Universidad Nacional del Nordeste
Facultad de Ciencias Exactas, Naturales y Agrimensura
Ingeniería del Software II

Arquitectura de Confianza Cero

Zero Trust Architecture (ZTA)

Ariel Antinori
ariel.antinori@comunidad.unne.edu.ar

Junio 2026

Resumen

La Arquitectura de Confianza Cero (Zero Trust Architecture, ZTA) representa un cambio paradigmático en el diseño de sistemas de seguridad para aplicaciones distribuidas. Bajo la premisa de que ninguna entidad—usuario, servicio o dispositivo—debe recibir confianza implícita por su ubicación de red, ZTA impone verificación explícita, mínimo privilegio y monitoreo continuo en cada capa del sistema. Este documento analiza sistemáticamente los fundamentos teóricos, los componentes técnicos y los patrones de implementación de ZTA en arquitecturas de microservicios, abordando tecnologías clave como SPIFFE/SPIRE, mTLS, OAuth 2.0, OpenID Connect, Open Policy Agent (OPA) y service meshes. Se discuten los desafíos operativos, casos de adopción industrial y tendencias emergentes incluyendo la integración con inteligencia artificial, entornos multi-nube y criptografía post-cuántica.

Palabras clave: Zero Trust Architecture (ZTA), microservicios, mTLS, SPIFFE/SPIRE, OAuth 2.0, OpenID Connect, Open Policy Agent, service mesh, microsegmentación, autenticación continua, DevSecOps.

Diseño de software bajo la premisa de red comprometida: autenticación y autorización granular en microservicios

---

Área temática: *Ciberseguridad · Arquitectura de Software · DevSecOps*

Institución: *Universidad Nacional del Nordeste — Ingeniería del SW 2*

Palabras clave: *Zero Trust Architecture (ZTA), microservicios, mTLS, SPIFFE/SPIRE, OAuth 2.0, OIDC, OPA, service mesh, microsegmentación, autenticación continua*

---

1. Introducción

La proliferación de arquitecturas de microservicios, la computación en la nube y los modelos de trabajo remoto han redefinido profundamente el perímetro de seguridad de los sistemas de información. El paradigma tradicional —conocido coloquialmente como el modelo de *castillo y foso* (*castle-and-moat*)— asumía que todo tráfico interno a la red corporativa era inherentemente confiable. Esta suposición resultó ser una falla estructural: cuando un atacante logra traspasar el perímetro (por ejemplo, mediante credenciales robadas o una vulnerabilidad en una VPN), obtiene acceso prácticamente irrestricto a recursos internos y puede moverse lateralmente sin obstáculos significativos (Northern Technologies Group, 2025).

La Arquitectura de Confianza Cero (*Zero Trust Architecture*, ZTA) surge como respuesta a esta limitación fundamental. Su premisa central es que ninguna entidad —usuario, dispositivo, servicio o segmento de red— debe recibir confianza implícita por su ubicación. En cambio, cada solicitud de acceso debe ser explícitamente verificada, autorizada con el mínimo privilegio necesario y supervisada de forma continua.

En el contexto del desarrollo de aplicaciones modernas, particularmente aquellas construidas sobre microservicios, ZTA impone un conjunto de controles técnicos que transforman la manera en que se diseña, despliega y opera el software. Este documento analiza sistemáticamente dicho paradigma: sus fundamentos teóricos, su formalización normativa, los patrones de implementación técnica y los desafíos que presenta su adopción en entornos distribuidos reales.

2. Contexto Histórico y Evolución del Paradigma

La genealogía de Zero Trust puede trazarse a través de varias etapas intelectuales y técnicas:

1994 — Primeras formulaciones conceptuales. Stephen Paul Marsh introduce la noción formal de confianza computacional en su tesis doctoral, estableciendo bases filosóficas que anticipan el modelo (Marsh, 1994, citado en Nasiruzzaman et al., 2024).

2003 — Jericho Forum y la des-perimeterización. El consorcio Jericho Forum publica los primeros lineamientos sobre *de-perimeterización*, argumentando que la frontera entre una organización y el mundo exterior se estaba volviendo irrelevante. El foro se disuelve en 2013, declarando que la des-perimeterización ya es un hecho consumado (Zscaler, 2025).

2009–2010 — Kindervag y el término "Zero Trust". John Kindervag, analista de Forrester Research, acuña formalmente el término *Zero Trust* en su informe seminal *"No More Chewy Centers: Introducing The Zero Trust Model of Information Security"* (2010). Su propuesta fue radical: la confianza debía eliminarse como categoría de diseño de seguridad. El argumento

central es que los profesionales de seguridad confían demasiado y verifican muy poco (1Password, 2024). Kindervag propuso tres principios fundacionales: (a) asegurar el acceso a todos los recursos independientemente de su ubicación, (b) adoptar el mínimo privilegio y controlar estrictamente el acceso, y (c) inspeccionar y registrar todo el tráfico.

2009–2014 — Google BeyondCorp. En 2009, en respuesta al ciberataque conocido como *Operación Aurora*, Google inicia internamente el proyecto BeyondCorp con el objetivo de permitir el trabajo remoto sin VPN, basando el acceso en la identidad y el contexto del dispositivo, no en la ubicación de red. La publicación del modelo en 2014 otorgó al concepto de Zero Trust una validación de escala industrial (Zscaler, 2025).

2017–2019 — Expansión y marcos complementarios. Gartner introduce CARTA (*Continuous Adaptive Risk and Trust Assessment*, 2017) y SASE (*Secure Access Service Edge*, 2019), marcos que comparten la filosofía de verificación adaptativa y continua con ZTA.

