

Software-Defined Perimeter (SDP) Architecture Guide v3.0

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

The permanent and official location for the Zero Trust (ZT) Working Group is
https://cloudsecurityalliance.org/research/working-groups/zero-trust

V3 DRAFT

© 2026 Cloud Security Alliance – All Rights Reserved. You may download, store, display on your computer, view, print, and link to the Cloud Security Alliance at https://cloudsecurityalliance.org subject to the following: (a) the draft may be used solely for your personal, informational, noncommercial use; (b) the draft may not be modified or altered in any way; (c) the draft may not be redistributed; and (d) the trademark, copyright or other notices may not be removed. You may quote portions of the draft as permitted by the Fair Use provisions of the United States Copyright Act, provided that you attribute the portions to the Cloud Security Alliance.

Acknowledgements

The **CSA Zero Trust (ZT) Working Group**

The scope of Zero Trust research and guidance necessarily includes cloud and on-premises environments, along with mobile endpoints. It applies to the Internet of Things (IoT) and operational technology (OT). The goals of the Cloud Security Alliance (CSA) Zero Trust (ZT) Working Group are to:

- Collaboratively develop and raise awareness of Zero Trust best practices as a modern necessity and a hybrid or cloud-appropriate approach to improving Information Security (InfoSec)
- Provide thought leadership and educate the industry about the strengths and weaknesses of different ZT approaches so organizations can make informed decisions based on their specific business needs, risks, and priorities
- Take a deliberate product- and vendor-neutral approach to architectures and implementation approaches for a mature Zero Trust implementation
- Take a technically sound position on Zero Trust and make defensible recommendations while remaining product- and vendor-neutral
- The workstream for this document is ZT5 – Pillar: Networks, led by Michael Roza and Philip Griffiths

Lead Authors

Philip Griffiths

Contributors

Michael Roza
Justin Bowen
Benfeng Chen
Dharnisha Narasappa
Rajesh Murthy
Surendra Narang
Tanya Tandon

Reviewers

Salim Aftab
Arun Agunuru
Deepak Antiya
Vamsi Krishna Datari
Jason Garbis
Vijayent Kohli
Pranay Mehta
Victor Monga
Ashley Moran
Oluwaseyi Ojo
Srihari Pakalapati
Karthikeyan Ramdass
Rajath Ravikumar
Patnana Sayesu
Rahul Sharma
Shashank Shelat
Paul Simms
Chris Steffen
Washima Tuleun

CSA Analysts

Erik Johnson

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Table of Contents

Acknowledgements ... 3
- Lead Authors ... 3
- Contributors ... 3
- Reviewers ... 3
- CSA Analysts ... 3

Table of Contents ... 4
Abstract ... 8
- Target Audience ... 9

Key Points ... 9
- Unique Challenges in Securing Enterprise Networks ... 9
- SDP Solves Security Challenges Through Zero Trust Principles ... 9
- Addressing the Needs of IT, OT, and Hybrid Environments ... 10
- A Structured Framework for SDP Implementation ... 10
- SDP as a Pillar of Secure by Design ... 10

Scope ... 11
Introduction ... 11
- What is SDP Now, and Why is it the Most Viable Option for ZTA? ... 11
- Core SDP Concepts ... 12
- SDP Architecture ... 13
  - Operational Considerations and Failure Modes ... 15
- SDP Scenarios ... 15

SDP and Modern Zero Trust ... 17
- Traditional VPN/Firewall Access vs. SDP ... 17
- Where SDP Ends and Microsegmentation Begins ... 18
  - Why This Division Works ... 20
  - Architectural Guardrails ... 21
  - Edge Cases and Overlap ... 21

Implementing SDP, Evolving its Mechanisms and Mapping Use Cases ... 21
- Implementation Framework ... 22
  - Why Map Transaction Flows in an SDP? ... 22
  - From Map to Policy ... 23
  - Automation and AI/ML Accelerators ... 23
- SDP as an Accelerator for Secure by Design ... 24
  - What This Means for CSA's SDP Architecture Guide v3.0 ... 25

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

SDP Architecture Components and Functions....26
Single-Packet Authorization and Its Evolution....26
Core SPA Design Principles....26
Benefits of SPA....27
Practical Limitations of SPA....27
Evolution Beyond Classic SPA....27
Summary of SPA....28
Extending SDP to Modern Use-Case Domains....28
Hybrid and Multi-Cloud Workloads....28
What's Different?....29
Architectural Adaptations....29
Implications for Agentic AI Systems....30
Operational Technology (OT) and Industrial Control (IC)....30
What's Different?....30
Architectural Adaptations....31
Edge and IoT Fleets....31
What's Different?....31
Architectural Adaptations....32
Cross-Cutting Lessons....32
SDP and Access Control....33
SDP and Enterprise Security Architecture....34
Architecture Considerations....34
Architecture Elements: Where an SDP Fits and Where It Doesn't....36
SIEM: From Passive Log Sink to Active Policy Trigger....39
Firewalls and NGFWs: Perimeter Rules Collapse, Inspection Remains....39
IDS/IPS: Fewer Alerts, Sharper Fidelity....39
VPN Retirement Playbook....39
Identity and Posture in the Policy Engine....40
Putting it Together....40
SDP Business Benefits....40
Sample Vendor Implementations....42
OpenZiti/NetFoundry....42
OpenZiti Advantages....43
Network-Infrastructure Hiding Protocol (NHP)....43
NHP Advantages....44
Bring Your Own Security (BYOS)....45
Architecture Overview....45
Core Technology....45
Hardware Security Architecture....45

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Integration with Existing Infrastructure....46
BYOS Advantages....46
AppGate....47
Addressing the Limitations of SPA....47
Alignment to Zero Trust....48
AppGate Advantages....48
Differentiation from SPA and Other Approaches....49
Conclusion and Future Outlook....49
Conclusion....49
Future Outlook....50
Glossary....52
Useful References....52
Hackathon References....53
Appendix 1: SDP Deployment Models....54
Enforcement Placement in SDP v3.0....55
Key Characteristics of SDP v3.0 Models....56
Appendix 2: SPA Details....57
Appendix 3: NHP Details....58
NHP: Zero Trust Implementation at the OSI Session Layer....58
NHP Architecture and Components....59
NHP Core Components....59
Components That Interact with NHP....60
NHP Workflow....60
Cryptographic Algorithms and Frameworks....61
Comparison Between SPA and NHP....62
NHP Integration with SDP....63
Integration Overview....63
Integration Process....64
Benefits of Integrating NHP with SDP....65
Implementation Considerations....66
Applicability Across SDP Deployment Models....66
Open Source Project - OpenNHP....66
Appendix 4: OpenZiti/NetFoundry Details....68
OpenZiti Architecture and Components....69
Software Components....70
Logical Components....71
Connection Security....72
Control and Link Connection Details....73
Controller API Connection Details....74

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Edge Connection Details....74
Service Connection Details....74
Comparison Between SPA and OpenZiti....75
Appendix 5: SDP Market Survey....77
How to Interpret This Information....79
Where Each Platform Fits Today....80
Typical Limitations....81

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Abstract

The increasing sophistication of cyber threats and the limitations of traditional network security models have led to the evolution of the **Software-Defined Perimeter (SDP)**. Originally conceptualized by the Cloud Security Alliance (CSA), SDP offers a dynamic, identity-centric approach to network security by enforcing the principle of least privilege access, unlike traditional perimeter-based security models, which rely on static defenses such as firewalls and VPNs. SDP leverages Zero Trust Architecture (ZTA) and dynamic principles to mitigate risks from lateral movement, unauthorized access, and insider threats.

The significance of SDP lies in its ability to radically reduce attack surfaces by making resources effectively invisible to unauthorized users across the network. By removing all exposed network presence and enforcing identity-bound connectivity, SDP renders most conventional network-based attacks, such as scanning, exploitation, and lateral movement, immediately ineffective. While the precise level of invisibility depends on implementation details (e.g., SPA over UDP vs. SPA over TCP, whether broad CIDR-based rules allow limited probing of reachable ranges), the operational outcome remains consistent: unauthorized users cannot meaningfully discover or interact with protected resources. This is particularly crucial in the era of cloud computing, remote work, and hybrid IT environments, where traditional security models struggle to keep pace with evolving threats.

Over the last few years, the CSA has published the SDP Specification (currently version 2.0) as well as an SDP Architecture Guide (also version 2.0). This update to the architecture guide consolidates and expands upon these foundations, reflecting the evolving landscape of cybersecurity. As Zero Trust has taken center stage, including several CSA papers applying Zero Trust principles to OT, IoT, and other domains, there is growing alignment with broader industry efforts. Notably, CISA's Secure by Design initiative reinforces the same principles championed by SDP by emphasizing proactive security integration from the ground up. As Jen Easterly (former Director of CISA) stated in late 2024, "We don't need more security products, we need more secure products." Our SDP Architecture Guide v3.0 update aims to incorporate and align with these changes.

A few areas we focused on when updating this guide to version 3.0 include:

- **Use Cases:** To include AI/ML, cloud, and on-premises environments (hybrid) and mobile endpoints, with an extension to IoT and OT. It also includes application and product-embedded SDP, such as agentic AI, making our products more secure by design
- **Implementation Techniques:** Versions 1.0 and 2.0 of the SDP Architecture Guide focused exclusively on Single Packet Authorization (SPA) to achieve "authenticate-before-connect." This has been updated to support other techniques for identity-embedded overlay networking
- **Implementation Guidance:** Building upon the work in the CSA Zero Trust Working Group papers, we outline a practical guide for discovering, documenting, and continuously refining the flows that an SDP must protect
- **Real-World Implementation:** We explore real-world SDP implementations, particularly those that focus on making our products more secure by design

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Target Audience

The target audience for this architecture guide includes security architects, network teams, information security professionals, and security leaders who implement SDP and Zero Trust solutions. Solutions providers, vendors, and service providers that support SDP implementations may benefit, as well.

Key Points

Unique Challenges in Securing Enterprise Networks

The security landscape has undergone a radical transformation with the rise of hybrid cloud environments, mobile workforces, the proliferation of IoT and OT devices, globally distributed applications, and Artificial Intelligence/Machine Learning (AI/ML). These dynamic and decentralized architectures dissolve the notion of a fixed network boundary, creating challenges for traditional perimeter-based security models that rely on stable trust zones and static controls. Identities, devices, and workloads now interact across multiple environments and control planes, often with transient connectivity and inconsistent visibility. This results in policy drift, unmanaged assets, and excessive privileges that expand the effective attack surface—for example, AI-driven workloads acting on behalf of users can amplify risk when granted excessive or static permissions. These realities demand a fundamentally different approach to network protection that aligns with the identity-centric, least privilege principles that define SDP architectures.

SDP Solves Security Challenges Through Zero Trust Principles

The SDP, as advanced by the CSA, eliminates flawed perimeter-based assumptions by making resources invisible to unauthorized users. Using dynamic, identity-centric authentication and authorization, SDP enforces the principle of least privilege at the connection point. It eliminates lateral movement and minimizes attack surfaces by creating individualized, encrypted micro-perimeters around each protected resource. As Zero Trust has gained traction, the term "Zero Trust Network Access (ZTNA)" has emerged for user-to-application access (essentially a VPN replacement). By contrast, SDP is the broader architecture: it renders services undiscoverable. It enforces "authenticate-before-connect" with per-resource micro-tunnels, extending beyond remote access to app-to-app and device-to-device connectivity, and minimizing network reachability itself.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Addressing the Needs of IT, OT, and Hybrid Environments

This version of the SDP Architecture Guide builds upon prior approaches that focused solely on SPA. It extends the framework to include emerging identity-embedded overlay networking methods, enabling flexible deployments across complex infrastructures. Whether protecting on-premises servers, cloud workloads, mobile devices, or IoT and OT assets, SDP adapts to the fluid and dynamic nature of the modern enterprise environment. The approach aligns with NIST SP 800-207 ZTA, supports PCI DSS v4.0 controls, maps naturally to IEC 62443 "zones and conduits" for OT, and aligns with NERC CIP (particularly 003-9/10/11, 005, 007, 010, and 013), while also supporting EU NIS2 risk-management measures.

A Structured Framework for SDP Implementation

This document provides practical guidance on discovering, documenting, and continuously refining the flows that an SDP must protect. It adapts step 2 of the CSA Zero Trust framework for implementing a Zero Trust security strategy, Map the Transaction Flows, and applies it directly to SDPs. Rather than a complete end-to-end lifecycle framework, this section focuses on how transaction-flow mapping drives the three core SDP capabilities: least privilege access, dynamic policy enforcement, and application-centric security. It also paves the way for real-time access decisions by binding policy evaluation to live posture signals (e.g., device health, identity risk, workload state), allowing sessions to be established, tightened, or terminated as posture changes occur. To make the process repeatable, we describe it in five iterative passes that can be automated over time.

