보안을 강화한 플랫폼 엔지니어링 적용 방안

김유란 [1], 유헌창 [2]
[1] 고려대학교 소프트웨어보안학과 석사과정
[2] 고려대학교 컴퓨터학과 교수

yuran3391@korea.ac.kr, yuhc@korea.ac.kr

Applying Platform Engineering with Enhanced Security

요 약

클라우드 네이티브 환경에서 데브옵스를 채택하고 내재화하는 과정에서 개발자의 인지부하가 발생하였다. 이를 해결하기 위해, 개발자가 개발에만 집중할 수 있도록 일관된 요구사항에 맞는 개발 환경을 제공하는 플랫폼 엔지니어링이 등장하였다. 하지만, 플랫폼 엔지니어링에서 전체 워크플로우 보안을 고려한 연구가 부족한 상황이다. 이를 보완하기 위해 데브섹옵스 관점을 적용하여 전체 워크플로우 보안 방안을 CI/CD 파이프라인 단계, 운영 단계로 나누어서 제안하였다. 또한, 신규 서비스를 런칭 한다고 가정 후 보안 적용 프로세스에 대해서 제안한다. 이렇게 전체 워크플로우의 보안을 고려함으로써, 모든 서비스에서 동일 수준의 보안을 유지할 수 있는 장점이 있다.

1. 서론

4 차 산업혁명 시대를 맞아 디지털 전환이 가속화됨에 따라, 적합한 사용자 서비스 제공을 위해 클라우드 전환이 추진되고 있다. 클라우드 전환 및 도입 효과를 높이기 위해 단순한 인프라 기술 위주의 클라우드 도입보다 클라우드의 장점을 최대한 활용할 수 있는 '클라우드 네이티브' 도입이 필요하다. 즉, 기존의 크고, 단일한 서비스 구조를 마이크로 서비스 아키텍처로 구현하여 개발, 배포, 운영함으로써 빠르고 안정적인 서비스를 제공해야 한다. 클라우드 네이티브의 핵심요소 중 하나인 데브옵스는 내가 구축하고 내가 실행한다(You build it, you run it)는 슬로건 하에 애플리케이션과 서비스를 빠른 속도로 제공할 수 있도록 조직의 역량을 향상시키는 문화, 철학, 방식 및 도구의 조합으로 해석할 수 있다. 많은 조직은 데브옵스를 받아들이고 내재화하는 과정에서 구체화 및 실체화 그리고 표준화가 필요하다고 판단했으며, 플랫폼 엔지니어링이 등장하게 되었다. Gartner 는 플랫폼 엔지니어링을 2023 년~2024 년 10 대 기술 트렌드로 선정하였으며, 2026 년까지 소프트웨어 엔지니어링 조직의 약 80%가 애플리케이션 제공을 위한 재사용 가능한 서비스, 구성 요소 및 도구의 내부 공급자로 플랫폼 팀을 구축할 것으로 예측하였다. [1]

하지만, 지금까지 플랫폼 엔지니어링에 연구에서는 워크플로우 전체를 아우르는 보안 적용 방안에 대한 고민이 충분하지 않았다. CNCF 의 연구[2], Humanitec 의 연구[3]에서는 런타임 보안, Secret, Identity 에 한정해 플랫폼 엔지니어링에서 보안을 바라보았다. 본 논문에서는 전체 워크플로우를 고려한 플랫폼 엔지니어링에서 보안 적용 방안을 제안한다. 이를 기반으로 개발 시 효율적인 보안 적용 뿐만 아니라 모든 서비스에 동일 수준의 보안을 유지할 수 있게 된다.

