NGINX
Part of F5

EBOOK

모던 애플리케이션과 API 보안

개발 단계에서 보호 기능 통합

작성자: NGINX 및 F5, Inc.의 보안 전문가

목차

서문

애플리케이션 보안은 NGINX에서 핵심 우선순위이며, 허용되지 않은 액세스, 데이터 수정 등 보안 위협에 대한 취약점을 제거하는 것이 매우 중요하다고 인식하고 있습니다. 점차 더 많은 애플리케이션이 데이터센터와 같은 보안 경계 내에서 뿐만 아니라 외부 네트워크를 통해서도 액세스할 수 있음에 따라, 애플리케이션의 공격 표면은 확대됐으며, 이를 악용하려는 잠재적인 악성 사용자의 수도 크게 늘어났습니다.

때로 보안은 애플리케이션을 설계 및 구축하는 팀에서는 후순위로 취급됐지만, 개발 프로세스와 CI/CD 파이프라인에 통합되어야 합니다. 개발 단계에서 애플리케이션 보안 정책을 부여하면 애플리케이션이 프로덕션 단계로 넘어갈 때 권한 없는 사용자가 민감한 데이터에 액세스 하거나, 탈취, 수정 또는 삭제할 수 있는 가능성을 최소화할 수 있습니다.

가장 기본적인 소프트웨어 대책은 웹방화벽(WAF)입니다. 웹방화벽은 웹 애플리케이션과 최종 사용자 간의 트래픽을 검사하고 필터링합니다. 웹방화벽은 흔히 정책이라고 일컫는 일련의 규칙을 통해 작동합니다. 이 규칙은 악성 트래픽을 필터링함으로써 애플리케이션의 취약점을 악용하는 것을 방지합니다. 웹방화벽의 가치는 신속하고 손쉽게 정책을 수정하거나 속도 제한을 적용함으로써 DDoS와 같은 공격에 더욱 빠르게 대응할 수 있을 때 그 효과를 더 확인할 수 있습니다.

많은 기업들 내에서 DevOps 팀은 회사의 경쟁력 유지를 위해 신속하게 애플리케이션을 제공하는 데 있어 SecOps가 걸림돌로 작용한다고 보고 있습니다. 그러나, 보안을 개발 프로세스에 통합하면, SecOps가 가이드라인과 정책을 제공하고, DevOps는 적절하다고 판단하면 이를 애플리케이션에 구현할 수 있는 모델을 채택할 수 있습니다. 이러한 "셀프서비스" 모델은 팀 간의 마찰을 최소화하고 개발 속도를 높입니다.

NGINX는 CI/CD 파이프라인에 쉽게 통합되는 모던 애플리케이션을 위한 경량 웹방화벽인 NGINX App Protect(NAP)에서 이 모델을 구현했습니다. NAP를 사용하면 DevOps CI/CD 프로세스에서 중단없이 SecOps 인증 보안을 적용하고, 컨테이너, 마이크로서비스와 같이 분산된 모던 애플리케이션 환경에서 애플리케이션 보안 컨트롤을 배포하고 관리할 수 있습니다.

이 eBook에서는 오늘날 기업들이 직면한 보안 문제를 살펴보고, NGINX 소프트웨어가 이러한 문제 해결에 어떻게 도움을 드리는지를 설명합니다.

라지브 카푸르(Rajiv Kapoor)
NGINX (part of F5)의 수석 제품 마케팅 매니저

1. 어려움없이 애플리케이션 보안을 구현할 수 있습니까?

작성자: 라지브 카푸르(Rajiv Kapoor), NGINX (part of F5)의 수석 제품 마케팅 매니저

애플리케이션 보안은 어렵고, 제대로 수행 하는 것은 더욱 어렵습니다. 쉽고 간단한 일이었다면, 임원들이 침해에 대한 책임을 지고 징역형을 선고받는 일은 없었을 것입니다! 오늘날 디지털 퍼스트(digital-first) 세상에서 사이버보안 위협으로부터 모든 애플리케이션을 보호하는 것은 기업들이 극복해야 하는 가장 시급한 과제 중 하나가 되었습니다.

기업들은 계속해서 공격자들을 저지하기 위해 노력하고 있습니다. 사이버보안 지출은 2020년에 1,230억 달러에 이를 것으로 예상되며, 클라우드 환경의 보안은 글로벌 팬데믹에 따른 경기 침체에도 불구하고 33% 증가할 것으로 예상됩니다.

하지만 결코 쉽지 않은 상황입니다.

- 지난 해 발견된 데이터 침해의 20% 이상이 코드 오류로 발생했으며, 이러한 공격 중 40% 이상은 웹 애플리케이션을 표적으로 삼았습니다.
- 해커는 39초마다 한 번씩, 하루 평균 2,244번 공격을 시도합니다.
- 평균적으로 회사 포트폴리오의 애플리케이션 중 단 5%만 적절하게 보호됩니다.
- 2020년, 평균 데이터 침해 비용은 회사당 386만 달러였습니다.
- 영국 기업의 39%는 팬데믹 기간 동안 보안 침해로 인해 이미 직원들을 해고했습니다.

새로운 보안 침해 사고가 매일 뉴스 헤드라인을 장식하면서, 기업의 리더들은 내부 프로세스에서 보안의 우선 순위를 더 높이기를 원하며, 또 그렇게 해야 합니다.

모두가 이미 애플리케이션 보안에 나서고 있는데, 제대로 수행되는 경우가 거의 없는 이유는 무엇입니까?

모두가 이미 애플리케이션 보안에 나서고 있는데, 제대로 수행되는 경우가 거의 없는 이유는 무엇입니까?

새로운 보안 문제를 야기하는 디지털 디스럽션(DIGITAL DISRUPTION)

지난 10년 동안 애플리케이션 개발은 점점 더 까다로워지는 비즈니스 기대에 따라 엄청난 변화를 겪었습니다. 이제 기업들의 절반 이상은 애플리케이션 없이는 운영이 불가능하다고 얘기합니다. 그 외 67%는 IT 및 비즈니스 프로세스 최적화와 같은 디지털 트랜스포메이션 노력이 신제품과 서비스 출시 속도를 향상시킨다고 믿고 있습니다.

이에 따라, 개발 팀은 클라우드 컴퓨팅과 마이크로서비스와 같은 첨단 기술을 DevOps 원칙과 함께 활용하면서 점차 더 혼잡해지는 시장에서 혁신에 박차를 가하고 경쟁력을 유지하고 있습니다.

그러나 대부분 그렇듯이, 이러한 진보에는 대가가 따릅니다. DevOps의 빠른 개발 속도로 인해 보안 팀은 간신히 이에 맞춰 적절한 보안 가드레일을 설치하고 있습니다. 이전에는 보안 기능의 구현이 릴리스에 앞서 최종 단계에서 이뤄졌지만, 애자일 소프트웨어 개발의 빠른 릴리스 주기와는 이제 더 이상 맞지 않습니다. 예를 들어, Amazon은 지난 2014년 연간 5천만 건의 프로덕션 배포를 달성했으며, 이는 초당 약 1개의 릴리스에 해당하는 것입니다.

개발 속도가 보안 팀이 구성을 확인하고 취약점을 검색할 수 있는 속도를 빠르게 앞지르는 것은 놀라운 일이 아닙니다. 특히 개발자가 현재 보안 전문가보다 500 대 1의 비율로 많다는 점을 고려하면 더욱 그렇습니다.

모던 애플리케이션 환경은 거의 무한대의 공격에 노출되어 있습니다.

모던 애플리케이션 환경은 거의 무한대의 공격 표면에 노출되고 있으며, 기업들 중 87%는 현재 멀티 클라우드 환경이며, 대부분은 여러 애플리케이션 아키텍처와 인프라 전반에 일관된 보안을 제공하기 위해 고군분투하고 있습니다. 기업들은 매번 공격의 중단 또는 침해 또는 공개적인 기업 이미지 훼손으로 수십억 달러의 피해를 감수해야 합니다. 악의적인 공격자들은 단 한 번만 성공하면 승리를 거둘 수 있기 때문 입니다.

지속적으로 가동 상태를 유지하고 안전하게 보호해야 한다는 압박이 가해지면서, DevOps는 점차 DevSecOps로 재구성되고 있습니다. DevSecOps에서는 보안이 "시프트 레프트(shifted left, 근본적인 문제의 해결)"되고, 소프트웨어 개발 라이프사이클 초기에 구현되며, 더 유연해 집니다. 즉, 보안은 프로세스와 툴에 통합되며, 다른 결함과 마찬가지로 문제를 찾아 더욱 효율적으로 해결할 수 있습니다.

안전한 개발 환경을 구축하기 위해 많은 노력을 기울이는 기업들

누군가에게 DevSecOps를 설명해달라고 요청하면, 다음 답변 중 하나를 들을 수 있을 것입니다.

- "DevOps에 보안 업무 부여"
- "코드로서의 보안"
- "모두가 IT 보안에 대한 책임을 가짐"

그러나, 개발 초기 단계에 보안 구현을 통합해야 한다는 필요성을 이해하는 것은 이를 구현하는 것과 매우 다릅니다. 또한 DevSecOps는 실제로 훨씬 더 미묘한 차이를 보이고 있습니다. 대부분 기업들은 DevSecOps의 의도를 이해하고 있지만, 정작 어떻게 해야 할지 확신하지 못하고 있습니다. 실제로 소프트웨어 개발 라이프사이클 전반에서 보안을 완전히 통합하는 비율은 14%에 불과합니다.

최근 연구에 따르면 보안 팀의 65%가 시프트 레프트(shift left, 근본적인 문제 해결)했다고 밝혔지만, 이 주장을 검증하는 데 필요한 스캔을 수행하는 경우는 1/5 미만에 불과합니다. 이 보고서에서는 대부분 보안 팀에 마이크로서비스, API, 클라우드 네이티브/서버리스와 같은 최첨단 애플리케이션 기술을 모니터링하고 보호할 프로세스가 마련되어 있지 않다고 지적하고 있습니다.

이러한 가시성 부족으로 인해 프로덕션 환경에서 문제가 발생해도 보안 조직은 전혀 모를 수도 있습니다. 게다가, 취약점이 개발 주기에서 늦게 발견될수록, 이를 해결하는 데 더 많은 비용이 소요됩니다. IBM System Science Institute의 연구에 따르면 초기 설계 단계에서 확인된 결함에 비해, 구현 단계에 발견한 결함을 수정할 때는 6배, 프로덕션 환경에서 발견한 결함은 100배나 더 많은 비용이 필요한 것으로 나타났습니다.

기업 중 거의 절반이 촉박한 마감 시한에 맞추기 위해 알면서도 취약한 애플리케이션을 배포한다고 인정했다는 것입니다.

더욱 충격적인 것은 기업 중 거의 절반이 촉박한 마감 시한에 맞추기 위해 알면서도 취약한 애플리케이션을 배포한다고 인정했다는 것입니다.

모두가 안전한 애플리케이션을 원하겠지만, 더 빨리, 더 자주 배포하기 위해 규정 준수와 보안 감독이 종종 간과되거나, 심지어 고의적으로 회피하는 경우도 있습니다.

