Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.32, NO.4, Aug. 2022 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2022.32.4.709

상용 오픈소스 취약점 스캐닝 도구의 성능 시험을 위한 효율적 평가 기준 개발 및 적용*

신 강 식,[1†] 정 동 재,[2] 최 민 지,[1] 조 호 묵[3‡]
[1,2,3]KAIST 사이버보안연구센터 (연구원, 선임연구원, 책임연구원)

A Study on the Development and Application of Efficient Evaluation Criteria for Performance Testing of Commercial Open Source Vulnerability Scanning Tools*

Kangsik Shin,[1†] Dong-Jae Jung,[2] Min-Ji Choe,[1] Ho-Mook Cho[3‡]
[1,2,3]KAIST Cyber Security Research Center (General, Senior, Principal Researcher)

요 약

최근 "Log4j 보안 취약점 사태" 가 발생함에 따라 오픈소스인 "Log4j"를 활용하는 정보시스템이 취약점에 노출되었다. 이번 사태로 인해 전 세계뿐만 아니라 국내 주요 정부 기관 또는 기업들의 정보시스템에 큰 취약점이 발생하여 오픈소스의 취약점에 대한 문제가 대두되었다. 오픈소스는 여러 장점에도 불구하고 오픈소스를 활용하여 개발하는 현재의 개발 패러다임으로 인해 소프트웨어 보안 취약점이 손쉽게 확산될 수 있다는 문제점이 많아 오픈소스의 안전성 및 신뢰성 확보하기 위해 오픈소스에 대한 취약점 점검이 필요하다. 하지만 오픈소스 취약점 스캐닝 도구는 종류도 많고 지원하는 언어와 기능들이 상이한 다형적인 특징을 가지고 있다. 따라서, 기존 소프트웨어 평가 기준으로는 평가하기 모호하고 장단점을 평가하기 어려우므로 본 논문에서는 오픈소스 취약점 분석 도구에 대한 새로운 평가 기준을 개발하였다.

ABSTRACT

The recent "Log4j Security Vulnerability Incident" has occurred, and the information system that uses the open source "Log4J" has been exposed to vulnerabilities. The incident brought great vulnerabilities in the information systems of South Korea's major government agencies or companies and global information systems, causing problems with open source vulnerabilities. Despite the advantages of many advantages, the current development paradigm, which is developed using open source, can easily spread software security vulnerabilities, ensuring open source safety and reliability. You need to check the open source. However, open source vulnerability scan tools have various languages and functions. Therefore, the existing software evaluation criteria are ambiguous and it is difficult to evaluate advantages and weaknesses, so this paper has developed a new evaluation criteria for the vulnerability analysis tools of open source

Keywords: opensource vulnerabilities, opensource analysis, sbom, vulnerabilities scanning tools

Received(05. 13. 2022), Modified(07. 18. 2022), Accepted(07. 20. 2022)

* 본 논문은 과학기술정보통신부 글로벌사이버보안기술연구(과제고유번호: 1711151346) 사업의 지원을 받아 수행된 연구임.

† 주저자, ksshin90@kaist.ac.kr
‡ 교신저자, chmook79@kaist.ac.kr(Corresponding author)

I. 서 론

2021년 12월 일명 “Log4j 보안 취약점 사태”가 발생함에 따라 “Log4j”를 활용하는 정부 기관 또는 기업들의 정보시스템들이 취약점에 노출되었다[1]. “Log4j”는 프로그램이 동작할 때 로그를 기록해 주는 소프트웨어로써 IT 기업뿐 아니라 웹사이트를 운영하는 非IT 기업과 정부 기관까지 “Log4j” 소프트웨어를 활용하고 있어 취약점에 얼마나 노출되어 있는지 파악하기조차 어려워 최악의 보안 결함으로 대두되었다[2]. 이미 2014년에 웹사이트에서 사용되는 “OpenSSL” 소프트웨어의 하트블리드 보안 취약점[3]도 발생하였는데, “OpenSSL”은 컴퓨터 간 안전한 통신을 하기 위해 “Log4j”와 마찬가지로 자주 활용되는 소프트웨어이다. “Log4j”와 “OpenSSL”은 모두 오픈소스라는 특징이 있는데, 누구나 참여해 프로그램 코드를 수정하고 사용할 수 있는 장점이 있어 소프트웨어 개발 영역에서 다양하게 활용되고 있다[4]. 하지만 이러한 장점으로 인해 오픈소스 소프트웨어에는 ‘치명적인 취약점’이 내재될 위험성을 가지고 있으며, 이 문제는 이미 오픈소스 보안성 강화의 필요성으로 이슈화 되었고, 우려가 커질 때 쯤 발생한 “Log4j 보안 취약점 사태”는 오픈소스 소프트웨어 생태계의 위험성에 대한 경각심을 일깨워주었다. [Fig 1]과 같이 오픈소스 보안 취약점은 매년 급속도로 증가하는 추세와 동시에 이를 활용하는 소프트웨어의 보안 취약점 또한 증가할 것으로 판단된다[5,6].

앞서 살펴본 “Log4j” 취약점 사태와 “OpenSSL” 하트블리드 취약점 사태는 현재의 개발 패러다임과 오픈소스 특징으로 인해 보안 취약점 발생 시 순식간에 위협 및 위험성이 확산되며, 빠르고 효과적인 대응이 어려운 문제점을 보여준다. 특히, 취약점이 노출된 오픈소스를 활용하여 새로운 프로그램을 제작할 경우 기존 오픈소스의 보안성에 대한 정확한 평가 기준과 위험성을 탐지하는 데 있어 상당한 어려움이 있기 때문에 문제가 더욱 심각하다.

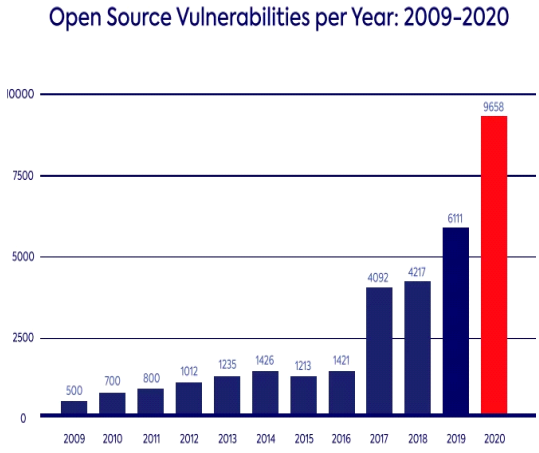


Fig. 1. Open Source Vulnerabilities per Year (WhiteSource, 2021)

따라서, 오픈소스의 취약점 분석 기능 평가하기 위해서는 위험성을 탐지할 수 있는 기준이 필요한데 그 기준을 마련하기 위해 오픈소스의 구성요소를 확인할 수 있는 SBOM(Software Bill of Materials)을 식별하고 분석하는 것이 중요하다. SBOM은 소프트웨어 구성요소의 세부 정보 및 공급, 종속, 계층적인 관계를 추적할 수 있으며, 소프트웨어를 구성하는 요소에 대한 투명성을 제공하여 취약성을 추적하고 문제가 발생할 때 문제점을 신속히 수정하는 데 필요한 정보를 담고 있다[7].