SDP as a Pillar of Secure by Design

By integrating directly with applications and infrastructure rather than acting as an external add-on, SDP aligns with the global Secure by Design efforts promoted by CISA and other regulators. The approach advocated in this document emphasizes the importance of embedding security at the architectural level from the outset, thereby addressing product-level risks before they can be exploited.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Scope

This document provides a comprehensive architecture and deployment guide for implementing SDP as a modern security framework to address the complex realities of today's enterprise environments. The increasing adoption of hybrid cloud environments, mobile endpoints, IoT, and OT devices, as well as decentralized application delivery, has rendered traditional perimeter-based security models ineffective. These evolving architectures introduce constant change, transient assets, visibility gaps, and inconsistent policy enforcement, all of which increase organizational risk.

The guide expands upon and updates prior CSA SDP specifications by incorporating SPA and newer identity-based overlay networking techniques. It presents practical guidance for implementing SDP to support real-world deployments across diverse environments, including enterprise data centers, cloud infrastructure, mobile workforces, and operational technology ecosystems.

The document outlines prescriptive guidance on core SDP concepts, architecture components, deployment models, transaction flow mapping, policy formulation, and continuous verification. It also addresses integration considerations with existing identity, access management, and security monitoring platforms to ensure cohesive Zero Trust enforcement.

The ultimate goal of this guide is to enable organizations to reduce attack surfaces, eliminate unauthorized lateral movement, and build scalable, resilient network architectures that align with Zero Trust principles and Secure by Design objectives.

Introduction

The SDP, introduced by the CSA, has evolved from a novel concept into a widely adopted architecture for implementing Zero Trust and Secure by Design principles. It underpins key elements of modern network security by reducing the attack surface, enabling identity-driven access, and rendering resources invisible to unauthorized users, endpoints, and other resources.

SDP separates the control and data planes, using proven standards such as mTLS, PKI/x509, and OIDC/SAML to enforce "authenticate-before-connect." Its adoption across commercial, open-source, and critical infrastructure use cases has proven its resilience, scalability, and alignment with Zero Trust initiatives, including CISA's Secure by Design program.

What is SDP Now, and Why is it the Most Viable Option for ZTA?

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Today, the SDP is a mature and proven architectural approach for implementing ZTA across modern, hybrid environments. Unlike legacy models that rely on static network boundaries, SDP enforces identity- and context-aware access at the connection level, ensuring resources remain invisible until authentication is verified. Its flexibility in integrating with cloud-native, on-premises, IoT, and OT systems, combined with support for Secure by Design principles and multiple "authenticate-before-connect" methods, makes SDP the most viable and future-proof strategy for organizations seeking to operationalize Zero Trust effectively and securely.

In practice, however, organizations must balance SDP's fine-grained access model against operational scale and administrative overhead. While per-resource, per-identity authorization is architecturally ideal, implementing and maintaining such granularity across large user populations often requires abstraction, grouping, and policy consolidation. As a result, many deployments adopt tiered or aggregated access models that preserve Zero Trust principles while remaining operationally sustainable. Achieving the appropriate balance between isolation, risk reduction, and operational efficiency remains one of the most critical and organization-specific aspects of successful SDP adoption.

While SDP's theoretical benefits are widely recognized, real-world deployments have revealed practical integration challenges—particularly around identity federation, controller scalability, and onboarding of legacy systems. Some SDP solutions have since internalized these lessons, embedding capabilities such as dynamic identity federation, horizontally scalable control and data planes, automated legacy integration, and application discovery. Others remain less mature, reflecting varying levels of architectural evolution across the market. These distinctions, further detailed in Appendix 5: SDP Market Survey, underscore the importance of evaluating SDP implementations in terms of design maturity and operational resilience.

Core SDP Concepts

SDP enforces secure access through five core principles:

- **Invisible Infrastructure:** Both the edge infrastructure and protected resources are hidden by default. WAN/edge interfaces cannot be probed or scanned, and unauthorized users cannot detect or reach any service. SPA and similar techniques enforce an "authenticate-before-connect" policy, allowing only validated endpoints to initiate sessions
- **Mutual Authentication and Encryption:** Devices and users must be validated using mTLS, end-to-end (E2E) encryption, certificates, and posture checks
- **Least Privilege Access:** Built on a deny-by-default and microsegmentation model, users and devices are granted access only to explicitly permitted services, significantly reducing opportunities for lateral movement and limiting the blast radius of insider- or compromised-account attacks
- **Dynamic Policy Enforcement:** Access rules are evaluated and enforced in real time based on user identity, device posture, and contextual signals, enabling adaptive, least privilege access. Policies are pre-defined by administrators but applied dynamically as conditions change

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Application-Centric Security:** SDP focuses on access control at the application (or service) layer, preventing broad network exposure and simplifying the management of service-level permissions. Some SDP solutions can provide network-level access when needed, particularly during discovery phases, to learn about services and access them on the network (see CSA Zero Trust guidance, Map The Transaction Flows for Zero Trust)

These principles create a flexible, scalable, threat-resilient architecture that replaces legacy perimeter-based models with identity-first connectivity. In practice, SDP also enables centralized management of access policies and monitoring, integrating with Identity and Access Management (IAM), Security Information and Event Management (SIEM) platforms, and endpoint management systems to provide unified visibility and control across distributed environments.

SDP Architecture

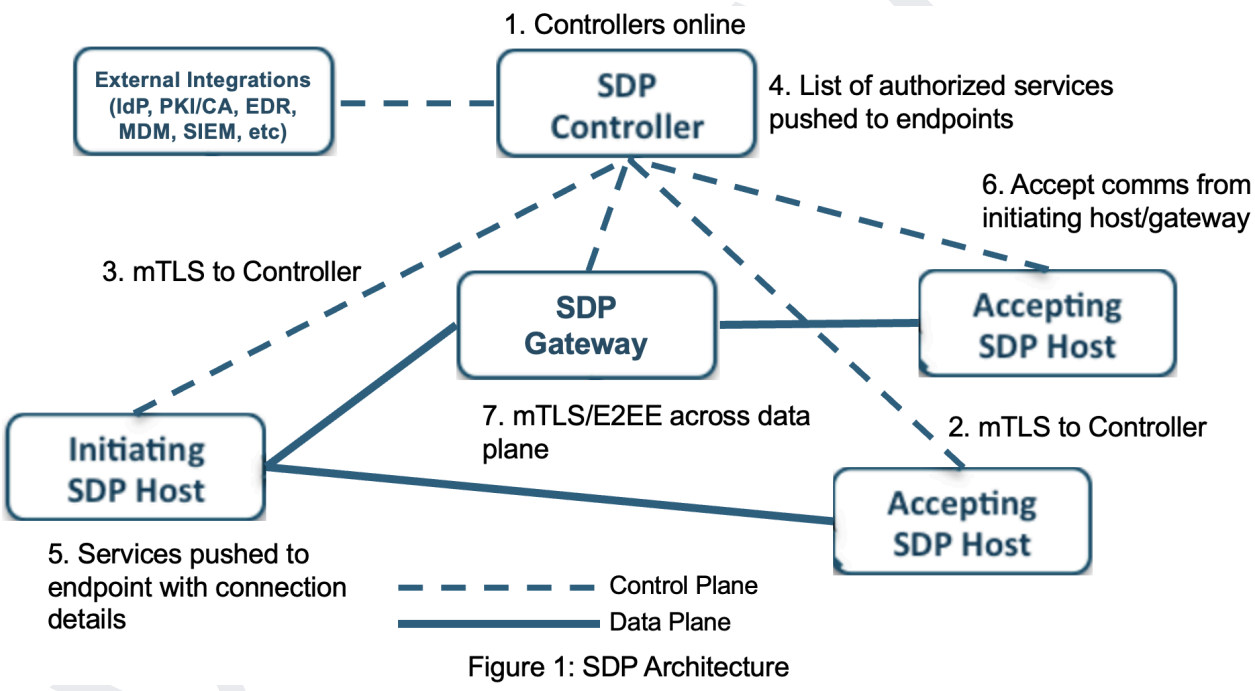


Figure 1: SDP Architecture

At its core, an SDP separates the control and data planes, enabling identity-based, scalable, and secure connectivity. SDP consists of two primary components: **SDP Hosts**, which initiate or accept connections (Initiating Hosts (IH); Accepting Hosts (AH)), and **SDP Controllers**, which manage authentication and policy enforcement via secure channels.

In most implementations, a policy enforcement function—often referred to as a Gateway or Policy Enforcement Point (PEP)—acts on Controller authorization to establish per-resource, encrypted micro-tunnels between trusted endpoints.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Traditionally, SDP systems used **Single Packet Authorization (SPA)** to implement "authenticate-before-connect," concealing infrastructure and allowing access only to verified entities. While SPA remains a valid approach, it is no longer the only option. Modern implementations increasingly leverage protocols like **Identity-First Connectivity** (**IFC**; e.g., **NetFoundry/OpenZiti**) and the **Network Infrastructure Hiding Protocol (NHP)**, offering advanced capabilities such as session-layer security, embedded application identity, posture checks, and encrypted overlays, often with improved performance and scalability.

The basic SDP workflow remains:

1. **Identity and Policy:** The Controller integrates with an Identity Provider (IdP) or native public key infrastructure (PKI) to manage identities and policy. IHs (and admins) authenticate to the Controller.
2. **Endpoint Enrollment and Exposure Control (IH + AH):** Initiating and Accepting endpoints to enroll with the Controller. Services on AHs stay dark to unsolicited traffic; IHs have no network reachability until policy allows a specific service.
3. **Authorization:** The Controller grants least privilege, per-service access based on identity, context, and posture.
4. **Connection Establishment:** After authentication and authorization, the Controller orchestrates session setup by assigning gateways to participating endpoints. Gateways perform preliminary authorization and coordinate with the controller to finalize tunnel establishment. End-to-end encryption is enabled from the initiating to the accepting endpoints, ensuring confidentiality and integrity across the overlay (ideally with private keys only ever generated on the endpoint to stay sovereign, rather than a no centralised key exchange across the wire):
    a. **SPA → mTLS/TLS:** SPA stealthily opens the door; the data session runs over mutually authenticated, encrypted channels.
    b. **NHP:** Session-layer handshake that both hides identities and establishes the encrypted channel.
    c. **Identity-First Connectivity:** Per service (i.e., least privilege), outbound-only connections with mTLS and E2E encryption (no route or service visibility without policy)
5. **Data Flow and Enforcement:** Traffic uses mutually authenticated, end-to-end encrypted, per-resource micro-tunnels, enforcing Zero Trust at the connection level.
6. **Continuous Evaluation:** Policies can be re-evaluated in real time based on changes in posture or risk (e.g., device health), triggering a step-up, restriction, or termination.

Rather than granting users access to a network, SDP enforces a user-to-application model mediated by controllers and connectors. Users are connected only to the specific services they are authorized to use. At the same time, the underlying networks remain isolated, hidden, and unreachable, significantly reducing the attack surface and the risk of lateral movement.

V8 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Operational Considerations and Failure Modes

Controller availability and failover behavior are critical in production deployments. SDP Controllers should operate in high-availability (HA) mode with synchronized state across nodes to ensure policy continuity. If a Controller becomes unreachable, implementations should enforce cached policies locally until reconnection.

HA must also be considered for the data plane. Most SDP implementations recommend deploying multiple connectors or gateways per application, distributed across availability zones or disaster-recovery regions. This allows active sessions to failover transparently if a connector becomes unavailable, ensuring that application access remains resilient even during localized infrastructure failures.

Depending on the environment and risk tolerance, deployments may use fail-closed (block new connections and maintain existing sessions) or fail-open (allow temporary continuity with monitoring) modes. Clear policy definition ensures predictable, resilient behavior without compromising Zero Trust principles.

