표시
	해결 방법	라운시큐어 문의를 통해 라이선스 재발급
유효하지 않은 라이선스 키	에러 메시지	"license key is invalid" 토스트 메시지 발생
	원인	유효하지 않은 라이선스 키 사용
	표시 위치	콘솔 로그
	해결 방법	라이선스 아이디 체크
업데이트 에러	에러 메시지	"Site id is invalid or Network error" 로그 표시
	원인	잘못된 사이트 아이디 혹은 네트워크 에러
	표시 위치	로그캣
	해결 방법	사이트 아이디와 네트워크 연결상태 확인
패턴/엔진 무결성 검증 실패	에러 메시지	"verify failed signed [00000000]" "verify failed ck.m_hash XXXXXXXX" 로그 발생
	원인	패턴/엔진 무결성 검증 실패
	표시 위치	로그캣
	해결 방법	라운시큐어 문의
mVaccine 초기화 실패	에러 메시지	"initialize mvaccine failed" 로그 표시
	원인	패턴/엔진 무결성 검증 연속 실패
	표시 위치	로그캣
	해결 방법	라운시큐어 문의

7. 오픈소스 라이선스

TouchEn mVaccine Android 모듈에 사용되는 오픈소스는 없습니다.

[부록]

1. FAQ

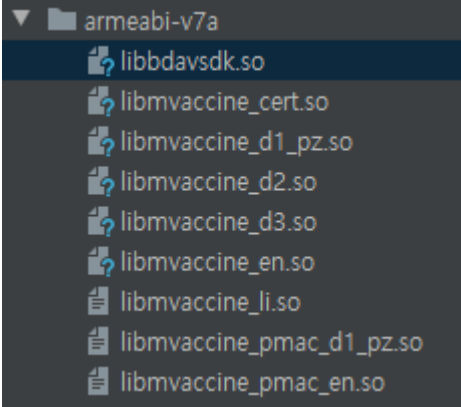
주로 문의되는 장애 해결 방법을 FAQ로 구성하였습니다.

1.1. 엔진 무결성 검증이 계속 실패

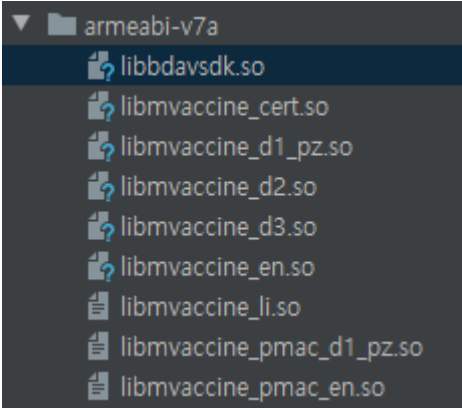
구분	내용
메시지	엔진 무결성 검증에 계속 실패할 경우 아래와 같은 로그가 발생 <pre>/data/user/0/com.TouchEn.mVaccine.DEMO/files/libsmartmedic-jni.so:mac D/mvaccine: verify failed signed [B@e85814b D/mvaccine: verify failed ck.m_hash[i] 1</pre>
원인	패키지 과정 중 mVaccine 엔진파일(SO)을 변조시키기 때문
해결방법	<ol style="list-style-type: none"> 1. build.gradle(app) 에 패키징 옵션추가 <div data-bbox="399 884 877 1041" data-label="Text"> <pre>packagingOptions { doNotStrip "**/*/*.so" }</pre> </div> 2. 재 빌드 android studio 메뉴 [Build] -> [Rebuild Project] => 정상적으로 적용이 되었다면 빌드 캐시가 남아 있어 제대로 반영이 안 될 수 있습니다. (JetBrain IDEA 특성상 캐시 이슈가 종종 발생하였습니다.) 따라서 반드시 ReBuild Project가 수행되어야 하며, 기존 APK를 삭제하고 다시 설치하여 테스트해야 합니다.

