# 정적 및 동적 분석 도구를 활용한 CWE-79 취약점 탐지 및 보안 분석 기법 연구

우정현[1], 서병석[2], 고광만[3]
[1]상지대학교 컴퓨터공학과 석사과정
[2,3]상지대학교 컴퓨터공학과 교수
2023015102@sj.sangji.ac.kr, seobs@sangji.ac.kr, kkman@sangji.ac.kr

# Cross-Check Based Vulnerability Analysis Using Static and Dynamic Analysis Tools: A Study on CWE-79 Detection

Jung-Hyun Woo[1], Kwang-Man Ko[2]
[1]Master's Program in Computer Science, Sangji University
[2]Professor of Computer Science, Sangji University

요 약

이 연구는 웹 애플리케이션 보안에서 흔히 발생하는 CWE-79(XSS) 취약점을 탐지하기 위해 정적 분석 도구와 동적 분석 도구의 이론적 비교를 수행하였다. SonarQube와 OWASP ZAP을 중심으로, 두 도구의 탐지 메커니즘과 장단점을 분석하고, 이들이 보안 점검에 미치는 영향을 논의하였다. 연구 결과, 정적 분석 도구와 동적 분석 도구는 상호 보완적인 관계에 있으며, 두 도구를 결합하여 사용할 때 취약점 탐지의 효율성과 정확도가 크게 향상될 수 있음을 확인하였다. 또한, 이 연구는 AI 기반 탐지 기법을 도입하기 위한 기초 자료를 제공하며, 향후 연구에서 AI를 활용한 보다 정교한 보안 분석 기법을 개발할 가능성을 제시하였다. 이번 연구는 실질적인 실험 데이터를 포함하지 않았지만, 이론적 분석을 통해 정적 및 동적 분석 도구의 보안성 강화 방안을 모색하고, 보안 분석 기법의 발전을 위한 기초 데이터를 마련하였다.

## 1. 서론

웹 애플리케이션 보안에서 CWE-79(XSS) 취약점은 가장 흔하고 치명적인 문제 중 하나로, 사용자의 입력을 적절히 처리하지 못해 발생하는 보안 약점이다. 이러한 취약점은 공격자가 악성 스크립트를 주입하여 사용자 세션을 탈취하거나 웹 페이지의 내용을 변조하는 등의 공격을 가능하게 한다. 이러한 위험성을 고려할 때, 효과적인 취약점 탐지 기법의 개발은 매우 중요한 과제이다.

기존의 보안 분석 도구들은 정적 분석과 동적 분석이라는 두 가지 주요 접근 방식을 사용한다. 정적 분석 도구는 소스 코드를 기반으로 잠재적인 보안 취약점을 식별하며, 동적 분석 도구는 실제로 실행 중인 애플리케이션을 테스트하여 취약점을 탐지한다. 이 연구에서는 정적 분석 도구인 SonarQube와 동적 분석 도구인 OWASP ZAP을 사용하여 CWE-79 취약점을 탐지하는 방법을 이론적으로 분석하고, 두 도구의 장단점을 고찰한다.

Jenkins와 GitHub을 연동한 자동화된 CI/CD 파이프라인을 구성하여, 보안 분석 과정의 효율성을 극대화하는 방법을 논의하고, 향후 AI 기반 탐지 기법을 도입하기 위한 기초 연구로서, 이론적 배경을 제공하는 데 중점을 둔다.

## 2. 에이전트 개발도구의 요구사항

### 2.1 정적분석도구

정적 분석 도구는 소스 코드를 실행하지 않고 코드 자체를 분석하여 잠재적인 보안 취약점을 식별한다. SonarQube는 이러한 정적 분석 도구의 대표적인 예로, 다양한 프로그래밍 언어를 지원하며 코드 품질과 보안 취약점을 관리하는 데 널리 사용된다. SonarQube는 코드의 구조적 문제, 잘못된 패턴, 그리고 보안 관련 문제를 식별하는 데 강점을 가진다. 그러나 정적 분석 도구는 실행 환경의 동적 요소를 반영하지 못하므로, 실제로 실행 시 발생할 수 있는 보안 취약점을 놓칠 수 있다.

### 2.2 동적 분석 도구

동적 분석 도구는 실제로 실행 중인 애플리케이션을 테스트하여 보안 취약점을 탐지한다. OWASP ZAP은 이러한 동적 분석 도구의 대표적인 사례로, 웹 애플리케이션의 런타임 환경에서 취약점을 탐지하고

공격 시나리오를 검증할 수 있다. 동적 분석 도구는 실행 환경에서의 보안 문제를 실시간으로 탐지할 수 있는 장점이 있지만, 코드의 전체적인 탐색이 불가능하고 실행되지 않은 코드 경로에 대한 분석이 어려운 한계가 있다.

