Journal of The Korea Institute of Information Security & Cryptology ISSN 1598-3986(Print)
VOL.33, NO.4, Aug. 2023 ISSN 2288-2715(Online)
https://doi.org/10.13089/JKIISC.2023.33.4.699

오픈소스 취약점 점검 도구 및 종합 보안 메트릭 설계를 통한 DevSecOps 구축방안 연구*

최 영 해,[1†] 노 형 준,[2] 조 성 윤,[3] 강 한 승,[4] 김 동 완,[5] 박 수 현,[6] 조 민 재,[7] 이 주 형[8‡]
[1]상명대학교 (학생), [2]소속 없음, [3]순천향대학교 (학생), [4]한양대학교 (학생),
[5]벤디트 (대표기술책임자), [6]테이텀 (개발이사), [7]레몬트리 (정보보안책임자), [8]공군 (대위)

A Study on the Development of DevSecOps through the Combination of Open Source Vulnerability Scanning Tools and the Design of Security Metrics*

Yeonghae Choi,[1†] Hyeongjun Noh,[2] Seongyun Cho,[3] Hanseong Kang,[4]
Dongwan Kim,[5] Suhyun Park,[6] Minjae Cho,[7] Juhyung Lee[8‡]
[1]Sangmyung University (Undergraduate Student), [2]No Affiliation, [3]Soonchunhyang University (Undergraduate Student), [4]Hanyang University (Undergraduate Student),
[5]Vendit (CTO), [6]Tatum (Engineering Director), [7]Lemon Tree (CISO), [8]ROKAF (Captain)

요 약

DevSecOps는 DevOps의 짧은 개발과 운용주기에 대응하기 위해 DevOps의 운용절차에 보안절차를 추가한 개념이다. DevSecOps 구축 시 빠른 개발 및 배포 주기를 지원하면서 안정적으로 보안성을 제공하기 위해 여러 단계의 취약점 점검 절차가 고려되어야 한다. 각 점검 단계별 활용 가능한 여러 오픈소스 취약점 점검 도구들이 존재하고 있으나 도구들이 지원하는 기능이 다양하고 점검결과들이 상이해 통합 운용 시 보안성 수준 평가 및 정보의 중요도 파악에 어려움이 있다. 본 논문은 오픈소스를 활용한 DevSecOps 구축 시에 보안점검 단계별 활용 가능한 오픈소스 취약점 점검 도구의 조합과 점검결과에 대한 통합적인 보안메트릭 설계방안을 제안한다.

ABSTRACT

DevSecOps is a concept that adds security procedures to the operational procedures of DevOps to respond to the short development and operation cycle. Multi-step vulnerability scanning process should be considered to provide reliable security while supporting rapid development and deployment cycle in DevSecOps. Many open-source vulnerability scanning tools available can be used for each stage of scanning, but there are difficulties in evaluating the security level and identifying the importance of information in integrated operation due to the various functions supported by the tools and different security results. This paper proposes an integrated security metric design plan for scurity results and the combination of open-source scanning tools that can be used in security stage when building the open-source based DevSecOps system.

Keywords: DevSecOps, open-source, security metric, vulnerability scanning tools

Received(02. 10. 2023), Modified(05. 09. 2023), Accepted(06. 13. 2023)

* 본 논문은 한국정보기술연구원의 차세대 보안리더 양성 프로그램(BOB)의 지원을 받아 수행한 연구임

† 주저자, ada_lovelace@kakao.com
‡ 교신저자, dokju@ajou.ac.kr(Corresponding author)

I. 서 론

최근 소프트웨어 산업계는 개발(Development)과 운영(Operations)의 합성어인 DevOps라는 개념을 적용한 개발 방법을 통해 고수준의 소프트웨어를 빠르게 배포하여 소프트웨어 lifecycle에 소모되는 비용을 줄이고 있다. DevOps는 개발과 운영의 통합으로, 특히 무중단 배포를 요구 하는 환경과 클라우드 컴퓨팅 환경에서 효과가 배가된다.[1]

그러나 DevOps가 도입된 후 빠른 속도와 높은 품질을 위해 자동화된 테스트와 배포 프로세스를 사용하면서, 보안 취약점이 있는 코드가 프로덕션 환경으로 바로 배포될 가능성이 높아져 보안 이슈의 주기도 짧아졌다. 그로 인해 보안 관련 이슈의 식별에 추가적인 시간과 비용이 필요하게 되어 개발 속도가 느려질 수 있다는 가능성이 제기되었다.