1.2. BitDefender BDAVSDK init 에러 발생


구분	내용
메시지	BDAVSDK init 할 때 UnsatisfiedLinkError가 발생합니다. <pre>java.lang.UnsatisfiedLinkError: Couldn't load bdaavsdk from loader dalvik.system.PathClassLoader[DexPathList[[zip fi at java.lang.Runtime.loadLibrary(Runtime.java:358) at java.lang.System.loadLibrary(System.java:526) at com.bitdefender.anti malware.BDAVSDK.init(BDAVSDK.java:42) at com.secureland.smartmedic.core.Util.bdInit(Util.java:1040) at com.secureland.smartmedic.core.SmartMedicUpdaterImpl.updateDualEngine(SmartMedicUpdaterImpl.java:1064) at com.TouchEn.mVaccine.b2b2c.activity.BackgroundScanActivity.update(BackgroundScanActivity.java:1460) at com.TouchEn.mVaccine.b2b2c.activity.BackgroundScanActivity.access\$1000(BackgroundScanActivity.java:79) at com.TouchEn.mVaccine.b2b2c.activity.BackgroundScanActivity\$MainThread.run(BackgroundScanActivity.java:1348)</pre>
원인	libbdaavsdk.so 파일이 jniLibs에 포함되지 않아 발생합니다.
해결방법	'libbdaavsdk.so' 파일을 정상적으로 추가합니다.

	
--	-----------------------------------------------------------------------------------

1.3. "initialize mVaccine failed" 에러 발생

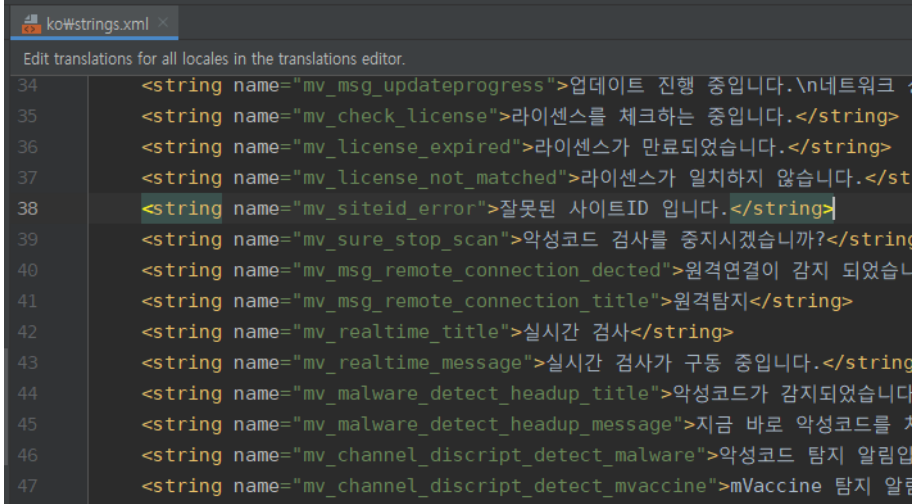
구분	내용
메시지	<p>mVaccine init 중 "initialize mvaccine failed" 에러 발생</p> <pre>initialize mvaccine failed</pre>
원인	mVaccine 엔진 로드 실패 시 발생합니다. (so파일이 없거나 무결성 검증실패)
해결방법	<p>mVaccine so 파일들이 정상적으로 추가되었는지 확인합니다.</p> 

1.4. "license expired" 토스트 메시지가 나타남

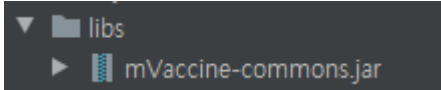
구분	내용
메시지	<p>백신 실행시 "license expired" 토스트 메시지가 나타남</p> 

원인	라이선스 기간이 만료 시 해당 메시지가 나타납니다.
해결방법	라운시큐어 문의를 통해 라이선스 재발급

1.5. “Resources\$NotFoundException” 에러 발생

구분	내용
메시지	<code>android.content.res.Resources\$NotFoundException: String resource ID #0x0</code>
원인	values/strings.xml 파일에 mVaccine 리소스가 포함되지 않아 발생합니다.
해결방법	values/strings.xml 파일에 mVaccine 리소스를 정상 포함시킵니다. 

1.6. ZipArchiveInputStream 클래스가 없다는 에러 발생

구분	내용
메시지	<code>Failed resolution of: Lorg/apache/commons/compress/archivers/zip/ZipArchiveInputStream;</code>
원인	‘mVaccine-commons.jar’ 파일이 libs에 포함되지 않아 발생합니다.
해결방법	‘mVaccine-commons.jar’ 파일을 libs 폴더에 포함시킵니다. 