2.3 CWE-79 취약점 탐지에 대한 기존 연구

CWE-79(XSS) 취약점은 웹 애플리케이션 보안에서 가장 빈번하게 발생하는 문제 중 하나로, 여러 연구에서 이를 탐지하고 방어하기 위한 다양한 접근 방식이 제안되었다. 정적 분석과 동적 분석 도구를 결합하여 XSS 취약점을 탐지하는 방법이 기존 연구에서 많이 다뤄졌으며, 두 도구의 조합이 보다 효과적인 보안 점검을 가능하게 한다는 결과도 보고되었다. 이번 연구는 이러한 기존 연구들을 바탕으로 정적 및 동적 분석 도구의 이론적 비교를 통해, 향후 보안 분석의 정확성과 효율성을 높이는 방법을 모색하고자 한다.

## 3. 연구방법

3.1 연구 설계

이 연구는 CWE-79(XSS) 취약점을 탐지하기 위해 정적 분석 도구와 동적 분석 도구의 성능을 이론적으로 비교하는 것을 목표로 한다. 연구에서는 SonarQube와 OWASP ZAP을 각각 사용하여 웹 애플리케이션 코드에서 발생할 수 있는 XSS 취약점을 탐지하고, 두 도구가 제공하는 탐지 결과의 차이와 보완적 가능성을 분석한다. 이를 통해 각 도구의 장단점을 파악하고, 향후 보안 분석에 필요한 개선 사항을 제안할 기초 데이터를 마련한다.

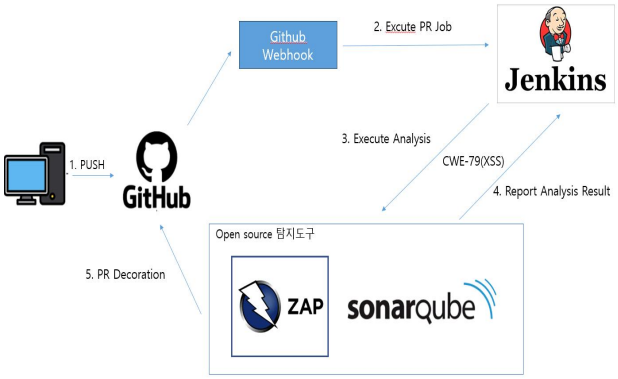


그림 1. CI/CD Pipeline for XSS Detection

그림 1은 CWE-79 취약점 탐지를 위해 연구에서 사용된 SonarQube와 OWASP ZAP의 전체적인 분석 과정 오버뷰를 보여준다. 이 과정은 Jenkins와 GitHub을 통해 자동화되며, 향후 AI 기반 탐지 기법의 도입 가능성을 탐색하기 위한 기초 작업으로 설계되었다.

3.2 실험 환경

실험은 Jenkins와 GitHub을 연동한 자동화된 CI/CD 파이프라인에서 수행된다. GitHub에 저장된 웹 애플리케이션 소스 코드를 Jenkins를 통해 자동으로 빌드하고, 빌드된 코드는 SonarQube와 OWASP ZAP을 사용해 분석된다. 분석 결과는 CI/CD 파이프라인에 통합되어 실시간으로 보고된다. 이 연구는 이러한 자동화된 프로세스를 활용하여 정적 및 동적 분석 도구의 성능을 비교하는 이론적 기반을 제공한다.

3.3 데이터 분석 및 논의

이 연구에서는 CWE-79(XSS) 취약점 탐지를 위해 생성된 웹 애플리케이션 소스 코드를 사용하여 데이터를 수집하였다. 연구의 목적에 맞게 특정 취약점을 인위적으로 포함시킨 코드베이스를 구축하였으며, 이를 통해 정적 분석 도구와 동적 분석 도구의 탐지 성능을 평가하였다. SonarQube와 OWASP ZAP을 사용하여 각각의 도구가 발견한 CWE-79(XSS) 취약점의 수와 유형을 기록하였으며, 탐지되지 않은 취약점도 함께 분석하였다.

수집된 데이터는 연구의 신뢰성을 확보하기 위해 여러 번 반복된 실험을 통해 얻어졌으며, 각 실험에서 동일한 조건을 유지하였다. 생성된 데이터는 재현 가능한 방법으로 구성되었으며, 다른 연구자들도 동일한 실험을 통해 유사한 데이터를 수집할 수 있도록 설계되었다.