DevSecOps는 이러한 DevOps에 보안(security)을 추가한 방식으로 DevOps에서 짧아진 개발과 운용주기로 발생할 수 있는 보안 이슈에 대응할 수 있 도록 개발 및 배포 과정에 보안 절차를 추가한 개념이다. DevSecOps를 통해서 안정적인 보안을 유지함과 동시에 기존의 DevOps가 가지는 장점인 빠른 개발과 배포 주기를 보강하는 시스템 구축이 가능하다[2].

DevSecOps 구축의 핵심에는 보안점검의 자동화를 통한 프로젝트의 취약점 의존성 식별이 있다. 최근 프로젝트의 생산성을 높이기 위해 라이브러리 및 오픈소스 코드에 대한 의존성이 지속적으로 높아지고 있으나 각 의존 요소별 공개된 최약점의 수가 너무 많고 코드에 대한 개선주기가 점차 빨라지면서 새로운 보안 문제에 대해 수동적으로 최신화를 수행하는 것이 현실적으로 불가능해졌다. 따라서, DevSecOps 구축 시에는 지속 통합(CI, continuous integration) 및 지속 배포(CD, continuous deployment) 과정에서 보안점검을 위한 여러 자동화 도구를 고려해 모든 릴리스, 배포 과정의 보안 검사를 자동으로 수행하면서 개발자와 보안 관리자 모두에게 적절한 피드백이 실시간으로 이루어지게 해야 한다[3].

현재 코드 내 존재하는 취약점을 식별하기 위한 많은 수의 오픈소스 취약점 점검 도구가 존재하나 각 도구의 점검 범위와 기능 및 출력 결과가 상이해[4] 통합 운용시 보안성 수준의 평가와 정보의 중요도 파악에 어려움이 있다. 따라서 본 논문에서는 오픈소스 기반의 취약점 도구들에 대한 보안 단계별 구분과 점검결과를 종합할 수 있는 보안메트릭 설계를 통해 효율적으로 DevSecOps를 구축할 수 있는 방안을 제시하고자 한다. 제안하는 DevSecOps 시스템은 보안점검을 위한 Security 단계 및 도구 선정부터 최종 보안 스코어링에 대한 결과 도출까지의 전 과정을 포함하며 추후 실무에 적용하기 위한 구체적인 구현 요소 예시를 제안하였다. 따라서, 다양한 방식으로 구현될 수 있는 DevSecOps 프로젝트에 대한 실효성 있는 일반화된 프레임워크를 제안하는 기여도를 지닌다.

본 논문의 구성은 다음과 같다. 2장에서 관련 연구와 배경 지식을 기술하고, 3장에서는 오픈소스 취약점 점검 도구를 기반으로 한 DevSecOps 아키텍처와 점검 결과를 종합한 보안 메트릭을 설계하고, 제안한 모델을 토대로 구현한 DevSecOps에서 실제 프로젝트를 구동한다. 마지막으로 결론을 맺는다.

II. 관련 연구 및 배경 지식

2.1 오픈소스 취약점 점검 도구

오픈소스 취약점 점검 도구는 NVD(National Vulnerability Database) 및 각종 보안 커뮤니티의 보안 권고 등을 기반으로 생성된 취약점 데이터베이스 정보를 통해 소스코드 및 바이너리에 대한 취약점을 분석하고 점검 결과를 출력하는 도구이다. 해당 도구들은 고유의 디지털 서명과 여러 가지 분석 알고리즘을 활용하여 취약점이 발생한 위치와 발견된 취약점의 종류 및 관련 보안 스코어를 제공하며 해결방안을 권고한다.

