
XPERTS26

XPERTS26

Seguridad de Código

Una sola Plataforma para proteger las aplicaciones

Disclaimer

Fortinet Confidential

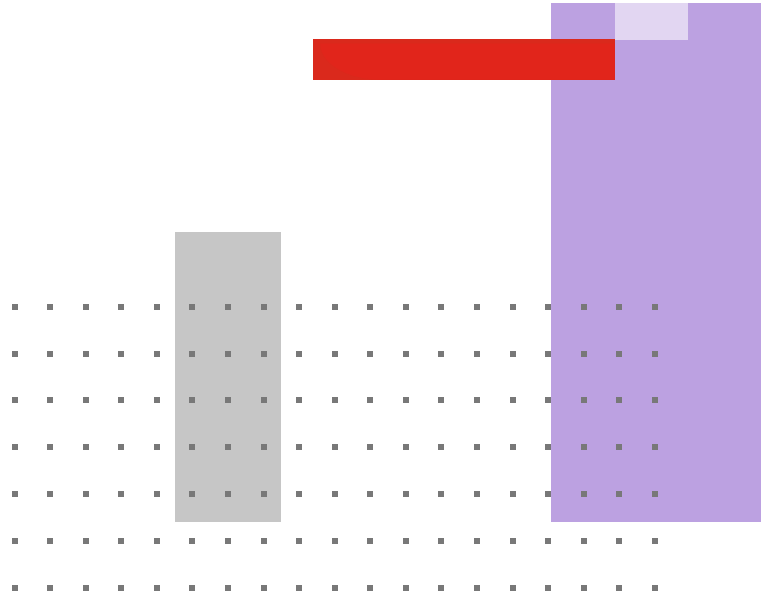
This document contains confidential material proprietary to Fortinet, Inc.

This document and information and ideas herein may not be disclosed, copied, reproduced or distributed to anyone outside Fortinet, Inc. without prior written consent of Fortinet, Inc.

This information is pre-release and forward looking and therefore is subject to change without notice.

The purpose of this document is to provide a statement of the current direction of Fortinet's product strategy and product marketing efforts.

Please note that this Product Roadmap is neither intended to bind Fortinet to any particular course of product marketing and development nor to constitute a part of the license agreement or any contractual agreement with Fortinet or its subsidiaries or affiliates.



XPERTS26

Presentadores

Juan Peña

Sr. Business Development Engineer

David Zambonino

Cloud Subject Matter Expert - LATAM

XPERTS26

Seguridad de Código

Estado actual

Las capacidades que impulsan FortiCNAPP

- Plataforma unificada para la seguridad de código a nube

Code Security

- Software Composition Analysis (SCA)
- Static Application Security Testing (SAST)
- Infrastructure as Code (IaC) Security

Cloud Risk

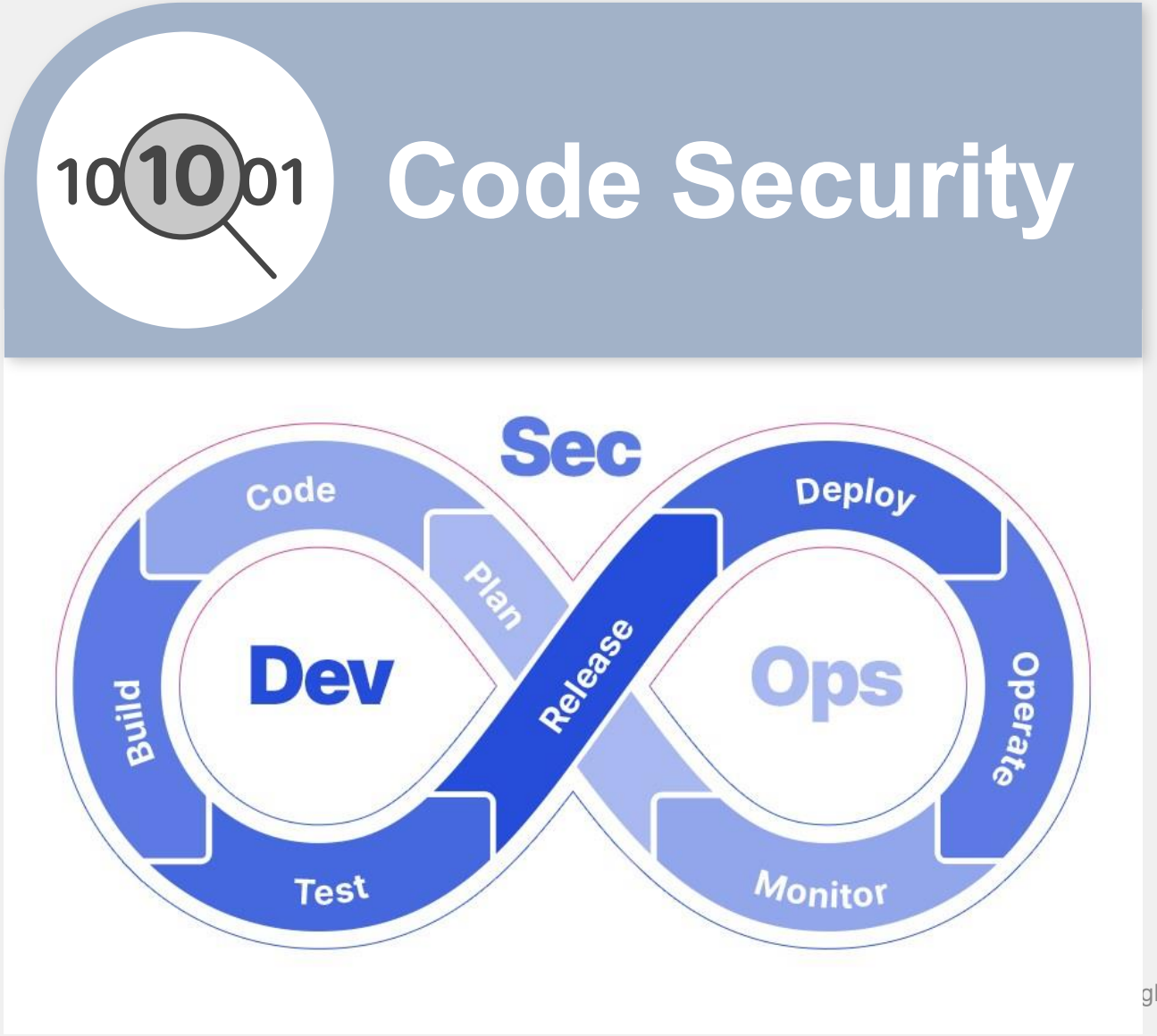
- Vulnerability Assessment
- Security Posture Mgmt (CSPM)
- Cloud Infrastructure Entitlement Mgmt (CIEM)
- Data Security Posture Mgmt (DSPM)

Threat Detection

- Cloud Detection & Response (CDR)
- Cloud Workload Protection (CWP)

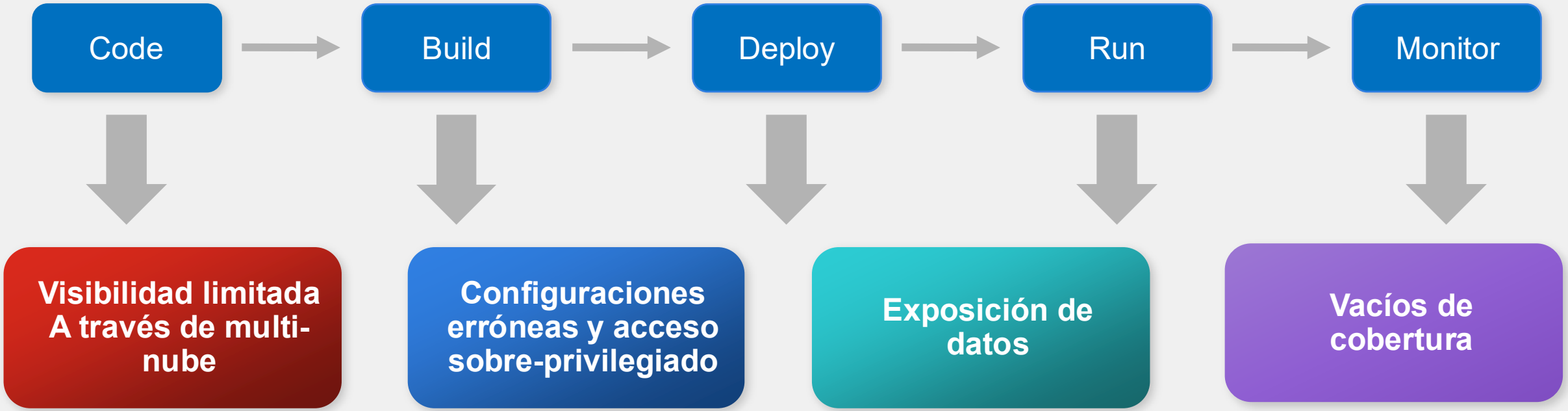
Las capacidades que impulsan FortiCNAPP

- A Plataforma unificada para la seguridad de código a nube



Los nuevos puntos ciegos del desarrollo de software

- Dónde están los vacíos actuales



Impacto masivo

Los pequeños errores tienen un impacto enorme.

