Red Hat Developer 발행

개발자 가이드: DevSecOps의 공급망 보안 설정

소프트웨어 개발 초기에 보안을 구축하는 5가지 단계

Collin Chau, Dash Copeland, Markus Eisele

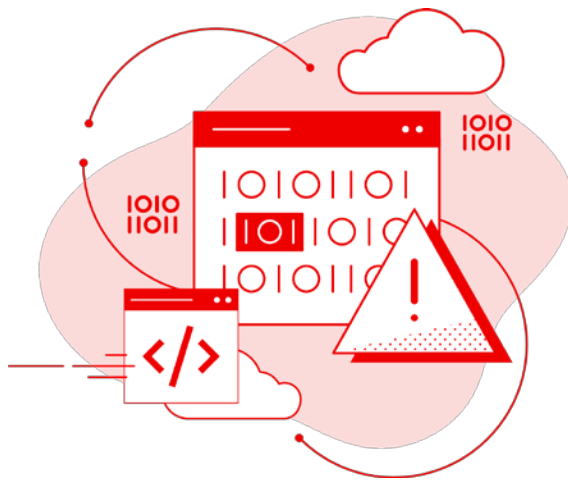
개발자 가이드: DevSecOps의 공급망 보안 설정

소프트웨어 개발 초기에 보안을 구축하는 5가지 단계

DevSecOps는 소프트웨어 개발 분야에서 상대적으로 새로운 용어로, 소프트웨어 개발 프로세스에 보안 사례를 통합하는 작업의 중요성을 인식하는 조직이 점점 더 늘어남에 따라 인기를 얻고 있습니다. DevSecOps는 개발 팀과 운영 팀 간의 협업과 자동화를 강조하는 DevOps 원칙과 보안 사례를 결합하여 소프트웨어 개발 주기 내에서 보안 문화를 형성합니다.

개발자는 소프트웨어를 구성하는 코드 작성을 담당하기 때문에 DevSecOps 사례를 구현하는 데 중요한 역할을 합니다. 그러나 대부분의 개발자는 강력한 보안 배경 지식을 갖추고 있지 않으며 보안 소프트웨어 구축을 위한 모범 사례를 잘 모를 수 있습니다.

이 가이드에서는 개발자에게 DevSecOps 팀의 일부로 보안 소프트웨어를 구축하기 위해 알아야 할 핵심 원칙, 툴, 기술을 포함하여 DevSecOps에 대한 소개를 제공합니다.



개발자 관점의 실질적인 보안

소프트웨어 개발자는 소프트웨어 보안을 중요하게 생각하며 자신이 개발하는 소프트웨어의 보안을 보장하는 정책의 중요성을 알고 있습니다. 2023년 Red Hat의 내부 설문조사에 따르면, 소프트웨어 개발자의 성과가 가장 저조한 분야는 '보안 취약점이 포함된 애플리케이션을 배포할 가능성을 최소화'하는 것이었습니다.

그러나 소프트웨어를 위협으로부터 안전하게 보호하기 위해서는 더 복잡한 개발 프로세스가 필요하고, 이는 이미 업무 부담이 과중한 개발자들에게 더 부담이 될 수 있습니다. 개발자는 잠재적인 취약점을 지속적으로 인식하고 입력 정보 검증, 디지털 서명, 데이터 암호화, 액세스 제어와 같은 보안 코딩 사례를 구현해야 합니다. 이러한 노력은 대부분 시간이 많이 소요되고 개발 프로세스 속도를 저하시킬 수 있기 때문에 소프트웨어를 신속하게 제공해야 할 때 어려움을 겪을 수 있습니다.

대부분의 개발자는 강력한 보안 배경 지식이 없으며 최신 보안 위협과 모범 사례에 익숙하지 않을 수 있습니다. 따라서 효과적인 보안 조치를 구현하기 어렵고 보안 위반 위험이 증가할 수 있습니다.

또한 소프트웨어 보안은 소프트웨어 개발 라이프사이클에서 지속적으로 진행되는 프로세스로, 시간이 지나도 소프트웨어를 안전하게 유지하기 위해 정기적인 업데이트와 유지 관리가 필요합니다. 이러한 유지 관리 작업은 개발자의 바쁜 일정을 더 악화시키고, 번아웃과 직무 불만족으로 이어질 수 있습니다.

소프트웨어 보안은 복잡하고 시간이 많이 소요될 수 있으며, 비즈니스 가치를 창출하고 사용자 경험을 향상하는 기능적인 소프트웨어 개발 작업에 방해가 될 수 있습니다. 그럼에도 불구하고 소프트웨어를 보안 위협으로부터 보호하고 사용자에게 안전하고 신뢰할 수 있는 소프트웨어를 제공해야 합니다. 또한 현대적인 사례에서는 보안 주제가 소스 코드 또는 비기능적 요구 사항에서 그치지 않고 전체 소프트웨어 공급망 전반에 영향을 줍니다.

소프트웨어 공급망 공격에 대처하기

소프트웨어 공급망(Software Supply Chain, SSC)은 소프트웨어를 개발, 사용, 배포하는 프로세스를 나타냅니다. 여기에는 보안 위험에 취약하게 만드는 수많은 당사자, 소프트웨어 구성 요소, 종속성이 포함될 수 있습니다.