2. 관련 연구

2.1. 데브옵스(DevOps)와 플랫폼 엔지니어링

데브옵스는 소프트웨어 제품과 서비스를 보다 빠르고 고품질로 생산하기 위해 개발(Dev)팀과 운영(Ops)팀의 협업과 통합에 중점을 두는 조직적 접근 방식, 문화로 정의할 수 있다.[4] 데브옵스 문화에서 개발자가 직접 사용할 소프트웨어를 찾아 학습, 배포, 관리하는 것이 권장되었고 이로 인해 자신의 서비스에 대해 많은 권한을 갖을 수 있다는 장점도 있었다. 하지만, 빠르게 변하는 시장 요구 사항을 충족하기 위해 다양한 신기술들이 등장하였고, 이에 따라 개발자의 인지 부하[5]가 증가하게 되었다. 이를 해결하고자 개발자가 개발에만 집중할 수 있게 도와주는 '내부 개발자 플랫폼(IDP)' 기반 플랫폼 엔지니어링이 등장하였다. IDP 에서 표준화된 환경, 도구, 재사용 가능한 구

성 요소, 중앙 집중식 라이브러리를 제공함으로써 개발자가 매번 처음부터 기능을 개발할 필요가 없게 되었다. 또한, 플랫폼의 일부를 공통 요소로 활용하여 개발 시간을 단축하며 고품질 서비스를 제공할 수 있게 되었다.

2.2. 데브섹옵스와 플랫폼 엔지니어링 보안

클라우드 성숙도가 높아짐에 따라 데브옵스에 보안이 더해진 데브섹옵스가 등장했다. 데브섹옵스에서는 효과적으로 소프트웨어를 제공하기 위해 개발 수명 주기의 최전선에서부터 애플리케이션 보안을 고려하여 근본적으로 안전한 소프트웨어를 제공해 큰 비용이 드는 보안 사고를 방지하고자 한다. [6]

플랫폼 엔지니어링은 이러한 소프트웨어 제공 및 수명 주기 관리를 위한 셀프 서비스 IDP 구축 및 운영에 중점을 두고 있으므로 데브섹옵스에서 구현하고자 했던 궁극적 목표를 포함해야 하며, 이를 기반으로 개발된 서비스는 동일한 수준의 보안을 유지할 수 있다. 하지만, 지금까지 플랫폼 엔지니어링에 연구에서는 전체 워크플로우에 대한 보안 고려가 부족했다. CNCF 에서는 Platform capabilities 중 플랫폼 보안을 런타임 동작 모니터링 및 빌드와 아티팩트 취약점에 한정하여 제시하였다. Humanitec 에서는 platform security plane 을 Identity, 런타임 보안을 중심으로 제시한다.

3. 플랫폼 엔지니어링에서 고려해야 하는 보안 요소

본 절에서는 데브섹옵스를 포괄하는 플랫폼 요소의 End-to-End 보안 적용 방안에 대해 CI/CD 파이프라인 보안, 배포 이후 보안, 관찰 가능성 확보, 접근 관리 체계 수립, 시크릿 관리 항목으로 나누어서 설명한다.

3.1. CI/CD 파이프라인 보안

CI/CD 파이프라인 보안 측면에서는 빌드 및 배포 단계부터 소스 코드 스캐닝, 테스트, 이미지 스캐닝 등의 단계를 거쳐 안전한 골든 이미지를 만들어야 한다. 모든 이미지가 동일한 보안 기준을 충족하여 배포 일관성을 유지하게 되면 운영상의 오류와 보안 공격을 최소화할 수 있기 때문이다.

3.2. 배포 이후의 보안

애플리케이션이 배포 된 이후 운영 단계에 대한 보안이 중요하다. 런타임 보안은 컨테이너 이미지가 런타임 환경에서 실행되기 전에 지정된 보안 정책을 준수하는지 확인하고, 컨테이너 런타임 환경에 허가 받지 않은 불법 파일 접근 시도와 원격코드 실행을 탐지하고 방지하는 기능을 제공한다.

3.3. 관찰 가능성(Observability) 확보

시스템 및 애플리케이션에 대한 가시성 확보를 위해서는 로깅 및 모니터링 시스템을 구축해야 한다. 로깅의 대표적인 분석 방식으로는 SIEM 이 있다. SIEM 을 통해 Audit 로그 등 다양한 보안 이벤트를 하나의 화면에서 살펴보고 관리할 수 있다.

모니터링은 인프라 그리고 애플리케이션으로 나눠볼 수 있다. 인프라 모니터링은 VM, Container 등 시스템 워크로드에 대한 CPU, Memory, Disk, Network 사용량을 수집 및 분석하여 트래픽 증가를 사전에 인지하고 대응할 수 있도록 체계를 구성하고 선제적으로 대응할 수 있어야 한다.