DEVOPS-SECOPS의 분리

가장 중요한 문제 중 하나는 오래되고, 뿌리 깊은 인식에 있습니다.
과거의 사일로화된 환경에서 팀들은 서로 독립적으로 작업했으며, 각 단계를 넘어갈 때는 엄격하게 계획된 이관 절차를 거쳐야 했습니다. 보안 운영 (SecOps) 팀은 종종 개발과 릴리스 프로세스의 최종 단계에서만 보안 기능을 구현했기 때문에 지연이 초래됐습니다. 또한, 팀 간의 협업은 시간이 지남에 따라 애자일 개발 방식을 보다 폭 넓게 수용하도록 개선되어야 했지만, 개발 팀과 보안 팀은 서로 다른 핵심 목표를 지향하면서 종종 조직 간의 불일치와 불가피한 마찰이 발생하고 있습니다.

그림 1: DevOps와 SecOps가 서로를 바라보는 시각

DevOps 팀이 혁신을 가동하는 엔진이라면, SecOps는 종종 전진 주행을 멈추게 만드는 브레이크로 간주됩니다. 48%의 기술 전문가들은 보안이 소프트웨어를 신속하게 제공할 수 있는 능력에 큰 제약이 된다고 믿고 있습니다.

반면, SecOps와 애플리케이션 보안 팀은 취약점, 잠재적 리스크, 가능한 규정 준수 문제를 식별하는 철저한 테스트에 중점을 두고 있습니다. 이들은 특히 애플리케이션 스택에 섀도우 IT – 승인 받지 않은 애플리케이션의 도입, 배포 및 구성 등 – 의 망령이 점차 커지는 상황에서 개발자들이 활개치고 있다고 느끼고 있으며, 개발자들이 전속력으로 밀고 나가는 것을 저지하는 데 어려움을 겪고 있습니다.

DEVOPS와 보안 팀은 모두 회사의 핵심 목표에 기여합니다.

DevSecOps에 "Sec"을 성공적으로 결합하기 위해서는 이러한 오래된 문화적 편견을 바꾸고, 보안을 수용해야 한다는 필요성을 강조하며, 팀들에게 올바른 툴과 자동화 기술을 제공함으로써 전체 조직의 속도를 늦추지 않고 더 현명한 결정을 내릴 수 있도록 해야 합니다. 명백히 다른 우선 순위에도 불구하고 DevOps와 보안 팀은 모두 회사의 핵심 목표인 고품질 제품의 신속한 생산에 기여하고 있습니다. 차이점은 그것이 의미하는 바를 측정하고 정의하는 데 사용되는 방법에 있습니다.

여기서 주목해야 하는 것은 개발이 앞으로도 느려지지 않을 것이라는 점입니다. 개발자의 38%는 이제 매달 또는 그보다 빨리 릴리스하고 있으며, 컨테이너의 54%는 5분 미만이면 실행됩니다. 보안을 종단간 포함시켜야 하는 핵심 기능이 아니라, 별도의 서비스 객체로 간주한다면 프로세스가 느려지고 효율성이 저하될 뿐입니다.

그 어느 때보다 바로 지금 DevOps에 보안이 적용되는 방식을 바꿀 때입니다.

속도와 보안의 격차 해소

애플리케이션 보안에 대한 주요 질문들은 다음과 같습니다. DevOps 팀과 애플리케이션 보안 팀이 보다 쉽게 협업하기 위해서는 무엇이 필요합니까? 기본 내장된 보안은 실제로 어떤 형태로 나타납니까?

DevOps 보안을 개선하려는 기업들은 다음과 같은 몇 가지 핵심 사항에 중점을 두고 있습니다.

최대한 보안 자동화 확대

자동화를 활용해 CI/CD 파이프라인에 직접 내장될 수 있는 보안 솔루션에 투자하십시오. 이를 통해 보안 때문에 개발 속도가 저하되는 일 없이, 애플리케이션을 더욱 쉽게 보호할 수 있습니다. 정적 코드 분석(static code analysis), 동적 분석(dynamic analysis), 펜 테스트(pen testing)와 같은 기술을 채택하면 위험이 줄어들며, 개발자에게 잠재적인 문제에 대해 경고를 보낼 수 있습니다. 모든 보안 문제를 독자적으로 처리할 수 있는 보안 전문가는 충분하지 않으므로, 가능하면 자동화를 활용하십시오.

게이트가 아닌 가드레일로서 보안 구축

보안이 게이트가 아닌 가드레일이 되도록 적절한 지침과 툴을 제공하십시오. 예를 들어, DevOps 팀이 템플릿 정책에 액세스할 수 있도록 하면 개발 프로세스에서 불필요한 지연을 발생시키지 않으면서 처음부터 보안 요구 사항에 맞게 애플리케이션을 조정할 수 있습니다. 애플리케이션과 보안 정책을 CI/CD 파이프라인의 일부로서 테스트할 수 있기 때문에 다른 기능 사양처럼 확인됩니다. 즉, 적절한 보안 컨트롤이 적용된 애플리케이션을 개발하고 테스트하는 데 필요한 모든 것을 개발자에게 제공하는 것이 중요합니다. 그렇지 않으면 개발자들은 이를 간과할 것입니다.

개발자들에게 보안과 셀프서비스 규정 준수에 관해 충분히 이해시키면 기업에 대한 취약점과 위험을 줄일 수 있습니다. 개발자는 항상 더 빨리 진행하기를 원하기 때문에, 안전하게 속도를 낼 수 있도록 해야 합니다.

보다 사전 예방적인 책임을 지원하는 추상 보안(Abstract Security)

일반 개발자들이 모든 최신 보안 동향을 파악하는 데 필요한 전문 지식 수준을 갖추고 있다고는 기대하기 어렵습니다. 프로그래밍 스킬을 최신 상태로 유지하는 것도 상당히 어려운 일이기 때문입니다. CI/CD 피드백 루프 내에서 단순하고 이해하기 쉬운 통찰력을 제공하는 솔루션을 선택함으로써 복잡성을 줄이고 보다 쉽게 개발자의 동참을 이끌어 낼 수 있습니다.

확장성 및 신뢰성이 뛰어난 능동형 보안 구축

모던 분산 환경을 비롯한 모든 환경에 대해 일관되고, 중앙 집중화된 셀프서비스 보안을 제공하는 솔루션을 선택하십시오. 예를 들어 AI 기반 보안 정책 엔진은 CI/CD 방법론에 따른 애플리케이션의 빠른 변화를 지원하는 데 필요한 적응성을 부여하는 한 가지 방법입니다. 최첨단 공격에 맞서 보안 정책을 적용하고 종속 관계를 파악함으로써 위험을 진단하고 보다 신속하게 조치를 취할 수 있습니다.

DEVOPS를 자유롭게 만드는 모던 애플리케이션 보안

오늘날, 통합 보안은 모든 DevOps 구현의 일상적인 부분이 되어야 합니다.

프로세스의 마지막 단계에서 단순히 보안을 끼워 넣던 시대는 지났습니다. 오늘날, 통합 보안은 모든 DevOps 구현의 일상적인 부분이 되어야 합니다. 마찰없이 능동적인 보안이 구현되도록 한다면, 개발 팀은 걱정없이 진행할 수 있습니다. 모던 애플리케이션 보안은 처리해야 하는 골치 아픈 추가 단계가 아니라, 기업들이 비즈니스 목표를 달성할 수 있도록 힘을 실어주고 훨씬 더 높은 수준으로 이끄는 강력한 지원 시스템이 될 수 있습니다.

2. Real-Time API 보안의 중요성

작성자: 라지브 카푸르(Rajiv Kapoor), NGINX (part of F5)의 수석 제품 마케팅 매니저

점차 디지털 전환이 확대되고 고객의 기대도 높아지면서 기업들은 다양한 최종 사용자의 요구와 경험을 지원하기 위해 창의적인 접근 방식을 채택하고 있습니다. 원격 의료에서 온라인 뱅킹에 이르기까지 Real-Time API는 디지털 비즈니스를 구축하는 기반으로서, 애플리케이션 개발자들이 고객의 요구에 부응할 수 있는 애플리케이션을 개발할 수 있도록 합니다.

API에 의해 즉시 지원되는 디지털 세계의 애플리케이션들이 폭발적으로 증가하면서, API 침해를 신속하게 탐지하고 보호해야 한다는 필요성이 대두되고 있습니다.

API는 사용자가 애플리케이션과 상호 작용하는 모든 단계의 핵심 구성 요소입니다.

API는 모던 애플리케이션을 위한 뼈대를 형성합니다. 이는 어디에나 있기 때문에 개발자들은 다른 소프트웨어 구성 요소에서 귀중한 정보를 입수하고 이를 애플리케이션에 통합할 수 있습니다. 예를 들어 Google Maps를 rideshare 앱에 내장시키거나 YouTube 영상을 웹 페이지에 포함시킬 수 있습니다. API는 로그인부터 피드백 작성에 이르기까지 사용자가 애플리케이션과 상호 작용하는 모든 단계의 핵심 구성 요소입니다. 빠른 속도(1Gbps)를 약속하는 5G의 등장은 오늘날 참을성 없는 사용자들이 열악한 앱 성능을 더욱 견디지 못하게 만들 수 있습니다. Real-Time API 응답성(30ms 미만의 전방위 API 호출 처리로 정의)을 달성하지 못하는 API 기반 비즈니스들은 디지털 시장을 잃게 되고, 수익 손실로 인해 디지털 전환 노력이 위태로워질 수 있습니다.

좋은 의도로 만들어진 모든 것처럼 API가 널리 활용되면서 부작용이 나타났습니다. 그것은 바로 나쁜 의도가 있는 공격자들에게는 애플리케이션을 악용할 수 있는 새로운 길을 열어준다는 것입니다. 가트너는 API 남용이 2022년까지 데이터 침해를 초래하는 엔터프라이즈 웹 애플리케이션에 대한 공격에 가장 많이 이용되는 경로가 될 것으로 예측했습니다.

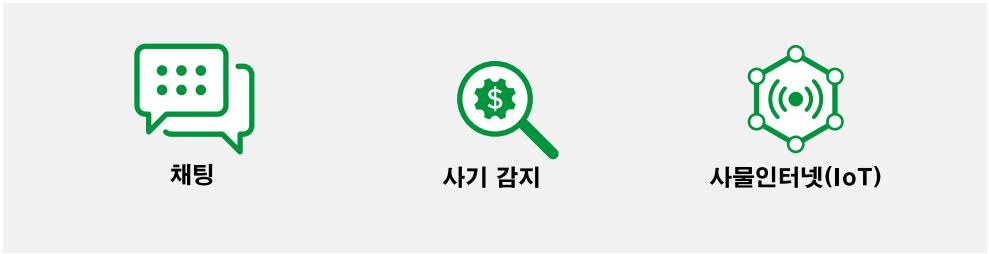


그림 2: Real-Time API 의 세 가지 일상적 활용 사례

API 보안이 중요한 이유

API에 애플리케이션 환경의 모든 데이터에 대한 액세스 권한이 부여됩니다.

여러 요인들로 인해 API는 보안 공격의 중요한 표적이 되고 있습니다. 가장 큰 문제 중 하나는 적절한 액세스 권한을 설정하지 못하는 것입니다. 사용자의 직접적인 액세스를 위한 것이 아니기 때문에, API에 애플리케이션 환경의 모든 데이터에 대한 액세스 권한이 부여됩니다. 이후, API 호출로 변환되는 최초 요청을 한 사용자에게 특정한 권한을 부여하고, API가 해당 권한만 상속하도록 하는 방식으로 액세스를 제어합니다. 하지만, 이러한 방식은 공격자가 사용자 인증 프로세스를 우회하여 API를 통해 직접 다운스트림 애플리케이션에 액세스하면, 무력화 됩니다. API는 무제한 액세스 권한이 있어 공격자들이 모든 것을 볼 수 있기 때문입니다.