2020 — NIST SP 800-207. El Instituto Nacional de Estándares y Tecnología de los Estados Unidos publica la publicación especial 800-207, convirtiéndola en el marco de referencia normativo más completo para ZTA a nivel global. En 2023, NIST publica SP 800-207A, extendiendo la guía específicamente hacia aplicaciones cloud-nativas en entornos multi-nube.

2021 — Mandato ejecutivo (EO 14028). La Orden Ejecutiva 14028 del gobierno de los Estados Unidos exige que todas las agencias federales adopten los principios de confianza cero para 2024, elevando ZTA de recomendación técnica a política de Estado.

2025 — Normalización ETSI. En septiembre de 2025, el Comité Técnico CYBER del Instituto Europeo de Normas de Telecomunicaciones (ETSI) publica la Especificación Técnica TS 104 102, que formaliza la metodología ZT-Kipling como estándar europeo (Wikipedia, 2025).

---

3. Fundamentos Conceptuales de Zero Trust

3.1. Principios Axiales

El modelo de Zero Trust descansa sobre un conjunto de principios que trascienden las tecnologías específicas de implementación. El NIST SP 800-207 establece los siguientes tenets fundamentales (NIST, 2020; Palo Alto Networks, 2026):

Nunca confiar, siempre verificar (*Never Trust, Always Verify*). Toda entidad —usuario, servicio, dispositivo— es tratada como potencialmente hostil hasta que demuestre su identidad y su autorización para el recurso solicitado. La ubicación de red (interna o externa) no otorga confianza implícita alguna.

Mínimo privilegio (*Least Privilege*). Cada identidad —humana o no humana— recibe únicamente los permisos estrictamente necesarios para cumplir su función, durante el tiempo mínimo requerido. Los privilegios son acotados a la sesión y se elevan solo cuando existe justificación explícita.

Asumir la brecha (*Assume Breach*). Los sistemas deben ser diseñados bajo la suposición de que el atacante ya ha obtenido acceso a algún componente. El objetivo no es prevenir toda intrusión, sino minimizar el radio de daño (*blast radius*) y detectar anomalías lo antes posible.

Verificación explícita y continua. La autenticación no es un evento único al inicio de la sesión, sino un proceso continuo que evalúa contexto, comportamiento e integridad de la identidad a lo largo del tiempo.

Acceso por sesión con política dinámica. El acceso se otorga por solicitud individual, no por pertenencia a una red. La política de acceso considera factores contextuales como el nivel de aseguramiento de identidad, la postura de seguridad del dispositivo y el comportamiento reciente de la entidad.

Tratar todos los recursos como externos. Independientemente de si un servicio se encuentra dentro o fuera de la red corporativa, debe ser tratado con el mismo nivel de desconfianza y controles de seguridad.

Gobernar todas las identidades consistentemente. Las políticas de acceso aplican de forma uniforme sobre identidades humanas y no humanas (servicios, pipelines CI/CD, agentes de software).

3.2. El Modelo NIST SP 800-207 y SP 800-207A

La publicación NIST SP 800-207 introduce un cambio paradigmático fundamental: el desplazamiento del foco de seguridad desde los parámetros de red —como direcciones IP, subredes y perímetros— hacia las identidades. El documento define ZTA como la arquitectura de ciberseguridad que implementa los principios de confianza cero mediante una infraestructura de red y políticas operativas específicas (NIST, 2020; CSRC, 2023).

La publicación SP 800-207A, coautoría de Zack Butcher (Tetrate) y Ramaswamy Chandramouli (NIST), extiende el modelo hacia aplicaciones cloud-nativas en entornos multi-nube. Introduce la necesidad de políticas de autenticación y autorización basadas en identidades de aplicación y servicio —además de las identidades de red y usuario— lo que requiere una plataforma compuesta por gateways de API, proxies sidecar e infraestructuras de identidad de carga de trabajo como SPIFFE (NIST SP 800-207A, 2023).

3.3. Componentes Lógicos: PEP, PDP y PA

El modelo NIST articula la ZTA alrededor de tres componentes lógicos principales:

Policy Enforcement Point (PEP) — Punto de Ejecución de Políticas. Es el guardián que intercepta cada solicitud de acceso y la evalúa contra las políticas vigentes. En arquitecturas de microservicios, el proxy Envoy (parte de Istio y otros service meshes) actúa como PEP universal al interceptar todo el tráfico entrante y saliente de cada servicio (Tetrate, 2023).

Policy Decision Point (PDP) — Punto de Decisión de Políticas. Es el componente que evalúa la solicitud de acceso según las políticas definidas y emite una decisión de permitir o denegar. Un ejemplo típico es Open Policy Agent (OPA) actuando como motor de decisión externo.

Policy Administrator (PA) — Administrador de Políticas. Gestiona el ciclo de vida de las políticas, incluyendo su creación, actualización, distribución y revocación. Se encarga de comunicar las decisiones del PDP al PEP.

Esta separación de responsabilidades es fundamental para implementar Zero Trust en microservicios: permite externalizar la lógica de autorización fuera del código de aplicación, garantizando consistencia, auditabilidad y facilidad de actualización de políticas.

---

4. El Problema de la Seguridad Perimetral en Arquitecturas Distribuidas

El modelo de seguridad basado en perímetro fue diseñado para una era en que las aplicaciones residían en servidores físicos en centros de datos propietarios y los usuarios trabajaban desde redes corporativas. En ese contexto, la frontera entre lo interno (confiable) y lo externo (hostil) era clara y estable.