1.7. Error inflating class: ... ProgressWheel 에러 발생

구분	내용
메시지	<code>Error inflating class com.TouchEn.mVaccine.b2b2c.util.ProgressWheel</code>
원인	해당 부분 난독화 처리가 안되어 있어 발생합니다.
해결방법	패키지명.R.styleable 에 대한 난독화 처리가 필요합니다.

```
-keep class com.TouchEn.mVaccine.b2b2c.demo.R$styleable{
    *;
}
```

1.8. 라이선스 만료 메시지를 변경

구분	내용
해결방법	<p>values/strings.xml 의 'mv_license_expired' 내용을 변경합니다. 단, mVaccine library 3.7.1.2298 버전부터 가능합니다.</p> <pre><string name="mv_license_expired">checking for license update</string></pre>

1.9. rooting_internal_check 옵션 2번 마운트검사 적용방법 (Constants.RT_INTERNAL_MNT)

구분	내용
해결방법	<p>1.AndroidManifest.xml에 MvIsolService를 등록합니다.</p> <pre><service android:name="com.TouchEn.mVaccine.b2b2c.service.MvcIsolService" android:enabled="true" android:isolatedProcess="true" android:process=":mvcIsolService" /></pre> <p>2.Application 클래스에 Constants.RT_INTERNAL_MNT 옵션을 넣고 initMvaccine() 함수를 호출합니다.</p> <pre>CommonUtil.initMvaccine(getApplicationContext(), Constants.RT_INTERNAL_MNT);</pre> <p>3. 앱이 아래와 같이Application 클래스를 사용할 경우, MvIsolService 프로세스에서 해당 클래스를 한번 더 호출합니다.</p> <p>[AndroidManifest.xml]</p> <pre><application android:name=".main.App" android:label="@string/mv_app_name"</pre> <p>[예제 : App.java]</p>

```
public class App extends Application {
    @Override
    public void onCreate() {
        super.onCreate();
    }
}
```

Application 클래스에 isolate 프로세스에서 사용하지 못하는 함수를 사용한다면 예외가 발생할 수 있기 때문에 예외 처리가 필요합니다.

예외 처리 방법은 두 가지 방법이 있습니다.

3.1. MvclsolService 프로세스 일 경우 Application의 onCreate 과정을 중지.

[CommonUtil.isMvclsolService 사용]

```
public class App extends Application {
    @Override
    public void onCreate() {
        super.onCreate();

        if(CommonUtil.isMvcIsolService(getApplicationContext()))
            return;
    }
}
```

3.2. isolate 프로세스에서 사용 불가능한 함수만 예외 처리

[CommonUtil.isIsolateService 사용]

```
if(!CommonUtil.isIsolateProcess(getApplicationContext())){

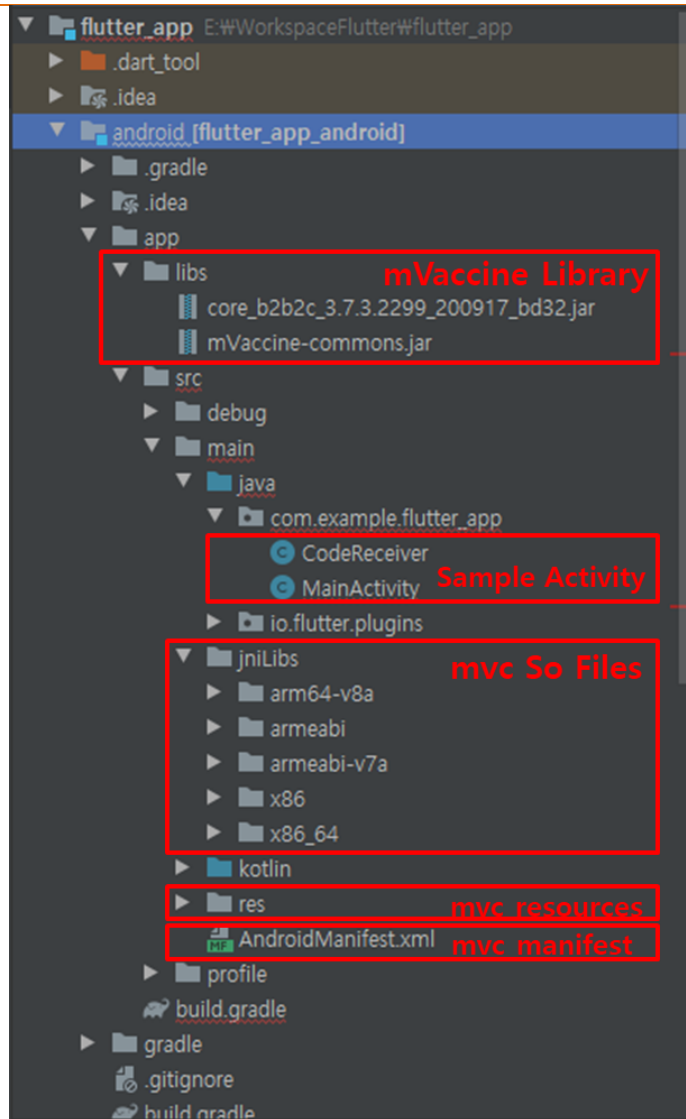
    try {
        ActivityManager activityManager = (ActivityManager)getApplicationContext()
            .getSystemService( s: "activity");
        activityManager.getRunningAppProcesses();
    } catch (SecurityException var2) {
    }
}
```

