# Mobile Integrity Checker 개발자 매뉴얼

**Team.** 범부

김용진, 김병민, 강원찬, 송채린

작성 일자 : 2026.04.03

# 목차

# Mobile Integrity Checker 개발자 매뉴얼

## 1. 프로젝트 개요

### 1.1 시스템 목적 및 범위

모바일 앱 환경에서 루팅, Frida, Xposed, 에뮬레이터, 디버그 모드 등의 보안 위협을 실시간으로 탐지하고, 앱의 무결성을 서버 측에서 검증하는 시스템입니다.

- 앱 위변조 탐지를 통한 금융·공공 앱 보안 강화
- 서버 기반 기준값 비교로 클라이언트 우회 방지
- 위험도 등급(Safe / Warning / Danger) 기반 자동 대응

### 1.2 기술 스택

| 구분 | 기술 |
|---|---|
| Android SDK | Retrofit2, OkHttp3, Gson, |
| 백엔드 서버 | Go 1.22, Gin Framework |
| ORM | GORM (PostgreSQL Driver) |
| 데이터베이스 | PostgreSQL 16 |
| 보안 및 정적 분석 | gosec, govulncheck |
| 설정 관리 | .env (godotenv) |
| 프론트엔드 | HTML |

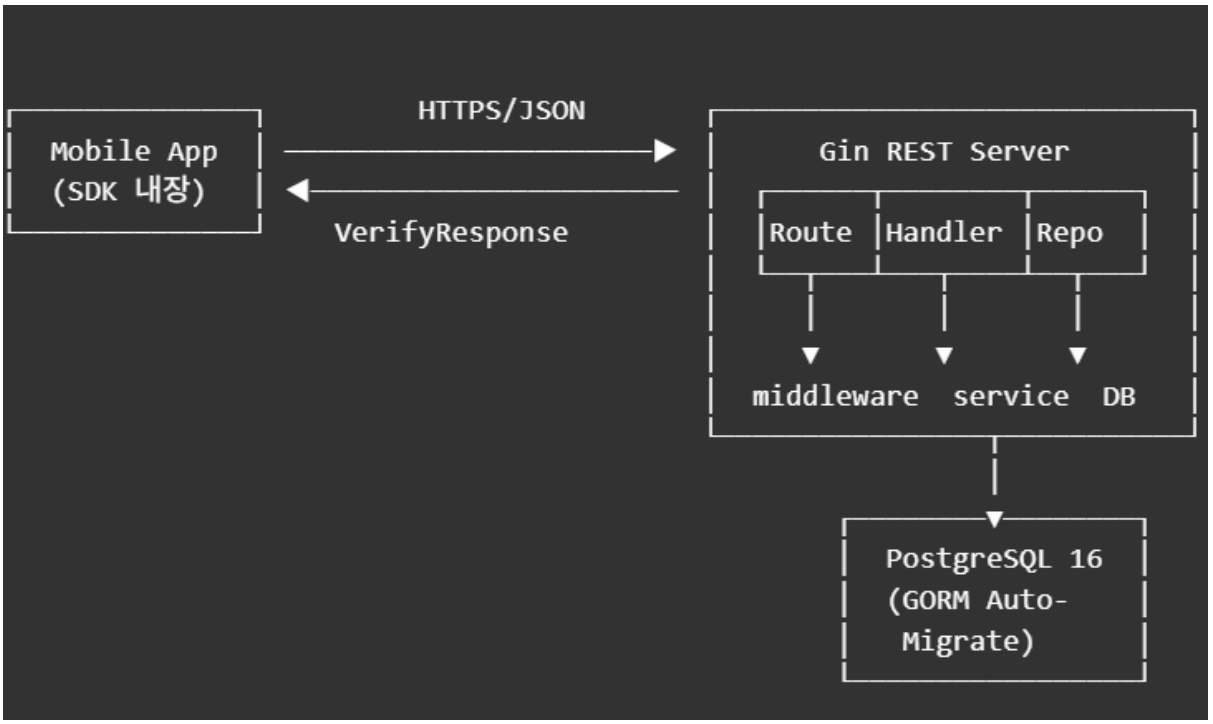
### 1.3 개발 환경 설정

- 협업 및 형상 관리 — Go/Gin 기반 스켈레톤 코드를 생성하여 협업 환경을 구축하였습니다.
- 데이터베이스 — PostgreSQL을 설치하고 프로젝트 전용 데이터베이스를 생성하였습니다.
- 자동화 — GitHub Actions를 활용하여 소스 코드 추가 시 취약점을 자동으로 점검하는 환경을 구성하였습니다.

