Special Report

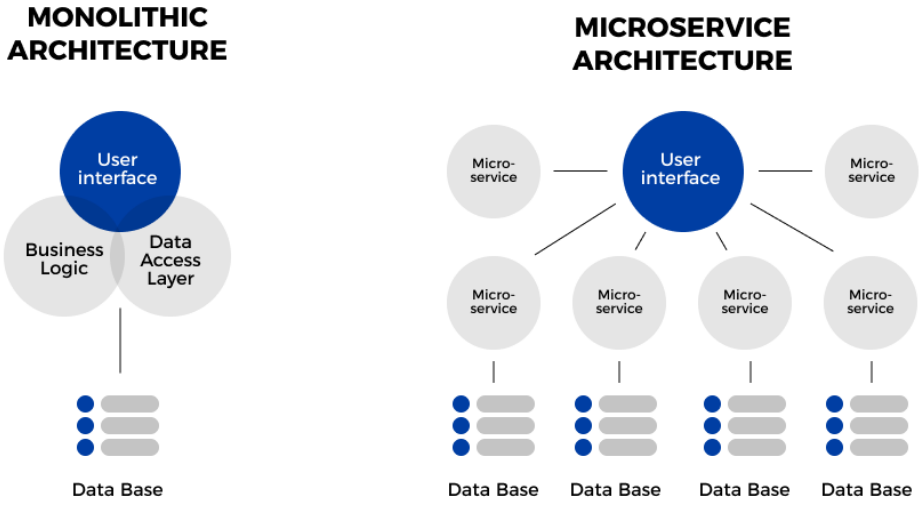
제로트러스트 보안전략 : 애플리케이션 및 워크로드 (Application&Wokrdload)

SI/솔루션사업그룹 보안 SI 사업팀 황병권 책임

■ 애플리케이션 및 워크로드 (Application&Wokrdload) 필러 개요

제로트러스트 아키텍처에서 어플리케이션 및 워크로드 필러는 기업 망의 관리 시스템, 프로그램 등 온프레미스 및 클라우드 환경에서 실행되는 모든 서비스를 포괄한다. 이는 조직의 비즈니스 로직과 데이터가 실제로 실행·처리되는 동적인 영역으로, 컨테이너나 가상머신(VM)과 같은 개별 실행 단위 보호 및 데이터의 안전한 전달 보장을 목표로 한다. NIST 가이드라인에서는 제로트러스트의 핵심을 개별 '리소스(Resource)' 보호에 두고 있으며, 애플리케이션과 워크로드는 가장 중요한 리소스에 해당한다.

디지털 전환이 가속화되면서 애플리케이션 아키텍처는 근본적으로 변화했다. 과거의 거대한 단일 애플리케이션(Monolithic)은 독립적으로 실행되었으나, 현재는 통신하는 수많은 작은 서비스의 집합(Microservices Architecture, MSA)으로 진화했다. 이로 인해 애플리케이션의 경계는 기존의 데이터센터에서 퍼블릭·프라이빗 클라우드, 하이브리드 환경으로 확장되었으며, 수많은 API 가 새로운 통신 통로이자 공격 표면이 되었다. 또한, 개발과 운영이 통합된 DevOps 문화와 CI/CD 파이프라인의 도입은 애플리케이션의 배포 속도를 획기적으로 높였지만, 동시에 보안이 고려되지 않은 코드가 운영 환경에 빠르게 배포될 수 있는 새로운 위험을 만들었다.



출처 : Cloudflight, “Monolithic Architecture vs Microservices Architecture”

그림 1. 어플리케이션 관리의 변화

이러한 환경에서 웹 방화벽(WAF)와 DDoS 장비 같은 전통적인 경계 기반 보안 모델은 어플리케이션 관리의 명백한 한계를 드러낸다. 경계 기반 보안 모델은 어플리케이션 공격 시 주로 외부에서 내부로 들어오는 공격을 통제하는 데 중점을 둔다. 클라우드 환경 내부, 쿠버네티스 클러스터 내부 등 서비스 간에 발생하는 어플리케이션 통신에 대한 가시성과 통제력을 경계하기는 어렵다. 경계를 통과한 위협이 내부망 안에서 자유롭게 확산(Lateral Movement)하는 것을 막기 어려운 구조이다. 결국, 암묵적인 신뢰에 기반한 기존의 방식으로는 동적으로 변화하고 분산된 현대의 애플리케이션과 워크로드를 더 이상 효과적으로 보호할 수 없게 된 것이다.

미국의 CISA 가 발표한 제로 트러스트 성숙도 모델 역시 이러한 변화를 반영하여 '어플리케이션 및 워크로드'를 핵심 필러로 지정하고, 성숙도 진화 경로를 제시한다. 이 모델에 따르면, 성숙한 제로트러스트 환경은 단순히 경계를 방어하는 수준을 넘어, 애플리케이션 수준의 마이크로세그멘테이션을 통해 워크로드 간의 모든 통신을 제어하고, 보안 SDLC 를 구현하여 개발 생명주기 전반에 보안을 내재화해야 한다고 강조한다. 국내 KISA 의 제로 트러스트 가이드라인 또한 어플리케이션 및 워크로드를 주요 필러로 정의하며, 국내 IT 환경과 규제에 맞는 보안 체계 구축의 중요성을 반복해서 강조하고 있다.

따라서 제로트러스트 환경에서 어플리케이션 및 워크로드 필러는 개별 애플리케이션의 보안을 강화하는 수준을 넘어, 서로 다른 환경에서 운영되는 서비스들을 일관된 정책과 중앙화된 가시성을 통해 통합 관리하는 데 핵심적인 의미가 있다. 이는 소프트웨어 개발 생명주기(SDLC) 초기 단계부터 보안을 내재화하는 'Shift Left' 접근을 통해 애플리케이션 자체의 신뢰도를 확보하고, 접근하는 주체의 컨텍스트와 API 호출의 적정성 등 모든 상호작용을 지속적으로 검증하여, 조직의 핵심 자산과 서비스를 안전하게 보호하는 실질적인 통제 체계를 구현하는 것을 목표로 해야 한다.

■ 애플리케이션 및 워크로드 (Application&Wokrdload) 필러의 주요 요소

애플리케이션 및 워크로드 필러는 제로트러스트 아키텍처 내에서 조직의 비즈니스 로직이 실행되고 데이터가 실질적으로 처리되는 동적인 영역을 보호하는 데 중점을 둔다. 온프레미스 데이터센터를 넘어 클라우드, 컨테이너, 서버리스 등 다양한 환경에 분산된 애플리케이션과 API 는 과거 경계 기반 보안 모델로는 더 이상 효과적으로 통제하기 어렵다.

특히 제로트러스트 환경에서는 애플리케이션의 위치가 아닌, 애플리케이션 자체의 신뢰도와 접근하는 주체, API 호출의 적정성 등 모든 상호작용을 지속적으로 검증해야 한다. 이를 위해 애플리케이션 인벤토리, 위험 관리, 접근 관리, 보안 테스트, 소프트웨어 개발 및 통합, 정책 및 통합 등 애플리케이션 필러 기준의 여러 관리적·기술적 요소들이 상호 유기적으로 결합되어야만, 조직의 핵심 비즈니스 로직과 데이터를 안전하게 보호하고 서비스의 무결성을 확보할 수 있다.

아래는 애플리케이션 필러의 주요 요소들과, 이를 구현하기 위한 구체적인 관리·기술 방안을 제로트러스트 성숙도 관점에서 정리한 내용이다.

1. 애플리케이션 인벤토리

제로트러스트 환경에서 애플리케이션 인벤토리 관리는 조직 내에서 운영되는 모든 응용 프로그램, 조직에서 공급하는 응용 프로그램, API 등을 식별하고 그 현황을 항상 최신으로 유지하는 애플리케이션 보안 관리의 출발점이다. 관리 범위는 자체적으로 개발한 업무 시스템이 운영되는 온프레미스 환경뿐만 아니라, 클라우드 서비스, SaaS 등을 포괄하며, 이들 모두를 통합된 자산관리 체계 내에서 목록화해야 한다. 이는 목록을 단순히 유지하는 수준을 넘어, CI/CD 파이프라인과 연계하여 애플리케이션의 생성, 배포, 변경, 폐기에 이르는 전체 생명주기에 걸쳐 주요 속성과 상태 정보가 실시간으로 갱신되는 동적 인벤토리 관리 체계를 의미한다.

정확한 인벤토리 관리를 위해서는 모든 애플리케이션에 대해 책임과 역할을 명확히 하는 소유자 관리가 이루어져야 하며, 비즈니스 영향도와 처리하는 정보의 민감도를 기준으로 중요도 관리 체계가 수립되어야 한다. 이렇게 정의된 소유자 및 중요도 정보는 향후 위험 평가, 접근 권한 부여, 취약점 조치 우선순위 결정 등 보안 활동의 객관적인 기준으로 활용된다. 성숙도가 높은 조직에서는 이러한 중요도에 따라 패치 확인이나 소스코드 점검과 같은 보안 활동이 자동으로 수행될 수 있다.

최적화된 애플리케이션 인벤토리 관리체계는 수동 관리를 벗어나 관리 시스템을 통해 자동화되며, 최종적으로는 모든 애플리케이션의 변화가 정책에 따라 시스템에 자동 등록되고 모니터링 시스템과 연동되어 실시간 가시성을 제공해야 한다. 이렇게 확보된 정보는 정교한 접근 제어, 신속한 취약점 관리, 효율적인 보안 정책 배포를 가능하게 함으로써 제로트러스트 실현의 핵심 기반으로 작동한다.

2. 애플리케이션 위험 관리

제로트러스트 환경에서 애플리케이션 위험 관리는 소프트웨어에 내재된 위험 요소를 지속적으로 식별하고 조치하여 공격 표면을 최소화하는 핵심적인 활동이다. 공격자가 악용할 경우, 애플리케이션에 존재하는 취약점을 이용해 정보 유출, 권한 상승과 같은 심각한 침해사고로 이어질 수 있으며, 널리 사용되는 오픈소스 라이브러리에 포함된 보안 약점 또한 조직 전체를 위험에 빠뜨릴 수 있다. 따라서 개발 단계부터 운영에 이르기까지 전 과정에 걸쳐 잠재적 위험을 체계적으로 관리하는 것이 필수적이다.