4. mini모드 실행시 해당옵션을 사용하여 루팅을 탐지합니다.

```
i.putExtra("rooting_internal_check",Constants.RT_INTERNAL_MNT);
```

1.10. Flutter에서 적용방법

구분	내용
해결방법	mVaccine Sample Project (01.module/./MVaccineDEMOForAndroidStudio) 를 참고 해서 mVaccine을 Flutter 프로젝트에 적용해 줍니다.



2. 프로젝트 상황에 맞게 Flutter 소스에서 mVaccine Android Native 스캔 함수를 호출 합니다. (full() 혹은 mini())

```
new MethodChannel(getFlutterView(), CHANNEL2).setMethodCallHandler(
    new MethodChannel.MethodCallHandler() {
        @Override
        public void onMethodCall(MethodCall call, MethodChannel.Result result) {
            if (call.method.equals("helloFromNativeCode")) {
                mini();
            }
        }
    });
```

1.11. 앱 강제 종료 후 Notification이 사라지지 않을 때

구분	내용
해결방법	Android OS에서는 앱을 강제종료 하게되면 onDestroy가 호출 안되는 경우가 있음

니다.

이를 해결하기 위해서는 Service를 이용하여 Task가 종료되었을 때를 확인해 Notification을 제거해 줍니다.

mVaccine에서는 아래 두 가지 방법을 제시해줍니다.

1. 라이브러리(v.3.8.0.2308 이상)에서 제공하는 ‘3.4.6. 백그라운드 앱 강제 종료 시 Notification 제거’ 를 참고하여 서비스를 등록합니다.

2. 서비스 직접 구현합니다.

2.1. Service를 상속받은 Class를 만들어 onTaskRemoved에 Notification 제거를 구현해 줍니다.

2.2. Android 9.0 이상에서 메모리 관리를 위해 백그라운드에 있는 앱을 절전모드로 유지하게 되어있습니다. 앱 절전 모드로 전환 시 Notification 통제가 불가능해지면 Notification을 제거해 주기 위해 onDestroy에도 Notification 제거를 구현 해 줍니다.

```
public class CheckProcessService extends Service {
    private final static int MESSAGE_ID = 12345;
    private final static int MESSAGE_ID1 = 123456;

    ...

    @Override
    public void onTaskRemoved(Intent rootIntent) {
        super.onTaskRemoved(rootIntent); //서비스 종료
        NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        mNotificationManager.cancel(MESSAGE_ID);
        mNotificationManager.cancel(MESSAGE_ID1);
        stopSelf(); //서비스 종료
    }

    @Override
    public void onDestroy() {
        super.onDestroy(); //서비스 종료
        NotificationManager mNotificationManager =
        (NotificationManager) getSystemService(NOTIFICATION_SERVICE);
        mNotificationManager.cancel(MESSAGE_ID);
    }
}
```

```
mNotificationManager.cancel(MESSAGE_ID1);
stopSelf(); //서비스 종료
}
}
```

2.2. Service를 AndroidManifest에 등록합니다. 반드시 stopWithTask속성을 false로 설정해 줍니다.

```
...
<service
    android:name=".service.CheckProcessService"
    android:stopWithTask="false" />
...
```

2.3. 초기 Activity의 onCreate에서 서비스를 실행시켜 줍니다.

```
...
public void onCreate(Bundle savedInstanceState) {
    this.startService(new Intent(this, CheckProcessService.class));
}
...
```