SDP Implementation Overview Diagram

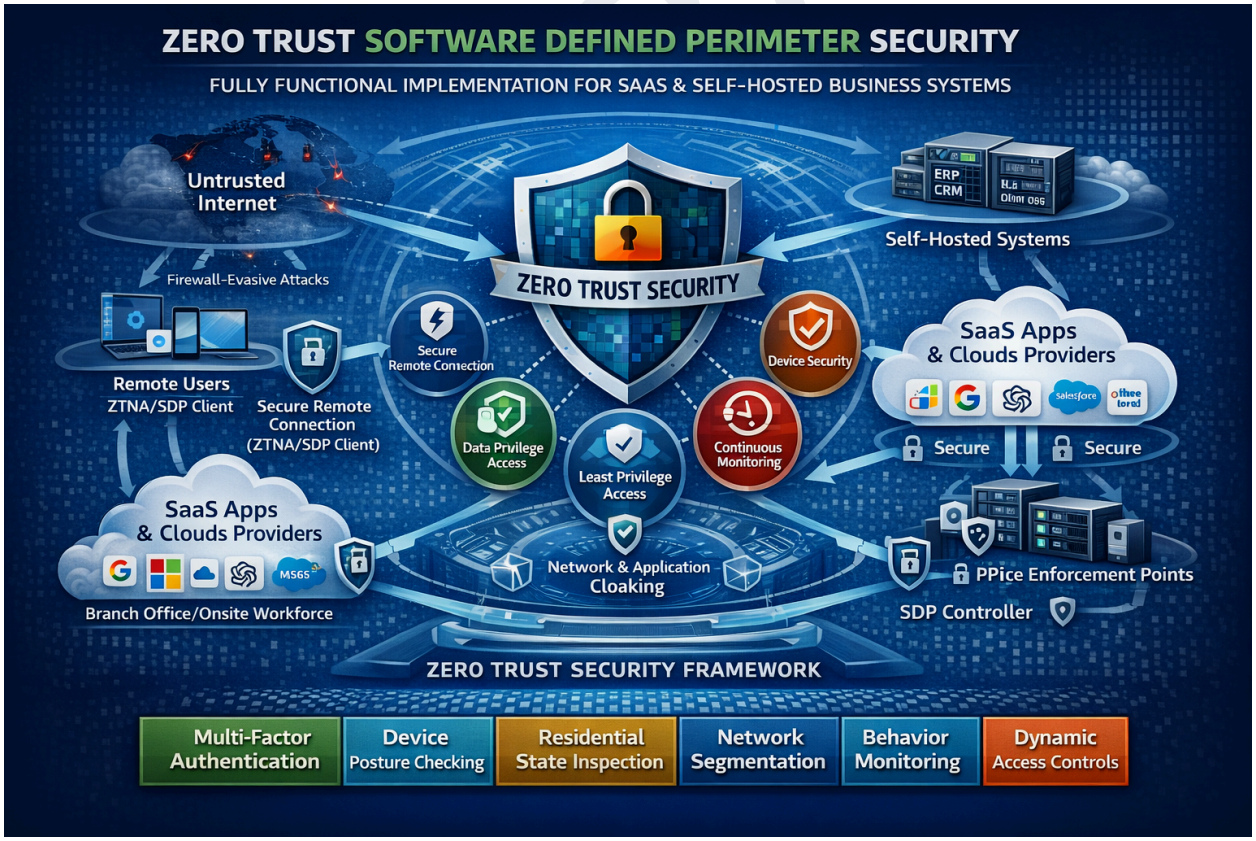


Figure 2: SDP Implementation Overview

© Copyright 2026, Cloud Security Alliance. All rights reserved.

SDP Scenarios

SDP supports real-world access control needs by enforcing identity-based, context-aware policies across diverse environments. Below is a practical summary of common scenarios and use cases.

Scenario/Use Case	What SDP Enables
Identity- and Attribute-Based Access Control	Enforce fine-grained policies by role and contextual attributes (e.g., Finance users on compliant devices to financial apps only). Identity context may be derived from an external IdP via tokens or assertions, or identity context may be derived from native PKI-based credentials within the SDP trust fabric.
Network Microsegmentation	Segment access at the user, device, or service level without requiring manual network changes.
Secure Remote Access (VPN Alternative)	Provide granular, policy-driven user and service access to cloud and on-prem resources, reducing reliance on traditional VPNs while preserving VPN usage for scenarios where network-level connectivity remains appropriate.
Third-Party Access	Provide secure, limited access to contractors or partners across environments.
Privileged Access Management	Restrict and log admin access and enforce time-based and device-based controls.
High-Value Application Access	Hide critical services and control access via identity, device, and posture checks.
Managed Server Access	Overlay complex topologies with just-in-time, policy-enforced admin access.
M&A or Network Integration	Rapidly connect disparate networks without altering the underlying infrastructure.
IaaS Cloud Security	Enforce consistent policies across multi-cloud and hybrid deployments.
Legacy App Authentication	Add MFA or 2FA to apps that can't be modified directly.
Compliance Reporting	Automate access logs and reduce audit scope via segmentation.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Distributed Denial-of-Service (DDoS) Resilience	Default-deny-gateways make systems invisible to unauthorized traffic, reducing exposure to targeted DDoS. In well-architected overlays, distributed gateways, smart routing, and redundant paths further mitigate volumetric DDoS impacts at the underlay.

Table 1: SDP Scenarios

SDP is **not** designed to:

- Control access to unauthenticated public services (e.g., public websites)
- Serve as endpoint protection
- Address all serverless or peer-to-peer scenarios, which vary by deployment model (for example, OpenZiti provides SDKs for app- and function-embedded SDP, supporting serverless use cases, among others)

However, by removing network visibility and enforcing identity-bound connections, SDP significantly reduces the risk of lateral movement and the overall attack surface, complementing endpoint and threat protection controls.

SDP and Modern Zero Trust

The current Zero Trust model, as defined by NIST SP 800-207 and supported by CISA's Zero Trust Maturity Model, is built around the following key principles:

- **Never Trust the Network:** Treat all networks (internal and external) as untrusted and potentially compromised. No access is granted based on network location
- **Strong, Continuous Identity and Posture Verification:** Continuously verify the identity and security posture of users, devices, workloads, and services using strong authentication, authorization, and context (risk, behavior, environment)
- **Fine-Grained, Least-Privilege Access per Request:** Enforce least privilege dynamically for each request or session, using granular policies and micro-segmentation at the app, service, and data layers
- **Assume Breach and Instrument Everything:** Design as if an attacker is already present; log and inspect traffic and activity across users, devices, workloads, and data, feeding analytics and automation to detect, contain, and recover from threats

SDP directly supports these principles by implementing an "authenticate-before-connect" model, ensuring that no network resource is accessible without a verified identity and context. SDP enables

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

secure access across any environment—on-premises, cloud, or hybrid—using strong identity, encrypted connections, and tightly scoped, policy-based authorization.

As a flexible overlay architecture, SDP centralizes visibility and access control, eliminating the need for network location or implicit trust. It enforces least privilege by default, dynamically authorizes connections based on business policy, and generates identity-rich logs for monitoring and compliance. These characteristics position SDP as a foundational implementation strategy for Zero Trust, aligned with the goals outlined in both government and industry Zero Trust frameworks.

Traditional VPN/Firewall Access vs. SDP

This comparison highlights how SDP transitions from location-based to identity-based access control, aligning with Zero Trust principles and eliminating implicit network trust.

Feature/Capability	Traditional VPN/Firewall Access	SDP
Access Model	Implicit trust is once connected to the network.	Authenticate-before-connect, identity-based access.
User and Device Verification	Usually one-time login.	Continuous verification per request.
Policy Enforcement	Network-level, coarse-grained.	Fine-grained, identity-, device-, and context-aware policy enforcement at the session and resource level.
Least Privilege	Often broad network access.	Access is limited to only the required apps/resources.
Third-Party Access	Full network VPN or firewall rule.	Scoped, temporary access per user/device.
Logging and Monitoring	Network logs tied to IP.	Identity-rich logs, allowing SIEM integration on exactly which identity is accessing which services, when, and for how long.
Cloud and Hybrid Support	Requires additional configuration for the cloud.	Native support for cloud, hybrid, and on-prem environments through software-based overlays and identity federation.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Security Posture	Lateral movement is possible after the connection.	Reduces attack surface, hidden from unauthorized users.
MFA/Privileged Access	Separate solutions needed.	Can integrate with PAM, MFA, and SSO, and can trigger step-up authentication based on access context or resource sensitivity.

Table 2: SDP vs. Traditional VPN/Firewall Access

Where SDP Ends and Microsegmentation Begins

Security teams often struggle to explain how SDPs, ZTNA, and microsegmentation fit together. This confusion is largely caused by framing these technologies solely by traffic direction.

In practice, SDP and ZTNA can implement microsegmentation themselves, especially when anchored in a strong identity, which should be inherent to SDP and explicit service definitions—capabilities that are inherent to modern SDP architectures. **The more meaningful distinction is not between SDP and microsegmentation but between connection-defined segmentation and topology-defined segmentation**.

Connection-defined segmentation governs who may establish a connection to a specific service based on identity, posture, and policy, eliminating implicit trust and pre-auth network exposure. Topology-defined segmentation governs where traffic may flow once connectivity exists, typically using workload placement, network boundaries, or observed traffic patterns.

These approaches overlap and frequently coexist. The practical division below reflects common deployment patterns rather than hard architectural limits, acknowledging that some platforms intentionally span multiple categories.

The table below summarizes these two segmentation models by contrasting their intent, enforcement scope, and operational characteristics. It illustrates how connection-defined segmentation, as implemented by modern SDP and identity-native ZTNA solutions, focuses on controlling the establishment of individual service connections across network boundaries, while topology-defined segmentation focuses on constraining traffic movement within a given network or control domain. Together, the comparison highlights where these approaches differ, where they overlap, and why they are often deployed in combination rather than as mutually exclusive controls.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Dimension	Connection-Defined Segmentation (SDP/Identity-Native ZTNA)	Topology-Defined Segmentation (Microsegmentation)
Primary Security Outcome	Attack-surface reduction and least privilege service access.	Lateral movement containment and blast-radius reduction.
Segmentation Intent	Control who may establish a connection to a specific service.	Control where traffic may flow once connectivity exists.
Primary Unit of Segmentation	Service or application identity.	Workload, network segment, port, or protocol.
Trust Model	No implicit trust; every connection is explicitly authorized.	Implicit trust within segments unless explicitly restricted.
Traffic Direction	Excels at north–south; can support east–west when identities are non-human, portable, and policy is enforced independent of network location.	East–west only; enforcement is limited to a single network, fabric, or control domain.
Typical Controls	Identity providers, policy engines, controllers, gateways/connectors, sidecars, embedded SDKs.	Host agents, hypervisors, container network interfaces (CNIs), kernel enforcement, switch access control lists (ACLs).
Network Boundary Awareness	Boundary-agnostic; works across clouds, VPCs, data centers, OT zones, and the WAN.	Boundary-bound; does not natively span network or administrative boundaries.
Pre-Auth Exposure	None—services are dark until policy allows a connection.	Resources visible within allowed segments.
Policy Expression	Identity- and intent-driven (who, what service, under what conditions).	Topology- and flow-driven (source, destination, port, protocol).
Change Cadence	Event-driven (authentication, posture, risk, context).	Lifecycle-driven (deployments, topology changes, discovered flows).
Operational Trade-Off	Requires explicit service definitions and identity modeling.	Easier to apply broadly; harder to keep precise at service granularity.
Rationale	Eliminates unnecessary exposure by preventing unauthorized connections from ever forming.	Limits damage after compromise by constraining reachable paths.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Where They Intersect	When services are first-class identities, SDP inherently enforces microsegmentation.	Some platforms enrich segmentation with identity and context.

Table 3: Comparison of Segmentation Models

Why This Division Works

- **Segmentation Intent:** North-south use cases emphasize connection-defined segmentation—eliminating attack surface through identity-based admission before connectivity exists. East-west environments historically rely on topology-defined segmentation to constrain lateral movement. In practice, identity-native SDP and ZTNA implementations can extend connection-defined controls inward, effectively delivering microsegmentation at the service layer when services are modeled explicitly. This is particularly crucial when microsegmentation must take place across network boundaries, where technologies such as network address translation (NAT) cause network identifiers to fail
- **Operational Fit:** SDP policy changes are typically event-driven (e.g., user login, device posture change), whereas microsegmentation policies often evolve with every code deployment within a CI/CD pipeline. Some platforms blur this distinction by tying identity-driven access to workload lifecycles
- **Performance Optimization:** SDP tunnels can terminate at WAN edges or gateways, minimizing round-trip latency for users. Microsegmentation control points sit next to workloads to avoid hair-pinning high-volume east-west traffic, which is essential given that, typically, every 1 MB of north-south traffic spawns ~20 MB of east-west chatter inside a data center (actual ratios will vary by application and must be measured during discovery). Identity-aware SDP/ ZTNA designs that enforce controls closer to workloads can help optimize both patterns simultaneously

Architectural Guardrails

- **Shared Policy Fabric:** Use common identity claims (x509/JWT, OIDC, SVIDs, device certificates) so that SDP and microsegmentation engines evaluate the same subject attributes even though they are enforced in different planes
- **Consistent Labeling:** Tag workloads and data flows once (e.g., env:prod, app:payments) and push those tags to both the SDP controller (for user-to-app rules) and to microsegmentation orchestrators (for app-to-app rules), and forward tagged flow logs to a SIEM or analytics engine for continuous risk assessment
- **Scope Creep Checks:** Feed SDP logs of authenticated sessions into the microsegmentation discovery engine; unusual east-west flows that appear without a preceding north-south request are prime indicators of compromise

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Edge Cases and Overlap

Some products blur the line: high-performance SDPs can form micro-tunnels between workloads (e.g., function-to-function inside Kubernetes), and host-based micro-seg platforms can gate user traffic. Nonetheless, framing SDP as a north-south first-packet guard and microsegmentation as east-west blast-radius containment provides architects with a clear mental model, while leaving room to consolidate tools when capabilities converge.