Un Secreto filtrado → Compromiso total

Una Librería vulnerable → Exposición global (Log4Shell)

Un error en IaC → Violación instantánea de datos en la nube

La creación de código asistido por IA acelera la entrega y el riesgo

El código generado por IA puede introducir vulnerabilidades ocultas o filtrar datos sensibles.

AI pipelines automatizados puede importar componentes no verificados o permisos excesivos.

Los despliegues impulsados por IA pueden saltarse comprobaciones de seguridad y crear configuraciones erróneas.

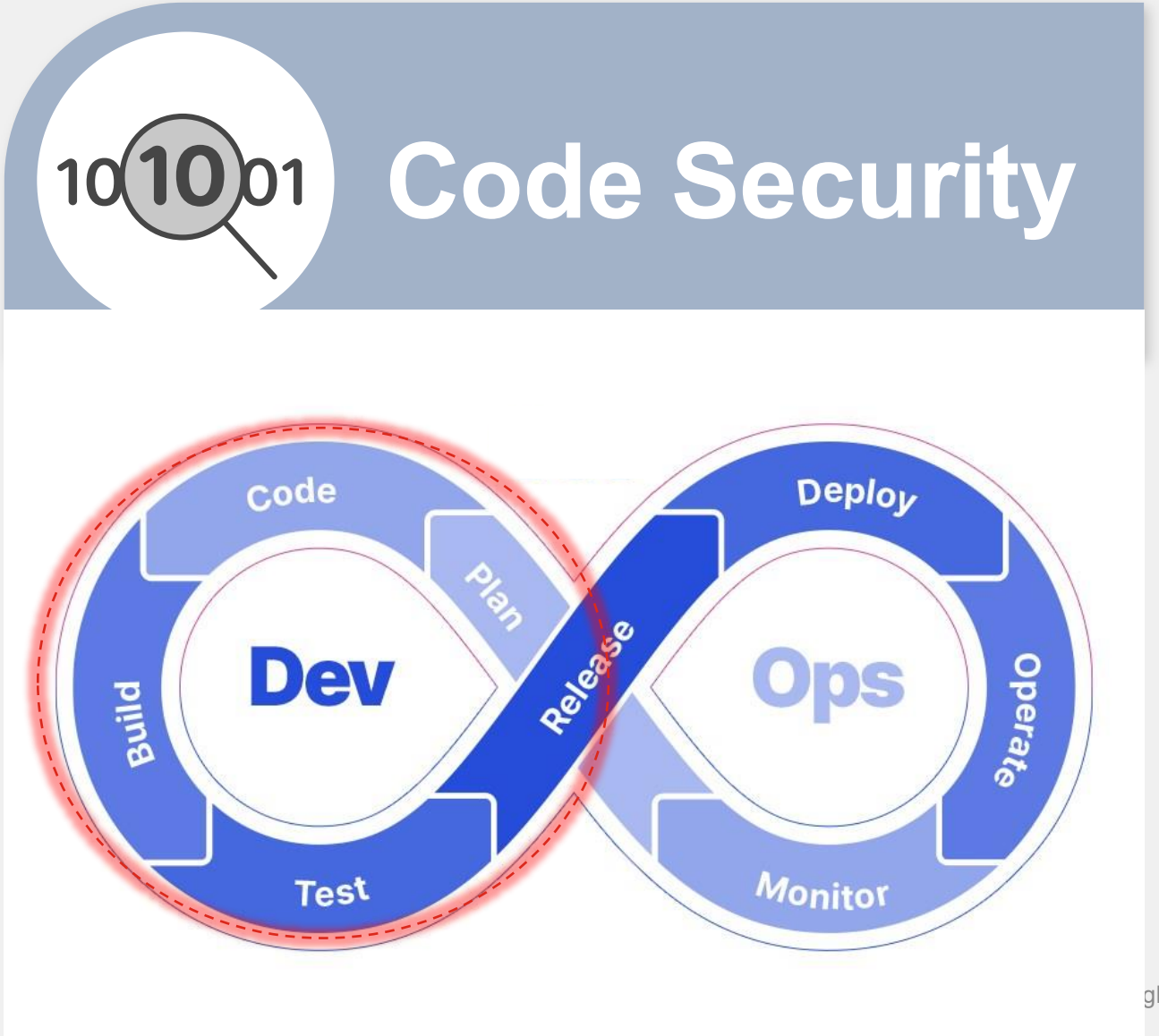
Las cargas de trabajo de IA pueden exponer datos sensibles o ser manipuladas mediante inyección rápida.

Las herramientas tradicionales pasan por alto riesgos específicos de IA y ahogan a los equipos en ruido.

Solución: Shift Left

Shift Left

Integrar la seguridad en primeras etapas de SDLC



Conceptos

Application Code

Programación de aplicaciones, APIs y servicios.

Python, Java, Go, Node.js…

Open-Source Packages

80% de las aplicaciones modernas provienen de estos paquetes.

Log4j, openssl, Django,debug, etc.

Code Repositories

Dónde reside el código, donde los equipos colaboran, dónde ahora apuntan los atacantes.

GitHub, GitLab, Bitbucket

Pipelines

Sistemas automatizados que apoyan la creación código para producción.

GitHub Actions, GitLab CI, Jenkins

Source Code Mgmt

proceso de rastrear modificaciones y gestionar cambios en el código fuente y los activos.

Git

Version Control & Git

Registro de cada cambio, branch, version o merge.

Git = Estándar utilizado en todas partes

¿Qué es una aplicación?

Frontend
Capa de interacción

Backend
Capa lógica

Data

Application Code
20%

3rd-party libraries
80%

Por qué?

- Elementos fundamentales, acelerar el desarrollo y reducir costes
- Evitan reinventar la rueda: El código abierto proporciona código probado y reutilizable para funciones comunes

Ejemplos Log4j, Lodash, debug, etc. (millones descargas / semana)

Retos?

- Grandes cadenas ocultas de dependencias → gran superficie de ataque difícil de ver
- Un paquete vulnerable afecta a millones (ex: Log4j)
- Las bibliotecas envejecidas o sin mantenimiento dejan debilidades duraderas
- Los ataques en la cadena de suministro atacan los propios paquetes(typosquatting, injection)
- Versiones difíciles de rastrear& CVEs en cientos de componentes de código abierto

Flujo de trabajo del código

Debilidades del código

Entrega de infraestructura mal configurada

Infra As Code

Terraform, Cloudformation, etc.

Application Code

Java, Python, etc.

IDE

Coding

VS Code • IntelliJ • PyCharm

Version Control

Sharing

GIT

Código malicioso insertado

Code Repos

Storing

GitHub • GitLab • Bitbucket

Secretos filtrados

Config Code

YAML, JSON, etc.

