#27 TECHARTIKEL

Zero-Trust-Architektur: Moderne Sicherheit für Microservices mit Keycloak, Istio und SPIFFE

06.05.2026

Dominik Breu

Inhalt

1. GRUNDLAGEN DER ZERO-TRUST-ARCHITEKTUR 4
1.1. DAS ENDE DER PERIMETERSICHERHEIT 4
1.2. ZERO-TRUST-PRINZIPIEN 4
2. DIE DREI SÄULEN EINER ZERO TRUST IMPLEMENTATION 5
2.1. KEYCLOAK ALS IDENTITY PROVIDER 5
2.2. ISTIO SERVICE MESH FÜR NETZWERKSICHERHEIT 6
2.3. SPIFFE/SPIRE FÜR WORKLOAD-IDENTITÄTEN 6
3. PRAKTISCHE UMSETZUNG UND TOKEN-FLOWS 7
3.1. ENDNUTZER-AUTHENTIFIZIERUNG 7
3.2. SERVICE-ZU-SERVICE-KOMMUNIKATION 7
3.3. AUTORISIERUNG MIT OPA 8
4. BETRIEB UND KONTINUIERLICHE SICHERHEIT 9
4.1. KEY ROTATION UND MANAGEMENT 9
4.2. MONITORING UND OBSERVABILITY 9
5. PASSKEYS: DIE ZUKUNFT DER NUTZERAUTHENTIFIZIERUNG 10
5.1. TECHNISCHE GRUNDLAGEN 10
5.2. IMPLEMENTIERUNGSSTRATEGIEN 11
6. FAZIT 12

Zero-Trust-Architektur: Moderne Sicherheit für Microservices mit Keycloak, Istio und SPIFFE

Die traditionelle Netzwerksicherheit mit ihrem "Castle & Moat"-Ansatz hat in der modernen Cloud-Ära ausgedient. Zero Trust bietet einen revolutionären Ansatz für die Absicherung verteilter Systeme und Microservices. In diesem Artikel werden die Kernkonzepte, technischen Bausteine und praktischen Implementierungsaspekte einer Zero-Trust-Architektur detailliert erläutert – von der Identity-Management-Ebene mit Keycloak über Service Mesh Security mit Istio bis hin zu modernen Authentifizierungsverfahren wie Passkeys.

1. GRUNDLAGEN DER ZERO-TRUST-ARCHITEKTUR

Um die Notwendigkeit und Bedeutung von Zero Trust zu verstehen, ist es wichtig, zunächst die Grenzen traditioneller Sicherheitsmodelle zu betrachten und zu erkennen, warum moderne Architekturen einen fundamentalen Paradigmenwechsel erfordern.

1.1. DAS ENDE DER PERIMETERSICHERHEIT

Das klassische Sicherheitsmodell “Castle & Moat” basierte auf der Annahme, dass alles innerhalb des Unternehmensnetzwerks vertrauenswürdig sei. Diese Herangehensweise funktionierte in Zeiten monolithischer Anwendungen und fester Bürostandorte. Mit dem Aufkommen von Cloud-Architekturen, Remote Work und Microservices hat sich die Situation jedoch grundlegend verändert. Es gibt nicht mehr ein klares “Innen” und “Außen” – Dienste laufen in verschiedenen Cloud-Umgebungen, Mitarbeiter greifen von überall auf Ressourcen zu, und Supply-Chain-Angriffe werden immer raffinierter.

Die Folge sind neue Bedrohungsszenarien: Angreifer, die einmal im Netzwerk sind, können sich lateral bewegen und von Service zu Service springen. Ein kompromittierter Dienst kann schnell zur Gefahr für die gesamte Infrastruktur werden. Diese Realität macht einen neuen Sicherheitsansatz unumgänglich.

1.2. ZERO-TRUST-PRINZIPIEN

Zero Trust folgt dem fundamentalen Prinzip “Never trust, always verify”. Jede einzelne Anfrage muss authentifiziert und autorisiert werden – unabhängig davon, ob sie von innerhalb oder außerhalb des traditionellen Netzwerkperimeters kommt. Dabei spielen drei Kernaspekte eine zentrale Rolle:

Kontinuierliche Verifizierung: Vertrauen wird niemals implizit gewährt, sondern muss für jede Transaktion explizit etabliert werden. Dies bedeutet, dass sowohl die Identität des Anfragenden als auch dessen Berechtigung für die spezifische Aktion überprüft werden.