기본 HTTP 웹 요청과 마찬가지로 API 호출은 URI, 메소드, 해더 및 기타 매개변수를 통합합니다. 이 모든 것은 공격에 악용될 수 있습니다. 안타깝게도 injection, 크리덴셜 Brute Force, 매개변수 변조, 세션 스누핑(session snooping)과 같은 대부분의 일반적인 웹 공격은 API에서 놀랍도록 강력한.위력을 발휘합니다. 그래서 공격자에게.API는 쉬운 표적입니다.

API를 보호하는 방법

라이프사이클의 모든 단계에서 API에 보안을 구현하는 것이 중요합니다. 설계와 개발 단계에서 엔지니어는 웹방화벽, 봇 보호, API 관리 솔루션, API 게이트웨이 및 기타 툴과의 통합에 필요한 로직를 작성해 개발, 테스트 및 프로덕션 환경에서 API를 안전하게 보호할 수 있어야 합니다.

그림 3: 보안 API의 네 가지 구성 요소

그리고, 다음 섹션에서 설명하는 바와 같이, 아래와 같은 기술을 배포해 제공(delivery) 단계에서 API를 보호합니다.

웹방화벽(WAF)

웹방화벽은 해당 API의 의도된 기능을 실행하지 않고, 공격자들이 정보를 훔치거나 악성 코드를 실행할 수 있는 애플리케이션 코드의 취약점을 악용하는 데 목적을 둔, 사실상 불법적인 요청을 인식합니다. 최소한 웹방화벽은 OWASP API Security Top 10과 같은 가장 일반적인 공격 유형으로부터 보호하는 것이 중요합니다.

NGINX App Protect는 F5의 Advanced WAF 제품의 기술에 기반합니다. CI/CD와 DevOps 워크플로우에 최적화됐으며 XML, JSON, 텍스트, HTML 요청과 응답 페이로드를 지원합니다. 고급 API 보호 프로파일은 파싱과 구문 적용(parsing and structure enforcement), 공격 시그너처(attack signatures), 메서드 적용(method enforcement), 경로 적용(path enforcement)을 통해 공격으로부터 보호합니다.

봇 차단

HTTP API는 봇과 기타 형태의 악의적이거나 원치 않는 자동화 기반 트래픽의 표적이 될 수 있습니다. 이제 F5의 일부인 Shape는 가시성, 스로틀링(throttling), 완화 옵션을 제공하는 솔루션인 API Defense™를 제공해, 온라인 사기와 애플리케이션 악용을 발생시키는 봇 및 기타 자동화된 공격으로부터 HTTP 기반 API를 보호합니다.

API 관리

여러 기능 중에서 API 관리 솔루션은 API 게이트웨이가 API 호출을 처리할 때 적용할 보안 정책을 정의하기 위한 인터페이스를 제공합니다.

NGINX Controller API Management Module에는 API specification을 기반으로 하는 암시적 URI 허용 리스트(allowinglist), 프로그래밍 가능한 속도 제한, 다중 속도 제한 정책, DoS(Denial of Service) 공격으로부터 보호하기 위한 스로틀링(throttling) 기능이 포함되어 있습니다.

API 게이트웨이

NGINX Plus와 같은 API 게이트웨이는 세 가지 주요 기능을 책임지는 관리자 역할을 통해 API 호출을 보호합니다.

그림 4: API 게이트웨이의 주요 기능

1. **인증 및 권한 부여**

   API 인증은 자신이 주장하는 본인이라는 것을 증명할 수 있는 인증된 클라이언트에만 액세스를 허용하는 것입니다.

   인증은 API의 핵심 기능이 아니기 때문에, 애플리케이션 코드 외부에서 수행하는 것이 좋습니다. 이에 따라, API 개발자들은 자체 인증 코드를 작성하지 않아도 되며, 모든 API에 대한 인증을 중앙에서 관리하면서 인증 요구 사항을 유연하게 유지할 수 있습니다. 예를 들어, 스포츠 웹사이트는 인증된 절차를 거치지 않은 사용자가 게임 점수를 반환하기 위해 API를 사용하는 것은 허용할 수 있지만, API를 사용해 콘텐츠를 편집하려는 사용자는 반드시 인증 절차를 거치도록 해야 합니다.

   이제 인증과 권한 부여의 차이점을 살펴보겠습니다. 인증은 사용자 계정을 확인하는 프로세스입니다. 권한 부여는 다음 단계로 진행되며, 특정 사용자가 수행할 자격이 있는 작업을 결정한 다음, 해당 정보를 서버에 전달합니다.

2. 속도 제한

속도 제한은 특정 클라이언트가 API를 호출할 수 있는 빈도를 제어합니다. 이는 백엔드 서비스의 과부하 방지와 클라이언트를 위한 공정한 사용 보장이라는 두 가지 주요 목적이 있습니다. 속도 제한의 한 예로 수요가 많은 기간 동안 초당 100개의 트랜잭션을 허용하는 것을 들 수 있습니다. 속도 제한은 개별 사용자 이름, 특정 IP 주소나 범위, 또는 모든 사용자 (예를 들어 피크 트래픽 시간)에 적용할 수 있습니다.

3. 입력 유효성 검사

입력 유효성 검사는 사용자 또는 애플리케이션이 제공한 입력이 올바른지 확인합니다. 올바른 유형의 문자 (자릿수, 문자, 구두점)로 구성되고, 올바른 크기이며, 사전에 정의한 허용치 세트 중 하나이며, 제공되고 있는 다른 값과 일치하는지 등을 검사합니다. 예를 들어, 우편 번호가 제공된 주소와 일치하는지, 또는 생년월일이 미래가 아닌지 확인할 수 있습니다. 입력 유효성 검사는 부적절하게 작성된 데이터가 정보 시스템에 입력되어 무결성을 위협하는 것을 방지합니다. 또한 악성 사용자를 탐지하는 중요한 방법으로서, 악성 사용자의 추가 요청을 차단할 수 있습니다.

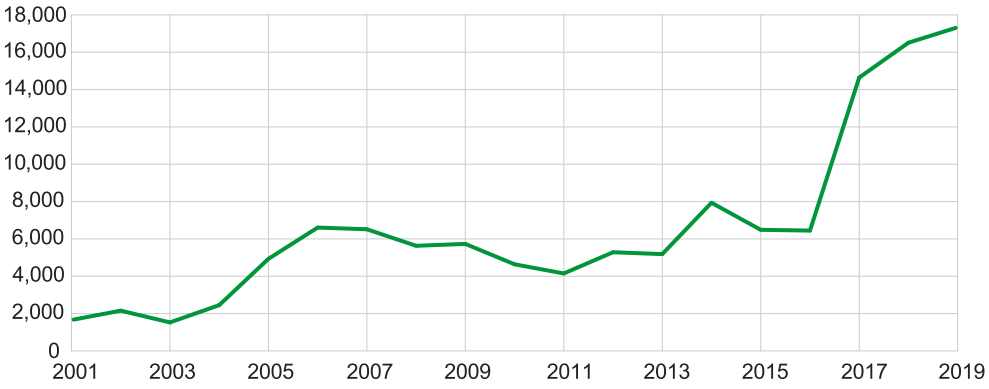
API는 오늘날 비즈니스 환경에서 성공하는 데 필요한 민첩성과 속도를 제공하는 전략적 필수 요소입니다. 그러나 보안 침해로 인한 비용이 증가하면서, 기업들은 API를 통해 데이터를 노출하더라도 매출과 순익에 영향을 미치는 보안 위험을 발생시키지 않기를 바라고 있습니다.

3. 오픈소스 취약점 증가에 따른 모던 웹방화벽의 필요성

작성자: 케빈 존스(Kevin Jones), NGINX (part of F5)의 수석 제품 마케팅 매니저

오늘날 보안은 웹 애플리케이션의 개발, 배포, 제공에서 중요한 부분으로 자리매김하고 있습니다. 기업들은 계속해서 빠른 속도로 애플리케이션을 제공하고 있으며, 이를 통해 경쟁력을 유지하고 있습니다. 끊임없는 전환의 과정 속에서 민첩성을 유지하기 위해 새로운 기술과 방법론을 채택하고 있습니다. 여기에는 가장 강력한 최신 오픈소스 툴, 구성 요소, 애플리케이션 스택 등이 포함됩니다. 이와 같은 빠른 속도를 통해 고객들이 애플리케이션에서 원하고 필요로 하는 것들을 제공할 수 있습니다. 하지만, 이처럼 빠르게 진행되는 혁신에는 어떤 위험이 있을까요?

그림 5: 2001년~2019년 CVE 취약점 사고 건 수

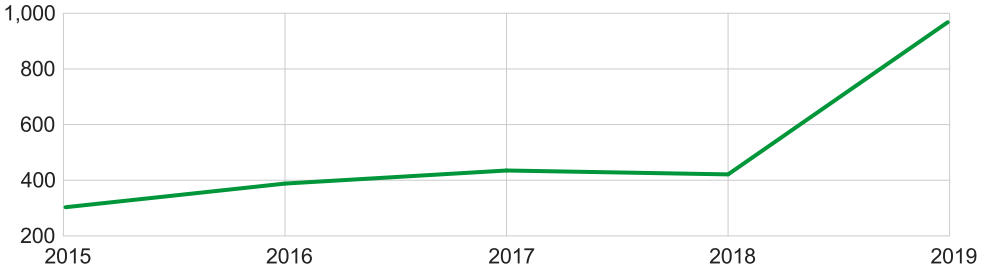


출처: NIST(National Institute of Standards and Technology)

오픈소스 취약점의 증가

실리콘 밸리(Silicon Valley)에 본사를 두고 있는 취약점 관리 전문업체인 RiskSense는 최근 오픈소스의 어두운 현실(The Dark Reality of Open Source)이라는 제목의 연구 보고서를 발간했습니다. 그 목표는 오픈소스 제품에서 야기되는 애플리케이션 보안에 대한 위협을 파악하는 것이였습니다. 놀랍게도 오픈소스 소프트웨어의 공통 취약점 및 노출(Common Vulnerabilities and Exposures,CVE) 수가 2018년 421개에서 2019년 968개로 130% 증가했습니다. 또한, 취약점이 공개된 후 National Vulnerability Database에 추가되는 데 평균 54일이 걸렸기 때문에, 해당 소프트웨어를 사용하는 조직은 "거의 2개월 동안 심각한 애플리케이션 보안 위험에 노출"됐습니다.

그림 6: 2015년–2019년 연간 오픈소스 취약점



출처: RiskSense 스포트라이트 보고서: 오픈소스의 어두운 현실(The Dark Reality of Open Source)

최근 몇 년 간, 오픈소스 소프트웨어 취약점은 Apache Struts exploit (CVE-2017-5638)과 같은 많은 주요 데이터 침해의 원인으로 지목됐습니다. 이 익스플로잇을 통해 공격자는 OGNL (Object-Graph Navigation Language)이 포함된 특정 HTTP 요청 데이터를 전달할 수 있으며, 이를 통해 Java 내에서 속성을 읽고 설정할 수 있을 뿐만 아니라 메서드 실행도 가능합니다. 이 때문에 공격자들은 원격 코드 실행(Remote Code Execution, RCE) 공격을 감행할 수 있었으며, 2017년에는 Equifax에서 1억4,300만 개 이상의 계정이 유출되었습니다. 오픈소스 환경의 수많은 취약점을 고려할 때, 이러한 악의적인 공격으로부터 사용자, 네트워크, 그리고 가장 중요한 데이터를 어떻게 보호할 수 있을까요?

