코드 기반 취약점 진단 도구를 사용한 DevSecOps 파이프라인 구현

김두영, 이해린, 장윤희, 전수연, 이광재*

*상명대학교

{201921235, 201921266, 201921268, 202021278}@sangmyung.kr, *begleam@smu.ac.kr

An Implementation of a DevSecOps Pipeline using a Codebase Vulnerability Testing Tool

Du-Yeong Kim, Yun-Hee Jang, Hae-Rin Lee, Su-Yeon Jeon, Kwangjae Lee*

*Sangmyung Univ.

요 약

본 논문은 DevSecOps의 파이프라인의 보안성을 높이기 위해서 기존의 프로토타입 기반 취약점 진단 방법에 추가로 코드 기반 취약점 진단 방법을 추가하는 방법을 제안한다. 또한 개발자가 개발단계에서부터 보안을 적용하기 위해서 시큐어코딩 가이드라인을 제공함을 제안한다. 개발자가 깃허브에 개발 소스 코드를 Push하면, 코드 기반 취약점 진단 도구인 CodeQL로 진단하고 시큐어코딩 가이드라인 보고서를 개발자에게 전달한다. 그리고 개발이 완료되어 배포한 경우에는 프로토타입 기반 취약점 진단 도구인 OWASP ZAP으로 추가 진단한다. 성능평가를 위해 취약점이 존재하는 중고거래 웹사이트를 구현하여 취약점 진단 범위와 수행시간을 측정하였고, 진단 시간이 늘었지만 기존 진단 도구에서 놓친 SSRF를 찾아낼 수 있었다. 추가로, 코드 기반 취약점을 분석한 가이드라인 보고서로 개발자가 올바른 시큐어코딩을 적용할 수 있도록 유도하였다.

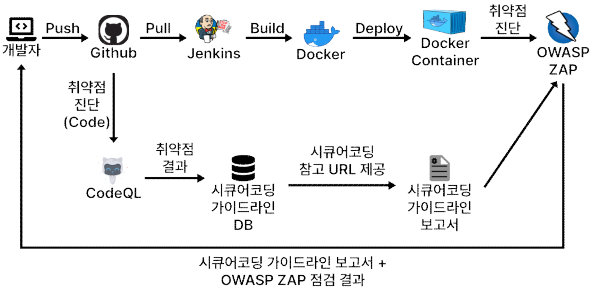
Ⅰ. 서 론

DevSecOps는 DevOps의 개발(Development), 운영(Operations)에 보안(Security)이 결합된 개념으로 개발 초기 단계부터 DevOps가 추구하는 애플리케이션의 빠른 배포에 취약점 진단을 통합하여 개발, 취약점 진단, 운영, 관리에 이르는 모든 영역에 지속적 보안을 적용하는 것을 의미한다[1]. DevOps에서 제시하는 지속적인 통합(Continuous Integration; CI)/지속적인 배포(Continuous Deployment; CD) 파이프라인의 빠른 시간에 개발 및 배포하는 목적을 맞추기 위해서는 보안 또한 CI/CD 파이프라인에 통합되어 자동화가 이루어져야 한다[2]. 따라서 DevSecOps도 CI/CD 파이프라인에 취약점 진단 도구를 결합하여 자동으로 취약점 진단을 수행하는 보안 통제 환경을 구성해야 한다[3]. 기존 DevSecOps 파이프라인은 주로 배포(Deploy) 다음 단계에서 진단하는 프로토타입 기반 취약점 진단 도구를 사용한다. 하지만 이 진단 도구는 배포 단계의 개발 프로토타입을 대상으로 하므로, 포괄적인 진단 결과만을 도출한다[4].

본 논문은 코드 기반으로 추가 분석하여 보안성을 높이는 DevSecOps 파이프라인을 제안한다. 기존 연구에서 주로 채택하는 프로토타입 기반 자동화 취약점 진단 도구에서 놓칠 수 있는 취약점을 두 단계의 진단으로 더 안전한 결과를 얻을 수 있다.

