Git-Checker

데이터 기반 오픈소스 신뢰성 진단 플랫폼

기능명세서 및 API 설계 종합 보고서

KISIA S-개발자 4기 | 5조 | 5-개발자
김이찬 김준서 김주환 심도연 황세웅

1. 서비스 소개

1.1 개요

Git-Checker는 오픈소스 도입 시 발생하는 **검수 비용과 보안 리스크를 데이터로 해소하는** 신뢰성 진단 플랫폼입니다.
GitHub 저장소 URL 하나만 입력하면 활성도 분석, CVE 취약점 스캔, 커뮤니티 건강도 평가, LLM 기반 README 문서 품질 평가를 수행하여 100점 만점의 종합 점수 및 즉시 활용 가능한 마크다운 리포트를 제공합니다.
오픈소스 도입 여부를 판단해야 하는 아키텍처 설계자, 테크리드, 개발 팀장이 별도의 조사 없이 빠르게 의사결정할 수 있도록 설계되었습니다.

1.2 개발 배경 및 필요성

오픈소스 도입 시 다수의 개발 조직이 Star 수 등 단순 인기 지표에 의존하고 있습니다.
그러나 해당 지표만으로는 유지보수 현황과 보안 상태를 판단할 수 없습니다.
Sonatype(2024) 보고서에 따르면 취약 컴포넌트 사용 사례**의 95%에서 패치 버전이 이미 존재함에도 미적용 상태**였습니다.
악성 패키지는 전년 대비 **156% 증가**하여 검수 없는 도입은 공급망 보안 위협으로 직결됩니다.
기존 도구는 **도입 후 운영 단계의 취약점 추적 또는 작성 코드 리뷰에 특화**되어 있습니다.
도입 여부를 판단하는 설계 단계에서 보안·활성도·커뮤니티·문서 품질을 종합적으로 확인하려면 여러 도구와 사이트를 개별 조회해야 합니다.
이 과정에서 검수 비용이 발생하고 지표 누락에 따른 판단 편향이 잘못된 도입 결정으로 이어질 수 있습니다.
Git-Checker는 이 문제를 해결하기 위해 **오픈소스 도입 의사결정 시점에 필요한 핵심 지표를 하나의 진단 리포트로 제공**합니다.

1.3 서비스 포지셔닝

비교 항목	코드 린팅 도구 (Super-Linter 등)	OSS 보안 플랫폼 (Socket.dev · Snyk)	Git-Checker
핵심 목적	작성 코드의 포맷팅·문법 검사	의존성 패키지 보안 스캔 및 취약점 관리	오픈소스 도입 전 신뢰성 종합 진단
분석 시점	코드 작성 후 (커밋·PR)	도입 후 운영 중 지속 모니터링	도입 전 의사결정 시점
분석 범위	PR·커밋 단위 소스코드	CVE + 악성 패키지 + 라이선스	보안 + 활성도 + 커뮤니티 + 문서 품질

출력 형태	GitHub PR 코멘트·콘솔 로그	웹 대시보드·IDE 플러그인	100점 종합 점수 + 마크다운 리포트

2. 핵심 기능

2.1 4대 핵심 기능 개요

기능명	가중치	설명
보안성 진단 Security Guard	**45%**	의존성 패키지의 알려진 CVE 취약점을 탐지하고 심각도(Critical·High·Medium·Low)별로 분류하여 감점 적용. OSV-Scanner와 Semgrep을 통해 이중 검증 수행
활성도 분석 Activity Analytics	**40%**	최근 커밋 날짜·빈도, 릴리즈 주기, 이슈/PR 응답 속도, 기여자 수를 수집하여 프로젝트가 실제로 유지보수 되고 있는지 정량적으로 평가
커뮤니티 건강도 Community Health	**10%**	Star 수·Fork 수·외부 기여자 비율 등 생태계 규모 데이터를 기반으로 프로젝트의 장기 생존 가능성을 보완 지표로 평가
문서화 품질 Documentation Quality	**5%**	LLM이 README 전문을 분석하여 설치 가이드·사용 예제·트러블슈팅 포함 여부와 품질을 정성 평가하고 한 줄 요약 제공

2.1-1. 가중치 선정 기준

가중치는 **OpenSSF Scorecard**, **CHAOSS Metrics Model**, **Zhao et al.(2021)**의 오픈소스 평가 지표 체계를 참고하여 설계하였습니다.
OpenSSF Scorecard는 보안 취약점과 유지보수 상태를 High/Critical 리스크로 분류하며, CHAOSS는 응답 속도·릴리즈 빈도·기여자 집중도를 프로젝트 건강도의 핵심 지표로 제시합니다.