2.1.1 ACR(Automatic Code Review)

코드에서 알려진 보안 이슈 및 취약점을 점검하는 단계로 주로 통합 개발 환경이나 빌드 도구에서 플러그인 방식으로 동작한다. PMD, DevSKim, FindSecBugs 등의 오픈소스가 있으며 자동적으로 사전에 설정된 보안 정책과 비교하여 위반 요소를 감지하고 보고한다.

2.1.2 SIS(Sensitive Information Scan)

코드 내에 존재하는 민감 정보를 탐지하는 단계로

주로 하드 코딩된 패스워드, 토큰 및 비밀 키 등을 감지한다. gitLeaks, Pre-commit, Trufflehog, Whispers 등 여러 오픈소스 점검 도구가 존재하며 시스템의 버전 관리 체계에 코드가 push될 때마다 자동적으로 수행되어야 한다.

2.1.3 SAST(Static Application Security Testing)

프로그램을 실제로 실행하지 않고 소프트웨어를 분석하는 단계로 오염 분석이나 데이터 흐름 분석 등을 통해 소스 코드상에서 취약한 함수나 오류, 버그 등을 탐지한다. CodeqQL, SonarQube, Bandit, Flawfinder 등의 오픈소스 도구가 존재하며 각각 지원하는 프로그래밍 언어와 플랫폼이 달라 프로젝트 성격에 맞는 적합한 도구 선택이 필요하다. 취약점에 대한 점검 범위가 넓으나 오탐률과 미탐률이 높다.

2.1.4 DAST(Dynamic Application Security Testing)

실제 또는 가상 프로세서에 프로그램을 실행하여 입력 값, 결과 값, 환경 정보 등 다양한 테스트 케이스를 통해 코드 커버리지와 크래시 발생 여부를 분석하는 단계이다. OWASP ZAP, Arachini, Nikto 등의 오픈소스가 존재하며 도구 선택 시 지원하는 프로그래밍 언어와 빌드 방법 등을 고려해야 한다. 정적분석보다 점검 범위가 좁으나 정확도가 높다.

2.1.5 SCA(Software Composition Analysis)

소프트웨어의 구성요소 중 사용중인 패키지, 라이브러리 및 오픈소스 등에 대한 의존성 점검을 수행한다. 오픈소스의 구성요소가 널리 사용된 것일수록 공격자에 의해 취약점이 발견되어 악용될 수 있는 가능성이 크기 때문에 높은 가중치가 적용된다. 패키지 관리, 소프트웨어 배포 전 라이센스 정보, 버전 정보 및 패치 상태를 확인하여 취약점에 대한 보안 이슈를 점검한다. OWASP Dependency-check, Dependabot, FOSSA 등의 오픈소스가 존재한다.

2.1.6 CaC(Compliance as Code)

개발된 소프트웨어가 사전에 정의된 보안정책을 준수하는지 점검하는 단계이다. DevSecOps 파이프라인에서 릴리즈 마다 프로젝트에 지정된 보안정책의 준수 여부를 점검하여 공격 표면을 관리함과 동시에 규정에 대한 적합성을 보장할 수 있다. Inspec, ServerSpec, OpenSCAP 등의 오픈소스가 존재한다.

2.2 취약점 점검 결과

2.2.1 CWE(Common Weakness Enumeration)

CWE는 공개적으로 알려진 소프트웨어의 보안약점을 의미하며 소프트웨어 취약점으로 이어질 수 있는 오류 목록을 말한다. MITRE에서 소프트웨어 취약점을 정의하기 위해 사전 분류 및 범주로 CWE를 도입해 분류하였다. “CWE-NNN”으로 표기하며 “NNN”에 식별번호를 붙여 구분한다. 대부분의 오픈소스 취약점 점검 도구의 점검 결과로 CWE가 제공된다[5].