웹방화벽은 어떻게 소프트웨어 보안을 지원하는가

퍼즐의 가장 중요한 부분은 웹방화벽입니다.

보안에 대한 전반적인 솔루션은 복잡할 수 있지만, 퍼즐에서 가장 중요한 부분은 웹방화벽(WAF)입니다. 웹방화벽은 웹 애플리케이션으로 향하는 악성 HTTP/S 트래픽을 필터링, 모니터링 및 차단하고, 허가없이 데이터가 애플리케이션에서 나가는 것을 방지함으로써 웹 애플리케이션을 보호합니다. 이는 악성 트래픽과 안전한 트래픽을 식별할 수 있도록 돕는 일련의 정책을 준수함으로써 수행됩니다. 다른 보안 전략과 마찬가지로, 다계층 방어 체계가 필요합니다. 웹방화벽이 모든 공격을 차단하는 것은 아니지만, 지원되는 버전에 패치를 설치하고 유지 관리하면, 제로 데이(zero-day) 공격을 완화시킬 수 있습니다.

다른 예를 살펴보겠습니다. OWASP(Open Web Application Security Project)에 따르면 웹 애플리케이션에 실행할 수 있는 가장 일반적인 악성 공격 중 하나가 SQL injection입니다. 공격자가 취약점을 악용해 웹 애플리케이션 내에서 악의적인 SQL 명령을 실행하면, 데이터베이스에서 민감한 데이터가 노출됩니다. 이 공격은 OWASP Top 10 공격 리스트에서 1위를 기록할 정도로 매우 자주 발생합니다.

NGINX는 사용자에게 보안이 얼마나 중요한지 잘 알고 있습니다. NGINX App Protect를 개발할 때, 보안 목표를 달성할 수 있도록 NGINX의 강력한 아키텍처를 활용했습니다. NGINX App Protect는 신뢰할 수 있는 고성능 모던 애플리케이션 보안 솔루션입니다. 시장을 선도하는 F5의 웹방화벽을 기반으로 설계된 이 제품은 기본적으로 NGINX Plus에서 실행되며 보안 제어 기능을 애플리케이션에 직접 통합합니다. NGINX App Protect를 개발할 때, 이전 NGINX 제품과 동일한 철학을 유지하면서 성능, 확장성 및 경량 아키텍처에 중점을 두었습니다.

NGINX App Protect에 대한 자세한 내용과 인프라 전체에 보안 보호를 제공하는 방법은 다음 장을 참조하십시오.

4. NGINX App Protect 소개: NGINX Plus를 위한 고급 F5 애플리케이션 보안 솔루션

작성자: 마크 캠벨(Mark Campbell), F5, Inc.의 제품 마케팅 담당 수석 매니저

디지털 트랜스포메이션을 진행하고 있는 기업들은 분명한 비즈니스 요구 사항들을 가지고 있습니다. 여기에는 모던 비즈니스 애플리케이션에 의한 고객 경험 개선, 시장에서 경쟁자들을 앞지르기 위한 애자일 방식 채택, 시장 우위를 활용한 새로운 수익원 창출 등이 포함됩니다. 이러한 노력은 개발 효율성을 높이고 컨테이너, 마이크로서비스, API를 통합하는 새로운 애플리케이션 아키텍처를 통해 뒷받침됩니다.

전통적인 애플리케이션에 대한 보안 제어가 항상 비즈니스 요구 사항에 부합하는 것은 아닙니다.

모던 애플리케이션의 경우, 민첩성과 출시 시간이 핵심입니다. 보안은 종종 부차적인 고려 사항이거나, 완전히 무시됩니다. 왜 그럴까요? 전통적인 애플리케이션들에 대한 보안 제어가 항상 비즈니스 요구 사항에 부합하는 것은 아니기 때문입니다. 예를 들어, 전통적으로 SecOps 팀이 구성하고 운영하는 웹방화벽(WAF)이 일반적으로 특정 사업부를 지원하는 DevOps 팀이 배포한 애자일 애플리케이션에는 적합하지 않습니다. 그 결과, 적절하지 않게, 또는 잘못 구성된 보안으로 인해 출시 시간이 지연되고 만족스럽지 못한 사용자 경험을 초래할 수 있습니다.

NGINX APP PROTECT 소개

NGINX의 수석 부사장 겸 총괄 관리자인 거스 로버트슨(Gus Robertson)은 "1월에 NGINX Controller 3.0을 발표한 후, 불과 몇 개월 만에 NGINX와 F5가 함께 더 나은 성과를 거두고 있다는 것을 입증하는 또 다른 제품을 출시하게 된 것을 매우 기쁘게 생각합니다. 우리는 계속해서 혁신에 박차를 가할 것이며, 이를 통해 디지털 트랜스포메이션의 여정을 진행하고 있는 고객들을 위해 더 높은 가치를 제공할 것입니다."라고 밝혔습니다.

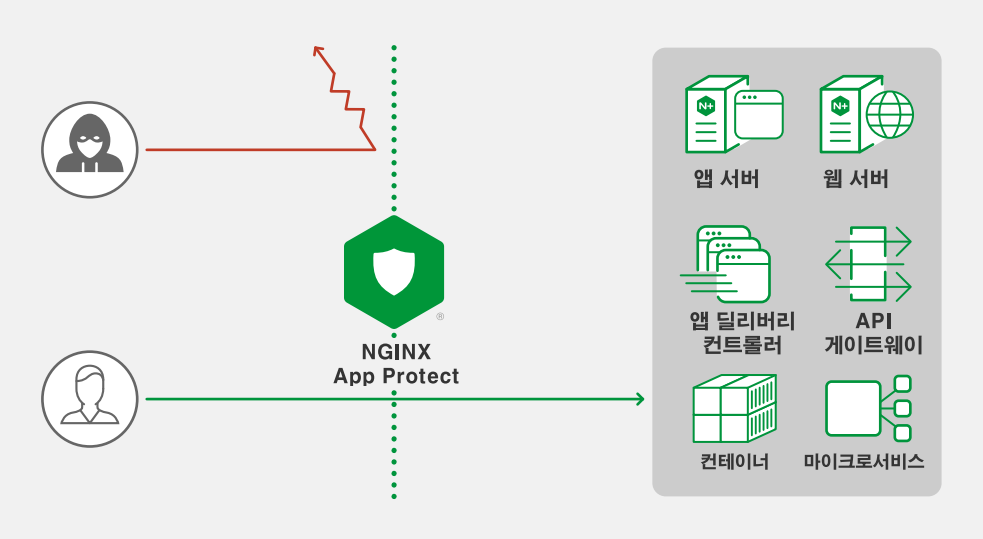


그림 7: NGINX App Protect를 사용해 신속한 위협 방어와 분석을 실행함으로써 애플리케이션 보호

NGINX App Protect는 Advanced F5 WAF 기술의 효율성과 NGINX Plus의 민첩성 및 성능을 결합한 새로운 차원의 애플리케이션 보안 솔루션입니다.

이 솔루션은 기본적으로 NGINX Plus에서 실행되며, 모던 DevOps 환경이 직면한 가장 어려운 과제들을 해결합니다.

- 보안 제어를 개발 자동화 파이프라인에 직접 통합
- 컨테이너, 마이크로서비스 등과 같은 모던 분산 애플리케이션 환경에 보안 적용 및 관리
- 릴리스 및 출시 속도에 영향을 미치지 않으면서 적절한 수준의 보안 제어 제공
- 보안 및 규정 요구사항 준수

강력한 F5 애플리케이션 보안

"차단 모드"에서 NGINX App Protect 보안 정책을 자신있게 배포할 수 있습니다.

NGINX App Protect의 보안 제어 기능들은 F5의 Advanced WAF 기술에서 직접 이식되어 ModSecurity와 같은 커뮤니티 지원 솔루션에서 대폭적인 업그레이드를 제공합니다. 포괄적인 웹방화벽 공격 시그너처 세트는 광범위한 현장 테스트를 거쳐 사실상 오탐(false positives)이 전혀 발생하지 않는 것으로 검증됐으므로, 프로덕션 환경에서도 "차단 모드"로 보안 정책을 자신있게 배포할 수 있습니다. NGINX App Protect는 OWASP Top 10 웹 애플리케이션 보안 위험으로부터 보호하고, 프로토콜을 준수하도록 하며, 일반적인 회피 기법에 맞서 방어합니다. 또한, 거부 리스트(denylisting)를 제공하고, 쿠키를 검사하며, API를 보호하는 것은 물론, F5의 Data Guard를 이용해 민감한 데이터의 유출을 방지합니다.

OWASP TOP 10 웹 애플리케이션 보안 위험

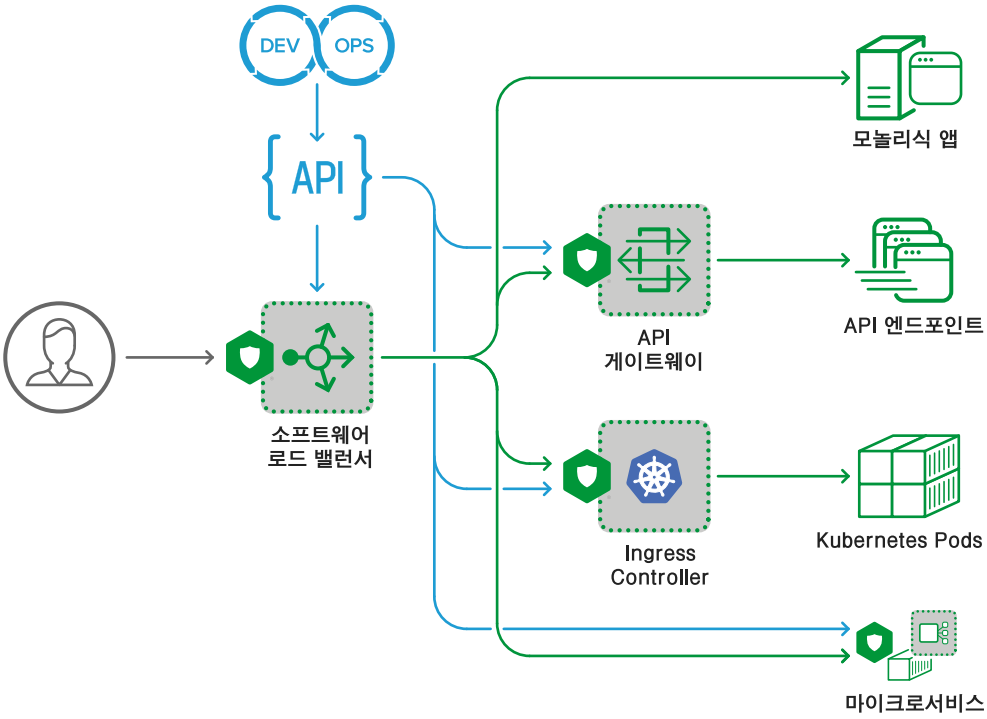
1. Injection
2. 취약한 인증
3. 민감한 데이터 노출
4. XML 외부 엔티티 (XXE)
5. 취약한 액세스 제어
6. 보안 설정 오류
7. XSS(Cross-Site Scripting)
8. 안전하지 않은 역직렬화
9. 알려진 취약점이 있는 구성 요소 사용
10. 비효율적인 로깅과 모니터링

모던 애플리케이션을 위한 설계

강력한 보안 제어가 애플리케이션의 운영 환경에서 구현될 수 없다면, 아무런 도움이 되지 않습니다. NGINX App Protect는 모던 애플리케이션 배포 토폴로지를 지원하도록 개발됐습니다. NGINX Plus의 일반적인 배포 모드는 다음과 같습니다:

- 로드 밸런서
- API 게이트웨이
- Kubernetes Pods용 Ingress controller
- 마이크로서비스용 Per–Pod 프록시

그림 8: NGINX App Protect는 인프라 전체에 배포합니다.