타사 라이브러리를 사용하거나 취약점이 있는 자체 라이브러리를 배포하면 금전적 손실, 평판 손상, 법적 책임을 비롯해 고객 신뢰를 잃을 수 있는 광범위한 결과를 초래할 수 있습니다. 프로젝트가 시작될 때 프로젝트에서 사용하는 소스 코드와 전이 종속성을 직접 제어, 감사, 보호하지 않는다면 단순히 보안 개발 사례를 구현하고 확립된 정책을 준수하는 것만으로는 이러한 위험으로부터 보호하기에 충분하지 않습니다.

"개인정보가 침해되거나 브랜드 신뢰도를 잃어 큰 손실을 입을 수 있습니다."

공급망 공격은 감지되지 않을 수 있으며 소프트웨어 생산자와 소비자에게 막대한 영향을 미칠 수 있습니다. 이러한 가시성 부족으로 인해 손상된 코드가 광범위하게 확산되고 취약점이나 악용 사례가 급증할 수 있습니다. 따라서 개발자는 독립적인 애플리케이션 구성 요소를 보호하는 것뿐만 아니라 소프트웨어 팩토리의 모든 디지털 진입점을 잠그고 보호해야 합니다. 소프트웨어 공급망의 한 측면에만 집중한다면 보안을 확장할 수 없고 충분하지도 않습니다.

"사용 중인 애플리케이션과 패키지에 그치지 않고... 포괄적이어야 합니다. 열면 안 되는 포트가 열려 있거나 비공개 항목이 공개되어 있는지도 확인해야 합니다."

Red Hat의 중앙집중식 플랫폼은 개발 및 프로덕션 환경 전반에서 클라우드 네이티브 애플리케이션을 안전하게 보호할 수 있도록 설계되었으며, 보안 및 컴플라이언스 기능이 통합되어 DevOps 팀과 보안 팀이 협력해 DevSecOps 요구 사항을 해결하는 데 도움이 됩니다.

Red Hat은 지난 30년간 신뢰할 수 있는 제품과 패키지를 구축하고 이를 기업이 사용하는 엔터프라이즈 소프트웨어에 안전하게 제공해온 경험을 기반으로, 개발자가 공급망 보안을 파악하고 탐색할 수 있도록 지원합니다. Red Hat은 신뢰할 수 있는 콘텐츠를 구축하고 제공하는 데 사용해온 것과 동일한 소프트웨어 공급망을 개발자와 보안 팀에 제공함으로써 공급망 보안이 소프트웨어 개발 프로세스와 관련된 모든 당사자의 공동 책임임을 인식합니다.

오늘날 보안과 안전한 공급망이 더 중요한 이유

디지털 트랜스포메이션은 끊임없이 지속되고 있으며, 기업 경영진에게 완전히 디지털화된 고객 경험이라는 새로운 요구 사항을 충족해야 하는 책임을 가중시키고 있습니다. 이제 모든 조직이 소프트웨어 중심 기업이 됨에 따라 기술 리더는 클라우드로 이동하여 유연성 및 확장과 같은 새로운 비즈니스 성과를 실현해야 합니다. 그러나 대부분의 조직은 복잡한 하이브리드 IT 환경에서 일관된 보안 및 성능을 유지하는 데 어려움을 겪고 있으며, 소프트웨어 팩토리에 대한 트랜스포메이션 노력이 지체되고 있습니다.

조직은 자금이 풍부하고 계속 커지는 위협 표면을 악용하려는 공격자들에 대항하고 있습니다. 보안 인시던트가 발생할 때마다 사이버 공격을 통해 공격자들이 얻는 이익이 급증하면서 분산 서비스 거부 공격(Distributed Denial-of-service, DDoS), 랜섬웨어, 제로데이 취약점 등이 지속적으로 발생하게 되었습니다. 악의적인 행위자의 사이버 공격 빈도가 급격하게 늘고 정교해짐에 따라 보안 문제에 대처하는 일이 그 어느 때보다 어려워졌습니다.

단절된 커뮤니케이션, 호환되지 않는 인터페이스, 너무 많은 이기종 제품에 대한 표준화 부족으로 인해 애플리케이션 개발 및 배포 시스템 전반에서 보안 효율성이 크게 저하되었습니다. 조직은 DevSecOps 사례를 지원하는 통합 보안 툴링과 프로세스가 필요하며, 그 이유 일부는 다음과 같습니다.

- **보안 개발을 위한 복잡성 증가.** 개발자는 안전하고 규정을 준수하는 코드 작성의 중요성을 이해하지만 현대적인 DevSecOps 중심 환경에서 이러한 문제를 해결하는 작업의 복잡성은 기하급수적으로 증가하고 있습니다. Red Hat의 내부 보안 성과 연구에 따르면, 개발자는 이러한 요구 사항을 충족하는 데 필요한 능력에 점점 더 만족하지 못하고 있습니다.

  "한 위치에서 모든 것이 실제로 최신 상태이고 취약점이 없는지 판단하기는 어렵습니다. Google Cloud 자체가 매우 크기 때문에 이처럼 다양한 제품을 사용하면 관리하기 어려울 수 있습니다. 사용하는 모든 제품에 취약점이 없도록 하는 것이 중요합니다."