취약점 조치는 운영 전환 이전에 개발 애플리케이션을 점검하는 수준으로는 한계가 있다. 성숙한 제로트러스트 환경에서는 소스코드 점검을 포함하여 개발 애플리케이션의 취약점을 지속적으로 관리하며, 운영 중에도 정기적인 점검을 통해 발견된 취약점을 자동화된 프로세스로 처리한다. 또한, 직접 수정이 어려운 상용 애플리케이션의 경우, 취약점 발견 시 IAM, Micro-Segmentation 등과 같은 시스템과 연계해 해당 애플리케이션의 계정과 권한을 회수하거나, 애플리케이션을 격리하는 등 보완 통제를 적용하여 위험을 완화해야 한다.

오픈소스 관리 또한 현대 애플리케이션 개발 환경에서 매우 중요하다. 오픈소스는 빠른 개발과 협업을 가능하게 하지만, 라이선스 이슈나 잠재적인 보안 위협을 내포할 수 있다. 초기에는 수동으로 라이선스와 취약점을 관리할 수 있으나, 점차 오픈소스 통합 관리 시스템을 통해 라이선스, SBoM(Software Bill of Materials), 외부 취약점 DB 등과 연계해 체계적으로 운영해야 한다. 최적화된 단계에서는 머신러닝을 통해 취약점을 점검하고 소스코드 수정을 자동화하여 관리함으로써, 오픈소스로부터 발생하는 위협을 선제적으로 통제할 수 있다.

3. 애플리케이션 접근 관리

제로트러스트 환경에서 애플리케이션 접근 관리는 단순히 로그인 성공 여부를 판단하는 것을 넘어, 인증된 사용자가 허용된 범위 내에서만 작업을 수행하도록 지속적으로 통제하고 모든 활동을 기록하여 위협을 차단하는 포괄적인 과정을 의미한다. 이는 애플리케이션에 대한 인증을 강화하고, 역할에 기반한 세밀한 권한을 부여하며, 모든 접근 이력을 관리하고, 애플리케이션 간 불필요한 통로를 차단하여 잠재적인 횡적 이동을 방지하는 것을 포함한다.

애플리케이션 인증 및 접근 관리는 기존의 1 차 인증만 수행하는 단계에 그치지 않고, 사용자의 역할과 부서 등 컨텍스트를 연동하여 메뉴 접근부터 조회, 수정, 삭제와 같은 세부 기능까지 권한을 분류하여 관리하는 방향으로 나아가야 한다. 일반적으로 SSO 나 정보보안포탈을 통해 구현되며 최적화된 제로트러스트 환경에서는 SDP 와 같은 기술로 사용자가 애플리케이션 서버에 직접 접근하기 전에 컨트롤러에서 보안 패킷과 단말 정보를 먼저 검증하는 선 인증 접속 방식을 도입하면 보안성을 근본적으로 강화할 수 있다.

이러한 모든 통제는 투명한 애플리케이션 접근 이력 관리를 통해 완성된다. 초기에는 애플리케이션 자체적으로 로그인 정보만 관리하는 수준에 머무를 수 있으나, 점차 SSO 로그인을 포함한 메뉴 조회, 데이터 생성·수정·삭제(CRUD) 이력까지 개별적으로 관리해야 한다. 궁극적으로는 통합 접근 관리 시스템을 활용하여 모든 애플리케이션의 접근 통제 이력을 중앙에서 통합 관리함으로써, 전사적인 가시성을 확보하고 신속한 위협 추적 및 분석을 지원해야 한다.

더 나아가, 제로트러스트는 애플리케이션 간 횡적 확산 및 우회 접속 차단을 매우 중요하게 다룬다. 초기 대응으로는 연동 API 나 소스코드 개발 시 취약점을 제거하고 DB Link 사용을 금지하는 정책을 적용할 수 있다. 하지만 보다 근본적인 통제를 위해서는 시스템 내에서 애플리케이션 단위로 개별 방화벽을 설정하는 것과 같은 Micro-Segmentation 을 적용하여 불필요한 횡적 확산과 우회 접속 경로를 원천적으로 차단해야 한다. 최적화된 환경에서는 이를 통합 모니터링 시스템과 연동하여 횡적 이동 시도와 같은 이상 행위를 지속적으로 탐지하고 대응하는 체계를 갖추게 된다.

4. 애플리케이션 보안 테스트

제로트러스트 환경에서 애플리케이션 보안 테스트는 개발 생명주기 초기에 보안을 내재화하는 'Shift Left' 개념을 실현하는 핵심적인 활동이다. 단순히 운영 단계의 보안에만 의존하는 것이 아니라, 개발 과정에서부터 코드 레벨의 취약점을 찾아 수정하고, 런타임 환경에서 발생할 수 있는 잠재적 위협을 사전에 식별하여 제거함으로써 애플리케이션 자체의 신뢰도를 확보하는 것을 목표로 한다.

정적 애플리케이션 보안 테스트(SAST)는 개발자의 관점에서 애플리케이션의 소스 코드, 바이너리, 바이트 코드를 직접 분석하여 SQL 인젝션이나 크로스 사이트 스크립팅(XSS)과 같은 코드 기반의 보안 취약점을 식별하는 '화이트박스' 방식이다. 초기에는 중요 애플리케이션을 대상으로 운영 전환 시점에만 수동으로 점검을 수행할 수 있었다. 이후, 성숙한 조직은 개발 생명주기 전반에 걸쳐 주기적인 점검을 자동화하고, 코딩과 설계 단계부터 보안 결함을 검증하여 개발 초기부터 보안을 보장하는 체계를 갖추어야 한다.

이와 상호 보완적으로 동적 애플리케이션 보안 테스트(DAST)는 실제 운영 중인 애플리케이션을 외부 공격자의 관점에서 분석하는 '블랙박스' 방식이다. 소스 코드 접근 없이 실제 요청과 응답을 분석하여 세션 관리 문제나 서버 설정 오류와 같은 런타임 환경의 취약점을 파악하는 데 효과적이다. 성숙도가 높아질수록 주기적인 분석을 넘어, 외부 공격을 지속적으로 시뮬레이션하고 테스트하며 애플리케이션의 보안 상태를 상시적으로 검증하는 단계로 발전해야 한다.

최근에는 SAST 와 DAST 뿐만 아니라, 오픈소스 라이브러리의 취약점을 분석하는 SCA(Software Composition Analysis, 소프트웨어 구성 분석)까지 애플리케이션 보안 테스트의 필수 요소로 여겨지고 있다. 이 세 가지 핵심 테스트 방식(SAST, DAST, SCA)을 개별적으로 운영하기보다는, 이들을 모두 지원하는 통합 애플리케이션 보안 테스트 플랫폼을 통해 SDLC(소프트웨어 생명주기) 정책에 반영하고 CI/CD 파이프라인에 보안을 완벽하게 내재화해 개발부터 배포, 운영까지 전 과정에 걸쳐 일관된 보안 관리를 수행하는 것이 최신 동향이다.

5. 리소스 승인 및 통합

제로트러스트 환경에서 리소스 승인 및 통합은 사용자가 인증 후에 애플리케이션 내부의 특정 기능이나 데이터에 접근할 때, 사전에 정의되고 승인된 권한만을 부여하는 핵심적인 통제 활동이다. 이는 단순히 로그인 성공 여부로 모든 기능을 허용하는 것이 아니라, 최소 권한 원칙에 따라 사용자의 역할과 책임에 맞는 최소한의 리소스 접근만을 허용함으로써 내부 위협과 권한 오남용의 위험을 크게 줄이는 것을 목표로 한다.

애플리케이션 승인 절차는 조직의 제로트러스트 성숙도에 따라 발전한다. 초기 단계에서는 별도의 리소스 승인 없이 로그인만으로 애플리케이션의 모든 기능에 접근하거나, 일부 애플리케이션에 한해 제한적인 접근 권한을 부여하는 수준에 머무를 수 있다. 성숙한 단계로 나아가기 위해서는 개별 애플리케이션별로 화면, 기능, 배치 작업 등 각각의 리소스에 대해 정식 승인 절차를 거쳐 접근 권한을 부여하는 체계를 갖추어야 한다.

최적화된 제로트러스트 환경에서는 이러한 승인 절차를 부서, 역할, 사용자별로 더욱 세분화하고, 모든 애플리케이션에 걸쳐 일관된 정책으로 통합 관리한다. 이를 통해 특정 사용자는 어떤 애플리케이션을 사용하더라도 자신의 역할에 맞는 특정 기능과 데이터에만 접근할 수 있게 되며, 이는 중앙화된 접근 관리 시스템(IAM, ICAM)을 통해 체계적으로 통제된다.

6. 소프트웨어 개발 및 통합

제로트러스트 환경에서 소프트웨어 개발 및 통합은 보안을 개발 생명주기의 가장 첫 단계부터 내재화하는 'Shift Left' 원칙을 적용하는 핵심적인 과정이다. 이는 단순히 완성된 애플리케이션의 취약점을 점검하는 것을 넘어, 코드를 작성하는 순간부터 안전한 코딩 규칙을 준수하고, 개발 과정에서 활용되는 외부 플랫폼과 오픈소스의 위협까지 체계적으로 관리하여 잠재적 위험 요소를 원천적으로 제거하는 것을 목표로 한다.

시큐어 코딩은 개발 과정에서 발생할 수 있는 보안 취약점을 최소화하기 위한 일련의 보안 활동을 의미하며, 안전한 소프트웨어를 개발하기 위해 지켜야 할 코딩 규칙과 소스코드 취약점 목록을 포함한다. 초기에는 개발자 대상의 교육과 자료 전달에 의존할 수 있으나, 보다 성숙한 단계에서는 시큐어 코딩 점검 시스템을 통해 개발 단계별로 코딩 취약점을 점검하고 조치해야 한다. 최적화된 환경은 개발자의 보안 인식을 높이기 위한 지속적인 교육과 함께, 기술적으로 시스템을 활용한 개발 단계별 점검 및 조치를 병행하여 문화와 기술 양측면에서 안전한 개발 환경을 보장할 수 있어야 한다.

