정 보 보 호 학 회 지
제 32 권 제5호, 2022. 10

OSS 추적성을 위한 SBOM 동향

김 선 우*, 손 경 호**

요 약

최근 몇 년 동안 엄청난 양의 데이터 혁신이 진행되어왔고, 그에 따라 소프트웨어 개발의 편리성을 위해 오픈소스를 사용하는 경우가 많아졌다. 이로 인해 소프트웨어 생산성 측면에서는 많은 도움이 되었지만, 보안 관점에서는 많은 문제를 야기했다. 이러한 OSS 사용에 따른 위험을 줄이고자 OSS 추적성을 위한 도구를 사용하는 방법이 지속적으로 개발되었지만, 아직까지도 OSS 사용에 따른 위험은 증가하고 있다. 이에 본 논문은 OSS 추적성의 보완을 위한 SBOM(Software Bill of Materials)의 정의와 현재 국외 SBOM 추진 동향에 대해 소개하고자 한다.

Ⅰ. 서 론

최근 몇 년 동안 엄청난 양의 디지털 혁신이 진행되어왔다. 과거에는 수많은 수동적 상호 작용이 필요했던 활동들이 이제는 간단한 작업으로도 쉽게 완료할 수 있다. 디지털 혁신은 사용자에게 매우 편리하게 다가갔지만, 광범위한 개발 시간과 노력이 필요하다. 이러한 문제를 해결하기 위해 최신 소프트웨어 시스템은 다른 사람이 만든 입증된 코드를 재사용하는 경우가 늘어났다.

라이브러리의 사용은 소프트웨어 프로젝트에 대한 의존성을 생성하여 현대 소프트웨어 엔지니어링의 중추를 이루게 되었다. 또한, 사용되는 각 라이브러리에는 고유한 의존성 집합이 있을 수 있으므로 전이 의존성이 생성된다. 따라서 소프트웨어 프로젝트에 대해 가져온 코드의 양은 개발자가 직접 작성한 코드에 비해 수십 배 초과할 수도 있다.

이러한 재사용 기회는 소프트웨어 생산성에 도움이 되지만 위험이란 대가를 치르게 되었다. 보안 관점에서 볼 때 이러한 각 의존성은 전체 프로젝트의 보안에 영향을 줄 수 있는 취약한 코드를 포함할 수 있기 때문에 부정적인 결과를 초래할 수 있다. 예를 들어 2017년 Equifax 데이터 침해 사건(CVE-2017-5638)과 2021년 발생한 Log4j 보안 취약점 사태와 같은 대규모 데이터 침해 사고는 오픈소스에 대한 경각심을 일깨워주었다[1].

이러한 오픈소스 사용에 따른 위험들은 소프트웨어 프로젝트의 의존성을 최신 상태로 유지하는 것의 중요성을 알려주었지만, 의존성 네트워크의 크기가 증가함에 따라 의존성을 업데이트하는 것은 점점 더 어려워지고 있다. 또한, 공개된 취약성의 수가 너무 많기 때문에 새로운 보안 문제에 대해 수동적으로 최신 의존성을 최신 상태로 유지하는 것은 불가능하다.

이로 인해 OSS 취약성을 수동으로 검사하는 것보다 소프트웨어 프로젝트에서 취약점 의존성을 식별하는 취약성 검사 도구들을 사용하는 것이 일반적이었다. 이로 인해 의존성의 취약성을 수동으로 하는 것보다 더욱 개선되었지만, 이러한 도구는 취약한 의존성의 실제 기능 대신 전체 의존성을 취약성으로 식별하는 것과 같은 문제가 있다. 또한, OSS 사용 여부가 제대로 확인되지 않거나, 소스 코드를 확보하기 어려운 외주 개발 모듈 등에 대해서는 OSS가 사용되었는지조차 확인하기 어려울 수 있다.

본 논문은 이러한 문제를 해결하기 위한 새로운 해결방식인 SBOM(Software Bill of Materials)의 정의와 현재 국외 SBOM 작성 및 관리 도구의 동향에 대해 소개한다.

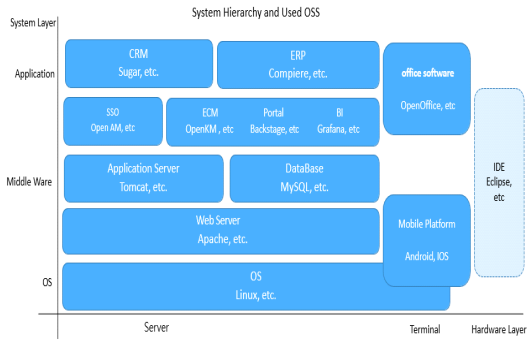
본 연구는 2022년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행되었습니다. (No. 2022-0-00277, SW 공급망 보안을 위한 SBOM 자동생성 및 무결성 검증기술 개발).
* 강원대학교 융합보안학과 (대학원생, kssw422@kangwon.ac.kr)
** 강원대학교 (교수, khson@kangwon.ac.kr)

Ⅱ. Open Source Software 취약성

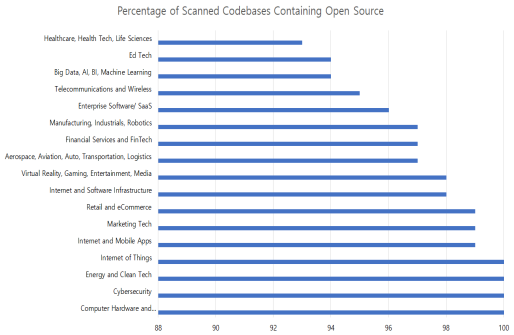
OSS는 시스템의 모든 계층에서 사용된다. 그림 1은 시스템의 각 계층 구조에 있어서 OSS의 사용 현황을 보여주며, 운영체제나 미들웨어, 애플리케이션, 하드웨어에서도 OSS가 사용되고 있는 것을 확인할 수 있다.

Synopsys에서 발표한 2022 오픈소스 보안과 리스크 분석(OSSRA)[2] 보고서에 따르면 오픈소스는 모든 산업에서 널리 사용되고 있으며 현재 운영되고 있는 대부분의 애플리케이션에 기반을 제공하는 것을 확인할 수 있다. 또한, 운영 리스크/유지보수 측면에서 2,097개의 코드 베이스 중 85%가 4년 이상 지난 오픈소스를 포함하는 것으로 나타났다.

이렇듯 OSS의 사용이 증가함과 동시에, OSS 취약성에 대한 관심과 보안에 관련한 연구들이 활발히 진행 중이다. 본 절에서는 OSS의 사용으로 인한 위험과 종류, OSS 취약성 관리 도구에 대해 소개한다.



〔그림 1〕 System Hierarchy and Used OSS



〔그림 2〕 Percentage of Scanned Codebases Containing Open Source

2.1. OSS 위험

Synk에서 발표한 2022 오픈소스 보안 보고서와 Synopsys에서 발표한 2022 오픈소스 보안 위험 분석 보고서에 있는 내용을 토대로 OSS 사용의 위험의 원인은 7가지로 나눌 수 있다[2][3].