3.4 AI 기반 탐지 기법 도입에 대한 기초 연구

이번 연구는 AI 기반 탐지 기법을 도입하여 HTML 파일에서 발생할 수 있는 XSS(크로스 사이트 스크립팅) 공격을 탐지하는 방법을 이론적으로 검토한다. AI를 활용한 보안 분석 기법은 전통적인 정적 및 동적 분석 도구의 한계를 보완하며, 더 높은 정확도와 효율성을 제공할 수 있는 가능성을 가진다. 그림 2는 AI 기반의 머신러닝 모델을 활용하여 HTML 파일에서 XSS 공격을 탐지하는 과정의 개요를 보여준다. 이 프레임워크는 HTML 파일에서 데이터를 추출하고, 데이터 처리와 특징 추출을 거친 후, 다양한 머신러닝 알고리즘을 적용하여 악성(Malicious) 또는 정상(Benign) 여부를 분류한다.

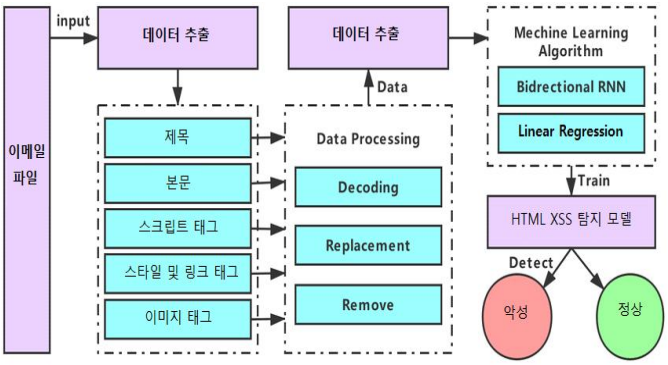


그림 2. AI 기반 HTML XSS 탐지 프레임워크

이 프레임워크에서는 다음과 같은 주요 단계가 포함된다:

데이터 추출: HTML 파일에서 제목, 본문, 스크립트 태그, 스타일 태그 등 다양한 요소를 추출한다. 이 단계에서는 XSS 공격이 삽입될 수 있는 다양한 HTML 요소를 분석 대상으로 삼는다.

특징 추출: 추출된 데이터로부터 중요한 특징을 식별한다. 이 특징들은 머신러닝 모델이 XSS 공격을 학습하고 탐지할 수 있도록 필요한 정보를 제공한다.

데이터 처리: 추출된 데이터를 디코딩, 교체, 제거와 같은 전처리 작업을 통해 정제한다. 이 과정은 데이터의 품질을 높여 머신러닝 모델의 학습 효율성을 향상시킨다.

머신러닝 알고리즘 적용:

선형 회귀 (Linear Regression): 선형 회귀는 입력 변수와 출력 변수 간의 선형 관계를 학습하여 XSS 공격 가능성을 예측하는 데 사용된다. HTML 파일의 특정 패턴이 XSS 공격과 어떤 상관관계를 가지는지 평가한다.

양방향 순환 신경망 (Bidirectional RNN): 양방향 RNN은 데이터를 순방향과 역방향으로 동시에 처리하여, HTML 데이터에서 더 깊은 컨텍스트를 학습하고, 복잡한 XSS 패턴을 효과적으로 탐지할 수 있도록 한다.

탐지 모델 훈련 및 예측: 훈련된 모델은 HTML 파일을 분석하여, 해당 파일이 XSS 공격을 포함하는지 여부를 탐지한다. 이 과정은 모델이 학습한 내용을 바탕으로, HTML 파일이 악성(Malicious)인지 정상(Benign)인지를 결정하는 단계이다.

이 연구는 이러한 AI 기반 탐지 기법이 기존의 정적 및 동적 분석 도구와 결합되어 XSS 공격을 더 효과적으로 탐지할 수 있는 가능성을 제시한다. 향후 연구에서는 이론적 분석을 바탕으로 실제 데이터셋을 이용해 모델의 성능을 검증하고, 실질적인 보안 향상을 위한 다양한 AI 알고리즘의 적용 가능성을 탐색할 예정이다.

## 4. 실험 결과

### 4.1 정적 분석 도구와 동적 분석 도구의 비교

이번 연구에서는 정적 분석 도구인 SonarQube와 동적 분석 도구인 OWASP ZAP을 활용하여 CWE-79(XSS) 취약점을 탐지하는 방법을 이론적으로 분석하였다. 두 도구는 각각 고유한 탐지 메커니즘을 가지고 있으며, 이로 인해 탐지 성능에 차이가 발생한다.

정적 분석 도구 SonarQube는 소스 코드의 구조적 문제와 잘못된 패턴을 탐지하는 데 효과적이다. SonarQube는 실행되지 않은 코드 경로까지 포함하여 코드를 전반적으로 분석할 수 있는 강점을 가진다. 이 도구는 특히 코드 리뷰 과정에서 발생할 수 있는 잠재적 취약점을 조기에 발견하는 데 유용하다. 그러나 정적 분석의 특성상, 실행 중 발생할 수 있는 런타임 취약점을 탐지하는 데는 한계가 있다.