2.2.2 CVE(Common Vulnerabilities and Exposures)

CVE는 공개적으로 알려진 소프트웨어 취약점을 가리키는 고유 식별 정보이며 “CVE-YYYY-N..N”으로 표기하며 “YYYY”에 발견된 연도가 표기되며 “N...N”에 식별번호가 붙게 된다. 실제 공격자의 exploit을 통한 악용 가능성이 있기 때문에 중요도가 더 높으며 오픈소스 취약점 점검 도구별 CVE에 대한 제공 가능한 정보가 상이하다[6].

2.2.3 CVSS(Common Vulnerability Scoring System)

비영리 기관 FIRST(Forum of Incident Response and Security Teams)에 의해 관리되고 있는 소프트웨어 및 하드웨어에 대한 취약점 수치 점수 산출 방법으로 CVE가 발견되면 NIST는 NVD를 통해 CVSS 점수를 산출하여 제공한다. 취약점의 위험도를 측정하기 위해 구성요소를 세 가지 영역으로 나누고 각 요소별 점수 채점을 통해 최종 점수를 산출한다. 많은 오픈소스 취약점 점검 도구들이 발견된 취약점에 대해 NIST에서 제공하는 CVSS를 제공하거나 자체 계산방법을 통한 CVSS를 제공하고 있다[7].

III. 오픈소스 기반 DevSecOps 구축 설계 및 구현

3.1 오픈소스 취약점 점검 도구 선정

DevSecOps의 보안점검 절차는 점검 범위, 성격 및 점검 시점에 따라 크게 여섯 개의 단계로 구분된다. 각 단계에는 여러 오픈소스 취약점 점검 도구가 존재하며 각 도구별 지원하는 프로그래밍 언어와 출력 결과는 Table 1과 같이 상이하다. 따라서, 우선적으로 고려할 사항은 취약점 점검 도구의 입력과 출력을 파악하는 것이다. 입력에 대한 고려사항으로는 프로젝트 종류 및 성격과 프로젝트에서 사용되는 개발언어 등이 있으며 출력에 대한 고려사항으로는 점검 도구가 제공하는 점검 결과 기준 및 측정 정확도, 결과 보고서의 포맷 등이 있다.

Table 1. Output Result of Open Source Scanner Tool

Tool	CWE	CVE	CVSS
gitleaks	X	X	X
gitguardian	X	X	X
CodeQL	O	X	X
OWASP ZAP	O	X	X
Dependency Check	O	O	O

3.2 DevSecOps의 보안 아키텍처 설계

DevSecOps 구축 시 핵심적으로 고려할 사항은 Security 단계를 DevOps에 적절히 추가하여 신속한 빌드와 배포를 지원함과 동시에 효율적이면서 효과적인 보안점검을 수행하는 것이다. 보안점검을 빌드와 배포과정에 종속할 경우 프로젝트의 규모와 사용되는 도구의 개수에 따라 점검을 위한 상당한 시간이 소요될 수 있으며 이는 DevOps의 신속성을 저해하는 결과로 이어진다. 따라서, Security 단계의 효율성을 높이기 위해 Fig. 1.과 같이 파이프라인 단위로 독립적인 보안점검 절차를 설정하는 기능을 구현하고 점검 단계별 사용되는 도구에 대한 그룹화와 발견된 이슈를 관리자 등의 사용자에게 신속하게 전파하는 기능이 구현되어야 한다. 또한, 개발 항목과 기능요소별 우선순위, 신속한 배포의 필요성 및 보안점검 결과 등을 모두 고려한 보안정책을 실시간으로

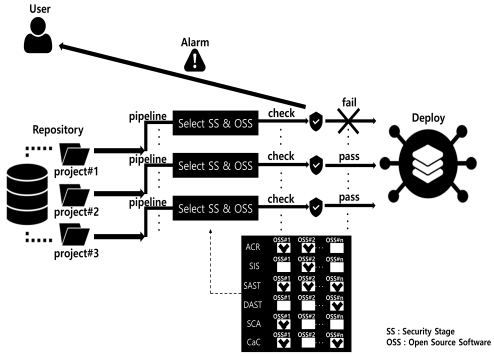

Fig. 1. Security Architecture of DevSecOps

수정하여 적용할 수 있는 가용성을 갖춰야 한다.