- 공개된 지식 : OSS 취약점은 OWASP(Open Web Application Security Project)와 NVD(National Vulnerability Database) 같은 조직을 통해 쉽게 알 수 있음[4][5]
- 보안 부족 : OSS는 보안에 대한 요구 사항이나 법적 의무가 제대로 이뤄지지 않으며, 안전하게 구현하는 방법을 알려주는 커뮤니티 지원이 부족함
- 지적 재산권 문제 : OSS에 적용할 수 있는 라이선스의 유형은 많지만, 라이선스 중 상당수는 서로 호환되지 않음. 그러므로 사용하는 구성요소가 많아지고, 사용하는 구성요소가 많을수록 모든 라이선스 규정을 추적하고 비교하기가 더 어려워짐
- 보증 부족 : OSS는 보안, 지원 또는 콘텐츠에 대해 어떠한 보증도 하지 않음
- 느슨한 통합 감독 : OSS 개발팀은 서로 사용하는 기능이나 라이선스에 대해 모를 수 있음
- 운영상의 미흡 : OSS 사용에 대한 책임자가 명확하지 않은 경우가 많음
- 열악한 개발자 관행 : 개발자가 OSS에서 코드의 섹션을 복사 붙여 넣을 경우 실수로 위험을 증가시킬 수 있음

또한, 취약성과 관련되어 CVE(Common Vulnerabilities and Exposures)는 공개적으로 알려진 보안 취약성 및 노출에 대한 정보를 나타내고 있으며, 이에 대한 취약성을 CVSS(Common Vulnerability Scoring System)를 통한 점수화로 취약성에 대한 등급을 매기고 있다.

OSS에 대한 취약성은 지속적으로 발생해왔으며 mend의 OSS 취약점 데이터베이스를 확인해보면 CVSS 최대 점수에 해당하는 CVE가 순위권 중 2022년도에만 4건이 발생했을 정도로 취약점에 노출된 것을 확인할 수 있다[5][6].

표 1은 OSS를 사용하였을 때의 위험한 취약점의

〔표 1〕 Typical CVE Types

CVE Number	Vulnerability	CVSS	CVSS Vector
CVE-2020-0187	BaseBlockCipher.java	5.5	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N
CVE-2020-11023	In jQuery versions greater than or equal to 1.0.3 and before 3.5.0	6.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CVE-2020-11022	In jQuery versions greater than or equal to 1.2 and before 3.5.0	6.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CVE-2019-11358	jQuery before 3.4.0	6.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CVE-2015-9251	jQuery before 3.0.0 is vulnerable to Cross-site Scripting (XSS)	6.1	CVSS:3.0/AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N
CVE-2020-8203	Prototype pollution attack when using _.zipObjectDeep in lodash before 4.17.20.	7.4	CVSS:3.1/AV:N/AC:H/PR:N/UI:N/S:U/C:N/I:H/A:H
CVE-2020-28500	Lodash versions prior to 4.17.21 are vulnerable to Regular Expression Denial of Service (ReDoS)	5.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:N/A:L
CVE-2020-7788	This affects the package ini before 1.3.6.	7.3	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L
CVE-2018-14719	FasterXML jackson-databind 2.x before 2.9.7	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVE-2018-1000613	Legion of the Bouncy Castle Java Cryptography API 1.58 ~ 1.60	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVE-2015-7501	A crafted serialized Java object, related to the Apache Commons Collections (ACC) library.	9.8	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVE-2020-8022	A Incorrect Default Permissions vulnerability in the packaging of tomcat on SUSE Enterprise Storage 5	7.8	CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H
CVE-2019-10744	Versions of lodash lower than 4.17.12 are vulnerable to Prototype Pollution.	9.1	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:N/I:H/A:H
CVE-2017-9224	An issue was discovered in Oniguruma 6.2.0	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVE-2017-9225	An issue was discovered in Oniguruma 6.2.0	9.8	CVSS:3.0/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H
CVE-2017-9226	An issue was discovered in Oniguruma 6.2.0	9.8	CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

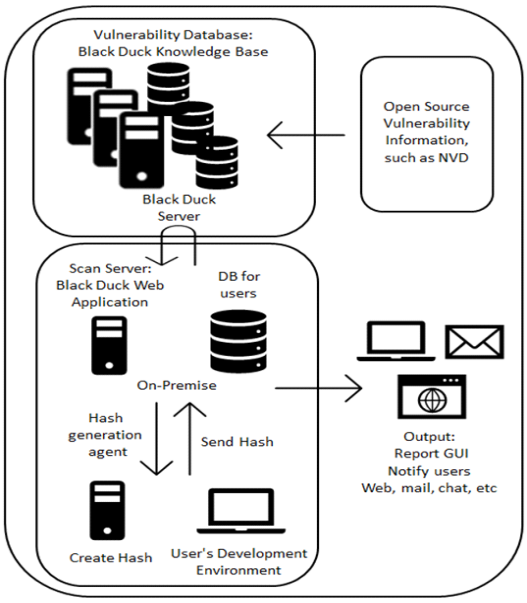
예시와 설명을 나타낸다. 기존에 이미 알려진 취약점들이지만 앞서 언급된 여러 이유로 인해 여전히 위험도가 높다.

2.2. OSS 취약성 관리 도구

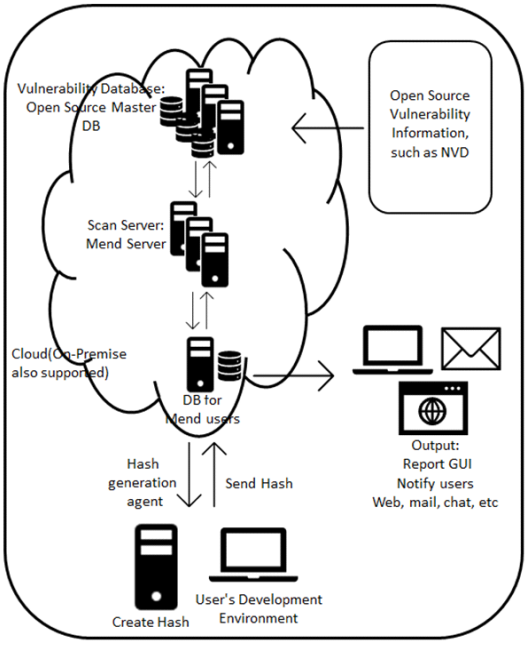
OSS 취약성 관리에 관한 정책이나 국제 표준은 명확하지 않아 기존 OSS 취약성 관리는 도구를 이용하는 경우가 많았다. 이번 절에서는 OSS 취약성 관리 도구에 대해 설명한다.

OSINT(Open-source intelligence)를 통해 얻은 OSS 취약성 관리 도구에 요구되는 주요 사항들은 다음과 같다.

- 스캔 대상의 폭과 정밀도 : OSS가 또 다른 OSS를 캡처하는 Deep License의 검증 및 패키지별 종속성
- 스캔 가능한 프로그래밍 언어(패키지 관리자) 및 OS 배포 범위 : 소스코드/바이너리, 패키지/컨테이너, 서버, 임베디드/웹 등에 대한 대응 범위로 대응 범위가 넓어도 정밀도가 나쁘면 문제 발생함
- 검출부의 투시도 정밀도, DB 내 데이터와의 매칭 정밀도 : 검출의 백그라운드에서 도출되는 취



〔그림 3〕 structure of Black Duck



〔그림 4〕 structure of mend

약성 ID, 패키지 ID 등 Key 정보가 되는 중간 데이터의 유무 등을 포함

- 취약성 DB의 양과 품질 : NVD 이외의 DB의 출처를 얼마나 가지고 있는지
- 취약성 감지 후의 핸들링, 유용성
- 새로운 취약성이 감지되었을 때의 통지 기능 및 유연성 : 검사에 의해 검출된 취약점의 시각화와 사용자가 사용하지 않는 부분의 검출 결과 배제
- 탐지된 취약점의 트리아지 기능 유무
- 스캔 및 DB 구축 시의 처리 속도
- CI/CD 플랫폼과의 연계 가능 여부
- 각종 시스템과의 연계 가능 여부

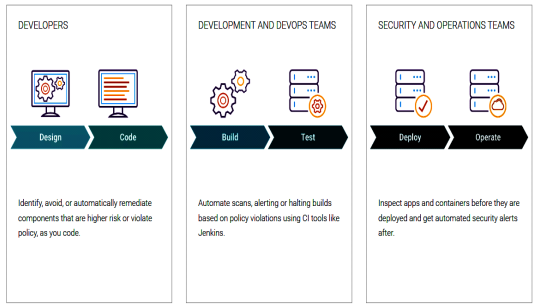
널리 사용되는 OSS 취약성 관리 도구로는 Synopsys의 Black Duck과 WhiteSource의 Mend가 있다. 그림 3은 Black Duck의 구성을 나타내며, 그림 4는 Mend의 구성을 나타낸다. Black Duck은 온프레미스 환경에서 제공하고 Mend는 클라우드 서비스로 제공하는 것을 표준 서비스로 한다.