Minimale Privilegien: Jeder Nutzer und jeder Service erhält nur die minimal notwendigen Rechte für seine Aufgabe. Diese Rechte sind zeitlich begrenzt und kontextabhängig.

Kontextbasierte Entscheidungen: Sicherheitsentscheidungen berücksichtigen den vollständigen Kontext einer Anfrage – wer fragt an, von wo, mit welchem Gerät, zu welcher Zeit und für welchen Zweck.

Nach dieser Einführung in die Grundprinzipien folgt nun eine detaillierte Betrachtung der technischen Bausteine, die für die praktische Umsetzung einer Zero-Trust-Architektur erforderlich sind.

2. DIE DREI SÄULEN EINER ZERO TRUST IMPLEMENTATION

Die technische Umsetzung von Zero Trust erfordert das Zusammenspiel spezialisierter Komponenten, die jeweils unterschiedliche Aspekte der Sicherheitsarchitektur abdecken. Diese drei Säulen bilden gemeinsam ein robustes Sicherheitsfundament für moderne Microservice-Architekturen.

2.1. KEYCLOAK ALS IDENTITY PROVIDER

Keycloak fungiert als zentraler Identity Provider (IdP) und bildet das Herzstück der Identitätsverwaltung in einer Zero-Trust-Architektur. Als Open-Source-Lösung bietet es eine umfassende Palette an Funktionen für die Verwaltung von Nutzer- und Service-Identitäten.

Die Hauptaufgabe von Keycloak besteht in der Ausstellung und Verwaltung von Sicherheitstokens (Security-Tokens). Es unterstützt moderne Standards wie OAuth 2.0, OpenID Connect (OIDC) und SAML, wodurch eine nahtlose Integration in bestehende Systeme möglich wird. Besonders wichtig für Microservice-Architekturen ist die Fähigkeit, kurzlebige Access Tokens, ID Tokens und Refresh Tokens auszustellen. Diese Tokens haben typischerweise eine Lebensdauer von nur wenigen Minuten, was das Risiko bei Kompromittierung minimiert.

Ein Schlüsselfeature für die Service-zu-Service-Kommunikation ist der “Token Exchange” nach RFC 8693. Dieser Mechanismus ermöglicht es einem Service, ein erhaltenes Nutzer-Token gegen ein Service-spezifisches Token einzutauschen. Dadurch wird eine kontrollierte Delegation von Rechten möglich, ohne dass sensible Nutzer-Tokens durch die gesamte Service-Kette weitergereicht werden müssen.

2.2. ISTIO SERVICE MESH FÜR NETZWERKSICHERHEIT

Istio implementiert die Sicherheit auf Netzwerkebene und fungiert als Policy Enforcement Point für die gesamte Service-Kommunikation. Als Service Mesh nutzt es Sidecar-Proxies (typischerweise Envoy), die neben jedem Service laufen und den gesamten ein- und ausgehenden Netzwerkverkehr kontrollieren.

Die wichtigste Sicherheitsfunktion von Istio ist die automatische Durchsetzung von Mutual TLS (mTLS) zwischen allen Services. Dies bedeutet, dass jede Verbindung zwischen zwei Services verschlüsselt und beidseitig authentifiziert wird – und das vollkommen transparent für die Anwendungen selbst. Die Entwickler müssen keinen Code für die Verschlüsselung schreiben; Istio übernimmt dies automatisch.

Zusätzlich zur Transport-Sicherheit validiert Istio auch JWT-Tokens. Am Ingress Gateway, dem Eintrittspunkt in das Service Mesh, werden eingehende Tokens gegen die JWKS (JSON Web Key Set) des Identity Providers geprüft. Durch RequestAuthentication und AuthorizationPolicy-Ressourcen können feingranulare Zugriffsregeln definiert werden, die auf Token-Claims, Scopes und anderen Attributen basieren.

2.3. SPIFFE/SPIRE FÜR WORKLOAD-IDENTITÄTEN

SPIFFE (Secure Production Identity Framework for Everyone) definiert einen Standard für Workload-Identitäten, während SPIRE die Referenzimplementierung dieses Standards darstellt. Gemeinsam lösen sie ein fundamentales Problem moderner Infrastrukturen: die automatische und sichere Vergabe von Identitäten an Workloads.

Jeder Workload erhält eine eindeutige SPIFFE ID, die ähnlich wie ein Service Principal funktioniert. Diese Identität wird in Form von SVIDs (SPIFFE Verifiable Identity Documents) ausgestellt, entweder als X.509-Zertifikate oder als JWT-Tokens. Der entscheidende Vorteil liegt in der Automatisierung: SPIRE übernimmt die Verteilung, Rotation und Verwaltung dieser Identitätsnachweise vollständig.