3.3 종합 보안 스코어링 설계

DevSecOps는 발생한 보안 이슈를 분석하여 그래픽 인터페이스를 이용한 통계기능과 보고서 작성기능 및 Quality Gate를 계산하여 임계 기준을 통과하지 못한 파이프라인에 대한 롤백 여부[8]와 개선사항 도출 등 CI/CD 전반에 대한 보안 수준 평가 기능이 탑재되어야 한다. 따라서 본 연구에서는 DevSecOps의 종합 보안 평가를 위한 스코어링 계산방법을 제안하고자 한다.

현재 대부분의 오픈소스 취약점 점검 도구들이 점검 결과로 발생한 취약점에 해당하는 CWE 정보를 제공하고 있으며 일부 도구는 CVE 정보를 포함하여 제공 중이다. 미국방성 산하의 MITRE는 CWE와 관련된 CVE 목록을 같이 제공하고 있고 각 CVE에는 CVSS가 할당되어있기 때문에 CWE를 통한 CVSS 유추가 가능하다. CVSS는 0점부터 10점까지 범위를 가지며 점수가 높을수록 위험한 취약점으로 분류되지만 점수가 낮더라도 통계적으로 많이 발생한 취약점일수록 공격자의 악용 가능성이 커지기 때문에 발생 빈도수가 함께 고려되어야 한다[9-11]. 따라서 NVD의 데이터베이스를 기반으로 통계에 기반한 빈도 점수를 정규화하여 추가로 합산하였으며 관련 정보는 Table 2과 같다.

발생한 보안 이슈 i에 대한 종합 스코어(S_i)는 표

Table 2. Vulnerability Result Evaluation

Value	5	4	3	2	1
CVSS	≥9.0	≥7.0	≥4.0	≥0.1	≥0
Freq.	≥2000	≥1500	≥1000	≥500	≥0

Table 3. Security Evaluation of Issue i

Value	5	4	3	2	1
Level	Critical	Major	Minor	Info	None
S_i	≥8.0	≥6.0	≥4.0	≥2.0	≥0.0

준화된 CVSS 점수(C_i)와 빈도 점수(F_i) 및 오픈 소스 점검 도구의 정확도(P_i)를 모두 고려하여 다음과 같이 계산된다.

$$S_i = (C_i \times P_i \times W_c + F_i \times W_f) \times W_s \quad (1)$$

여기서 W_c, W_f, W_s는 각각 CVSS 점수, 빈도 점수, 보안 단계에 대한 가중치를 의미하며 0~2 사이의 실수값을 가진다. 하이퍼 파라미터로서 사용자가 조정 가능한 값이다. 보안 이슈 i에 대한 보안 Level 및 정규화 점수는 Table 3과 같이 정의한다.

파이프라인 j에 대한 보안 등급(G_j)은 발생한 보안 이슈를 종합하여 0~1 사이의 값을 가지는 하이퍼 파라미터인 임계 비율($\Im$)과 보안 Level에 대한 정규화 점수를 토대로 다음과 같이 판정된다.

$$G_j = \begin{cases} \Im_e \le \Sigma_{Critical} S / \Sigma S & : E \\ \Im_d \le \Sigma_{Major} S / \Sigma S & : D \\ \Im_c \le \Sigma_{Minor} S / \Sigma S & : C \\ \Im_b \le \Sigma_{Info} S / \Sigma S & : B \\ Otherwise & : A \end{cases} \quad (2)$$

최종적으로 Quality Gate(QG)는 보안 점검 Pass를 위한 하이퍼 파라미터인 임계 비율($\Im_p$)을 기준으로 다음과 같이 계산된다.

$$QG = \begin{cases} \Im_p \ge \Sigma_{Critical+Major} S / \Sigma S & : Pass \\ Otherwise & : Fail \end{cases} \quad (3)$$