Pipelines

Building

Shipping

Azure DevOps, Github Actions, Jenkins

Supply Chain Attack

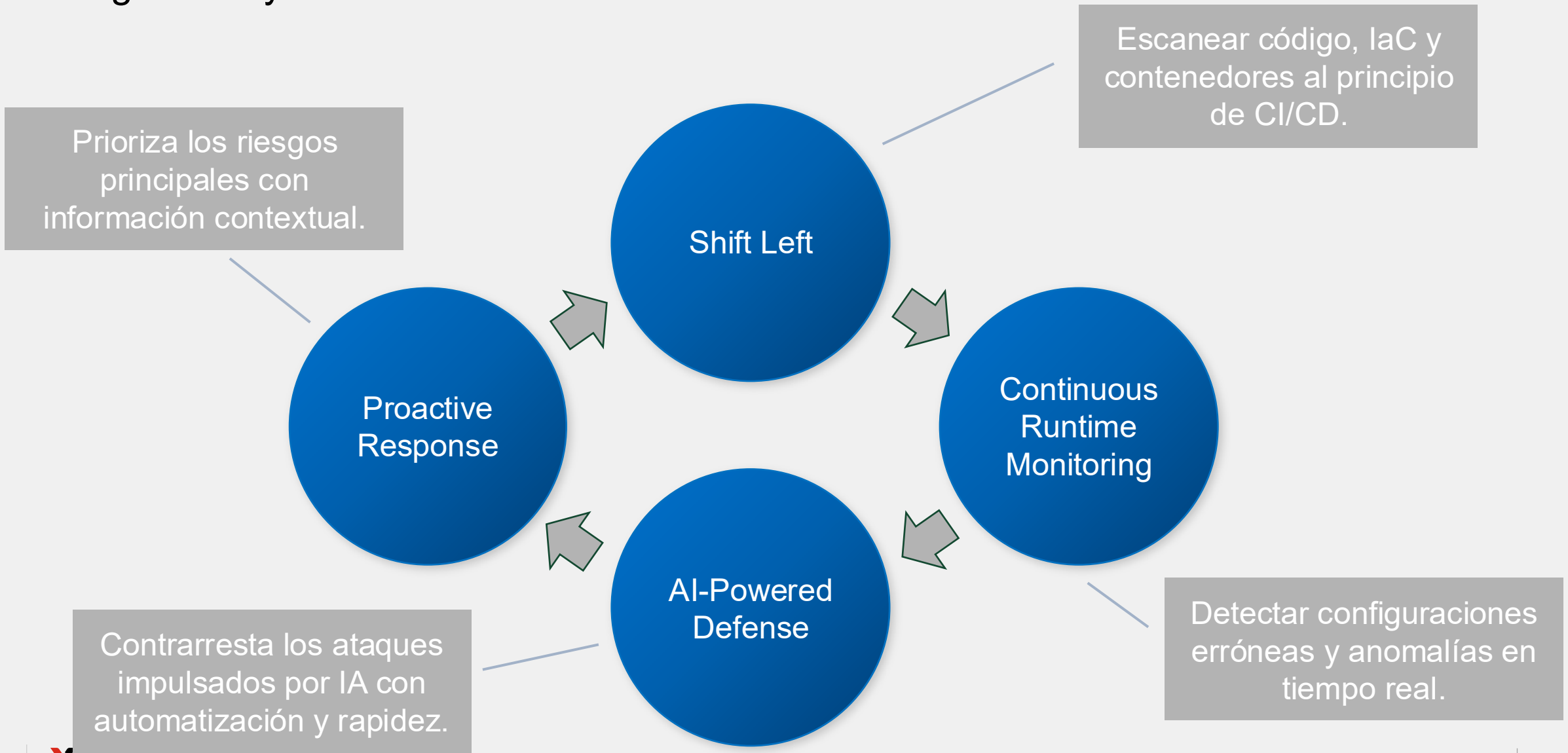
Cloud

Running

GCP, Azure, AWS, Private Cloud

Cambio de seguridad en la era de la IA

- Shifting Security Left



Tipos de escaneos disponibles

- Obtener visibilidad de la cadena de suministro de software

SAST

Static application security testing

Analiza el código fuente sin ejecutarlo para identificar vulnerabilidades y fallos de código

Supports *Go, Java, JavaScript, PHP, Python, Typescript*

SCA

Software Composition Analysis

Identifica y analiza componentes y dependencias de terceros en tu base de código

Secrets

Secrets

Escanea contraseñas abiertas en texto plano

DAST

Dynamic Application Security Testing

Simula exploits usando la URL frontal de la aplicación, utilizando el complemento de producto FortiDAST

Infrastructure as Code

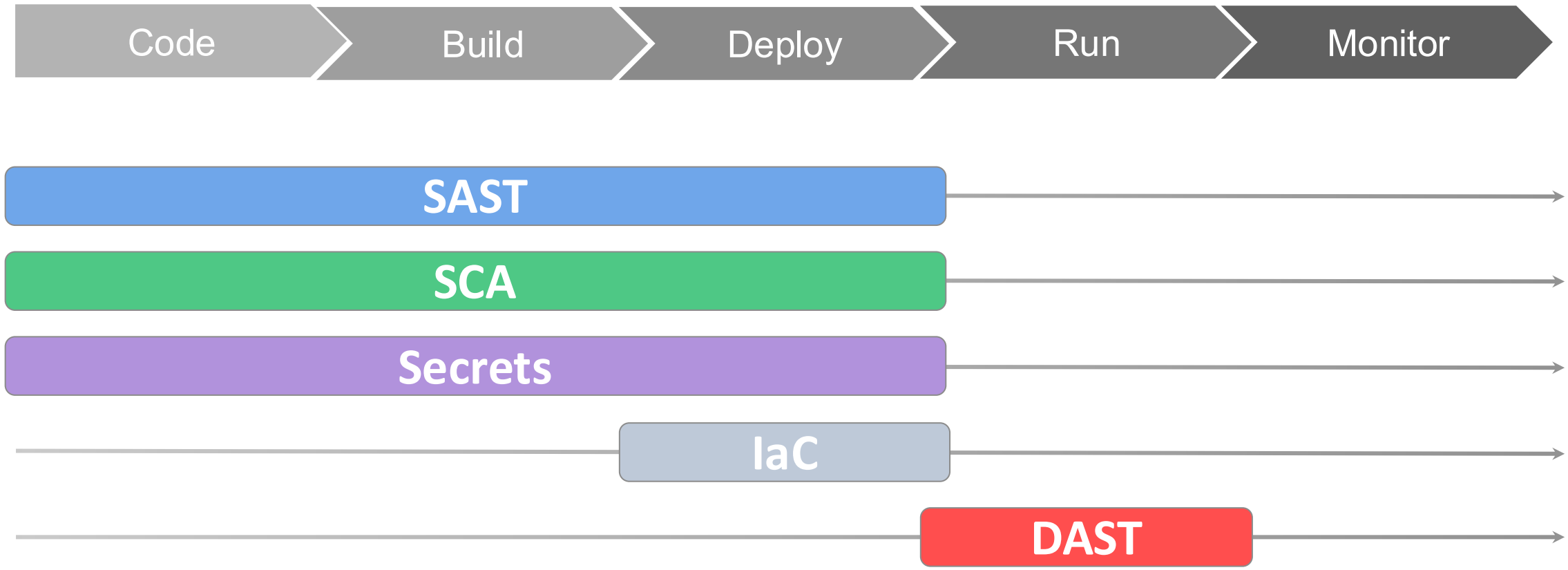
Infrastructure as Script security scanning –

scanning IaC scripts like terraform, etc.

Supports Terraform, Cloud Formation, Docker and Kubernetes

DevSecOps y los escaneos

- A lo largo de todo el ciclo de vida del desarrollo de software (SDLC)




FortiCNAPP

Code Security: SAST

- Static Application Security Testing

Escanea tu código fuente en busca de problemas de seguridad y patrones de código inseguros antes de que se construya o despliegue.

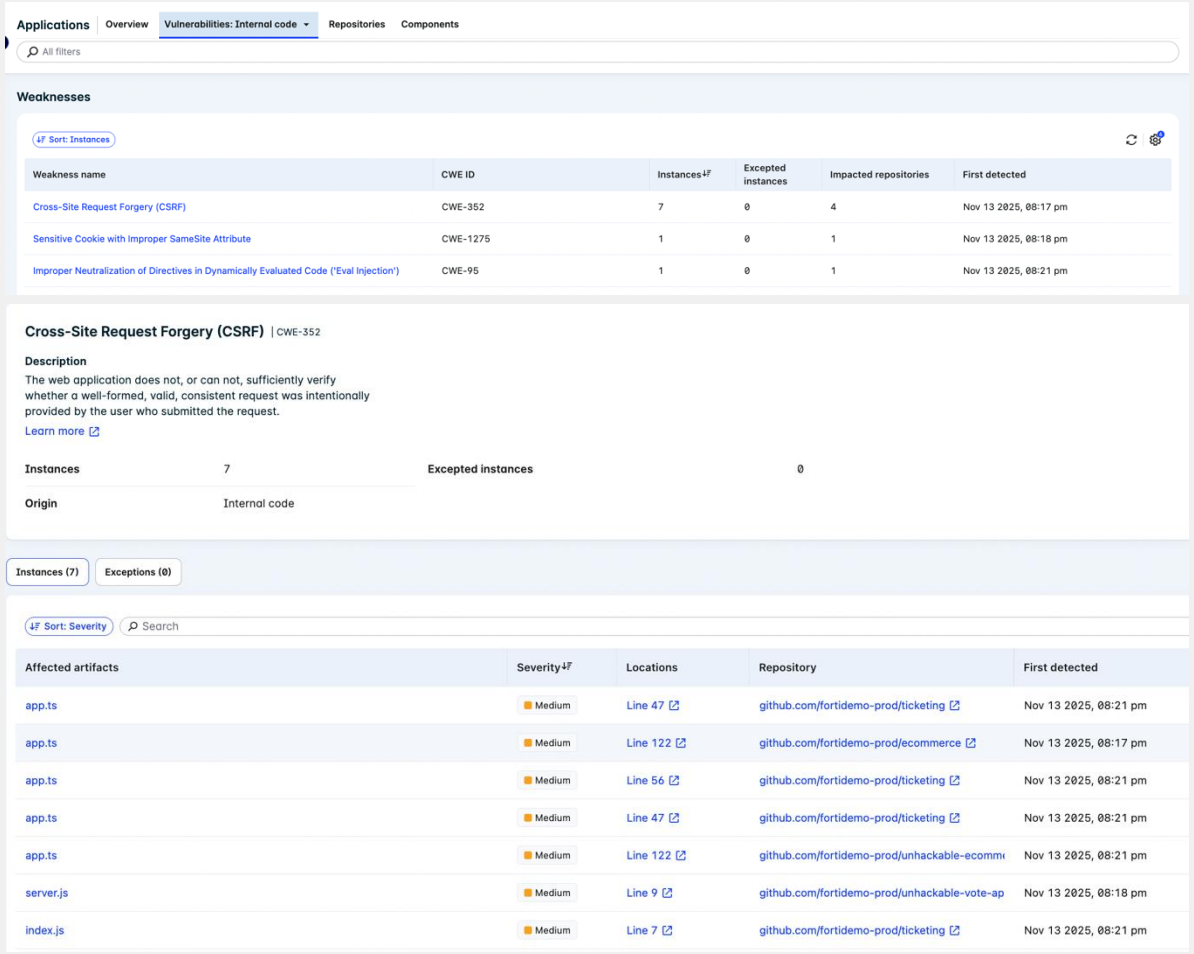
SAST identifica:

- Patrones de codificación inseguros (ejemplos: injections, unsafe deserialization, insecure crypto)
- Uso de APIs peligrosas y funciones de alto riesgo
- Las formas comunes en que la entrada no confiable puede llegar a operaciones sensibles (Detección basada en patrones)

```
sast:
  scan:
    enabled: true
    severityThreshold: high        # only fail on High/Critical
    languages: [java, javascript]  # optional filter
    excludePaths: ["test/", "docs/"]
```

Los hallazgos se exportan en SARIF y pueden ser consumidos por herramientas de terceros: GitHub, GitLab, Azure DevOps, IDE extensions, etc.