지표	가중치	근거
보안성	**45%**	취약 컴포넌트의 95%에서 패치가 미적용 상태를 반영
활성도	**40%**	유지보수가 중단된 프로젝트는 보안 패치 자체가 불가능하므로 최우선 지표로 판단

커뮤니티	**10%**	장기 생존 가능성의 보완 신호로 유효하나 단독 판단 근거로는 부족
문서화	**5%**	도입 판단에 참고가 되나 보안·활성도 대비 직접적 리스크 영향이 낮음

2.2 주요 사용자 경험(UX) 기능

① 원스톱 분석 대시보드

GitHub URL 하나를 입력하면 레이더 차트와 Green/Yellow/Red 신호등 체계의 체크리스트로 결과를 즉시 시각화합니다. 복잡한 수치 해석 없이 직관적인 판단이 가능합니다.

② AI README 프리뷰 및 요약

LLM이 방대한 README를 핵심만 요약하고, react-markdown을 통해 실제 GitHub와 유사한 가독성 높은 프리뷰를 제공합니다. 긴 문서를 모두 읽지 않아도 도입 여부 판단이 가능합니다.

③ 원클릭 마크다운 리포트 복사

분석된 모든 표와 점수를 Markdown 형식으로 즉시 복사할 수 있습니다. Notion·Slack·Teams에 붙여넣기만 하면 별도 문서 작성 없이 팀 의사결정 자료로 활용 가능합니다.

④ 등급별 권한 관리

게스트(GUEST): 일일 5회 분석 제한 — 서버 자원 보호 및 남용 방지

로그인 유저(USER): JWT 기반 일일 분석 + 개인 분석 이력 관리

2.3 접근 권한 정책

등급	조건	분석 횟수	비고
게스트 (GUEST)	비회원 접속	일일 5회 제한	서버 자원 보호 및 남용 방지
회원 (USER)	소셜 로그인 (JWT 기반)	일일 20회 제한	개인 분석 이력 저장 및 조회 기능

3. 기능명세서

3.1 기능 명세 도출 기준

① 개발 목적 키워드 추출: 데이터 기반 신뢰성 진단, 실시간 보안 취약점 스캔, 마크다운 리포트 출력 등 서비스 정의에서 기능 단위를 분해하였습니다.

② UX 시나리오 역추적: 사용자가 URL을 입력한 뒤 결과를 타 문서화 플랫폼에 붙여넣기까지의 전체 흐름을 단계별로 분해하여 누락 기능을 발굴하였습니다.

③ 팀원 5인 개별 명세 교차 검토: 각자 도출한 기능 목록을 합산·중복 제거 후 대분류/중분류/소분류로 계층화하였습니다.

3.2 기능 명세 전체 목록

대분류	중분류	소분류(기능명)	핵심 로직 / 키워드
1. 인증/인가	인증	**소셜 로그인**	Firebase Auth 기반 구글 소셜 로그인, JWT 토큰 발급
	인증	**로그아웃**	클라이언트 로컬 토큰 삭제
	권한 제한	**분석 횟수 제한**	Firebase DB 카운팅(게스트 5회 / 회원 20회)
	권한 제한	**잔여 횟수 조회**	유저 유형별 잔여 횟수 반환
2. 분석 요청	입력 처리	**GitHub URL 입력**	GitHub URL 입력값 수신 및 화면 표시
	유효성 검사	**URL 형식·경로 검증**	github.com 허용, owner/repo 패턴 검증
	URL 처리	**GitHub 주소 정규화**	URL에서 owner/repo 추출, .git 접미사 제거
3. 분석	분석 요청	**저장소 분석 요청 (POST)**	분석 작업 생성, analysis Id 발급
	데이터 수집	**저장소 메타데이터 수집**	GitHub REST API로 Stars·Forks·Commits·README 수집
	보안 스캔	**의존성 취약점 탐지**	OSV-Scanner + Semgrep 이중 검증
	AI 분석	**README 품질 평가 및 저장소 분류**	README 정성 평가 및 한 줄 요약 생성 + 프레임 워크 /라이브러리/ 개인 레포 구별
	진행 제어	**분석 상태 조회**	백엔드 이벤트 기반 상태 추적 상태 반환
4. 결과 출력	결과 조회	**분석 결과 조회**	보안 활성도 커뮤니티 문서화 점수 근거·세부 분석 결과 조회 + 최종 점수 산출
	상세 데이터 출력	**상세 분석결과 조회**	보안, 활성도, 문서화 항목 별 상세 분석결과 출력