2.2.1. Black Duck

Black Duck은 미국 Synopsys에서 만든 SCA(Software Composition Analysis) 도구로 애플리케이션 및 컨테이너에 포함된 OSS와 타사 코드에서 발생하는 보안, 품질, 라이선스 컴플라이언스의 리스크 관리를 지원한다. Black Duck의 다중 요소 오픈소스 감지와 4백만 개 이상의 구성요소로 구성된 KnowledgeBase는 모든 애플리케이션 또는 컨테이너의 구성에 대한 완전한 가시성을 제공한다[7].

Black Duck의 특징은 다음과 같다.

- 항상 최신 OSS 정보 확인 가능 : Cybersecurity Research Center(CyRC)가 수집한 400만 건 이상의 OSS 구성요소를 포괄하는 OSS 프로젝트, 약 2,650개의 라이선스 유형, 13만 건 이상의 취약성 정보에 대한 포괄적인 데이터베이스 Black



〔그림 5〕 Integrate and automate open source governance into DevSecOps

〔표 2〕 Introducing open source vulnerability analysis and management tools

Tool	Features
Synk	• Create project snapshots to enable continuous monitoring • Includes secret detection to detect and highlight passwords such as keys, credentials, and PII • Unlike tools that use entropy checks or regular expressions, AI machine learning improves the probability of accurate detection
mend	• Digital Signature Open Source Management • Database operates on SaaS, updating security vulnerability details in real time • File hash value extraction technology and encrypted communication enable detection of source code without risk of gastric leakage
BlackDuck	• Support for Snippet detection algorithms • Various analysis techniques such as Signature Matching, Dependency Matching, and String Matching • Always keep up to date with OSS information
Labrador	• Use VUDDY technology to identify function vulnerabilities • Detect modified open source components using CENTRIS technology • Provides breakthrough accuracy with component, file, and functional 3-layer vulnerability analysis
Veracode SCA	• Agent-based scanning techniques • Code scanning method based on software development life cycle (SDLC) • Vulnerable Method Detection • Enables scanning of Docker containers and images • Policy assessment, analysis, and reporting capabilities

Duck Security Advisories(BDSA)를 지속적으로 업데이트함

- 다양한 프로그램 환경 대응 : 소스코드가 있으면 언어에 관계없이 해시 값으로 매칭 가능
- 다양한 도구와의 연계성 : IDE, 패키지 매니저, CI/CD 등 대부분의 대중적인 개발 도구나 REST API에 대해 사용하기 쉬운 OSS 통합 연계 가능
- 다각적이고 포괄적인 위험 감지 : 이진, 컨테이너 이미지의 해석 외에 코드에 포함된 OSS의 스니펫까지도 특정해 가시화 가능

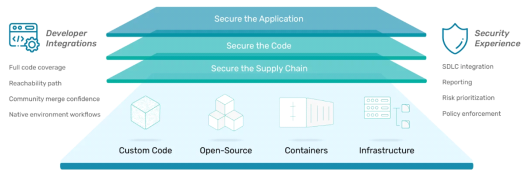
Black Duck은 DevSecOps에 오픈소스 거버넌스 통합 및 자동화도 지원한다. Black Duck 자동 정책 관리를 사용하면 오픈소스 사용, 보안 위험 및 라이선스 컴플라이언스에 대한 정책을 미리 정의하고 개발자가 이미 사용하는 도구를 사용하여 SDLC(software development life cycle) 전반에서 시행을 자동화할 수 있다.

2.2.2. Mend

WhiteSource의 mend는 소프트웨어에서 사용되고 있는 오픈소스의 구성요소를 식별하여 더욱 쉽고 효율적으로 관리할 수 있는 오픈소스 보안 취약점 및 라이선스의 규정을 점검해 주는 플랫폼이다. mend는 세계 최초 오픈소스 보안 취약점 자동 탐지 솔루션으로 업계에서 가장 포괄적인 오픈소스 취약점 데이터베이스를 보유하고 있다[8].

mend의 특징은 다음과 같다.

- 정확한 데이터베이스와의 매칭 : 오검출 회피를 위해 전 세계의 OSS 정보를 데이터베이스화
- 다양한 프로그램 환경에 대응 : 20개 이상의 프로그래밍 언어 개발 환경을 지원하며 오픈소스 관리 솔루션을 제공함
- 다양한 도구와 시스템 환경의 통합 지원 : 일반적인 빌드 도구나 CI 서버와 통합하여 OSS 관리가 가능함
- 운용 후의 지속적인 감시 : 자동 트래킹 및 Alert 기능을 제공함



〔그림 6〕 Discover the Mend Application Security Platform

mend는 정적 코드 분석 플랫폼인 mend SAST와 소프트웨어 구성 분석 플랫폼인 mend SCA, 공급망 보안 플랫폼인 mend Supply Chain Defender를 지원하여 AppSec 결과를 개선한다.

2.3. 기존 OSS 관리 도구의 한계

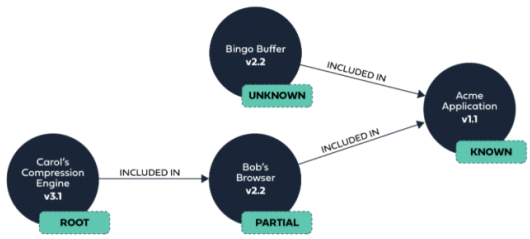
표 2는 앞서 설명한 BlackDuck과 mend를 포함한 상용화된 오픈소스 취약성 분석 및 관리 도구 5가지를 선별하였고, 이에 대한 간단한 소개를 나타낸다. 현재까지 다양한 OSS 취약성 분석 및 관리 도구가 상용화가 되었지만, 여전히 OSS 추적성을 위한 한계가 존재한다.

기존의 OSS 관리 도구는 취약점이 발견되면 감염된 라이브러리를 사용하는 애플리케이션을 식별하는데 시간이 많이 걸리고, 크기에 따른 시간 지연과 같은 단점이 존재한다. 또한, 취약점이 있는 라이브러리를 사용하는 애플리케이션을 식별해도 오탐지 가능성과 OSS의 가시화가 부족한 경우도 존재한다. 이러한 문제로 인해 OSS 추적성을 위한 새로운 방법으로 SBOM(Software Bill of Materials)을 사용하는 방법이 현재 연구중이다.

Ⅲ. SBOM(Software Bill of Materials)

SBOM은 2021년 5월 미국 바이든 행정부의 '국가의 사이버보안 향상에 대한 행정명령(EO 14028)을 통해 널리 알려졌다. SBOM은 식품 패키지에서 모든 재료를 찾을 수 있는 것처럼 소프트웨어 제품에 포함된 모든 구성요소의 목록이다. 공급업체는 일반적으로 구성요소가 무엇인지 설명하기 위해 이러한 청구서를 작성한다. 또한, SBOM에는 이러한 구성요소의 종속성 및 계층 관계에 대한 정보도 포함되어 있다[9].