FortiCNAPP muestra CWEs (weaknesses)



Code Security: SCA

- Software Composition Analysis

Escaneos en busca de vulnerabilidades en **librerias/componentes** utilizado por su aplicación

SCA identifica:

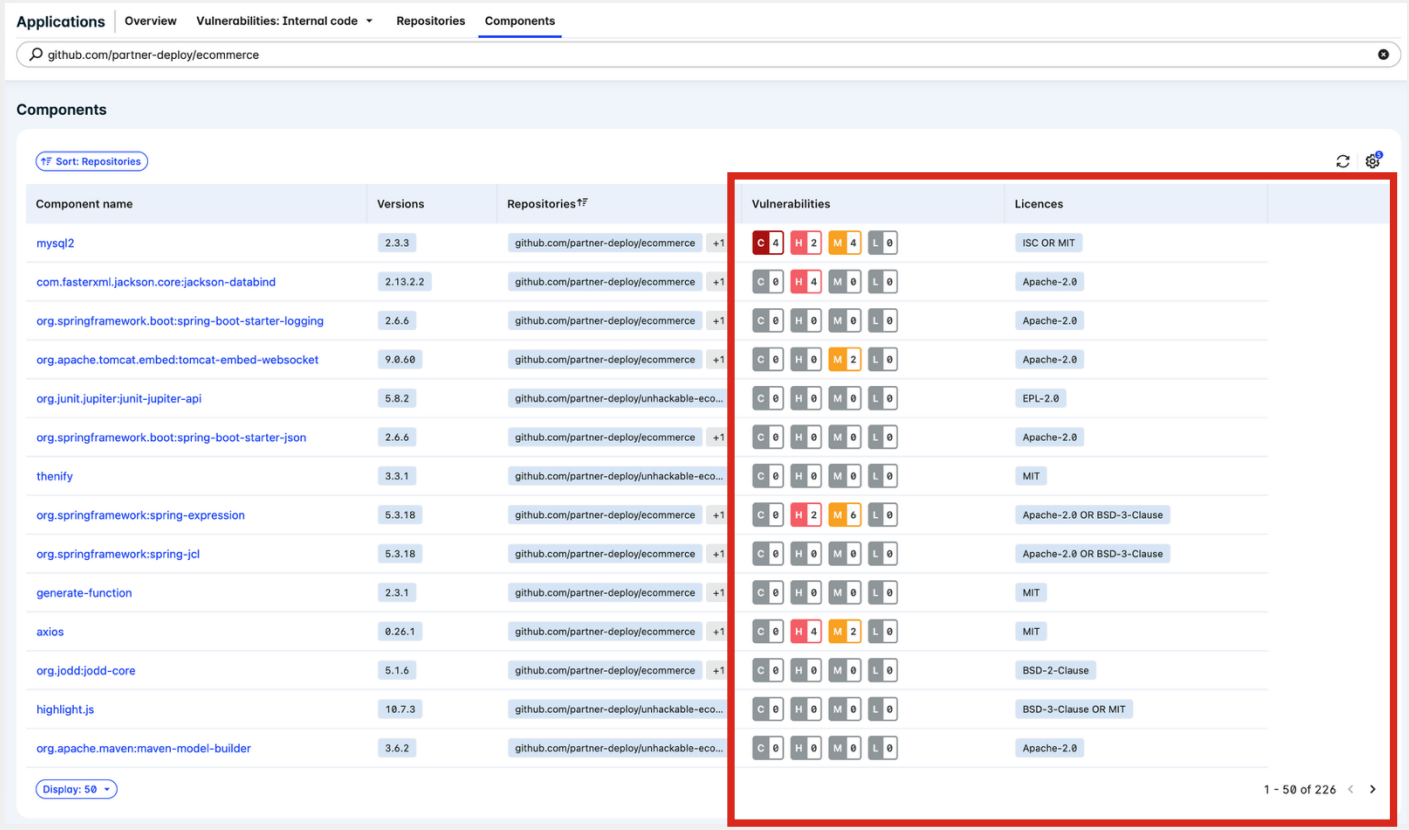
- Librerias de terceros y sus versiones
- Vulnerabilidades conocidas(CVE) en las versiones usadas
- Licencias aplicables a estos componentes

codesec.yml –Local o en repositorio para añadir una política de licencia

```
sca:
    scan:
        licenseDetection: true
        licensesNotAllowed: []
        licenseCategoriesNotAllowed: [forbidden, restricted]
```

SBOM salida usando Sarif, CycloneDX, SPDX, Lacework, GithLab puede importarse a herramientas de terceros

FortiCNAPP muestra SBOM (software bills of material)



Code Security: Secrets detection

Encuentra secretos filtrados en tu código fuente

Los secretos expuestos y codificados pueden dar a usuarios maliciosos acceso autenticado a tus servicios.

Secret type	Severities	Categories	Instances
Generic Password Credentials	High	Generic	2
Aws Credentials	Critical	AWS	2
Postgresql Connection Strin...	High	PostgreSQL	1

unhackable-vote-app / apps / result / server.js ↑ Top

Code | Blame | 92 lines (76 loc) · 2.32 KB | Raw

```
26
27    var pool = new pg.Pool({
28      connectionString: 'postgres://postgres:postgres@'+db+'/postgres'
```

Instances (1) | Exceptions (0)

Sort: 0 | Search

Repository	Severity	Category	Found in	Last scan	Visibility	Act
github.com/partner-deploy/unhackable-vote-app	High	PostgreSQL	apps/result,	Sep 16 2025, 05:37 pm	Unknown	•••

ne) {

Full list of detectable secret types:
https://docs2.fortinet.com/document/lacework-forticnapp/latest/administration-guide/121557/detectable-secrets

Code security: infrastructure as code (IaC)

- Encuentra configuraciones arriesgadas de servicios en la nube antes de que se aprovisionen

Crear, probar y utilizar políticas personalizadas
Escrito en la lengua Rego
  - Verifica infrastructure as code (IaC) para configuraciones erróneas de riesgo

Anular la severidad de las políticas IaC integradas

VSCode plugin para IaC

IaC Technology	Format
CloudFormation	JSON, YAML
Kubernetes	YAML
Terraform	HCL, JSON Plan