최근 “미국 사이버보안 개선에 관한 행정 명령”에 따르면, 미국 정부는 소프트웨어 서비스 업체에 미국 연방 기관에 판매하거나 제공하는 소프트웨어에 대한 SBOM 제출을 요구하였다[8]. 오픈소스 보안 취약점 문제 및 위험성으로 인해 오픈소스 취약점에 대한 정밀한 분석이 필요한 시점이며, 시중에는 이미 다양한 분석 도구가 상용화되어 활용되고 있지만, 오픈소스 취약점 분석 결과가 도구마다 매우 상이하므로 오픈소스 취약점 분석 도구를 효율적으로 평가할 수 있는 기준이 필요하며, 이를 위해 오픈소스 취약점 분석 도구의 기능 중 가장 중요한 부분은 취약점 정보와 SBOM 정보를 찾아내는 것이다. 따라서 올바른 취약점 정보와 SBOM 정보에 대한 정확성을 평가할 수 있는 기준이 필요하며, 오픈소스 취약점 분석 도구를 평가하고 장단점을 확인할 수 있는 기준을 마련해야 하는데, 이를 위해 오픈소스 취약점 분석 도구에 대한 올바른 평가 기준을 개발하고 적용할 수 있는 방안이 필요하다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구와 배경 지식을 기술하고, 3장에서는 본 논문에서 효율적인 평가 기준을 수립하고 적용에 관한 연구를 설명한 후 도출된 평가 기준으로부터 상용 오픈소스 취약점 분석 도구를 평가하고 검증한다. 마지막으로 결론을 맺는다.

II. 관련 연구 및 배경 지식

2.1 오픈소스 취약점 분석 도구

오픈소스 취약점 분석 도구는 NVD(National Vulnerability Database) 및 각종 보안 커뮤니티의 보안 권고를 기반으로 생성된 취약점 데이터베이스 정보를 통해 오픈소스에 대한 취약점 분석 및 점검 결과를 제시하는 도구이다. 해당 도구들은 취약점뿐만 아니라 오픈소스가 어떠한 구성요소로 구현되었는지 SBOM 분석을 통해 제공하며, 고유의 디지털 시그니처 또는 다양한 분석 알고리즘을 활용하여 오픈소스 소프트웨어 소스코드 내 알려진 취약점들 찾아주고 해결 방안을 제공하는 기능을 가진다.

2.1.1 CVE(Common Vulnerabilities and Exposures)

CVE는 공개적으로 알려진 소프트웨어의 취약점을 가리키는 고유 식별 정보이며, 운영체제뿐만 아니라 하드웨어, 소프트웨어 등 다양한 영역에서 알려진 취약점에 대한 우선순위를 지정하고 식별하는 방식을 표준화한 것이다. 이는 "CVE-YYYY-NNNN..N"으로 표기하며 "YYYY"에는 발견된 연도와 임의의 번호를 붙여 취약점 식별 번호를 사용한다. CVE 정보는 오픈소스 소프트웨어 도구 평가 시 알려진 CVE를 얼마나 찾을 수 있는지에 대한 정확도를 평가하기 위해 필수적인 정보로 사용된다.

2.1.2 SBOM(Software Bill of Materials)

SBOM은 하드웨어 장비 구매 시 제품을 구성하는 모든 부품에 대한 목록을 일컫는 BOM(Bills of Material)에서 파생된 것으로, 소프트웨어를 구성하는 요소의 정보 및 관계를 일목요연하게 나타내며, 여러 구성요소의 세부 정보 및 공급망 관계, 종속성 및 계층적 관계를 추적할 수 있다. SBOM의 목적은 소프트웨어를 구성하는 구성요소에 대한 투명성과 취약성을 추적하고 신속히 수정할 수 있는 정보를 제공하는 것이다[9].

판매되는 상용 소프트웨어뿐 아니라 무료로 배포되는 소프트웨어에는 오픈소스 이외에도 여러 소프트웨어로 구성 될 수 있으며, 그 오픈소스 또한 다른 소프트웨어로부터 재생성된 오픈소스일 수 있다. 예를 들어 [Fig 2]와 같이 "Acme Application v1.1" 소프트웨어에는 "Bingo Buffer v2.2"와 "Bob's Browser v2.2"로 구성되어 있는데, "Bob's Browser v2.2" 의 경우 "Carol's Compression Engine v3.1"을 구성요소 포함하여 소프트웨어를 개발한 것이다.

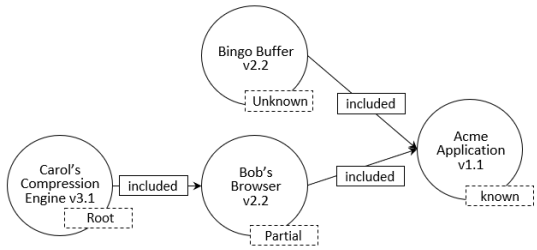


Fig. 2. Conceptual SBOM tree with upstream relationship assertions(NTIA)

또한, "Acme Application v1.1" 소프트웨어는 "Carol's Compression Engine v3.1"의 소프트웨어를 포함하고 있으므로 "Carol's Compression Engine v3.1" 소프트웨어에 취약점이 발생하면 "Acme Application v1.1" 소프트웨어에도 잠재적 위험성을 가진다고 판단할 수 있다. SBOM은 소프트웨어 구성요소와 취약점 상관관계 등의 구성을 알 수 있고 보안 패치와 수정이 필요할 때 유지보수가 편리하다는 장점이 있어 비용 절감, 보안 위험 차단 등 다양한 장점이 있다.

2.2 관련 연구

2.2.1 오픈소스 취약점 스캐닝 도구 분석

2.2.1.1 Semgrep

Semgrep는 Return To Corporation(r2c)에서 개발하였으며, 2009년 Facebook에서 개발한 프로그램 분석 라이브러리인 pfff를 기반으로 만들어졌다. 20개 이상의 언어를 지원하며 런타임 에러, 로직 버그, 보안 취약점 등을 분석하고 해결 방안을 제시해준다. 해당 룰은 1500개 이상의 오픈소스 커뮤니티에서 만들어진 정보를 기반으로 오픈소스를 스캔하고 분석한다. CLI로 로컬에서 실행할 수 있으며, Github, Jenkins와 같은 CI 연동을 지원한다[10].

2.2.1.2 VisualCodeGrepper

VisualCodeGrepper는 C/C++, C#, VB,

Java 및 PL/SQL 언어기반의 코드를 분석할 수 있는 도구이다. 검사 이외에도 Microsoft의 사용이 금지된 함수 리스트들과 언어별로 구성 파일을 통해 사용자가 원하는 취약점 정보를 스캔할 수 있는 기능이 있다. 대상 소프트웨어의 스캔이 완료되면 위험도별로 순위가 나타나며 버퍼 오버플로, XSS, SQLi 등의 보안 문제를 식별할 수 있다[11].