	결과 공유	**마크다운 리포트 복사**	Markdown 형식 결과 표 제공 및 클립보드 복사

4. API 설계

4.1 설계 원칙

본 프로젝트는 코딩 전 API 명세를 먼저 확정하는 API-First Design을 채택하였습니다. 프론트엔드·백엔드가 명세를 단일 진실의 원천(SSOT)으로 삼아 병렬 개발이 가능하며 소통 비용을 최소화합니다.

원칙	내용
데이터 포맷	JSON 통일 — React/TypeScript 파싱, Spring Boot Jackson 직렬화, GitHub REST API·LLM API 응답 포맷과 일관성 확보
응답 구조(성공)	{ "success": true, "message": string, "data": object }
응답 구조(에러)	{ "success": false, "message": string, "error": { "code": string, "detail": string } }
API 네이밍	복수형 명사 사용 (/analyses, /results), HTTP 메서드로 행위 표현, 동사 API 금지
HTTP 메서드	읽기 → GET, 생성·요청 → POST, 수정 → PUT/PATCH, 삭제 → DELETE
내부 처리	보안 스캔·AI 분석·점수 산출 등 외부 노출 불필요 항목은 INTERNAL로 분류하여 엔드포인트 미공개

4.2 API 엔드포인트 전체 목록

대분류	소분류(기능명)	Method	Endpoint
1. 인증/인가	로그인	POST	*/api/auth/login*
	로그아웃	POST	*/api/auth/logout*
	유저 유형별 분석 횟수	GET	*/api/userRemainCount*
	분석 횟수 제한	-	*QuotaInterceptor()*
2. 분석요청	GitHub URL 전송 및 검증	POST	*/links/github*
	유효성 검사	-	*validateGitHubUrl()*
	URL 처리	-	*parseOwnerRepo()*
3. 분석	저장소 정보 분석	POST	*/api/activity*
	스캔 서비스 상태	GET	*/api/health*

	보안 스캔 (osv + semgrep)	POST	/api/scan
	README LLM 분석요청	POST	/api/aianalyze
	진행 제어	-	data.event
4. 결과 출력	분석 결과 조회	-	receiveRepositoryUrl()
	분석 결과 항목별 상세 조회	-	receiveRepositoryUrl()
	분석 기록 조회	GET	/api/resultrecord
	마크다운 리포트 복사	GET	/api/copy

4.3 대표 API 요청·응답 상세

GET /api/userRemainCount — 유저 유형별 분석횟수 조회

서비스의 인증 핵심으로, Firebase 인증 토큰을 기반으로 사용자 유형(게스트/회원)을 식별하고, 유형별 잔여 분석 횟수를 조회 및 관리합니다.

성공 응답 (200 OK):

```
{
  "success": true,
  "message": "잔여 분석 횟수 조회에 성공했습니다.",
  "data": {
    "userType": "user",
    "remainCount": 17
  }
}
```

에러 응답 (401 Unauthorized):

```
{
  "success": false,
  "message": "인증에 실패했습니다.",
  "error": {
    "code": "INVALID_TOKEN",
    "detail": "Firebase 인증 토큰이 유효하지 않거나 만료되었습니다."
  }
}
```

POST /api/activity — 저장소 정보 분석

서비스의 핵심 진입점 중 하나로, 사용자가 입력한 GitHub URL와 유저의 uid를 서버로 전송하여 github 저장소 정보 분석 작업을 내용을 생성합니다.