Las arquitecturas de microservicios en entornos cloud-nativos disuelven completamente esa frontera. Varios factores estructurales explican por qué el modelo perimetral es insuficiente en este contexto:

Expansión de la superficie de ataque. Cada microservicio expone un conjunto de endpoints de API. En una aplicación empresarial típica pueden coexistir decenas o cientos de servicios comunicándose entre sí, cada uno como potencial vector de ataque (Cerbos, 2025). Un informe de la industria indica que las brechas relacionadas con APIs aumentaron un 321% en 2024, con la autorización inadecuada como causa principal en el 68% de los incidentes (Markaicode, 2025).

Comunicación este-oeste sin inspección. En arquitecturas de microservicios, la mayor parte del tráfico es inter-servicio (tráfico *este-oeste*), no el tráfico usuario-aplicación (*norte-sur*). Los modelos perimetrales protegen exclusivamente la frontera exterior, dejando el tráfico interno sin cifrar, sin autenticar y sin autorizar. Un análisis en entornos de producción encontró casos donde tráfico de pago y credenciales de usuarios fluían en texto plano entre servicios dentro del mismo clúster (DZone, 2025).

Identidades efímeras y dinámicas. En entornos Kubernetes, los pods se crean y destruyen continuamente. Las identidades de red —como las direcciones IP— son volátiles y no pueden usarse como base confiable para el control de acceso. Una dirección IP que perteneció a un servicio legítimo puede reasignarse a otro en cuestión de segundos.

Movimiento lateral como amenaza central. Cuando un atacante compromete un servicio dentro del clúster, el modelo perimetral no ofrece protección adicional. El atacante puede moverse libremente hacia otros servicios, bases de datos y sistemas críticos. Reportes recientes indican que el 89% de las organizaciones experimentaron un incidente de seguridad en Kubernetes en el último año, frecuentemente mediante movimiento lateral tras el compromiso inicial de un pod (Groundcover, 2026).

Amenazas internas (*insider threats*). El modelo perimetral tampoco protege contra actores maliciosos internos o credenciales comprometidas de usuarios legítimos que, una vez autenticados, tienen acceso amplio y sin restricciones.

La conclusión es directa: en arquitecturas distribuidas cloud-nativas, solo un enfoque de Zero Trust puede proveer protección suficiente. Como señala Hofmann (JAX London, 2025), si se lleva el problema de seguridad de los microservicios a su conclusión lógica, la confianza cero es el único paradigma que ofrece cobertura adecuada.

---

5. Zero Trust Aplicado al Desarrollo de Microservicios

5.1. Identidad como Nuevo Perímetro

El desplazamiento desde el perímetro de red hacia la identidad como base del control de acceso es el cambio más profundo que introduce ZTA. En microservicios, esta identidad tiene dos dimensiones complementarias:

La **identidad humana** corresponde a los usuarios finales o administradores que interactúan con el sistema. Se gestiona mediante protocolos estándar como OAuth 2.0 y OpenID Connect (OIDC).

La **identidad de carga de trabajo** (*workload identity*) corresponde a los propios microservicios, funciones serverless, jobs de CI/CD y agentes de software que se comunican entre sí. Esta dimensión es frecuentemente subestimada pero resulta crítica en arquitecturas distribuidas.

5.2. Identidad de Carga de Trabajo: SPIFFE y SPIRE

El estándar SPIFFE (*Secure Production Identity Framework For Everyone*) define una identidad verificable para software en entornos dinámicos y heterogéneos. SPIRE es su implementación de referencia, gestionada por la Cloud Native Computing Foundation (CNCF).

SPIFFE/SPIRE resuelve un problema crítico en entornos Zero Trust: cómo otorgar identidad a los servicios sin distribuir secretos estáticos (contraseñas, claves API hardcodeadas). El mecanismo es el siguiente:

Cada carga de trabajo, al iniciarse, demuestra su naturaleza e identidad mediante un proceso de atestación (*attestation*). SPIRE verifica atributos como el nombre del pod en Kubernetes, la imagen de contenedor o la función de nube, y emite un documento de identidad de corta duración denominado SVID (*SPIFFE Verifiable Identity Document*). Los SVIDs son certificados X.509 con un tiempo de vida típico de una hora, que se renuevan automáticamente, eliminando la necesidad de credenciales estáticas (Sameerbhanushali, 2026).

Estos certificados de corta vida son fundamentales para Zero Trust porque: (a) reducen drásticamente la ventana de explotación si una credencial es comprometida, (b) generan un rastro auditable de cada identidad emitida, y (c) se integran nativamente con mTLS y otros mecanismos de seguridad.

Como destacan los investigadores del paper *"Zero Trust Security Model Implementation in Microservices Architectures Using Identity Federation"* (arXiv, 2511.04925, 2025), SPIFFE/SPIRE permite construir identidades de extremo a extremo que facilitan conexiones mTLS seguras e identidades de carga de trabajo auditables en entornos multi-dominio.

5.3. Autenticación Mutua: mTLS

El *Mutual TLS* (mTLS) es el mecanismo fundamental para la autenticación entre servicios en una arquitectura Zero Trust. A diferencia del TLS estándar —donde solo el servidor se autentica ante el cliente— en mTLS ambas partes deben presentar certificados válidos para establecer la comunicación.

En el contexto de microservicios, mTLS garantiza que:

- Cada solicitud entre servicios está cifrada en tránsito, incluyendo el tráfico este-oeste dentro del clúster.
- Cada servicio puede verificar criptográficamente la identidad del servicio que lo invoca.
- Los certificados son gestionados automáticamente por el service mesh, sin intervención manual.

