스패로우

구성 요소 분석 기술

화이트 페이퍼

주식회사 스패로우

SPARROW

CONTENTS

1 핵심 기술 개요

당사의 솔루션에는 복잡하고 다양한 기술이 적용됩니다. 그 중에서 핵심 기술의 종류는 크게 분석 기술과 통합 기술이라는 두 가지로 구분할 수 있습니다. 분석 기술은 솔루션의 코어라고 볼 수 있는 분석 엔진의 분석 수행 방법에 관련된 기술입니다. 여기에 포함된 기술의 종류는 정적 분석, 동적 분석, 구성 요소 분석으로 나누어집니다. 통합 기술은 이러한 분석의 결과를 활용하는 방법과 관련된 기술입니다. 여기에는 개별 분석 기술의 결과를 종합적으로 판단해서 결과를 표시하는 결과 통합 기술과 사용자의 소프트웨어 통합 및 배포(CI/CD) 파이프라인에 당사의 솔루션을 연결하여 활용하는 DevSecOps 기술이 포함됩니다.

구성 요소 분석 기술은 당사의 전주기적 통합 애플리케이션 보안 테스팅 솔루션에서 제공하는 분석 기술인 정적 분석, 동적 분석, 구성 요소 분석 중 하나입니다. 세 가지 기술은 분석에 사용하는 목적물(분석 대상)의 차이를 기준으로 분류되며, 각 기술을 통해 분석을 수행하는 절차 혹은 방법(분석 방법), 분석 후 표시되는 내용(분석 결과)에도 고유한 특징이 있습니다. 이 문서에서는 구성 요소 분석 기술의 분석 대상에 따른 분석 방법 및 분석 결과에 대한 설명을 간략히 서술하도록 합니다.

2 구성 요소 분석 기술

오픈소스 소프트웨어는 소프트웨어 개발이라는 과정에서 중요한 구성 요소입니다. 대규모 소프트웨어를 개발하기 위해서는 다양한 경로로 공유되는 오픈소스 소프트웨어를 사용할 수밖에 없습니다. 구성 요소 분석(SCA, Software Composition Analysis)은 소프트웨어 애플리케이션에서 사용하는 오픈소스 및 서드파티 라이브러리를 식별하고 여기서 알려진 보안 취약점과 라이선스 준수 문제 등 관련된 정보를 제공하는 기술입니다.

소프트웨어 개발 프로세스에서 오픈소스 사용이 증가함에 따라 이러한 구성 요소의 보안과 라이선스 관리가 중요한 이슈로 부각되고 있습니다. 오픈소스에 포함된 문제는 해당 오픈소스를 사용하는 소프트웨어의 문제로 직결될 수 있기 때문입니다. 따라서 구성 요소 분석 기술로 오픈소스의 라이브러리와 버전을 찾고 데이터베이스와 비교하여 취약점과 같은 잠재적 위협 또는 라이선스와 같은 법적 문제를 사전에 파악하는 것이 중요합니다.

당사의 구성 요소 분석 기술은 다양한 형태의 소프트웨어에서 정보를 추출하고, 추출된 정보를 통해 오픈소스 컴포넌트를 식별합니다. 이 과정에서 패키지 매니저 설정 파일, 바이너리 파일, 안드로이드 바이너리 파일 등 다양한 소스에서 데이터를 수집하고 분석하며, 파일 해시 조회 기술을 통해 오픈소스 컴포넌트를 정확히 매칭합니다.

또한, 널리 사용되는 Docker 컨테이너 이미지 파일을 분석하는 경우 컨테이너 이미지의 구조를 분석하여 파일의 이름 및 버전을 판별하고 파일을 추출합니다. 이렇게 추출된 파일은 다시 분석되어 오픈소스 컴포넌트를 식별하는 과정을 거치게 됩니다.

이러한 분석 기술을 구현하기 위해서는 오픈소스 컴포넌트에 대한 정보를 미리 수집하고 정리하여 데이터를 구조화해야 합니다. 오픈소스 소프트웨어는 수많은 리포지토리에 분산된 상태로 저장되어 있는 방대한 자료입니다. 하나의 오픈소스에는 여러 가지 버전이 포함되며, 버전마다 취약점 정보 및 라이선스 정보에 차이가 있을 수 있습니다.