# 2. 시스템 아키텍처

## 2.1 전체 구조도

본 시스템은 Mobile SDK, API 서버, 데이터베이스의 3계층 구조로 구성됩니다. SDK에서 수집한 기기 상태 데이터를 서버의 API 엔드포인트로 전달하며, 서버는 DB의 기준값과 대조하여 위험도를 판정합니다.




## 2.2 모듈별 역할

| 패키지 | 역할 |
|---|---|
| repository | DB 초기화, CRUD 함수 (Baseline, VerifyRequest, VerifyDataSet, User) |
| model | GORM 엔티티 + API 요청/응답 DTO 정의 |
| handler | Gin 라우터 핸들러 (컨트롤러 계층) |
| service | 검증 비즈니스 로직, 위험도 산출 |
| middleware | 로그인 여부 확인 |

| | |
|---|---|
| BeombuSecurityEngine | 기기 정보 수집 및 서버 전송 총괄 |

# **3.** 기능 명세 및 상세 설계

## **3.1** 핵심 기능 명세

| 기능 **ID** | 기능명 | 상세 내용 |
|---|---|---|
| F-01 | 정적 무결성 검증 | APK 해시 및 서명 해시 값을 기준 데이터와 대조하여 앱의 물리적 변조 여부 확인 |
| F-02 | 동적 환경 분석 | 기기의 루팅 상태, 디버깅 모드 활성화 여부, 에뮬레이터 접속 여부 실시간 감지 |
| F-03 | 보안 도구 탐지 | Frida, Xposed 등 메모리 변조 및 API 후킹에 악용되는 프레임워크의 실행 여부 확인 |
| F-04 | 위험도 기반 제어 | 탐지 항목별 가중치를 합산하여 위험 점수를 도출하고, 등급별 서비스 이용 허용 여부 결정 |

## **3.2** 데이터베이스 설계

테이블 구성

| 테이블 | **PK** | 컬럼 | 용도 |
|---|---|---|---|
| baselines | package_name | signature_hash, installer_pkg, allowed_devices | 관리자가 등록한 앱 기준값 |
| verify_requests | id (uint) | ID,UUID, PackagenNme,VersionCode, ApkHash,SignatureHash, Rooted,DebugEnabled,IsFrida, IsXposed,IsEmulator,ExcutedAt | 클라이언트 원본 요청 저장 |
| verify_data_sets | id (uint) | ID,UUID,ApkHash,Rooted, BebugEnabled,RiskScore,RiskLevel | 비교 검증 결과 영구 저장 |
| users | id (string) | password, is_admin | 관리자 계정 (초기값: admin/admin) |

**Repository** 함수 매핑

| 함수 | **SQL** 동작 | 대상 테이블 |
|---|---|---|
| InitDB() | AutoMigrate + Seed | 전체 |
| RegisterBaseline() | INSERT | baselines |
| GetBaseline(packageName) | SELECT WHERE | baselines |
| SaveVerifyRequest() | INSERT | verify_requests |
| SaveVerifyResult() | INSERT | verify_data_sets |
| GetRecentResults(uuid, duration) | SELECT → DTO 가공 | verify_data_sets |
| CleanupByRiskLevel() | DELETE (등급별 보존 기간) | verify_data_sets |
| AdminLogin(packageName) | SELECT WHERE | users |

데이터 보존 정책 **(TTL)**

| 위험 등급 | 보존 기간 | 삭제 조건 |
|---|---|---|
| Safe | 7일 | risk_level = 'Safe' AND executed_at < now - 7d |
| Low | 7일 | risk_level = 'Low' AND executed_at < now - 7d |
| Warning | 1개월 | risk_level = 'Warning' AND executed_at < now - 1M |
| Danger | 3개월 | risk_level = 'Danger' AND executed_at < now - 3M |
| Critical | 3개월 | risk_level = 'Critical'' AND executed_at < now - 3M |

데이터 조회 최적화: UID 및 실행 시간 기반의 복합 인덱스(Composite Index)를 생성하여 대량의 검증 이력을 빠르게 조회합니다.