El overhead de rendimiento introducido por mTLS es típicamente inferior a 2 milisegundos cuando se utilizan proxies modernos como Envoy con conexiones persistentes (Developers.dev, 2026). Es importante notar que mTLS autentica servicios, no usuarios; la autorización basada en usuarios requiere mecanismos adicionales como JWT y OAuth2 (DEV Community, 2026).

5.4. Autenticación de Usuarios: OAuth 2.0 y OpenID Connect

Para la dimensión de identidad humana, los estándares OAuth 2.0 y OIDC constituyen la base del control de acceso en microservicios.

OAuth 2.0 es un framework de autorización que permite a los clientes obtener acceso limitado a recursos en nombre de un usuario, sin exponer sus credenciales. En arquitecturas de microservicios, los tokens de acceso (generalmente JWT) son emitidos por un servidor de autorización central (*Identity Provider*) y presentados en cada solicitud a los servicios.

OpenID Connect (OIDC) extiende OAuth 2.0 añadiendo una capa de identidad, definiendo el mecanismo de SSO (*Single Sign-On*) y de acceso basado en *claims* a través de API gateways o patrones Backend-for-Frontend (BFF). OIDC Federation 1.0 extiende este modelo permitiendo la federación automatizada y segura de múltiples dominios de confianza mediante declaraciones de entidad firmadas (arXiv, 2511.04925, 2025).

Los tokens JWT emitidos por el proveedor de identidad son propagados a través de los microservicios, permitiendo que cada servicio valide independientemente la identidad y permisos del solicitante sin consultar centralmente al servidor de autorización en cada solicitud.

La arquitectura moderna de producción combina estas tecnologías de la siguiente manera: API Gateway o Ingress → OAuth2/OIDC → JWT → mTLS interno con SPIFFE/SPIRE (DEV Community, 2026).

5.5. Autorización Granular: Open Policy Agent (OPA)

Una vez establecida la autenticación, el siguiente desafío es la autorización granular. OPA (*Open Policy Agent*) es un motor de políticas de código abierto, de propósito general, que desacopla la toma de decisiones de autorización del código de aplicación.

OPA permite definir políticas como código usando el lenguaje declarativo Rego, y evaluarlas de forma centralizada o distribuida. En el contexto de microservicios con Istio, los proxies Envoy utilizan el filtro de *External Authorization* para delegar cada solicitud al motor OPA antes de permitir que alcance el servicio destino (AWS, 2025; OPA Docs, 2025).

Las ventajas de este enfoque son múltiples:

- Las políticas de autorización son externas al código del servicio, garantizando consistencia entre todos los microservicios.
- Las políticas pueden actualizarse sin redesplegar los servicios.
- Las políticas son versionables, testeables y auditables como código fuente.
- OPA puede integrarse en sidecars para evaluar políticas localmente, evitando latencia de red adicional (Permit.io, 2025).

Como señalan los investigadores de Tetrate y NIST, OPA es apropiado no solo para los cinco controles de política del marco ZTA de NIST, sino también para aplicar políticas de negocio específicas más allá del framework base (Tetrate, 2023).

Un sistema de autorización contemporáneo en microservicios debe responder preguntas compuestas del tipo: *"¿Está el Servicio A autorizado a acceder al Recurso X en nombre del Usuario Y?"* Esta complejidad requiere un sistema de autorización desacoplado como OPA, en lugar de lógica de autorización embebida en cada servicio (Cerbos, 2025).

5.6. Service Mesh: Istio y Envoy como Plano de Control de Seguridad

Un service mesh es una capa de infraestructura dedicada que gestiona la comunicación entre microservicios. En el paradigma Zero Trust, el service mesh actúa como el mecanismo de implementación de los controles de seguridad, desplazando esa responsabilidad fuera del código de aplicación.

Istio es el service mesh más ampliamente adoptado en entornos Kubernetes. Su proxy de datos —Envoy— se despliega como *sidecar* junto a cada microservicio, interceptando todo el tráfico entrante y saliente. Este patrón convierte a Envoy en un PEP universal que puede:

- Implementar mTLS automáticamente entre todos los servicios del mesh.
- Validar SVIDs de SPIFFE como identidades de servicio.
- Delegar decisiones de autorización a OPA mediante el filtro de External Authorization.
- Generar telemetría detallada de todo el tráfico para observabilidad y auditoría.

Como describen los investigadores en el estudio sobre ZTA en Java Microservices (Kesarpu, 2025), la combinación de Istio, OAuth 2.0 y Spring Security constituye una implementación completa de ZTA para microservicios en el ecosistema JVM.

El service mesh permite, en palabras de Tetrate (2023), mover las preocupaciones de seguridad fuera de la aplicación y hacia el mesh: el cifrado en tránsito, la autenticación de identidad de servicio y la autorización entre servicios son gestionados por la infraestructura, no por cada desarrollador individualmente.

---

6. Microsegmentación y Control de Movimiento Lateral

La microsegmentación es una técnica de seguridad de red que divide el entorno de aplicación en zonas aisladas con controles de acceso independientes. En el contexto de microservicios, la microsegmentación opera al nivel de servicio o pod, no al nivel de subred tradicional.

El objetivo principal de la microsegmentación en Zero Trust es contener el movimiento lateral: si un servicio es comprometido, el atacante no debe poder alcanzar otros servicios sin superar controles de autenticación y autorización explícitos. Es la analogía digital de las puertas cortafuegos en un edificio: si un área es comprometida, la amenaza no puede propagarse libremente (Techspective, 2025).