1.12. Android13 기기에서 Notification이 뜨지 않을 때

구분	내용
해결방법	<p>Android13부터 알림(Notification) 런타임 권한 사항이 변경되어 사용자로부터 알림에 대한 권한을 허락받아야 Notification 기능을 제공할 수 있습니다. 이에 따라, 알림 허용 권한이 없는 경우에 Notification이 뜨지 않는 현상이 발생합니다.</p> <p>이에 대해 mVaccine은 두 가지 동작을 제공합니다.</p> <ol style="list-style-type: none"> 알림 사용이 불가능한 경우에 code 반환 알림 사용 가능 여부 체크 API 제공 <p>1. 알림 사용이 불가능한 경우에 code 반환</p> <p>mVaccine에서는 Notification을 사용하는 기능 실행 시, NOTI가 뜨는 시점에 알림 허용 권한 체크를 하여 알림 사용 설정이 되어있지 않을 경우에 POST_NOTI_NO(1300)를 CodeReceiver를 통해 반환합니다. (Android 13 이상만 알림 사용 설정을 확인합니다.)</p>

```
public class CodeReceiver extends BroadcastReceiver {

    @Override
    public void onReceive(Context context, Intent intent) {

        ....

        switch (i)
        {

            ....

            case com.secureland.smartmedic.core.Constants.POST_NOTI_NO:
//1300
                // 노티가 떠있지 않은 경우
                Log.e("CodeReceiver",
                    "com.secureland.smartmedic.core.Constants.POST_NOTI_NO");
                break;

        }
    }
}
```

반환된 POST_NOTI_NO를 통해 앱은 알림 사용이 불가능한 상황임을 인지하여 대응해야 합니다. (CODE 결과에 따른 동작은 mVaccine이 아닌 결과를 받은 앱에서 결정합니다.)

2. 알림 사용 가능 여부 체크 API 제공

mVaccine에서는 NOTI가 뜨는 시점에 알림 허용 권한 체크를 하는 것 대신 미리 알림 사용이 가능한지 체크할 수 있도록 API를 제공합니다.

앱은 CommonUtil.checkNotiPermission 함수를 사용하여, mVaccine 기능을 실행시키지 않아도 알림 사용 가능 여부를 확인할 수 있습니다.

```
if (!CommonUtil.checkNotiPermission(getApplicationContext())) {
    //알림 권한 설정이 되어 있지 않은 경우
}
```

1.13. "Input dispatching timed out" 에러 발생

구분	내용
----	----

메시지	<p>mVaccine 악성앱 검사 혹은 루팅 검사 도중 중 "Input dispatching timed out"</p> <p>ANR 발생</p> <pre> ANR in com.TouchEn.mVaccine.b2b2c (com.TouchEn.mVaccine PID: 32543 Reason: Input dispatching timed out (e07200b com.Touch Parent: com.TouchEn.mVaccine.b2b2c/.main.MainActivity Load: 11.73 / 11.31 / 11.28 ----- Current CPU Core Info ----- - offline : - online : 0-7 - AP Temp = 429 </pre>
원인	<p>동기적으로 동작하는 함수가 Main Thread(UI Thread)에서 작업시간이 길어져 발생하는 ANR입니다.</p>
해결방법	<p>문제가 발생하는 Activity에서 동기적으로 동작하는 함수들을 비동기적으로 동작하는 함수들로 변경해줍니다.</p> <p>(ex : MINI모드 backgroundScan 옵션 true설정, 루팅 API CommonUtil.c(this, true, message, 0, cListener) 사용 등...)</p>

1.14. 32bit 단말에서만 악성앱 오탐

구분	내용
원인	<p>악성앱 탐지에 useDualEngine 옵션 true로 사용시 32bit 단말 한정 듀얼 엔진을 사용합니다.</p> <p>악성앱 패턴을 기반으로 탐지하는 mVaccine 엔진과 달리 듀얼 엔진은 다른 방식으로 탐지하기 때문에 고객센터에서 앱 내에 사용하는 라이브러리, API 등에 따라 일반앱을 악성앱으로 인식하는 경우가 있습니다.</p>
해결방법	<p>해당 앱의 패키지명을 자사 패턴에 화이트 리스트로 등록이 필요합니다.</p> <p>화이트 리스트 패턴 등록 방식은 패턴이 릴리즈 될 때까지 시간이 걸리기 때문에, 긴급하게 패치가 필요할 시에는 악성앱 탐지의 useDualEngine 옵션을 false로 사용하는 방법이 있습니다.</p>