2.2.1.3 FlawFinder

FlawFinder는 C/C++ 소스 코드를 검사하고 가능한 보안 취약점("결함")을 위험 수준별로 분류 및 결과 보고를 통해 프로그램의 결함을 찾을 수 있다. 버퍼 오버플로 위험이 있는 함수인 strcpy(), strcat(), gets(), sprintf() 및 scanf() 같은 잘 알려진 C/C++ 함수의 내장 데이터베이스를 사용하여 동작하며, 주석과 문자열 내부를 제외하여 위험 수준을 추정하기 위해 매개 변수도 분석하여 취약점을 찾는다[12].

2.2.1.4 Bandit

Bandit는 Python으로 작성된 코드에서 알려진 취약점을 스캔하는 소스 코드 보안 분석 도구이다. AST(추상 구문 트리)를 구축하는 파일을 처리하여 취약점 정보를 스캔한다. 테스트 결과는 발견된 문제를 식별하고 문제 번호와 문제 설명을 제공한다. 또한, 식별된 문제의 심각도 및 기밀성 영향을 나타내는 위험의 심각도 및 신뢰 수준을 제공하고, 문제가 감지된 줄 번호와 코드 자체가 표시되어 결과를 출력한다[13].

2.2.1.5 Sparrow SCA

Sparrow SCA는 사용 중인 오픈소스 소프트웨어를 자동으로 식별하고 분석하여 보안 취약점과 라이선스 위반 사항을 검출한다. 오픈소스 소프트웨어의 소스 코드를 직접 사용하거나 압축파일, 혹은 바이너리 형태로 사용하는 경우 모두 분석이 가능하다. 또한, 의존성 파일 분석과 코드 스니펫 분석 지원을 통해 정확한 분석이 가능하며, 웹 기반 사용자 인터페이스로 웹 브라우저를 통해 제공하고 다양한 형태의 보고서 지원한다[14].

2.2.1.6 Labrador

Labrador는 2015년 한국, 미국 등 4개국 연구팀의 공동 연구를 통해 개발된 오픈소스 라이선스 점검 및 보안 취약점 자동 분석 플랫폼이며, 독자적인 기술인 VUDDY(A Scalable Approach For Vulnerable Code Clone Discovery) 알고리즘을 통해 Code Clone으로 발생한 함수의 취약점을 탐지할 수 있으며, CENTRIS 기술을 활용하여 수정된 오픈소스 컴포넌트 정확히 탐지할 수 있다. 또한, SBOM 시각화를 통한 소프트웨어 구성요소 분석 가능 및 SBOM 표준 문서 생성이 가능하다[15].

2.2.1.7 WhiteSource

WhiteSource는 디지털 시그니처 방식을 사용하여 소프트웨어에서 사용되고 있는 오픈소스의 구성요소를 식별하고 더욱 쉽고 효율적으로 관리할 수 있는 오픈소스 보안 취약점 및 라이선스의 규정을 점검해 주는 도구이다. WhiteSource는 세계 최초 오픈소스 보안 취약점 자동 탐지 솔루션으로 업계에서 가장 포괄적인 오픈소스 취약점 DB를 보유하고 있으며, 데이터베이스가 SaaS를 기반으로 운용되어 보안 취약점 관련 상세 정보가 실시간으로 업데이트가 가능하다[16].

2.2.1.8 Snyk

Snyk는 .Net, C/C++ 등 11개의 언어를 지원하고 있으며, 소스 코드, 오픈 소스 종속성, 컨테이너 이미지 및 코드 구성으로서의 인프라의 취약점을 테스트하고 컨텍스트, 우선순위 지정 및 수정을 제공한다. 또한, 개발 도구 및 자동화 파이프라인에 직접 통합되어 코드, 종속성, 컨테이너 및 인프라의 보안 취약점을 찾는 플랫폼으로 다양한 개발 언어를 지원하며 언어마다 여러 빌드 도구와 패키지 관리를 지원한다[17].

2.2.1.9 BlackDuck

BlackDuck은 전 세계 90% 이상의 시장 점유율에 걸맞게 신뢰성을 인정받은 OSS(Open Source Software) 솔루션으로 오픈소스가 가진 보안 취약점과 라이선스를 동시에 분석할 수 있다. Snippet

검출 알고리즘 지원하며 SBOM을 생성하고 검증하기 위해 고유한 탐지 기술을 보유하고 있다. 또한, SBOM을 통해 타사의 컴포넌트와 오픈소스의 컴포넌트 추적이 가능하다[18].

2.2.1.10 Parasoft

Parasoft은 C 및 C++에 대한 런타임 오류 감지를 시작으로 정적 코드 분석, 단위 테스트를 위한 기능도 추가했으며 애플리케이션 보안, 기능 테스트 및 서비스 가상화를 포함하도록 확장한 프로그램이다. C 및 C++ 프로그램에서 런타임 오류를 찾는 메모리 오류 감지 기술을 개발하였으며, 라이센스 관련 문제도 식별할 수 있다. 최근에는 머신러닝 기술을 도입하여 학습을 통해 더욱 빠르고 정확한 분석을 할 수 있다[19].

2.2.2 오픈소스 SW 취약점 분석 도구 관련 연구

(국내 연구 논문)을 살펴보면, "A Study on Analysis of Open Source Analysis Tools in Web Service" 연구에서는 사용자가 무료로 쉽게 웹 서비스 보안 취약점을 진단할 수 있도록 여러 오픈소스 기반의 보안 취약점 진단 도구를 개발했다. 웹 서비스의 보안 약점을 진단하는 도구의 적합성 평가 및 기능 분류가 명확하지 않아서 진단 도구를 선택하고 활용하는 데 어려움이 있으며, 알려진 취약점 항목과 소프트웨어 보안 약점 진단 가이드 등을 통해 웹 서비스 보안 취약점을 진단하는 도구에 대한 분석 기준을 제시하였다[20].

"The Software Quality Testing on the basis of the International Standard ISO/IEC 25023" 연구에서는 소프트웨어 품질 평가를 위해 국제 표준 문서 ISO/IEC 9126-2의 평가 모델과 ISO/IEC 25023의 평가 모델에 대한

차이점을 비교하고 평가 모델인 8가지 품질 특성, 즉 기능성, 신뢰성, 사용성, 유지보수성, 이식성, 효율성, 상호운영성, 보안성적인 측면에서 평가 메트릭을 제시하여 331개의 융합 소프트웨어를 분류해 테스트하고 차이점을 분석했다[21].

"Comparing Open Source Static Security Analysis Tools based on Software Weakness" 연구에서는 오픈소스로 공개된 정적분석 기반의 보안 약점 진단 도구를 대상으로 국내에서 의무화된 보안 약점에 대한 탐지 능력을 분석하는데, 먼저 진단 도구 분석 기준을 제시하였으며 진단 도구 6종에 대한 분석 결과를 설명하였다[22].

"Usability Quality Evaluation Criteria of e-Learning Software Applying the ISO Quality Evaluation System" 에서는 이러닝 소프트웨어의 품질 평가 기준을 구축하기 위해 소프트웨어 제품 평가 표준인 ISO/IEC 25000 중 사용성에 대한 정보를 바탕으로 이러닝 소프트웨어의 품질 요구사항을 분석하고 사용성에 대한 특성에 맞춰 평가 기준을 구축했다[23].