또한, 현대 개발 환경은 깃허브(GitHub)와 같은 외부 소스코드 호스팅 서비스 및 오픈소스 활용이 필수적이므로, 이에 대한 위협 여부 탐지 체계를 갖추는 것이 매우 중요하다. 이러한 외부 플랫폼은 해커의 공격 발판으로 악용되거나 플랫폼 자체의 취약점에 노출될 수 있다. 단순히 외부 개발 환경 사용을 금지하는 정책은 비공식적인 사용(Shadow IT)을 낳을 수 있으므로 효과적이지 않다. 따라서 주기적으로 오픈소스 통제와 라이선스를 관리하는 단계를 거쳐, 최종적으로는 주기적인 취약점 통제 및 조치와 함께 실시간 모니터링을 수행하여 외부 개발 환경으로부터 유입될 수 있는 위협에 선제적으로 대응해야 한다.

7. 소프트웨어 개발 생명주기(SDLC) 및 지속적인 통합과 배포(CI/CD)

제로트러스트 환경에서 CI/CD(지속적 통합/지속적 배포) 파이프라인은 단순히 개발 생산성을 높이는 자동화 도구를 넘어, 조직의 보안 소프트웨어 개발 생명주기(SDLC) 정책을 체계적으로 이행하고 강제하는 핵심 프레임워크 역할을 수행한다. 이는 개발자가 코드를 공유 저장소에 병합하는 순간부터 빌드, 테스트, 배포, 그리고 운영에 이르기까지 전 과정에 걸쳐 보안을 내재화하고, 수동 개입으로 인한 오류와 보안 공백을 최소화하여 신뢰할 수 있는 소프트웨어를 지속적으로 제공하는 것을 목표로 한다.

이러한 보안 SDLC 의 첫 단계는 안전한 코드 통합에서 시작된다. 초기에는 관리자가 여러 개발자의 코드를 수동으로 점검하고 빌드할 수 있으나, 성숙한 조직은 CI 서버를 통해 코드 통합 프로세스를 자동화하고, 코드 검사 도구와 테스트 프레임워크를 연동하여 코드 병합 과정에서부터 보안 검증을 자동으로 수행한다. 이어서 보안 빌드 프로세스는 통합된 코드를 검사하고 오픈소스 라이브러리 취약점 및 SQL 인젝션과 같은 공격을 사전에 탐지하는 단계를 거친다. 최적화된 환경에서는 빌드의 결과물인 아티팩트에 GPG 서명과 같은 암호화 기술을 적용하여 위변조를 방지하고, 배포 전 서명 검증을 통해 무결성을 보장한다.

이렇게 생성된 결과물은 보안 아티팩트 관리 체계에 따라 안전하게 관리되어야 한다. 단순히 특정 매체에 저장하는 단계를 넘어, 전용 시스템을 통해 버전 관리, 서명 검증, 접근 제어를 수행하고, 최종적으로는 암호화된 보안 저장소에서 체계적으로 관리되어야 한다. 보안 배포 프로세스 역시 수동 배포의 위험성을 제거하고 자동화를 지향한다. 성숙한 파이프라인은 배포 전 변경 영향 분석과 보안 취약점 평가를 포함한 공식적인 검토 및 승인 프로세스를 거치며, 승인된 빌드는 아티팩트의 서명을 검증한 뒤 배포 자동화 도구로 자동 배포된다.

소프트웨어 배포 이후에도 제로트러스트 원칙은 지속적인 모니터링 및 감사를 통해 유지된다. 문제 발생 시 수동으로 로그를 분석하는 단계를 벗어나, SIEM 과 같은 시스템과 연동하여 로그를 중앙에서 수집하고 위협을 식별하며, 최적화된 단계에서는 머신러닝과 AI 를 활용하여 잠재적 위협을 예측하고 선제적으로 조치한다. 마지막으로, 이 모든 기술적 통제는 사람, 즉 개발자를 위한 보안 교육으로 완성된다. 조직 내 보안 인식 문화를 촉진하기 위해 주기적인 교육을 실시하고, 이를 자동화된 시스템으로 관리하며, 보안 교육을 수료한 개발자만이 개발에 참여하도록 통제하는 것이 가장 성숙한 보안 SDLC 의 모습이다.

8. 클라우드 워크로드 보호

제로트러스트 환경에서 클라우드 워크로드 보호는 동적으로 생성·변경·소멸하는 클라우드 자원(가상머신, 컨테이너 등)의 특성을 이해하고, 애플리케이션과 데이터가 실행되는 전 과정을 보호하는 것을 목표로 한다. 온프레미스와 달리 물리적 통제가 어려운 클라우드 환경에서는 구성 오류, 취약점, 시스템 무결성 훼손, 악성코드 감염 등이 더 심각한 위협으로 작용할 수 있으므로, 다층적이고 자동화된 보안 체계가 필수적이다.

워크로드 보호의 기반은 강화, 구성, 취약점 관리에서 시작된다. 초기에는 온프레미스에서 사용하던 점검 도구를 활용할 수 있으나, 성숙한 단계에서는 여러 클라우드 환경을 단일 콘솔에서 통합 관리하며 워크로드의 시스템 구성과 애플리케이션/운영체제 취약점을 지속적으로 점검하고 확인해야 한다. 또한, 클라우드 네트워크 방화벽, 가시성, 세분화를 통해 기존의 물리적 네트워크 관리에서 벗어나 SDN(소프트웨어 정의 네트워크)을 기반으로 워크로드 단위의 방화벽 기능을 제공하고, 이를 통해 각 워크로드를 외부 위협으로부터 보호하고 모니터링해야 한다.

더 나아가 시스템 무결성 보증을 위해 워크로드의 환경 설정 및 구성, 권한 등을 실시간으로 모니터링하고, 이상 발생 시 자동으로 조치하는 체계를 갖추어야 한다. 애플리케이션 제어 및 허용 목록 관리 역시 중요하다. 단순히 블랙리스트 기반으로 차단하는 것을 넘어, 중앙 관리 시스템을 통해 화이트리스트 기반으로 허용된 애플리케이션만 실행되도록 제어하고, 승인 절차와 연동하여 허가된 애플리케이션을 자동으로 적용하는 것이 바람직하다.

이러한 기반 위에 악용 방지 및 메모리 보호, 호스트 기반 침입 방지, 멀웨어 방지 스캐닝과 같은 능동적인 위협 대응 기술이 적용되어야 한다. 성숙한 제로트러스트 환경에서는 알려진 위협은 물론, 알려지지 않은 위협과 파일리스 공격까지 방어하기 위해 머신러닝과 샌드박스 기술을 활용하며, 위협 탐지 시 대상을 자동으로 격리하는 등 자동화된 대응을 수행한다. 이 모든 활동의 중심에는 엔드포인트 감지 및 대응(EDR)과 행동 모니터링이 있다. 개별 백신 프로그램을 넘어, 중앙 관리 시스템을 통해 클라우드 환경 내 모든 워크로드의 행위를 모니터링하고, 위협 발생 시 삭제, 격리, 차단 등의 조치를 자동으로 수행하여 클라우드 환경 전반의 보안을 완성한다.

9. SaaS 관리 플랫폼(SMP)

제로트러스트 환경에서 SaaS 관리 플랫폼(SMP, SaaS Management Platform)은 조직 내에서 사용하는 모든 서비스형 소프트웨어(SaaS) 애플리케이션을 중앙에서 통합적으로 관리하고 운영하는 역할을 수행한다. 클라우드 기반 SaaS 애플리케이션의 도입이 증가하면서 관리되지 않는 Saas 가 늘어나고, 개별 서비스마다 운영 및 보안 정책이 분산되어 일관성을 유지하기 어려워지는 문제가 발생한다. SMP 는 이러한 문제를 해결하기 위해 모든 SaaS 애플리케이션의 사용 현황, 관리, 보안 정보를 단일 대시보드에 집계하여 제공한다.

성숙한 제로트러스트 환경에서의 체계적인 SaaS 관리는 단순히 사용 현황을 추적하는 것에서 그치지 않는다. SMP 는 IAM 시스템 등과 연동하여 현재 사용 중인 모든 SaaS 애플리케이션을 추적하고, 누가 어떤 애플리케이션을 얼마나 자주 사용하는지에 대한 정보를 집계한다. 최적화된 단계에서는 라이선스 관리, 사용자 온보딩 및 오프보딩, 애플리케이션 내 사용자 그룹 관리와 같은 운영 작업을 중앙에서 직접 수행하여 관리 효율성을 극대화한다.

또한, SMP 는 보안 및 규정 준수를 위한 중앙 제어 센터로 기능한다. 개별 SaaS 애플리케이션에 일일이 접속하여 보안 설정을 관리하는 대신, SMP 를 통해 데이터 보호, 접근 제어 및 기타 보안 설정을 중앙에서 일괄적으로 적용하고 관리할 수 있다. 이를 통해 조직의 보안 규정과 지침을 실시간으로 모든 SaaS 애플리케이션에 최신화하고 일관되게 운영함으로써, 분산된 클라우드 환경 전반에 제로트러스트 원칙을 효과적으로 확장할 수 있다.

최근에는 이러한 SaaS 관리에 SMP 보안 기능을 포함하여, 더 넓은 클라우드 보안 프레임워크를 구현하는 추세가 나타나고 있다. 예를 들어, SASE(보안 액세스 서비스 엣지) 와 같은 통합 보안 시스템은 내재된 CASB(클라우드 접근 보안 브로커) 기능을 통해 SaaS 가시성을 확보하고 데이터 보안 정책을 적용하는 역할을 수행한다. 더 나아가 성숙한 클라우드 보안 전략은 IaaS, PaaS 인프라의 형상을 관리하는 CSPM(클라우드 보안 형상 관리), 가상머신이나 컨테이너 등 실제 워크로드를 보호하는 CWPP(클라우드 워크로드 보호 플랫폼) 기능까지 통합하여, 인프라부터 워크로드, SaaS 애플리케이션 접근에 이르기까지 일관된 제로트러스트 보안을 적용한다.