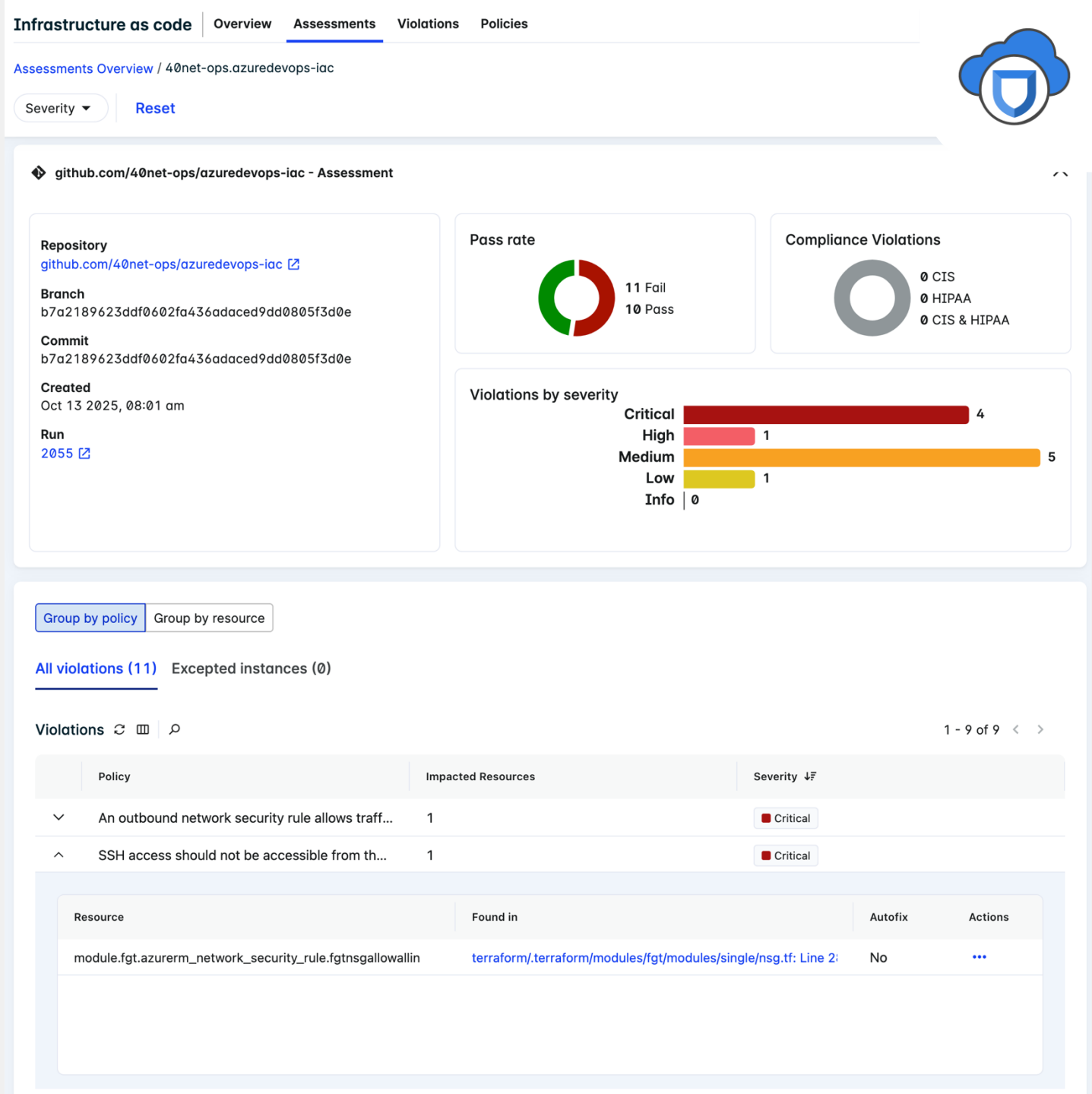
El analizador estático FortiCNAPP IaC se basa en el popularOpen Policy Agent (OPA)

El escaneo IaC funciona a través de la interfaz de línea de comandos(CLI) con Herramientas CI/CD como: Jenkins, Circle CI, GitHub Actions and GitLab Pipelines, Azure DevOps

Code Security: IAC

- Ejemplo

Ejemplo de escaneo de un Terraform IaC despliegue desde la cadena de GitHub Actions y publica los resultados en FortiCNAPP

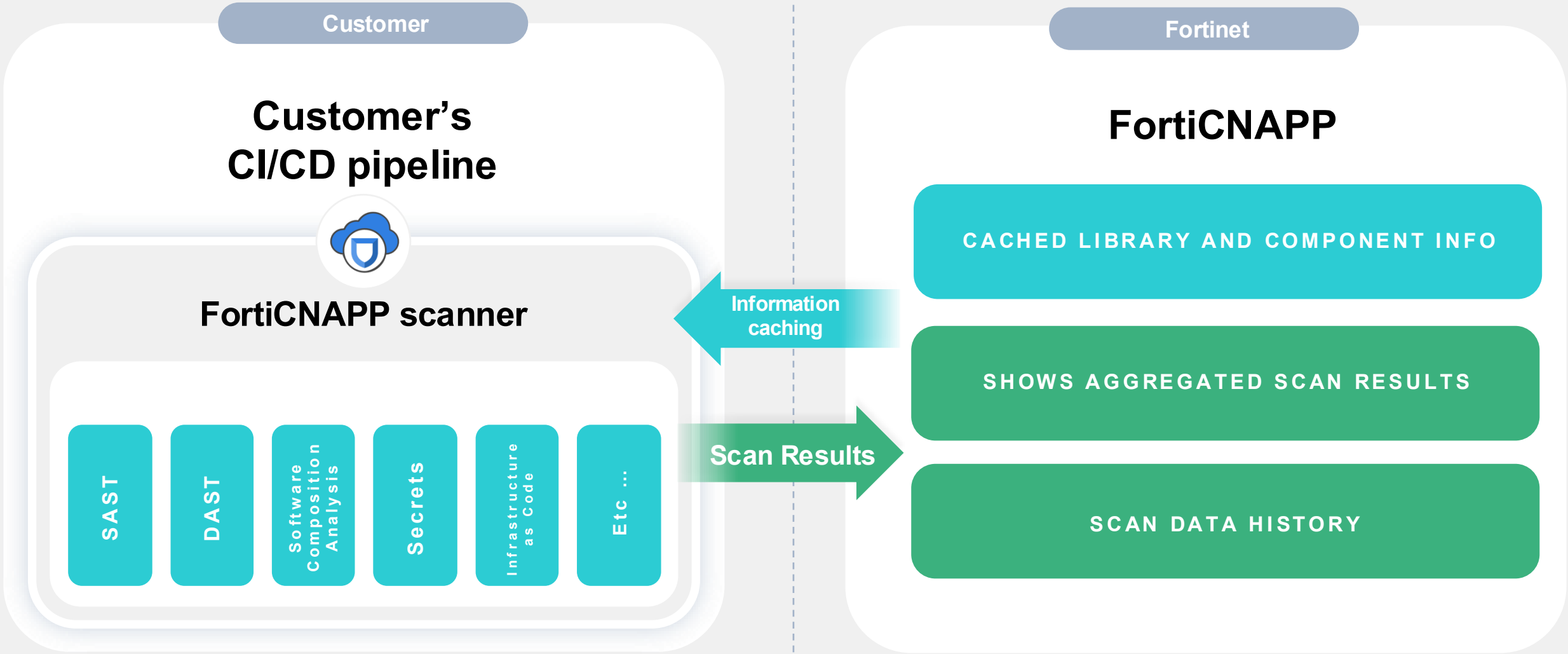


XPERTS26

Arquitectura y Plataforma

Arquitectura de seguridad de código de FortiCNAPP

Integración del proceso de CI/CD



Arquitectura de seguridad de código de FortiCNAPP

Integración de la gestión del control de código fuente

Customer

Customer's SCM system

GitLab

GitHub

Bitbucket

Actions on scan results

Copy of code

Fortinet

FortiCNAPP

SCAN CODE RESULTS

SCAN DATA HISTORY

Herramientas de CI/CD compatibles con FortiCNAPP

Integración continua/Entrega continua (CI/CD)

GitLab

GitHub

Azure DevOps

AWS CodePipeline

Bitbucket

circleci

Google Cloud Build

JFrog

Bamboo

Jenkins

Travis CI

Steps:

1. Developer copies code segment into CI/CD configurations
2. CI/CD tools download FortiCNAPP CLI or container
3. Scans are performed
4. Only results (and small code snippets) are uploaded to FortiCNAPP

Scanning is performed on the CI/CD infrastructure

XPERTS26

Vibe Coding & Claude Code Plugin

El Problema: Código IA ≠ Código Auditado

El cambio de paradigma

- Asistentes IA como Claude Code generan features completas, refactorizan módulos y hacen commits a una velocidad que ningún equipo de seguridad puede seguir
- El código en repositorios es, cada vez más, código que ningún humano escribió
- Cada herramienta de seguridad existente asume revisión humana días después del commit
- El código IA es probabilístico: el mismo prompt produce resultados distintos, ninguno auditado

La brecha de seguridad

La verificación de seguridad necesita ser:

✓ **Determinista**
✓ **Trazable a una política**
✓ **Defendible ante un auditor**

Claude Code genera el código.
FortiCNAPP lo verifica.
El loop cierra en segundos.

4 Dimensiones de Escaneo Automático

Activadas automáticamente dentro de la sesión de Claude Code

Secrets Detection

- Detecta API keys, tokens y credenciales al instante que se escriben al disco
- Evita que secretos lleguen al repo, historial git o logs de CI
- Una vez comprometido, sólo queda rotar — mejor prevenir

SCA — Software Composition Analysis

- Verifica cada dependencia agregada por Claude contra la base CVE completa
- Sin números CVE inventados — lookup autoritativo con CVSS score
- Guía de remediación accionable incluida

SAST — Static Application Security Testing

- Detecta inyecciones, patrones inseguros, errores de autenticación/autorización
- Identifica policy IDs específicos y ubicación exacta (archivo + línea)
- Sin descripciones vagas — findings precisos para fix preciso

IaC Misconfiguration Scanning

- Cubre Terraform, CloudFormation, Kubernetes manifests y Helm charts
- Mapeado a frameworks: CIS, SOC 2, PCI-DSS y NIST
- Reglas curadas para infraestructura cloud segura por defecto

Dos Modos de Escaneo

Pre-Commit (Default)

- Intercepta el comando git commit antes de ejecutar
- Escanea archivos en staging (staged files only)
- Bloquea el commit si hay findings Críticos o Altos
- El código nunca llega al historial del repositorio
- Claude recibe el motivo del rechazo, genera el fix, re-stagea y reintenta — todo en la misma sesión

Recomendado

Post-Task

- Se ejecuta como stop hook al finalizar cada tarea de Claude
- Escanea todos los archivos modificados en la sesión
- Si detecta findings Críticos/Altos, Claude los recibe y corrige en la misma sesión
- Si no hay issues, permanece silencioso (zero noise)
- Ideal para tareas que no involucran commits directos

La seguridad ocurre CON el trabajo, no DESPUÉS del trabajo.