En Kubernetes, la microsegmentación se implementa mediante Network Policies que definen qué pods pueden comunicarse entre sí. La configuración predeterminada —donde todos los pods de un namespace pueden comunicarse entre sí— es insegura desde una perspectiva Zero Trust. La postura correcta es *denegar todo por defecto* y permitir explícitamente solo las comunicaciones necesarias (AccuKnox, 2025; AccuKnox, 2026).

Como señala Kindervag en sus principios originales, la filosofía correcta es una lista blanca explícita de servicios autorizados con denegación por defecto de todo lo demás, lo que reduce drásticamente los vectores de ataque disponibles (AccuKnox, 2026).

La microsegmentación opera en combinación con mTLS: incluso si la comunicación de red entre dos pods está permitida por Network Policy, mTLS garantiza que la autenticación mutua ocurra antes de que cualquier dato sea intercambiado.

---

7. Gestión de Secretos y Credenciales Efímeras

Uno de los problemas de seguridad más comunes en arquitecturas de microservicios es la gestión inadecuada de secretos: contraseñas de bases de datos, claves API, certificados TLS y otros materiales criptográficos que frecuentemente terminan hardcodeados en código fuente, imágenes de contenedor o variables de entorno sin protección adecuada.

Zero Trust aborda este problema mediante el principio de credenciales efímeras (*short-lived credentials*): en lugar de distribuir secretos de larga duración, los servicios obtienen credenciales de corta vida a través de mecanismos autenticados.

Las plataformas de gestión de secretos más adoptadas en este paradigma incluyen:

HashiCorp Vault es la solución más difundida. Permite a los servicios autenticarse ante Vault utilizando su SVID de SPIFFE como credencial de entrada, y obtener a cambio secretos dinámicos —como credenciales de base de datos generadas en tiempo real con TTL corto. El resultado es un flujo donde no existen secretos estáticos que distribuir ni rotar manualmente: Vault genera credenciales dinámicas por solicitud y las revoca automáticamente al expirar (Sameerbhanushali, 2026).

AWS Secrets Manager, GCP Secret Manager, Azure Key Vault son equivalentes gestionados por los principales proveedores de nube, con integración nativa con sus respectivos modelos de identidad de servicio.

En entornos Kubernetes con Zero Trust, el flujo correcto de gestión de secretos es: (1) el pod se inicia y recibe su SVID de SPIRE automáticamente; (2) usa ese SVID para autenticarse ante Vault; (3) Vault emite credenciales dinámicas de corta vida; (4) el pod accede al recurso (base de datos, API externa) con esas credenciales; (5) las credenciales expiran y se renuevan automáticamente. En ningún punto existen credenciales estáticas almacenadas en el sistema.

Los pipelines CI/CD también deben operar bajo principios Zero Trust, utilizando identidades de corta vida para sus agentes. GitLab's 2026 Global DevSecOps Report señala que la proliferación de identidades de máquina en pipelines —tokens de CI/CD, conexiones de servicio, tokens de despliegue— con privilegios excesivos y sin seguimiento adecuado es un riesgo activo en operaciones cotidianas (Cloudaware, 2026).

---

8. Integración con DevSecOps y la Cadena de Suministro de Software

Zero Trust no es únicamente un paradigma de infraestructura en tiempo de ejecución (*runtime*); debe integrarse en el ciclo de vida completo del desarrollo de software, desde la escritura de código hasta el despliegue en producción.

Este enfoque —frecuentemente denominado *DevSecOps* con principios Zero Trust— implica los siguientes controles en cada etapa de la cadena de entrega:

Fase de desarrollo: Escaneo estático de código (SAST) para detectar secretos hardcodeados, vulnerabilidades conocidas y mal uso de APIs de seguridad. Commits firmados criptográficamente para garantizar la integridad del código fuente.

Fase de construcción (CI): Escaneo de imágenes de contenedor para detectar vulnerabilidades en dependencias. Firma de imágenes con herramientas como Cosign para garantizar que solo imágenes verificadas pueden desplegarse en producción. Autenticación de los agentes de CI con credenciales de corta vida, no con tokens estáticos.

Fase de despliegue (CD): Implementación de admission controllers en Kubernetes que verifiquen la firma de imágenes y la conformidad con políticas de seguridad antes de permitir que un pod se inicie. Restricción de privilegios de despliegue al mínimo necesario.

Fase de runtime: Enforcement de Network Policies, mTLS, validación de JWT y políticas OPA en cada solicitud. Rotación automática de secretos y certificados.

El enfoque DevSecOps + Zero Trust puede resumirse en: cada transferencia en el flujo de trabajo debe probar identidad, delimitar permisos y llevar evidencia hacia adelante (Cloudaware, 2026). La seguridad no es una fase terminal, sino un atributo transversal a todo el ciclo de vida del software.

---

9. Monitoreo Continuo, Observabilidad y Telemetría

El principio de *verificación continua* de Zero Trust requiere que el sistema no solo autentique y autorice en el momento del acceso, sino que monitoree de forma permanente el comportamiento de todas las entidades para detectar anomalías o comportamientos sospechosos.

En una arquitectura ZTA madura, el componente de observabilidad incluye:

Telemetría del service mesh. El proxy Envoy en Istio genera métricas, logs distribuidos y trazas de cada solicitud inter-servicio. Esta telemetría permite mapear cada interacción entre servicios —condición previa fundamental: no se puede asegurar lo que no se puede ver (Developers.dev, 2026).

User and Entity Behavior Analytics (UEBA). Plataformas de análisis de comportamiento que establecen líneas base de acceso normal para cada entidad y alertan ante desviaciones estadísticas significativas. Un servicio que normalmente accede solo a la base de pagos y de repente consulta la base de usuarios representa una anomalía que debe investigarse.

