안전한 소프트웨어를 위한 애플리케이션 보안 테스팅 통합 & 자동화 방안

비즈니스 화이트 페이퍼

주식회사 스패로우

SPARROW

CONTENTS

안전한 소프트웨어를 위한 애플리케이션 보안 테스팅 통합 & 자동화 방안

AI 기술의 비약적 발전으로 인해 소프트웨어 개발 프로세스의 품질과 생산성이 급격히 성장하여 소프트웨어 서비스의 규모 및 다양성이 변화하고 있습니다.

이러한 변화 속에서 애플리케이션 환경도 진화하고 있습니다. 마이크로서비스 아키텍처와 동적 인프라가 보편화되고, 빠른 개발 주기를 맞추기 위해 오픈소스 소프트웨어 의존도는 계속 높아지고 있습니다. 그 결과, 현대 애플리케이션은 전통적인 아키텍처와 비교할 수 없을 정도로 복잡하게 상호 연결되어 있습니다. 애플리케이션 아키텍처의 구조적 복잡성은 애플리케이션 공급자에게 더 많은 선택지를 제공하는 동시에 공격자에게는 그만큼 넓은 공격 표면을 열어주었습니다. 실제로 최근 몇 년간 주요 통신사, 클라우드 서비스 등 핵심 인프라에서 대규모 정보 유출 사고가 잇따라 발생했습니다. 이는 애플리케이션 보안 위협이 단순한 우려가 아닌 현실적 위협임을 보여줍니다.

이에 대응하는 움직임도 빨라지고 있습니다. 기업은 물론이고 각국 정부, 비영리 조직, 글로벌 커뮤니티가 나서서 보안 규제와 컴플라이언스 요구사항을 강화하고 있습니다. 이전에는 고려하지 않던 영역까지 보안 관리 대상에 포함되면서, 관리 범위가 지속적으로 확대되고 있습니다.

따라서 애플리케이션 보안을 위해 공급자가 모니터링해야 하는 보안 위협에 대한 부담도 가중되고 있습니다. 복잡한 서비스 인프라, 증가하는 현실적 보안 위협, 높아지는 규제 요구를 모두 만족시키기 위해 보안 담당자에게도 자동화된 테스팅 도구 및 실질적인 관리 방법이 필요합니다.

Sparrow Enterprise는 소프트웨어 보안을 위해 통합 애플리케이션 보안 테스팅을 제공하고 애플리케이션의 보안 정보를 단일 플랫폼에서 관리하는 도구로써 시장을 선도하고 있습니다. 1) 애플리케이션 개발부터 운영까지 전주기에 적용할 수 있는 포괄적이고 강력한 분석 기능, 2) 애플리케이션 생태계 전반에 걸쳐 발견된 취약점에 대한 시각화, 3) 위험도 기반 분류 체계를 통한 조치 우선순위 지정, 4) 개발-보안-운영 팀간 협력 프로세스를 위한 다양한 기능을 통해 사용자가 체계적으로 보안 위협에 대처할 수 있는 방법을 제시합니다. 또한 AI 기술을 취약점 조치에 적용하는 연구 등 지속적으로 새로운 애플리케이션 보안 기술을 개발하고 있습니다.

Sparrow Enterprise는 고객에게 세 가지 핵심 가치를 제공합니다. 첫째, 정부 규제 및 컴플라이언스 요구사항을 충족할 수 있는 기반을 마련합니다. 둘째, 취약점에 대한 평균 복구 시간(MTTR)을 단축하여 운영 효율성을 높이고 비용을 절감합니다. 셋째, DevSecOps 프로세스를 통해 애플리케이션 보안을 개발 단계부터 통합하고 지속적으로 관리할 수 있게 합니다. Sparrow Enterprise를 통해 실질적인 애플리케이션 보안 정책을 수립할 수 있는 방법을 모색해보시길 권합니다.

현대 애플리케이션 보안의 핵심, ASPM

애플리케이션 보안 태세 관리(Application Security Posture Management, ASPM)란 조직 전체의 애플리케이션 보안 상태를 중앙에서 지속적으로 평가하고 관리할 수 있도록 유지시키는 체계를 가리킵니다.