요청 (Request Body):

```
{
```

```
    "repositoryUrl": "https://github.com/StayPotato/WebGoat_1",
    "uid": "guest_5g5cpsik"
}
```

성공 응답 (200 OK):

```
{
  "owner": "string",
  "repoName": "string",
  "description": "string",
  "primaryLanguage": "string",
  "stars": 0,
  "forks": 0,
  "openIssues": 0,
  "pushedAt": "2026-03-30T07:46:35",
  "hasReadme": true,
  "readmeContent": "string (markdown)"
}
```

에러 응답 (500 Internal Server Error):

```
{
    "errorCode": "INTERNAL_ERROR",
    "message": "서버 내부 오류가 발생했습니다.",
    "timestamp": {time}
}
```

POST /api/scan — 보안스캔 (osv + semgrep)

서비스의 핵심 진입점 중 하나로, 사용자가 입력한 GitHub URL을 서버로 전송하여 분석 작업을 생성합니다. 이후 모든 조회 API는 이 요청에서 발급된 analysisId를 기반으로 동작합니다.

요청 (Request Body):

```
{
  "gitUrl" : "https://github.com/spring-projects/spring-boot"
}
```

성공 응답 (200 OK):

```
{
  "scanId": UUID,
  "summary": {
    "osv": {
      "totalUniqueVulnerabilities": 19,
      "affectedPackages": 4,
      "bySeverity": { "CRITICAL", "HIGH", "MODERATE", "LOW" },
      "packages": ["name", "version", "ecosystem", "vulnCount",
"maxSeverity", "scope"]
    },
    "semgrep": {
      "totalFindings": 165,
      "bySeverity": { "ERROR", "WARNING", "INFO" },
      "topRules": ["rule", "count"]
    }
  }
```

```
}
```

에러 응답 (error):

```
{
    "errorCode": "PRIVATE_REPOSITORY",
    "message": "Repository not found or access denied."
}
```

POST /api/aianalyze — README LLM 분석요청

서비스의 핵심 진입점 중 하나로, /api/activity에서 받아온 Readme 데이터를 LLM에게 전송하여 분석 작업을 생성합니다.

요청 (Request Body):

```
{
  "repoName": "KISIA_API_study",
  "repoUrl": "https://github.com/StayPotato/KISIA_API_study",
  "readmeContent": "string (markdown)"
}
```

성공 응답 (200 OK):

```
{
    "installationGuide": true,
    "usageExample": true,
    "troubleshooting": false,
    "qualityScore": 62,
    "summary": "string",
    "classification": "library"
}
```

에러 응답 (400 Bad Request):

```
{
    "errorCode": "VALIDATION_ERROR",
    "message": "GitHub 저장소 URL 형식이 아닙니다."
}
```

5. 기술 스택

5.1 기술 스택 전체 구성

영역	담당자	기술 스택	선정 이유
프론트엔드	김이찬	React.js	분석 데이터 실시간 시각화, 컴포넌트 재사용성
		JavaScript, Axios	빠른 UI 개발과 일관된 화면 구성, 반응형 대시보드 효율적 구현
인증	김주환	Firebase Auth+ JWT	소셜 로그인 간편 적용, 백엔드 토큰 검증

백엔드	김준서·김이찬	**Java + Spring Boot**	대량 API 호출(GitHub·LLM·스캔 도구)과 비즈니스 로직의 안정적 처리
		Spring Security	인증·인가와 API 보호 체계적 관리, 게스트/회원 접근 제어
		WebClient (비동기)	GitHub API 대량 비동기 호출 최적화
보안 스캔	김이찬·심도연	**OSV-Scanner**	의존성 CVE 전수 조사
		Semgrep	소스코드 정적 분석·보안 허점 탐지
데이터 수집	김준서·심도연	**GitHub REST API**	Stars·Forks·Commits·Issues·README 수집
AI / LLM	황세웅	**LLM(ChatGPT, Gemini, Claude) API**	README 요약·정성 분석·프레임워크 판단
결과 출력	황세웅	**Clipboard API**	마크다운 결과 즉시 복사 → Notion·Slack 활용
배포·운영	김이찬	**Docker + Linux**	실행 환경 일관성 유지, 배포 편의성 확보
		Firebase Hosting	프론트엔드 정적 파일 배포

5.2 기술 스택 선정 기준

모든 기술 스택은 다음 기준을 공통 적용하여 선정하였습니다.