동적 분석 도구 OWASP ZAP은 실제로 실행 중인 애플리케이션을 테스트하여 취약점을 탐지하는 데 중점을 둔다. ZAP은 웹 애플리케이션의 런타임 환경에서 발생할 수 있는 실제 공격 시나리오를 모의실험하는 데 탁월한 성능을 보인다. 그러나 실행되지 않은 코드 경로에 대해서는 탐지가 불가능하며, 전체적인 코드의 보안성을 보장하기 위해서는 정적 분석 도구와의 결합이 필요하다.

### 4.2 정적 분석과 동적 분석의 보완적 사용

이론적 분석 결과, 정적 분석 도구와 동적 분석 도구는 각각의 장단점이 명확하게 구분되며, 이 두 가지 도구를 보완적으로 사용하는 것이 CWE-79(XSS) 취약점을 탐지하는 데 매우 효과적일 수 있다는 결론에 도달할 수 있다. 정적 분석 도구는 코드의 구조적 안전성을 보장하고, 동적 분석 도구는 실행 환경에서의 실제 보안성을 확인하는 역할을 한다. 두 도구의 결합 사용은 취약점 탐지의 범위와 정확도를 크게 향상시킬 수 있다.

### 4.3 AI 기반 탐지 기법의 가능성

이번 연구에서는 AI 기반 탐지 기법의 도입 가능

성에 대해 이론적으로 검토하였다. AI는 데이터 학습을 통해 새로운 유형의 취약점을 탐지하고, 정적 분석 도구와 동적 분석 도구가 놓칠 수 있는 미묘한 패턴을 식별하는 데 유리할 수 있다. 향후 연구에서는 AI를 활용한 보안 분석 기법을 개발하여, 기존의 정적 및 동적 분석 도구의 한계를 보완하고, 보안 점검의 정확성과 효율성을 한층 더 향상시킬 수 있을 것으로 기대된다.

## 5. 결론

이번 연구는 정적 분석 도구인 SonarQube와 동적 분석 도구인 OWASP ZAP을 활용하여 CWE-79(XSS) 취약점을 탐지하는 방법을 이론적으로 분석하였다. SonarQube는 소스 코드 내에서 발생할 수 있는 잠재적 취약점을 폭넓게 탐지하는 데 강점을 보였으며, 실행되지 않은 코드 경로까지 분석할 수 있다는 장점을 가졌다. 반면, OWASP ZAP은 실제로 실행 중인 애플리케이션의 런타임 환경에서 발생할 수 있는 취약점을 탐지하는 데 뛰어난 성능을 발휘하였다. 두 도구의 비교를 통해, 각각의 접근 방식이 보안 점검에 서로 다른 방식으로 기여할 수 있음을 확인할 수 있었다.

연구 결과, 정적 분석 도구와 동적 분석 도구는 상호 보완적인 관계에 있으며, 두 도구를 결합하여 사용하는 것이 CWE-79(XSS)와 같은 취약점을 보다 효과적으로 탐지하는 데 유리하다는 결론에 도달하였다. 또한, 이번 연구는 향후 AI 기반 탐지 기법을 도입하기 위한 기초 연구로서, AI가 기존의 정적 및 동적 분석 도구의 한계를 극복하고 보안 점검의 정확성과 효율성을 한층 더 향상시킬 수 있는 가능성을 탐색하였다.

이론적 분석에 중점을 둔 이번 연구는 실질적인 실험 결과를 포함하지 않았으나, 정적 및 동적 분석 도구의 이론적 기초를 확립하고, 향후 연구에서 실험적 접근을 통해 이를 검증할 수 있는 기초 자료를 제공하였다. 앞으로의 연구에서는 AI 기반 기법을 포함한 보다 정교한 보안 분석 방법을 개발하여, 현재의 도구들이 놓칠 수 있는 취약점을 보다 정확하게 탐지할 수 있는 방법을 모색할 예정이다.

## 참고문헌

[1]정철모, "정적 분석 도구와 동적 분석 도구를 활용한 소프트웨어 보안 취약점 탐지 기법 비교 연구," 한국산업융합학회논문집, 제19권, 제12호, pp. 96-103, 2016.

[2]S. L. Garfinkel, "Automated Web Application Security Testing," IEEE Security & Privacy, vol. 4, no. 4, pp. 16-23, 2006.

[3]Y. Hwang, K. G. Shin, and S. P. Shieh, "Web Application Vulnerability Detection Using Static Analysis," Proceedings of the 7th International Conference on Information Security Practice and Experience (ISPEC), Guangzhou, China, 2011, pp. 109-124.

[4]OWASP Foundation, OWASP Testing Guide v4, New York, NY: OWASP Foundation, 201