ASPM을 위해서는 개발부터 운영까지 전주기에 걸친 통합 보안 분석, 모든 애플리케이션 자산의 수집 및 연동, 그리고 리스크 기반의 우선순위 평가와 시각화가 필수적입니다. SAST, DAST, SCA, IaC 스캔 등 다양한 테스팅 결과를 자동으로 수집하고, 이를 종합적으로 분석하여 사용자가 효율적으로 관리할 수 있어야 합니다.
Sparrow Enterprise는 이러한 ASPM 요구사항을 충족하기 위해 다음 기능을 제공합니다.

요구사항 1 : 보안 상태 전반에 대한 가시성 확보

Sparrow Enterprise의 핵심 기능은 애플리케이션의 전주기에 걸친 보안 상태의 가시성을 확보해주는 것입니다. Sparrow Enterprise에서는 애플리케이션의 생명 주기 및 분석 대상에 따라 소스 코드 분석, 컴포넌트 분석, 웹 취약점 분석을 모두 수행할 수 있습니다.

통합 애플리케이션 보안 테스팅

소스 코드 분석은 정적 분석을 통해 소스 코드에 존재하는 잠재적 보안약점 및 품질 관련 문제를 정확하고 신속하게 검출합니다. 애플리케이션 소스 코드, 패키징된 파일, IaC(Infrastructure as Code) 및 관련된 모든 소스 코드를 자산으로 등록하고 분석합니다. 따라서 소스 코드를 릴리즈 서버 또는 운영 환경에 배포하기 전에 소프트웨어 패키지 혹은 서비스에서 발생할 수 있는 문제가 무엇인지 식별할 수 있습니다. 이 문제를 해결함으로써 실제 운영 환경에서 발생할 수 있는 문제를 사전에 예방하게 됩니다. 실제 문제가 발생하지 않더라도 소프트웨어의 보안과 품질을 보장함으로써 소프트웨어와 관련된 고객의 리스크를 관리할 수 있습니다.

컴포넌트 분석은 소스 코드나 바이너리 파일에 포함된 오픈소스를 진단하여 라이선스 관련 정보를 확인할 수 있는 오픈소스 소프트웨어 분석 방법입니다. 컴포넌트 분석을 통해 개발자가 프로그램에서 사용한 오픈소스를 식별함으로써 프로그램이 의존하는 오픈소스를 미리 확인할 수 있습니다. 특히, 오픈소스 컴포넌트가 취약한 컴포넌트인 경우 이미 취약성이 알려진 컴포넌트의 CVE ID 및 CVSS 점수를 제공합니다. 프로그램의 의존성을 분석한 경우 애플리케이션의 직간접적으로 활용하고 있는 오픈소스 컴포넌트에 대한 정보를 알 수 있습니다.

웹 취약점 분석은 동적 분석을 통해 실행 중인 웹 애플리케이션을 분석할 수 있습니다. 테스트 환경이나 운영 환경에 올려진 URL만 입력하면 자동으로 해당 URL에서 접근할 수 있는 모든 경로를 수집합니다. 수집한 정보를 바탕으로 웹 애플리케이션의 URL 파라미터, 헤더 및 바디 입력 값, 입력 필드, 링크 등 모든 이벤트 요소, 웹 앱에 접속할 때 사용한 쿠키, 호출 가능한 API 등을 사용해서 대상을 분석하게 됩니다. 분석한 결과, 발견된 취약점으로 인해 웹 애플리케이션의 중요 정보가 유출되거나 웹 앱 서버가 탈취되어 정보가 조작될 수도 있습니다. 웹 취약점 분석으로 확인한 문제는 소스 코드 분석이나 컴포넌트 분석에 비해 즉각적인 영향을 미치기 때문에 신속한 조치가 필요합니다.

지속적인 모니터링

만약 정기적으로 분석을 수행해야 하는 경우 분석 예약을 사용하면 특정 시간이나 특정 API의 엔드포인트가 호출되는 시점에 분석을 수행할 수 있습니다. 특히, 웹 사이트 URL 혹은 Git 저장소의 경우 변경된 버전으로 검사를 수행할 수 있기 때문에 지속적인 모니터링이 가능합니다.

이미 분석된 결과의 경우 보안 정보가 업데이트되면서 기존 분석에서 식별한 컴포넌트로부터 예전에는 발견되지 않았던 새로운 취약점이 알려지는 경우가 있습니다. Sparrow Enterprise는 새로운 컴포넌트 취약점이 발견되었다는 사실을 사용자에게 알려주고 해당 취약점을 이슈로 등록할 수 있습니다.