### **3.3** 위험도 산출 로직

위험 요소별 가중치

| 탐지 항목 | 필드 | 조건 | 가중치 |
|---|---|---|---|
| App Integrity | apkHash | 불일치 | 100점 |
| Sign Integrity | signatureHash | 불일치 | 100점 |
| Frida Detection | isFrida | True | 100점 |
| Xposed Check | isXposed | True | 80점 |
| Unregistered Package | UnregisteredPackage | True | 50점 |
| Rooting Check | rooted | True | 40점 |
| Debug Check | DebugEnabled | True | 20점 |
| Emulator Check | isEmulator | True | 10점 |

등급 분류 기준

| **Risk Score** 범위 | 등급 | **allow** 값 |
|---|---|---|
| 0 ~ 9 | Safe | true |
| 10~19 | Low | true |
| 19 ~ 79 | Warning | true (경고 포함) |
| 80 ~90 | Danger | false |
| 90~100 | Critical | false |

**Risk Analysis Engine** 처리 흐름

| 단계 | 핵심 컴포넌트 | 설명 |
|---|---|---|
| 요청 대기 | VerifyQueue | 유저로부터 접수된 앱 검증 요청 데이터를 Queue에 안전하게 보관 |
| 병렬 처리 | Worker / StartWorkerPool | 여러 Worker가 Queue에서 요청을 가져와 동시에 처리하여 과부하 방지 |
| 위험도 평가 | VerifyApp | 루팅, 해킹 툴, 위변조 여부 등을 분석하여 위험 점수와 등급 판정 |
| 결과 저장 | VerifyApp | 최종 결과 데이터를 DB(repository.SaveVerifyResult)에 영구 기록 |

3.4 SDK 데이터 수집 로직

- UUID : SharedPreferences활용하여 기기 고유의 UUID 생성 및 관리
- APK Hash: PackageName + VersionCode + Signature + 값을 해쉬값으로 암호화
- Env Check : 루팅,디버깅등을 구분지어 줍니다

# 4. API 설계 및 명세

## 4.1 설계 원칙 및 통신 규격

- 외부 연동이 용이하도록 표준 API 규격을 지정하여 작성되었습니다.
- Gin Framework를 활용하여 효율적인 API 경로 라우팅을 구현하였습니다.

```
api.POST("/verify", handler.VerifyHandler)
api.POST("/baseline",  handler.RegisterBaselineHandler)
api.GET ("/baseline/:packageName", handler.GetBaselineHandler)
api.GET ("/results", handler.GetResultsHandler)
api.POST("/cleanup", handler.CleanupHandler)
```

## 4.2 주요 API 목록

| Method | 경로 | Handler | 설명 |
|---|---|---|---|
| POST | /api/verify | VerifyHandler | 클라이언트 APK 정보 수신, 유효성 검사 후 위험도 분석 수행 |
| POST | /api/baseline | RegisterBaselineHandler | 앱 정상 기준 정보(packageName, apkHash, signatureHash) 등록 |
| GET | /api/baseline/:packageName | GetBaselineHandler | 패키지명으로 기준값 조회. 미등록 시 404 반환 |
| GET | /api/results | GetResultsHandler | UUID와 기간(hours) 파라미터로 검증 결과 조회 |
| POST | /api/cleanup | CleanupHandler | 위험 등급별 보존 기간 경과 데이터 자동 삭제 트리거 |

### 4.3 엔드포인트 상세 명세

#### 4.3.1. 앱 무결성 검증 (Verify)

클라이언트 상태 정보를 기반으로 위험 점수를 계산하고 검증 결과를 반환한다.

- URL: POST /api/verify

```
Request Body
{
"Uuid": "string",
"PackageName": "string",
"VersionCode": int,
"ApkHash": "string",
"SignatureHash": "string",
"Rooted": Boolean,
"DebugEnabled": Boolean,
"IsFrida": Boolean,
"IsXposed":Boolean,
"IsEmulator": Boolean,
"ExecutedAt":"String"
}
```

결과: 성공 시 200 OK 및 위험도 점수 반환

#### 4.3.2. 기준값(Baseline) 등록

앱의 정상 상태(해시, 버전 등)를 서버에 등록한다.