10. 보안 액세스 서비스 엣지(SASE)

제로트러스트 아키텍처에서 SASE(Secure Access Service Edge)는 특정 시스템이나 솔루션을 지칭하기도 하지만, 개념적으로 이해하면 분산된 사용자와 클라우드 중심의 업무 환경에 맞춰 네트워크와 보안을 클라우드 기반의 단일 서비스로 통합하여 제공하는 보안 프레임워크로 이해해야 한다. 즉, 제로트러스트를 효과적으로 구현하기 위해 클라우드 엣지에서 반드시 수행되어야 할 핵심 기능들의 집합이다. SASE 는 사용자의 위치에 상관없이 모든 리소스에 대해 일관되고 강력한 보안을 적용하는 것을 목표로 한다.

SASE 프레임워크의 네트워킹 기반은 고급 SD-WAN 기능으로, 하드웨어에서 추상화된 가상 네트워크 오버레이를 생성하여 분산된 조직 환경에서도 데이터센터 및 클라우드 애플리케이션에 안정적이고 빠른 연결을 보장한다. 제로트러스트 관점에서 잘 구성된 SD-WAN 은 트래픽을 지능적으로 라우팅하여 모든 연결이 보안 검사를 위해 SASE 의 클라우드 인프라를 거치도록 하는 토대를 마련한다.

SASE 프레임워크의 네트워킹 기반은 고급 SD-WAN 기능으로, 하드웨어에서 추상화된 가상 네트워크 오버레이를 생성하여 분산된 조직 환경에서도 데이터센터 및 클라우드 애플리케이션에 안정적이고 빠른 연결을 보장한다. 제로트러스트 관점에서 잘 구성된 SD-WAN 은 트래픽을 지능적으로 라우팅하여 모든 연결이 보안 검사를 위해 SASE 의 클라우드 인프라를 거치도록 하는 토대를 마련한다.

SASE 의 핵심 접근 제어 모델은 제로 트러스트 네트워크 액세스(ZTNA) 이다. 이는 기존 VPN 을 대체하는 기술로, 어떤 사용자나 기기도 기본적으로 신뢰하지 않는다는 원칙하에 최소 권한 액세스를 지원한다. 성숙한 ZTNA 는 단순히 초기 접근 인증에서 그치지 않고, 접속 후에도 사용자의 행위에서 이상 징후가 탐지되면 자동으로 추가 인증을 요구하거나 세션을 차단하는 등 지속적인 검증을 수행한다.

또한 SASE 는 클라우드 및 웹 트래픽을 보호하기 위한 다양한 보안 기능을 통합한다. 대표적인 기능은 CASB(클라우드 접근 보안 브로커) 보안 정책을 통해 클라우드 애플리케이션의 가시성을 확보하고 민감 데이터를 보호하는 기능이다. 해당 기능은 아래 별도의 항목에서 조금 더 자세히 다룰 예정이다.

보안 웹 게이트웨이(SWG)는 웹으로 향하는 모든 사용자 트래픽을 검사하여 멀웨어나 악성코드를 필터링하고 조직의 보안 정책을 강제하는 역할을 한다. 성숙한 SWG 는 단순 URL 필터링을 넘어, 실시간 위협 인텔리전스 DB 와 연동하여 알려지지 않은 위협까지 차단하고, 맬웨어 방지 스캔, 애플리케이션 제어 기능까지 통합하여 포괄적인 웹 위협 방어 체계를 구축한다.

서비스형 방화벽(FWaaS)은 클라우드에서 제공되는 차세대 방화벽(NGFW) 기능이다. 이는 URL 필터링, IPS, DNS 보안과 같은 고급 L7 제어 기능을 포함하며, 물리적인 방화벽 장비 없이도 모든 네트워크 경계를 사이버 위협으로부터 보호한다. 성숙한 SASE 프레임워크 기반의 FWaaS 는 일부 중요 서버만이 아닌, 조직의 전체 네트워크 경계를 포괄적으로 보호하는 역할을 수행한다. 이처럼 SASE 는 여러 핵심 보안 기능들을 유기적으로 결합하여 제로트러스트 원칙을 네트워크 엣지 단에서 실현하는 통합 프레임워크로 기능한다.

11. 클라우드 엑세스 보안 브로커(CASB)

CASB(클라우드 액세스 보안 브로커)는 클라우드 서비스 사용자와 클라우드 서비스 제공자 사이에 위치하여, 온프레미스 기반의 조직의 보안 정책을 클라우드 기반 리소스까지 확장하고 일관되게 적용하는 핵심적인 보안 정책 시행 지점이다. 제로트러스트 환경에서 CASB 는 조직이 통제하기 어려운 외부 클라우드 서비스에 대한 가시성을 확보하고, 데이터 보안과 위협 방지, 컴플라이언스 준수를 가능하게 하는 중추적인 역할을 수행한다.

CASB 의 가장 기본적인 기능은 클라우드 가시성 확보다. 성숙한 CASB 는 인증, 암호화, 악성코드 탐지 등 모든 클라우드 보안 영역의 현황을 실시간 대시보드로 제공하여, 조직 내에서 어떤 클라우드 서비스가 어떻게 사용되고 있는지에 대한 완전한 가시성을 제공한다. 이를 통해 관리되지 않는 'Shadow IT'를 탐지하고 통제할 수 있다.

확보된 가시성을 기반해 클라우드 데이터 보안 기능이 작동한다. CASB 의 핵심 구성 요소인 DLP(데이터 손실 방지)는 클라우드 내에서 저장되거나 이동하는 모든 데이터에 대해 기업의 보안 정책을 확장 적용한다.

최적화된 환경에서는 DLP 뿐만 아니라 암호화, 데이터 유출 관련 이상 행위 탐지 기능까지 통합 운영하고 지속적으로 모니터링하여 데이터 유출 위험을 최소화한다.

또한 CASB 는 강력한 위협 방지 기능을 제공한다. 일반적인 사용자 사용 패턴을 분석하여 비정상적인 활동을 식별하고, 적응형 접근 제어 및 악성코드 탐지 기능을 활용하여 내부 및 외부 위협으로부터 조직을 보호한다. 성숙한 CASB 시스템은 접근 제어, 악성코드 탐지, 암호화 등 다양한 보안 위협에 대해 실시간 탐지 및 모니터링 체계를 운영한다. 마지막으로, 컴플라이언스 준수는 CASB 의 중요한 역할 중 하나다. 조직은 CASB 를 통해 데이터 3 법, PCI DSS, HIPAA 와 같은 다양한 규제 표준 준수 여부를 모니터링할 수 있으며, 최적화된 단계에서는 자동화된 시스템으로 각종 규정 요구사항을 상시 만족시키며 운영한다.

12. 정책 및 프로세스

제로트러스트 환경에서 정책 및 프로세스는 다양하게 분산된 애플리케이션과 워크로드를 일관된 보안 수준으로 관리하기 위한 최상위 거버넌스 체계이다. 효과적인 통제를 위해서는 먼저 용어를 통일하고 정책을 통합하는 과정이 필수적이다. 각기 다른 팀이나 환경에서 서로 다른 기준과 절차를 따른다면 제로트러스트의 핵심인 일관성 있는 보안 정책 적용이 불가능하기 때문이다.

성숙한 정책 및 프로세스 정의는 조직의 비즈니스 구조와 서비스 방향성을 반영하여 수립되어야 하며, 컴플라이언스 준수와 업무 효율성을 모두 고려해야 한다. 초기에는 표준화된 정책 없이 필요시마다 절차를 정의하거나, 기본적인 정책이 있더라도 현행화가 이루어지지 않는 단계에 머무를 수 있다. 하지만 제로트러스트를 지향하기 위해서는 명확한 정책과 프로세스를 정의하고, 이를 준수하도록 지속적인 통제가 이루어져야 한다.

최적화된 단계에서는 이러한 정책과 프로세스가 일부 수기 작업에 의존하는 것을 넘어, 시스템으로 자동화되어 모든 애플리케이션과 워크로드에 명확하게 적용된다. 또한, 실시간 모니터링 체계를 통해 정책 준수 여부를 상시 확인하고, 변화관리가 이루어지는 선순환 구조를 갖추어야 한다. 이렇게 잘 정의되고 통합된 정책과 프로세스는 다른 모든 기술적 통제를 하나로 묶어주는 거버넌스 역할을 수행하며, 조직 전체의 보안 수준을 실질적으로 향상시키는 기반이 된다.

이처럼 어플리케이션 및 워크로드 필러는 제로트러스트 아키텍처에서 가장 방대한 영역을 다루고 있으며, 위에서 다룬 주요 요소 SMP, SASE, CASB 등은 솔루션으로 맵핑될 수 있으며 적용 범위가 조직마다 다르게 정의되거나 기능이 중복될 수 있어 명확한 접근이 필요하다. 따라서 조직은 먼저 관리해야 할 영역을 정확히 식별하고, 위에서 다룬 주요 요소들을 기반으로 일관된 정책과 프로세스를 수립하는 것이 무엇보다 중요하다.

애플리케이션 및 워크로드 필러의 고도화는 조직의 전체 소프트웨어 생명 주기에 제로트러스트 원칙을 일관되게 적용할 수 있는 관리 체계와 기술적 토대를 마련하여, 각 애플리케이션 단위에서 발생할 수 있는 위협을 사전에 방지하고 신속하게 대응할 수 있는 환경을 실현하게 한다. 또한, 이 필러의 효과적인 구현은 조직 내 핵심 비즈니스 로직과 데이터가 처리되는 지점을 직접적으로 보호하고, 변화하는 IT 환경과 갈수록 정교해지는 사이버 위협으로부터 조직의 핵심 서비스를 효과적으로 방어하는 데 필수적인 역할을 수행한다.