If you are interested in learning more about microsegmentation, CSA is currently writing a paper about it which is coming soon and will be published here.

Implementing SDP, Evolving its Mechanisms and Mapping Use Cases

This section translates the SDP and Zero Trust architectural principles into actionable implementation use cases. It introduces a structured framework that begins with defining the protect surface and mapping transaction flows, establishing visibility through telemetry feeds and flow inventories, and progressing toward risk evaluation, maturity assessment, and policy formulation. The framework demonstrates how to evolve from flow analysis to enforceable Zero Trust policies, providing context for updates in CSA's SDP Architecture Guide v3.0. It further examines how SDP components and deployment models adapt across diverse environments—hybrid and multi-cloud infrastructures, OT, and edge and IoT ecosystems. The section concludes with an exploration of SPA, detailing its operational benefits and inherent limitations in modern deployments.

Implementation Framework

This subsection explains **how to discover, document, and continuously refine the flows that an SDP must protect** and how this mapping drives the three core SDP capabilities: least privilege access, dynamic policy enforcement, and application-centric security. It adapts proven transaction-flow-mapping guidance from the CSA Zero Trust paper, Map the Transaction Flows for Zero Trust, and applies it directly to SDPs.

Why Map Transaction Flows in an SDP?

Core SDP Concept	Contribution of Transaction Flow Mapping
Least Privilege Access	Identifies exactly which user/device needs which application flow so SDP controllers can issue the minimal set of allow-rules and block all other east-west traffic.

V5 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Dynamic Policy Enforcement	Supplies real-time telemetry (flows, identities, device posture) to the policy engine, which uses it to create, adjust, or revoke micro-tunnels on demand.
Application-Centric Security	Produces an application-level topology, including URLs, ports, service calls, and APIs, enabling the SDP to cloak entire services rather than raw subnets.

Table 4: Contributions of Transaction Flow Mapping

The CSA Zero Trust methodology refers to this step as **Mapping the Transaction Flows.** For an SDP, the same step unfolds in five repeatable passes that can be automated over time:

- **Scope the Protect Surface:** Enumerate the application or service to be hidden and its Data, Applications, Assets, Services (DAAS) elements
- **Discover Live Glows:** Use packet brokers, Switched Port Analyzer (SPAN), cloud flow logs, service-mesh telemetry, or the SDP's network-level discovery probes to capture who (ID), what (device), where (IP/FQDN), when, and how (protocol) each transaction occurs
- **Classify and Label:** Tag each flow with business context (e.g., "Customer-Portal-Read," "Build-Pipeline-Push") so the policy engine can reason at an application level. Vendor features such as Flow-Label Discovery or Discovery and Policy AI accelerate this step
- **Model Access Paths:** Draw a before/after diagram showing the current any-to-any paths and the intended SDP micro-tunnels. This becomes the authoritative source for least privilege baselines
- **Validate and Iterate:** Replay logs, run synthetic tests, and solicit app-owner feedback; feed drift or new dependencies back into the model. In mature environments, this loop is continuous and largely automated

Practitioners should ensure three key artifacts emerge from this process: (1) a consolidated flow inventory enriched with observed telemetry, (2) annotated data flow diagrams at the application layer, and (3) a risk and maturity matrix highlighting gaps. These outputs directly inform least privilege policies that SDP platforms can enforce or automatically suggest.

For practitioners, the deliverables of "mapping transaction flows" can be summarized as the following checklist (inputs → artefacts → policy outputs):

1. Protect surface definition (DAAS scope)
2. Observed transaction flows (telemetry: SDP logs / NetFlow / service mesh / cloud flow logs)
3. Consolidated flow inventory (labeled)
4. Annotated application flow diagrams
5. Risk and maturity matrix
6. Proposed least-privilege policies (baseline + suggested)

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

From Map to Policy

- **Generate Micro-Segments:** Group flows that share identical labels (e.g., app, purpose, compliance scope), then create a single SDP policy object per group
- **Bind Identity and Posture:** Link each flow's source to its federated identity claims and device trust score; import from IdP, PKI, certificate state, or EDR
- **Set Dynamic Conditions:** Attach contextual triggers (e.g., time, location anomalies, certificate status, threat intelligence scores) to ensure optimal performance. When a condition flips, the controller tears down the tunnel instantly, fulfilling **dynamic policy enforcement**
- **Test and Simulate:** Run "policy-suggest" in **report-only** mode first; verify that no legitimate flows break before enforcing

Automation and AI/ML Accelerators

Capability	Benefits of Mapping the Transaction Flows
Discovery and Policy AI	Clusters flows, labels services, and predicts least privilege sets, cutting manual rule-writing.
Unsupervised Anomaly Detection	Flags previously unseen flows for review to prevent silent scope creep.
Continuous Validation Pipelines	Injects synthetic traffic each CI/CD cycle; blocks deployment if the new flow isn't in the approved map.
Policy Drift Detection	Monitors runtime flows against approved patterns and alerts on deviations, helping prevent unintended expansion of authorized connectivity over time.
Lifecycle Governance	Applies expiry, revalidation, and ownership to flow policies to keep authorizations aligned with the current application architecture.

Table 5: Benefits of Flow Mapping for Automation and AI/ML Accelerators

By grounding SDP policy in a rigorously maintained transaction-flow map, implementers ensure that every tunnel enforces the principle of least privilege, adapts in real time, and remains tightly aligned with the application, not the network.

AI-assisted discovery and policy generation should be treated as recommendations rather than autonomous enforcement actions, with a human review loop to validate suggested changes before deployment.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

SDP as an Accelerator for Secure by Design

Jen Easterly's 2024 challenge, "*We don't need more **security** products; we need more **secure** products,*" has become the rallying cry of CISA's **Secure by Design** program. CISA's joint guidance defines three duties for every vendor: **(1) take ownership of customer-security outcomes, (2) embrace radical transparency and accountability, and (3) lead from the top.** An SDP naturally maps to each duty because it embeds least privilege access, continuous policy governance, and application-centric controls into the product itself rather than adding them later.

CISA Secure by Design Principles	What an Embedded SDP Delivers
Own Customer-Security Outcomes	Services start in a **deny-by-default** state, accepting traffic only from strongly identified users/devices, and exposing **no open ports** to the internet, thereby removing the customer's burden of perimeter hardening.
Radical Transparency and Accountability	Every micro-tunnel is authenticated, authorized, and logged at the service layer, providing vendors with tamper-proof evidence of secure coding claims and customers with real-time assurance metrics.
Lead From the Top	Engineering teams adopt SDP and policy pipelines early in the SDLC, so security gates are automated in CI/CD rather than relegated to post-release patches.

Table 6: SDP and Secure by Design Principles

A growing ecosystem of open-source projects lets engineering teams "bake in" Zero Trust controls rather than bolt them on. At the network-overlay tier, projects such as **OpenZiti** (Identitiy-First Connectivity) provide SDKs or lightweight agents that developers can ship alongside their binaries, eliminating exposed ports, enforcing mutual authentication, and SDP out of the box. At the service-to-service tier, the Secure Production Identity Framework for Everyone (SPIFFE)/SPIRE (SPIFFE Runtime Environment) framework issues cryptographic workload identities that applications can request programmatically. At the same time, CNCF service-mesh projects—such as Istio, Linkerd, and Consul Connect—inject mutual TLS and fine-grained policy (but not full SDP) at runtime with minimal code changes, delivering secure-by-default communication inside Kubernetes or VM estates. Together, these tools make it practical to embed the same least privileged, identity-driven principles championed by SDP directly into new or existing products.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

An example of this is **EdgeX Foundry**, which, in its 4.0 2025 release, illustrates the model at scale. The industrial IoT framework now provides end-to-end authentication built on OpenZiti's SDP fabric, giving every microservice an implicit, mutually-authenticated tunnel that users cannot misconfigure or expose.

What This Means for CSA's SDP Architecture Guide v3.0

- **Design Guidance:** Recommend that product teams treat the SDP controller, policy engine, and identity broker as first-class modules in their reference stacks, alongside logging and observability
- **Open-Source Pathways:** Highlight projects that lower adoption barriers and encourage community scrutiny, fulfilling the transparency mandate
- **Certification/Maturity Mapping:** Show how an embedded-SDP product can evidence compliance with Secure by Design principles and the CSA SDP maturity levels concurrently

By integrating SDP capabilities deep inside the software supply chain, vendors answer CISA's Secure by Design challenge with concrete, verifiable controls, transforming "more secure products" from a slogan into a shipping requirement.

SDP Architecture Components and Functions

An SDP enforces Zero Trust principles through three primary components:

- **Initiating Host (IH):** A client device (e.g., laptop, mobile, server) that initiates connections
- **Accepting Host (AH):** A protected resource or service that receives authorized connections
- **SDP Controller/Fabric:** The central infrastructure that provides the authority to authenticate identities, authorize access, and orchestrate connections, possibly over a separate data plane (or Turn-Server), across the architecture

These components interact over mutually authenticated and encrypted channels, often enforced by mechanisms such as SPA, the **Network** Infrastructure **Hiding Protocol (NHP)**, or **OpenZiti**. The architecture should cleanly separate the control plane (authentication and policy) from the data plane (actual traffic), providing scalability, resilience, and security.

Single-Packet Authorization and Its Evolution

A foundational principle of many SDP architectures is the **"authenticate-before-connect"** model, which compensates for TCP/IP's inherently open nature. Single-Packet Authorization (SPA) is a mechanism historically used to enforce this model by validating identity and authorization before any network-facing component, such as a controller or gateway, accepts a connection.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

SPA uses a cryptographically protected, single message to convey authentication and authorization data. Until a valid SPA message is received and verified, protected services remain non-responsive ("dark") to unsolicited traffic. This approach significantly reduces the attack surface by preventing service discovery, port scanning, and unauthorized connection attempts.

While SPA is frequently associated with SDP, it is one mechanism among several that can be used to achieve "authenticate-before-connect." Modern SDP systems may instead rely on mutual TLS (mTLS), client certificates, identity brokers, or embedded identity-aware overlays to achieve equivalent or stronger guarantees.

Core SPA Design Principles

Regardless of implementation, SPA mechanisms should adhere to the following principles:

- **Encrypted and Authenticated Payloads:** All SPA data must be protected with cryptographic mechanisms to ensure confidentiality and integrity
- **Self-Contained Authorization:** The SPA message must carry all required identity and authorization information; external packet headers must not be trusted
- **Silent Failure:** Invalid or unauthorized SPA attempts must elicit no outward response
- **Replay Protection:** Implementations must prevent replay attacks using nonces, timestamps, or sequence validation
- **Strong Key Management:** Secure key distribution, rotation, and revocation are critical to maintaining trust

Benefits of SPA

When implemented correctly, SPA provides several security advantages:

- **Service Cloaking:** Protected services are not discoverable via reconnaissance or scanning
- **Reduced Exposure to Zero-Day Exploits:** Attacks targeting services are ineffective if unauthorized users cannot establish a connection
- **Reconnaissance Resistance:** Protocol-level enumeration is prevented by default

Practical Limitations of SPA

Despite its benefits, SPA has operational and architectural limitations:

- **Client Dependency:** SPA typically requires specialized client software, which may be impractical for Bring-Your-Own-Device (BYOD), IoT, or constrained devices
- **NAT and Firewall Traversal:** UDP-based SPA can be blocked in restricted networks, requiring TCP- or HTTPS-based fallbacks
- **Key Compromise Impact:** Shared-secret SPA models require coordinated re-keying if credentials are exposed

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **High-Frequency Workloads:** SPA introduces per-connection overhead that can be inefficient for short-lived or high-rate service interactions
- **Operational Visibility:** "Dark" services complicate troubleshooting and require alternative monitoring approaches

Evolution Beyond Classic SPA

These limitations have driven the evolution of SDP implementations toward **connection-defined, identity-native approaches** that preserve "authenticate-before-connect" semantics without relying on single-packet constructs.

Newer designs - such as NHP-style models and platforms built on **OpenZiti** - extend the core SPA concept by:

- Establishing mutually authenticated, identity-bound overlays rather than single-packet triggers
- Supporting non-human identities (services, workloads, devices) as first-class principals
- Maintaining persistent or multiplexed tunnels to reduce per-connection overhead
- Enforcing policy continuously across the lifetime of a connection
- Operating independently of the underlying network topology or addressability

In these architectures, "service darkening" is achieved not through a single authorization packet but through identity-gated connectivity, where unauthorized entities cannot route, resolve, or establish sessions to protected services.

Summary of SPA

SPA remains a valid and effective mechanism for enforcing "authenticate-before-connect" in certain SDP deployments. However, it should be viewed as an **implementation technique rather than a defining requirement of SDP**. Modern SDP architectures increasingly generalize the SPA concept into identity-driven, policy-enforced connectivity fabrics that provide the same security outcomes—often with greater scalability, flexibility, and operational resilience.