SBOM은 코드 관리 도구를 통해 배포된 모든 사용자 지정 코드에 대한 버전 및 라이선스 정보 및 커밋



〔그림 7〕 Conceptual SBOM tree with upstream relationship assertions

또는 버전 제어 정보와 함께 모든 타사 구성요소에 대한 세부 정보를 포함한다.

SBOM은 오픈소스와 독점 소프트웨어를 포함할 수 있으며, 배포가 가능하게 하거나 액세스를 제한할 수 있다. 또한, SBOM에는 그림 7과 같이 표준 데이터 형식으로 개별 구성요소를 고유하게 식별이 가능한 기능을 가진 기준선 속성이 포함되어야 한다.

기존 OSS 관리 도구와 비교하여 SBOM은 다양한 장점이 있다. 먼저 SBOM은 애플리케이션을 취약한 종속성에 매핑할 수 있는 기능을 제공하여 기존 OSS 관리 도구의 단점을 보완할 수 있으며, 위험한 오픈소스 사용의 우선순위를 정하는 데 도움을 줘서 알려진 취약점이 있는 라이브러리의 사용을 줄일 수 있다.

NTIA에서는 소프트웨어 공급망 투명성 구조화 보고서를 통해 SBOM의 최소 구성요소를 발표하였다.

〔표 3〕 SBOM Baseline Attributes

Baseline	Description
Author Name	author of the SBOM
Timestamp	date and time when the SBOM was last updated
Supplier Name	name or other identifier of the supplier of a component in an SBOM entry
Component Name	name or other identifier of a component
Version String	version of a component
Component Hash	cryptographic hash of a component
Unique Identifier	additional information to help uniquely define a component
Relationship	association between SBOM components

NTIA에서 발표한 SBOM의 기본 구성요소는 표 3과 같다[10].

SBOM은 아직 표준화된 형식이 존재하지 않아 다양한 SBOM 형식이 존재한다. 그러나 미국 사이버보안 및 인프라 보안국(CISA)과 기타 기관에서 언급된 주요 SBOM 형식은 SWID 태그(Software Identification Tags)와 CycloneDX, SPDX(Software Package Data Exchange)이다. 이중 CycloneDX와 SPDX 형식만이 주로 사용되고 있다. 이에 본 절에서는 CycloneDX와 SPDX SBOM 형식에 대해 소개한다[11][12].

3.1. CycloneDX

CycloneDX는 국제 웹 보안 표준기구(Open Web Application Security Project, OWASP)가 주도하여 만든 SBOM 데이터 형식이다. CycloneDX는 처음부터 BOM(Bill of Materials) 형태로 설계되었다. 또한, CycloneDX는 SaaSBOM(Software-as-a-Service BOM)을 비롯한 다양한 사용 사례를 충족하도록 설계되었다.

CycloneDX는 하드웨어, 클라우드 및 SaaS와 관련하여 현대 소프트웨어 에코시스템의 복잡성에 대응하는 중첩 또는 계층형 접근에 대한 다른 시스템 및 BOM에서의 구성요소, 서비스와 취약점 참조를 해주는 BOM-Link 기능을 지원한다. BOM-Link는 JSON과 XML 형식에서 모두 지원된다. 사용자는 외부 BOM의 URL 또는 외부 BOM의 시리얼 번호와 버전을 사용하는 BOM-Link URI도 참조할 수 있다.

CycloneDX는 CPE(Common Platform Enumeration)와 SWID, PRUL(Package URL)의 세 가지 필드를 통해 취약점을 식별할 수 있으며 SPDX 라이선스 ID와 표현식을 포함한 기존 사양을 통합한다. 또한, CycloneDX는 소스 코드를 쉽게 사용할 수 있고 수정이 가능하며 재배포가 가능한 오픈소스 구성요소의 동적인 특성을 가지고 있다[11][12][13]. 표 4는 CycloneDX SBOM의 구조와 설명을 나타낸다.

〔표 4〕 CycloneDX Document contains and Description

Contain	Description
BOM Metadata Information	Including the tools used to produce the SBOM, the supplier, manufacturer, and the assembled software, component, firmware, or device that the SBOM describes
Components Information	Describe the complete inventory of first-party and third-party components.
Service Information	Describe external APIs that the software may call.
Dependency Relationships	Describes direct and transitive relationships.
Compositions	Describe constituent parts (including components, services, and dependency relationships) and their completeness.
Vulnerabilities	Known vulnerabilities inherited from the use of third-party and open source software and the exploitability of the vulnerabilities can be communicated with CycloneDX.
Extentions	Multiple extension points exist throughout the CycloneDX object model allowing fast prototyping of new capabilities and support for specialized and future use cases.

3.2. SPDX

Linux Foundation에서 운영하는 프로젝트인 SPDX는 공유 및 수집을 위한 소프트웨어 패키지와 관련된 정보에 대해 공통 데이터 교환 형식을 만드는 것이 목적으로 진행된 SBOM 정보를 전달하기 위한 ISO/IEC 표준이다. SPDX는 소프트웨어 구성요소와 라이선스, 관련 구성요소, 저작권 및 보안 정보를 여러 파일 형식으로 쉽게 설명할 수 있다. SPDX는 소프트웨어 패키지 및 관련 구성요소에 대한 정보를 쉽게 수집하고 분석할 수 있도록 RDFa, .xlsx, .spdx, .xml, .json, .yaml과 같은 여러 파일 형식을 지원한다[11][12]. 표 5는 SPDX SBOM의 구조와 설명을 나타낸다.

〔표 5〕 SPDX Document contains and Description

Contain	Description
SPDX document creation information	One instance is required for each SPDX document produced
Package information	A package in an SPDX document can be used to describe a product, container, component, packaged upstream project sources, contents of a tarball, etc
File information	A file's important metadata, including its name, checksum, licenses and copyright, is summarized here.
Snippet information	Snippets can optionally be used when a file is known to have some content that has been included from another original source.
Other licensing information detected	Provides a way to summarize other license information that may be present in software being described, such as custom or proprietary licenses.
Relationships between SPDX elements information	Stores ways in which different specifications can relate to each other.
Annotations information	Used by others to review SPDX documents and communicate information from the review
Review information	The review information section is included for compatibility with SPDX 1.2, and is deprecated since SPDX 2.0

Ⅳ. SBOM 작성 및 관리 도구 동향

본 절에서는 SBOM을 작성하는 대표적인 도구 및 관리 동향에 대해 살펴보도록 한다. 각각의 도구들은 FLOSS(Free/Libre and Open Source Software) 제품으로 누구나 상업적 또는 독점적 제한 없이 소스 코드와 설명서에 접근할 수 있고, 최근까지 활성화된 도구들을 기준으로, 총 4개의 도구를 선정했다. 대표적으로 살펴볼 SBOM 작성 및 관리 도구들은 CycloneDX-CLI, FOSSology, Tern, ScanCode Toolkit이다.

CycloneDX는 응용 프로그램 보안 컨텍스트 및 공급망 구성요소 분석에 사용하도록 설계된 SBOM 사양이다. FOSSology는 리눅스 재단 프로젝트로 오픈소스 라이선스 컴플라이언스 소프트웨어 시스템 및 툴킷이다. Tern은 컨테이너 이미지에 설치된 패키지의 메타데이터를 찾기 위한 검사 도구이다. ScanCode Toolkit은 수백 개의 타사 패키지에 대한 라이선스 및 원본 정보 데이터를 검색하고 정규화하는 툴킷이다[14].