애플리케이션 모니터링은 애플리케이션 특성에 따라 수집하고 확인해야 하는 모니터링 정보가 상이하기에 사용하는 언어, 프레임워크에 최적화된 정보를 수집하기 위해 개발팀과 충분한 협의가 필요하며 본 논문에서는 안전하고 신뢰성 있는 운영을 위해 별도의 SRE 조직을 개설하고 운영하는 것을 권장한다.

3.4. 접근 관리체계 수립

인증(AuthN)을 위해서는 식별자(Identifier)를 통한 식별(Identify)이 선행되어야 하며, 식별된(Identified) 사용자에게 권한을 부여하는 것을 인가(AuthZ)라고 한다. 이러한 프로세스를 위해 IAM 관리가 필수적이다. 조직에서는 이러한 IAM 을 통해 사람 및 시스템에 계정 관리와 더불어 권한 있는 액세스 관리(PAM)을 통해 클라이언트(사람/시스템)에 대한 접근을 보고하고 관리하는 기능을 제공한다. PAM 솔루션은 기본적으로 접근제어 뿐만 아니라 세션관리 및 모니터링을 제공할 수 있어야 하며, IAM 과의 연계를 통한 통합보안 환경을 제공하는 것이 모범사례로 간주될 수 있다.

3.5. 시크릿 관리

클라우드의 도입과 함께 전통적 보안 모델인 ‘Castle and Moat’의 패러다임에서 ‘Zero Trust’로 변화되어야 한다. 클라우드 기반의 워크로드는 더 이상 네트워크 방화벽 내부와 같은 특정 환경에서만 동작하지 않으며 멀티/하이브리드 클라우드, 컨테이너, 서버리스, 데이터베이스 등 시스템이 분산되어 있어 관리가 어려워지고 있다. 이것은 보안 관점에서 ‘공격 표면이 넓어진다’는 것을 의미할 수 있다. 이렇게 관리해야 할 시크릿이 무분별하게 퍼져 나가는 ‘시크릿 스프롤 현상’을 방지하기 위해 자격증명 기반의 시크릿 중앙관리 방안을 적용해야 한다. 대표적인 시크릿에는 PKI 인증서, 데이터베이스 및 클라우드 크레덴셜, API 토큰 등이 있다. 이러한 시크릿은 인증/인가를 통해 검증된 사용자/시스템에게 최소 권한 및 최소 시간의

원칙을 기반으로 발급되어야 한다.

3.6. 암호화

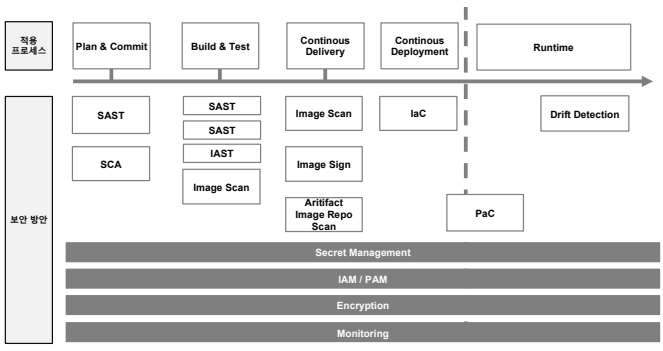
가명 정보 처리 가이드에 따르면 빅데이터, AI 등 다양한 융·복합 산업에서의 데이터 활용이 급증함에 따라 개인정보, 가명정보, 익명 정보에 대한 관리 및 암호화가 필요하다. 이를 충족하기 위한 중요 대표 기술로는 삭제 기술 (마스킹), 암호화 (양방향 암호화, 형태 보존 암호화), 무작위 기술 (토큰화)등이 있다.

플랫폼 엔지니어링에서는 사용 조직에서 필요로 하는 요건에 상응하는 암호화 방식을 제공해야 한다. 암호화 적용을 고려해야 하는 대상 시스템에는 소스 코드, 데이터 베이스, 스토리지 등이 있다. 데이터 베이스 암호화는 컬럼 레벨 암호화, 애플리케이션 암호화, 데이터 암호화로 구분할 수 있다. 스토리지 암호화는 전체 디스크 암호화, 파일 수준 암호화, 클라이언트 암호화로 구분할 수 있다.