Extending SDP to Modern Use-Case Domains

The original SDP pattern focused on cloaking datacenter workloads from remote users, but digital transformation has pushed these same principles into very different environments. Each domain still benefits from least privilege, dynamic policy, and application-centric security; however, the threat model, connectivity assumptions, and operational constraints differ sufficiently that the architecture must be tailored for success.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Hybrid and Multi-Cloud Workloads

Modern enterprises increasingly span multiple public clouds, SaaS platforms, edge environments, and on-premises infrastructure. In SDP v3.0, this reality is no longer limited to traditional application workloads. Agentic AI systems—composed of autonomous agents, models, tools, and data services—are inherently multi-cloud and cross-domain by design.

Workloads are highly ephemeral: containers scale in seconds, serverless functions execute for a single request, and autonomous AI agents dynamically create, delegate, and retire tasks as part of AI/ML pipelines. Agentic systems frequently span multiple trust domains, invoking external APIs, proprietary tools, partner services, and internal data sources. As a result, the control plane must cope with machine-generated identities, short-lived execution contexts, and heterogeneous metadata across clouds and vendors.

In this environment, security controls must attach to the workload or agent identity itself and move with it, rather than being bound to static IP addresses, subnets, or firewall rules. This principle applies equally to microservices, batch jobs, and long-running AI agents coordinating work across clouds.

What's Different?

- **Instant, Elastic Endpoints:** Autoscaling groups, serverless cold starts, and autonomous AI agents create and retire workloads and identities faster than traditional asset or configuration management can track. SDP v3.0 assumes identities are created and revoked continuously and enforces policy dynamically
- **Poly-IdP Reality:** Enterprises routinely operate multiple identity providers: workforce IdPs (e.g., Okta), cloud IAM systems, SaaS-native OIDC providers, and specialized IdPs for workloads and agents. Agentic AI systems may introduce additional non-human identities that do not map cleanly to user-centric IAM models. SDP v3.0 treats identity federation as a baseline requirement
- **Tag-Rich and Semantic Context:** Every entity—human or non-human—arrives with metadata such as environment, ownership, compliance classification, sensitivity, and operational role. In agentic systems, this context may include agent role, tool capability, or data domain. SDP v3.0 policies are expected to exploit this semantic richness rather than rely solely on network location
- **Cross-Domain Agent Graphs:** Agent-to-agent (A2A) and agent-to-service interactions frequently span private data centers, VPCs, SaaS platforms, and public AI services. These interactions are bursty and short-lived, making static network allowlists impractical and unsafe
- **Cloud Backbone Hair-Pins as an Anti-Pattern:** East-west traffic within or between clouds may already traverse provider backbones. Forcing agent or service traffic through centralized gateways introduces unnecessary latency, path inflation, and operational coupling. SDP v3.0 assumes policy decisions can be made locally without mandatory traffic hair-pinning

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Architectural Adaptations

- **Federated Policy Plane:** Policy remains logically centralized, but authorization is executed locally via stateless trust brokers or controllers deployed per cloud region, VPC, or trust domain. This enables low-latency decisions and avoids centralized chokepoints (or hair-pins), while preserving consistent policy semantics across domains
- **In-Process SDP SDKs/Sidecars:** For cloud-native workloads, serverless functions, and autonomous agents, SDP enforcement may be embedded directly in the application or delivered via lightweight sidecars. This allows agent-to-agent and agent-to-tool connections to originate as identity-bound, outbound-only sessions, independent of pod, node, or VM IP addresses
- **Latency-Aware Overlay Placement:** Mature SDP v3.0 implementations continuously evaluate latency, proximity, and trust boundaries to optimize controller federation and overlay routing. This is particularly important for agentic systems that orchestrate workflows across clouds and external services
- **Tag-to-Identity Mapping:** Cloud IAM roles, workload tags, and agent metadata are combined into composite identities within the SDP policy graph. For example, an AI agent operating on regulated data may inherit stricter posture requirements or additional authorization checks without requiring static rule changes
- **Event-Driven Policy Updates:** Use cloud-native events (e.g., AWS EventBridge) to send "function started/terminated" notifications to the SDP controller; policies expire automatically when the function completes
- **Drift-Aware Auto-Revocation:** When workloads or agents scale down, move regions, or change posture, SDP v3.0 controllers detect drift and instruct enforcement points to tear down stale sessions within seconds. This is critical for limiting blast radius in autonomous systems where actions may propagate rapidly

Implications for Agentic AI Systems

In agentic AI workflows, SDP v3.0 provides the connective tissue that allows agents, tools, and services to collaborate securely across domains without exposing network surfaces. Rather than relying on static VPNs, shared subnets, or long-lived credentials, agent interactions are authorized per call, audited per flow, and revoked automatically when context changes.

This approach enables enterprises to scale agentic systems while maintaining Zero Trust guarantees, reducing attack surface, and preserving operational agility in hybrid and multi-cloud environments.

Applying SDP and Zero Trust to AI and agentic systems is a topic of great interest to the CSA. We have other papers in progress, so contact us here if you are interested in contributing or reading more.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Operational Technology (OT) and Industrial Control (IC)

Industrial control systems (ICS) drive physical processes, such as manufacturing lines, energy grids, and transportation systems, prioritizing safety and availability above all else, including running deterministic processes that must keep operating even if the WAN dies. Many devices are decades old and cannot run modern agents, yet they sit on increasingly converged networks where ransomware or lateral movement can halt production.

What's Different?

- **Safety-Critical Uptime:** Stopping a packaging line or turbine trip can cost millions or threaten lives
- **Legacy Protocols:** Modbus, DNP3, and PROFINET are often unauthenticated and broadcast-chatty
- **Purdue Segmentation Under Strain:** L0–L3 were traditionally isolated with no direct internet access, but modern needs—remote access, cloud, and IT/OT convergence—are pushing connectivity deeper into OT networks. This must not compromise critical operation while being consistent with IEC 62443
- **Vendor Lock-In Devices:** Firmware is often signed exclusively by the OEM, making agent installation impossible

Architectural Adaptations

- **Zone-Resident Controllers:** A lightweight SDP controller instance resides in Level-3/DMZ, allowing local authorization to continue during a WAN outage. It syncs policy to the mothership opportunistically, or the orchestration runs completely on-premises and locally
- **Protocol-Aware Micro-Tunnels:** The gateway parses industrial frames and whitelists exact register ranges or function codes (e.g., Modbus 0x03 read-holding-register), denying all other traffic
- **Agentless Inline Gateways:** Inline devices (DIN-rail or rack-mount) that ingest 2-wire or serial traffic, encapsulate it in an SDP tunnel, and re-emit on the far side (zero firmware change)
- **Safety-Side Fail-Open Logic:** Policy defines graceful degradation—read-only if controller unreachable, or local human override switch, meeting IEC 62443 safety clauses
- **Digital Twin Replay:** Capture authorized OT flows into a replayable model; any runtime packet that deviates from the model triggers an automatic SDP disconnect and an ICS SOC alert
- **Time-Bound Vendor Windows:** Temporary access tokens for OEM maintenance crews that self-expire; access is possible only via the SDP broker with a posture-checked laptop

An example of SDP (and Zero Trust networking principles) being applied to OT and IC comes from Siemens, with their SINEC Secure Connect offering: SINEC Secure Connect Zero Trust - Siemens Global.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Edge and IoT Fleets

Smart meters, surveillance cameras, HVAC controllers, robots, and legacy PLCs now sit at the edge of every enterprise. They run on modest CPUs, speak dozens of proprietary protocols, change networks as they roam, and are often physically reachable by attackers. Traditional perimeter security assumes a stable location and a rich operating system, assumptions that break down completely in IoT/OT environments. An SDP overlay brings the Zero Trust mandate ("always verify, never trust") directly to these constrained, often-unpatchable devices by cloaking them, authenticating them with lightweight cryptography, and allowing only policy-approved micro-tunnels for the precise functions they must perform.

What's Different?

- **Severe Resource Limits:** Many endpoints have only a few hundred kilobytes of RAM and limited power; heavyweight agents or frequent TLS handshakes are not feasible options
- **Intermittent, Metered Links:** Devices connect via NAT'ed 4G/5G, LoRa, satellite, or consumer Wi-Fi, so every round-trip and every byte incurs a cost in terms of both money and latency
- **Roaming Identities:** IP addresses change constantly; relying on subnets for security or policy enforcement is futile
- **Physical Exposure and Supply-Chain Risk:** Attackers can tamper with hardware in the field or slip counterfeit boards into procurement pipelines
- **Flat Legacy Networks:** OT topologies were designed for availability, not segmentation, so once an attacker lands, lateral movement is trivial unless contained

Architectural Adaptations

- **Embedded Overlay SDKs:** Ultra-light libraries (<50 kB) compiled into firmware enable each process to originate an SPA handshake and then maintain an identity-bound DTLS or mTLS-over-QUIC tunnel—no open ports, no banner leakage
- **Fan-Out Edge Gateways:** When code cannot be embedded (e.g., 1990s PLCs), a transparent "tap-and-wrap" gateway terminates SDP externally while communicating internally in raw Modbus, BACnet, or CANbus, providing segmentation without modifying legacy firmware
- **Hardware-Rooted Bootstrap:** Devices enroll automatically using TPM attestation, eSIM/SIM credentials, or printed hardware serial numbers, which are mapped to X.509 certificates. No human-driven key loading is required, and counterfeit boards are rejected during enrollment
- **Bandwidth-Sparing Heartbeats:** After establishment, tunnels remain active with small SPA keep-alives and delta policy updates, updating only when rules change, thereby reducing airtime and preserving battery life
- **Local Policy Caching:** A trimmed-down decision engine runs in the gateway (or on the device for higher-end nodes), allowing authorized flows to continue for hours if the central controller is offline—critical for safety-of-life OT processes

V6 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Contextual Revocation and Micro-Segmentation:** Policies examine geo-location, posture, behavior, and firmware version. A camera that suddenly reports from an unexpected country is forced to re-authenticate; a compromised sensor cannot pivot to HVAC or IT VLANs because no route exists outside its micro-segment

Cross-Cutting Lessons

Across cloud, plant-floor, and edge deployments, the same five architectural tenets keep an SDP resilient and usable, even though the mechanics differ in each domain:

- **Identity Outranks Topology:** The only constant is a cryptographic identity that follows the workload or device, whether it is a Lambda function labeled by cloud tags, a PLC identified by a laser-etched serial number, or a sensor bootstrapped with a TPM key. Policies should be bound to that identity graph, not to IP addresses, SIM cards, or VLANs that change frequently
- **Control and Data Planes Must Respect Physics:** Central policy engines simplify governance, but decision caches should be located near the traffic they manage. Place sidecar brokers within every cloud VPC, Level-3 gateways within each OT zone, and lightweight policy stores on edge gateways. That local presence keeps latency low and allows factories, field sensors, or autoscaling clusters to run safely when the WAN experiences blips
- **Least Privilege Scales Better Than Rule Sprawl:** Begin with "deny all," then let the flow-discovery and Policy-AI modules watch live traffic to recommend the minimum set of tunnels each workload truly needs. The larger and more ephemeral the fleet, the more valuable that automated suggestion loop becomes Further, identity-based services scale much more effectively than ACLs, while providing a single place/API to audit, rather than 15-20 policies/rules across multiple tools, firewalls, and other infrastructure to allow communication between workloads.
- **Fail Gracefully, Never Catastrophically:** A lost controller should degrade to read-only, last-known-good, or human-override, not to "wide open" or total shutdown. Hybrid clouds tolerate a brief 503, but a packaging line or smart-grid relay cannot; bake those differences into fallback logic
- **Visibility is Half the Victory:** SDP's per-tunnel logs - who, what, where, with what posture - provide SecOps and GRC teams with a level of fleet-wide telemetry that traditional OT networks and VPNs have never offered. Stream the data to SIEMs for correlation and feed risk scores back to the SDP for auto-quarantine, closing the loop from detection to enforcement in seconds

Tailoring gateway placement, identity bootstrap, and transport choices allows implementers to apply SDP's core principles to hybrid clouds, factory floors, and far-flung IoT edges, domains the original "remote-access" SDP never contemplated, without compromising on least privilege, dynamic policy, or application-centric control.

SDP and Access Control

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

SDP can enforce access control by preventing network-level access from unauthorized users using unvalidated devices. Because SDP provides a deny-all gateway, it allows or prevents network packets from flowing between the IH and AH. The value of SDP as an emerging architecture lies in its ability to enable access control management, setting standards for implementing user access management, network access control, and system authentication control, thereby protecting information assets and network services. At a minimum, SDP enables organizations to define and control access policies that determine which identities (human or system) should be permitted to access which target network services and from which validated devices.