3.4 제안 아키텍처 기반의 DevSecOps 구현

제안한 아키텍처를 기반으로 DevSecOps를 Fig. 2.와 같이 구현하였다. 보안 관리자를 사용자로 가정하였고, 사용자는 React JS로 구현한 웹 페이지에 접속하여 빌드, 보안점검 및 소프트웨어 배포를 수행하고 감시할 수 있다. 빌드부터 배포까지의 각 단계별 수행 결과는 웹페이지에서 시각적으로 제

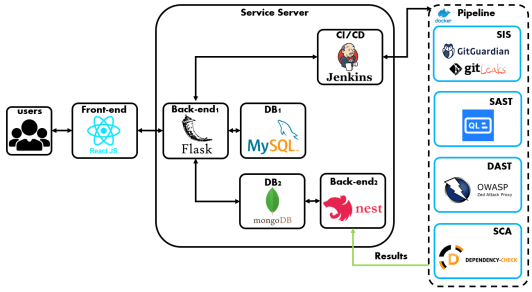

Fig. 2. Software Configuration of Developed DevSecOps

공된다. 프론트엔드와 백엔드 간 통신은 Flask API 서버가 담당하고 있으며 CI/CD 플랫폼인 Jenkins와 데이터베이스 MySQL 및 MongoDB와 연동된다. CI/CD 내의 각 파이프라인은 설정된 보안정책에 따라 보안점검이 수행된다. 점검 결과는 json 형태로 출력되며 해당 내용은 nestJS API 서버를 통해 mongoDB 데이터베이스에 저장된다. Flask API 서버는 mongoDB 데이터베이스의 내용을 토대로 발생한 보안 이슈의 목록과 세부 정보 및 종합 보안 스코어를 계산하여 React JS 프론트엔드와 통신하며 웹페이지에서는 해당 정보를 적절한 형태로 가공하여 사용자에게 제공한다. MySQL 데이터베이스는 CI/CD 설정, 계정정보, 보안정책 등 솔루션 전반에 걸친 구성 정보를 관리하기 위해 사용된다. 소스코드에 대한 레포지토리는 깃허브를 선택하였으며 Jenkins와 API와 연동되어 파이프라인 생성, 삭제 및 배포가 가능하도록 구현하였다.

취약점 점검을 위한 보안 단계 선택을 위해 최근 통합 개발 환경에 내장되는 ACR과 회사별 사내 보안정책에 의존하는 CaC를 제외한 SIS, SAST, DAST, SCA를 고려하였다. 프로젝트에 사용되는 프로그래밍 언어는 Python 및 JavaScript를 가정하였다. SIS 점검 도구로는 깃허브와 연동 가능한 GitGuardian과 gitLeaks를 선택했다. SAST 점검 도구로는 쿼리 기반의 보안점검이 가능하고 상세한 점검 결과를 제공하는 CodeQL을 사용하였으며 DAST와 SCA에는 가장 활발한 OWASP(Open Web Application Security Project) 프로젝트들 중 하나인 ZAP과 Dependency Check를 선택했다. 보안점검 도구들은 각각 도커 컨테이너로서 독립적으로 동작한다. 점검결과를 종합하여 계산 시 SIS의 점검결과 정확도를 1로 산정했으며 나머지 도구 들은 각자 점검결과에서 제공하는 수치를 사용

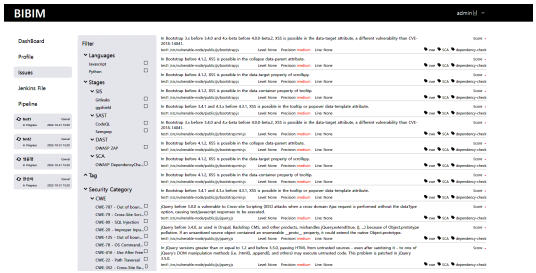

Fig. 3. Security Architecture of DevSecOps

하였다.