4. 데브섹옵스를 반영한 보안 방안

4.1. 단계 별 보안 고려사항

효과적인 보안 프로세스를 구현하기 위해서는 Shift-Left 개념을 적용한 CI/CD 파이프 라인을 구성하는 것과 배포 이후 운영 단계에서 동일한 보안 품질을 유지하는 것이 중요하다. (그림 1)에서는 CI/CD 파이프라인의 전 영역에서 보안을 고려할 사항과 운영 단계에서 보안을 고려할 사항으로 나누어서 제시한다.



(그림 1) CI/CD 파이프라인, 운영 단계에서 고려사항

전통적인 애플리케이션 보안 방식에서는 SSDLC 를 위해 SAST/DAST 를 적용하였으며, 이를 통해 신속하고 반복적인 작업을 자동화하고 CI/CD 파이프라인의 전 영역에서 보안을 적용할 수 있었다. 최근에는 클라우드 네이티브 워크로드의 대중화로 이러한 CI/CD 파이프라인에 컨테이너화 단계가 추가되었으며, 이에 따른 이미지 스캐닝, 서명 등도 반드시 추가로 검증되어야 한다. 이렇게 안전하게 검증된 아티팩트를 쿠버네티스에 배포한다고 가정할 경우에도 사전에 정의된 정책을 수립하고 배포 이전 단계에서 검증하는 것을 권고한다. 이러한 패러다임은 인프라를 코드로 관리하는 IaC 에도 유사하게 적용할 수 있다. IaC 특성상 인프라 정보를 선언적 언어로 작성하고, 코드 작성 단계부터 테스트, 스캐닝 및 정책을 적용하여 보다 안전한 CI/CD 파이프라인을 구현할 수 있다.

배포된 인프라 및 애플리케이션은 운영 단계에서도 동일한 품질로 유지되어야 한다. GitOps 를 도입하여 SSOT 를 구현하고, 선언적으로 작성된 리소스에 대한 Desired State 유지를 위해 Drift 를 감지하고 Remediation 하는 방법을 적용하는 것이 그 예이다.

쿠버네티스 생태계에서는 ArgoCD 같은 표준 도구를 통해 Git 에 저장된 매니페스트와 쿠버네티스에 배포된 리소스 상태 정보가 동일한지 주기적으로 확인하고 변경이 발생할 경우 코드를 기준으로 복구할 수 있다. 유사한 동작 방식으로는 IaC 도구인 Terraform 으로 작성된 구성 파일과 배포된 리소스에 대한 Drift 가 발생할 경우 알림을 발생하거나 필요시 원본 구성 파일을 기준으로 되돌리는 기능이 있다. 위 기능을 활용하면 동일한 품질의 리소스를 유지할 수 있어 Day2 에 발생할 수 있는 악의적인 변경과 거버넌스 위반 사항 등을 관리할 수 있다.

각 프로세스에 대한 세부 보안 적용 방안과 관련 솔루션 그리고 고려되어야 하는 순서는 <표 1>과 같다.

<표 1> CI/CD 파이프라인 보안 방안

CI/CD : 안전한 파이프라인	Plan & Commit	SAST : Git Repo 정적 테스트,IDE Plugin을 활용한 정적 테스트	SonarQube, Snyk
		SCA : 소프트웨어 구성 검사	Mend.io, Synopsys
	Build & Test	IAST : 보안 테스트로 취약한 코드 특정	Nexsus
	Build & Test , Continous Delivery	이미지 스캐닝	Trivy, Clair
	Continous Delivery	이미지 서명	Cosign, notary
		이미지 저장소 스캔	Harbor
	Continous Deployment	(Infra) 메니페스트 스캐닝 : Kubernetes Yaml 파일 사전 검증	kube-scan, Kubesec
		(APP) 연결된 저장소에서 배포 구성을 확인하고 안전한 최신 이미지 버전의 배포를 진행	ArgoCD/Fluxd
CI/CD : 코드형 인프라 관리	IaC	IaC 코드에 대한 취약점 분석(스캐닝)과 정책을 적용하여 배포	IaC Scan : Snyk, tfsec, trivy
			IaC Policy : HashiCorp Sentinel, OPA
	PaC	정책 기반의 관리(PaC) : 모든 환경이 일관되게 규정 준수를 충족을 위해 인프라 배포 제한 관리	OPA, Kyverno, Gatekeeper
배포 이후 Runtime 보안	Drift Detection	운영 시 변경사항(Drift) 최소화	HCP Terraform, Terraform Enterprise, Synk, Spacelift
	Runtime Defense	이미지가 런타임 환경에서 실행되기 전에 지정된 보안 정책을 준수하는지 확인하고, 컨테이너 런타임 환경에 허가 받지 않은 불법 파일 접근 시도와 원격코드 실행을 탐지하고 방지	Sysdig, Falco, Palo alto Prisma Cloud