- URL: POST /api/baseline

```
Request Body
{SharedPreferences
  "packageName": "com.bumboo.app",
  "versionCode": 10,
  "apkHash": "sha256-hash-value",
  "signatureHash": "sig-hash-value"
}

Response Body
{
  "status": "success",
  "MessageDigest": "com.bumboo.app 기준값이 등록되었습니다"
}
```

#### 4.3.3. 기준값 조회

특정 패키지의 기준 데이터를 조회한다.

- URL: GET /baseline/:packageName

```
Response Body
{
  id: 16,
  uuid: "550e8400-e29b-41d4-a716-446655440000",
  packageName: "com.example.bankapp" ,...
}
```

#### 4.3.4. 검증 결과 이력 조회

특정 디바이스(UUID)의 최근 검증 이력을 시간 단위로 조회한다.

- URL: GET /results?uuid={uuid}&hours={hours}

```
Response Body
{
  "status": "success",
  "count": 1,
  "results": [
    {
      "id": 101,
      "uuid": "device-unique-id-123",
      "packageName": "com.bumboo.app",
      "riskScore": 0,
      "riskLevel": "LOW",
      "allow": true,
      "executedAt": "2026-04-02T10:00:00Z"
    }
  ]
}
```

#### 4.3.5. 관리자 로그인

관리자 페이지 접근을 위한 인증을 수행한다.

- URL: POST /api/login

```
Request Body
{
  "id": "admin",
  "password": "password123"
}

Response Body
{
  "status": "success",
  "message": "admin님 환영합니다!"
}
```

### 4.4 에러 코드 정의

공통 에러 응답 형식

시스템 전반에서 사용하는 표준 에러 응답 구조이다.

```
{
  "risk_score": -1,
  "status": "error",
  "message": "에러 메시지 내용"
}
```

상태 코드 및 메시지 규격

| **HTTP** 상태 코드 | 메시지**(**예시**)** | 발생 사유 |
|---|---|---|
| **400 Bad Request** | 잘못된 요청 형식입니다 | JSON 파싱 실패 또는 필수 필드 누락 |
| **400 Bad Request** | packageName(은)는 필수입니다 | 유효성 검사 실패 (필수 파라미터 누락) |
| **401 Unauthorized** | 아이디 또는 비밀번호가 일치하지 않습니다 | 로그인 실패 (계정 정보 불일치) |
| **404 Not Found** | {packageName} 에 대한 기준값이 없습니다 | 조회 대상 데이터가 DB에 존재하지 않음 |
| **500 Internal Server Error** | 서버 오류가 발생했습니다 / 기준값 등록 실패 | DB 처리 오류 또는 서버 내부 로직 장애 |

참고사항

- CleanupHandler는 별도의 파라미터 없이 호출 시 위험도 등급별로 데이터를 자동 정리합니다.

# **5.** 라이브러리 및 보안 관리 **(SBOM)**

## **5.1** 오픈소스 명세

프로젝트 시작 단계에서 SBOM 계획을 수립하여 사용되는 외부 라이브러리의 투명성을 확보하였습니다.

**Go** 백엔드 의존성

| 패키지 | 버전 | 용도 |
|---|---|---|
| github.com/gin-gonic/gin | v1.9+ | HTTP 라우팅 및 미들웨어 |
| gorm.io/gorm | v1.25+ | ORM |
| gorm.io/driver/postgres | v1.5+ | PostgreSQL 드라이버 |
| github.com/joho/godotenv | v1.5+ | .env 파일 로드 |

**Android SDK** 의존성

| 패키지 | 버전 | 용도 |
|---|---|---|
| androidx.core:core-ktx | 1.x+ | Android Kotlin 확장 |
| com.squareup.retrofit2:retrofit | 2.x+ | HTTP 통신 클라이언트 |
| com.squareup.okhttp3:okhttp | 4.x+ | HTTP 클라이언트 (Retrofit 기반) |
| org.jetbrains.kotlinx:kotlinx-coroutines-android | 1.x+ | 비동기 처리 |
| com.squareup.retrofit2:converter-gson | 2.9+ | Json을 객체로 변환 |
| kotlinx-coroutines-android | 1.7.3 | 네트워크 통신시 UI스레드 정지방지 |

## **5.2** 보안 점검 가이드

- 정기 실행 도구 — govulncheck를 활용하여 환경 구성 및 정기적인 보안 점검을 수행합니다.