"Quality Evaluation of Criterion Construction for Open Source Software" 연구에서는 오픈소스의 사용이 증가하고 있으나, 오픈소스 소프트웨어 특성상 품질의 중요성을 고려하지 않고 오픈소스가 배포되고 있다고 지적했다. 또한, 명확한 평가 기준이 없고 미흡한 부분이 많아 실질적으로 적용하기 어려운 문제점을 개선하기 위해 해당 연구에서는 일관성 있는 정량화된 기준을 구축하여 오픈소스 소프트웨어의 품질 평가를 진행했다[24].

"A Study on security vulnerability of Open Source" 연구에서는 보안 취약점이 존재한 오픈소스의 사용으로 인해 발생한 공격 사례를 분석하고, 이러한 피해를 줄이기 위해 사용할 수 있는 보안 정적분석 도구인 FindSecurityBugs, LAPSE+, Labrador, Yasca를 소개하고 분석했다[25].

(국외 연구 논문)을 살펴보면, "Evaluation of Software Product Functional Suitability: A Case Study." 연구에서 품질 모델, 평가 프로세스 그리고 ISO/IEC 25000 국제 소프트웨어 표준에서 제안한 Functional Suitability를 평가하기 위한 품질 평가 방법을 제시했다[26].

"STUDYING OPEN SOURCE VULNERABILITY SCANNERS FOR VULNERABILITIES IN WEB APPLICATIONS " 연구는 3개의 취약성 스캐너 w3af, Skipfish 및 OWASP Zed Attack Proxy를 사용하여 OWASP 상위 10개 위협을 평가했다. 분석 보고서를 추출하여 같은 방법의 실험을 통해 수집된 웹 애플리케이션의 취약점 정보를 분석하고 각 스캐너의 실행 시간만을 비교해 도구들의 성능 평가를 실험하였다[27].

"A Systematic Mapping Review of Usability Evaluation Methods for Software Development Process" 연구에서는 평가 항목 중

사용성이 소프트웨어 품질을 평가하기 가장 중요한 측면이라고 강조하였으며, 소프트웨어의 사용성에 대한 평가 기법을 정의하기 위해 다양한 연구를 매핑하여 최적의 평가 기법을 설명했다[28].

"A Comparison of Open-Source Static Analysis Tools for Vulnerability Detection in C/C++ Code" C/C++ 코드의 잠재적인 취약점을 탐지하기 위한 맞춤형 정적분석 도구를 개발하였으며, 기존 도구와 다르게 기존 코드의 취약점 이외에도 서브 코드의 라이브러리 취약점을 탐지할 수 있다[29].

"A Review of Software Quality Models for the Evaluation of Software Products"에서는 소프트웨어의 구성요소 중요성을 강조하였으며, 소프트웨어 평가 기준에 대한 구성요소, 재사용성 등에 대한 평가 기준 모델을 비교하고 주요 평가 모델에 대한 장점과 문제점을 지적했다[30].

"Evaluation of Open-Source IDE Plugins for Detecting Security Vulnerabilities"에서는 취약점을 식별하고 보고할 수 있는 5개의 오픈소스 IDE 플러그인을 평가했다. 플러그인이 감지할 수 있는 취약점 범주의 수, 플러그인이 취약점을 얼마나 잘 감지하는지, 플러그인 출력이 개발자에게 얼마나 사용자 친화적인지 평가하고 비교했다[31].

2.2.3 연구 동향 및 한계점

[Table 1]은 10개의 오픈소스 도구들의 소개 자료, 매뉴얼, 실제 시스템 분석 바탕으로 소프트웨어 진흥법 제49조 제2항 "소프트웨어 기술성 평가 기준 지침" 평가 기준에 따른 [별표 2] 상용 소프트웨어 평가 항목 및 배점 한도(제3조 제2항 관련)에 각 분류별로 항목 개수 별로 나눠 직접 점수로 매칭 평가한 결과이다[32]. 점수 매칭 기준은 평가가 가능할 경우 2점을, 평가하기 모호한 경우 절반인 1점으로, 평가가 불가능할 경우 0점으로 평가하였다. 예를 들어 기준 지침 평가 기준 중 "Functional"에서 세부 항목으로 "구현 완전성", "보안성" 등 5개의 세부 평가 항목으로 나눠져 있는데,

세부 평가 항목에 대해 모두 평가가 가능할 경우 10점 만점이 된다. "Semgrep" 도구는 "Functional" 항목에서 "구현 완전성", "보안성" 등 5개의 세부 항목을 평가할 수 있는 기능이 존재하지 않아 0점으로 평가하였다. 오픈소스 취약점 분석 도구는 기존의 소프트웨어 기술성 평가 기준 지침에 부합되지 않는 평가 기준이 많다. [Table 1]의 전체적인 점수에 관한 결과를 보면 먼저 첫 번째 "Semgrep" 도구부터 "Bandit" 도구의 경우 오픈소스 취약점 분석 도구로서 역할을 하고 있지만, 기존 평가대로라면 기능성, 효과성, 지원성의 경우 평가할 수 항목이 없어 0점의 점수를 받으며, 특정 기능에만 평가할 수 있는 항목이 존재하기 때문에 기존 평가 기준으로는 도구마다 점수가 다양하게 분포되어있는 걸 볼 수 있다. 또한, 점수 부여가 불가능한 평가 기준이 다수 존재하여 평가 시 평가 가능 여부 대한 검증으로 인해 평가 시간을 증가시키는 문제가 있다. 결론적으로 오픈소스 취약점 분석 도구를 분석한 결과 오픈소스 소프트웨어 취약점 분석 도구는 다형적

Table 1. Existing "Software Evaluation Criteria Guidelines" Matching Evaluation Information
(* 5 : number of detailed funtionals, 10 : sum of scores)

Functions / Tools	Functional (5/10)*	Usability (6/12)	Portability (3/6)	Efficiency (3/6)	Maintainability (4/8)	Reliability (4/8)	Support (5/10)
Semgrep	0	0.83	3.33	3.33	2.5	0	0
VisualCodeGrepper	0	0.83	3.33	3.33	2.5	0	0
FlawFinder	0	0.83	3.33	3.33	2.5	0	0
Bandit	0	0.83	3.33	4.99	2.5	0	0
Sparrow SCA	4	5.81	3.33	4.99	3.75	1.25	2
Labrador	4	6.64	3.33	4.99	5	1.25	2
WhiteSource	2	5.81	3.33	4.99	5	1.25	2
Snyk	2	5.81	3.33	4.99	5	1.25	2
BlackDuck	4	6.64	3.33	4.99	5	1.25	2
Parasoft	2	6.64	3.33	4.99	5	1.25	2

인 특징으로 인해 기존 평가 방법으로는 평가하기 매우 복잡하고 모호한 부분이 많다. 이러한 모호한 문제는 국제 평가 표준인 “ISO/IEC 25010”, “ISO/IEC 25023”, “ISO/IEC 25041” 등을 기반으로 국내 “소프트웨어 기술성 평가 기준 지침” 평가 항목이 수립되었기 때문에 국내 평가 항목은 물론 국제 표준 평가 기준에도 마찬가지로 모호한 평가 방법이 나타난다.