요구사항 2 : 개발 환경 통합

Sparrow Enterprise는 애플리케이션 개발 과정에서 DevSecOps를 구현하는 것을 목표로 설계되었습니다. 애플리케이션 보안을 개발/운영 프로세스 전반에 융합한 형태의 흐름을 만들기 위해서는 다른 개발 환경 관련 서비스와 연동이 핵심입니다. 소프트웨어 개발 방법론의 관점뿐만 아니라 실제 사례에서도 개발 초기에 보안 문제를 빠르게 발견하고 조치함으로써 수정에 드는 시간과 비용이 감소합니다. 시프트 레프트(Shift-Left) 방식의 보안 전략을 세우기 위해 다음과 같은 기능을 활용할 수 있습니다.

SCM(Source Code Management) 시스템 제어

SCM(Source Code Management) 시스템에서 코드가 변경될 때 Sparrow Enterprise의 정책 기준에 따르도록 설정할 수 있습니다. SCM은 애플리케이션 개발 과정에서 코드의 버전을 관리하는 Git 또는 SVN 서버를 의미합니다. 해당 서버에 Sparrow Enterprise를 연결하고 활성화하면 새로 변경된 코드와 함께 푸시를 시도할 때마다 자동으로 분석하게 됩니다. 여기서 Sparrow Enterprise에 통과 기준을 설정하면 분석 결과가 기준을 넘지 못하는 경우 변경된 코드는 푸시에 실패하도록 설계되어 있습니다.

IDE 플러그인 지원 및 개발 프로젝트 분석

개발 환경에서 널리 사용되는 IDE(Integrated Development Environment) 플러그인을 통해 Sparrow Enterprise 서버나 클라이언트에 따로 접속하지 않더라도 IDE 환경에서 분석을 수행할 수 있습니다. Eclipse, IntelliJ, Visual Studio, Visual Studio Code 등에서 플러그인을 설치하면 웹뷰 형태의 UI를 통해 분석 및 결과 확인 등 간단한 기능을 직접 사용할 수 있습니다. 플러그인을 설치하지 않은 경우 Sparrow Enterprise에서 직접 Eclipse 프로젝트 또는 Visual Studio 프로젝트를 선택하는 방법으로도 분석을 수행할 수 있습니다.

Git 저장소 분석

앞서 설명한 SCM 시스템 제어 기능을 설정해서 Git 서버를 직접 제어하지 않더라도 Git 저장소 URL, 사용자 ID 및 토큰을 입력함으로써 Git 저장소의 최신 커밋을 수시로 분석할 수 있습니다. 이 기능을 통해 Git 서버, GitHub, GitLab과 같은 외부 저장소를 사용하는 경우에도 Sparrow Enterprise와 연결하여 보안약점을 분석하고 대응할 수 있습니다.

요구사항 3 : 리스크 기반 우선순위 부여

테스팅을 통해 소스 코드의 잠재적 보안 약점, 웹 애플리케이션의 보안 취약점 또는 오픈소스 컴포넌트의 알려진 취약점 등을 분석하면 분석 대상의 크기에 따라 적게는 수십 개에서 수십만 개의 이슈가 검출됩니다. 보안 테스팅 도구를 활용하는데에 있어 직면하는 대표적인 어려움이 바로 식별된 수많은 보안 취약점에 대한 체계적인 조치 계획을 수립하고 실질적인 해결 방안을 마련하는 일입니다. Sparrow Enterprise는 검출된 이슈 중 사용자가 빠르게 처리해야 하는 이슈를 선별할 수 있도록 우선순위를 부여하기 위해서 다음과 같은 평가 기준을 제공합니다.

이슈 위험도

보안 담당자는 보안 이슈의 위험성이 중대할수록 문제를 신속하게 처리해야 합니다. 이슈를 조치할 우선순위를 결정하기 위한 평가 기준으로 Sparrow Enterprise는 이슈의 위험도를 매우 높음, 높음, 보통, 낮음, 매우 낮음이라는 다섯 단계로 구분합니다.