1.15. "SharedPreferences in credential encrypted storage are not available until after user is unlocked" 에러 발생

구분	내용
----	----

메시지	<pre>java.lang.RuntimeException: Unable to instantiate application kr.co.jbbank.privatebank.application.BaseApplication: java.lang.IllegalStateException: SharedPreferences in credential encrypted storage are not available until after user is unlocked ...</pre>
원인	rooting_internal_check 옵션 2번 마운트검사 적용이 되어있는 상태에서 mvclsolService 예외 처리가 제대로 되지 않아 발생하는 문제입니다.
해결방법	<p>“1.9. rooting_internal_check 옵션 2번 마운트검사 적용방법”에서 예외처리를 참고하여 예외처리를 진행해줍니다.</p> <p>정상적으로 예외처리를 했음에도 불구하고 나타나는 경우는 Application 클래스의 최 상단, 상위 클래스에서 예외처리 하지 않는 경우입니다.</p> <p>이 경우 사이트에서 Application를 자사 Application 클래스를 생성해 사용하고 있는지 확인이 필요합니다.</p>

1.16. 폭넓은 패키지(앱) 가시성 (QUERY_ALL_PACKAGES) 권한 심사가 거부되었을 경우

구분	내용
	<p>Android 11 이상을 실행하는 기기에서 Android API 수준 30 이상을 타겟팅 하는 앱의 경우 기기에 설치된 앱 인벤토리를 확인할 수 있게 해주는 QUERY_ALL_PACKAGES 권한을 포함해 위험성이 높거나 민감한 권한의 사용을 제한합니다.</p> <p>악성앱 탐지의 경우 기기에 설치된 패키지에 대한 정보를 가져와 악성앱 여부를 파악하고 있기 때문에 QUERY_ALL_PACKAGES가 없는 경우 악성코드/루팅 탐지가 되지 않는 현상이 발생합니다.</p>
원인	<p>단, QUERY_ALL_PACKAGES권한을 사용하려면 앱이 아래의 허용되는 사용 범위에 속해야 하며 QUERY_ALL_PACKAGES 권한이 허용되는 용도는 다음과 같습니다.</p> <ul style="list-style-type: none"> - 인식 또는 상호 운용성의 목적으로 기기에 설치된 모든 앱을 검색해야 하는 앱과 관련되며, 이 경우 권한 사용이 가능할 수 있습니다. 허용되는 용도에는 기기 검색, 바이러스 백신 앱, 파일 관리자, 브라우저가 포함됩니다. - 금융 거래 기능(예: 전용 은행, 전용 디지털 지갑)과 관련된 검증 가능한 핵심 목적을 가진 앱. 이러한 앱은 보안 관련 목적으로만 설치된 앱을 대상으로 폭넓은 가시성을 획득할 수 있습니다. <p>위의 용도에 해당하지 않는 앱의 경우 권한 요청 심사에서 요청이 거부될 수 있습니다.</p>

	<p>니다.</p>
<p>해결방법</p>	<p>QUERY_ALL_PACKAGES 권한을 사용할 수 없는 경우, queries에 intent 등록 방식을 통해 특정 intent filter를 사용하는 패키지를 가져올 수 있습니다.</p> <p>단, intent 등록 방식은 제한된 패키지 가시성을 이용한 방법이며,</p> <ul style="list-style-type: none"> - 제한된 패키지 가시성으로는 타겟팅이 불가능한 신규 악성앱 탐지가 불가능합니다. - 제한된 패키지 가시성 사용 시 사용자는 타겟팅 된 악성 앱만 탐지가 가능하며 타겟팅 되지 않은 신규 악성 앱에 보안 위협 노출 가능성이 있습니다. <p>따라서, 해당 방식은 QUERY_ALL_PACKAGES권한 사용에 비해 약 90%탐지율 정도를 제공하고 있음을 명시합니다.</p> <p>[적용 방법]</p> <ul style="list-style-type: none"> - AndroidManifest.xml에 다음과 같이 queries에 intent를 등록합니다. <div data-bbox="403 958 1417 1344" style="border: 1px solid black; padding: 10px;"> <pre><manifest packages="..."> <queries> <intent> <action android:name="*" /> </intent> </queries> </manifest></pre> </div> <ul style="list-style-type: none"> ※ 단, <queries>는 Android Gradle 플러그인 4.1.0 (Gradle 6.5 이상) 이상을 사용하는 경우에만 지원됩니다. ※ 해당 조건을 충족하지 못할 경우에는 빌드 과정에서 하단 오류가 발생할 수 있습니다. <div data-bbox="478 1574 1417 1624" style="border: 1px solid black; padding: 5px;"> <p>Error: unexpected element <queries> found in <manifest></p> </div> <p>또는 매니페스트 병합 로그로 이동하는 빌드 출력 창에 오류가 표시될 수 있습니다.</p> <div data-bbox="478 1715 1417 1760" style="border: 1px solid black; padding: 5px;"> <p>Error: Missing 'package' key attribute on element package</p> </div>