탁월한 속도 보장

NGINX APP PROTECT는 BYTECODE로 컴파일되기 때문에, 트래픽이 매우 빠르게 처리됩니다.

안타깝게도 보안을 위해 성능 저하를 감수하거나, 성능을 위해 보안 수준을 낮춰야 하는 경우가 많습니다. 예를 들어 ModSecurity 컨트롤에는 정규 표현식에 대한 평가가 포함됩니다. 따라서, 추가 컨트롤을 활성화할 때마다 직접적으로 성능이 저하되기 때문에 많은 관리자들은 매우 적은 수의 컨트롤만을 구현하고 있습니다. 이와 달리, NGINX App Protect 컨트롤은 bytecode로 컴파일되기 때문에, 실행하는 공격 시그너처 수에 관계없이 트래픽이 매우 빠르게 처리됩니다. 그 결과, Core Rules Set v3가 활성화된 ModSecurity 구현에 비해 최대 20배의 초당 Throughput과 요청을 달성했습니다.

DEVOPS는 혁신에 집중할 수 있도록 지원

SecOps와 DevOps 간의 관계는 특히, 보안 요구 사항이 릴리스 속도에 지장을 주는 경우 종종 불편해질 수 있습니다. 정적 애플리케이션 보안 테스트(Static Application Security Testing, SAST)와 소프트웨어 구성 분석(Software Composition Analysis, SCA)은 개발 초기에 보안 결함을 찾아내는 데는 훌륭한 툴이지만, 애플리케이션이 릴리스 게이트를 통과하기 전에는 많은 취약점이 발견되지 않습니다. 애플리케이션을 개발로 되돌려 보내면 비용이 증가하고 생산성이 저하됩니다. 애플리케이션이 아직 개발 파이프라인에 있는 동안 결함을 찾아내는 것이 보안 정책을 조정하건, 아니면 코드를 수정하건 관계없이 훨씬 더 효율적입니다.

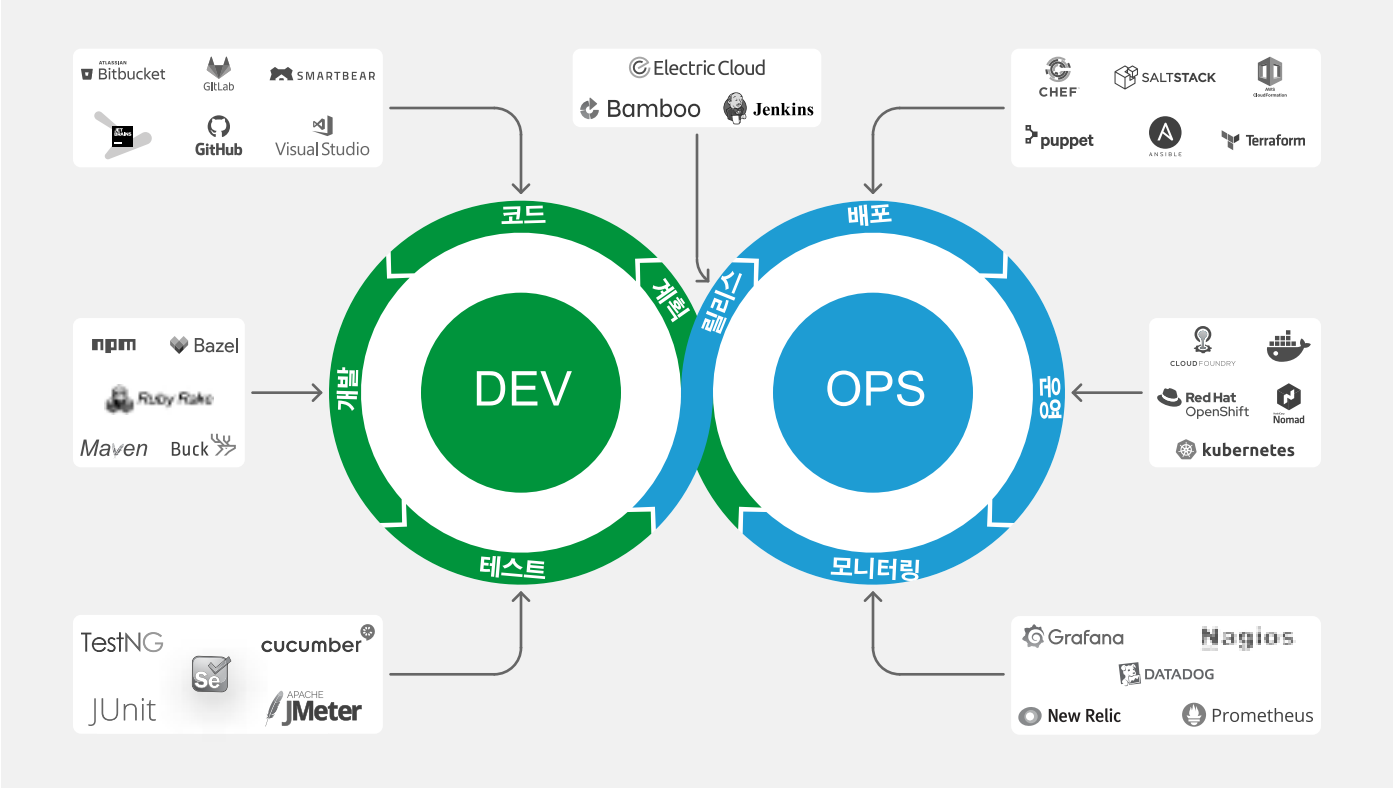


그림 9: DevOps 애플리케이션 라이프사이클과 생태계

NGINX App Protect는 DevOps 친화적이며 공통 개발 파이프라인에 쉽게 통합됩니다. NGINX App Protect의 선언형 구성 기능을 사용하면 보안이 DevOps CI/CD 자동화의 일부가 되어 애플리케이션 기능 사양의 다른 부분과 마찬가지로 테스트를 받을 수 있습니다. 본질적으로 보안 정책과 구성은 소스 코드 저장소에서 가져온 "코드"로서 사용됩니다. SecOps 팀은 보안 정책을 만들고 유지함으로써 비즈니스를 보호하는 데 필요한 제어 체계가 제대로 갖춰져 있는지 확인합니다. 이는 릴리스 속도를 유지하는 데 도움이 될 뿐만 아니라, DevOps와 SecOps 팀 간의 간극을 해소하는 데도 도움이 됩니다.

NGINX App Protect는 애플리케이션 가까이에 배치되도록 설계된 모던 WAF이며, 경계 WAF 이상의 추가 보호를 제공합니다. DevOps와 CI/CD 프레임워크에 완전히 통합되어 다음을 수행할 수 있습니다.

- 강력한 보안 컨트롤을 NGINX Plus에 원활하게 통합 가능
- 다른 WAF를 능가하는 성능을 통한 사용자 경험 향상
- 모던 애플리케이션을 제공하면서 복잡성 및 무분별한 툴 도입 감소

5. NGINX App Protect로 PCI DSS 규정 준수 달성

야니프 사즈만(Yaniv Sazman), F5, Inc.의 수석 제품 매니저

디지털 트랜스포메이션은 보안 환경을 바꾸어 놓았습니다. 기업들이 비즈니스 민첩성을 높이기 위해 모놀리식 애플리케이션에서 클라우드 네이티브 마이크로서비스 아키텍처로 전환함에 따라 전통적인 디지털 보안은 더 이상 존재하지 않습니다. 마이크로서비스는 모든 네트워크를 통해 통신하기 때문에, 모던 웹 사이트와 웹 애플리케이션은 모놀리스보다 사이버 공격에 더 취약하며 모든 규모의 회사 네트워크를 손상시키는 가장 쉬운 방법 중 하나가 되었습니다. 기업들은 보안과 민첩성 사이에서 적절한 균형을 찾아야 합니다.

신용카드 업계는 계속해서 사이버 공격의 대표적인 표적이 되고 있습니다. 이 장에서는 기업들이 신용카드 거래를 처리할 때 직면하는 특정 보안 및 규정 준수 문제들을 짚어보고, 특히 웹방화벽 및 NGINX App Protect와 같은 기술들이 규제 요구 사항을 충족하는 데 어떻게 도움이 되는지에 대해 설명합니다.

PCI DSS 준수는 모던 애플리케이션에서 매우 중요

PCI DSS(Payment Card Industry Data Security Standard)는 신용카드 결제 처리와 관련된 모든 당사자가 카드 소유자 데이터 보호를 위해 취해야 하는 조치를 기술하고 있습니다. 첫 번째 요구 사항은 "카드 소유자 데이터 보호를 위해 방화벽 구성을 설치하고 유지 관리할 것"입니다. 또한 요구 사항 6.6은 일반 대중을 대상으로 하는 웹 애플리케이션의 소유자에게 "웹 기반 공격을 탐지 및 방지하는 자동화된 기술 솔루션(예를 들어, 웹방화벽)을 설치하여 데이터를 보호해야..."라고 명시하고 있습니다. . ."

웹방화벽 설치는 단순하게 "설치하면 끝나는" 문제가 아닙니다.

안타깝게도 웹방화벽 설치는 "설치하면 끝나는" 단순한 문제가 아닙니다. 다양한 공격들이 이루어질 수 있으며, 공격자들은 끊임없이 새로운 공격을 시도하고 있습니다. 이에 따라, PCI DSS 규정 준수 유지는 모던 애플리케이션이 직면한 가장 중요한 과제 중 하나입니다.

이 표준의 요구사항 6.5에는 웹방화벽이 "최소한" 방어해야 하는 취약점들을 열거하고 있습니다.

- Injection flaws, 특히 SQL injection뿐만 아니라 OS Command Injection, LDAP, XPath injection flaws 등이 있습니다.
- 버퍼 오버플로우
- 안전하지 않은 암호화 저장소
- 안전하지 않은 통신
- 부적절한 오류 처리
- XSS(Cross-Site Scripting)

- 사이트 간 요청 위조 공격(CSRF)
- 취약한 인증과 세션 관리
- 부적절한 액세스 제어(안전하지 않은 직접 객체 참조, URL 액세스 제한 실패 등)

PCI DSS 목록은 일반적으로 사용되는 다른 취약점 목록인 Open Web Application Security Project(OWASP) Top 10과도 완전히 겹치지 않습니다, 여기에는 XML 외부 엔티티, 구성 설정 오류(기본 구성 사용 등), 안전하지 않은 역직렬화, 불충분한 로깅과 모니터링 등이 추가되어 있습니다.

NGINX APP PROTECT, PCI DSS 요구 사항을 완벽하게 준수

PCI DSS를 준수하고 계속해서 증가하는 취약점으로부터 애플리케이션을 보호하기 위해서는 NGINX App Protect와 같은 모던 웹방화벽 솔루션이 필요합니다. 이는 나열된 PCI DSS 취약점, OWASP Top 10 등으로부터 보호합니다.