〔Table 2〕는 기존 평가 방법과 제안하는 평가 방법에 대한 장단점을 비교한 표이다. 〔Table 1〕 의 기존 평가 방법은 오픈소스 취약점 분석 도구의 다형적인 특성이나 오픈소스 취약점 분석에 필수 요소인 SBOM을 고려하지 않아 오픈소스 취약점 분석 도구를 평가하는데 적용하기는 현실적으로 불가능하다. 따라서, 〔Table 2〕에 보는 것과 같이 오픈소스 취약점 분석 도구에 대한 특징을 분석하고 도구마다 상이한 특징을 고려하며, SBOM의 중요 정보를 활용하여 기존 평가 기준을 개선하여야 한다. 또한, 여러 오픈소스 취약점 분석 도구 간 교차 비교와 종합적으로 기능을 상세 분석하여 오픈소스 취약점 분석 도구의 특성에 맞는 효율적이고 적합한 평가 기준 을 만드는 것이 필요하다.

본 논문에서는 국내 “소프트웨어 기술성 평가 기준 지침”을 비롯한 국제 소프트웨어 평가 기준인 “ISO/IEC 25010”, “ISO/IEC 25023”, “ISO/IEC 25041”등, 주요 국내외 논문에서 오픈소스 SW의 검증 및 평가 시에 중요하게 다루는 기준을 바탕으로 〔Table 1〕와 같이 평가 항목들을 각 도구에 마다 평가 할 수 있도록 단순화 하여 각 기능들을 평가하였다. 과반수의 도구가 평가 가능한 기준과 오픈소스 취약점 분석 도구와 오픈소스의 다형적 특징들 그리고 SBOM의 주요 정보를 활용해 평가 기준을 연구 및 수립했다. 검증으로는 여러 오픈소스 취약점 분석 도구 중 4개를 선정해 개발한 평가 기준을 적용하여 테스트를 진행하였으며, 해당 평가 기준이 오픈소스 소프트웨어를 평가하는 데 적합한지 아닌지를 검증하였다.

III. 오픈소스 취약점 분석 도구에 대한 평가 기준 연구 및 검증

3.1 평가 기준 연구 및 수립

소프트웨어 진흥법 제49조 제2항 “소프트웨어 기술성 평가 기준 지침” 평가 기준에 따른 〔별표 2〕 상용 소프트웨어 평가 항목 및 배점 한도(제3조 제2항 관련)를 바탕을 기반으로 SBOM 중요 정보를 활용해 평가 기준을 연구했다〔32〕. 기존 “소프트웨어 기술성 평가 기준 지침” 평가 기준으로는 다형적 특징을 가진 오픈소스 취약점 분석 도구에 대해 평가를 하기에 상당히 모호하고 어려운 평가 기준이 많았다. 본 논문에서는 오픈소스 취약점 분석 도구들에 대한 특징을 먼저 매뉴얼 및 기능에 대한 분석을 하고 직접 오픈소스 분석 대상 샘플을 이용해 직접 사용하여 체계적으로 분석하고, 오픈소스 취약점 분석 도구를 다각도로 활용 및 사용하여 도구에 대한 평가 항목에 대한 기준을 연구하였다.

오픈소스 취약점 분석 도구는 언어 및 기능별로 다양한 도구들이 존재하였으며, 실제 사용자가 원하는 기능에 맞게 여러 도구 중 하나를 선택할 수 있도록 하기 위해 올바른 도구에 대한 평가 기준이 존재하지 않았다. 특히, 취약점 또는 오픈소스 내 컴포넌트를 정확히 탐지할 수 있는 기준이 필요하다고 판단하였다. 따라서 오픈소스 취약점 분석 도구들에 대한 평가를 위해 평가 부분, 평가 항목을 수립 및 연구에 있어 4개의 오픈소스 취약점 분석 도구와 10개의 오

Table 2. Compare the Strength and Weakness of the exist and the proposal

Classification	Common	Different Features	Strength	Weakness	Notes
Exist	Functionality, usability, maintainability	Reliability, efficiency and portability	Public Confidence	Too much and Inconsistent evaluation items	
Proposal		CVE / Component detection evaluation	Open Source Vulnerability SBOM, CVE Detection Accuracy	Results that depend on the analysis target	

픈소스를 활용하여 체계적인 평가를 진행하였다.

각 플랫폼의 실제 성능을 살펴보고, 외부적인 특징, 그 외 다양한 특성을 종합적으로 분석하여 비교하였으며, 분석한 결과 다형적 특징을 갖는 오픈소스 취약점 분석 도구에 대해 평가를 하기 위해서는 기능성, 사용성, 유지관리성 세 가지의 평가 부분과 정확성, 범용성 등 10가지의 평가 항목에 대한 기준을 수립했으며, 순서는 다음과 같다.

(Step 1) 오픈소스 취약점 분석 도구의 특징을 분석하여 분석할 오픈소스 소프트웨어를 직접 분석 해보고 및 매 평가한 결과로 [Table 1]을 도출했다.

(Step 2) 기존 [Table 1]의 평가 항목 결과를 분석하고 문제점을 확인하였으며, 오픈소스 취약점 분석 도구는 제조사가 다른 만큼 지원하는 기능과 취약점 탐지 방식도 다양해 평가 기준이 모호하며 오픈소스 소프트웨어의 특성이나 분석 도구의 기능을 구분하지 않고 있어 실제 적용하기는 현실적으로 불가능한 점을 확인하였다.

(Step 3) 오픈소스 취약점 분석 도구 기능을 정확히 평가할 수 있도록 평가 기준을 재분류하고 취약점 분석 도구에 맞는 평가 기준을 연구했다.

오픈소스 취약점 점검 도구의 목적 중 하나는 취약점에 대한 발 빠른 대처가 우선인데, "업데이트 용이성"은 제로데이 취약점이 발생하면 빠르게 패치하여 패치된 내용이 사용자에게 전달이 잘 되는지를 평가하는 항목이다. 최근 소프트웨어는 대부분 오픈소스를 사용하여 구성되며 상용 소프트웨어를 개발하기 위해서는 여러 개발자의 협업으로 생성된다. 이를 평가하기 위해 "조직 및 프로젝트 관리"항목은 협업을 위한 공간과 프로젝트의 버전 관리 같은 협업에 대한 기능을 평가할 수 있는 평가 기준을 연구하고 수립하였다.

(Step 4) 마지막으로 평가 기준에 대한 수립은 "Quality Evaluation of Criterion Construction for Open Source Software" 등과 같이 새로운 평가 항목에 기준을 수립한 논문처럼 평가항목에 대한 객관적이고 정량적으로 평가 할 수 있도록 기존 품질 모델에 대한 개선점과 불필요한 기존 항목들을 삭제하고 평가 기준을 수립하고 검증하였다.