- **오픈소스 구성 요소 사용 증가.** 많은 조직에서 오픈소스 구성 요소를 사용해 소프트웨어를 개발하는데, 이러한 구성 요소의 보안은 공급망에 침투하려는 공격자에 의해 손상될 수 있습니다. 기업에는 오픈소스를 바로 사용할 수 있는 준비성이 중요합니다. Red Hat 설문조사에 참여한 조직 3곳 중 2곳은 현재 새로운 애플리케이션을 빌드할 때 내부 개발을 강화하기 위해 오픈소스 소프트웨어(Open Source Software, OSS)를 사용하고 있으며, 나머지 조직도 향후 사용할 계획입니다.[1]

- **매일 발견되는 새로운 공격 벡터.** 조직은 점점 더 타사 툴과 서비스 종속성에 의존하고 있습니다. 이로 인해 벤더가 소프트웨어를 사전에 적절하게 보호하지 않은 경우 본질적인 보안 위험이 발생하고 개발 라이프사이클 초기에 애플리케이션 릴리스가 손상됩니다. 최근 연구에 따르면, 매달 12억 개 이상의 종속성이 다운로드되며, 프로젝트 취약점 7개 중 6개가 전이 종속성에서 비롯됩니다.[2]

  "모든 소프트웨어는 다양한 패키지를 조합하여 구축됩니다. 그리고 이러한 소프트웨어는 다운로드한 수백 가지의 다양한 패키지를 추적하는 또 다른 장소이며, 사용자는 설치의 일부로 해당 소프트웨어에 좋은 보호 장치가 있기를 희망합니다. 그러나 모든 단일 공급업체와 해당 공급업체의 모든 종속성을 일일이 수동으로 검토할 수는 없습니다."

- **더 엄격해진 규제 요구 사항.**[3,4] 엄격한 데이터 거버넌스 및 컴플라이언스가 적용되는 많은 산업에서 조직은 소프트웨어의 모든 코드 종속성 추적을 포함하여 소프트웨어 공급망에서 더 강력한 사이버 복원력을 위한 보안 조치를 구현해야 합니다. 이러한

1 함께하면 더 강력한 DevOps와 오픈소스(Better Together: DevOps and Open Source Go Hand in Hand), IDC Perspective, 2022년.

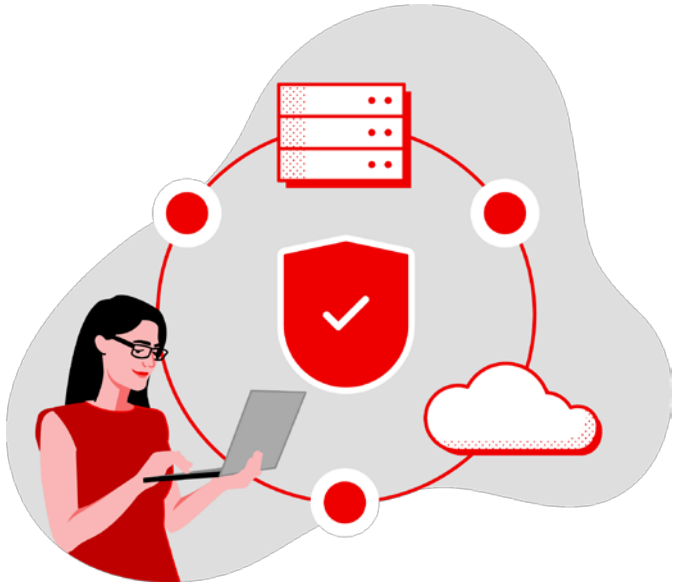
2 제8차 소프트웨어 공급망 현황 연례 보고서(8th Annual State of the Software Supply Chain), Sonatype.

3 국가 사이버 보안 개선에 관한 백악관 행정 명령(EO 14028).

4 유럽연합 집행위원회에서 제안한 사이버복원력법(Cyber Resilience Act, CRA).

조치는 지난 3년에 걸쳐 소프트웨어 공급망 공격이 연평균 742% 증가했다는 놀라운 사실에 따른 것입니다.[5] Gartner는 2025년까지 전 세계 조직의 45%가 소프트웨어 공급망 공격을 경험할 것이라고 예측했으며, 이는 2021년 대비 3배 증가한 수치입니다.[6]

소프트웨어 공급망을 보호함으로써 조직은 재정적 손실, 평판 손상, 최종 사용자 피해를 초래하는 사이버 공격, 데이터 침해 및 기타 보안 인시던트의 위험을 줄일 수 있습니다. 소프트웨어 공급망 보안에 대한 모범 사례를 구현하면 전반적인 소프트웨어 품질과 보안을 개선하고 시스템과 데이터를 공격으로부터 보호할 수 있습니다.



5 Sonatype, 제8차 소프트웨어 공급망 현황 연례 보고서(8th Annual State of the Software Supply Chain).

6 Gartner 선정 2022년 사이버 보안 부문 상위 7개 트렌드(Gartner's 7 Top Trends in Cybersecurity for 2022).

DevSecOps 전략의 중요성

소프트웨어 공급망 보안에 대한 위협으로 인해, 보안이 소프트웨어 개발 라이프사이클에서 기본적이고 지속적인 측면인 DevSecOps 전략을 위해 DevOps 사례가 대대적으로 변화했습니다.[7] 소프트웨어 엔지니어링 리더는 소프트웨어 개발 라이프사이클 초기에 소프트웨어 구성 요소 및 종속성의 보안에 중점을 둠으로써 소프트웨어 공급망의 위험을 완화합니다. 이들은 소프트웨어 팩토리에서 일관되고 반복 가능하며 자동화된 운영을 위해 모든 단계에서 통합 보안 게이트를 시행합니다.