NGINX App Protect는 모던 인프라를 위해 설계됐으며 어디에나 설치할 수 있습니다. 그리고 CI/CD 파이프라인에 "코드로서" 직접 삽입되며 기존 웹방화벽보다 애플리케이션에 더 가깝기 때문에 보안 정책을 신속하게 업데이트 할 수 있습니다. NGINX App Protect는 모든 플랫폼(퍼블릭과 프라이빗 클라우드, VM, 컨테이너 등)과 활용 사례(API 게이트웨이, Kubernetes Ingress controller 포함)에 배포되기 때문에 전체 인프라 전반에서 일관된 성능과 동일한 수준의 보호를 달성할 수 있습니다.

NGINX APP PROTECT는 6,000개 이상의 시그너처를 처리합니다

NGINX App Protect는 알려진 신종 공격들을 처리하기 위해 최소 2개월마다 업데이트되는 6,000개 이상의 보안 정책(시그너처) 셋을 처리합니다.

또한, NGINX App Protect는 시그너처 외에도 다음을 수행합니다:

- 요청별로 HTTP 프로토콜 및 회피 기법 검사를 수행함으로써 HTTP 메시지 내용의 불법 메타문자, 잘못된 길이 등과 같은 오류를 감지합니다. 이러한 이상 징후는 알려지지 않은 공격(제로 데이) 가능성을 나타낼 수 있으며, 이들의 존재는 해당 트래픽 내에 존재할 수 있는 다른 증거들을 보강합니다.
- JSON과 XML 콘텐츠를 처리하고, 잠재적으로 악의적인 injection에 대한 페이로드를 확인할 수 있습니다.
- 데이터를 마스킹(응답 스크러빙으로도 불림)해 응답이 민감한 정보를 노출하지 않도록 하는 고유한 기능을 제공합니다. 노출되어서는 안되는 기밀 데이터를 애플리케이션이 반환할 때 응답 스크러빙을 항상 활성화하는 것이 좋습니다.

6. NGINX App Protect를 통한 민첩한 경계 보안

작성자: 마크 캠벨(Mark Campbell), F5, Inc.의 제품 마케팅 담당 수석 매니저

컴퓨터 보안의 맥락에서 경계란 애플리케이션과 그 내부의 기타 인프라 요소에 대한 "신뢰 영역"을 설정하는 개념적인 선입니다.

성과 해자(castle-and-moat) 접근방식을 사용하는 기존 인프라 환경에서 경계는 인트라넷(내부 네트워크)을 엑스트라넷 또는 인터넷에서 분리시켰습니다. 인트라넷은 안전하다고 여겼으며, 위협은 외부에서만 오는 것으로 간주했습니다. 보안 태세는 다소 정적이고 패킷 검사와 액세스 제어 조치를 통해 인트라넷 주변에 방어 장치를 구축하는 것으로 구성됐습니다.

외부 공격자들은 보안 제어를 우회할 수 있는 방법을 모색했습니다

그러나 시간이 지나면서 외부 공격자들은 보안 제어를 우회한 뒤 인트라넷의 구성 요소를 손상시키거나, 외부 공격을 시도하면서 요청이 경계 내에서 시작된 것처럼 보이게 만드는 방법을 찾아냈습니다. 인트라넷이 안전하다고 간주됐기 때문입니다. 이는 신뢰 영역이 설정되기 전에 모든 엔티티(내부와 외부)를 지속적으로 평가하는 제로 트러스트 보안 모델(Zero Trust security model)로 변경하는 계기가 됐습니다. 인트라넷은 이제 더 이상 안전한 것으로 간주되지 않습니다.

디지털 트랜스포메이션, 마이크로서비스 등 새로운 앱 아키텍처도 추가 보안 문제를 초래했습니다. 많은 기업 애플리케이션이 퍼블릭 클라우드에서 호스팅되거나 클라우드와 온-프레미스 토폴로지에 분산되어 있어 애플리케이션을 보호하는 보안 인프라는 더 이상 로컬 관리자의 통제를 받지 않습니다. 이에 따라, 많은 기업들은 이제 개별 애플리케이션(또는 이들 간에 직접적인 구조적 종속성이 있는 소규모 애플리케이션 그룹)을 중심으로 경계를 설정합니다. 아래 그림에서 녹색 점선은 경계를 나타냅니다.

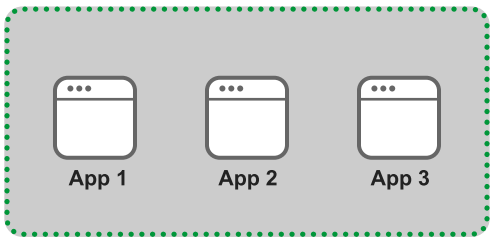
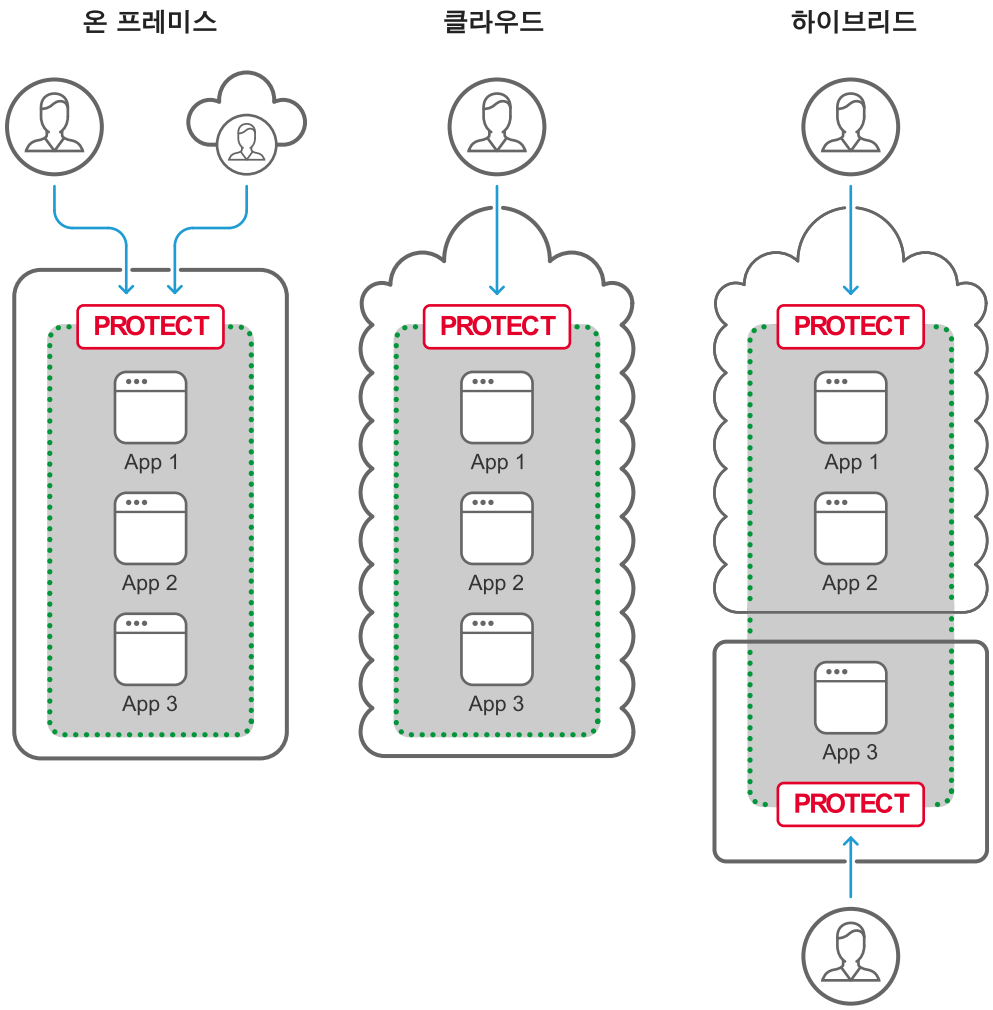


그림 10: 관련 애플리케이션 그룹 주변의 경계

아키텍처 모델에 관계없이 들어오는 트래픽을 검사하고, 내부의 애플리케이션을 보호하는 보안 정책을 적용하기 위해 경계에 배치된 "게이트키퍼"가 있습니다. 이 게이트키퍼를 엣지라고 부릅니다. 그림 11에 있는 세 가지 일반적인 배포 패턴에서 엣지는 **PROTECT**라는 단어를 둘러싼 빨간 상자로 표시됩니다.

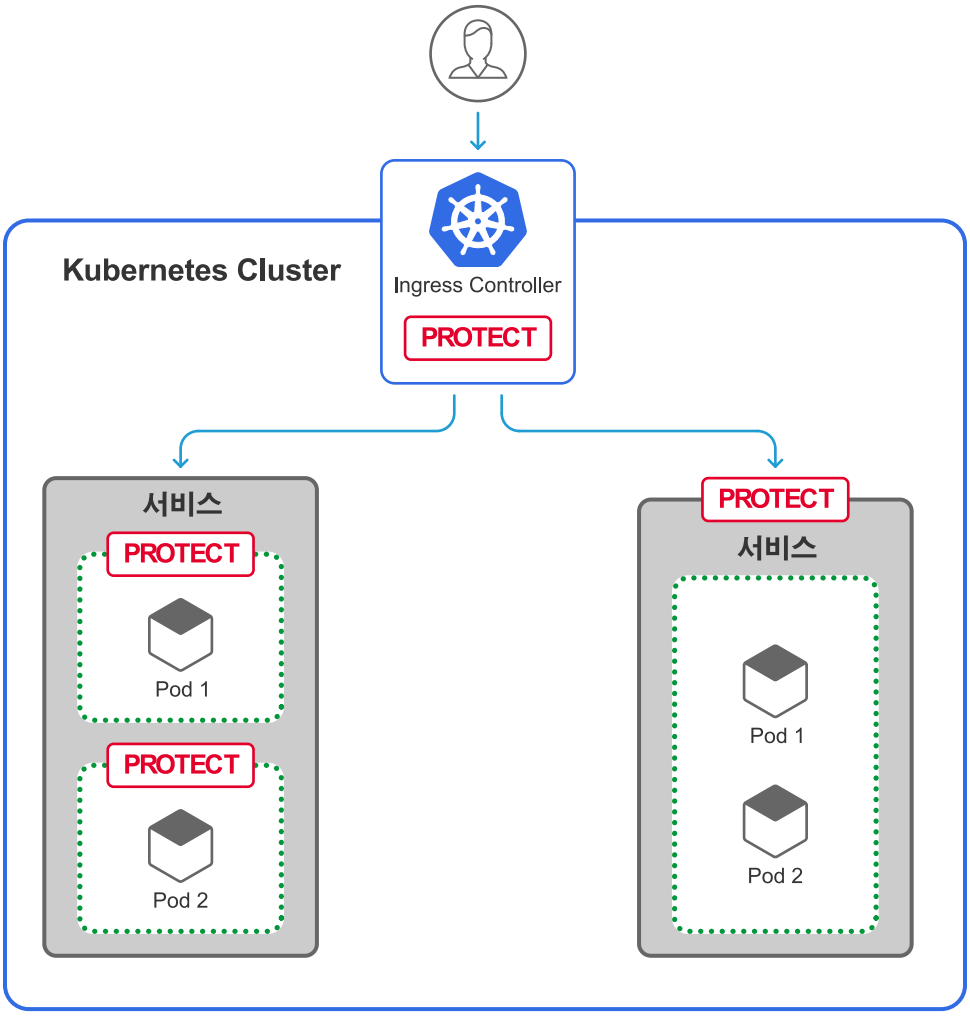
그림 11: 세 가지 일반적인 배포의 엣지



Kubernetes 프레임워크와 같은 컨테이너형 아키텍처에서도 동일한 개념이 적용됩니다. Ingress controller는 전체 Kubernetes 클러스터의 엣지 역할을 수행하며, 외부 클라이언트의 액세스를 관리하고 클러스터의 Kubernetes 서비스에 대한 요청을 라우팅합니다. 그러나, 그림 12에서 볼 수 있듯이, 보안 정책은 클러스터 내에서도 더욱 세분화된 수준(per-Pod 또는 per-Service)으로도 적용될 수 있습니다.