## **5.3** 오픈소스 라이선스 준수사항

| 패키지 | 라이선스 | 주요 준수사항 |
|---|---|---|
| github.com/gin-gonic/gin | MIT | 저작권 고지 유지 |
| gorm.io/gorm | MIT | 저작권 고지 유지 |
| gorm.io/driver/postgres | MIT | 저작권 고지 유지 |
| github.com/joho/godotenv | MIT | 저작권 고지 유지 |
| androidx.core:core-ktx | Apache-2.0 | NOTICE 파일 포함 |
| com.squareup.retrofit2:retrofit | Apache-2.0 | NOTICE 파일 포함 |
| com.squareup.okhttp3:okhttp | Apache-2.0 | NOTICE 파일 포함 |
| kotlinx-coroutines-android | Apache-2.0 | NOTICE 파일 포함 |

공통 준수사항

- MIT / BSD-3-Clause: 소스코드 또는 바이너리 배포 시 원본 저작권 고지문을 유지해야 합니다.
- Apache-2.0: 배포 시 NOTICE 파일을 포함해야 하며, 수정 사항이 있을 경우 명시해야 합니다.

# 6. 테스트 및 품질 관리

## 6.1 테스트 전략

가상 아티팩트 생성: 실제 기기 환경 외에도 테스트 최적화를 위한 가상의 아티팩트를 생성하여 다양한 케이스를 검증합니다.

※ Unit test 결과 내용 추가 예정

## 6.2 테스트 케이스

| 구분 | 대상 | 도구 | 상태 |
|---|---|---|---|
| 단위 테스트 | repository 함수 | Go testing | 진행 중 |
| 통합 테스트 | API 엔드포인트 | Postman / http test | 진행 중 |
| 보안 테스트 | 루팅/Frida 탐지 정확도 | 실 디바이스 | 예정 |

사용자 리뷰 및 내부 QA 결과를 수용하여 기능 개발용 브랜치를 운영하고 릴리즈 노트를 작성합니다.

## 6.3 영향도 분석 (Impact Analysis)

### IA-01 | 위험도 점수 기준 변경 (가중치 / 등급 임계값 수정)

변경 위치: `service/risk_service.go`

| 영향 범위 | 파일 | 변경 내용 |
|---|---|---|
| 응답 구조 | `model/VerifyResponse` | RiskScore 범위가 달라지면 클라이언트 측 해석 기준도 변경 필요 |
| DB 저장값 | `model/VerifyDataSet.RiskLevel` | 등급 문자열(Safe/Warning/Danger) 변경 시 저장값 일관성 훼손 |
| 데이터 정리 | `repository/CleanupByRiskLevel()` | 등급명 변경 시 삭제 쿼리 조건도 동일하게 수정 필요 |
| 프론트엔드 | `index.html` | 등급별 색상, 레이블 표시 로직 수정 필요 |

**IA-02 |** 탐지 항목 추가 **(**새로운 보안 필드 추가**)**

변경 위치: `model/VerifyRequest`

| 영향 범위 | 파일 | 변경 내용 |
|---|---|---|
| 결과 저장 모델 | `model/VerifyDataSet` | 동일 필드 추가 필요 (요청과 결과 구조 동기화) |
| 위험도 로직 | `service/risk_service.go` | 새 필드에 대한 가중치 및 판정 로직 추가 |
| 핸들러 | `handler/verify_handler.go` | validateRequest() 유효성 검사 항목 추가 여부 검토 |
| DB 스키마 | `repository (Auto Migrate)` | GORM AutoMigrate로 자동 반영되나, 기존 데이터 NULL 처리 확인 필요 |
| 프론트엔드 | `index.html` | 새 항목 표시 UI 추가 필요 |