3.2 오픈소스 분석 대상 수집 및 선정

[Table 3]와 같이 GitHub의 최상위 랭크 된 오픈소스 중 자주 사용되는 7개의 언어로부터 생성된 프로젝트에서 공정한 평가를 위해 2021년 상반기 버전의 소스 코드(tag)를 기준으로 대상을 선정하였다. 단, 2021년 상반기 소스가 없는 경우 근접한 과거의 소스를 선정하여 평가를 위한 오픈소스 항목을 선정하여 평가 및 검증했다.

Table 3. Open Source Vulnerability Analysis Target Software

OpenSource	Version	Language
HtmlUnit[33]	2.50.0	JAVA,JS,HTML
Decompress[34]	v4.2.1	JS(Node.js)
Xstream[35]	XSTREAM_1_4_17	JAVA
Shadowsock[36]	4.4.0.0	C#
scikit-learn[37]	0.24.2	Python
React[38]	v17.0.2	JS
redis[39]	6.2.3	C
mongodb[40]	r4.4.6	C++
opencv[41]	4.5.2	C++
godot[42]	3.3-stable	C

3.3 오픈소스 취약점 분석 도구 평가 검증

평가 항목에 대한 활용성을 평가하기 위해 오픈소스 취약점 분석 도구 중 4개를 직접 사용하였다.

오픈소스 취약점 탐지 및 분석에 대한 성능을 평가하기 위해 기능성, 오픈소스 취약점 분석 도구의 전반적인 UX/UI와 편의성을 평가할 수 있는 사용성, 마지막으로 조직 및 프로젝트 관리 기능을 평가할 수 있는 유지관리성으로 분류하여 평가를 진행했다. 평가에 관한 결과는 라이센스 위반 및 비방 관련 문제로 인해 오픈소스 취약점 분석 도구의 제품명 대신 A, B, C, D로 각각 표기하였으며 그 중 무료/유료버전 마다 기 능이 달라 평가할 수 없는 기준의 경우는 제외하였다.

3.3.1 기능성

기능성 평가 항목은 정확성와 범용성을 평가하기 위한 항목이다. 기능성은 오픈소스 취약점 분석 도구의 실질적인 기능이 제대로 동작하고 목표에 도달 할 수 있는 기능을 갖추었는지 평가하는 항목이다. [Table 4]는 기능성 평가 결과이며,

(기능성)의 대표적인 평가 기준은 정확한 컴포넌트 및 CVE를 탐지하는 것이 목적이다. 도구 간 검증을 통해 정확한 탐지 결과를 정답지로 분류하고 컴

Table 4. Overall Evaluation Table(Functionality)

	A	B	C	D
Good	4	2	1	8
Average	1	2	3	1
Bad	3	4	5	0

포넌트 탐지 결과와 CVE 탐지 결과에 대한 정확도를 구했다.

(범용성)의 경우 소스코드 분석시 외부 유출 방지 기능이 존재한다거나 취약점에 대한 우선순위를 관리할 수 있는 기능 등을 평가한다. 총 9개의 세부 항목 중 "D"의 8개 항목이 "우수"를 받아 가장 우수한 결과가 나왔다. 기능성 부분에 있어 "D"의 가장 큰 장점은 다른 도구에 비해 함수 단위로 취약점 분석이 가능한 점과 사용자가 정의한 특정 함수에 대해서도 취약점 분석이 가능한 점이다. 기능성에 대한 자세한 평가 항목은 Appendix A에 첨부하였다.

3.3.2 사용성

〔Table 5〕은 사용성 평가 항목에 관한 결과이며, 사용성 평가 항목 세부 항목으로는 사용성, 분석 데이터 입력 및 출력, 인터페이스, 운영환경 적합성, 사용방법 용이성이 있다.

(사용성)에는 도움말과 매뉴얼에 대한 상세 평가 정보와 매뉴얼의 구성에 대해 평가를 하고,

(분석 데이터 입력 및 출력)은 분석할 수 있는지 용량의 제한과 분석 소요시간, 다중 분석이 가능한지 등을 평가한다.

(인터페이스)에서는 UX/UI에 대한 전반적인 평가를 진행하여 사용자의 편의성을 평가하였다.

(운영환경 적합성)에서는 취약점 점검횟수에 제한이 있는지 별도로 운영되는 취약점 데이터베이스등 실제 운영시 필요한 서비스에 대한 평가를 진행한다.

(사용 방법 용이성)에는 개발 IDE(API) 제공을 지원하는지 오픈소스 분석시 별도 클라이언트 프로그램 설치가 필요한지, 다양한 운영체제와 사용 환경을 지원하는지를 평가한다. 총 21개의 항목 중 "D"의 14개 항목이 "우수"를 받아 가장 우수한 결과가 나왔다. 사용성의 자세한 평가 항목은 Appendix B에 첨부하였다.

Table 5. Overall Evaluation Table(Usability)

	A	B	C	D
Good	7	10	9	14
Average	4	1	5	4
Bad	6	6	7	3

Table 6. Overall Evaluation Table(Maintainability)

	A	B	C	D
Good	3	2	2	3
Average	0	0	2	1
Bad	2	2	3	3

3.3.3 유지관리성

유지관리성은 문제 진단 및 해결 지원과 업데이트 용이성 그리고 조직 및 프로젝트 관리에 대한 평가로 이루어진다.

(문제진단 및 해결 지원 평가 기준)에서는 Q&A 또는 FAQ를 운영하거나 오픈소스 취약점 분석 시 발생하는 오류에 대한 정보와 해결 방안을 제시하는지를 평가한다.

(업데이트 용이성)은 사용자에게 업데이트에 대한 주기 안내와 오픈소스 취약점에 대한 지속적인 업데이트 추적이 가능한지 여부를 평가한다.

(조직 및 프로젝트 관리)에서는 조직별 팀별로 작업 공간을 분리해 오픈소스 취약점 분석 도구 사용 시 조직 또는 팀마다 작업 공간을 사용하고 조직 관리를 할 수 있는지 평가하는 항목이다. 〔Table 6〕는 유지관리성 평가에 관한 결과이며 "A" 제품과 "C" 가 공동으로 3개의 "우수" 평가를 받았다. 특히 "A"는 FAQ와 Chat Bot 시스템을 통한 고객 응대가 매우 적극적이고, 조직 또는 팀별 작업 공간이 있어 유지관리성에서 좋은 평가를 보여주었다. "D"의 경우 업데이트 용이성이 부족했으나 조직 및 프로젝트 관리 부분에서는 가장 좋은 평가를 받았다. 유지관리성에 대한 자세한 평가 항목은 Appendix C 에 첨부하였다.

3.3.4 평가 결과 및 분석

기능성에서 가장 좋은 평가를 받은 것은 "우수" 8개로 "D"가 가장 좋은 평가를 받았으며, 사용성에서도 "D"가 "우수" 14개로 가장 좋은 평가를 받았다.

Table 7. Overall Evaluation Table(Total Score)

	A	B	C	D
Good	14	14	12	25
Average	5	3	10	6
Bad	11	12	15	6

유지관리성의 경우 "A"가 "우수" 3개로 "D"와 같은 평가를 받았다. [Table 7]은 평가 결과에 대한 종합 합계 평가표이다. "우수"가 가장 많은 제품은 25개로 "D"가 4개의 제품 중 가장 좋은 평가를 받았다. "A"와 "B"는 공동으로 14개의 "우수"를 받았으며, "C"가 "우수" 12개를 받아 4개의 제품 중 가장 안 좋은 평가를 받았다.