사용자는 점검결과 발생한 개별 보안 이슈 목록을 Fig. 3과 같이 확인할 수 있다. 개별 보안 이슈들은 발생한 취약점에 대한 설명과 함께 CVSS 스코어, 이슈가 발생한 코드 내 위치, 점검 정확도, 취약점 유형 등의 정보를 제공한다. 이때, 필터 기능을 통해 파이프라인과 점검 보안 단계, 점검 도구 및 CWE 목록 등을 선택하여 관련 보안 이슈들을 세부적으로 확인할 수 있다.

Fig. 4.의 Dashboard 페이지는 현재 개발 중인 프로젝트와 파이프라인에 대한 전체적인 보안 정보를 제공한다. Dashboard 페이지에서는 전체 프로젝트에서 발생한 보안 이슈들의 통계와 보안 등급 및 기타 통계 그래프가 시각적으로 제공된다. 이 페이지에서 사용자 구분과 필요에 따른 맞춤형 정보가 제공된다.

Table 4는 깃허브에 공개된 오픈소스 중 NodeJS로 개발된 취약한 프로젝트인 DVNA(MIT license, 587 stars)[12]를 본 논문에서 개발한 DevSecOps를 통해 보안점검을 수행한 결과이다. DVNA는 OWASP Top 10 취약점을 포함한 다양

Fig. 4. Security Dashboard

Table 4. Security Check Results of DVNA

Level	Critical	Major	Minor	Info	None
Count	5	86	109	12	0
$\Im$	$\Im_p$	$\Im_e$	$\Im_d$	$\Im_c$	$\Im_b$
Value	0.5	0.2	0.2	0.2	0.2
Grade	D		QG	Fail	

Table 5. Security Check Results of DVNA

OWASP TOP 10 Vulnerability	Count
Injection (SQL, NoSQL, OS)	4
Broken Authentication and Session Management	26
Cross-Site Scripting (XSS)	5
Broken Access Control	16
Security Misconfiguration	12
Insecure Cryptographic Storage	2
Insufficient Transport Layer Protection	14
Insufficient Authentication and Authorization	1
Using Components with Known Vulnerabilities	14
Insufficient Logging and Monitoring	12

한 보안 취약점을 가지고 있으며, 다양한 취약점을 시연하는 예제로 사용되고는 한다. Table 5는 OWASP Top10의 취약점 검출 개수이다. 스코어링의 임계 비율을 $\Im_e = \Im_d = \Im_c = \Im_b = 0.2$로 지정한 결과 해당 프로젝트는 등급 D를 얻었다. 또한, $\Im_p = 0.5$,로 지정했기 때문에 QG를 통과하지 못해 소프트웨어 배포 단계에서 Fail이 되었다. 취약점 점검 결과 해당 프로젝트는 발견된 취약점에 대한 수정조치가 즉시 수행되어야 한다.

IV. 결 론

본 논문에서는 점검 도구의 단계별 구분과 취약점 점검 결과에 대한 종합적인 보안 스코어링을 기반으로 오픈소스 취약점 점검 도구 기반의 DevSecOps 아키텍처를 제안하였다. 또한, 설계한 아키텍처를 기반으로 웹 기반의 DevSecOps 시스템을 구현해 DevOps와 보안요소를 효과적으로 연동하기 위한

세부적인 구현방안을 제시하였다. 본 논문에서 제안한 아키텍처와 구현예시를 통해 DevSecOps 구축자의 노력을 경감시킬 수 있기를 기대한다.