To be clear, SDP doesn't attempt to displace existing identity and access management solutions. But SDP extends and improves upon user authentication-only access control. Instead, SDP significantly decreases the attack surface by integrating user authentication and authorization with other components (see Core SDP Concepts section). For example, our favorite user, Jane, might not have the credentials to sign in to the company's production financial management server. Still, if it is visible on her device on the network, it represents a risk. SDP hides the company's production financial management server from Jane's device. Even if an attacker gains a foothold on Jane's machine, SDP will prevent connections from her device to the financial management server for which she lacks credentials. Having an SDP client on Jane's device provides an additional layer of protection. Even if Jane has credentials for the financial management server, the attacker would still find it hard to connect with multi-factor user authentication and strong device validation enabled. To fully realize these benefits, applications must be reachable only through SDP-controlled paths. Partial deployments, such as remote-access-only use cases, may still allow alternate network access unless complementary controls are in place.

SDP achieves the least privilege principle by combining network enforcement with identity-centric and descriptive access rules. SDP Guidance is published under the Zero Trust Working.

SDP and Enterprise Security Architecture

This section has two goals. First, it will help readers planning to deploy SDP understand how it can fit into their enterprise security architecture, especially regarding existing tools and technologies with which it will coexist, integrate, or replace. Second, it will highlight the areas where SDP will coexist and integrate with existing IT and security infrastructures, allowing system designers to envision the security architecture for their target deployment more clearly and to document it, including SDP.

Architecture Considerations

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Enterprise security architectures[1] can be complex, with numerous organizational stakeholders. The following sections explore areas architects must consider as they plan to deploy and adopt SDP across their enterprise, addressing various user populations, networks, server environments, and security and compliance requirements.

Ultimately, SDP is an architecture that can be the basis for enterprise security architectures because it allows for key concepts (see Core SDP Concepts section) such as:

- **Invisible Infrastructure:** Unauthorized users cannot detect or connect to protected resources. SPA and similar protocols or techniques ensure that only authenticated devices can initiate connections
- **Mutual Authentication and Encryption:** Devices and users must be validated using mTLS, E2E encryption, certificate pinning, and posture checks
- **Least Privilege Access:** Users and devices are granted access only to explicitly permitted services, eliminating lateral movement and reducing insider threat risks
- **Dynamic Policy Enforcement:** Access rules are created and revoked in real-time, enabling adaptive, context-aware access control
- **Application-Centric Security:** SDP focuses on access control at the application (or service) layer, preventing broad network exposure and simplifying the management of service-level permissions

At the highest level, architects need to understand that SDP ensures secure connections irrespective of the underlying IP infrastructure and that several questions need answering:

- **How does an SDP deployment fit into existing network topologies and technologies?** Architects must decide which SDP deployment models they use and understand that gateways may represent an additional in-line network component for some models. Architects don't have to pick a single model for all aspects of the environment; they can use models where it makes sense. This may have implications for the network, such as requiring changes to the firewall or routing to ensure that protected servers are rendered invisible and accessible only through the SDP Gateway
- **How does SDP affect monitoring and logging systems?** Because SDP uses mTLS between the IH and AH, this network traffic becomes opaque to intermediary services that may be in place for security, performance, or reliability monitoring. Architects must understand the systems in operation and how changes to network traffic may affect them. On the flip side, because SDP systems typically provide richer, identity-centric logging of user access, they can often augment and enhance existing monitoring systems. Additionally, all dropped packets from SDP Gateways and Controllers can be logged in a SIEM for further analysis. The who, when, what, and from-where information for every connection is easier to collect when an SDP is in place
- **How does SDP affect application release/DevOps processes and toolset, including API integrations?** Many organizations have adopted high-velocity application release processes,

[1] Sometimes referred to as enterprise information security architecture - Wikipedia

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

such as DevOps or CI/CD[2]. These processes and their supporting automation frameworks require thoughtful integration with security systems, and SDP is no exception. During DevOps, SDPs can effectively secure authorized users' connections to the development environment (as well as across environments, as long as the SDP technology also supports non-human identities (NHI) and connections). They can also be used during operations to protect connections from legitimate users to protected servers and applications. Security architects must understand their chosen SDP deployment model and how their organization's DevOps mechanisms will interact and integrate with it. Security teams should also examine the set of APIs supported by their SDP implementation, as API integration is often required for DevOps toolsets

- **How does SDP impact users, especially business users?** Security teams typically strive to make their solutions work as transparently as possible for users. SDP supports thi– if the principle of least privilege is achieved, users will have full access to everything they need and won't notice that unnecessary access has been removed. Depending on the SDP deployment model, users will run the SDP client software on their devices. Security architects should collaborate with IT to model and plan for the user experience, client software distribution, and device onboarding processes

Architecture Elements: Where an SDP Fits and Where It Doesn't

An SDP does not float above the existing stack; it plugs directly into every control plane that a hybrid enterprise already runs. The diagram earlier in the paper illustrates a highly simplified environment featuring on-premises and remote users, IaaS and SaaS workloads, and a familiar set of security, IT, and compliance tools. The table and notes below map each tool to its new role once an SDP is introduced, highlighting whether the **SDP replaces, augments, or complements** the control, and what design tweaks are required.

V3 DRAFT

[2] https://en.wikipedia.org/wiki/DevOps

© Copyright 2026, Cloud Security Alliance. All rights reserved.

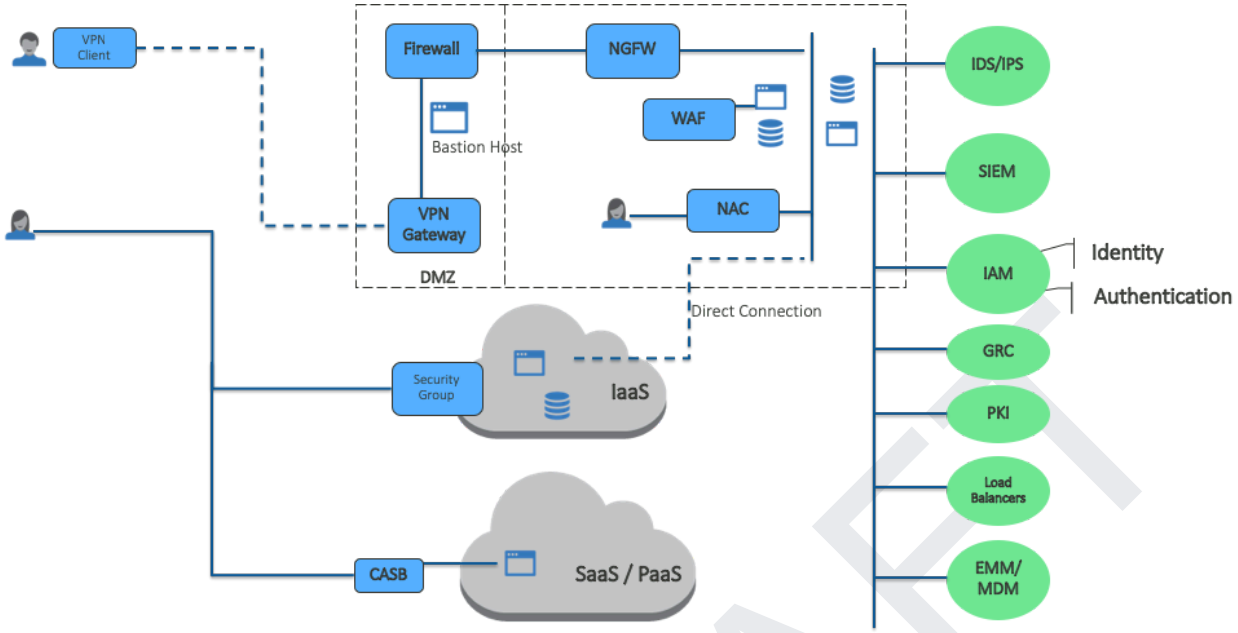


Figure 2: SDP Plugs into Every Control Plane

Legacy/Adjacent Control	Traditional Purpose	SDP Relationship	Key Integration Moves
VPN Concentrators	Encrypted "full-tunnel" access for remote users and sites.	**Replaces:** SDP gives application-specific, just-in-time tunnels that remove broad network reach and eliminate back-haul latency.	Retire split/full-tunnel VPN profiles; point remote users at the SDP client instead.
Traditional Firewalls	Permit/deny by IP, port, and protocol at the perimeter.	**Overlaps → shrinks rule-set:** The gateway function is absorbed by the SDP access node, leaving the firewall to enforce coarse deny-lists and NAT.	Collapse thousands of allow rules to a catch-all "deny-unknown" stance; keep a firewall for routing and DDoS filters.
Next-Generation Firewalls (NGFWs)	Deep-packet inspection, URL filtering, sandboxing, threat-intel blocks.	**Augments:** NGFW still inspects east-west and outbound traffic after the SDP has	Feed user/device IDs from SDP into NGFW; disable duplicate

© Copyright 2026, Cloud Security Alliance. All rights reserved.

		authenticated and admitted the flow.	SSL-break-and-inspect on SDP-encrypted tunnels.
Intrusion Detection/Prevention (IDS/IPS)	Detect signatures or anomalies in network traffic.	**Shifts focus:** network IDS sees far less because SDP encrypts and pre-authorizes traffic; host-IDS becomes the primary signal.	Point SDP dropped-packet logs to the IDS/IPS or SIEM, so they keep visibility of rejected probes.
SIEM/SecOps Platforms	Aggregate, analyze, and alert on security event data.	**Feeds and is fed by SDP:** SDP sends high-fidelity "who-what-where" connection logs; SIEM risk scores can call the SDP API to quarantine users in real time.	Create bi-directional API hooks: SDP streams JSON logs; SIEM can invoke disconnect/policy-update actions.
Identity and Access Management (IAM)	Authenticate users/devices; store attributes and groups.	**Integrates:** SDP is an IAM policy-enforcer—it consumes claims (SAML, OIDC, LDAP) and enforces them at connection time	Map authoritative attributes (e.g., role, MAM status, geofence) into SDP policies; enable step-up MFA triggers.
Host/Container Agents (EDR, workload firewalls)	Endpoint visibility, process control, and micro-firewall.	**Complements:** SDP governs ingress/egress; host agents still watch local behavior and supply device-health signals.	Pass EDR risk scores to SDP posture checks; use SDP to cut the network when EDR flags ransomware.
Cloud-Native Controls (security groups, NSGs)	Coarse subnet/port ACLs around IaaS workloads.	**Overlays:** Retain as a back-stop but collapse to "deny all-unknown"; let SDP enforce fine-grain access.	Tag-to-policy sync; drive cloud-native ACLs from the same labels the SDP consumes ("env:prod", "app:pay-api").
NAC	Grant/deny switch-port or Wi-Fi access based on	**Feeds SDP** with real-time device-health scores; campus NAC still	Sync NAC trust score into SDP posture check; let SDP cut WAN access if NAC flags a rogue device.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

	MAC, 802.1X, and posture.	enforces layer-2 quarantine.	
CASB	Inline/API control for SaaS usage, DLP, and session sanitization.	**Complements:** CASB keeps content controls; SDP supplies stronger user/device identity.	Pass SDP user and device claims to CASB; avoid double proxying by letting CASB inspect traffic already admitted by SDP.
GRC Platform	Aggregate controls, evidence, and risk registers for auditors.	**Consumes SDP logs** as proof of least privilege and Zero Trust.	Export SDP policy changes and per-tunnel logs to GRC dashboards for continuous compliance reporting.
PKI/CA	Issue and rotate X.509 certificates for users, devices, and workloads.	**Optionally supplies crypto identity** for SDP mTLS and SPA handshakes.	Automate short-lived cert issuance (minutes to hours); embed CA root in SDP clients and gateways.
Load Balancers	Distribute L4/L7 traffic and provide basic health checks.	**Co-exists:** SDP tunnels can terminate in front of or behind the LB (if the SDP overlay cannot handle load balancing itself).	If behind, carry user identity in headers for downstream apps; if in front, treat LB VIP as the SDP resource.
EMM/MDM	Enforce mobile/endpoint configuration and compliance.	**Feeds device posture into SDP;** SDP enforces network cut-off for non-compliant phones/laptops.	Pull MDM compliance flag via API; block or restrict SDP access until the device is remediated.

Table 7: Legacy Controls and Their Relationship to SDP

SIEM: From Passive Log Sink to Active Policy Trigger

SDP gateways export a single, high-value record for every connection: user/device identity → service → time/location → decision. Feeding that straight into the SIEM collapses the puzzle of correlating disparate firewall, VPN, and endpoint logs. More importantly, the SIEM can now **push context back:** if it detects a surge of failed log-ins or threat-intel matches, an automated playbook calls the SDP API to drop the session or step up authentication. The result is a closed-loop, risk-adaptive control plane,

© Copyright 2026, Cloud Security Alliance. All rights reserved.

rather than yesterday's forensic repository.

Privacy note: SDP logs are rich in user metadata; mask PII as required by regulations and store hashes of identifiers if full-fidelity isn't mandatory.