SIEM (Security Information and Event Management). Centralización y correlación de logs de múltiples fuentes (identity provider, API gateway, service mesh, Kubernetes audit logs) para detectar patrones de ataque que no serían visibles en ninguna fuente individual.

Telemetría de identidad. Los proveedores de identidad (como Keycloak, Okta o Azure AD) generan logs de autenticación que, correlacionados con la telemetría de red, permiten detectar uso anómalo de credenciales (logins desde ubicaciones inusuales, rotación anormal de tokens, escalaciones de privilegios no esperadas).

La telemetría de SPIRE —que registra la atestación de cada workload y cada SVID emitido— también es fundamental para demostrar ante auditorías de cumplimiento que toda comunicación de servicio fue autenticada y que no existen credenciales de larga duración en uso (Sameerbhanushali, 2026).

---

10. Desafíos e Implicancias Operativas

La adopción de Zero Trust en arquitecturas de microservicios enfrenta desafíos técnicos, organizativos y económicos que deben evaluarse cuidadosamente.

Complejidad de implementación

ZTA requiere la integración coherente de múltiples tecnologías: SPIFFE/SPIRE para identidad de workloads, un IdP para identidades humanas, un service mesh para enforcement de mTLS, OPA para políticas de autorización, una plataforma de gestión de secretos y herramientas de observabilidad. La orquestación de estos componentes representa una complejidad operativa significativa.

Una revisión sistemática de 74 artículos publicados entre 2016 y 2025 (PMC, 2025) identifica que autenticación, autorización y control de acceso son los componentes de ZTA más consistentemente implementados, mientras que la auditoría, la orquestación y la percepción del entorno permanecen poco desarrollados en la mayoría de las implementaciones.

Overhead de rendimiento y latencia

La introducción de mTLS, la validación de tokens JWT en cada solicitud y las consultas al motor OPA añaden latencia a cada transacción. Sin embargo, mediciones en entornos de producción con Envoy demuestran que el overhead de mTLS es típicamente inferior a 2ms con conexiones persistentes (Developers.dev, 2026). El impacto real depende de la escala del sistema y del volumen de solicitudes inter-servicio.

Costos de migración y operativos

La transición desde una arquitectura perimetral hacia ZTA implica costos de hardware, software, licencias y —frecuentemente— contratación de personal especializado. Investigaciones recientes (Wiley, 2025) señalan que las organizaciones frecuentemente subestiman los costos de mantenimiento continuo, incluyendo la actualización de políticas, las medidas de visibilidad y los controles de acceso.

Configuración errónea como vector de riesgo

Paradójicamente, una implementación incorrecta de ZTA puede introducir nuevos riesgos. Políticas mal configuradas pueden bloquear comunicaciones legítimas en producción o, en el caso opuesto, dejar brechas no cubiertas. La filosofía de *deny all by default* requiere un mapeo exhaustivo previo de todas las comunicaciones legítimas entre servicios.

Brechas en auditoría y cumplimiento

La misma revisión sistemática citada anteriormente señala que muchas implementaciones descuidan mecanismos robustos de auditoría, a pesar de que la trazabilidad completa es un requisito fundamental de marcos regulatorios como GDPR, HIPAA y SOC 2 (PMC, 2025).

Fricción organizacional

ZTA puede percibirse como un obstáculo a la velocidad de desarrollo. La clave para superar esta resistencia es demostrar que la seguridad granular es un habilitador, no un freno: cuando las políticas se gestionan como código (policy-as-code) integrado en el pipeline CI/CD, la fricción operativa se minimiza significativamente.

11. Casos de Referencia y Adopción Industrial

Google BeyondCorp. El caso más documentado de implementación Zero Trust a escala. Google eliminó la dependencia de la VPN corporativa y basó todo el acceso —tanto interno como externo— en la identidad verificada del usuario y la postura del dispositivo. Este modelo permitió a todos los empleados trabajar de forma segura desde cualquier red, sin distinción entre interna y externa.

Microsoft Zero Trust. Microsoft implementó ZTA internamente y lo formalizó en su *Zero Trust Deployment Center*, que incluye guías detalladas para identidades, dispositivos, aplicaciones, datos, infraestructura y redes. La documentación de Sunkara (2025) compara las implementaciones prácticas de Google y Microsoft como referencias industriales.

Sector financiero y DevSecOps cloud-native. Estudios publicados en ScienceDirect (2025) analizan cómo el sector financiero adopta ZTA en entornos DevOps avanzados con microservicios, combinando CI/CD continuo con verificación de seguridad en cada etapa del pipeline para cumplir con regulaciones del sector.

Amazon Web Services (AWS) y Kubernetes. El blog de código abierto de AWS documenta implementaciones concretas de Zero Trust en Amazon EKS (Elastic Kubernetes Service) usando Istio con OPA como servicio de autorización externo, como referencia de implementación reproducible (AWS, 2025).

---

12. Tendencias Emergentes: IA, Multi-cloud y Agentes de Software

El campo de ZTA continúa evolucionando aceleradamente. Varias tendencias emergentes merecen atención en el contexto académico:

Agentes de IA como nuevas identidades de workload. La proliferación de agentes de software basados en modelos de lenguaje de gran escala introduce un nuevo tipo de entidad que requiere identidad verificable y autorización granular. Los sistemas de microservicios modernos incluyen cada vez más agentes de IA que invocan APIs, acceden a bases de datos y coordinan acciones en nombre de usuarios (Cerbos, 2025). El marco SPIFFE/SPIRE y los principios de Zero Trust son directamente aplicables a estas nuevas entidades.