Ⅱ. 본론

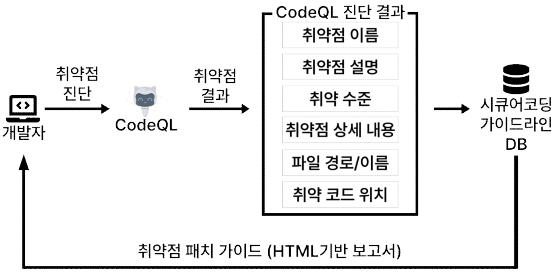
본 논문에서 제안하는 DevSecOps 파이프라인은 그림 1과 같다. 먼저, 개발자가 소스 코드를 깃허브(GitHub)에 Push하면 CodeQL이 먼저 그 소스 코드를 데이터로 삼아 취약점 진단을 수행하고 취약점 진단한다[5]. 그리고 배포 단계에서 개발 프로토타입을 OWASP ZAP에 넘겨 추가적인 취약점 진단을 수행한다[6]. 이때, CodeQL에서 이미 진단한 취약점 목록이 있으므로 진단 목록을 넘겨 중복된 진단을 방지한다. 이를 통해 기존의



[그림 1] 제안하는 DevSecOps 파이프라인 개념도

취약점 진단 외에도 개발자가 코드 작성 단계에서 직접 취약점을 패치할 수 있으므로 한층 더 강화된 보안성을 지닌 웹 애플리케이션을 제작할 수 있다.

제안하는 아이디어인 CodeQL을 사용한 코드 기반 취약점 진단의 세부 과정은 그림 2와 같다. 개발자는 깃허브에 소스 코드를 Push하면, 쿼리 명령을 통해 취약점 진단을 수행하며, 진단 결과를 다수의 csv 파일로 출력한다. 그리고 출력 파일들과 CWE 목록을 담고있는 시큐어코딩 가이드라인 DB를 연동하여 HTML 형태의 보고서로 변환한다. 변환과정은 먼저 모든 결과 파일들을 순회하고 하나씩 실행하며 진단한다. 다음으로 취약점 이름에 맞게 각각의 실행 결과 파일들을 생성하여 준다. 마지막으로 작성된 csv 파일들을 토대로 시큐어코딩 가이드라인 URL을 담은 데이터베이스에서 참고할 수 있는 URL을 가져와 취약점 이름, 상세 내용 등을 HTML 형태로 제공한다. 따라서 개발자가 소스 코드를 깃허브에 Push만 하면 개발자가 직접 취약점을 올바르게 패치 할 수 있다. 추가로 본 논문에서 개발 자동화를 구현하기 위해 웹훅(Webhook) 설정을 하였다. 웹훅을 사용하면, 개발자가 직접 빌드(Bulid)하는 것이 아니라 깃허브 Push 시, 자동으로 개발 코드를 Pull하여 젠킨스(Jenkins)에서 빌드를 수행한다.



[그림 2] CodeQL을 사용한 코드 기반 취약점 진단의 세부 과정

Ⅲ. 실험 결과

제안한 DevSecOps 파이프라인의 실험을 위해 취약점을 내포한 중고거래 사이트를 제작하였다. 이 사이트에서 의도적으로 만든 취약점은 교차 사이트 스크립팅(Cross-Site Scripting; XSS), SQL 삽입(SQL Injection), 파일 업로드 취약점(File Upload), 경로 조작 취약점(Path Traversal), 서버 측 요청 위조(Server-Side Request Forgery; SSRF) 등 다양한 취약점을 내포하여 제작하였다[7]. 개발환경 중 프로그래밍 언어는 Python 3.10.8 버전, 웹 프레임워크는 Flask 2.2.2 버전, 데이터베이스는 SQLite3 3.32.2 버전으로 구축하였다.

성능평가를 위해서 기존 프로토타입 기반 취약점 진단 도구와 본 논문에서 제안하는 코드 기반 취약점 진단 도구의 취약점 탐지 결과와 수행시간을 비교하였다. 먼저 OWASP ZAP만으로 취약점을 진단한 결과 웹 개발 당시 넣었던 취약점인 XSS, SQL Injection, Path Traversal이 발견되었지만 SSRF 공격 취약점은 탐지하지 못하는 것을 확인할 수 있었고, 개발된 웹에 대한 취약점 진단 수행시간은 5분 50초가 소요된 것을 확인할 수 있었다. 반면 CodeQL을 이용하여 웹 애플리케이션 취약점 진단을 한 결과, 개발된 웹에 내포한 취약점들인 XSS, Path Traversal, SQL Injection과 OWASP ZAP에서 발견하지 못한 SSRF가 발견된 것을 확인할 수 있었고, 취약점 진단 수행시간은 9분 32초가 소요된 것을 확인할 수 있었다.