**IA-03 | Baseline** 구조 변경 **(**기준값 필드 추가 **/** 수정**)**

변경 위치: `model/Baseline`

| 영향 범위 | 파일 | 변경 내용 |
|---|---|---|
| 핸들러 유효성 검사 | `handler/verify_handler.go` → `RegisterBaselineHandler()` | 새 필수 필드 추가 시 유효성 검사 로직 수정 |
| 비교 로직 | `service/risk_service.go` | Baseline과 VerifyRequest 비교 부분에서 새 필드 반영 |
| DB 스키마 | `repository` (AutoMigrate) | 컬럼 추가는 자동 반영, 컬럼 삭제·수정은 수동 마이그레이션 필요 |
| API 명세 | POST `/api/baseline` | 요청 바디 스펙 변경으로 클라이언트 연동 스펙 업데이트 필요 |

**IA-04 |** 데이터 보존 정책 변경 **(TTL** 기간 수정**)**

변경 위치: `repository/CleanupByRiskLevel()`

| 영향 범위 | 파일 | 변경 내용 |
|---|---|---|
| 핸들러 | `handler/verify_handler.go` → `CleanupHandler()` | 직접 영향 없음, 호출 구조 유지 |
| 운영 정책 문서 | 개발자 매뉴얼 3.2절 | TTL 정책 표 업데이트 필요 |
| GitHub Actions | CI/CD 자동화 스케줄 | 정리 주기가 변경된 경우 트리거 주기 동기화 필요 |

**IA-05 |** 인증 방식 변경 **(**로그인 **/** 사용자 모델 수정**)**

변경 위치: `model/User`, `handler/verify_handler.go` → `LoginHandler()`

| 영향 범위 | 파일 | 변경 내용 |
|---|---|---|
| 요청/응답 모델 | `model/LoginRequest`, `model/LoginResponse` | 필드 추가 시 동일하게 수정 |
| DB 조회 | `repository/AdminLogin()` | 조회 조건 또는 반환값 변경 필요 |
| 라우터 | `router/router.go` | 미들웨어 적용 범위 변경 시 라우팅 그룹 수정 필요 |
| 보안 | 전체 API | 토큰 기반 인증 도입 시 모든 핸들러에 미들웨어 적용 범위 검토 필요 |

**IA-06 | API** 경로 변경

변경 위치: `router/router.go`

| 영향 범위 | 파일 | 변경 내용 |
|---|---|---|
| 핸들러 연결 | `handler/verify_handler.go` | 경로 변경 자체는 핸들러 무관하나,<br>파라미터 방식 변경 시 수정 필요 |
| 클라이언트 SDK | Android (Kotlin) | 엔드포인트 URL 하드코딩된 경우 전체 수정 필요 |
| 프론트엔드 | `index.html` | fetch/axios 호출 URL 수정 필요 |
| API 문서 | 개발자 매뉴얼 4.2절 | API 경로 업데이트 필요 |

# **7.** 유지보수 및 문제해결

## **7.1** 개발 역할 분담

| 이름 | 담당 역할 |
|---|---|
| 김용진 | 초기 기획, 기술 스택 선정, DB 및 SDK 설계, WBS 작성 총괄 |
| 김병민 | 서버 API 개발, 프론트엔드 웹 모니터링 환경 구축 |
| 강원찬 | 위변조 및 해킹 도구 탐지 로직 구현, 통신 기능 및 CI/CD 자동화 구축 |
| 송채린 | 데이터베이스 저장 및 쿼리 최적화, 최종 보고서 작성 |

## **7.2** 히스토리 및 장애 대응

- 운영 중 발생하는 버그는 재현 및 디버깅 과정을 거쳐 코드 수정 후 테스트를 진행합니다.
- 장애 대응 및 운영 가이드북은 중앙 집중형 위키를 통해 지속적으로 업데이트됩니다.

자주 발생하는 문제

| 증상 | 원인 | 해결 |
|---|---|---|
| DB 연결 실패 | .env 파일 누락 또는 환경 변수 오류 | .env 파일 확인, DB_HOST/PORT 검증 |
| AutoMigrate 오류 | 모델 구조체 변경 후 마이그레이션 충돌 | 수동 ALTER TABLE 또는 DB 초기화 |
| 기준값 없음 로그 | 해당 패키지의 Baseline 미등록 | 관리자 페이지에서 기준값 등록 |