Die Integration mit Istio ist nahtlos – die Sidecar-Proxies nutzen die von SPIRE bereitgestellten SVIDs automatisch für die mTLS-Verbindungen. Dies eliminiert das manuelle Zertifikatsmanagement vollständig und macht die Implementierung von Zero Trust auf Netzwerkebene praktikabel.

Mit diesem Verständnis der technischen Bausteine können wir nun die konkreten Kommunikationsabläufe und Token-Flows in einer Zero-Trust-Architektur betrachten.

3. PRAKTISCHE UMSETZUNG UND TOKEN-FLOWS

Die praktische Implementierung von Zero Trust zeigt sich besonders in den konkreten Kommunikationsabläufen zwischen Nutzern, Services und Sicherheitskomponenten. Diese Flows müssen sorgfältig orchestriert werden, um sowohl Sicherheit als auch Performance zu gewährleisten.

3.1. ENDNUTZER-AUTHENTIFIZIERUNG

Der Authentifizierungsprozess für Endnutzer beginnt typischerweise an der Benutzeroberfläche. Nutzer melden sich über einen OIDC-Flow bei Keycloak an, wobei moderne Authentifizierungsmethoden wie Multi-Faktor-Authentifizierung oder Passkeys zum Einsatz kommen können. Nach erfolgreicher Authentifizierung stellt Keycloak ein JWT-basiertes Access Token aus.

Dieses Token ist bewusst kurzlebig – typischerweise mit einer Gültigkeit von maximal fünf Minuten. Es enthält Claims über die Identität des Nutzers, dessen Rollen und Berechtigungen. Wenn die Benutzeroberfläche nun einen API-Call an einen Backend-Service macht, wird dieses Token im Authorization-Header mitgesendet.

Am Eingang des Service Mesh prüft das Istio Ingress Gateway die Gültigkeit des Tokens. Es lädt dazu die öffentlichen Schlüssel von Keycloak über dessen JWKS-Endpoint und verifiziert die Signatur. Nur bei erfolgreicher Validierung wird die Anfrage an den Ziel-Service weitergeleitet.

3.2. SERVICE-ZU-SERVICE-KOMMUNIKATION

Die Service-zu-Service-Kommunikation erfordert besondere Aufmerksamkeit, da hier das Risiko von “Confused Deputy”-Angriffen besteht. Ein Service darf niemals einfach das erhaltene Nutzer-Token an nachgelagerte Services weitergeben. Stattdessen kommt der Token Exchange zum Einsatz.

Wenn beispielsweise ein Orders-Service den Payments-Service aufrufen muss, kontaktiert er zunächst Keycloak und tauscht das ursprüngliche Nutzer-Token gegen ein Service-Token ein. Dieses neue Token hat entscheidende Eigenschaften:

Subject-Wechsel: Das neue Token hat als Subject den ServiceAccount des anfragenden Services (z.B. "orders-service") statt des ursprünglichen Nutzers. Dies macht transparent, welcher Service die Anfrage stellt.

Audience-Scoping: Das Token ist spezifisch für den Ziel-Service ausgestellt (Audience: "payments-service"). Es kann nicht für Anfragen an andere Services missbraucht werden.

Reduzierte Lebensdauer: Service-Tokens haben eine noch kürzere Lebensdauer als Nutzer-Tokens – oft nur zwei Minuten. Dies minimiert das Zeitfenster für potenzielle Angriffe.

3.3. AUTORISIERUNG MIT OPA

Während Keycloak und Istio grundlegende Authentifizierungs- und Autorisierungsfunktionen bieten, erfordern komplexere Szenarien oft feingranulare, kontextabhängige Entscheidungen. Hier kommt der Open Policy Agent (OPA) ins Spiel.

OPA fungiert als zentraler Policy Decision Point und trennt die Autorisierungslogik von der Anwendungslogik. Policies werden in der deklarativen Sprache "Rego" geschrieben, was eine klare und nachvollziehbare Definition von Zugriffsregeln ermöglicht. Ein typisches Szenario könnte so aussehen: Ein Service fragt OPA, ob ein bestimmter Nutzer mit einer bestimmten Rolle zu einer bestimmten Tageszeit auf eine spezifische Ressource zugreifen darf.