■ 주요 시스템별 제로트러스트 기능 구현

제로트러스트 환경을 성공적으로 구현하기 위해서는 기술적 방안과 이를 수행할 수 있는 시스템이 필수적이다. 제로트러스트 아키텍처는 "신뢰하지 않고 항상 검증한다"는 원칙을 기반으로 하며, 이를 실현하기 위해 각 시스템 별 상태를 확인하고, 지속적으로 검증하며, 최소 권한 접근 보장 등을 수행하는 시스템을 갖춰야 한다.

아래 주요 시스템 등은 각각 제로트러스트 환경에서 중요한 역할을 담당하며, 이들 시스템은 상호 연계되어 조직의 보안 태세를 강화할 수 있다. 각 시스템 별로 제로트러스트 환경 구현을 위해 수행해야 할 기능과 이를 통해 조직이 얻을 수 있는 보안 강화 효과를 구체적으로 살펴보고자 한다.

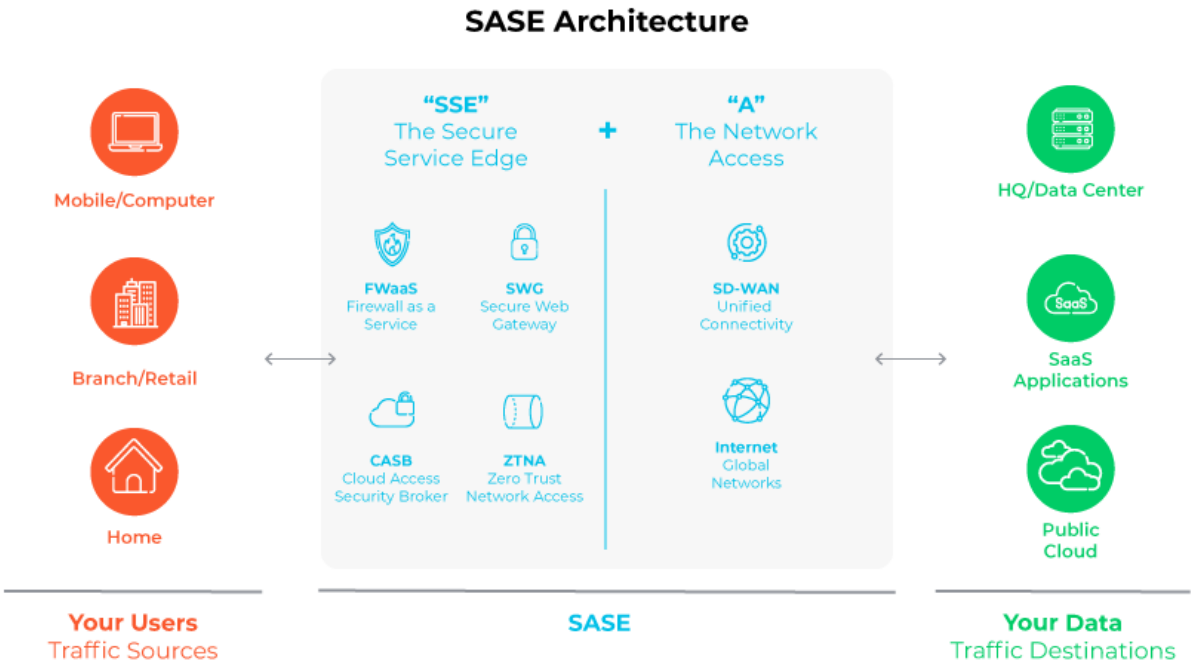
출처 : SK 쉴더스, "제로트러스트의 시작:SKZT 로 완성하다"

그림 2. 어플리케이션 필러 주요 시스템

1. SASE (Secure Access Service Edge, 보안 액세스 서비스 엣지)

SASE 는 기존의 온프레미스 중심 네트워크 및 보안 아키텍처의 한계를 극복하기 위해 등장한 클라우드 네이티브 프레임워크이다. 이는 사용자와 기기가 어디에 있든, 애플리케이션과 데이터가 어디에 있든 관계없이 일관된 보안 정책과 최적화된 네트워크 성능을 제공하는 것을 목표로 한다. SASE 는 단순히 여러 보안 시스템을 합친 것이 아니라, 네트워크 기능과 보안 기능이 클라우드 상에서 본질적으로 융합된 구조를 가진다.

SASE 의 구조를 정확히 이해하기 위해서는 SSE(Secure Service Edge)와의 관계를 파악하는 것이 중요하다. SSE 는 클라우드를 통해 제공되는 보안 기능의 집합으로, 주로 ZTNA(제로 트러스트 네트워크 액세스), CASB(클라우드 접근 보안 브로커), SWG(보안 웹 게이트웨이), FWaaS(서비스형 방화벽)등을 핵심 구성 요소로 한다. 즉, SSE 는 '보안'에 중점을 둔다. SASE 는 이러한 SSE 의 모든 보안 기능에 A(Access)의 기능인 지능형 네트워크 기능 SD-WAN 을 결합하여 완성되는 더 넓은 개념의 프레임워크이다.



출처 : Paloalto, "SASE vs. SSE: What Is the Difference?"

그림 3. SASE Architecture

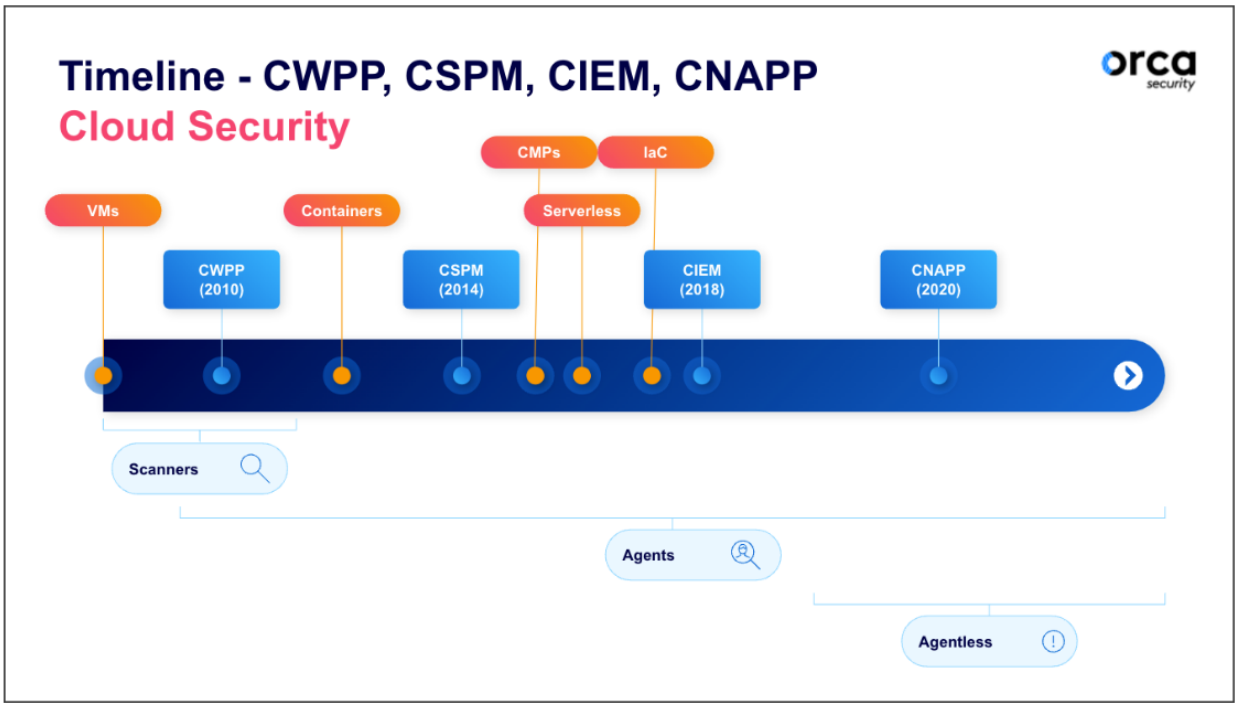
제로트러스트 관점에서 SASE 는 분산된 환경의 모든 '접근'을 통제하는 핵심적인 역할을 수행한다. 하지만 최근 기술력이 뛰어난 글로벌 기업들은 SASE 의 범위를 더욱 확장하고 있다. 글로벌 벤더의 경우 통합 플랫폼은 전통적인 SASE 의 기능을 넘어 클라우드 환경 내부를 보호하는 CNAPP(클라우드 네이티브 애플리케이션 보호 플랫폼)의 영역까지 포함하기도 한다. 즉, 하나의 플랫폼 안에서 CWPP(클라우드 워크로드 보호) 나 CSPM(클라우드 보안 형상 관리) 기능 등을 함께 제공하는 것이다. 다만, 이는 벤더의 기술력과 플랫폼 설계 역량에 따라 상이하며, 대부분 별도의 라이선스로 분리되어 제공되므로 조직의 필요에 따라 기능 범위를 신중하게 검토해야 한다.

예를 들어 팔로알토 네트웍스의 Prisma SASE 같은 플랫폼은 기본적으로 SASE 로 분류되지만, 라이선스 정책에 따라 CWPP, CSPM 등 CNAPP 의 주요 기능들을 추가하여 활용할 수 있다. 더 나아가 최근 팔로알토 네트웍스가 아이덴티티 기업인 사이버아크(CyberArk)를 인수하기로 한 것은 SASE 플랫폼이 나아가야 할 방향을 명확히 보여준다. 과거 대부분의 보안 침해가 도용된 자격 증명에서 시작되었다. 때문에 제로트러스트 아키텍처 측면에서 SASE 플랫폼에 강력한 IAM 및 특권 접근 관리(PAM) 기능을 통합하여 '접근'뿐만 아니라 '신원'까지 완벽하게 통제하는 통합 보안 플랫폼으로 진화하려는 것이다. 벤더에 따라 SASE 로 구현할 수 있는 범위는 확장될 수 있지만, 이러한 단일 플랫폼 접근 방식은 특정 벤더에 대한 종속성 심화, 각 기능별 최고 수준의 성능대비 일부 기능의 제약, 복잡한 라이선스 체계 등의 문제점을 야기할 수 있으므로 도입 시 장단점을 함께 고려해야 한다.