위험도의 단계는 기본적으로 발생 방법 및 결과의 영향도에 따라 나누어집니다. 특정 이슈의 공격으로 인해 발생할 수 있는 결함이나 영향의 정도에 따라 문제가 시급하고 위험하다고 판단되는 경우 위험도를 매우 높음 및 높음으로 표시합니다. 만약 공격으로 인해 일반적인 오류가 발생하거나 보안상 다른 공격으로 이어질 진입 루트가 될 수 있다면 위험도는 보통 및 낮음으로 분류됩니다. 품질이나 보안에 영향을 미치지 않는 경우에는 위험도가 매우 낮다고 판단합니다.

위험도를 결정하는 두 번째 기준은 실제 발생한 보안 공격 정보에 기반합니다. Sparrow 개발센터에서는 최근 5년 동안 발생한 CVE(Common Vulnerabilities and Exposures) 정보를 CWE(Common Weakness Enumeration)에 매핑합니다. 여기서 유추할 수 있는 평균적인 CVSS(Common Vulnerability Scoring System) 점수와 발생 횟수 등으로 산출된 점수를 기준으로 위험도를 지정합니다. 이 경우 특정 이슈의 CVSS 점수가 높거나 최근 5년간 자주 발생한 경우 위험도가 매우 높음으로 평가되고 보안과 관련 없는 품질 문제 혹은 CWE에 매핑이 되지 않은 이슈인 경우 위험도가 매우 낮게 평가됩니다.

신뢰 지수

보안 테스팅 도구는 분석 대상에 포함된 세밀한 위협도 검출하게 됩니다. 따라서 이슈의 위험도와 함께 검출된 이슈가 얼마나 신뢰할 만한지를 바탕으로 우선순위를 결정할 필요가 있습니다. 신뢰 지수는 이슈가 검출된 과정에서 직간접적으로 영향을 주는 정보를 바탕으로 평가된 점수입니다. 예를 들어, 소스코드 분석에서는 이슈 검출 과정의 추정이나 가정이 적고 값의 범위가 명확할수록 신뢰 지수가 높아집니다. 컴포넌트 분석에서는 검출된 이슈에 연관된 컴포넌트에 대한 정보가 정확할수록 높은 신뢰 지수를 표시합니다. 웹 취약점 분석에서는 검출된 이슈가 발생할 가능성이 높을수록 해당 항목의 신뢰 지수는 자연스럽게 높아집니다. 신뢰 지수로 이슈의 우선순위를 세운다면 이슈 중에서도 중요한 항목을 우선적으로 검토함으로써 효율적인 대응 전략을 수립할 수 있습니다.

적용 사례 및 방법

일반적으로 소프트웨어 개발 환경에서 Sparrow Enterprise를 통해 ASPM를 강화하려면 먼저 다음과 같은 설정이 필요합니다.

1. Sparrow Enterprise와 개발 서버 연결
   a. Dev 브랜치로 코드 변경 전 분석 수행
   b. 코드 변경을 허용하는 기준 설정
   c. Release 브랜치에 머지 발생 시 분석 예약 API 설정
2. 개발자 로컬에 IDE 플러그인 설치
   a. Dev 브랜치에서 Release 브랜치로 머지를 시도할 때 위에서 수행한 분석 결과 확인
3. Sparrow Enterprise에 LDAP 인증 등으로 계정 생성
   a. 다수의 사용자가 분석 결과 확인
4. 빌드된 서비스의 테스트 서버 URL 또는 API 테스트 실행

베스트 시나리오

개발 서버에서 신규 Release 브랜치가 생성되면 Sparrow Enterprise에 해당 릴리즈의 프로젝트가 생성되도록 설정합니다. 개발자가 로컬에서 작성한 코드를 개발 서버의 Dev 브랜치로 푸시를 시도합니다. 개발 서버에 코드 변경 요청이 발생한 경우 Sparrow Enterprise의 형상 관리 API가 호출되어 해당 코드를 대상으로 새로 생성된 프로젝트에서 수시 분석이 수행됩니다. 이 때 수행되는 분석은 소스코드 분석 및 오픈소스 분석입니다. 미리 설정한 기준에 맞는 경우 푸시는 성공하지만 기준을 통과하지 못하는 경우 푸시에 실패하게 됩니다. 코드를 푸시하지 못한 경우 개발자는 IDE 플러그인을 통해 LDAP 인증 계정으로 분석 결과를 확인합니다. 개발자는 직접 본인의 소스코드를 검토하고 수정한 후 다시 푸시를 시도합니다. 이런 방식으로 모든 개발자가 Dev 브랜치에 소스코드를 업로드하게 됩니다.