1.17. Android resource linking failed 컴파일 에러 발생

구분	내용
----	----

메시지	<pre> Android resource linking failed ... 'shortService' is incompatible with attribute foregroundServiceType (attr) flags [camera=64, connectedDevice=16, dataSync=1, location=8, mediaPlayback=2, mediaProjection=32, microphone=128, phoneCall=4]. error: failed processing manifest. </pre>
원인	<p>프로젝트가 targetSDK 34 미만일 경우 mVaccine CS AAR 라이브러리 사용시 나오는 문제입니다. AAR의 manifest에서 ForegroundServiceType을 지정하는 과정에서 targetSDK 34 미만에서 지원하지 않는 Type을 지정해서 발생하는 문제입니다.</p>
해결방법	<p>targetSDK 34 미만에서는 ForegroundServiceType을 필수로 지정 할 필요 없기 때문에 아래와 같이 프로젝트 manifest에 ForegroundServiceType 지정이 없는 서비스로 replace 하도록 컴포넌트를 등록해줍니다. ('3.2.2.3. 컴포넌트 등록' 참고)</p> <pre> <service android:name="com.TouchEn.mVaccine.b2b2c.service.OnInstallService" android:process=":remote" tools:node="replace" /> <service android:name="com.TouchEn.mVaccine.b2b2c.service.DetectionResultSendService" tools:node="replace"/> <service android:name="com.TouchEn.mVaccine.b2b2c.service.RemoveNotificationService" android:stopWithTask="false" tools:node="replace" /> <service android:name="androidx.work.impl.foreground.SystemForegroundService" tools:node="replace" /> </pre>

2. 기능 동작 화면

mVaccine UI 모드는 총 3가지로 FULL모드, MINI 모드, UI FULL 모드입니다.

FULL 모드		MINI 모드	
패키지 검사 화면	검사 결과 화면	패키지 검사 화면	검사 결과 화면

최초구동시
제품 이미지 나타남

UI FULL 모드		
악성코드 검사 화면	검사 결과 화면	악성 앱 탐지 화면

이상 [부록 2. 기능 동작 화면] 내용을 마칩니다.

3. targetSdkVersion에 따른 변경 사항

3.1. targetSdkVersion 26이상

구분	내용
이슈	실시간 검사(ScanReceiver)가 되지 않을 때
OS버전	Android 8.0 이상
해결방법	AndroidManifest에 실시간 검사(ScanReceiver)를 등록하는 것이 아닌 동적으로 실시간 검사를 등록합니다. (본 문서 3.4.1.2 동적 등록 가이드 참고)

3.2. targetSdkVersion 28이상

구분	내용
이슈	앱에서 HTTP 통신 불가능
OS버전	Android 9.0 이상
해결방법	<p>앱에서 HTTP 통신을 사용할 경우 AndroidManifest.xml 내 android:usesCleartextTraffic="true" 로 설정합니다.</p> <pre><application android:allowBackup="false" android:icon="@drawable/ic_launcher" android:label="TouchEn mVaccine" android:usesCleartextTraffic="true" android:theme="@style/AppTheme"></pre>

구분	내용
이슈	앱에서 탐지된 악성코드 삭제 불가능
OS버전	Android 9.0 이상
해결방법	<p>Manifest.xml에 권한 추가</p> <pre>permission android:name="android.permission.REQUEST_DELETE_PACKAGES" /></pre>

3.3. targetSdkVersion 29 이상

구분	내용
----	----

이슈	악성코드/루팅 탐지 화면이 띄워지지 않는 현상
OS버전	Android 10 이상
해결방법	<p>AndroidManifest.xml에 권한 추가</p> <pre> <uses-permission android:name="android.permission.FOREGROUND_SERVICE" /> <uses-permission android:name="android.permission.USE_FULL_SCREEN_INTENT" /> </pre>