결론적으로 SASE 는 벤더의 기술력, 글로벌 PoP(Point of Presence) 커버리지, 실제 통합 수준에 따라 제공 기능과 품질의 차이가 크다. 따라서 도입 시에는 국내 비즈니스 환경의 특수성, 특히 망분리 정책과의 호환성을 반드시 고려해야 한다. 클라우드 기반으로 동작하는 SASE 의 특성상 망분리 환경에서는 기능이 제약될 수 있으므로, 온프레미스와의 연동 방안 등을 포함한 면밀한 아키텍처 설계가 선행되어야 한다.

2. CNAPP (Cloud-Native Application Protection Platform, 클라우드 네이티브 애플리케이션 보호 플랫폼)

CNAPP 은 복잡한 최신 클라우드 환경의 보안을 위해 등장한 통합 보안 플랫폼이다. 클라우드 네이티브 애플리케이션은 컨테이너, 인스턴스 등 동적인 기술로 구축되어 멀티·하이브리드 클라우드에 배포된다. 이러한 환경에서는 기존의 파편화된 개별 보안 시스템으로는 전체적인 가시성을 확보하기 어렵고, 보안팀은 수많은 경고 속에서 실제 위협을 식별하는 데 어려움을 겪는다. CNAPP 는 이러한 문제를 해결하기 위해 개발부터 운영까지 클라우드 애플리케이션의 전체 생명주기에 걸친 보안 기능을 단일 플랫폼에서 통합하여 제공한다.



출처 : orca security, "Know Your Cloud Security Acronyms"

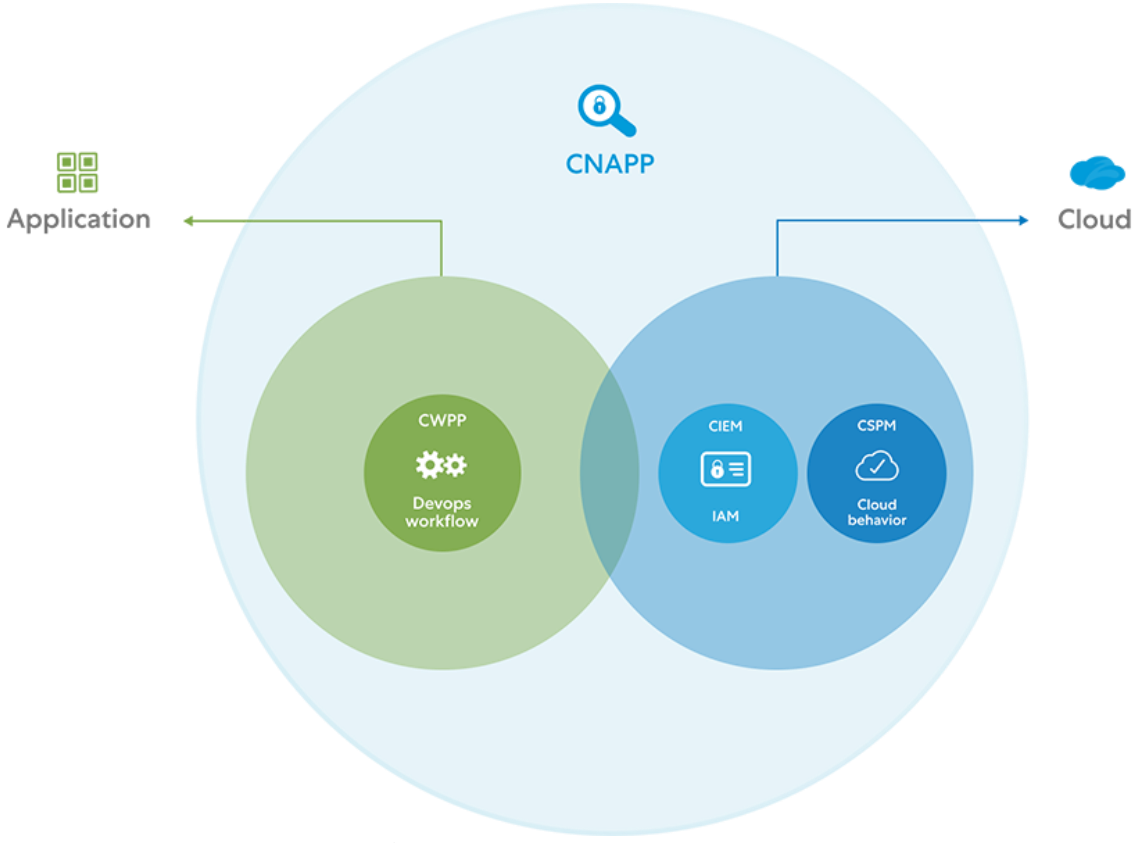
그림 4. A History of Cloud Security Technologies(by Gatner)

CNAPP 는 클라우드 보안의 진화 과정을 보여주는 개념으로, 여러 개별 보안 시스템들이 하나의 플랫폼으로 통합된 형태이다. 이 통합의 핵심 구성 요소는 다음과 같다.

(1) CSPM (Cloud Security Posture Management, 클라우드 보안 형상 관리)
클라우드 인프라의 보안 설정을 자동으로 지속 모니터링하는 기능으로 설정 오류, 정책 위반, 오픈 스토리지 버킷과 같은 규정 준수 이슈를 식별하여 클라우드 환경 자체의 안전한 보안 '태세'를 확립하고 유지하는 역할을 수행한다.

(2) CWPP (Cloud Workload Protection Platform, 클라우드 워크로드 보호 플랫폼)
가상머신, 컨테이너 등 클라우드에서 실행되는 워크로드 자체를 보호하는 데 중점을 둔다.
실시간 위협 탐지, 취약점 스캐닝, 시스템 무결성 보증 등의 기능을 통해워크로드가 동작하는 동안 발생하는 다양한 위협을 능동적으로 방어하는 역할을 수행한다.

(3) CIEM (Cloud Infrastructure Entitlement Management, 클라우드 인프라 권한 관리)
복잡한 클라우드 환경 내 사용자 및 서비스 계정의 권한을 관리하고 적용한다. 과도하게 부여되었거나 사용하지 않는 권한을 식별하고 회수함으로써 최소 권한 원칙을 실현하고 계정 탈취로 인한 위험을 근본적으로 줄이는 역할을 수행한다.



출처 : Gartner, "How to protect your clouds with CSPM, CWPP, CNAPP, and CASB"

그림 5. Cloud Technologies Simply

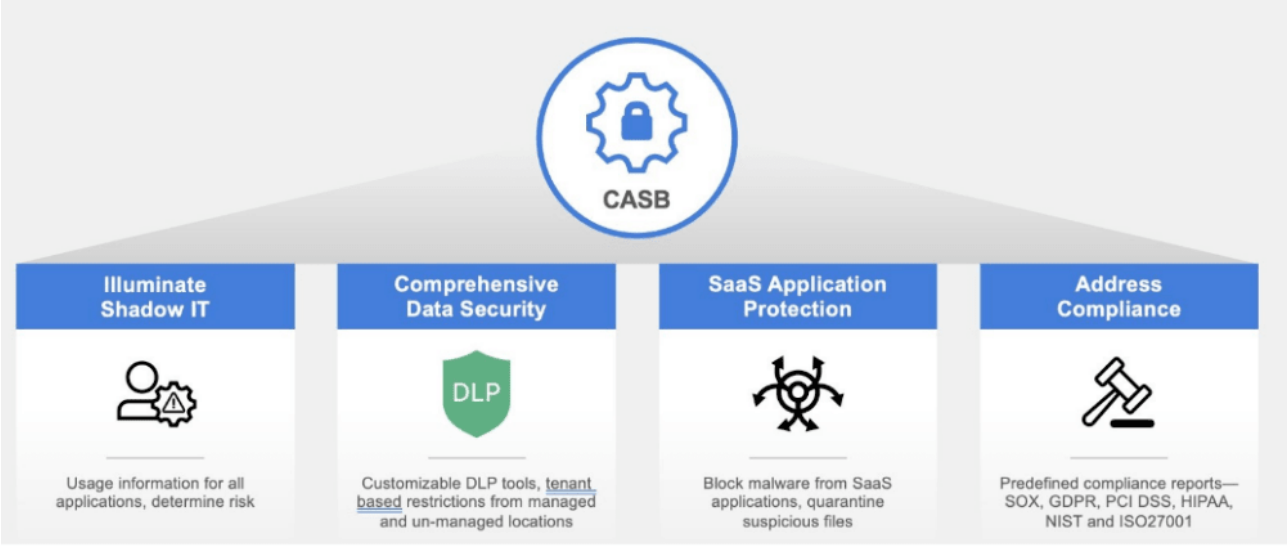
이처럼 성숙한 CNAPP 는 CSPM, CWPP, CIEM 과 같은 핵심 기능을 통합하고, 나아가 코드형 인프라(IaC) 스캐닝, 쿠버네티스 보안 형상 관리(KSPM)까지 확장하여 클라우드 기반 어플리케이션 개발 초기부터 운영까지 전반적인 보안을 책임진다.

CNAPP 의 동작 방식은 취약점, 설정 오류, 사용자 행동 등 클라우드 환경의 모든 데이터를 수집하고(침입), 비정상적이거나 알려진 위협 패턴을 분석하여(이해), 최종적으로 보안팀에 위험 우선순위가 높은 경고와 컨텍스트를 시각화하여 제공함으로써 신속한 해결을 돕는다. 이를 통해 조직은 파편화된 시스템 운영으로 인한 비효율성과 비용을 줄이고, 통합된 가시성을 바탕으로 클라우드 보안을 크게 강화할 수 있다. 제로트러스트 아키텍처에서 SASE 가 '클라우드로의 안전한 접근'을 책임진다면, CNAPP 는 '클라우드 내부의 애플리케이션과 인프라'를 안전하게 보호하는 핵심적인 역할을 수행한다.