개발이 마무리되면 Dev 브랜치의 변경된 코드를 Release 브랜치에 머지하고 빌드하게 됩니다. 이 경우, Sparrow Enterprise의 분석 예약 API를 호출하여 전수 분석을 수행하도록 합니다. 그러면 개발자가 개별적으로 수행한 수시 분석에서는 발견하지 못한 연관된 이슈가 검출될 가능성이 있습니다. 여기서도 소스코드 분석 및 오픈소스 분석이 수행됩니다. 보안 관리자는 전수 분석 결과를 확인하고 발견된 이슈의 조치 기한 및 방법을 설정한 다음, 이슈를 처리할 담당자를 지정합니다. 이 과정에서 분석이 불필요한 파일이 있다면 해당 파일을 분석 자산에서 제외할 수 있습니다. 제외된 파일 자산에서 검출된 내용은 해당 프로젝트에서는 더 이상 이슈로 검출되지 않게 됩니다.

Release 브랜치에 머지된 소스코드의 보안 분석 및 조치를 완료한 후에는 해당 브랜치에서 빌드한 패키지를 테스트 서버에 올려서 서비스가 제대로 동작하는지 기능 테스트를 수행하게 됩니다. Sparrow Enterprise도 웹 취약점 분석을 통해 실행 중인 애플리케이션 서비스의 취약점을 분석합니다. 해당 애플리케이션의 접근 주소를 입력해야 하지만, 만약 서비스의 일부만 확인하려는 경우 OpenAPI 명세를 업로드해서 해당 API에서 접근할 수 있는 부분만 검사할 수도 있습니다. 분석에서 서비스에 포함된 Java 파일이 발견되는 경우 해당 파일에 대한 소스코드 분석 및 컴포넌트 분석도 동시에 수행됩니다. 이렇게 발견된 분석 결과를 보안 관리자가 확인하고 앞서 전수 분석과 동일하게 이슈의 조치 기한 및 방법을 설정한 다음, 이슈 담당자를 지정합니다.

검출된 이슈를 해결하기 위해서는 이런 프로세스가 계속 반복되어야 합니다. 개발자는 이슈를 수정하기 위해 Dev 브랜치에 수정된 코드를 업로드하고, 다시 해당 코드가 Release 브랜치에 머지되어, 서비스에 문제가 없는지 동작을 테스트해야 하기 때문입니다. 이러한 과정을 거쳐 보다 안전하고 안정적인 소프트웨어가 만들어집니다.

더 편리한 애플리케이션 보안 관리

애플리케이션에 포함된 취약점이 무엇인지를 확인하는 작업만으로는 실제적으로 발생하고 있는 애플리케이션 보안 이슈에 충분히 대응할 수 없습니다. 적어도 취약점의 특성과 시스템의 구조적 특징, 포함된 자산의 현황 등 복합적인 변수들을 동시에 고려하는 역량이 필요합니다. 하지만 수많은 보안 취약점을 다각도로 분석하고 일관성 있게 결정을 내리는 과정은 경험이 풍부한 보안 전문가에게도 결코 쉬운 일이 아닙니다. 스패로우는 이러한 난제를 해결하기 위해 AI 기술을 활용한 접근 방법을 솔루션에 적용하기 위해 연구 개발을 진행하고 있습니다.

AI 기술을 접목하면 시스템의 고유한 특성 및 발견된 보안 취약점을 종합적으로 분석하여 최적화된 조치 방안을 자동으로 제안할 수 있습니다. 즉, 보안 취약점에 대해 구체적이고 실행 가능한 조치 방법을 상세히 안내하여 개발자와 보안 담당자가 즉시 실무에 적용할 수 있도록 합니다. 궁극적으로 기존의 알고리즘 기반의 우선순위 설정을 벗어나 시스템 환경과 비즈니스 영향도 등 다양한 변수를 종합적으로 고려함으로써 훨씬 정교하고 개별 사용자의 상황에 맞는 조치를 취할 수 있는 혁신적인 변화가 일어날 것으로 기대합니다.