시장은 전반적으로 기업의 경쟁 수단인 탁월한 소프트웨어 경험을 빠르고 안전하며 지속적으로 배포할 수 있는 애플리케이션 플랫폼으로 이동하고 있습니다. 그러나 기업이 이러한 병행 작업을 실행하는 데 어려움을 겪는 경우가 많은 것이 현실입니다. 기업이 직면하는 과제는 다음과 같습니다.

- 레거시 애플리케이션과 인프라를 유지 관리하고 개선하는 일은 복잡하며 이미 제한된 IT 리소스에 부담을 줍니다.
- 현대적인 프레임워크 및 클라우드 네이티브 애플리케이션 아키텍처를 사용하여 새로운 애플리케이션을 빌드하고 실행하면 개발 팀의 업무 부담이 가중됩니다.
- 보안은 대부분 애플리케이션 개발 라이프사이클 종료 시점에 보안 및 IT 운영 팀에서 후순위로 처리하는 경우가 많으며, 이는 애플리케이션 개발 팀 및 기타 팀과 거의 협업하지 않고 이루어집니다.
- 서로 다른 애플리케이션 보안 및 DevOps 툴, 사례, 단절된 프로세스로 인해 툴이 무질서하게 확장되고, 이는 협업, 가시성, 생산성을 저해하고 인적 오류를 증가시킵니다.

결과적으로 조직은 조기에 발견한 경우 해결이 쉽고 비용이 적게 드는 보안 문제를 제때 발견하지 못하는 경우가 많습니다. 이로 인해 보안 위반 위험이 증가하고 애플리케이션 개발 및 제공 속도와 효율성이 떨어집니다.

7 소프트웨어 엔지니어링 리더가 소프트웨어 공급망 보안 위험을 완화하는 방법(How Software Engineering Leaders Can Mitigate Software Supply Chain Security Risks), Gartner, 2021년.

개발자를 위한 DevSecOps 모범 사례

소프트웨어 공급망 보안과 DevSecOps 모두 보안을 우선시하는 소프트웨어 개발의 중요한 접근 방식이지만, 소프트웨어 개발 프로세스에서 서로 다른 측면에 중점을 둡니다.

- **안전한 소프트웨어 공급망**은 공급망을 통해 이동하는 소프트웨어의 보안에 중점을 둡니다.
- **DevSecOps**는 보안 코딩 사례 및 지속적인 모니터링과 보안 개선을 통해 보안 소프트웨어 구축을 강조합니다.

따라서 다른 하나 없이는 완벽한 결과를 얻을 수 없습니다. 개발자는 소프트웨어 자체를 개발하기 위해 DevSecOps 사례를 수용하고 이러한 사례를 기반으로 보안 소프트웨어 공급망을 구축하는 것이 중요합니다. 개발자에게 가장 중요한 모범 사례를 간단히 살펴보겠습니다.

초기에 잦은 빈도로 보안 구현

보안을 처음부터 소프트웨어 개발 프로세스에 통합하고 개발 라이프사이클의 모든 단계에서 고려해야 합니다. 여기에는 보안 테스트, 취약점 검사, 코드 분석, 보안 코딩 사례가 포함됩니다.

"보안 부분에 대한 자동화와 점검은 많지 않습니다. 저는 자동화에 관심이 있습니다. 수동 점검이 가능하지만 문제를 나중에 발견하는 것보다 조기 발견에 도움이 되는 것이 더 효과적이기 때문입니다."

가능한 한 모든 위치에서 보안 자동화

자동화를 구현하면 보안 테스트를 간소화하고 프로세스를 검토하여 효율성과 효과를 높일 수 있습니다. 이러한 예로는 자동화된 취약점 검사, 자동화된 테스트, 지속적인 통합/배포 파이프라인이 있습니다.

개발, 보안 및 운영 팀 간 협업 강조

DevSecOps는 조직 내 여러 부서 간의 협업과 팀워크라는 원칙을 기반으로 합니다. 팀은 협력을 통해 보안 문제를 더 빠르고 효과적으로 식별하고 해결할 수 있습니다.

보안 코딩 사례 활용

입력 검증, 데이터 암호화, 액세스 제어와 같은 보안 코딩 사례는 소프트웨어 취약점을 방지하는 데 도움이 될 수 있습니다. 개발자는 보안 코딩 사례 교육을 받아야 하며 작성하는 코드의 보안을 보장하기 위해 확립된 코딩 표준을 준수해야 합니다.

정기적인 보안 평가 수행

보안 평가를 정기적으로 수행하면 소프트웨어의 잠재적인 취약점과 보안 문제를 식별하는 데 도움이 됩니다. 여기에는 침투 테스트, 코드 검토, 취약점 검사 등의 기술이 포함될 수 있습니다.

지속적인 보안 모니터링 및 개선

보안은 지속적인 프로세스이므로 보안을 유지하기 위해 소프트웨어를 정기적으로 모니터링하고 업데이트해야 합니다. 여기에는 새로운 보안 위협에 대한 모니터링, 필요에 따른 소프트웨어 업데이트, 보안 문제 발생 시 해결이 포함됩니다. 조직은 보안을 지속적으로 모니터링하고 개선함으로써 보안 침해 위험을 줄이고 소프트웨어와 데이터를 보호할 수 있습니다.