3. CASB (Cloud Access Security Broker, 클라우드 액세스 보안 브로커)

CASB 는 클라우드 서비스 사용자와 클라우드 서비스 제공자(CSP) 사이에 위치하여, 조직의 보안 정책을 클라우드 환경까지 확장하고 일관되게 적용하는 보안 정책 시행 지점이다. 제로트러스트 아키텍처에서 SASE 가 '클라우드로의 안전한 접근 경로'를 보호하고, CNAPP 가 '클라우드 인프라와 워크로드 자체'를 보호한다면, CASB 는 '사용자와 SaaS 애플리케이션 간의 상호작용'을 안전하게 만드는 데 특화된 역할을 한다. CASB 는 SASE 프레임워크 내에 통합된 기능으로 제공되거나, 조직의 보안 요구에 따라 독립된 시스템으로 구축 및 운영될 수 있다.

CASB 의 주요 기능은 크게 네 가지로 분류된다. 첫째, 가시성 확보를 통해 조직 내에서 사용되는 모든 승인 및 비승인 클라우드 어플리케이션을 탐지하고 위험도를 평가한다. 둘째, 데이터 보안을 위해 클라우드에 저장되거나 이동하는 민감 정보에 대해 실시간 DLP(데이터 손실 방지) 정책을 집행하여 무단 공유를 차단한다. 셋째, 위협 방지를 위해 사용자 행동을 분석하여 비정상 행위를 탐지하고 랜섬웨어나 내부자 위협 등으로부터 클라우드 환경을 보호한다. 마지막으로, 규정 준수를 위해 HIPAA, PCI DSS, GDPR 등 국내외 주요 규제에 대한 데이터 정책 및 보안 준수 현황을 지속적으로 모니터링한다.



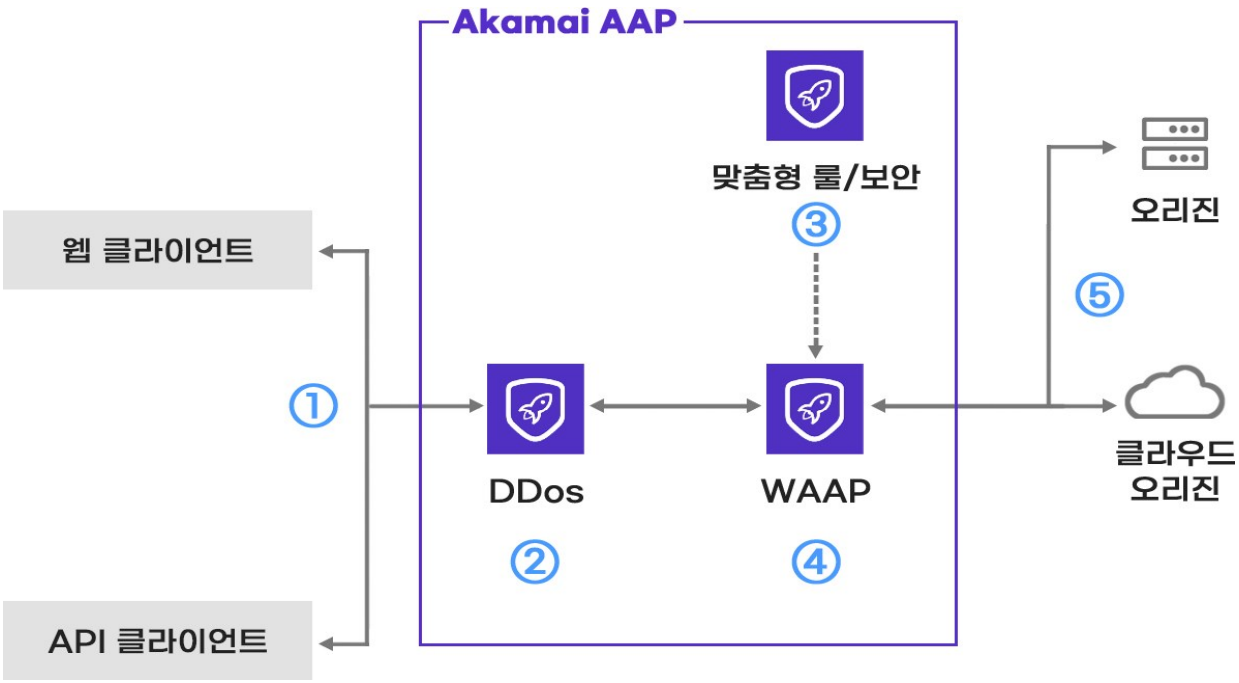
출처 : Fortinet, "What Is CASB (Cloud Access Security Broker)?"

그림 6. CASB Key Features

이러한 기능들은 주로 세 단계의 프로세스를 거쳐 체계적으로 작동한다. 먼저 탐지 및 식별(Discovery) 단계에서 조직 내에서 사용 중인 모든 클라우드 서비스를 식별하여 가시성을 확보한 다음 분류(Classification) 단계에서 각 서비스와 데이터의 위험 요소와 민감도를 평가한다. 마지막 수정(Remediation) 단계에서 평가 결과를 바탕으로 DLP, 접근 차단, 암호화 등 조직의 정책에 맞는 보안 통제를 자동으로 적용하여 지속적으로 클라우드 환경을 보호한다. 이처럼 CASB 는 다른 클라우드 보안 시스템들과 상호 보완적으로 연동하며, 제로트러스트 원칙을 SaaS 애플리케이션 영역까지 확장하는 필수적인 역할을 담당한다.

4. WAAP (Web Application and API Protection, 웹 애플리케이션 및 API 보호)

최근 IT 인프라가 클라우드·API 중심으로 변화함에 따라, 기존의 경계형 WAF(웹 방화벽)나 DDOS 방어 시스템만으로는 고도화된 최신 위협에 효과적으로 대응하기 어려운 한계가 명확해졌다. WAAP 는 이러한 한계를 극복하기 위해 등장한 클라우드 네이티브 보안 모델로, 분산된 환경 전반에 걸쳐 웹 애플리케이션과 API 를 포괄적으로 보호하는 데 중점을 둔다. 제로트러스트 환경에서 WAAP 는 온프레미스에 종속되지 않고 네트워크(L3/L4) 및 애플리케이션(L7) 계층의 공격을 실시간으로 탐지하고 차단하는 통합된 구조를 제공한다.



출처 : 굿모닝아이텍(주), "Akamai AAP 소개자료"

그림 7. Akmai WAAP 구성도

WAAP 의 핵심적인 역할 중 하나는 정교한 API 보안이다. WAAP 는 단순히 API 요청을 허용하거나 차단하는 것을 넘어, 트래픽의 구조, 메타데이터, 행위 특성까지 정밀하게 분석하여 비정상적인 호출이나 데이터 유출 시도 등 고도화된 위협을 체계적으로 방어한다. 또한, API 게이트웨이의 역할을 일부 수행하며 모든 API 요청에 대한 인증과 권한을 관리하고, 미 승인된 숨겨진 API 를 탐지하는 등 API 의 전체 생명주기에 걸쳐 일관된 보안 정책을 적용한다.

더 나아가 WAAP 는 정교한 봇(Bot) 트래픽 탐지 및 차단, IP 평판이나 지리적 위치 등 최신 위협 인텔리전스를 활용한 실시간 위협 차단, 그리고 개별 서비스에 맞는 맞춤형 규칙 및 가상 패치적용 등 다층적인 웹 공격 방어 기능을 포함한다. 결론적으로 WAAP 는 클라우드 기반 인프라 위에서 웹 및 API 보안 정책의 중앙 집중적 집행, 지속적인 검증, 자동화된 대응을 통해 경계가 없는 제로트러스트 아키텍처 기반으로 다양한 API 를 사용하는 환경에서 실질적인 어플리케이션 보호 체계를 실현하는 시스템으로 동작한다.

5. SCA (Software Composition Analysis, 소프트웨어 구성 분석)

SCA 는 조직 내에서 사용되는 다양한 오픈소스 소프트웨어에 대해 취약점, 라이선스, 지원 종료(EOL/EOS), 버전 등을 통합적으로 식별, 분석, 모니터링하여 소프트웨어 공급망의 보안을 강화하는 시스템이다. 기존의 취약점 관리 시스템이 운영체제나 서버 등 인프라 전반의 취약점을 다룬다면, SCA 는 애플리케이션을 구성하는 오픈소스 구성 요소의 보안성과 신뢰성에 집중한다는 점에서 차별화된다.

SCA 의 핵심 기능은 개발 프로젝트에서 사용 중인 모든 오픈소스의 목록, 즉 SBOM(Software Bill of Materials)을 자동으로 수집하고 생성하는 것에서 시작한다. 이를 바탕으로 알려진 취약점(CVE), 라이선스 준수 여부, 지원 중단 상태 등을 실시간으로 분석하고, 신규 위협이 탐지되면 즉시 경고를 보낸다. 제로트러스트 환경에서는 이러한 분석 결과를 바탕으로, 정책에 위배되는 오픈소스의 사용을 CI/CD 파이프라인 단계에서 자동으로 차단하거나 안전한 버전으로 업데이트하는 등 지속적이고 자동화된 통제를 수행하여 신뢰할 수 없는 외부 요소가 운영 환경에 유입되는 것을 원천적으로 방지한다.