Firewalls and NGFWs: Perimeter Rules Collapse, Inspection Remains

Because an SDP gateway will not even expose a listening port until identity and posture are validated, most "allow" rules on the edge firewall become redundant. What remains are high-level deny rules (e.g., known-bad IP ranges), anti-DDoS protections, and the NGFW's content-inspection functions. Best practice is to carry the SDP user/device identity into the NGFW via XF-Headers or a shared user-ID database so analysts can trace a packet from outer perimeter to workload in one click (or have an SDP solution that can do L7 inspection at ingress or egress so that you have a unified rule engine).

IDS/IPS: Fewer Alerts, Sharper Fidelity

Encrypted, mutually authenticated tunnels mean opportunistic scans never reach the workload stack; they are terminated at the SDP handshake. That slashes IDS noise. What remains are high-confidence alerts on misbehaving approved clients (detected by the host IDS) and on blocked connection attempts (logged by the SDP). Tune budgets accordingly, scale down network-IDS appliances and invest in host-based visibility and behavioral analytics.

VPN Retirement Playbook

- Remove split/full-tunnel profiles from endpoint managers
- Roll out the SDP client (or browser-based ZTNA) with per-app entitlements
- Use usage telemetry to identify the small subset of protocols that still require a full tunnel (e.g., legacy SMB printer admin); carve those out behind a small residual VPN pool, or, better yet, wrap them in an SDP connector

Outcome: Users experience faster SaaS access (no backhaul), SecOps achieves zero lateral reach, and infrastructure teams eliminate an internet-facing attack target.

Identity and Posture in the Policy Engine

SDP does not replace IAM; it enforces it. Map out which attributes drive the most granular decisions (e.g., `role=field-technician`, `device_state=managed`, `patch<=7days`). Feed them continuously to the SDP controller via SCIM or IdP APIs. The same attributes should flow to NGFW dynamic groups and cloud security-group tags so every tier speaks the same "language."

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Putting it Together

An SDP's value crystallizes when these elements **act as one system:**

1. **IAM** asserts identity
2. **SDP** authenticates, checks posture, and grants a micro-tunnel
3. **NGFW / host-IDS** (optionally) inspect the now-trusted traffic
4. **SIEM** records the event and, if risk spikes, instructs SDP to cut the tie

That loop replaces brittle, rule-sprawl perimeters with a living, context-aware fabric, without discarding proven investments in inspection and logging. Architects should aim for shared labels, identities, and API-level feedback across all tiers; then SDP becomes the policy spearpoint and every adjacent tool amplifies its reach.

SDP Business Benefits

SDP technologies offer a wide range of business benefits, including cost reduction, operational efficiency, risk management, compliance, cloud adoption, third-party access, and business innovation. The key benefits are summarized in Table 8: Overview of Business Benefits of SDP below.

Area	SDP Benefits
Cost and Labor Efficiency	• Eliminates or reduces reliance on traditional network security components, lowering licensing, maintenance, and support costs • Reduces or eliminates MPLS and leased-line expenses by reducing usage of private backbone networks • Simplifies policy management, reduces dependence on legacy security tools, and lowers operational overhead
User Experience and Workforce Productivity	• Secure access by eliminating traditional VPN client • Provides seamless, location-independent access for users • Enhances employee satisfaction and operational efficiency by providing faster, more reliable connectivity across devices and environments
Operational Agility	• Enables dynamic, policy-driven access management initiated by IT or IAM events • Improves IT responsiveness to business and security requirements • Accelerates service delivery and reduces operational delays

DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Governance, Risk, and Compliance (GRC) Enhancement	• Minimizes attack surfaces and prevents network- and application-based attacks • Increases access visibility and strengthens governance controls • Integrates with SIEM and GRC platforms to enhance compliance reporting, risk monitoring, and incident response
Compliance Scope and Cost Reduction	• Centralized access control from registered devices to specific applications or services • Improves auditability and traceability of access activities • Uses micro-segmentation to reduce the scope of regulated systems, lowering compliance and reporting costs
Secure Cloud and Hybrid Adoption	• Streamlines security architecture for hybrid, public, private cloud, and on-premises deployments • Reduces time-to-market and implementation complexity • Maintains or enhances security controls for cloud workloads
Third-Party and Partner Access	• Simplifies secure, limited access for external contractors, suppliers, and partners • Eliminates the need for VPNs or full network access for external parties • Reduces risk exposure and administrative overhead when working with third parties
Mergers, Acquisitions, and Divestitures (M&A)	• Provides secure, flexible access to rapidly connect or disconnect networks and users during M&A activities • Reduces operational risk and complexity during business transitions • Accelerates business integration or separation efforts

Table 8: Overview of Business Benefits of SDP

Sample Vendor Implementations

OpenZiti/NetFoundry

OpenZiti provides an enhanced alternative to SPA and other SDP techniques, enhancing security, reliability, and scalability. OpenZiti is pure open-source software that provides Zero Trust security and overlay networking. Commercially implemented (or productized) versions exist, such as NetFoundry.

At its core, OpenZiti is based on a fundamental shift in how secure connectivity is modeled. Traditional networking approaches, including SPA and SDP, are rooted in IP addresses. However, IP addresses are

© Copyright 2026, Cloud Security Alliance. All rights reserved.

neither secure identities nor effective proxies for them. They are not applications; they are difficult to manage at scale, and they frequently lead to over-permissioning, configuration errors, and operational fragility. Microsegmentation and Zero Trust architectures built on IP networks inherit these limitations.

OpenZiti replaces this model with an identity-first paradigm. Instead of centering connectivity on IP addresses, OpenZiti uses identities, services, and policies as its primary abstractions. Every participant - whether a human user, application workload, Kubernetes service, OT device, IoT sensor, or AI agent - is treated as a first-class identity. Connectivity is granted only when explicitly authorized by policy, independent of network location or topology. This allows heterogeneous environments, enterprise data centers, public clouds, edge sites, and mobile networks, to be unified into a single logical domain and segmented according to business or application intent rather than network boundaries.

This approach is implemented natively in software, not by automating IP-based assumptions. OpenZiti models what organizations actually want to protect: applications, services, and interactions. For backward compatibility, OpenZiti can also carry IP-based traffic for legacy systems while still enforcing stronger, more auditable identity-based authorization and policy controls than traditional network security mechanisms.

OpenZiti ensures that protected resources are invisible to unauthorized entities and accessible only to authenticated and authorized identities, providing a strong foundation for ZTA. It enforces an "authenticate-before-connect" model and relies exclusively on outbound-only connections to policy enforcement points. As a result, no inbound firewall ports are required at either the source or destination, eliminating exposure to external network attacks and removing the complexity of firewall ACLs, VPNs, MPLS, private APNs, port forwarding, and bastion hosts. Consistent with Zero Trust principles, OpenZiti incorporates strong cryptographic identity, continuous authorization, microsegmentation, least privilege access, mTLS, end-to-end encryption, device posture checks, private DNS, and more, making protected services secure by default.

OpenZiti Advantages

- **Identity-First Connectivity for Any Entity:** OpenZiti treats identity as the primitive for all connectivity, not just for human access. Humans, workloads, containers, OT devices, and AI agents all participate using the same identity and policy framework. This eliminates the traditional split between user-centric ZTNA and east–west or machine-to-machine networking, enabling consistent security across modern and legacy environments alike
- **Absolute Deny-by-Default and Zero Inbound Exposure:** Every OpenZiti connection is identity-aware, enabling detailed auditing, service usage tracking, health monitoring, and forensic analysis. Applications are decoupled from network topology and can be deployed or moved without changing DNS, firewall rules, or load balancers, simplifying operations and compliance in highly dynamic environments

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Policy-Based Connectivity Across Network Boundaries:** OpenZiti assumes the underlying network is untrusted, fragmented, and potentially hostile. It does not require shared address space, transitive trust, or network-layer continuity. Secure connectivity can be established across clouds, enterprises, partners, vendors, and sovereign boundaries using identity and policy alone. This makes OpenZiti particularly well suited for multi-domain architectures, supply chains, and emerging AI ecosystems
- **Application-Native Zero Trust Networking:** Zero Trust is most effective when enforced as close to the application as possible. OpenZiti supports embedding identity and policy enforcement directly into applications through lightweight SDKs, eliminating reliance on the operating system or network perimeter alone. Where embedding is not feasible, host-based tunnelers or gateways provide equivalent protection. This flexibility supports cloud-native microservices, DevOps workflows, and agent-based systems
- **Strong Cryptographic Security by Default:** OpenZiti manages the full lifecycle of cryptographic identity and trust establishment. It uses x509 and JWTs for strong internal identity and optionally integrates external identity providers (following RFC 7030) via third-partyCAs, external JWT signers, and more. All connections are protected with mTLS and end-to-end encryption by default, ensuring confidentiality and integrity while minimizing operational overhead. Metadata and traffic patterns are also protected, preventing inference of service type, port usage, or network topology
- **Operational Visibility and Portability:** Every OpenZiti connection is identity-aware, enabling detailed auditing, service usage tracking, health monitoring, and forensic analysis. Applications are decoupled from network topology and can be deployed or moved without changing DNS, firewall rules, or load balancers, simplifying operations and compliance in highly dynamic systems

For more details about the OpenZiti protocol, see Appendix 4: OpenZiti/NetFoundry Details.

Network-Infrastructure Hiding Protocol (NHP)

NHP is a lightweight, cryptography-driven Zero Trust networking protocol at the OSI 5th layer (session), offering an enhanced alternative to SPA. Its improved architecture boosts security, reliability, and scalability. Beyond its role as an SDP component, NHP can also serve as a versatile "authenticate-before-connect" secure handshake protocol for various user scenarios outside SDP.

For detailed information about the NHP architecture and workflow, refer to Appendix 3: NHP Details.

NHP Advantages

The advantages of NHP are:

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Security:** As discussed, the SPA packet may be captured or altered during network transmission, posing a potential MITM risk. Some SPA implementations utilize PKI/RSA to mitigate this problem; however, the computational cost of RSA signature verification and the complexity of PKI key distribution render this approach too costly. In NHP, the modern Noise Protocol Framework is introduced for message encryption and mutual verification, providing a proven, secure, and fast solution. Instead of PKI, NHP introduces identity-based cryptography (IBC) to simplify key management and reduce the cost of key distribution
- **Reliability:** The client sending the SPA packet is unaware of the outcome of SPA authentication. Consequently, it must repeatedly attempt to establish a connection to the protected server (Controller or Gateway) at arbitrary intervals after sending the SPA packet. This approach relies heavily on network speed and may fail under poor network conditions, resulting in unreliable performance across different network environments. In NHP, if authentication succeeds, the client receives a confirmation message containing the protected server's port status and opening time, allowing it to know precisely when to establish the connection. If the authentication fails, NHP maintains a "silent" mechanism, and the client receives no response
- **Scalability:** The SPA server is typically deployed on protected servers (Gateway or Controller) and provides two major features: 1) SPA packet decryption and user/device authentication, and 2) network access control. In scenarios with many concurrent clients, the SPA packet decryption and user/device authentication processes can consume significant bandwidth and computing resources on the hosted servers, thereby limiting the service's scalability. In NHP, these two features are decoupled into separate components that can be deployed independently across a cluster of servers, enabling high scalability
- **Extensibility:** While the benefits of SPA apply to a broad range of network services, SPA was not designed to be an extensible protocol supporting various user scenarios beyond SDP. NHP, on the other hand, is designed as a general-purpose connection management protocol for the 5th layer of the network. In addition to hiding server ports, it can hide the IP address by providing private domain name resolution. It supports any scenario that requires “authenticate-before-connect” secure communications

For more information about NHP, see Appendix 3: NHP Details.

Bring Your Own Security (BYOS)

BYOS delivers Zero Trust network access through physical isolation devices that create individual network microsegments at the endpoint level, positioning between the protected asset and the untrusted network.

Architecture Overview

The **BYOS Secure Networking Platform** consists of three primary components:

v3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **BYOS Secure Edge™:** Physical hardware devices deployed at network edges that enforce isolation and access control, available in multiple form factors from USB-C dongles to rack-mountable appliances
- **BYOS Secure Lobby™:** Software-defined overlay network (cloud or customer-hosted) creating encrypted tunnels between authenticated edges through outbound-only Layer 2 TLS1.3 + AES512 connections
- **BYOS Management Console™:** Centralized control plane for policy definition, fleet management, and identity mapping across all deployed edges

Core Technology

BYOS implements hardware-enforced microsegmentation at the physical layer. Each BYOS device acts as a gateway, a DHCP server, and a DNS server for connected assets, creating a logical microsegment of one. Protected devices become invisible and unreachable, the asset sees only the BYOS device on its local network segment, and all inbound connection attempts are dropped silently at the hardware layer. Administrators define which traffic routes through the Secure Lobby overlay, the local network, or the internet. Hardware-enforced traffic filtering at OSI layers 1-4 prevents ARP spoofing, DNS hijacking, man-in-the-middle attacks, rogue DHCP servers, enumeration, lateral movement, and other network-based attacks.