DevSecOps의 공급망 보안 설정: 이상적인 환경

소프트웨어 공급망 보안은 효과적인 구현을 위해 포괄적인 DevSecOps 접근 방식이 필요한 복잡하고 다면적인 분야입니다. 성공적인 DevSecOps 사례의 핵심은 기업이 소프트웨어 공급망에서 오픈소스 소프트웨어를 보호하는 방법에 있습니다.

초기에 오픈소스 코드와 해당 타사 종속성을 지속적으로 모니터링하고 테스트하지 않으면 보안 위험을 초래합니다. 개발 팀과 보안 팀은 서비스 인시던트의 우선순위를 정하고 예방하기 위해 필수적인 보안 유지 관리 활동을 추적하고 수행할 수 없습니다.

Red Hat은 기업이 보안 요구 사항을 충족하면서 애플리케이션을 더 신속하게 출시할 수 있도록 혁신 주기에 맞춰 공급망 복원력을 개선합니다. Red Hat Trusted Software Supply Chain은 클라우드 서비스로 제공되며, 신뢰할 수 있는 소프트웨어 공급망을 일관되게 코딩, 구축, 모니터링할 수 있는 DevSecOps 프레임워크를 제공합니다. 단 몇 번의 클릭만으로 소프트웨어 개발 라이프사이클의 모든 단계에 보안 가드레일을 통합하여 애플리케이션에서 오픈소스 소프트웨어 및 타사 종속성 사용을 보호하여 가치 창출 시간을 앞당길 수 있습니다.

개발자는 사용자의 신뢰에 관심을 기울이며,[8] Red Hat은 개발자가 업무 부담 없이 신뢰를 구축할 수 있기를 바랍니다. Red Hat 솔루션은 개발자가 전체 소프트웨어 개발 라이프사이클에 걸쳐 보안을 초기에 통합(shift left)하는 데 도움이 됩니다(그림 1).

그림 1: 코드에서 프로덕션에 이르는 소프트웨어 개발 라이프사이클 초기에 소프트웨어 구성 요소와 종속성의 보안을 유지합니다.

8 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).

성공적인 DevSecOps 사례 지원

성공적인 DevSecOps 구현은 애플리케이션 파이프라인 훨씬 이전에 시작됩니다. 첫 번째 단계로, 조직은 기본 인프라 및 애플리케이션 서비스가 빌트인 보안 툴 및 기능으로 사전 강화된 엔터프라이즈 오픈소스 기반에서 실행되도록 보장하고자 할 것입니다. Red Hat 플랫폼은 취약점을 신속하게 모니터링, 식별, 해결하는 전담 제품 보안 팀에서 관리합니다. 보안이 강화된 Linux 컨테이너에 보안 업데이트가 지속적으로 제공되고 보안 채널을 통해 배포됩니다.

개발자는 클라우드 기반 애플리케이션의 모든 측면에 대한 보안 검사와 지침이 필요합니다. 소프트웨어 패키지 외에도 툴링, 애플리케이션 구성, 그리고 인프라를 포함한 전체 솔루션 아키텍처에 대한 보안 솔루션이 필요합니다.

또한 개발자는 오픈 하이브리드 클라우드에 대한 조직의 요구 사항에 맞게 사용 옵션에 가장 적합한 임의의 풋프린트로 워크로드를 이동할 수 있는 유연성이 필요합니다. 신뢰할 수 있고 업계에서 입증된 컨테이너 오케스트레이션 플랫폼에 구축하면 표준과 일관성의 장점을 활용하여 Quarkus와 같은 쿠버네티스 네이티브 Java 프레임워크 등에 계속 투자할 수 있습니다.

원활한 개발 경험을 위한 툴, 라이브러리, 확장 프로그램을 조합하면 현대적인 클라우드 네이티브 환경에 맞는 애플리케이션을 규모에 따라 안전하게 생성할 수 있습니다. 팀은 널리 사용되는 통합 개발 환경(Integrated Development Environment, IDE) 툴을 기반으로 로컬 머신의 중앙집중식 개발자 워크스페이스를 통해 쿠버네티스에서 몇 분 만에 코드를 제공할 수 있습니다. 소프트웨어 팩토리 전반에서 사용되는 것과 동일한 SSO(Single Sign-On) 툴을 적용하여 개발 워크스페이스와 소스 코드 액세스를 보호하세요.

1 **즉시 사용 가능한 신뢰할 수 있는 이미지 및 라이브러리 활용**
Java, Node.js, Python, Go, Red Hat Enterprise Linux(RHEL) 패키지를 비롯해 널리 사용되는 애플리케이션 프레임워크에서 라이브러리 형태로 신뢰할 수 있는 콘텐츠를 활용하여 최신 취약점과 보안 위험을 제어하세요. 상호 연결된 소프트웨어 팩토리 전반에서 조정을 공유하고 더 손쉬운 협업을

장려하는 모범 사례를 통해 콘텐츠를 표준화하여 팀의 업무 부담을 줄이도록 합니다. Red Hat 모범 사례를 기반으로 이미 검사를 거치고 강화된 소프트웨어 개발 라이프사이클(Software Development Life Cycle, SDLC)을 위해 엄선된 OSS 콘텐츠를 사용하면 위험 프로필을 최소화할 수 있습니다.