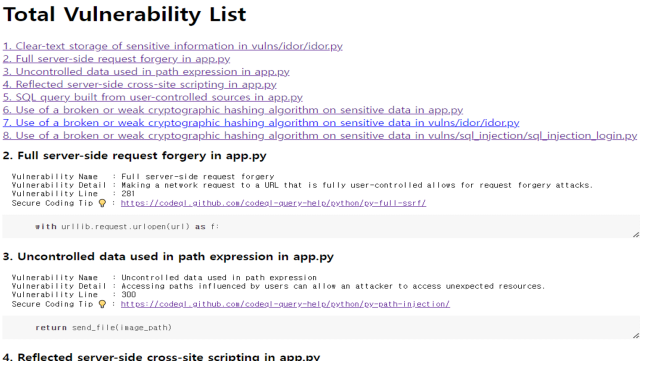
[표 1] 실험용 중고거래 사이트의 취약점 진단 결과 비교

취약점	OWASP ZAP	CodeQL
XSS	O	O
SQL Injection	O	O
Path Traversal	O	O
SSRF	X	O

[표 2] 실험용 중고거래 사이트의 취약점 진단 수행시간 비교

취약점	OWASP ZAP	CodeQL
소스 코드 취약점 진단	-	9분 32초
프로토타입 취약점 진단	5분 50초	5분 50초

표 1은 실험 대상의 취약점 진단 결과를 비교한 내용이다. 프로토타입 기반 취약점 점검 도구인 OWASP ZAP은 최신 등장하는 취약점들에 대한 대비가 부족한 결과를 확인하였다. 이는 배포 단계의 개발 결과물을 대상으로 하였기 때문에 소스 기반의 검사보다 부족한 것으로 판단된다. 또한 표 2는 실험 대상의 취약점 진단 수행시간을 비교한 내용이다. 제안하는 방식은 두 부분에서 취약점 진단을 수행하므로 총 걸리는 시간이 당연히 많아야 한다. 하지만 개발 자동화로 인해서 개발 과정 중에 취약점을 진단하는 것은 큰 문제가 되지 않으리라 판단한다. 그리고 소스 코드를 작성하는 개발자에게 스스로 보안 패치를 수행할 수 있도록 보안 지식을 전달한다는 점은 제안한 방법의 장점이다. 다만 CodeQL에서 이미 진단한 취약점 목록을 제외하고 OWASP ZAP을 수행했음에도 수행시간이 줄어들지 않았다는 점은 예상했던 결과를 벗어난다. 추가로 제안한 시큐어코딩 가이드라인 보고서는 그림 3과 같다. 이 보고서는 CodeQL의 쿼리 결과와 시큐어코딩 가이드라인 DB를 활용하여 개발자에게 취약점 패치 가이드를 할 수 있다.



[그림 3] 제안하는 시큐어코딩 가이드라인 보고서

Ⅳ. 결론

본 논문에서는 CodeQL 취약점 진단과 시큐어코딩 가이드라인 제공 기능을 통해 개발자는 코드를 GitHub에 Push만 하면 개발을 하면서 취약점을 진단할 수 있다. 이러한 진단을 통해 보안성 측면에서 늘어나는 최신 취약점 유형과 웹 프레임워크에 대해서 우수하다는 것을 확인하였다. CodeQL은 사용자가 직접 쿼리를 작성할 수 있으므로 기존에 있던 CWE 쿼리를 발전시키거나 새로운 취약점에 대해 보안 연구자가 쿼리를 작성한다면 더욱 다양한 취약점도 찾아낼 수 있다. 웹 서비스의 규모가 커지고, 공격 벡터가 많아지고 취약점이 패치될수록 속도 측면에서 더욱 좋아질 것으로 예상한다. 또한, 본 논문에서 제공해 주는 시큐어코딩 가이드라인 보고서를 참고하여 개발자가 따로 개발하는 과정에서 취약점에 알맞은 근본 원인에 대해 빠르고 올바르게 시큐어코딩을 적용할 수 있어 보안성 향상을 기대할 수 있다.

참 고 문 헌

[1] Mun Jin Young, "A Study on Security in the Cloud Native Environment From the Perspective of DevSecOps", M.S. thesis, KU, 2020.

[2] H. Myrbakken and R. Colomo-Palacios, "DevSecOps: A multivocal literature review," *Commun. Comput. Inf. Sci.*, vol. 770, pp. 17 - 29, Sep. 2017.

[3] Kim Se-hwan. "A Study on Web Application Security Management Plan in DevOps Environment", M.S. thesis, KU, 2021.

[4] OWASP ZAP. "Getting Started," zaproxy.org. [Online]. Available: https://www.zaproxy.org/getting-started [Accessed Nov 23 2022].

[5] CodeQL documentation. "About CodeQL", codeql.github.com. [Online]. Available: codeql.github.com/docs/codeql-overview/about-codeql/.[Accessed Nov 23 2022].

[6] OWASP ZAP. "owasp zap api," zaproxy.org. [Online]. Available: https://www.zaproxy.org/docs/api/?python#basics-on-the-api-request [Accessed Nov 23 2022].

[7] So Jin-sook. "Software Development Security Guide." Ministry of Public Administration and Security, KR, 2021.