기준	내용
라이선스	MIT · Apache 2.0 계열 우선 선정, LGPL 등 제한적 라이선스는 수정 범위 사전 검토
유지보수 활성도	**GitHub Star 수, 최근 커밋 빈도, 릴리즈 주기 확인**
커뮤니티 응답성	이슈·PR 응답 속도, Stack Overflow 등 외부 지원 생태계 규모 검토
프로젝트 적합성	서비스 목적(도입 전 진단)에 부합하는 기능 제공 여부 판단

5.3 오픈소스 라이선스 검토

오픈소스	라이선스	상업적 사용	수정 시 공개 의무
React.js / Next.js	MIT	✅ 가능	없음
Firebase(Google)	Apache 2.0	✅ 가능	없음
OSV-Scanner	Apache 2.0	✅ 가능	없음

Semgrep	LGPL 2.1	✅ 가능	라이브러리 자체를 수정하지 않으면 공개 의무 없음
React.js / Next.js	MIT	✅ 가능	없음
GitHub REST API	ToS (공개 API)	✅ 가능	Rate Limit 준수 필요

6. CI/CD

6. CI/CD 파이프라인 및 개발 흐름

단계	주요 활동
① 기획	서비스 목표 정의, 요구사항 분석, GitHub API 수집 범위 및 LLM 요약 기준 수립
② 설계	API-First 명세 확정으로 프론트·백 병렬 개발 기반 마련, React — Spring Boot — Firebase 간 데이터 흐름 설계
③ 구현	FE: React.js 기반 UI·대시보드 구현 / BE: Spring Boot API·LLM 연동 모듈 개발 / WebClient 비동기 통신 적용
④ 검증·테스트	API 단위 테스트, LLM 요약 정합성 확인, 프론트·백 통합 테스트
⑤ 배포·운영	Docker 컨테이너화, Linux 서버 + Firebase Hosting 배포

7. 백엔드 디렉토리 구조 및 SBOM

7.1 Spring Boot 백엔드 디렉토리 구조

```
src/main/java/com/kisia/gitchecker/
├── GitcheckerApplication.java          # Spring Boot 기동
├── config/
│   ├── AsyncConfig.java                # 비동기 처리 설정
│   ├── FirebaseConfig.java             # Firebase Admin SDK 초기화
│   ├── SecurityConfig.java             # Spring Security 필터 체인·CORS
│   └── WebMvcConfig.java               # Interceptor(횟수 제한) 등록
├── controller/
│   ├── ActivityController.java         # 활성도 분석 요청
│   ├── AiController.java               # AI 분석 요청
│   ├── AuthController.java             # 로그인/로그아웃/게스트 세션
│   └── ScanController.java             # 보안 스캔 요청
├── dto/
│   ├── request/
```

```
│   │   ├── AnalysisUrlRequest.java       # 분석 URL 요청 DTO
│   │   └── ScanRequest.java              # 보안 스캔 요청 DTO
│   └── response/
│       ├── AiAnalysisResponseDto.java    # AI 분석 응답 DTO
│       ├── AnalysisUrlResponse.java      # URL 분석 응답 DTO
│       ├── RepositoryInfoResponseDto.java # 저장소 정보 응답 DTO
│       └── ScanResponse.java             # 보안 스캔 응답 DTO
├── exception/
│   ├── GlobalExceptionHandler.java       # 전역 예외 처리
│   ├── PrivateRepositoryException.java   # 비공개 저장소 예외
│   └── ScanException.java                # 스캔 실패 예외
├── jwt/
│   ├── JwtAuthFilter.java                # JWT 토큰 검사 미들웨어
│   └── QuotaInterceptor.java             # 게스트/회원 횟수 제한
├── service/
│   ├── ActivityService.java              # 활성도 지표 수집·분석
│   ├── AiService.java                    # LLM API 연동·프롬프트 생성
│   ├── AuthService.java                  # Firebase 토큰 검증·권한 부여
│   ├── FirebaseDbService.java            # Firebase DB CRUD
│   ├── GithubApiClientService.java       # GitHub REST API 통신
│   ├── MarkdownService.java              # 마크다운 리포트 생성
│   └── ScanService.java                  # 보안 스캔 실행·결과 처리
└── resources/
    ├── application.yaml                  # 애플리케이션 설정
    └── gitchekcer-firebase-adminsdk-*.json # Firebase 인증 키
```

7.2 SBOM (CycloneDX 1.5) 구성 요약

CycloneDX Gradle 플러그인을 통해 빌드 시 SBOM을 자동 생성하도록 구성하였습니다. 생성된 SBOM에는 모든 직접·간접 의존성의 컴포넌트명, 버전, 라이선스, 해시값이 포함됩니다.