개발자는 보안을 위해 올바른 일을 하고 싶어합니다. 이들은 전문적인 우수성에 관심을 갖고 있지만, 그보다 사용자의 신뢰를 얻고 유지하는 것을 더 중요하게 생각합니다. 활발하게 유지 관리되고 조직의 보안 표준을 충족하는 강력한 타사 소프트웨어 라이브러리, 프레임워크, API, 툴에 대한 액세스를 확보하면 개발자는 모든 종속성과 툴을 평가하고 보안을 유지할 필요없이 새로운 기능 구축에 집중할 수 있습니다.[9]

2 입증되고 엄선된 패키지의 안전한 액세스와 통합이 가능한 고가용성 컨테이너 레지스트리 유지 관리

세분화된 역할 기반 액세스 제어(RBAC)를 사용하여 컨테이너 레지스트리와 그 안에 저장된 이미지에 대한 액세스를 제한하여 무단 액세스의 위험을 줄이세요. 애플리케이션과 서비스 배포에 사용되는 이미지를 안전하게 저장하고 관리하여 신뢰할 수 있는 이미지만 프로덕션에서 사용되도록 보장합니다. 루트리스(rootless) 컨테이너 이미지를 실행하여 호스트에 영향을 주지 않고 컨테이너 내에서 안전하게 패키지를 설치하고 서비스를 실행합니다.

소프트웨어 팩토리 전반의 투명성과 가시성을 높여 보안 팀과 DevOps 팀 간의 신뢰를 구축하세요. 검증 및 인증을 위한 이미지 서명을 허용하면 악성 코드가 레지스트리에 추가되는 것을 막을 수 있습니다. 코드 무결성을 보장하기 위해 소프트웨어 빌드 자료의 신뢰성을 확인하고 변조를 차단하도록 합니다. 신뢰할 수 있는 소스에서 제공하는 소프트웨어 구성 요소의 출처를 증명하는 디지털 서명 및 인증서 사용을 지원합니다. 위조 및 악의적인 수정 위험을 줄이는 신뢰 체인에 관한 소프트웨어 아티팩트 공급망 수준(Supply-Chain Levels for Software Artifact, SLSA) 표준을 사용하여 빌드 및 파이프라인 모두의 소프트웨어 출처를 표시합니다.

9 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).

개발자가 보안 문제를 해결하기 위해 시스템에 액세스해야 하는 경우가 있다는 점을 고려해야 합니다. 대규모 조직에서는 권한 있는 액세스를 실행하기 위해 RBAC 및 GitOps와 같은 신뢰 게이팅 메커니즘을 마련해야 합니다. 나아가 모든 권한 있는 세션에 대한 감사 가능한 기록이 필요합니다.[10]

3 보안 모범 사례를 통해 코드 관리에서 소스 코드 및 종속성 보호

잠재적인 취약점, 맬웨어 또는 기타 악성 코드가 소프트웨어 팩토리 전반에서 사용되기 전에 분석하고 감지하세요. 이미지가 코드 리포지토리에 커밋되기 전에 자동 코드 분석을 사용하여 이미지의 잠재적인 보안 취약점과 기타 보안 문제를 검사합니다. 종속성을 주의 깊게 관리하고 빌드 프로세스에 사용되는 모든 라이브러리나 구성 요소에 취약점이 있는지 정기적으로 감사해야 합니다. 구성 요소 분석은 조직이 소프트웨어 공급망에서 타사 구성 요소의 위험을 식별하고 평가하는 데 도움이 됩니다.

많은 개발자는 심각한 취약점이 포함된 어떠한 요소도 배포해서는 안 된다고 믿기 때문에 심각한 취약점을 초기에 발견하여 프로덕션 환경에 배포할 수 있는지 여부를 결정하는 것이 중요합니다. 개발자는 더 적극적으로 대응하여 초기에 취약점을 발견하기를 원하지만, 동시에 코드 작성과 신속한 제공에도 집중해야 합니다. 보안 점검 및 수정을 자동화할 수 있다면 취약점을 수동으로 추적할 필요가 없으며 보안 모범 사례를 쉽게 준수할 수 있습니다.[11]

보편적인 암호화 서명을 통해 코드의 변조를 방지하고 모든 활동에 대한 공개적이고 변경 불가능한 오픈소스 로그를 통해 모든 제출을 자동으로 확인합니다. 코드와 아티팩트가 강력한 Identity 및 액세스 관리(Identity and Access Management, IAM) 정책으로 보호되는 버전 제어 시스템에 안전하게 저장되도록 보장합니다. 암호화 및 보안 백업 시스템을 구현하여 정기적으로 취약점을 감사하고 외부 위협으로부터 보호하는 리포지토리에서 소스 코드를 안전하게 유지하세요.

10 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).
11 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).

4 자동화된 신뢰 및 승인 게이트 체인으로 CI/CD 파이프라인 강화

소프트웨어 종속성의 흐름을 제어하고, 신뢰할 수 있는 패키지만 빌드 및 배포에 사용되도록 보장하여 소프트웨어 팩토리에서 보안 위협이 있는 파이프라인이 실행되지 않도록 합니다. 각 아티팩트 구축 방법에 대한 메타데이터를 사용하여 소프트웨어 구성 요소 목록(Software Bill of Material, SBOM)을 먼저 자동 생성함으로써 빌드를 구성하는 다양한 소프트웨어 구성 요소의 사용을 관리하고 보호하세요. 개발 프로세스에 사용되는 모든 소프트웨어 구성 요소의 버전 제어, 감사, 추적 기능을 통해 산업 표준에 따라 출처를 인증합니다.