IA para detección adaptativa de políticas. Investigaciones recientes proponen integrar detección de anomalías basada en IA y actualización automática de políticas como respuesta predictiva a nuevas tendencias de amenaza. El paper de arXiv (2511.04925, 2025) identifica la detección de brechas potenciada por IA como una extensión natural de las implementaciones ZTA actuales.

Multi-cloud y federación de confianza. En entornos multi-nube, los workloads se distribuyen entre AWS, Azure, GCP y entornos on-premises. La federación de SPIFFE permite extender la identidad verificable entre dominios de confianza distintos, habilitando mTLS y autorización coherente en entornos heterogéneos. IETF WIMSE (*Workload Identity in Multi-System*

Environments) es un estándar emergente que busca formalizar esta interoperabilidad (arXiv, 2504.17759, 2025).

Policy-as-Code maduro. La gestión de políticas de autorización como código fuente —con versionamiento, pruebas automáticas y despliegue en pipelines CI/CD— se consolida como práctica estándar. La integración de Rego (lenguaje de OPA) en herramientas de análisis estático y linters (como Regal) facilita la detección temprana de políticas incorrectas.

Criptografía post-cuántica. Con el avance de la computación cuántica, la resistencia de los algoritmos criptográficos actuales (RSA, ECDSA) utilizados en mTLS y PKI está siendo evaluada. NIST ha iniciado el proceso de estandarización de algoritmos resistentes a quantum, lo que eventualmente requerirá actualizaciones en las infraestructuras de SPIFFE/SPIRE y certificados TLS.

---

13. Conclusiones

La Arquitectura de Confianza Cero representa un cambio paradigmático fundamental en la manera de diseñar, construir y operar software. Su premisa —asumir que la red ya está comprometida y verificar explícitamente cada acceso— no es un ejercicio de paranoia sino una respuesta racional a la realidad de los entornos distribuidos modernos.

En el contexto del desarrollo de microservicios, ZTA se traduce en un conjunto coherente de patrones técnicos: SPIFFE/SPIRE para identidad de workloads, mTLS para autenticación mutua entre servicios, OAuth 2.0 y OIDC para identidades humanas, OPA para autorización granular basada en políticas como código, service meshes (Istio/Envoy) como plano de ejecución universal, y plataformas de gestión de secretos para credenciales efímeras.

Ninguno de estos componentes actúa de forma aislada; la fortaleza de ZTA reside en su carácter sistémico: cada capa de seguridad asume que la anterior puede estar comprometida. Esta defensa en profundidad limita drásticamente el radio de daño de cualquier brecha, convirtiendo incidentes que en el modelo perimetral serían catastróficos en eventos contenidos y reversibles.

Los desafíos de implementación —complejidad operativa, overhead de rendimiento, costos de migración y fricción organizacional— son reales y no deben minimizarse. Sin embargo, el costo de no implementar ZTA se vuelve cada vez más alto: el informe IBM Cost of a Data Breach 2025 estima el costo promedio global de una brecha en 4,44 millones de dólares, y la tendencia es ascendente.

El camino hacia Zero Trust es incremental, no una revolución instantánea. La estrategia recomendada por múltiples investigadores y organizaciones es: (1) mapear todas las comunicaciones existentes entre servicios, (2) establecer identidad verificable para cada workload, (3) implementar mTLS y micro-segmentación, (4) externalizar la autorización hacia un motor de políticas, y (5) establecer monitoreo continuo antes de avanzar hacia policies más restrictivas.

Para los desarrolladores y arquitectos de software del siglo XXI, Zero Trust no es una opción técnica entre varias posibles: en el contexto de sistemas distribuidos, cloud-nativos y con superficies de ataque en constante expansión, es el estándar de diseño que la práctica profesional responsable exige.

---

14. Referencias

Las referencias se presentan en orden alfabético. Se incluyen fuentes académicas, normativas e industriales de referencia.

AccuKnox. (2025). *Best Microsegmentation Tools For Zero Trust Security 2025*. AccuKnox Blog. https://accuknox.com/blog/microsegmentation-tools-zero-trust

AccuKnox. (2026). *How To Implement Zero Trust Architecture (Reduce Security Risks)*. AccuKnox Blog. https://accuknox.com/blog/zero-trust-architechture

Amazon Web Services. (2025). *Achieving Zero Trust Security on Amazon EKS with Istio*. AWS Open Source Blog. https://aws.amazon.com/blogs/opensource/achieving-zero-trust-security-on-amazon-eks-with-istio/

Cerbos. (2025). *Zero-Trust for Microservices, a Practical Blueprint*. Cerbos Blog. https://www.cerbos.dev/blog/zero-trust-for-microservices

Cloudaware. (2026). *Implementing Zero Trust in DevSecOps Workflow in 2026*. Cloudaware Blog. https://cloudaware.com/blog/zero-trust-devsecops/

DEV Community. (2026). *Authorization methods in .NET microservices*. https://dev.to/belochka1-04/authorization-methods-in-net-microservices-1dl0

Developers.dev. (2026). *Zero Trust Microservices: mTLS & Identity Architecture Guide*. https://www.developers.dev/tech-talk/the-architect-s-decision-implementing-zero-trust-security-in-microservices-at-scale.html

DZone. (2025). *Building Bulletproof Infrastructure With Istio and OPA*. https://dzone.com/articles/zero-trust-api-security-istio-opa

Groundcover. (2026). *Zero Trust in Kubernetes: Principles, Architecture & Best Practices*. https://www.groundcover.com/learn/security/zero-trust-kubernetes

Hofmann, M. (2025). *Zero Trust with Microservices - it's easier than you think!* JAX London 2025. https://jaxlondon.com/blog/zero-trust-with-microservices-its-easier-than-you-think/

IBM. (2025). *Cost of a Data Breach Report 2025*. IBM Security.