3.4 평가 기준 수립 검증 및 적용결과

오픈소스 취약점 분석 도구와 SBOM 주요 정보를 활용해 평가 기준을 개발하고 오픈소스 취약점 분석 도구에 적합한 평가 기준을 적용한 결과 [Fig 3]과 같이 기능성, 사용성, 유지관리성 세 가지의 평가 부분과 정확성 범용성 등 10가지의 평가 항목을 선정하였다.

첫 번째 평가 항목인 기능성에는 "정확성"과 "범용성"을 평가하며, "정확성"은 오픈소스 취약점 점검 도구에서 가장 중요한 오픈소스의 컴포넌트를 정확하게 분류하고 취약점을 정확하게 탐지하는지를 평가한다. "범용성"은 사용자 관점에서 유연한 사용 될 수 있도록 도와주는 기능이 포함되어 있는지 평가하는 항목이다.

두 번째 평가 항목인 사용성에는 "사용자 학습 용이성", "분석 데이터 입력 및 출력", "인터페이스", "운영환경 적합성", "설치 및 사용방법 용이성" 총 다섯 가지의 평가항목이 있다. 먼저, "사용자 학습 용이성"은 사용자 매뉴얼이 사용 순서에 맞게 제작됐는지 평가하는 항목이다. 그리고 "분석 데이터 입력 및 출력"과 "인터페이스" 는 분석에 있어 데이터 입력 및 출력과 사용자의 요구 조건에 맞게 다양한 인터페이스 조정이 가능한지 평가하는 항목이다. 마지막으로 "운영환경 적합성"과 "설치 및 사용방법 용이성" 은 오픈소스 취약점 분석 도구가 다른 환경으로 옮기는 것이 얼마나 용이한지 그리고 설치 과정의 용이성에 대해 평가하는 항목이다.

세 번째 평가 항목인 "유지관리성" 은 "문제진단 및 해결 지원", "업데이트 용이성", "조직 및 프로젝트 관리" 총 세 가지의 평가 항목이 있다. 먼저, "문제진단 및 해결 지원"은 오픈소스 취약점 분석 도구 사용 시 문제와 문의 사항 및 예외 처리에 대한 대처를 평가하는 항목이다.

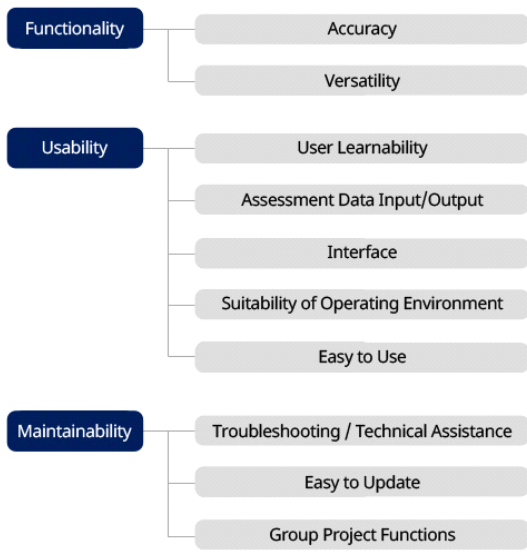


Fig. 3. Evaluation part and evaluation items

IV. 결론 및 향후 연구

이를 기반으로 오픈소스 취약점 분석 도구를 평가하기 위한 기준을 수립하고, 직접 평가를 통해 오픈소스 취약점 분석 도구에 대한 평가 기준의 유효함과 효율성을 검증하였다. 향후 연구에서는 오픈소스 취약점 분석 도구에서 정확성을 검증할 수 있는 세부 평가 항목을 추가적으로 도출하여 새로 개발되는 오픈소스 취약점 분석 도구에 대해 효율적으로 평가할 수 있는 기준을 수립하고 평가하는 연구를 진행할 예정이다.

References

[1] "Apache Log4j Security Vulnerabilities", https://logging.apache.org/log4j/2.x/security.html, Aceessed. May, 2022
[2] "Log4Shell", https://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-44228, Aceessed. May, 2022
[3] "HeartBleed", https://ko.wikipedia.org/wiki/%ED%95%98%ED%8A%B8%EB%

B8%94%EB%A6%AC%EB%93%9C, Accessed. May, 2022
[4] "Open Source Software", https://opensource.org/, Aceessed. May, 2022
[5] Synopsys, "2020 Open Source Security and Risk Analysis Report", 2020
[6] "whitesourcesoftware", "All About WhiteSource's 2021 Open Source Security Vulnerabilities Report", AYALA GOLDSTEIN, APRIL 14, 2021
[7] "SBOM", https://en.wikipedia.org/wiki/Software_bill_of_materials, Aceessed. May, 2022
[8] Lee Tae-joon, Park Chun-sik, Lee Hee-jo, "U.S. Cybersecurity Executive Order and Our Countermeasures from a Software Security Perspective", KISA Report volume 12, 2021.
[9] "CVE", https://cve.mitre.org/, Aceessed. May, 2022
[10] "Semgrep", https://semgrep.dev/, Aceessed. May, 2022
[11] "VisualCodeGrepper", https://github.com/nccgroup/VCG, Aceessed. May, 2022
[12] "FlawFinder", https://dwheeler.com/flawfinder/, Aceessed. May, 2022
[13] "Bandit", https://github.com/PyCQA/bandit, Aceessed. May, 2022
[14] "Sparrow SCA", https://www.sparrowfasoo.com/ko/product/sca, Aceessed. May, 2022
[15] "Labrador OSS", https://iotcube.com/products/labrador-oss/?lang=ko, Aceessed. May, 2022
[16] "WhiteSource Bolt", https://www.whitesourcesoftware.com/free-developer-tools/bolt/, Aceessed. May, 2022
[17] "Snyk", https://snyk.io/, Aceessed. May, 2022
[18] "BlackDuck", https://www.blackducksoftware.com/, Aceessed. May, 2022
[19] "ParaSoft", https://www.parasoft.com/, Aceessed. May, 2022
[20] Jeong-Seok Yoo, et al. "A Study on Analysis of Open Source Analysis Tools in Web Service" Proceedings of the Korea Information Processing Society Conference 21.1 pp. 475-478, 2014
[21] Hye-Jung Jung, "The Software Quality Testing on the basis of the International Standard ISO/IEC 25023", Journal of the Korea Convergence Society, Vol. 7. No. 6, pp. 35-41, 2016
[22] Jiho Bang, Rhan Ha, "Comparing Open Source Static Security Analysis Tools based on Software Weakness." Proceedings of the Korean Information Science Society Conference, pp. 753-755, 2013
[23] Ha-Yong Lee, "Usability Quality Evaluation Criteria of e-Learning Software Applying the ISO Quality Evaluation System" Journal of Digital Convergence, Vol. 16. No. 5, pp. 239-245, 2018
[24] S.-W. Kang and H.-S. Yang, "Quality Evaluation of Criterion Construction for Open Source Software," Journal of Digital Convergence, vol. 11, no. 2, pp. 323 - 330, Feb. 2013.
[25] Shin-wook Heo, Young-jin In, Chang-jun Park, Ho-won Kim, "A Study on security vulnerability of Open Source", Proceedings of Korea Computer Congress, 2016
[26] Rodríguez, Moisés, Jesús Ramón Oviedo, and Mario Piattini. "Evaluation of Software Product Functional Suitability: A Case Study.", Software Quality Professional 18.3, 2016.
[27] Deepika Sagar, Sahil Kukreja, Jwngfu Brahma, "STUDYING OPEN SOURCE VULNERABILITY SCANNERS FOR VULNERABILITIES IN WEB APPLICATIONS", IIOAB Journal 9(2) pp. 43-49 January, 2018
[28] Paz, Freddy, and José Antonio Pow-Sang. "A systematic mapping review of usability evaluation methods for softw