Die Integration erfolgt meist über Sidecars oder als Library innerhalb der Services. OPA erhält die relevanten Kontextinformationen – Token-Claims, Request-Details, Umgebungsvariablen – und trifft basierend auf den definierten Policies eine Entscheidung. Diese Architektur ermöglicht es, Autorisierungsregeln zentral zu verwalten und konsistent über alle Services hinweg durchzusetzen.

Nachdem die technischen Flows etabliert sind, ist es entscheidend, den kontinuierlichen Betrieb und die Wartung der Zero Trust Infrastruktur zu gewährleisten.

4. BETRIEB UND KONTINUIERLICHE SICHERHEIT

Zero Trust ist kein einmaliges Projekt, sondern ein kontinuierlicher Prozess. Der sichere Betrieb erfordert durchdachte Prozesse für Wartung, Überwachung und Incident Response.

4.1. KEY ROTATION UND MANAGEMENT

Die regelmäßige Rotation von Schlüsseln und Zertifikaten ist fundamental für die Aufrechterhaltung der Sicherheit. Dies betrifft verschiedene Ebenen der Architektur: die Signing Keys von Keycloak für JWTs, die Root-Zertifikate von SPIFFE/SPIRE für die Workload-Identitäten und die TLS-Zertifikate für externe Endpoints.

Bei der Rotation ist besondere Sorgfalt geboten. Es muss eine Überlappungsphase eingeplant werden, in der sowohl alte als auch neue Schlüssel akzeptiert werden. Dies verhindert Authentifizierungsfehler während des Rollouts. Typischerweise wird der neue Schlüssel zunächst zusätzlich zum alten publiziert, dann wird auf den neuen Schlüssel für neue Tokens umgestellt, und erst nach Ablauf aller mit dem alten Schlüssel signierten Tokens wird dieser entfernt.

Die Automatisierung dieser Prozesse ist kritisch. SPIRE übernimmt beispielsweise die automatische Rotation von SVIDs, während für Keycloak-Schlüssel entsprechende Automation-Pipelines implementiert werden sollten.

4.2. MONITORING UND OBSERVABILITY

Effektives Monitoring ist unerlässlich, um Sicherheitsvorfälle zu erkennen und die Gesundheit der Zero Trust Infrastruktur zu überwachen. Wichtige Metriken umfassen die Anzahl fehlgeschlagener Authentifizierungen, die Token-Exchange-Rate, die Latenz von Policy-Entscheidungen und die Erfolgsrate von mTLS-Verbindungen.

Auf Istio-Ebene sollten 4xx-Fehler nach Policy-Namen aufgeschlüsselt überwacht werden. Ein plötzlicher Anstieg kann auf Konfigurationsfehler oder Angriffsversuche hindeuten. Die Token-Exchange-Logs von Keycloak geben Aufschluss über ungewöhnliche Delegation-Muster, die auf kompromittierte Services hinweisen könnten.

SPIRE liefert wichtige Metriken zur SVID-Ausstellung und -Rotation. Fehler bei der SVID-Erneuerung können zu Ausfällen führen und sollten proaktiv alarmiert werden. Die Integration dieser Metriken in zentrale Monitoring-Lösungen wie Prometheus und Grafana ermöglicht eine ganzheitliche Sicht auf die Sicherheitslage.

Mit der etablierten Zero-Trust-Infrastruktur rückt nun ein weiterer kritischer Aspekt in den Fokus: die sichere Authentifizierung der Endnutzer, die den ersten Vertrauensanker in der gesamten Kette darstellen.

5. PASSKEYS: DIE ZUKUNFT DER NUTZERAUTHENTIFIZIERUNG

Im Kontext von Zero Trust stellt die Nutzerauthentifizierung den ersten und kritischsten Vertrauensanker dar. Passkeys repräsentieren hier einen revolutionären Fortschritt, der traditionelle Passwörter obsolet macht und Phishing-Angriffe strukturell unmöglich macht.

5.1. TECHNISCHE GRUNDLAGEN

Passkeys basieren auf dem FIDO2-Standard und nutzen asymmetrische Kryptografie für die Authentifizierung. Im Gegensatz zu Passwörtern, die zwischen Nutzer und Server geteilt werden, verbleibt bei Passkeys der private Schlüssel sicher auf dem Gerät des Nutzers. Der Server kennt nur den öffentlichen Schlüssel.

Die Implementierung erfordert serverseitig die Unterstützung des WebAuthn-Protokolls. Bei der Registrierung generiert das Nutzergerät ein Schlüsselpaar und sendet nur den öffentlichen Schlüssel an den Server. Bei der Authentifizierung beweist das Gerät den Besitz des privaten Schlüssels durch eine kryptografische Signatur, ohne den Schlüssel selbst preiszugeben.