References

[1] In-seok Jeon, "Integrated management of DevOps in a non-disruptive environment considering security," *Journal of The Korea Institute of Information Security & Cryptology*, 25(1), pp. 47-52, Feb, 2015

[2] Jin-Keun Hong, "Component Analysis of DevOps and DevSecOps," *Journal of The Korea Convergence Society*, 10(9), pp. 47-53, Aug. 2019

[3] Moon-Hwan Kim and Kyung-Won Oh, "DevSecOps for Military Robot SW Development," *Journal of the KNST*, 5(2), pp. 122-127, Sep. 2022

[4] Kang-Sik Shin, Dong-Jae Jung, Min-Ji Choe, and Ho-Mook Cho, "A Study on the Development and Application of Efficient Evaluation Criteria for Performance Testing of Commercial Open Source Vulnerability Scanning Tools," *Journal of The Korea Institute of Information Security & Cryptology*, 32(4), pp. 709-722, Aug. 2022

[5] "CWE", https://cwe.mitre.org/, Accessed. Jul, 2023

[6] "CVE", https://cve.mitre.org/, Accessed. Jul, 2023

[7] "CVSS", https://nvd.nist.gov/vuln-metrics/, Accessed. Jul, 2023

[8] Giovanni Bernardo, "DevSecOps pipelines improvement: new tools, false positive management, quality gates and rollback," Computer Engineering Department, Politecnico di Torino, Webthesis Libraries, 2022.

[9] Kyung-Sung, "Web-Based Information Security Leveling Tool," *Korea Society of Computer Information(KSCI)*, 10(4), pp. 375-384, Sep. 2005

[10] Joon-Seon Ahn, Ji-Ho Bang and Eun-young Lee, "Quantitative Scoring Criteria on the Importance of Software Weaknesses," *Journal of The Korea Institute of Information Security & Cryptology*, 22(6), pp. 1407-1417, Dec. 2012

[11] Dong-Su Seo, "Definition of Security Metrics for Software Security-enhanced Development," *Journal of Internet Computing and Services(JICS)* 17(4), pp. 79-86, Aug. 2016

[12] "DVNA", https://github.com/appsecco/dvna/, Accessed. Jul, 2023

〈저자소개〉

최 영 해 (Yeonghae Choi) 학생회원
2017년 3월~현재: 상명대학교 정보보안공학과
〈관심분야〉 DevSecOps, 클라우드 보안, 웹 프로그래밍

노 형 준 (Hyeongjun Noh) 학생회원
2023년 2월: 세종대학교 정보보호학과 학사
〈관심분야〉 정보보호, 웹 프로그래밍

조 성 윤 (Seongyoon Cho) 학생회원
2018년 3월~현재: 순천향대학교 정보보호학과
〈관심분야〉 DevSecOps, 시스템 보안, AI 보안

강 한 승 (HanSeoung Kang) 학생회원
2020년 3월~현재: 한양대학교 물리학과
〈관심분야〉 양자보안, 인공지능

김 동 완 (Dongwan Kim) 정회원
2019년~2022년: 크래프톤 안티치트 엔지니어
2022년~현재: 벤디트 CTO
〈관심분야〉 정보보호, 악성코드 분석, 사이버 보안, 시스템 보안

박 수 현 (Suhyun Park) 종신회원
2003년 3월: University of Canterbury 전기전자공학 학사
2006년 12월: University of Canterbury 전기전자공학 석사
2009년~2020년: ㈜안랩 수석연구원
2021년~현재: 주식회사 데이텀 개발이사
〈관심분야〉 네트워크 보안, IoT 보안, 클라우드 보안

조 민 재 (Minjae Cho) 정회원
1998년 2월: 중앙대학교 컴퓨터공학과 학사
2000년 2월: 중앙대학교 컴퓨터공학과 석사
2000년 3월~2021년: 아톤 CISO, 야후코리아/네이버,쿠팡/우아한형제들 보안관리자
2022년~현재: 레몬트리 CISO
〈관심분야〉 클라우드 보안, DevSecOps, 개발 보안, 사이버 보안, 인프라 보안

이 주 형 (Juhyung Lee) 정회원
2019년 2월: 아주대학교 국방디지털융합학과 학사
2020년 3월~현재: 아주대학교 국방디지털융합학과 석박사 통합과정
2019년 6월~2021년 12월: 공군작전정보통신단 체계개발실
2021년 12월~현재: 사이버작전사령부
〈관심분야〉 클라우드 보안, 인공지능, 무인시스템