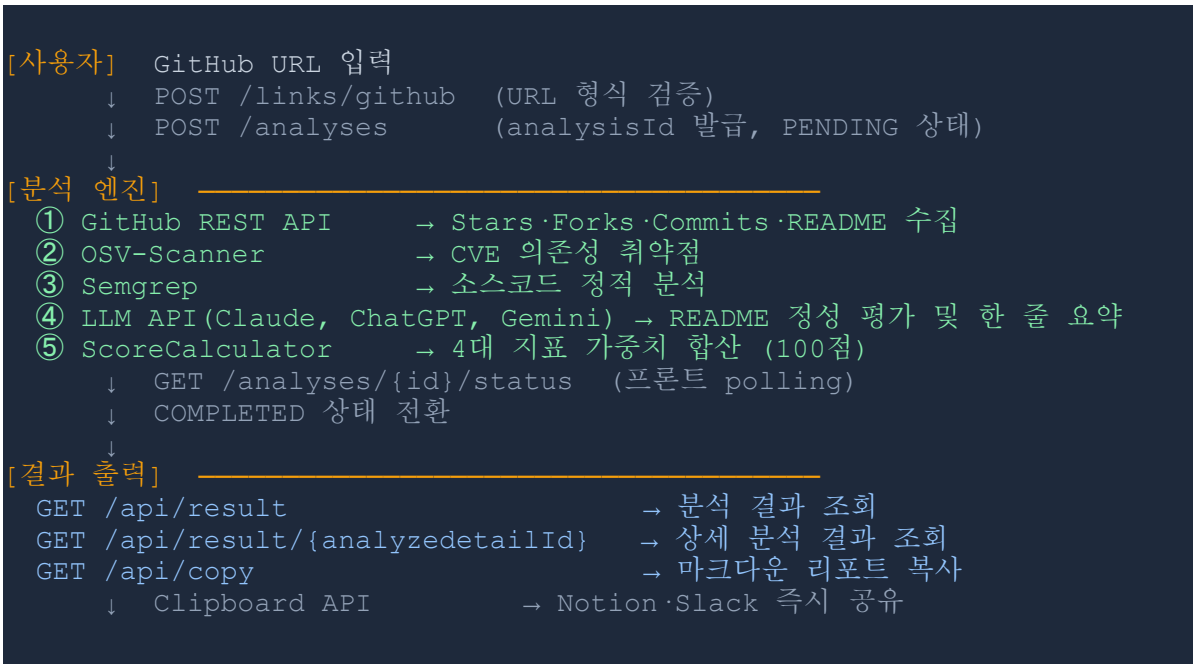

```
{
  "bomFormat": "CycloneDX",
  "specVersion": "1.5",
  "serialNumber": "urn:uuid:6612d444-24b6-4fd9-a552-e52af89dcbab",
  "version": 1,
  "metadata": {
    "timestamp": "2026-03-31T15:29:16Z",
    "component": {
      "group": "com.kisia",
      "name": "gitchecker",
      "version": "0.0.1-SNAPSHOT",
      "type": "application"
    }
  },
  "components": [
    {
      "group": "org.springframework",
      "name": "spring-aop",
      "version": "7.0.6",
```

```
    "licenses": [{ "license": { "id": "Apache-2.0" } }],
    "type": "library"
   },
   {
    "group": "commons-codec",
    "name": "commons-codec",
    "version": "1.19.0",
    "type": "library"
   }
  ]
}
```


※ 위 내용은 자동 생성된 SBOM(sbom.json)의 구조 발췌이며, 해시·라이선스 전문 등 상세 데이터를 포함한 전체 파일은 프로젝트 저장소에 포함되어 있습니다.

8. 분석 파이프라인 전체 흐름

사용자가 GitHub URL을 입력하는 순간부터 마크다운 리포트를 복사하기까지의 전체 흐름입니다.



참고 자료

[1] KISIA S-개발자 4기 교육자료. (2026). Day 1: API 설계편 — 오픈소스 활용. Day1_API_Design_v3.pdf

[2] Sonatype. (2024). 10th Annual State of the Software Supply Chain Report. sonatype.com

[3] OpenSSF Best Practices Working Group. Concise Guide for Evaluating Open Source Software. best.openssf.org

[4] Zhao, Y. et al. (2021). "Evaluation indicators for open-source software: a review." Cybersecurity, Springer.

[5] NIPA. (2022). 2022 오픈소스SW(OSS) 실태조사 보고서.