Ein entscheidender Sicherheitsaspekt ist die verpflichtende User Presence Verification und die empfohlene User Verification. Dies bedeutet, dass der Nutzer physisch anwesend sein und sich meist zusätzlich durch Biometrie oder PIN authentifizieren muss. Diese Kombination macht Passkeys zu einer inhärenten Zwei-Faktor-Authentifizierung.

5.2. IMPLEMENTIERUNGSSTRATEGIEN

Für Entwicklerteams stehen verschiedene Implementierungswege zur Verfügung, die jeweils unterschiedliche Trade-offs zwischen Kontrolle und Implementierungsaufwand bieten.

Eigenentwicklung bietet maximale Flexibilität und verhindert Vendor Lock-in. Teams implementieren die WebAuthn-Spezifikation direkt und haben volle Kontrolle über User Experience und Datenflüsse. Der Nachteil liegt im hohen Entwicklungsaufwand und der Komplexität der korrekten Implementierung aller Sicherheitsaspekte.

SDK-Integration stellt einen Mittelweg dar. Bibliotheken wie SimpleWebAuthn für TypeScript oder java-webauthn-server von Yubico bieten vorgefertigte Bausteine für die WebAuthn-Implementierung. Teams müssen jedoch weiterhin Recovery-Flows und Nutzer-Migration selbst entwickeln.

Passkey-as-a-Service Lösungen wie Hanko oder Corbado bieten die schnellste Time-to-Market. Sie übernehmen die komplette Implementierung inklusive Recovery-Mechanismen und Management-Oberflächen. Der Trade-off liegt in reduzierter Flexibilität und potenziellem Vendor Lock-in.

Kritisch für den Erfolg ist die Planung der Nutzer-Migration und Recovery-Szenarien. Nutzer müssen schrittweise von Passwörtern zu Passkeys migriert werden, und es braucht robuste Prozesse für den Fall verlorener Geräte.

Nach der umfassenden Betrachtung aller Komponenten und Aspekte einer Zero-Trust-Architekturfolgt nun die Zusammenführung der Erkenntnisse und ein Ausblick auf zukünftige Entwicklungen.

6. FAZIT

Die Implementierung einer Zero-Trust-Architekturstellt einen fundamentalen Paradigmenwechsel in der IT-Sicherheit dar. Statt auf veraltete Perimeter-basierte Sicherheitskonzepte zu setzen, etabliert Zero Trust ein System kontinuierlicher Verifizierung und minimaler Privilegien.

Die drei technischen Säulen – Keycloak für Identity Management, Istio für Service Mesh Security und SPIFFE/SPIRE für Workload-Identitäten – bilden gemeinsam ein robustes Sicherheitsfundament. Ihre Integration ermöglicht es, jeden Request zu authentifizieren und zu autorisieren, ohne die Entwicklungsgeschwindigkeit zu beeinträchtigen.

Besonders wichtig ist das Verständnis, dass Zero Trust kein Produkt ist, das man kaufen und installieren kann. Es ist eine Architektur-Philosophie, die kontinuierliche Aufmerksamkeit erfordert. Die Implementierung erfordert sorgfältige Planung der Token-Flows, durchdachte Key-Rotation-Prozesse und umfassendes Monitoring.

Die Evolution der Nutzerauthentifizierung durch Passkeys zeigt, dass Zero Trust nicht bei der Infrastruktur aufhört. Die Absicherung des ersten Vertrauensankers – der Nutzeridentität – ist ebenso kritisch wie die Absicherung der Service-Kommunikation.

Für Entwicklungsteams, die mit modernen Stacks wie Spring Boot und Quarkus arbeiten, bietet die vorgestellte Architektur einen praxiserprobten Weg zur Implementierung höchster Sicherheitsstandards. Die Investition in Zero Trust zahlt sich nicht nur durch erhöhte Sicherheit aus, sondern auch durch verbesserte Compliance, bessere Auditierbarkeit und letztendlich durch das Vertrauen der Nutzer in die Sicherheit ihrer Daten.

Die Zukunft gehört Architekturen, die Sicherheit nicht als nachträgliche Ergänzung, sondern als integralen Bestandteil von Anfang an betrachten. Zero Trust ist nicht nur eine Antwort auf heutige Bedrohungen – es ist die Grundlage für die sichere, vernetzte Zukunft unserer digitalen Infrastrukturen.