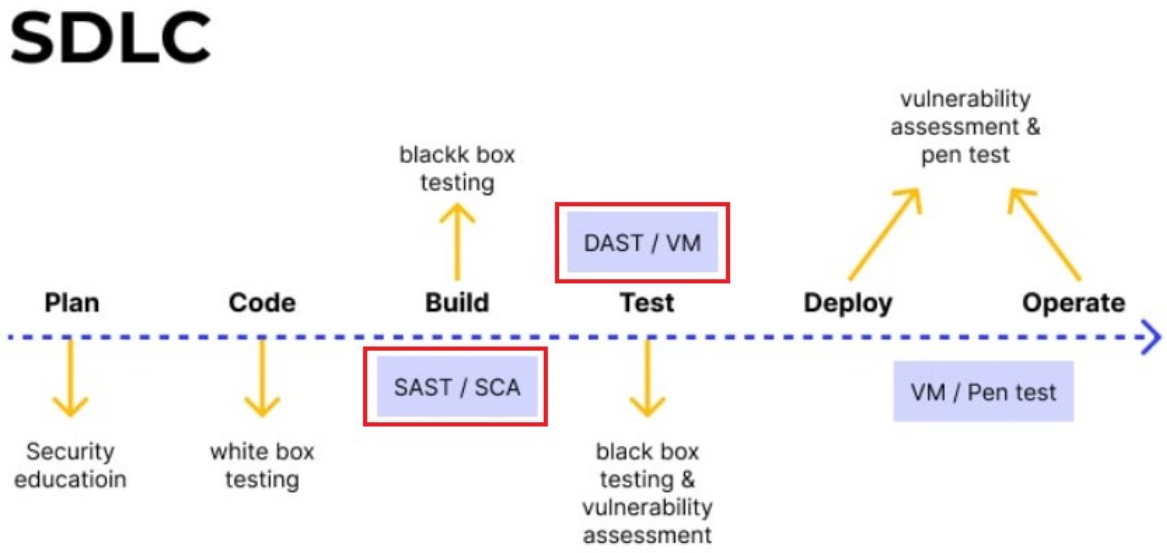
최근 보안 시장에서 SCA 는 독립된 시스템으로 존재하기보다 다른 보안 체계와 통합되는 추세가 뚜렷하다. 첫째, SAST, DAST 와 결합하여 개발자가 작성한 코드와 가져온 오픈소스 코드를 모두 분석하는 통합 애플리케이션 보안 테스트(AST) 플랫폼 형태로 제공된다. 둘째, 클라우드 네이티브 환경에서는 SASE 나 CNAPP 과 같은 통합 보안 플랫폼의 핵심 모듈로 포함되는 경우가 많다. 이는 컨테이너와 마이크로서비스 환경에서 오픈소스 활용이 폭발적으로 증가함에 따라, 클라우드 애플리케이션의 안전성을 확보하기 위해 SCA 가 필수 요소로 자리 잡았기 때문이다.

궁극적으로 제로트러스트 관점에서 SCA 는 단순히 취약점을 찾는 도구를 넘어, 신뢰할 수 없는 외부 요소로 취급되어야 하는 소프트웨어 공급망 전체의 무결성을 지속적으로 검증하는 핵심 시스템으로 기능한다.

6. SAST/DAST (Static/Dynamic Application Security Testing, 정적/동적 에플리케이션 보안 테스트)
제로트러스트 환경의 애플리케이션 보안은 소프트웨어 개발 생명주기(SDLC) 전반에 걸쳐 보안을 내재화하는 'Shift Left' 개념에서 출발한다. 이는 보안 위협을 개발 초기 단계에서부터 식별하고 제거함으로써, 운영 환경에 배포되는 애플리케이션 자체의 신뢰도를 근본적으로 확보하는 것을 목표로 한다. SAST 와 DAST 는 이러한 보안 SDLC 를 구현하는 핵심적인 자동화 테스트 방식이다.

SAST 는 소프트웨어 공급망 공격과 취약한 개발 코드로 인한 위협이 증가함에 따라, 애플리케이션을 운영 환경에 배포하기 전에 코드 레벨에서 보안 취약점을 사전에 탐지하고 조치하는 역할을 수행한다. SAST 는 애플리케이션이 구동되기 이전인 개발 및 코딩 단계에서 소스 코드, 바이너리 등을 직접 분석하여 취약한 함수 사용, 미흡한 입력값 검증, 인증·권한 처리 오류 등 잠재적인 보안 결함을 식별하는 '화이트박스' 방식이다. 제로트러스트 아키텍처에서 SAST 는 코드 작성부터 배포 전까지 지속적인 보안 검증을 자동화함으로써, 신뢰할 수 없는 코드가 운영 환경에 유입되는 것을 원천적으로 차단하는 핵심적인 역할을 한다.

DAST 는 SAST 와 상호 보완적으로 작동하는 '블랙박스' 방식의 테스트다. 코드 내부를 분석하는 것이 SAST 라면, DAST 는 실제 구동 중인 애플리케이션을 외부 공격자의 관점에서 테스트한다. DAST 는 소스 코드를 보지 않고 웹 애플리케이션이나 API 에 실제 공격과 유사한 요청을 보내고 그 응답을 분석하여, SQL 인젝션, 세션 하이재킹, 서버 설정 오류 등 운영 환경에서만 드러나는 취약점을 탐지한다. 이를 통해 SAST 만으로는 발견하기 어려운, 실제 서비스 환경과 외부 연동 요소까지 고려한 실전적인 보안 결함을 검증할 수 있다.



출처 : Wallarm, “Dynamic Application Security Testing: Advantages and Disadvantages”

그림 8. SDLC 내 SCA,SAST,DAST

어플리케이션 및 워크로드 필러는 제로트러스트 아키텍처에서 조직의 비즈니스 로직과 데이터가 실제로 실행되고 처리되는 동적인 영역을 다룬다. SASE, CNAPP, CASB 와 같은 클라우드 네이티브 보안 프레임워크를 중심으로 사용자의 안전한 접근을 보장하고, 클라우드 내부의 인프라와 워크로드, SaaS 애플리케이션 사용 전반을 보호한다. 나아가 WAAP 를 통해 웹과 API 의 경계를 방어하고, SCA, SAST/DAST 를 SDLC 에 통합하여 소프트웨어 공급망부터 코드 레벨까지 신뢰성을 확보하는 심층 방어 체계를 구축한다.

어플리케이션 및 워크로드 필러의 주요 시스템들은 식별자 필러의 IAM, 네트워크 필러의 ZTNA 등 다른 필러의 핵심 시스템들과 유기적으로 연동된다. 이러한 상호 연동을 통해 온프레미스와 클라우드를 아우르는 복잡한 환경에서도 애플리케이션 접근, 데이터 흐름, 개발 및 배포 전 과정에 걸쳐 '지속적인 검증'과 '최소 권한' 원칙을 일관되게 적용하고 강화할 수 있다.

■ 맺음말

제로트러스트 아키텍처에서 애플리케이션 및 워크로드 필러는 앞서 다룬 주요 요소와 시스템들을 기준으로 다양한 환경에 일괄적이고 체계적인 보안 아키텍처를 수립하고 관리할 수 있다. 앞선 내용과 같이, 소프트웨어 개발 생명주기(SDLC) 내의 개발 코드에 대한 보안체계(SCA, SAST, DAST 등)부터, 사용자의 안전한 접근을 보장하는 클라우드 네이티브 보안 프레임워크(SASE, WAAP, CASB 등), 그리고 클라우드 내부의 인프라와 워크로드를 직접 보호하는 CNAPP 에 이르기까지, 애플리케이션 및 워크로드 필러는 다층적이고 심층적인 방어 체계를 요구한다.

제로트러스트 아키텍처 내에서 애플리케이션 및 워크로드 필러는 특성상 보안 뿐만 아니라 인프라와 개발 영역까지 깊숙이 관여한다는 점에서 거버넌스의 중요성이 더욱 두드러진다. 여기서 거버넌스란, 단순히 규칙을 만드는 것을 넘어 개발, 인프라, 보안팀이 공동의 목표 아래 협업할 수 있도록 명확한 정책과 프로세스, 그리고 역할과 책임(R&R)을 정의하는 통합적인 관리 체계를 의미한다.

성공적인 제로트러스트 전환은 기술의 도입만으로는 불가능하다. 반드시 조직 문화와 협업 체계의 변화가 동반되어야 하기 때문이다. 이런 부분이 반영되지 않는다면 실무 환경에서는 각 부서 간의 이해관계가 복잡하게 얽힐 수 있다. 예를 들어, 개발 부서는 신속한 배포를, 인프라 부서는 안정성을, 그리고 보안 부서는 통제를 최우선으로 여기면서 충돌이 발생할 수 있다.

결론적으로, 온프레미스, 하이브리드, 멀티 클라우드 등 다양한 환경으로 워크로드가 계속해서 확장되는 상황에서, 제로트러스트 아키텍처는 조직의 나침반 역할을 할 핵심 기반이 될 수 있다. 이를 통해 보호해야 할 애플리케이션 및 워크로드의 적용 범위를 명확히 정의하고, 이에 필요한 기술들을 체계적으로 식별하여 조직의 실제 환경에 최적화된 형태로 구현하는 것이 무엇보다 중요하다. DevSecOps 와 같은 긴밀한 협업 체계를 바탕으로 제로트러스트 원칙을 적용해 나갈 때, 비로소 조직은 끊임없이 변화하는 디지털 환경 속에서 핵심 자산인 애플리케이션과 데이터를 안전하게 관리하고 지켜낼 수 있을 것이다.

■ 참고 문헌

[1] KISA, “제로트러스트가이드라인 V2.0”, 2024.12

[2] NIST SP 800-207, “Zero Trust Architecture”, 2020.08

[3] NIST SP 1800-35 Final,“Implementing a Zero Trust Architecture:High-Level Document”, 2025.06

[4] NIST SP 800-218, “Secure Software Development Framework (SSDF)”, 2022.02

[5] NIST SP 800-204, “Security Strategies for Microservices-based Application Systems”, 2019.08

[6] NIST SP 800-190, “Application Container Security Guide”, 2017.09

[7] CISA, “CISA Zero Trust Maturity Model V2”, 2023.11

[8] DoD, “Zero Trust Overlays”, 2024.06

[9] 국가사이버안보센터, “국가 망 보안체계 보안 가이드라인(Draft)”, 2025.01

■ 참고 자료

[1] SK쉴더스, “제로트러스트의 시작:SKZT로 완성하다” – 브로슈어

[2] SK쉴더스, “SW 공급망 보안 위협과 대응 방안”

[2] Gartner, “Best Security Service Edge Reviews 2025”

[3] Gartner, “What is Cloud-Native Application Protection Platforms?”

[4] Akamai, “What Is Web Application and API Protection (WAAP)?”

[5] ATLASSIAN, “Microservices vs. monolithic architecture”

[6] ATLASSIAN, “What is SDLC? Software Development Life Cycle Explained”

[7] Paloalto, “Secure Access Service Edge (SASE) Technical Guides”