- per-Pod 보호(왼쪽)에서 Pod는 하나 이상의 컨테이너에 애플리케이션 또는 애플리케이션 구성 요소를 포함하는 경계를 정의합니다.
- per-Service 보호(오른쪽)에서 서비스는 하나 이상의 Pod를 통해 애플리케이션 배포의 인스턴스를 노출합니다. 경계는 서비스 뒤쪽의 pod 주변에 설정됩니다.

그림 12: Kubernetes Cluster의 엣지



NGINX APP PROTECT를 이용한 경계 보호

NGINX App Protect는 F5의 시장 선도적인 웹방화벽 기술을 기반으로 개발된 모던 애플리케이션 보안 솔루션입니다. NGINX App Protect를 사용하면 온프레미스, 클라우드, 하이브리드, 마이크로서비스 기반 또는 컨테이너형 배포 환경 또는 애플리케이션 아키텍처에 관계없이 경계를 보호하며 민첩하게 애플리케이션 보안을 강화할 수 있습니다. 자세한 내용은 제4장, NGINX App Protect 소개: NGINX Plus를 위한 고급 F5 애플리케이션 보안 솔루션을 참조하십시오.

NGINX App Protect를 이용해 엣지에서 보안을 구현하면 경계 외부에서 보안을 수행할 수 있는 주요 이점이 있습니다. 기본적으로 트래픽 검사와 액세스 제어는 경계를 통과하기 전에 위협을 차단하기 위해 엣지에서 이루어집니다. 애플리케이션으로 가는 라스트 홉(last hop)인 엣지는 애플리케이션에 대한 위협 유형과 수를 가장 잘 볼 수 있는 곳입니다.

아래 스니펫은 경계 내에서 별도로 액세스되는 세 개의 앱(**app1**, **app2**, **app3**)을 보호하도록 NGINX App Protect를 구성합니다.

```
load_module modules/ngx_http_app_protect_module.so;
error_log /var/log/nginx/error.log debug;

http {
     # Enable NGINX App Protect in 'http' context
     app_protect_enable on;

     # Enable remote logging
     app_protect_security_log_enable on;

     # Default JSON security policy
     app_protect_policy_file "/etc/nginx/NginxDefaultPolicy.json";

     # Set remote logging options (in referenced file) and log server
     # IP address/port
     app_protect_security_log    "/etc/nginx/log-default.json"
                                 syslog:server=127.0.0.1:515;

     server {
          listen 80;
          server_name app1.com;
          # JSON policy for app1
          app_protect_policy_file "/etc/nginx/NginxApp1Policy.json";

          location / {
               proxy_pass http://www.app1.com:8080$request_uri;
          }
     }
```

(이어서)

```
    server {
        listen 80;
        server_name app2.com;
        # JSON policy for app2
        app_protect_policy_file "/etc/nginx/NginxApp2Policy.json";

        location / {
            proxy_pass http://www.app2.com:8080$request_uri;
        }
    }
    server {
        listen 80;
        server_name app3.com;
        # JSON policy for app3
        app_protect_policy_file "/etc/nginx/NginxApp3Policy.json";

        location / {
            proxy_pass http://www.app3.com:8080$request_uri;
        }
    }
}
```

아래 스니펫은 경계 내에서 단일 애플리케이션에 내장되고 표시되는 **app1**, **app2**, **app3**을 보호하도록 NGINX App Protect를 구성합니다.

```
load_module modules/ngx_http_app_protect_module.so;
error_log /var/log/nginx/error.log debug;

http {
    server {
        listen   80;
        server_name app.com;

        # Enable NGINX App Protect in 'http' context
        app_protect_enable on;

        # Enable remote logging
        app_protect_security_log_enable on;

        # Default JSON security policy
        app_protect_policy_file "/etc/nginx/NginxDefaultPolicy.json";
        # Set remote logging options (in referenced file) and log
        # server IP address/port
        app_protect_security_log    "/etc/nginx/log-default.json"
                                    syslog:server=10.1.20.6:5144;
```

(이어서)

```
            location / {
                # Main JSON policy file
                app_protect_policy_file "/etc/nginx/policy/policy_main.json";
                proxy_pass http://app.com$request_uri;
            }
            location /app1 {
                # JSON policy file for app1
                app_protect_policy_file "/etc/nginx/policy/policy_app1.json";
                proxy_pass http://app.com$request_uri;
            }

            location /app2 {
                # JSON policy file for app2
                app_protect_policy_file "/etc/nginx/policy/policy_app2.json";
                proxy_pass http://app.com$request_uri;
            }

            location /app3 {
                # JSON policy file for app3
                app_protect_policy_file "/etc/nginx/policy/policy_app3.json";
                proxy_pass http://app.com$request_uri;
            }
        }
    }
```

두 구성 모두 각 애플리케이션에 대해 별도의 `app_protect_policy_file` 지시문이 있으며, 보안 요구사항이 다르기 때문에 각각 고유한 보안 정책을 할당합니다.

추가 NGINX App Protect 구성에 대해서는 설명서를 참조하십시오.

NGINX APP PROTECT를 통한 CI/CD 파이프라인의 경계 보안

NGINX APP PROTECT는 모던 애플리케이션의 확장성 및 자동화와 관련한 보안 문제를 해결합니다.

NGINX App Protect는 모던 애플리케이션의 확장성 및 자동화와 관련한 보안 문제를 해결합니다. NGINX App Protect를 CI/CD 파이프라인의 여러 통합 지점에 직접 삽입하면, 더 코드 가까이에서 애플리케이션, Pod, 서비스를 보호하고 개발, 운영, 보안 간의 격차를 해소할 수 있습니다.

애플리케이션 개발 주기에 보안을 직접 통합하면 보안 테스트를 수행하고 자동화하여 애플리케이션과 그 구성 요소에 대한 보안 위험을 발견할 수 있습니다. 보안 규정 준수 요구사항을 준수하지 않는 경우, 보안 정책 적용을 정의하고 애플리케이션 게시(publication)를 되돌릴 수 있습니다. 실제로, 게시(publication)하기 전에 NGINX App Protect를 새로운 버전의 애플리케이션에 통합하면, 안전하게 보호되는 애플리케이션을 계속해서 제공할 수 있습니다.

7. NGINX App Protect로 Kubernetes에서 애플리케이션 보호

아미르 라우닷(Amir Rawdat), NGINX (part of F5)의 기술 마케팅 엔지니어

기업들은 서비스와 애플리케이션을 시장에 빨리 출시해야 한다는 사실을 알고 있습니다. 그렇지 않으면 경쟁업체가 분명 먼저 내놓을 것이기 때문입니다. 그러나 웹 애플리케이션은 사이버 공격의 주요 표적이며, 이를 정신없이 빠르게 업데이트하면 잠재적인 보안 취약점이 QA 단계에서 걸러지지 못하고 프로덕션 환경으로 넘어갈 위험이 커집니다.

많은 요인들로 인해 강력한 보안 표준을 일관되게 적용하기 어렵습니다.

많은 요인들로 인해 강력한 보안 표준을 일관되게 적용하기 어렵습니다. 코드를 빨리 프로덕션 환경에 배포해야 한다는 압력 때문에 보안을 등한시하려는 유혹이 생깁니다. 취약점 스캐너와 같은 자동화 툴은 모든 문제를 포착하지는 못하기 때문에, 지나치게 의존하면 위험합니다. 여러 다양한 부서의 개발 팀에서 제공한 코드를 조합하는 경우, 누가 보안 적용에 대한 책임을 맡고 있는지 명확하지 않게 됩니다. 프로덕션 환경에서 여러 애플리케이션과 애플리케이션 버전을 실행하면 애플리케이션 보안의 틈새가 몇 배 더 벌어질 수 있습니다.

이에 따라, 웹방화벽(WAF)과 같은 보안 툴의 필요성이 그 어느 때보다 절실해졌습니다. 이들 보안 툴은 종종 로드 밸런싱 프록시와 통합되며, 회사 네트워크의 엣지(또는 프론트 도어)에 배포되어 보안 경계를 구축합니다.

모던 애플리케이션 및 인프라에 대한 보안 침해로 인해 이 접근 방식에 필요한 두 가지 개선 사항이 밝혀졌습니다.

- **경계 보안만으로는 충분하지 않습니다.** 쉽게 보호되는 단일 경계는 거의 없으며, 웹방화벽과 같은 프록시 기반 보안 툴은 보호하는 애플리케이션에 더 가깝게 배포해야 합니다.
- **보안은 더 이상 CISO 및 SecOps 팀만이 담당하는 영역이 아닙니다.** DevOps 팀은 CI/CD 파이프라인의 일부로서 보안 정책을 수용, 테스트 및 배포하는 데 중요한 역할을 합니다.

NGINX PLUS INGRESS CONTROLLER와 NGINX APP PROTECT

NGINX Plus Ingress Controller for Kubernetes release 1.8.0 이상을 사용하면, Ingress Controller 내에 NGINX App Protect 웹방화벽을 내장할 수 있습니다.