당사에서는 오픈소스에 대한 자료를 스캔하여 수집하는 데이터 레이크(Data Lake)와 수집된 자료를 정리하여 구조화한 데이터 웨어하우스(Data Warehouse)를 자체적으로 운영하고 있습니다. 데이터 레이크와 데이터 웨어하우스에 저장된 오픈소스 소프트웨어에 대한 정보를 주기적으로 업데이트함으로써 구성 요소 분석 기술을 통해 최신 오픈소스 컴포넌트를 검출할 수 있게 됩니다.

2.1 바이너리 분석

바이너리 파일은 컴퓨터가 직접 실행할 수 있는 형태의 파일로서 exe, dll과 같은 실행 파일이 여기에 해당합니다. 바이너리 파일은 대부분 단일 파일의 형태입니다. 당사의 바이너리 분석 기술은 이러한 바이너리 파일의 특징에 주목하여 파일 내용을 해시로 만들어 비교 분석하는 방법을 사용합니다.

즉, 소프트웨어에 포함된 바이너리 파일의 파일 내용 해시가 오픈소스 컴포넌트의 파일 내용 해시와 동일하다면 해당 오픈소스 컴포넌트를 사용하고 있다고 판단합니다. 윈도우 바이너리 파일과 안드로이드 바이너리 파일의 경우에는 오픈소스 컴포넌트 정보를 추론하기 위해 파일 내용 해시 이외에도 파일 설정에 기록되어 있는 정보를 추가적으로 활용합니다.

2.2 소스 코드 분석

당사의 구성 요소 분석 중 소스 코드 분석 기술은 소스 코드 형식의 오픈소스 컴포넌트를 식별하는데 사용됩니다. 소스 코드 형태로 포함되는 오픈소스 컴포넌트는 일반적으로 여러 개의 파일이 하나의 컴포넌트를 구성하고 있습니다. 따라서 파일 내용 해시만을 사용해서 분석하는 경우 분석 대상이 컴포넌트를 구성하는 소스 코드 파일 중 일부만 같은 다른 컴포넌트로 인식하는 오탐이 발생할 수 있습니다.

따라서 소스 코드를 분석하는 경우 소스 코드 확장자 및 파일의 상대 경로를 확인하여 해당 파일의 상대 경로에 식별하려는 컴포넌트의 경로가 포함되었는지를 비교합니다. 이런 방식으로 특정 컴포넌트가 소스 코드 형태로 포함되었는지를 구분합니다.

2.3 의존성 분석

Maven, Gradle, NPM 등의 경우 패키지 매니저를 사용하여 외부 패키지를 쉽게 다룰 수 있도록 도와줍니다. 패키지 매니저는 설정 파일을 통해 오픈소스 소프트웨어에 대한 정보와 해당 소프트웨어에 연결되는 오픈소스 컴포넌트 정보를 관리하고 있습니다.

패키지 매니저의 설정 파일에서는 특정 오픈소스 소프트웨어의 이름과 버전뿐만 아니라 해당 소프트웨어가 사용하고 있는 오픈소스 컴포넌트의 이름과 버전이 명시적으로 작성되어 있습니다. 당사의 의존성 분석 기술은 패키지 매니저를 통해 소프트웨어에 추가된 오픈소스 컴포넌트 정보를 추출하고 이 정보를 직접적으로 사용되는 오픈소스(직접 의존성) 및 간접적으로 사용되는 오픈소스(간접 의존성)로 구분합니다.

만약 패키지 매니저 설정 파일에 컴포넌트 정보가 명시되지 않았다면 다른 패키지 매니저 설정 파일을 함께 분석하게 됩니다. 패키지 매니저에 따라 서브 프로젝트를 설정하거나 다른 파일과 연결된 오픈소스 컴포넌트 정보를 정의하고 추가하는 방식이 다릅니다. 따라서 패키지 매니저에 따라 작성된 설정 파일을 해석한 다음, 정확한 컴포넌트 정보를 추출하는 것이 중요합니다.

2.4 컨테이너 분석

컨테이너 이미지는 압축 파일처럼 여러 파일과 디렉토리를 하나로 묶어둔 패키지입니다. 이미지 내부에는 애플리케이션 실행 파일과 관련 라이브러리, 의존성 패키지들, 설정 파일들과 같은 다양한 파일들이 포함될 수 있습니다. 이러한 파일은 컨테이너 이미지 안에 레이어(layer) 구조로 저장됩니다. 레이어 구조에는 일정한 규칙이 있기 때문에 이 구조를 파악함으로써 컨테이너에 포함된 구성 요소를 식별할 수 있습니다.