are development process." International Journal of Software Engineering and Its Applications 10.1, pp. 165-178, 2016.
[29] Arusoaie, Andrei, et al. "A comparison of open-source static analysis tools for vulnerability detection in c/c++ code." 2017 19th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC). IEEE, 2017.
[30] Miguel, José P., David Mauricio, and Glen Rodríguez. "A review of software quality models for the evaluation of software products." arXiv preprint arXiv:1412.2977, 2014
[31] Li, Jingyue, Sindre Beba, and Magnus Melseth Karlsen. "Evaluation of open-source IDE plugins for detecting security vulnerabilities." Proceedings of the Evaluation and Assessment on Software Engineering. pp. 200-209. 2019.
[32] "Guidelines for Evaluating the Technological Competencies of Software Business Entities", https://www.law.go.kr/LSW/admRulLsInfoP.do?admRulSeq=2100000195991, 2020
[33] "htmlunit", https://github.com/HtmlUnit/htmlunit, Aceessed. May, 2022
[34] "decompress", https://github.com/kevva/decompress, Aceessed. May, 2022
[35] "xstream", https://github.com/x-stream/xstream, Aceessed. May, 2022
[36] "shadowsocks", https://github.com/shadowsocks/shadowsocks-windows, Aceessed. May, 2022
[37] "scikit-learn", https://github.com/scikit-learn/scikit-learn, Aceessed. May, 2022
[38] "react", https://github.com/facebook/react, Aceessed. May, 2022
[39] "redis", https://github.com/redis/redis, Aceessed. May, 2022
[40] "mongo", https://github.com/mongodb/mongo, Aceessed. May, 2022
[41] "opencv", https://github.com/opencv/opencv, Aceessed. May, 2022
[42] "godot", https://github.com/godotengine/godot, Aceessed. May, 2022

A. Evaluation Area 1: Functionality

Area	Item	Criteria
Functionality	Accuracy	① Can it assess a single library or a single open source?
		② Can it assess S/W by function units?
		③ Can it assess vulnerabilities for specific functions (custom functions, etc.)?
		④ Does it provide a code-level backporting plan?
		⑤ Is there a detailed definition and explanation of which attacks (attack method, attack vector, etc.) can occur in the area where the vulnerability is found?
		⑥ Does it accurately detect components in the scanned software?
		⑦ Are there various ways to solve software vulnerabilities? (Version update, vulnerability location, patch code, etc.)
		⑧ Are the CVE detected actually in the scanned software?
	Versatility	⑨ Can it scan source code without the risk of outleakage?
		⑩ Are there priorities or other management features for vulnerabilities?
		⑪ Can the user generate a report by extracting only the desired information?

B. Evaluation Area 2: Usability

Area	Item	Criteria
Usability	User Learnability	⑫ Are help and manuals provided in various languages?
		⑬ Are help and manuals produced in the same order as the order of use?
		⑭ Does the manual include vulnerabilities of open source scanningtools and how to detect components?
	scanningData Input/Output	⑮ Are there various forms of scanningdata input?
		⑯ Can tag and version information be entered when entering scanningdata of repository such as GitHub?
		⑰ Is it possible to assess multiple sources at the same time?
		⑱ How long does it take to assess a source code?
		⑲ Does it take more than 1 second when data is loaded on the result screen?
		⑳ Is there a size limit of the file to be assessed?
	Interface	㉑ Are the software components easy to grasp at one look?
		㉒ Can the vulnerability scanninginformation be compared?
		㉓ Does it help with grasping the progress of the task?
		㉔ Does it provide SBOM report generation?
		㉕ Are there various types of reports that can be generated?
		㉖ Can the users each customize the UX/UI settings?
		㉗ Can the user leave memos in the scanningresult?
	Suitability of Operating Environment	㉘ Does it have a separate vulnerability database?
		㉙ Is there a limit to the number of open source vulnerability scans?
	Easy to Use	㉚ Does it support development tools or APIs?
		㉛ Does it require a separate client program to be installed for assessment?
		㉜ Does it support various OS and user environments?

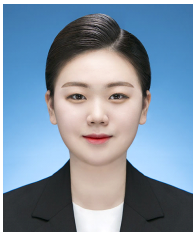
C. Evaluation Area 3: Maintainability

Area	Item	Criteria
Maintainability	Troubleshooting / Technical Assistance	㉝ Does it have Q&A or FAQ for assistance?
		㉞ When an error occurs, can the problem be identified from the error code?
		㉟ When an error occurs, does it provide the definition and solution of the error code?
	Easy to Update	㊱ Does it inform the user of the vulnerability update cycle?
		㊲ Is open source vulnerability update continuously trackable?
	Group Project Functions	㊳ Can the workspace be separated by group or team?
		㊴ Can vulnerabilities or licensing policies be customized for each group or team?

〈저자소개〉

신 강 식 (Kangsik Shin) 정회원
2016년 2월: 충남대학교 컴퓨터공학과 학사
2018년 2월: 충남대학교 컴퓨터공학과 석사
2020년~현재: 한국과학기술원 사이버보안연구센터 연구원
〈관심분야〉 악성코드 분석, 사이버보안, 시스템 보안

정 동 재 (Dong-Jae Jung) 정회원
2011년: 아주대학교 정보 및 컴퓨터공학부
2013년: 한국과학기술원 정보보호대학원 석사
2020년: 한국과학기술원 정보보호대학원 박사
2020년~현재: 한국과학기술원 사이버보안연구센터 선임연구원
〈관심분야〉 악성코드 분석, 시스템보안, 흐름 분석



최 민 지 (Min-Ji Choe) 정회원
2020년: 순천향대학교 정보보호학과 학사
2020년~현재: 한국과학기술원 사이버보안연구센터 연구원
〈관심분야〉 악성코드 분석, 시스템/사이버 보안

조 호 묵 (Ho-Mook Cho) 정회원
2006년: 아주대학교 정보통신공학과 정보보호학 (공학석사)
2018년: 전남대학교 정보보안협동과정 (이학박사)
2014년~현재: 한국과학기술원 사이버보안연구센터 책임연구원/실장
〈관심분야〉 악성코드 분석, AI 보안, 사이버보안