팀에서 코드 컴파일링, 이미지 빌드, 테스트 실행 시 모든 입력과 출력이 안전한지 확인하도록 빌드 프로세스 전반에 정기적인 보안 점검을 통합하여 CI/CD 파이프라인을 자동화합니다. 교차 빌드 손상을 통한 변조를 방지하는 강력한 보호 조치를 시행하세요. 리포지토리에 저장된 빌드 아티팩트에 영향을 미치는 소스 코드와 OSS 종속성에 대한 변경 사항 또는 무단 수정을 즉시 감지하고 경고해야 합니다. 특정 애플리케이션에 사용된 구성 요소 버전을 확인하고 해당 변경 사항의 영향을 파악하여 SDLC의 위험을 완화하세요.

조직이 소프트웨어 공급망 내에서 소프트웨어 구성 요소의 허용 가능한 사용 및 동작을 규정하는 엔터프라이즈 계약을 정의하고 시행할 수 있도록 하세요. 구성 오류를 자동 감지하고 배포를 중지 및 롤백하는 코드형 정책(Policy as Code, PaC)으로 릴리스 정책을 적용하여 선언적 상태로 지속적 배포를 구현합니다. 바이너리 분석을 포함한 보안 검사를 지속적으로 구현하여 컴파일된 코드와 이미지에서 일반적인 취약점 및 노출(Common Vulnerabilities and Exposures, CVE)과 바이러스 관련 취약점을 검사하고 식별합니다. 이러한 보안 격차가 악용되지 않도록 자동 수정 작업을 트리거하세요.

개발자는 애플리케이션 보안을 보장하기 위해 종속성을 자동으로 추적하고 유지 관리하는 포괄적이고 정확한 인벤토리가 필요합니다. 종속성 관리 규모 때문에 수동으로 수행하는 것은 불가능합니다. 애플리케이션은 수백 개의 다양한 패키지로 구성되며 각 패키지는 해당 종속성을 통해 보안

취약점을 초래할 수 있습니다. 이러한 종속성으로 인해 개발자가 필수 패키지 업데이트를 설치하지 못하는 경우가 많습니다. 따라서 개발자는 종속성을 추적하고 최신 상태를 유지하기 위해 도움이 필요합니다.[12]

5 배포된 워크로드에 대한 취약점과 위협에 대한 상황별 인사이트를 통해 런타임에서 애플리케이션 모니터링

적절한 액세스 제어, 위협 방지 및 이상 감지, 네트워크 세분화, 런타임 취약점 감지를 구현하여 런타임에서 배포 환경의 안전을 보장하세요. 모든 구성 요소와 해당 소스에 대한 완벽한 엔드 투 엔드 가시성을 제공하여 악성 구성 요소로 인해 발생하는 위험 프로필의 변화를 지속적으로 모니터링하고 사전에 식별해야 합니다. 잠재적인 보안 인시던트를 즉시 감지하고 경고를 발생시켜 조치를 취하도록 지시하는 모니터링 시스템과 로깅 시스템을 구현하세요. 여기에는 조직이 위험 상태를 이해하고 정보에 기반한 결정을 내리는 데 도움이 되는 상세 리포트와 분석이 포함됩니다.

개발자는 영향을 받는 패키지가 사용되는 방식에 따라 보안 취약점이 미치는 영향을 판단해야 합니다. 심각도만으로는 결정을 내리는 데 충분하지 않습니다. 실제 중요성을 결정하기 위해서는 패키지 사용 방법과 애플리케이션 배포 환경에 대한 상황별 정보가 필요합니다. 개발자는 애플리케이션의 전체 아키텍처에 걸쳐 취약점을 쉽게 추적하고 관리할 수 있는 단일 위치가 필요합니다.[13]

소프트웨어 팩토리에 대한 보안 평가 및 연방 정부의 사이버 보안 명령을 충족하는 컴플라이언스 및 감사를 지원하세요. 서비스를 연결하고 가용성 영역 전반에서 올바른 사용자에게 데이터를 전송할 때 API 보안을 강화하는 조치를 취해야 합니다. 이와 동시에, 유휴 상태, 전송 중 또는 런타임에서 데이터 무결성과 기밀성을 지속적으로 보호해야 합니다. 배포 중에는 API 관리, 암호화된 통신, 보안 인증과 같은 보안 기술을 사용하여 중요한 데이터 및 시스템에 대한 무단 액세스를 차단합니다.

12 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).
13 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).

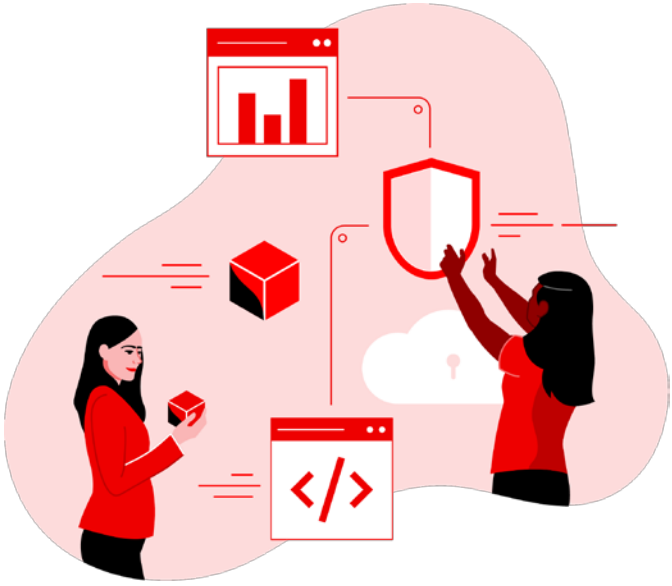
Red Hat이 보안 중심의 클라우드 네이티브 개발을 지원하는 방법

온프레미스 또는 하이브리드 환경과 멀티 클라우드 환경 전반에서 소프트웨어 공급망 보안 전략을 지원하는 오픈소스 소프트웨어를 사용하세요. 소프트웨어 공급망 보안 전략을 위한 Red Hat의 DevSecOps 접근 방식은 다음과 같은 장점을 제공합니다.