Kesarpu, S. (2025). *Zero-Trust Architecture in Java Microservices*. International Journal of Networks and Security. https://www.academicpublishers.org/journals/index.php/ijns/article/view/5451

Kindervag, J. (2010). *No More Chewy Centers: Introducing The Zero Trust Model Of Information Security*. Forrester Research.

Markaicode. (2025). *Zero-Trust APIs in 2025: OAuth 3.0 and Open Policy Agent (OPA) Deep Dive*. https://markaicode.com/zero-trust-apis-oauth-opa-guide/

Nasiruzzaman, M., Ali, M., Salam, I., & Hossain, M. H. (2024). *The Evolution of Zero Trust Architecture (ZTA) from Concept to Implementation*. arXiv preprint arXiv:2504.11984. https://arxiv.org/abs/2504.11984

National Institute of Standards and Technology. (2020). *Zero Trust Architecture* (NIST Special Publication 800-207). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-207

National Institute of Standards and Technology. (2023). *A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments* (NIST Special Publication 800-207A). U.S. Department of Commerce. https://csrc.nist.gov/pubs/sp/800/207/a/final

Northern Technologies Group. (2025). *Zero Trust Architecture in 2025: Shifting from Perimeter Security to Never Trust, Always Verify*. https://ntgit.com/zero-trust-architecture-in-2025-shifting-from-perimeter-security-to-never-trust-always-verify/

Open Policy Agent. (2025). *OPA-Envoy Plugin*. OPA Documentation. https://www.openpolicyagent.org/docs/envoy

Open Policy Agent. (2025). *Tutorial: Istio*. OPA Documentation. https://www.openpolicyagent.org/docs/envoy/tutorial-istio

Palo Alto Networks. (2026). *What Is NIST SP 800-207? Zero Trust Architecture Framework*. Cyberpedia. https://www.paloaltonetworks.com/cyberpedia/what-is-nist-sp-800-207

Permit.io. (2025). *Authorization with Open Policy Agent (OPA)*. https://www.permit.io/blog/authorization-with-open-policy-agent-opa

PMC — PubMed Central. (2025). *A Systematic Literature Review on the Implementation and Challenges of Zero Trust Architecture Across Domains*. Sensors 2025, 25, 6118. https://pmc.ncbi.nlm.nih.gov/articles/PMC12526847/

Researchgate — Adanigbo et al. (2024). *Implementing Zero Trust Security in Multi-Cloud Microservices Platforms: A Review and Architectural Framework*. International Journal of Advanced Multidisciplinary Research Studies, 4(6), 2402–2409.

Sameerbhanushali. (2026). *SPIFFE & SPIRE: The Workload Identity Standard Quietly Powering Zero Trust*. Substack. https://sameerbhanushali.substack.com/p/spiffe-and-spire-the-workload-identity

Seraphic Security. (2026). *Adopting Zero Trust in 2025: A Practical Guide*. https://seraphicsecurity.com/learn/zero-trust/adopting-zero-trust-in-2025-a-practical-guide/

ScienceDirect — Elsevier. (2025). *Enhancing cloud-native DevSecOps: A Zero Trust approach for the financial sector*. https://www.sciencedirect.com/science/article/abs/pii/S0920548925000042

Techspective. (2025). *Breach Ready: Rethinking Zero Trust and Lateral Movement Defense*. https://techspective.net/2025/04/22/breach-ready-rethinking-zero-trust-lateral-movement-defense/

Tetrate. (2023). *Understanding Istio and Open Policy Agent (OPA)*. Tetrate Blog. https://tetrate.io/blog/understanding-istio-and-open-policy-agent-opa

Tetrate. (2023). *NIST SP 800-207A Explained: A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Cloud Environments*. https://tetrate.io/blog/nist-sp-800-207a-explained-zero-trust-architecture-model-for-access-control

The Backend Developers. (2026). *Zero-Trust Service-to-Service Auth in 2026: mTLS, SPIFFE, and Identity Boundaries*. https://thebackenddevelopers.substack.com/p/zero-trust-service-to-service-auth

Wikipedia. (2025). *Zero Trust Architecture*. Wikipedia. https://en.wikipedia.org/wiki/Zero*trust*architecture

Wiley Online Library. (2025). *Zero Trust Architecture as a Risk Countermeasure in Small–Medium Enterprises and Advanced Technology Systems*. Risk Analysis. https://doi.org/10.1111/risa.70026

Wiz. (2025). *What Is Zero Trust Architecture? A Complete Guide For Cloud Security*. Wiz Academy. https://www.wiz.io/academy/compliance/zero-trust-architecture

Zscaler. (2025). *A brief(er) history of zero trust: Major milestones in rethinking enterprise security*. CXO Revolutionaries. https://www.zscaler.com/blogs/product-insights/brief-er-history-zero-trust-major-milestones-rethinking-enterprise-security

arXiv. (2025). *Zero Trust Security Model Implementation in Microservices Architectures Using Identity Federation*. arXiv:2511.04925. https://arxiv.org/abs/2511.04925

arXiv. (2025). *Identity Control Plane: The Unifying Layer for Zero Trust Infrastructure*. arXiv:2504.17759. https://arxiv.org/pdf/2504.17759

Documento generado con fines académicos. Última actualización: junio 2026.