당사의 컨테이너 분석 기술은 컨테이너 이미지의 레이어 구조를 분석하고 이미지 파일 안에 있는 파일을 추출하여 정보를 수집합니다. 추출된 정보는 앞서 설명한 바이너리 분석, 소스 코드 분석 및 의존성 분석 기술 과정을 다시 한번 거친 다음, 구성 요소 분석을 수행하게 됩니다.

2.5 데이터 레이크 및 데이터 웨어하우스

당사에서는 구성 요소 분석에 사용하기 위해 소프트웨어 리포지토리에서 직접 오픈소스 컴포넌트에 대한 데이터를 수집하고 저장하는 데이터 레이크를 운영하고 있습니다. 데이터 레이크에서 수집하는 정보는 오픈소스 컴포넌트의 이름, 버전, 리포지토리 정보 등을 포함하는 메타데이터, OSV(Open Source Vulnerabilities) 혹은 CVE(Common Vulnerabilities and Exposures)와 같은 보안 취약점 정보에서 가져온 취약점 데이터, SPDX(Software Package Data Exchange) 등에서 제시한 라이선스 분류에 따른 라이선스 데이터로 구분됩니다.

수집된 원시 데이터 DB는 원본 데이터와 유사한 형식으로 저장되어 있습니다. 이 데이터를 구성 요소 분석 엔진에서 활용하기위해서는 원본 데이터를 가공하고 적절한 형식으로 변환해서 필요한 데이터만 저장하고 매핑할 필요가 있습니다. 이렇게 필요한 데이터를 적절한 형식으로 쌓아둔 저장소가 데이터 웨어하우스입니다.

데이터 웨어하우스는 전세계 어디서나 접속할 수 있도록 클라우드 환경에 배포하여 웹 서버 형태로 운영되고 있습니다. 또한, 사용자가 공개된 클라우드 환경에 접속할 수 없는 환경인 경우에도 구성 요소 분석을 실행할 수 있도록 폐쇄망을 위한 오프라인 데이터 웨어하우스도 별도로 지원하고 있습니다.

구성 요소 분석 엔진은 데이터 웨어하우스에서 제공하는 데이터 조회 API를 사용합니다. 데이터 웨어하우스의 API를 통해 앞서 구성 요소 분석을 통해 추출한 분석 대상의 데이터를 데이터 웨어하우스의 데이터와 비교합니다. 그런 다음, 일치하는 오픈소스 컴포넌트의 정보를 분석 결과로 가져오게 됩니다.

2.6 SBOM 생성

SBOM(Software Bill of Materials)이란 소프트웨어를 구성하는 재료라고 할 수 있는 오픈소스, 서드파티 컴포넌트, 상용 라이브러리 등을 목록으로 정리한 파일입니다. SBOM은 다수의 공급 및 수요가 얽혀있는 소프트웨어 공급망에서 특정 소프트웨어 프로그램에 사용된 구성 요소의 투명성과 신뢰성을 확인하기 위해 필수적인 데이터입니다.

당사의 구성 요소 분석 기술에서는 SBOM을 분석하고 생성할 수 있습니다. SBOM을 생성하기 위해 앞서 설명한 바이너리 분석, 소스 코드 분석, 의존성 분석, 컨테이너 분석을 통해 식별된 컴포넌트를 사용합니다. 당사의 기술로 생성할 수 있는 SBOM 형식에는 국제적인 표준이라고 볼 수 있는 SPDX(Software Package Data Exchange), CycloneDX, SWID(Software Identification) 태그 및 NIS SBOM표준이 포함됩니다. 생성된 SBOM의 종류에 따라 컴포넌트의 이름 및 버전과 같은 기본 정보와 함께 컴포넌트에서 발견된 취약점 및 라이선스 정보도 확인할 수 있습니다.

3 맺음말

지금까지 구성 요소 분석 기술의 의미, 주요 기능의 동작, 특징에 대해 설명했습니다. 구성 요소 분석과 같은 핵심 기술로 분석할 수 있는 범위는 지속적인 연구 개발을 통해 확장되고 있으며 정확도 또한 향상되고 있습니다. 그리고 이렇게 수행된 분석의 결과를 직접적으로 활용할 수 있도록 오픈소스, 라이선스 등을 포함한 다양한 설명을 포함하고 있습니다. 당사의 구성 요소 분석을 사용하는 분석 솔루션을 직접 경험해보시고 당사에서 제공하는 기술 지원 및 컨설팅 서비스를 통해 소프트웨어 개발 및 운영에 활용해보시기 바랍니다.

추가적인 문의 사항은 스패로우 고객센터(https://cs.sparrow.im/ko/tickets)를 방문하세요.