3.4. targetSdkVersion 30 이상

구분	내용
이슈	악성코드/루팅 탐지가 되지 않는 현상
OS버전	Android 11 이상
해결방법	<p>Manifest.xml에 권한 추가</p> <pre> <permission android:name="android.permission.QUERY_ALL_PACKAGES" /> </pre>

3.5. targetSdkVersion 31 이상

구분	내용
이슈	targetSdkVersion 31 이상에 구성요소가 intent-filter를 사용하거나 service, broadcast receiver를 포함하면 android:exported 속성을 명시적으로 선언해야 한다.
OS버전	Android 12 이상
해결방법	<p>AndroidManifest.xml intent-filter를 사용하는 구성요소, service, broadcast receiver에 exported 추가</p> <pre> <activity android:name=".MainActivity_sample" ... android:exported="true"> <intent-filter> <action android:name="android.intent.action.MAIN" /> <action android:name="android.intent.action.VIEW" /> </intent-filter> </pre>

	<pre> <category android:name="android.intent.category.LAUNCHER" /> </intent-filter> </activity> ... <receiver android:name=".CodeReceiver" android:exported="false"> <intent-filter> <action android:name="com.TouchEn.mVaccine.b2b2c.FIRE" /> </intent-filter> </receiver> </pre>
--	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

3.6. targetSdkVersion 33 이상

구분	내용
이슈	Notification 알람이 뜨지 않는 현상. Android 13(API 수준 33)에서는 앱에서 예외 없는 알람을 보내기 위해서는 새로운 런타임 권한 POST_NOTIFICATIONS를 선언해야 한다.
OS버전	Android 13 이상
해결방법	AndroidManifest.xml에 권한 추가 <pre> <uses-permission android:name="android.permission.POST_NOTIFICATIONS"/> </pre>

3.7. targetSdkVersion 34 이상

구분	내용
이슈	런타임 등록 broadcast receiver는 내보내기 동작을 지정해야 함 https://developer.android.com/about/versions/14/behavior-changes-14?hl=ko#runtime-receivers-exported registerReceiver 사용시 런타임 에러 발생. One of RECEIVER_EXPORTED or RECEIVER_NOT_EXPORTED should be specified when a receiver isn't being registered exclusively for system broadcasts
OS버전	targetSDK 34 이상
해결방법	OS버전O(8.0) 이상 registerReceiver 등록시 내보내기 동작 지정.

	<pre> if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) { registerReceiver(codeReceiver, intentFilter2, RECEIVER_NOT_EXPORTED); } else { registerReceiver(codeReceiver, intentFilter2); } </pre>
--	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

구분	내용
이슈	<p>포그라운드 서비스 유형은 필수 항목임</p> <p>https://developer.android.com/about/versions/14/behavior-changes-14?hl=ko#fgs-types</p>
OS버전	targetSDK 34 이상
해결방법	<p>아래 포그라운드 서비스에 foregroundServiceType shortService로 지정</p> <pre> <!-- Android 운영체제에서 엠백신 스레드 종료처리 방지용 서비스 추가 --> <service android:name="com.TouchEn.mVaccine.b2b2c.service.OnInstallService" android:process=":remote" android:foregroundServiceType="shortService" /> <!-- Q 대응 헤드업 노티를 띄워주기 위해서 사용 --> <service android:name="com.TouchEn.mVaccine.b2b2c.service.DetectionResultSendService" android:foregroundServiceType="shortService" /> <!-- WorkManager ForegroundService 적용 --> <service android:name="androidx.work.impl.foreground.SystemForegroundService" android:foregroundServiceType="shortService" tools:node="merge" /> </pre>

구분	내용
이슈	<p>Google Play 스토어에서는 프로필에 맞지 않는 앱의 기본 USE_FULL_SCREEN_INTENT 권한을 취소시킴.</p> <p>https://developer.android.com/about/versions/14/behavior-changes-all?hl=ko#secure-fsi</p>

OS버전	targetSDK 34 이상
해결방법	<p>USE_FULL_SCREEN_INTENT 권한 제거.</p> <pre><!-- android 10 에서 탐지 결과표시를 헤드업노티로 띄워주기 위해 사용 --> <uses-permission android:name="android.permission.USE_FULL_SCREEN_INTENT" /></pre>