<표 2>에서 CI/CD 파이프라인 단계 및 Day 2 에 대한 편차 관리에 집중했으며, 위 워크플로우의 전 영역에 걸쳐 표준화된 보안 플랫폼 보안 요소로 고려할 사항은 시크릿 관리, 계정관리/접근관리, 암호화, 모니터링이 있다. 다음 4 가지 요소는 CI/CD 파이프라인 단계 뿐만 아니라 실제 운영 중에 사람 및 시스템에서 주기적으로 사용되어야 하기 때문에 Day0 ~ Day2 전 영역에서 고려되어야 한다.

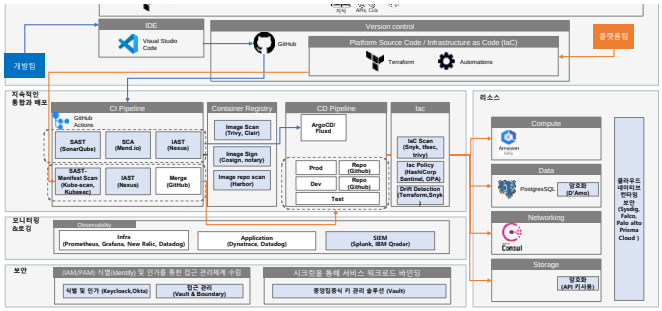
운영의 Observability 영역은 플랫폼 팀에서 표준으로 제공이 가능하지만 본 논문에서는 별도의 SRE 조직에서 환경 및 워크로드 특성을 고려해 최적화된 형태로 적용하는 것을 권고한다. 각 요소들에 대한 세부 보안 적용 방안과 관련 솔루션은 <표 2>과 같다.

<표 2> Runtime 보안 방안

시크릿 관리	API 서비스를 워크로드에 바인딩 하기 위해 통합이 필요한 경우	중앙 집중식 키 관리 솔루션을 통해 주기적으로 시크릿을 교체하거나 필요할 때 발급 및 폐기	Vault
계정 관리/ 접근 관리	사용자/ 시스템의 접근 관리가 필요한 라이프 사이클 전 타임	IAM 솔루션을 활용한 식별 및 인가	Okta, Keycloak, LDAP, AD
		IAM과 연결된 PAM 솔루션 사용한 접근 관리	Delinea Secret Server, CyberArk Privileged Access Manager, Hashicorp Vault & Boundary, BeyondTrust
암호화	사용 조직에서 다양한 요건 준수를 위해 암호화를 필요로 하는 경우	데이터 베이스 암호화 : 컬럼 레벨 암호화, 애플리케이션 암호화, 데이터 암호화	SecureDB, D'Amo
		스토리지 암호화: 전체 수준 암호화, 클라이언트 암호화	API 키 사용
모니터링	인프라 운영 시	별도 SRE 조직을 개설하고 운영	
	애플리케이션 운영 시	애플리케이션 특성에 따라 수집하고 확인해야 하는 모니터링 정보가 상이하여 개발팀과 충분히 협의 후 적합한 방안 준비	
	SIEM	보안 장비의 이벤트나 시스템 로그 등을 주로 임계치 기반으로 연관 분석하여 위협을 탐지	Splunk, IBM QRadar

4.2. 프로세스 별 보안 적용 방안

신규 서비스를 런칭한다고 가정했을 때 어떤 프로세스로 보안을 강화할 수 있을지 (그림 2)에서 인프라 배포 단계, 애플리케이션 배포 단계, 전 영역에 대한 관리로 나누어서 고려해본다.