Principios de Diseño Clave

Developer Experience First

Escanea únicamente los archivos que Claude modificó en la tarea actual. Caché SCA por hash de manifest — archivos sin cambios no cuestan tiempo adicional.

Security Context Awareness

Al iniciar la sesión, el plugin inyecta contexto de seguridad en la conversación. Claude escribe secure defaults desde el inicio: CIDR restrictivos, cifrado habilitado por defecto.

Exact File Matching

Los findings filtran sólo los archivos que realmente cambiaste. Sin falsos positivos por proximidad de directorio — si editaste main.tf, ves hallazgos de main.tf.

Escaneo Paralelo (v1.7.0)

IaC y SCA corren simultáneamente para minimizar el tiempo de espera. Log rotation automático y output limpio en texto plano sin ruido interno.

Remediación Precisa

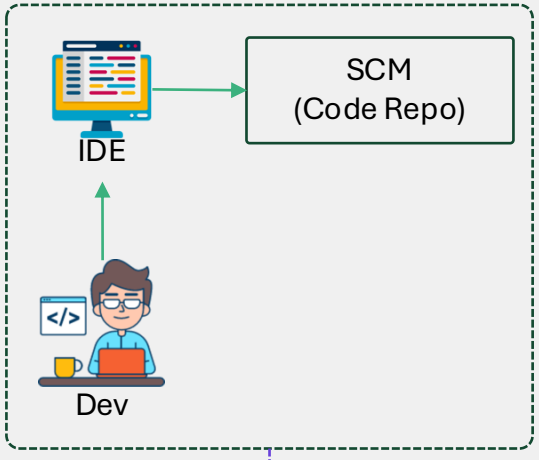
Cada finding incluye: policy ID exacto, archivo y línea, patrón de remediación recomendado. Precisión trazable para auditores y fixes de calidad para Claude.

Primera Línea, No Única Línea

Complementa — no reemplaza — el escaneo en CI/CD pipeline. La detección determinista maneja la verificación; Claude maneja la conversión de finding en fix.