각각의 도구들은 구현 방식과 구성요소에 차이가 있기 때문에 여러 가지 형태를 취할 수 있다. 따라서 획일화된 다섯 가지 분석 기준을 통해 각각의 도구들의 유사성과 차이점을 구별할 수 있도록 하였다. 다섯 가지 분석 기준은 다음과 같다.

- 구성요소
- 라이선스와 저작권 표현 방법
- 의존성 표현 방법
- 메타데이터 표현 방법
- 취약성 표현 방법

4.1. CycloneDX-CLI

CycloneDX는 응용 프로그램 보안 컨텍스트 및 공급망 구성요소 분석에 사용되도록 설계된 경량 SBOM 사양이다. CycloneDX는 apache-2.0 라이선스로 라이선스된 FLOSS(Free/Libre and Open Source Software) 제품이다. 이 제품은 기계가 읽을 수 있는 형식과 SBOM을 취약점, 종속성 그래프 및 BOM 설명자를 염두에 둔 추가 기능과 공유하는 방법을 제공한다. 또한, SPDX 라이선스 목록을 사용하고 SPDX 식별자가 있는 라이선스를 검색한다. CycloneDX는 SBOM을 JSON 또는 XML 파일로 표현하고 구문 분석하기 쉽고 기계가 읽을 수 있는 것을 목표로 한다[15].

4.1.1. 구성요소

BOM(Bill of Materials) 스키마 v1.2의 문서에서 알 수 있듯이 각 FLOSS 종속성은 필요한 정보가 포함된 구성요소로 표시된다[16]. 이름, 게시자, 저작권 등 미리 정의된 유형은 소프트웨어 구성요소만을 설명하는 것이 아니라 하드웨어 장치 또는 소프트웨어 컨테이너가 포함되어 있다. 또한, 구성요소의 종류를 설명하기 위해 MIME 유형을 정의할 수 있다. 이름, 그룹 및 버전과 같은 일반적인 식별 속성 외에도 패키지 PURL(Uniform Resource Locator)을 사용하여 소프트웨어 구성요소를 정확하게 식별할 수 있다.

CycloneDX는 각 구성요소의 해시를 통해 구성요소가 변경되었는지 확인할 수도 있다. 구성요소의 저작권은 간단한 작성자 및 게시자 문자열로 표시되거나 공급자, 자세한 연락처 정보가 있는 보다 정교한 조직 엔티티로 표시될 수 있다.

4.1.2. 라이선스와 저작권 표현 방법

라이선스의 정의는 SPDX 표현 문자열로 구성되며, SPDX 표현 문자열은 서로 다른 규정 준수 정보와 하나 이상의 라이선스 유형 엔티티 구조를 설명한다. 표현식의 예로는 "Apache-2.0 AND (MITOR GPL-2.0-only)"가 있으며, SPDX 식별자, 이름, 라이선스 텍스트 및 라이선스 정의에 대한 URL이 있는 구조화된 라이선스 유형 개체로 표현될 수 있다. 또한 이 정의는 라이선스가 SPDX 라이선스 목록에 없는 경우 라이선스 이름과 텍스트별로 해당 라이선스를 문서화하는 옵션을 설명한다.

4.1.3. 의존성 표현 방법

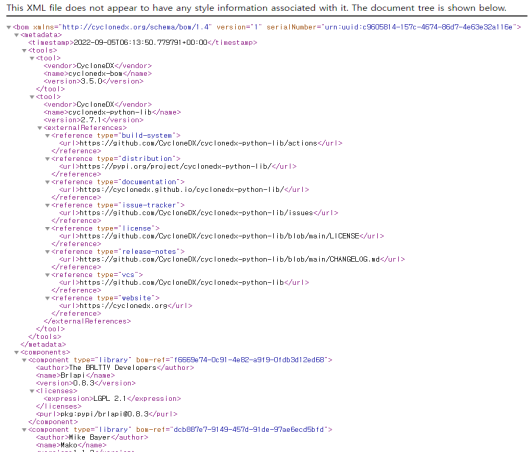
다른 방법으로 다른 구성요소에 대한 종속성을 문서화하는 것이 가능하다. 가장 간단한 방법은 전체 BOM이 있는 구성요소의 구성요소 속성에서 하위 구성요소를 참조하는 것이다. 또 다른 방법은 구성요소의 생성, 배포, 수정 및 재배포를 포함하여 처음부터 끝까지 복잡한 공급망 시나리오를 문서화할 수 있는 가계도 속성을 사용하는 것이다. commit 속성을 사용하여 구성요소가 부모, 자식 또는 변형에서 어떻게 벗어나는지 설명할 수 있다. 마지막 방법은 종속성 그래프 확장을 사용하는 것이다. 구성요소 종속성의 그래프 표현은 구성요소에 대한 참조를 사용하여 구성요소 외에 정의된다.

4.1.4. 메타데이터 표현 방법

메타데이터는 두 가지 방법으로 문서화할 수 있다. 첫 번째는 최상위 BOM 요소의 메타데이터 속성으로, 대부분 전체 BOM의 저자 정보를 기술한다. 또한, BOM 문서 작성에 관련된 문서 도구를 사용할 수 있다. 두 번째는 BOM 설명자 확장은 제3자가 존재할 경우 구성요소에 대한 훨씬 더 신뢰할 수 있는 세부 정보를 문서화하기 위해 제조업체 및 공급업체에 대한 더 많은 메타데이터 옵션을 추가한다. 이 확장은 통합된 v1.2와 마찬가지로 BOM v1.1에만 적용된다. 일반적으로 확장 가능성을 통해 SBOM의 일부가 되어야 하는 사용자 정의 메타데이터를 쉽게 정의할 수 있어야 한다.

4.1.5. 취약성 표현 방법

CycloneDX는 보안 컨텍스트를 염두에 둔 SBOM 사양이라고 설명한다. 취약성 확장은 보안 취약성을 문서화하는 기능을 제공한다. CycloneDX SBOM의 구성요소의 취약성 속성은 식별자 공통 취약성 및 노출(CVE)과 소스에 대한 URL뿐만 아니라 의사 결정 프로세스를 지원하기 위한 점수, 등급 및 권장 사항을 나타낼 수 있다.



〔그림 8〕 CycloneDX-bom creation result file

4.2. FOSSology

FOSSology는 오픈소스 라이선스 컴플라이언스 소프트웨어 시스템 및 도구이다. FOSSology는 2007년 12월에 FLOSS로 공개되었으며 GPL-v2.0 라이선스로 부여되었으며, 이후 FOSSology는 리눅스 재단으로 이전되어 오픈 컴플라이언스 프로그램 중 하나로 자동화 컴플라이언스 툴링의 일원이다.

FOSSology는 소프트웨어를 검색하는 여러 가지 방법을 제공한다. FOSSology는 세 개의 에이전트 Nomos, Monk, Ninka를 사용하며, 각기 다른 접근 방식과 알고리즘을 사용하여 분석된 모든 프로젝트 파일에 포함된 라이선스 정보를 찾는다. 또한, 모든 에이전트가 동일한 라이선스 결정에 동의하는 경우 에이전트의 공동 노력에 따라 자동으로 의사 결정을 내릴 수 있다.

FOSSology의 목표는 단순히 라이선스 컴플라이언스 문제를 검색하고 해결하는 것뿐만 아니라 이전 검색 정보를 향후 검색을 위해 데이터베이스에 저장하여 검색 시간과 성능을 향상시키는 것이다. FOSSology는 SPDX 형식뿐만 아니라 Debian 저작권 형식으로도 보고서 작성이 가능하다[17].