(그림 2) 플랫폼을 활용하여 신규 서비스 런칭 시 보안 적용 프로세스

우선 개발자를 위한 표준 인프라 구성방안에 대한 보안 표준 프로세스를 제시한다. 개발자는 개발을 위한 인프라 구축을 위해 IDP 에서 제공하는 카탈로그 템플릿을 통해 생성을 요청한다. 해당 카탈로그는 표준 IaC 모듈을 기반으로 컴퓨팅, 네트워크, 데이터베이스 등의 자원을 배포한다. 또한, 해당 모듈은 플랫폼 팀에서 보안/ 컴플라이언스 팀과 사전 검증을 통해 사용할 이미지 종류, 인스턴스 유형, 네트워크 대역(VPC CIDR), 보안그룹 규칙, 관리형 데이터베이스(RDS 등) 유형, 암호화 활성화 여부 등을 반영한다.

뿐만 아니라 IaC 계획단계와 배포단계 사이에 Policy as Code 을 적용하여 보안/컴플라이언스 팀에서 한 번 더 검증을 마쳐 최종 승인 절차를 추가하여 보안성을 높일 수 있다.

다음으로 다음은 애플리케이션 개발 단계이다. 이 단계에서는 데브섹옵스를 고려한 안전한 CI/CD 파이프라인을 제공한다. 개발자는 서비스 개발에 필요한 소스 코드를 시큐어 코딩을 비롯해 SonarQube 를 통한 SAST 와 OWASP ZAP 를 통한 DAST 를 수행하여 검증한다. 검증을 마친 아티팩트는 Nexus 와 같은 프라이빗 아티팩트 저장소에 업로드하고 권한 기반의 접근 제어하여 안전하게 관리한다. 또한, 해당 아티팩트를 컨테이너화 할 때에도 Clair, Anchor 를 통해 이미지 스캐닝 단계를 거친 뒤 Cosign 을 통해 서명 단계를 적용한다. 서명이 완료된 이미지는 Harbor 와 같은 프라이빗 저장소에 푸시 하고 필요 시 저장소에서 제공하는 이미지 스캐닝 엔진을 통해 2 차 검증을 진행한다. 실제 제작된 이미지를 Kubernetes 에 배포한다고 가정할 경우에는 플랫폼 팀에서 제공하는 ArgoCD 와 같은 표준 도구를 사용하고 작성된 매니페스트 파일에 대한 검증은 kube-scan 등과 같은 매니페스트 스캐닝 도구로 검증 단계를 거치게 된다.

최종적으로 워크로드에 배포하기 전에는 IaC 보안에서 언급한 것과 동일하게 Policy as Code 을 적용하여 Shift Left 의 최종 단계에 정책 적용 후 런타임에 배포해야 한다.

5. 결론 및 향후 연구

워크로드 전체에 걸친 보안을 고려한 플랫폼을 활용하면 개발 단계부터 보안을 통합하여 관리하여 잠재적인 취약점을 최소화할 수 있다. 뿐만 아니라 새로운 기술 및 서비스를 안전하고 빠르게 개발하고 배포할 수 있는 기반이 된다. 향후 Google Cloud 의 DevOps Research and Assessment(DORA) 지표를 기반으로 팀의 생산성을 저해하지 않고 보안을 강화하는 방안에 대해 연구할 계획이다.

참고문헌

[1] Gartner, What Is Platform Engineering?
[2] CNCF, Platforms White Paper
[3] Gartner, Privileged Access Management Reviews and Ratings
[4] R. Jabbari, N. bin Ali, K. Petersen and B. Tanveer, "What is DevOps?: A systematic mapping study on definitions and practices", Proc. Sci. Workshop XP, 2016.
[5] Microsoft, Platform engineering guide
[6] K. Carter, "Francois Raynaud on DevSecOps", IEEE Software., vol. 34, no. 5, pp. 93-96, Sep. 2017.