Security by design
Drill-down

Shift Security Left



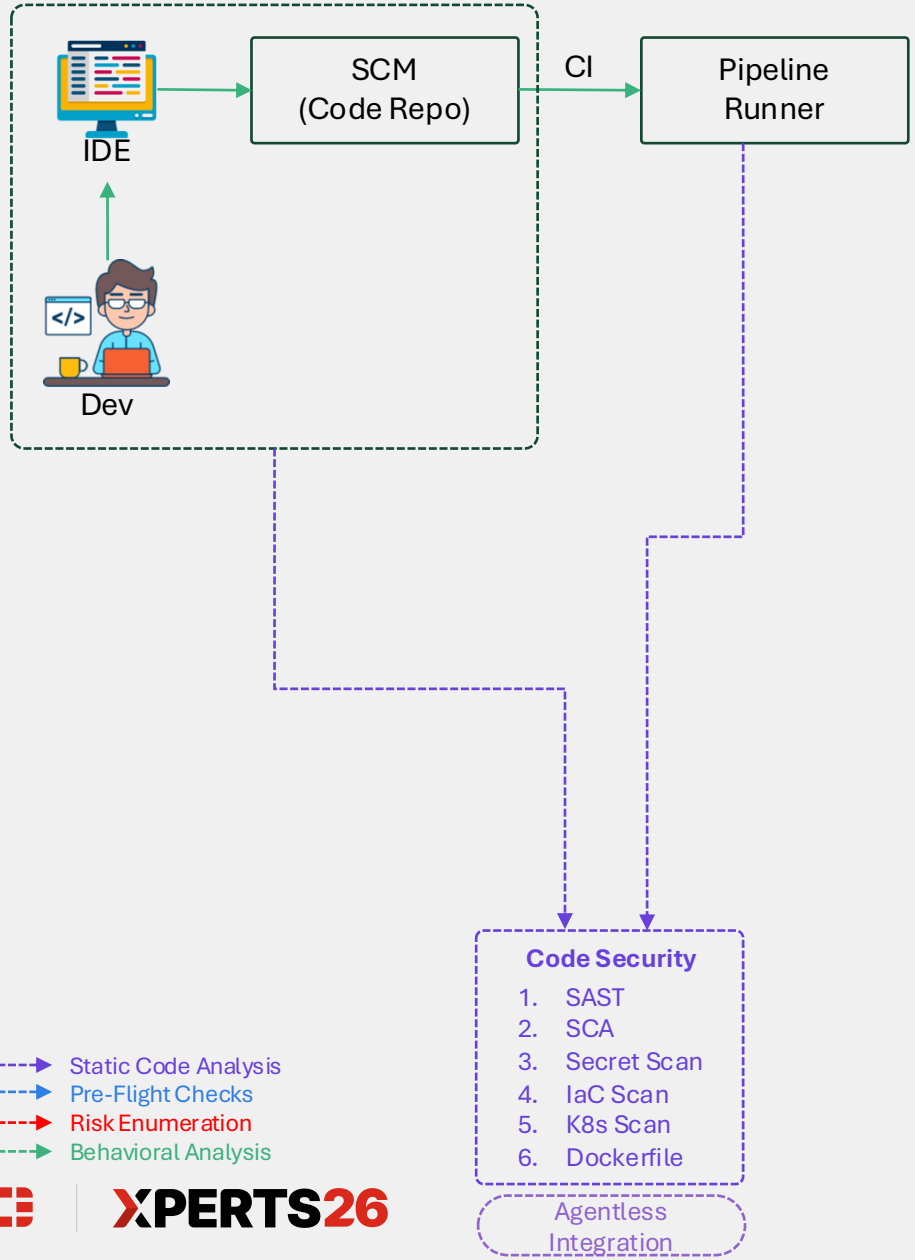
Code Security

1. SAST
2. SCA
3. Secret Scan
4. IaC Scan
5. K8s Scan
6. Dockerfile

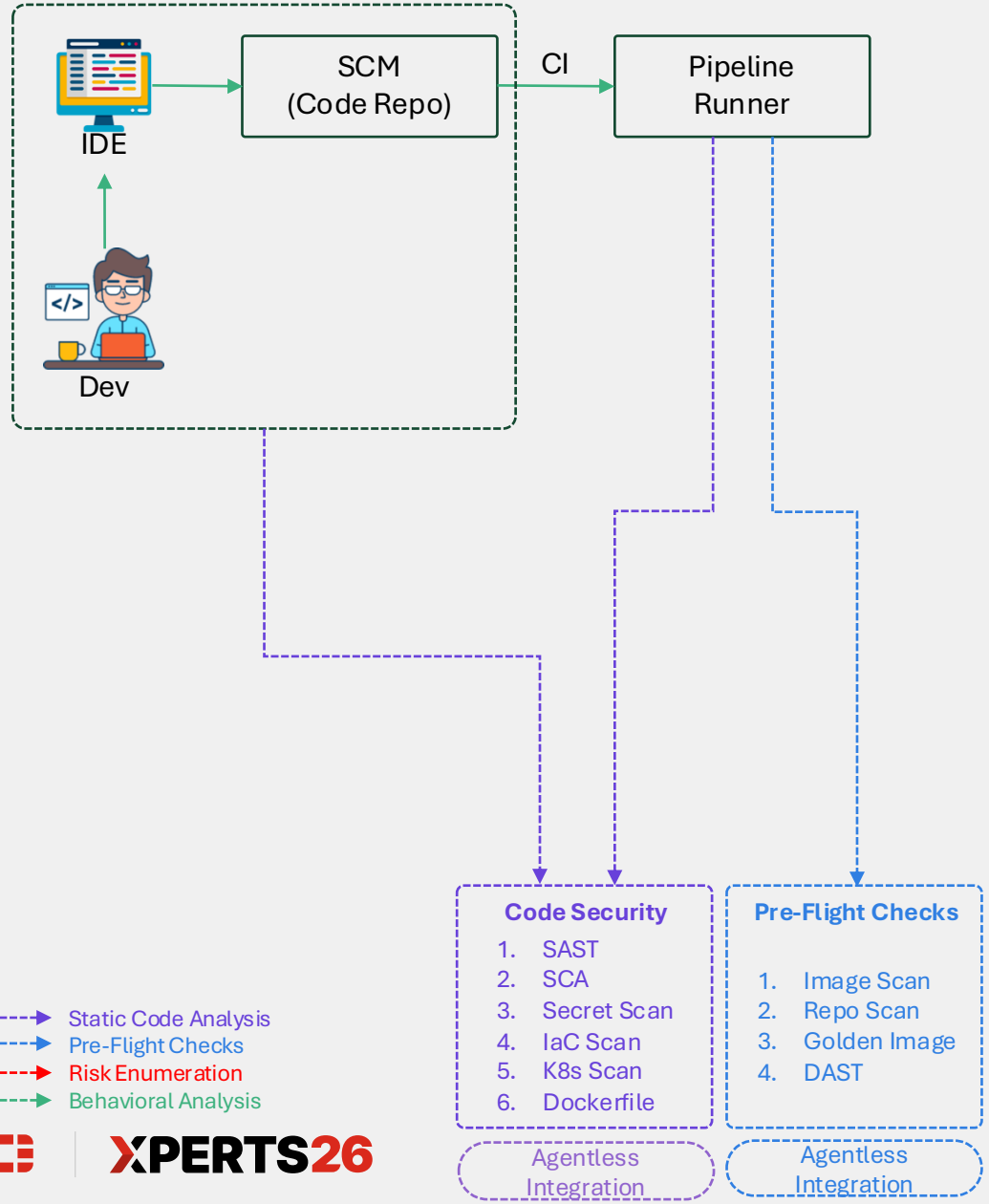
Agentless Integration

Static Code Analysis
Pre-Flight Checks
Risk Enumeration
Behavioral Analysis

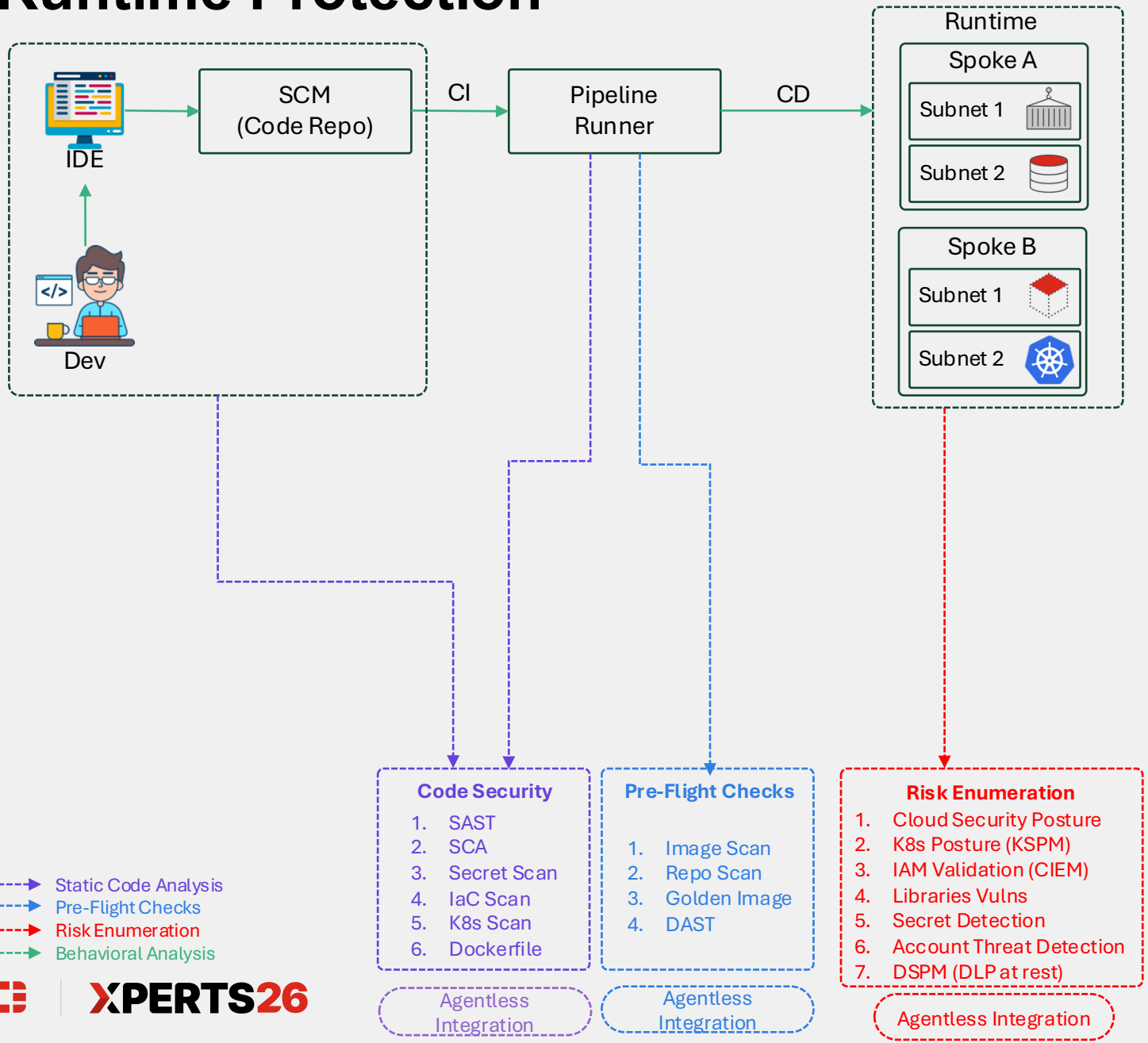
Pipeline Integration



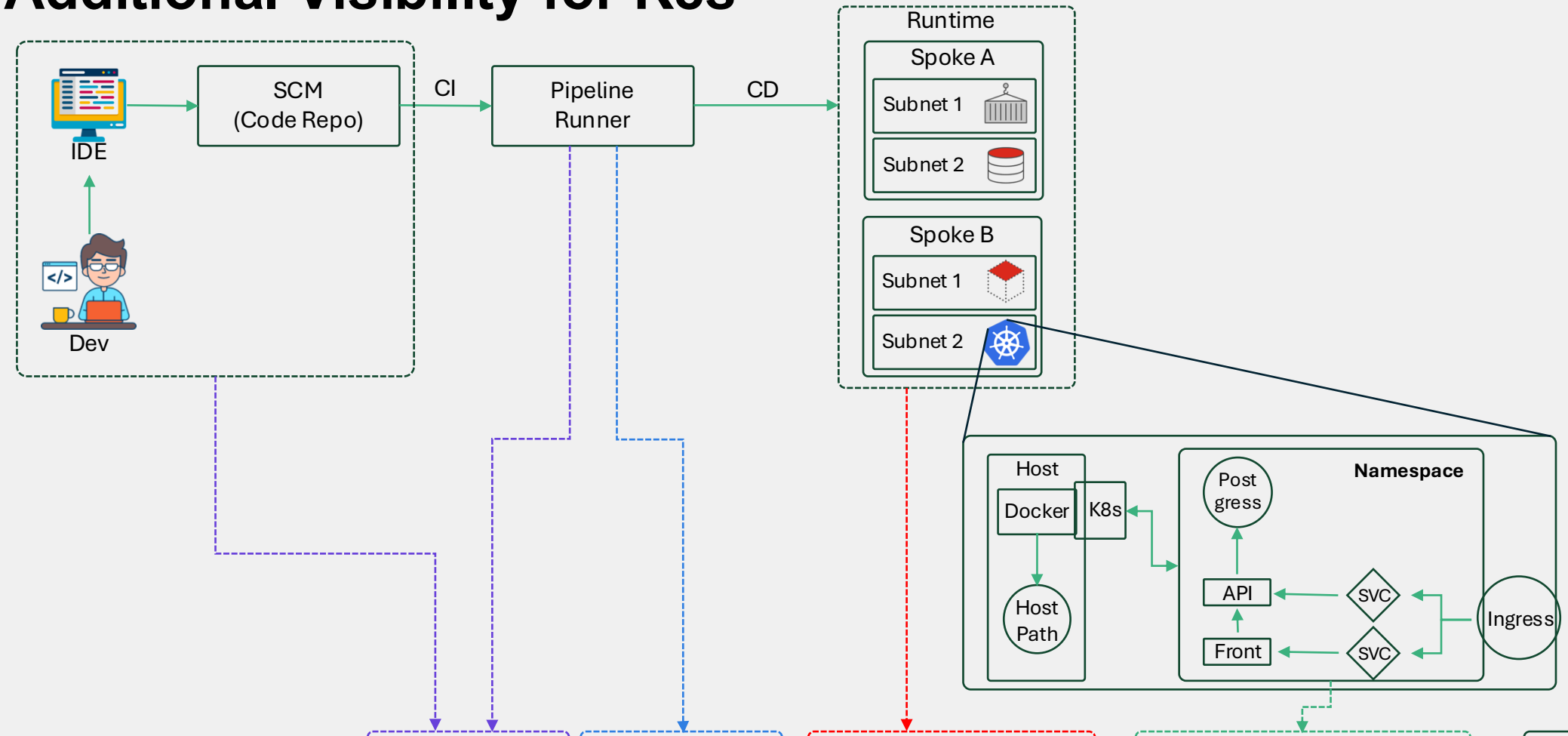
Pre-Flight Checks



Runtime Protection



Additional Visibility for K8s



Code Security
1. SAST
2. SCA
3. Secret Scan
4. IaC Scan
5. K8s Scan
6. Dockerfile