4.2.1. 업로드

FOSSology는 웹 서버이며, 이를 사용하기 위해서는 사용자가 대상 소프트웨어를 업로드해야 한다. 업로드/구성요소는 업로드, 업로드 트리, p-파일의 세 가지 데이터베이스 테이블로 나눈다. 업로드 테이블에는 파일 이름 및 대상 소프트웨어가 업로드된 방법과 같은 기본 데이터가 포함된다. 업로드 트리 테이블에는 업로드된 대상 소프트웨어와 모든 파일 간의 관계가 포함되므로 파일과 상위 소프트웨어 간의 관계가 정확하게 설명된다. 파일 형식 테이블에는 각 파일을 고유하게 구별하기 위한 모든 파일 해시가 포함되어 있다.

4.2.2. 라이선스 및 저작권이 표현되는 방법

FOSSology는 업로드된 소프트웨어의 각 파일을 검사하여 라이선스 일치 여부를 확인한다. 즉, 각 파일에는 가능한 라이선스 목록이 있다. 테이블 p-파일에는 파일 정보와 각 파일의 고유 식별을 보장하는 해시가 포함되어 있다. 라이선스 파일 테이블은 스캐너, 에이전트 또는 사용자가 라이선스 스탬프를 찍은 라이선스 시간을 결정하는 방법에 대한 중요한 정보와 결합된 라이선스와 파일을 모두 연결한다.

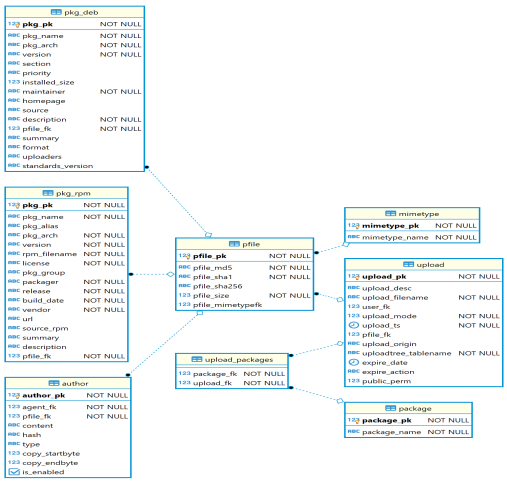
라이선스 참조 테이블에는 알려진 모든 라이선스, 라이선스 이름, GPL-v2 등이 포함되어 있다. 이러한 방식으로, 분석된 프로젝트 라이선스와 포함된 모든 종속성이 라이선스 컴플라이언스를 위해 스캔되었으며, 문제가 식별될 경우 이에 대한 경고를 제공한다. 라이선스는 각 소프트웨어 파일 및 업로드와 관련된 데이터베이스 엔티티로 표시되고 라이선스에 대한 업로드 관계는 분석된 전체 프로젝트 라이선스를 나타낸다. 저작권은 라이선스와 마찬가지로 각 파일에 대해 스캔된다.

4.2.3. 의존성이 표현되는 방법

테이블 업로드 트리는 파일, 디렉터리, 특수 및 해당 업로드 및 패키지에 대한 링크 간의 모든 관계를 설명한다. 파일 모드에서는 모든 파일을 개별적으로 지정할 수 있다. 상위 필드는 파일이 속한 패키지, 프로젝트, 컨테이너 등을 결정하고 업로드 파일이 서로 종속되는 방식을 결정하기 위해 지정된다. 하지만 기술적 관점에서 구성요소가 다른 구성요소와 연결되는 방식을 문서화하는 것은 아직 불가능하고 또한 동적으로 연결된 라이브러리 역시 확인할 수 없다.

4.2.4. 메타데이터가 표현되는 방법

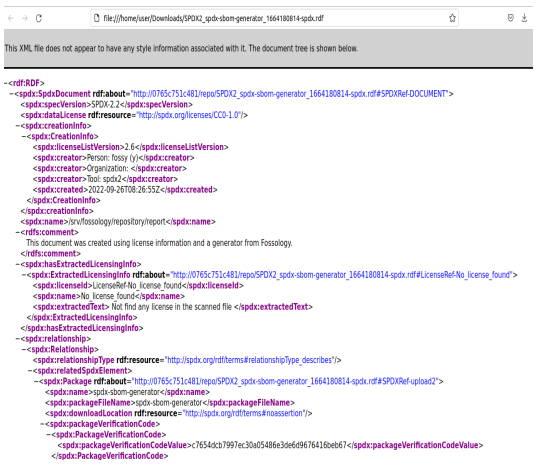
그림 9는 FOSSology RPM과 Debian 패키지 관리자 ERD를 나타낸다. 패키지 에이전트는 RPM과 Debian 패키지 관리자를 사용하여 알려진 패키지를 검색하여 모든 메타데이터를 데이터베이스에 저장한다. 이 다이어그램은 패키지 메타데이터가 데이터베이스에 저장되는 방법을 보여준다. 테이블 pkg rpm은 RPM의 모든 데이터를 저장하고, pkg deb는 debian 패키지 관리자의 모든 데이터를 저장한다. 또한 MIME 유형 표는 각 파일에 더 많은 메타데이터를 추가하는 데 도움이 된다.



〔그림 9〕 FOSSology RPM and Debian Package Manager ERD

4.2.5. 취약성이 표현되는 방법

FOSSology 자체는 취약점을 핵심 사용 사례가 아닌 공급망 내의 도구로 간주한다.



〔그림 10〕 Fossology SPDX Creation Results File

4.3. Tern

Tern은 컨테이너 이미지에 설치된 패키지의 메타데이터를 찾기 위한 검사 도구이다. Tern은 VMWare가 2017년에 시작한 BSD-2 라이선스에 따라 ACT의 멤버로 라이선스가 부여된다. Tern은 다른 것과 달리 패키지 라이선스를 위해 컨테이너와 이미지 계층을 스캔하는 것을 전문으로 하며 파이썬과 셸로 프로그래밍이 되어있다[18].

4.3.1. 구성요소

Tern의 구성요소는 세 가지 주요 요소로 나뉜다.

- 이미지 계층
- 패키지
- 파일

Tern은 Docker 파일처럼 컨테이너를 계층으로 나누는 컨테이너 스캔에 특화되어 있기 때문에 각 이미지 계층에는 포함된 모든 패키지의 목록이 있다. 패키지에는 포함된 모든 파일의 전체 목록과 함께 모든 중요한 정보가 포함되어 있다. 파일에는 파일에 속할 수 있는 여러 패키지 목록을 포함하여 각 파일에 대한 모든 필요한 정보가 포함되어 있다.

4.3.2. 라이선스와 저작권 표현 방법

메타데이터가 Tern 캐시에서 추출되기 때문에 패키지는 선언된 라이선스를 가지고 있다. 다른 하나는 저작권이 있는 패키지에 포함된 모든 라이선스를 나타낸다. 파일의 경우 SPDX 표현식을 통해 다중 라이선스 시나리오를 표현할 수 있는 파일의 모든 라이선스 목록이 있으며 이 경우 목록을 비워 둘 수 있다. 라이선스 준수와 관련된 패키지에 대한 통지가 있는 경우 분석기가 실행되는 동안 채워지는 출처 목록에 표시된다.

4.3.3. 의존성 표현 방법

Tern은 의존성을 각 이미지 계층에 포함된 패키지와 파일의 목록으로 나타내며, 패키지의 경우 포함된 모든 파일의 목록을 포함한다. 이 파일에는 일부 패키지가 다른 패키지와 공통 파일을 가지고 있는 경우 해당 패키지에 속할 수 있는 패키지 목록이 있다.