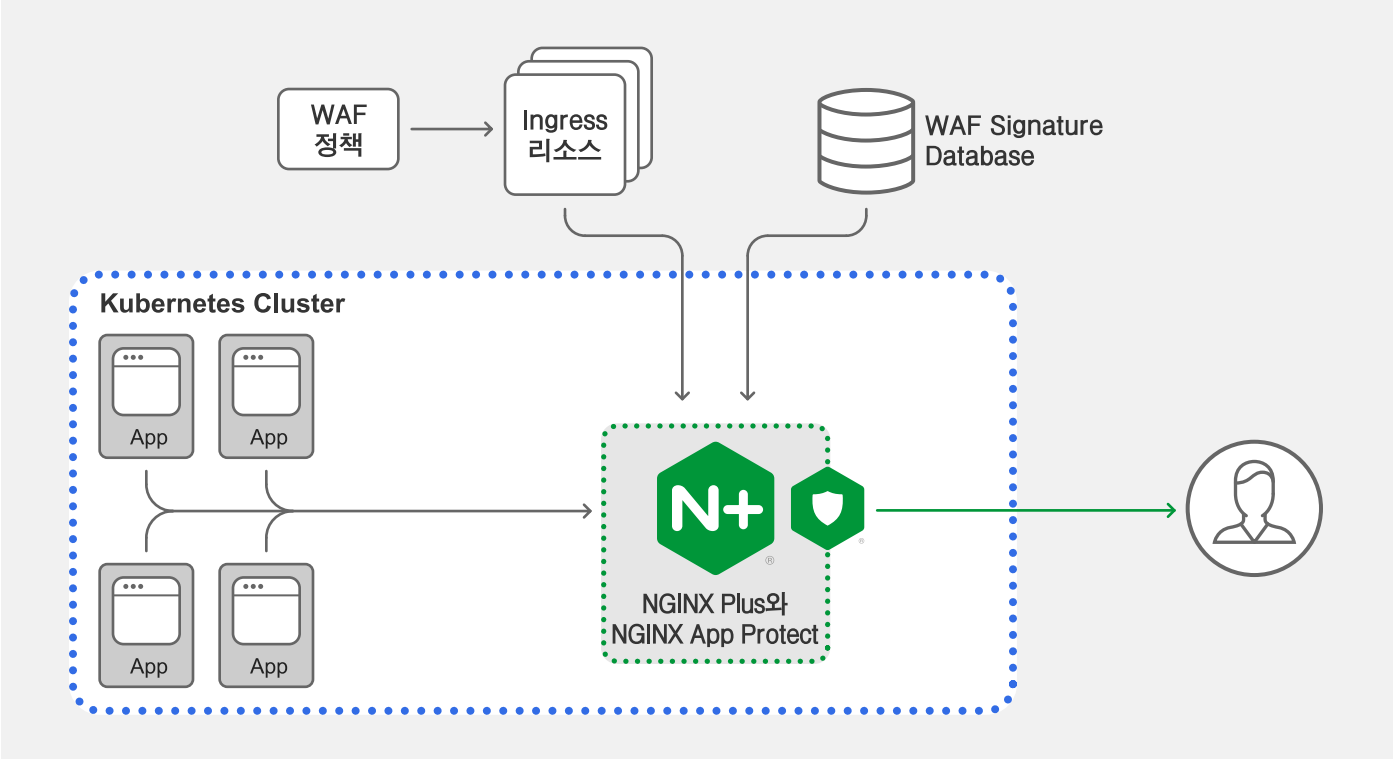


그림 13: NGINX App Protect를 통합한 NGINX Plus Ingress Controller

NGINX App Protect를 NGINX Plus Ingress Controller에 통합하려면 NGINX Plus 및 App Protect에 대한 Subscription(구독)이 필요 합니다. 몇 가지 간단한 단계만 거치면 통합 NGINX Plus Ingress Controller 이미지 (Docker 컨테이너)를 구축할 수 있습니다. Helm 차트를 사용하거나 Red Hat OpenShift를 포함해 지원되는 플랫폼에서 NGINX Ingress Operator를 사용해 이미지를 수동으로 배포할 수 있습니다. 그런 다음 익숙한 Kubernetes API를 사용해 보안 정책과 구성을 관리합니다.

웹방화벽을 NGINX PLUS INGRESS CONTROLLER에 통합하는 것이 왜 중요한가?

NGINX App Protect WAF를 NGINX Plus Ingress Controller에 통합하면 다음과 같은 세 가지 고유한 이점을 누릴 수 있습니다.

- **애플리케이션 경계 보호** – 올바르게 설계된 Kubernetes 배포 환경에서 Ingress Controller는 Kubernetes 내에서 실행되는 서비스로 향하는 데이터 플레인 트래픽을 위한 유일한 진입점이며, 따라서 보안 프록시에 이상적인 위치입니다.
- **데이터 플레인 통합** – Ingress Controller에 웹방화벽을 내장하면 별도의 웹방화벽 장치가 필요하지 않습니다. 이러한 방식으로 복잡성, 비용 및 장애 지점 수를 줄일 수 있습니다.
- **컨트롤 플레인 통합** – 이제 WAF 구성은 Kubernetes API로 관리되기 때문에 CI/CD 프로세스를 훨씬 쉽게 자동화할 수 있습니다.
  Ingress Controller 구성은 Kubernetes 역할 기반 액세스 제어(Role–Based Access Control, RBAC) 방식을 따르기 때문에 WAF 구성을 전담 DevSecOps 팀에 안전하게 위임할 수 있습니다.

App Protect의 구성 오브젝트는 Ingress Controller(YAML 파일 사용)와 NGINX Plus (JSON 사용) 모두에서 일치합니다. 마스터 구성은 쉽게 변환되고 어느 장치로든 배포될 수 있으므로, WAF 구성을 코드로 관리하고 모든 애플리케이션 환경에 쉽게 배포할 수 있습니다.

NGINX PLUS INGRESS 컨트롤러에서 APP PROTECT 구성

다음과 같은 두 가지 새로운 커스텀 리소스를 사용해 NGINX Plus Ingress Controller에서 App Protect를 구성할 수 있습니다.

- APPolicy는 App Protect가 적용할 WAF 정책을 정의합니다. WAF 정책은 독립 실행형 App Protect JSON 형식 정책의 YAML 버전입니다.
- APLogConf는 App Protect 모듈의 로깅 동작을 정의합니다.

Ingress Controller 이미지에는 빌드 때 내장되는 App Protect 시그너처 세트도 포함되어 있습니다.

적절한 APPolicy와 APLogConf 리소스를 배포하면, Annotations(주석) 세트를 사용해 Kubernetes Ingress 리소스에서 이 리소스를 참조할 수 있습니다.

```
apiVersion: extensions/v1beta
kind: Ingress
metadata:
  name: cafe-ingress
  annotations:  a
    kubernetes.io/ingress.class: "nginx"
    appprotect.f5.com/app-protect-policy: "default/dataguard-alarm"
    appprotect.f5.com/app-protect-enable: "True"
    appprotect.f5.com/app-protect-security-log-enable: "True"
    appprotect.f5.com/app-protect-security-log: "default/logconf"
    appprotect.f5.com/app-protect-security-log-destination:
"syslog:server=10.27.2.34:514"
spec:
     ...
```

그런 다음, AppProtect는 Ingress Controller가 처리하는 모든 요청을 검사하고 잠재적으로 차단합니다.

APPolicy와 APLogConf 리소스는 DevSecOps 팀이 소유한 다른 네임스페이스에서 정의될 수 있습니다. 이를 통해, 예를 들어 보안 정책을 전담 팀에 위임하는 대기업에서는 우려 사항들을 안전하고 확실하게 분리할 수 있습니다.

App Protect 정책은 OWASP Top 10, 크로스 사이트 스크립팅(XSS), injection, 회피 기법, 정보 유출(Data Guard 포함) 등 다양한 유형의 위협으로부터 웹 애플리케이션을 보호합니다. 아래 샘플 APPolicy 커스텀 리소스는 차단 모드에서 Data Guard 위반을 활성화합니다.

```
apiVersion: appprotect.f5.com/v1beta1
kind: APPolicy
metadata:
  name: dataguard-alarm
spec:
  policy:
    applicationLanguage: utf-8
    blocking-settings:
    violations:
      - alarm: true
        block: true
        name: VIOL_DATA_GUARD
```

(이어서)

```
    data-guard:
      creditCardNumbers: true
      enabled: true
      enforcementMode: ignore-urls-in-list
      maskData: true
      usSocialSecurityNumbers: true
    enforcementMode: blocking
    name: dataguard-alarm
    template:
    name: POLICY_TEMPLATE_NGINX_BASE
```

로깅

App Protect와 NGINX Plus Ingress Controller의 로그는 보안 팀이 일반적으로 DevOps와 애플리케이션 소유자와 독립적으로 운영되는 방식을 반영하기 위해 분리되어 있습니다. 매개 변수를 syslog Pod의 클러스터 IP 주소에 대한 app-protect-security-log-destination Annotation으로 설정해 Kubernetes Pod에서 연결할 수 있는 모든 syslog 목적지로 App Protect 로그를 전송할 수 있습니다 (위 Ingress 리소스를 예시로 참고하십시오). 또한, APLogConf 리소스를 사용해 관심 있는 App Protect 로그를 지정하고, 어떤 로그가 syslog Pod에 푸시되는지 암시적으로 지정할 수 있습니다. NGINX Plus Ingress Controller 로그는 모든 Kubernetes 컨테이너와 마찬가지로 로컬 표준 출력으로 전달됩니다.

리소스 임계값

마지막으로 NGINX Plus Ingress Controller의 NGINX App Protect는 App Protect 프로세스의 CPU와 메모리 사용량 모두에 대해 설정 가능한 리소스 보호 임계값을 제공함으로써 다른 프로세스에서 메모리 부족이 발생하지 않도록 합니다. 이는 리소스 공유에 의존하고 잠재적으로 '노이지 네이버(noisy neighbor)' 문제로 어려움을 겪을 수 있는 Kubernetes와 같은 다중 테넌트 환경에서 특히 중요합니다. 다음 샘플 ConfigMap은 App Protect 프로세스에 대한 리소스 임계값을 설정하는 것입니다.

```
kind: ConfigMap
apiVersion: v1
metadata:
  name: nginx-config
  namespace: nginx-ingress
data:
  app_protect_physical_memory_util_thresholds: "high=100 low=10"
  app_protect_cpu_thresholds: "high=100 low=50"
  app_protect_failure_mode_action: "drop"
```

높은 임계값(high)은 App Protect가 failure 모드로 들어가는 사용률을 설정하고, 낮은 임계값(low)은 failure 모드를 종료하는 사용률을 설정합니다. 메모리 사용률은 각각 100%와 10%이며, CPU의 경우 100%와 50%입니다. app_protect_failure_mode_action의 값 하강은 App Protect가 연결을 닫아, failure 모드에 있는 동안 트래픽을 거부함을 의미합니다.

NGINX Plus Ingress Controller의 NGINX App Protect 구성과 문제 해결에 관한 자세한 내용은 Ingress Controller 설명서를 참조하십시오. 다른 App Protect 사용 사례에 대한 상세한 내용은 NGINX App Protect 설명서를 참조하십시오.

미래의 통합

릴리스 1.8.0의 Ingress 리소스 구성은 주석(Annotations)을 사용해 App Protect 정책을 참조합니다. 현재는 검사할 요청과 검사하지 않는 요청을 이상적으로 세분화해 제어하는 기능을 아직 제공하지 않습니다.

NGINX Plus Ingress Controller의 향후 릴리스에서는 NGINX Ingress 리소스에 통합된 보다 상세하고 사용자 지정 가능한 구성을 볼 수 있습니다. 이는 요청에 WAF 정책을 적용하는 방식을 추가로 제어할 수 있도록 할 것입니다.

요약

모던 컨테이너형 애플리케이션의 경우, 모든 ingress 트래픽("north–south")은 신뢰할 수 없는 반면, 내부적으로 생성된 트래픽("east–west")은 올바르게 구성되고 신뢰할 수 있다고 가정하는 것이 무방합니다. 이 경우 Ingress Controller는 WAF와 같은 보안 프록시에 이상적인 위치입니다.

NGINX Plus Ingress Controller와 NGINX App Protect는 완벽하게 지원되는 WAF를 통합하는 유일한 Ingress Controller 구현입니다. Ingress Controller에 WAF를 내장하면 데이터 플레인 장치를 하나로 통합하고 구성에 Kubernetes API를 활용함으로써 효율성을 더욱 향상시킬 수 있습니다.

8. NGINX App Protect로 애플리케이션 보호

오늘날 애플리케이션 환경은 급격하게 변화했습니다. 모던 애플리케이션은 컨테이너로 실행되는 마이크로서비스이며 API를 통해 통신하며 자동화된 CI/CD 파이프라인을 통해 배포됩니다.

DevOps 팀은 릴리스 속도나 성능을 저하시키지 않으면서 분산된 환경 전반에 보안 팀이 승인한 보안 컨트롤을 통합해야 합니다. NGINX App Protect는 애플리케이션이 코드 단계에서 고객에게 제공될 때까지 DevOps 환경에서 원활하게 실행되도록 설계된 모던 애플리케이션 보안 솔루션입니다.

지금 NGINX App Protect와 NGINX Plus 30일 무료 평가판을 시작하거나, F5에 연락하여 활용 사례를 요청하십시오. 제품 설명서를 읽고 전체 F5 웹 애플리케이션과 API 보호 솔루션에 대해 자세히 알아볼 수도 있습니다.