Agentless Integration

Pre-Flight Checks
1. Image Scan
2. Repo Scan
3. Golden Image
4. DAST

Agentless Integration

Risk Enumeration
1. Cloud Security Posture
2. K8s Posture (KSPM)
3. IAM Validation (CIEM)
4. Libraries Vulns
5. Secret Detection
6. Account Threat Detection
7. DSPM (DLP at rest)

Agentless Integration

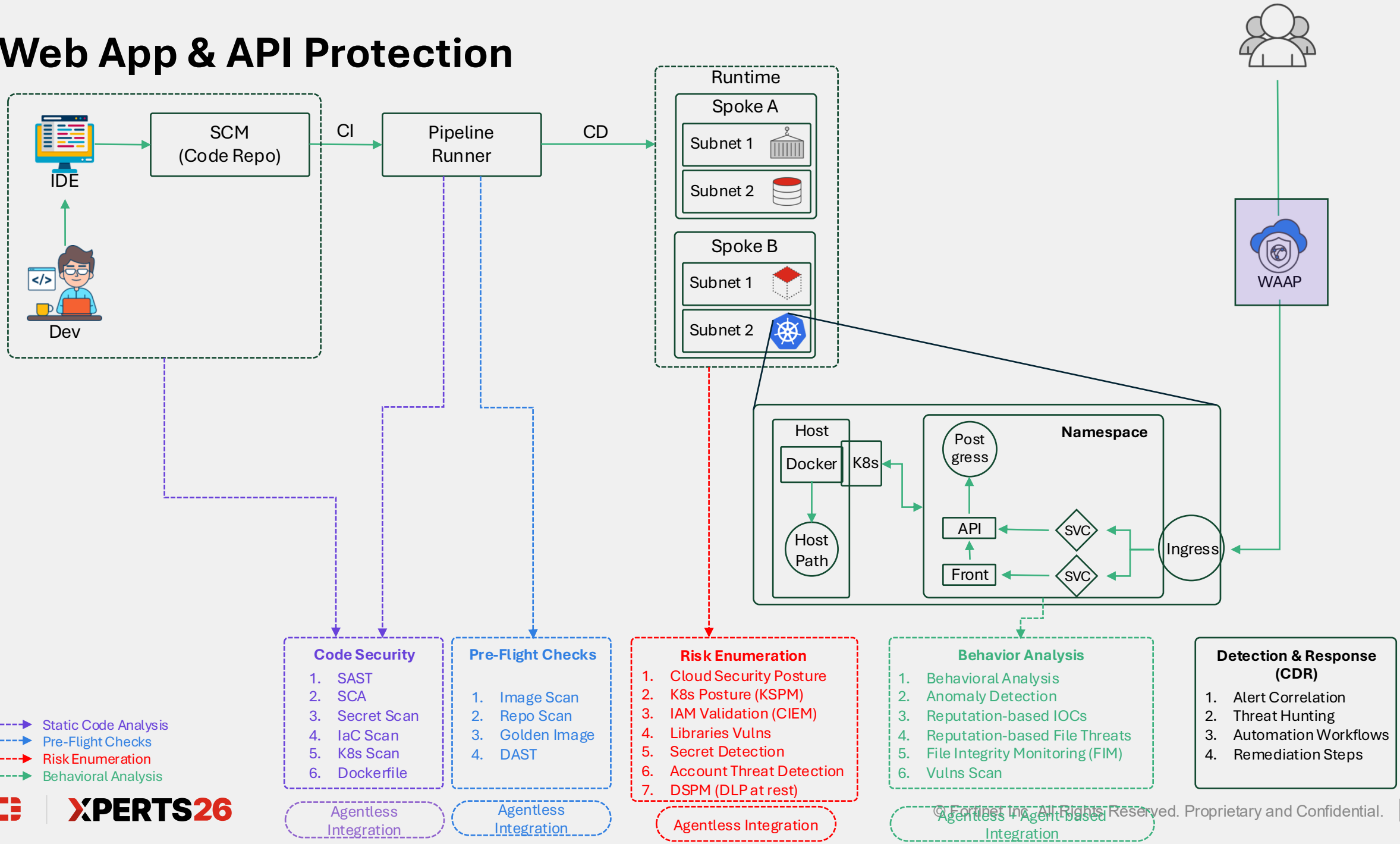
Behavior Analysis
1. Behavioral Analysis
2. Anomaly Detection
3. Reputation-based IOCs
4. Reputation-based File Threats
5. File Integrity Monitoring (FIM)
6. Vulns Scan

Agentless + Agent-based Integration

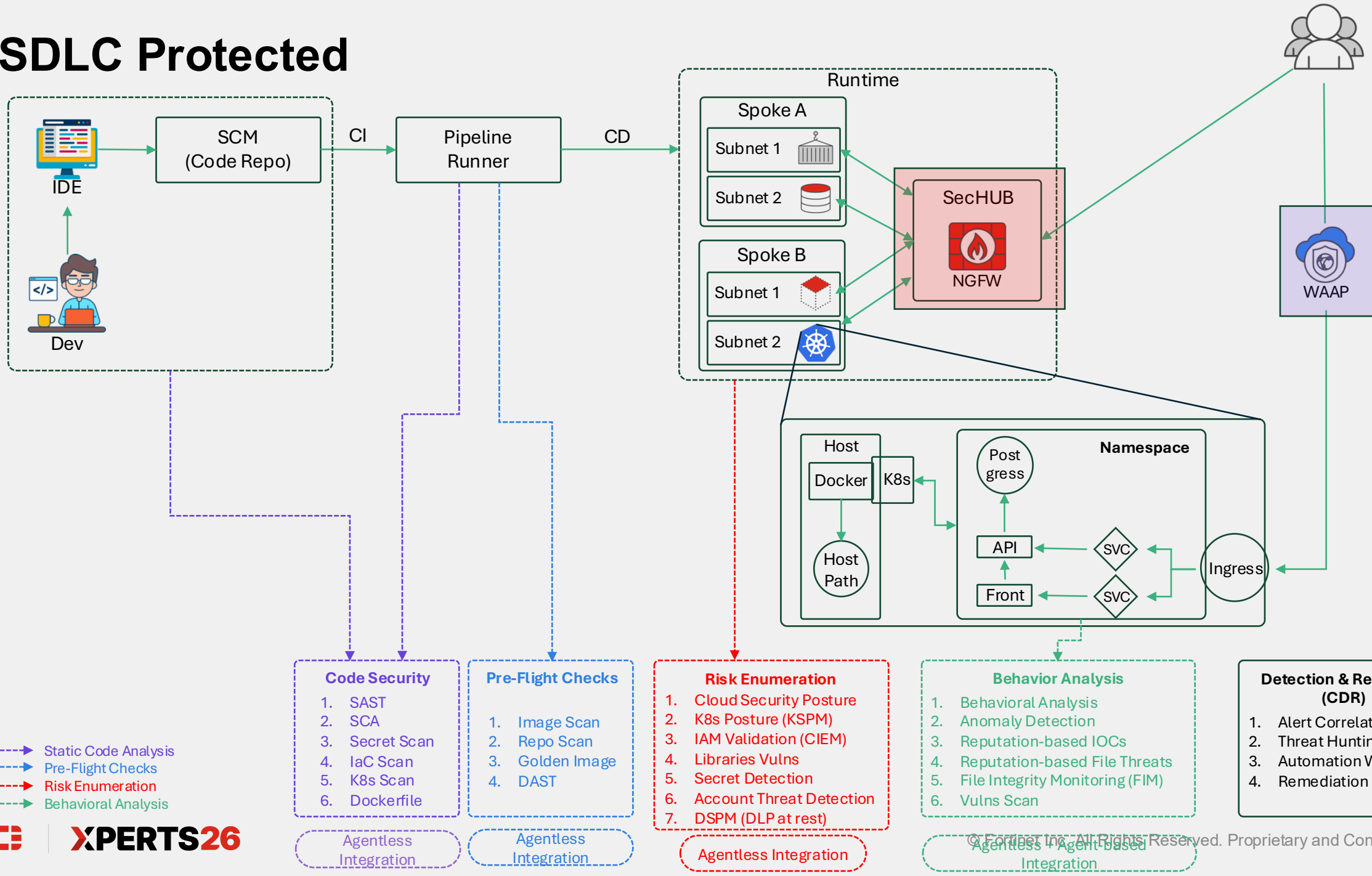
Detection & Response (CDR)
1. Alert Correlation
2. Threat Hunting
3. Automation Workflows
4. Remediation Steps

Static Code Analysis
Pre-Flight Checks
Risk Enumeration
Behavioral Analysis

Web App & API Protection



SDLC Protected




XPERTS26