- **통합된 보안 툴 에코시스템을 활용하여 소프트웨어 공급망과 관련된 잠재적 위험을 식별하고 평가합니다.** 통합된 셀프 서비스 포털에서 표준화되어 사용되는 엄선된 서비스를 통합할 수 있습니다. Red Hat은 30년에 걸친 엔터프라이즈급 Linux 경험과 7년의 엔터프라이즈급 쿠버네티스 경험을 바탕으로 한 보안 솔루션은 물론, 운영 보안 요구 사항을 유지 관리하면서도 신속한 혁신에 도움이 되는 DevSecOps 툴을 통해 소프트웨어 공급망을 보호하는 사전 예방적 접근 방식을 제공합니다.
- **보안 소프트웨어 공급망을 통해 고객과 사용자는 사용 중인 소프트웨어에 대한 신뢰를 높일 수 있습니다.** 소프트웨어에 유입되는 취약점 및 위협 발생 위험을 줄이면 고객 충성도와 브랜드 평판이 향상됩니다. 또한 새로운 소프트웨어 기능과 업데이트를 더 빠르게 출시하여 변화하는 고객 선호도에 발맞출 수 있습니다.
- **소프트웨어 공급망 보안 솔루션을 구현하여 산업 규정 및 표준에 대한 컴플라이언스를 개선합니다.** Red Hat은 조직이 규정 미준수로 인한 높은 벌금과 처벌을 피하는 동시에 소프트웨어의 전반적인 품질을 개선하도록 지원할 수 있습니다. 이를 통해 안정적이고 신뢰할 수 있는 소프트웨어를 제공하여 보안 문제를 미연에 방지할 수 있습니다.
- **개발자는 클라우드 기반 애플리케이션의 모든 측면에 대한 보안 검사와 지침이 필요합니다.** 소프트웨어 패키지 외에도 툴링, 애플리케이션 구성, 그리고 인프라를 포함한 전체 솔루션 아키텍처 전반에 대한 포괄적인 보안 솔루션이 필요합니다. 이와 더불어 데이터 정확성에 대한 높은 신뢰도가 필요합니다. 종합적인 보안 전략을 위해 다양한 보안 데이터 소스와 교차 참조해야 합니다.[14]

14 개발자 보안 성과 디스커버리 리포트(Red Hat, 2023년 1월).

Red Hat은 파트너와 협력하여 포괄적인 DevSecOps 에코시스템을 구축하는 툴과 서비스는 물론, 오픈 하이브리드 클라우드 전반에서 보안 중심 애플리케이션을 빌드, 배포, 실행하는 데 필요한 강력한 포트폴리오를 제공하는 전문성과 능력도 제공합니다. 이를 통해 프로세스를 개선하고, 보안을 저해하지 않고 애플리케이션 개발을 가속화하며, 협업 문화 조성과 위험 감소를 실현할 수 있습니다.



추가 정보

Red Hat Developer에서 소프트웨어 공급망 보안 전략을 지원하는 데 도움이 되는 튜토리얼, e-book, 기타 학습 리소스를 살펴보세요.

- DevOps
- DevSecOps
- GitOps
- 보안 코딩
- 소프트웨어 공급망 보안
- Red Hat OpenShift를 위한 개발자 샌드박스

작성자 소개

Collin Chau는 민첩한 개발자가 고품질의 디지털 경험을 계획, 코딩, 테스트, 빌드, 배포할 수 있도록 돕는 일을 좋아합니다. 지속적인 테스트 및 애플리케이션 릴리스 자동화를 위해 DevOps 팀을 확장하는 동시에, 프로덕션 환경에서 SRE를 위해 IT 서비스 상태를 모니터링 및 트러블슈팅하고, ITOps 팀이 하이브리드 클라우드 서비스를 연결 및 브로커링할 수 있도록 지원했습니다.

Dash Copeland는 기술 전문가를 위한 탁월한 경험을 구축하는 데 열정을 가진 사용자 환경 설계자이자 연구원입니다. 엔터프라이즈 제품 분야에서 10년 이상의 경험을 보유하고 있으며, Linux, 컨테이너, 쿠버네티스를 사용해 파트너와 개발자를 위한 디지털 경험을 설계했습니다.

Markus Eisele은 Java 지지자, O'Reilly 저자, 독일 JavaLand 컨퍼런스 창립자로, 전 세계 Java 컨퍼런스 연사로 유명하며 엔터프라이즈급 Java 분야의 유명 인사입니다.

또한 업계에서 16년 이상의 전문 경험을 보유하고 있으며, Fortune 500대 기업을 위한 대규모 엔터프라이즈급 애플리케이션을 설계하고 개발했습니다. 숙련된 팀 리더이자 아키텍트로서 자동차 기업, 금융 및 보험 기업에서 최대 규모의 통합 프로젝트를 구현하는 데 도움을 주었습니다.