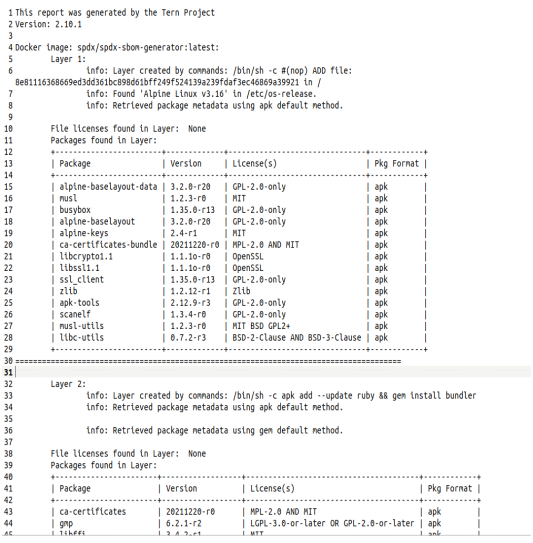
4.3.4. 메타데이터 표현 방법

현재 Docker만 지원하기 때문에 모든 이미지 메타데이터는 Docker 허브에서 가져온다. Docker 이미지 및 인스턴스에는 컨테이너에 대한 모든 메타데이터가

포함된다. 패키지와 파일의 경우 캐시에서 메타데이터가 제공되고 있으며, 누락된 정보가 있을 경우 Tern은 누락된 정보를 채우도록 경고를 발행한다. Tern 데이터 모델이 나타내듯이 각 요소에는 메타데이터가 포함되어 있다.

4.3.5. 취약성 표현 방법

취약성 검사는 CVE Binary Tool 확장에 의해 제공되며, 이 확장 도구는 분석기와 분리된 결과를 보고한다. CVE Binary Tool 확장에 사용할 수 있는 출력 형식은 YAML 및 JSON뿐이며, 분석기 출력 필드를 제외하고 기본 분석기 출력과 같다. 여기에는 각 계층 검색에 대한 추가 정보가 포함된 표로 각 계층에 대한 취약성이 포함되어 있다.



```
This report was generated by the Tern Project
Version: 2.10.1

Docker image: spdx/spdx-sbom-generator:latest:
        Layer 1:
                info: Layer created by commands: /bin/sh -c #(nop) ADD file:
  8e81116368669ed3dd361bc898d61bff249f524139a239fdaf3ec46869a39921 in /
                info: Found 'Alpine Linux v3.16' in /etc/os-release.
                info: Retrieved package metadata using apk default method.

        File licenses found in Layer:  None
        Packages found in Layer:
        +----------------------+-----------+-----------------------------------------+-------------+
        | Package              | Version   | License(s)                              | Pkg Format  |
        +----------------------+-----------+-----------------------------------------+-------------+
        | alpine-baselayout-data | 3.2.0-r20 | GPL-2.0-only                          | apk         |
        | musl                 | 1.2.3-r0  | MIT                                     | apk         |
        | busybox              | 1.35.0-r13 | GPL-2.0-only                           | apk         |
        | alpine-baselayout    | 3.2.0-r20 | GPL-2.0-only                            | apk         |
        | alpine-keys          | 2.4-r1    | MIT                                     | apk         |
        | ca-certificates-bundle | 20211220-r0 | MPL-2.0 AND MIT                       | apk         |
        | libcrypto1.1         | 1.1.1o-r0 | OpenSSL                                 | apk         |
        | libssl1.1            | 1.1.1o-r0 | OpenSSL                                 | apk         |
        | ssl_client           | 1.35.0-r13 | GPL-2.0-only                           | apk         |
        | zlib                 | 1.2.12-r1 | Zlib                                    | apk         |
        | apk-tools            | 2.12.9-r3 | GPL-2.0-only                            | apk         |
        | scanelf              | 1.3.4-r0  | GPL-2.0-only                            | apk         |
        | musl-utils           | 1.2.3-r0  | MIT BSD GPL2+                           | apk         |
        | libc-utils           | 0.7.2-r3  | BSD-2-Clause AND BSD-3-Clause           | apk         |
        +----------------------+-----------+-----------------------------------------+-------------+
================================================================================

        Layer 2:
                info: Layer created by commands: /bin/sh -c apk add --update ruby && gem install bundler
                info: Retrieved package metadata using apk default method.

                info: Retrieved package metadata using gem default method.

        File licenses found in Layer:  None
        Packages found in Layer:
        +----------------------+-----------+-----------------------------------------+-------------+
        | Package              | Version   | License(s)                              | Pkg Format  |
        +----------------------+-----------+-----------------------------------------+-------------+
        | ca-certificates      | 20211220-r0 | MPL-2.0 AND MIT                       | apk         |
        | gmp                  | 6.2.1-r2  | LGPL-3.0-or-later OR GPL-2.0-or-later   | apk         |
```

〔그림 11〕 TERN SBOM Results File

4.4. ScanCode toolkit

일반적인 SW 프로젝트는 수많은 타사 패키지를 재사용하는 경우는 많아 라이선스 및 원본 정보는 항상 쉽게 찾을 수 없고 정규화되어 있지 않다.

Scancode toolkit은 이러한 데이터들을 검색하고 정규화 해주며 Apache-2.0, CC0-1.0 및 MIT, GPL-3.0, BSD와 같은 플러그인에 대한 여러 라이선스로 라이선스가 부여된 FLOSS 제품이다. 그리고 Scancode toolkit은 라이선스 검색 분야에서 가장 오래되고 많이 사용되는 제품 중 하나이다. ScanCode toolkit의 검색 결과에는 각 파일에 대한 파일 경로, 탐지된 라이선스, 저작권 등의 정보가 포함된다[19].

4.4.1. 구성요소 정의

Scancode toolkit은 구성요소를 패키지 및 파일로 간주한다. 이 모델에는 필요한 정보가 들어있는 기본 패키지가 있다. 기본 패키지의 후속 패키지인 패키지는 모든 기본 패키지 정보와 훨씬 확장된 정보 옵션을 가진 메소드를 상속한다. Scancode toolkit에는 패키지 클래스에서 상속된 패키지 유형의 확장된 목록이 있으며 이 목록에 지정된 정보가 있다. 만약 파일이 여러 패키지에서 반복되는 경우 파일에는 패키지 목록을 포함하여 라이선스, 저작권, 소유자 및 작성자 등 모든 중요한 정보가 포함되어 있다.

4.4.2. 라이선스와 저작권이 표현되는 방법

알려진 라이선스 목록은 검색 중에 참조할 수 있도록 라이선스 모듈 내부 데이터 디렉토리에 저장된다. 런타임과 관련하여, 모델은 기본 규칙과 연결하거나 SPDX 규칙과 같이 한번 사용자 정의하기 위해 실행 가능한 정보를 보유하는 라이선스 클래스를 가지고 있다.

4.4.3. 의존성이 표현되는 방법

Scancode toolkit은 다른 제품들과 달리 패키지 수준과 파일 수준 모두에 대한 의존성을 모델링한다. 종속 패키지 클래스 목록을 사용하는 Scancode Toolkit은 검색된 패키지와 관련된 모든 패키지를 URL과 함께 설명한다.

4.4.4. 메타데이터가 표현되는 방법

스캔 단계 동안, Scancode toolkit은 사용 가능한 소스 코드로부터 패키지에 대한 메타데이터를 수집한다. 이 데이터는 소스 코드의 상태 및 공급업체의 유지관리 상태에 따라 달라진다. 패키지 클래스에는 분석된 프로젝트의 각 패키지에 대한 메타데이터가 가장 많이 포함되어 있다. Scancode toolkit은 공급업체, 개발자 및 커뮤니티와 같은 책임 당사자의 목록 외에도 모든

중요한 연락처 정보와 함께 제공한다.

4.4.5. 취약성이 표현되는 방법

ScanCode Toolkit 로드맵에는 취약성 검색을 위해 VulnerableCode라는 공급망 도구가 포함되어 있고 CVE 및 NVD를 사용하며, 여전히 진행 중이다.

```
# Document Information

SPDXVersion: SPDX-2.2
DataLicense: CC0-1.0
DocumentNamespace: http://spdx.org/spdxdocs/scancode_toolkit_v31_1_0-df0f7652-4614-4716-a50b-f461da89349c
DocumentName: SPDX Document created by ScanCode Toolkit
LicenseListVersion: 3.17
SPDXID: SPDXRef-DOCUMENT
DocumentComment: <text>Generated with ScanCode and provided on an "AS IS" BASIS, WITHOUT WARRANTIES
OR CONDITIONS OF ANY KIND, either express or implied. No content created from
ScanCode should be considered or used as legal advice. Consult an Attorney
for any legal advice.
ScanCode is a free software code scanning tool from nexB Inc. and others.
Visit https://github.com/nexB/scancode-toolkit/ for support and download.</text>


# Creation Info

Creator: Tool: scancode-toolkit 31.1.0
Created: 2022-09-14T06:05:00Z


# Package

PackageName: scancode_toolkit_v31_1_0
SPDXID: SPDXRef-001
PackageDownloadLocation: NOASSERTION
PackageVerificationCode: c76540cb7997ec30a05486e3de6d9676416beb67
PackageLicenseDeclared: NOASSERTION
PackageLicenseConcluded: NOASSERTION
PackageLicenseInfoFromFiles: 0BSD
PackageLicenseInfoFromFiles: AGPL-1.0-only
PackageLicenseInfoFromFiles: AGPL-3.0-only
PackageLicenseInfoFromFiles: AGPL-3.0-or-later
PackageLicenseInfoFromFiles: Apache-2.0
PackageLicenseInfoFromFiles: BSD-2-Clause
PackageLicenseInfoFromFiles: BSD-3-Clause
PackageLicenseInfoFromFiles: BSD-4-Clause
PackageLicenseInfoFromFiles: BSD-4-Clause-UC
PackageLicenseInfoFromFiles: BSD-Source-Code
PackageLicenseInfoFromFiles: CAL-1.0-Combined-Work-Exception
PackageLicenseInfoFromFiles: CC-BY-NC-ND-1.0
PackageLicenseInfoFromFiles: CC-BY-NC-ND-4.0
PackageLicenseInfoFromFiles: CC-BY-NC-SA-1.0
PackageLicenseInfoFromFiles: CC-BY-NC-SA-2.0
PackageLicenseInfoFromFiles: CC-BY-NC-SA-2.5
PackageLicenseInfoFromFiles: CC-BY-NC-SA-3.0
PackageLicenseInfoFromFiles: CC-BY-NC-SA-4.0
PackageLicenseInfoFromFiles: CC-BY-ND-1.0
PackageLicenseInfoFromFiles: CC-BY-ND-2.0
PackageLicenseInfoFromFiles: CC-BY-ND-2.5
PackageLicenseInfoFromFiles: CC-BY-ND-3.0
PackageLicenseInfoFromFiles: CC-BY-ND-4.0
PackageLicenseInfoFromFiles: CC-BY-SA-1.0
```

〔그림 12〕 ScanCode Toolkit SPDX Creation Results File

Ⅴ. 결 론

디지털 혁명이 찾아오면서 소프트웨어의 개발 수는 급격히 증가하였고, 소프트웨어 개발에서의 OSS 사용은 점점 필수가 되어가지만, 그에 따른 취약성 공격은 지속적으로 증가하고 있다. 이러한 문제로 OSS 추적성은 점점 중요시되고 이에 따른 OSS 취약성 관리 기술들이 많이 개발되고 연구 중에 있다.

그러나 아직까지 명확한 OSS 취약성 관리에 관한 정책이나 국제 표준이 존재하지 않아 기존 OSS 취약성 관리를 위해 도구를 이용하는 경우가 많다. OSS 취약성 관리 도구를 통해 OSS 사용에 따른 위험 감지는 개선되었지만 OSS 취약성 관리 도구만을 사용하기에는 많은 한계가 있었다.

이러한 문제에 대해 본 논문에서는 OSS의 한계를 설명하고 현재 전 세계적으로 각광받고 있는 SBOM의 정의와 필요한 이유에 대해서 설명한다. 이를 통해 OSS의 취약점 관리와 가시성을 높일 수 있도록 하며, SBOM의 작성 및 관리를 위한 상용화 도구 중 대표적인 몇 가지를 분석하여 데이터, 취약성 표현방법 등을 기술했다. 추후, 본 논문에서 분석한 SBOM 상용화 도구를 통해 SBOM을 실제 작성하고, SBOM을 구성하는 요소들을 비교 분석한다. 이를 통해 각 상용화 도구들이 NTIA의 SBOM 필수 구성요소나 SPDX의 SBOM 체크리스트 등 SBOM에서 필요한 요소들이 구현되어 있는지 실증을 진행할 예정이다.

참 고 문 헌

[1] Kumar, Subodha, and Rakesh R. Mallipeddi. "Impact of cybersecurity on operations and supply chain management: emerging trends and future research directions." Production and Operations Management.

[2] https://www.synopsys.com/software-integrity/resources/analyst-reports/open-source-security-risk-analysis.html

[3] https://snyk.io/reports/open-source-security/ https://owasp.org/

[4] https://nvd.nist.gov/

[5] https://www.mend.io/vulnerability-database/

[6] https://www.first.org/cvss/v3.1/specification-document

[7] https://www.synopsys.com/software-integrity/security-testing/software-composition-analysis.html

[8] https://www.mend.io/open-source-security/

[9] Executive Office of the President of U.S. (2021).Improving of Nation’s Cybersecurity (Executive Order 14028 of May 12, 2021)

[10] Framing Software Component Transparency: Establishing a Common Software Bill of Materials (SBOM) Second Edition

[11] Survey of Existing SBOM Formats and Standards - Version 2021

[12] https://vimeo.com/730359322?utm_campaign=5370367&utm_source=affiliate&utm_channel=affiliate&cjevent=5fd8e2401d2f11ed831e03b10a180512&clickid=5fd8e2401d2f11ed831e03b10a180512

[13] https://www.csoonline.com/article/3668530/sbom

-formats-spdx-and-cyclonedx-compared.html
[14] AL SAMMAN, A. B. D. U. L. L. A. H. "MODELING FLOSS DEPENDENCIES IN PRODUCTS.“
[15] https://cyclonedx.org/docs/1.2/xml
[16] https://github.com/CycloneDX/bom-examples
[17] https://github.com/fossology/fossology
[18] https://github.com/tern-tools/tern
[19] https://github.com/nexB/scancode-toolkit

〈저자소개〉

김 선 우 (Sun-Woo Kim)
2022년 8월 : 강원대학교 컴퓨터정보통신공학과 졸업
2022년 9월~현재 : 강원대학교 융합보안학과 석사과정
<관심분야> SW/HW 공급망 보안, 개인정보 비식별, SBOM

손 경 호 (Kyung-Ho Son)
종신회원
2015년 8월 : 성균관대학교 컴퓨터공학과 박사졸업
2001년 1월~2018년 8월 : 한국인터넷진흥원 팀장/단장/센터장
2018년 9월~현재 : 강원대학교 교수
<관심분야> IoT/CPS보안, 보안성 시험·인증, 개인정보 비식별&Mydata 활용