Hardware Security Architecture

BYOS devices implement defense-in-depth at the physical level through a hardened OS, a custom recompiled kernel, and signed proprietary binaries.

The critical architectural innovation is the **Protocol Break**: traffic entering via Ethernet passes through a System-on-Module, undergoes quantum-resistant hardware encryption, traverses the Layer 2 stack, and exits via physically separate Layer 1 communications hardware. This split-interface architecture creates an air gap at the silicon level between the protected asset and the network. Even with complete OS compromise of the protected asset, attackers cannot cross this hardware boundary to manipulate network traffic, disable security policies, or communicate directly with the network stack.

Integration with Existing Infrastructure

BYOS complements existing security controls through comprehensive integration:

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Identity and Authorization:** OIDC/SAML federation for user authentication, MFA for step-up authentication, and hardware-enforced authentication for non-human identities and legacy systems. Hardware-bound authentication is unspoofable and unphishable
- **Security Operations:** SIEM/SOAR integration exports comprehensive connection logs, including identity, device fingerprint, source/destination, protocol, and disposition. Works alongside vulnerability scanners, threat-detection platforms, asset-management systems, and network-monitoring tools without requiring infrastructure replacement
- **Zero Trust Ecosystem:** Compatible with ZTNA/SASE solutions either as first-hop security or as secure endpoints

BYOS Advantages

- **Hardware-Enforced Isolation:** Software-based segmentation relies on the protected device's operating system to enforce policy. A compromised OS can disable or bypass software controls. BYOS physically separates enforcement (a hardware device) from the protected asset (the operating system), making bypass impossible even with a full OS compromise
- **Legacy and Unmanaged Device Support:** OT devices, medical equipment, IoT sensors, and legacy systems often cannot run security agents due to incompatible operating systems, resource constraints, vendor-locked firmware, or certification requirements. BYOS requires zero software installation on protected assets, enabling security for devices that are otherwise unprotectable
- **Attack Surface Reduction:** BYOS eliminates traditional network perimeters and DMZs. All BYOS Edges establish outbound-only tunnels to the Secure Lobby with zero inbound listening ports. East-west traffic between assets is emulated as north-south traffic within the Layer 2-encrypted tunnel. All "inbound" connections occur within the outbound tunnel established by the destination Edge device. Protected assets become invisible to reconnaissance tools because there is no network path to reach them, eliminating zero-day exploitation vectors and removing the need for traditional perimeter security infrastructure
- **Ransomware Blast Radius Containment:** BYOS controls the entire network stack, not the protected asset. The asset has no visibility into network topology, routing, or other devices. If ransomware achieves a complete system takeover, it cannot propagate because the compromised asset does not know where other assets exist or how to reach them. Even with complete OS control, the attacker cannot bypass hardware-enforced isolation or manipulate the network stack on physically separate hardware
- **Differentiation from Software-Based SDP:** BYOS addresses the fundamental limitation of software-based SDP: reliance on the endpoint operating system for security enforcement. When the OS is compromised, software agents can be disabled, policies can be bypassed, and security controls can be subverted. By moving enforcement to a separate physical device with its own hardened OS and protocol break architecture, BYOS ensures that security remains intact even when the protected asset is fully compromised. This makes BYOS particularly suited for high-assurance environments, legacy OT systems, and scenarios where agent installation is impossible or prohibited

© Copyright 2026, Cloud Security Alliance. All rights reserved.

AppGate

AppGate ZTNA delivers highly secure, highly performant access for all users, devices, and workloads—regardless of location. It enforces Zero Trust principles by dynamically provisioning encrypted, point-to-point connections between authenticated entities and authorized resources. Unlike cloud-routed architectures that rely on intermediaries, AppGate's direct-routed approach creates isolated network segments on demand, minimizing latency, eliminating unnecessary routing, and preventing unauthorized access to assets.

AppGate integrates multiple authentication mechanisms—including SPA, mTLS, and identity-based policy evaluation—to enable fine-grained, identity-centric access control. This unified, distributed architecture cloaks infrastructure, continuously validates user and device trust, and enforces least privilege access across hybrid, multi-cloud, and on-premises environments.

Addressing the Limitations of SPA

SPA delivers valuable "authenticate-before-connect" protection, cloaking resources, and reducing reconnaissance risk. However, SPA alone cannot address the full complexity of modern enterprise environments, which require a broader context that includes identity, device posture, and dynamic conditions.

AppGate addresses these gaps by embedding SPA into a holistic Zero Trust model. Instead of relying solely on pre-authentication packets, AppGate continuously evaluates identity signals, device health, environmental factors, and risk insights from EDR, MDM, and identity systems. Even if an SPA packet were intercepted, an attacker cannot initiate or maintain a session without meeting AppGate's continuous trust requirements.

This shifts control from static packet filtering to adaptive, identity-driven access—combining SPA's stealth benefits with modern Zero Trust intelligence.

Alignment to Zero Trust

AppGate ZTNA aligns directly with Zero Trust principles defined by NIST 800-207 and the CSA's SDP architecture. Its identity-centric policy engine and point-to-point tunnels ensure no network access is granted until identity, device posture, and contextual risk are verified.

Key elements include:

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Identity-Centric Policy Enforcement:** Access decisions adapt to user identity, device health, and environmental context rather than static IP constructs
- **Mutual Authentication and Encryption:** SPA and mTLS verify both ends before a connection is established
- **Dynamic Entitlement Resolution:** Contextual metadata drives real-time assignment or removal of access
- **Micro-Segmented Connectivity:** Each entitlement creates an isolated "segment of one" to prevent lateral movement

AppGate Advantages

- **Identity-Centric Policy Enforcement:** AppGate uniquely identifies every user, device, and workload through credentials, device posture, geolocation, and risk signals from integrated systems. Policies are generated in real time, ensuring that only properly validated entities can access specific resources. This eliminates overprivileged access and significantly reduces the risk of lateral movement
- **Cloaked Infrastructure and Attack Surface Reduction:** AppGate extends SPA with cryptographic validation, replay protection, and timestamp controls. Even if an SPA packet is intercepted, attackers cannot connect without mTLS validation and policy approval. Gateways accept only outbound-initiated connections, protecting them from unsolicited traffic and volumetric attacks. This layered approach keeps infrastructure invisible while enforcing continuous Zero Trust controls
- **Direct-Routed Architecture for Superior Performance:** AppGate avoids the performance penalties of cloud-routed ZTNA by establishing direct, software-defined tunnels between authenticated users and authorized resources. Eliminating unnecessary routing hops preserves bandwidth, minimizes latency, and supports demanding workloads, including development pipelines, remote operations, and industrial control systems
- **Outbound-Only, Resource-Specific Connectivity:** Gateways expose no inbound listening ports, removing common attack vectors. Access is permitted only to explicitly authorized resources, creating isolated microsegments without the need for complex VLAN or subnet management
- **Dynamic Entitlement and Continuous Validation:** Access adjusts automatically as context changes. Degraded device posture, expired MFA, or increased risk triggers immediate entitlement updates. Integrations with existing security telemetry allow organizations to automate policy enforcement without rebuilding identity frameworks
- **Seamless Interoperability with Existing Systems:** AppGate integrates with identity providers, SIEMs, endpoint protection tools, and network security platforms. Organizations can deploy quickly and extend Zero Trust policies across hybrid and multi-cloud environments using flexible APIs and automation workflows
- **Scalability and Deployment Flexibility:** Deployable on-premises, in any cloud, or as a hybrid, AppGate scales horizontally to support global enterprises. Its distributed control plane ensures low-latency enforcement and maintains access even during WAN issues or controller failover

V5 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Compliance and Audit Support:** Identity-mapped logs and isolated resource access simplify compliance with standards such as NIST 800-207, ISO 27001, PCI DSS, HIPAA, and IEC 62443
- **Reduced Operational Complexity:** Policy-based access replaces static firewalls and VPN rules, reducing manual overhead and accelerating onboarding
- **Business Continuity and Resilience:** Distributed enforcement, multi-region identity support, and high availability ensure secure, uninterrupted operations during disruptions

Differentiation from SPA and Other Approaches

AppGate unifies SPA's cloaking strengths with adaptive, identity-driven Zero Trust controls and a direct-routed architecture. Unlike cloud-routed ZTNA that forces traffic through vendor gateways, AppGate delivers point-to-point connectivity for superior performance and lower risk. Its outbound-only, resource-specific connections eliminate inbound exposure and prevent discovery, delivering a highly secure, highly performant Zero Trust access model at enterprise scale.

Conclusion and Future Outlook

Conclusion

Over the last few years, SDP has matured from a "remote-access hardening" pattern into a pluggable Zero Trust connectivity architecture, which is part of the reason we felt compelled to create a v3.0 update to this document. SDP cleanly separates control and data planes, centers decisions on identity and posture, and enforces "authenticate-before-connect" across multiple methods, including SPA, the NHP, and overlay/mesh approaches such as OpenZiti. These options expand implementers' choices while preserving the same policy model of Initiating Hosts, Accepting Hosts, and a Controller/Policy Engine, and provide both commercial and open-source options.

The practical outcome is consistent: invisible infrastructure, mutual authentication (mTLS/certs), least privilege micro-tunnels, and dynamic, application-centric policy, with every permitted connection logged at high fidelity. This replaces brittle, perimeter-centric rules with a living, context-aware fabric that also integrates with existing inspection and logging investments through closed-loop automation.

As a result, SDP can now span across diverse deployment models (client↔gateway, client↔server, server↔server, and mixed topologies) and use-case domains (hybrid/multi-cloud, OT/industrial, and edge/IoT fleets). It does so without forcing wholesale network changes, reducing attack surface and lateral movement while improving resilience and user experience. This has caused the lines between microsegmentation solutions and SDP/ZTNA to overlap increasingly; both aim to enforce least privilege, identity-centric policy at the connection or session boundary. The distinction is often less about goals and more about where enforcement is applied (SDP at connection setup, microsegmentation inside

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

established networks). Together, they can be viewed as complementary layers of a converging Zero Trust security fabric.

Beyond security outcomes, organizations adopt SDP for material business benefits, including lower costs and operational overhead, faster service delivery, simpler third-party access, improved auditability, and reduced compliance scope, while aligning with Zero Trust and Secure by Design objectives.

This guide has demonstrated how to map transaction flows, select the appropriate enforcement method (e.g., SPA, NHP, overlay), and integrate SDP with IAM, SIEM, EDR, and cloud-native controls to form a cohesive, risk-adaptive system. The conclusion is straightforward: SDP remains the most viable and future-proof method for operationalizing Zero Trust at connection time.

We don't claim to have all the answers. We invite you to join the CSA working group, share operational lessons, and help drive the next round of patterns and proofs.

Future Outlook

As organizations advance their Zero Trust journey, the role of SDP will continue to evolve. What began as a focused mechanism for cloaking services and enforcing authentication before connection is now expanding into a broader connectivity fabric that bridges IT, OT, cloud, and edge environments. The future of SDP lies not only in addressing today's access challenges but also in converging with adjacent Zero Trust technologies, embedding directly into applications and platforms, and adapting dynamically to business and threat contexts.

Here are a few areas we believe it will continue to adapt and evolve:

- **Convergence of Zero Trust Approaches:** SDP, microsegmentation, and broader ZTNA patterns are beginning to blur. We expect to see a single, unified policy fabric that spans users, devices, apps, workloads, OT/IoT, and cloud platforms, reducing silos and simplifying governance, particularly as cross-cutting topics such as AI/ML permeate the enterprise
- **Policy and Enforcement Convergence:** As Zero Trust architectures mature, the convergence of SDP, ZTNA, and microsegmentation is increasingly enabled by shared, programmable policy models and distributed enforcement mechanisms. Technologies such as policy-as-code frameworks (e.g., Open Policy Agent) and lower-level enforcement mechanisms (e.g., eBPF) enable identity-aware, context-driven intent to be defined once and enforced consistently across the control and data planes. Some existing SDP implementations, including those built on OpenZiti/NetFoundry, already demonstrate this approach by combining identity-centric access, application-level segmentation, and distributed enforcement within a unified policy fabric
- **Expansion into New Domains:** SDP enforcement will extend beyond traditional apps and VMs into PaaS services, serverless functions, containers, Kubernetes clusters, IoT fleets, and OT/industrial systems, creating an identity-first foundation across heterogeneous environments

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Embedded, Secure by Design Connectivity:** SDP capabilities will increasingly be embedded into applications, developer frameworks, and products. Instead of bolt-on controls, identity-aware connectivity becomes integral to the software itself, shifting Zero Trust left in the lifecycle
- **Dynamic, Risk-Adaptive Policy:** The integration of real-time risk signals (from EDR, SIEM, threat intelligence, UEBA, and posture assessment) into the SDP control plane enables connections to adapt continuously to context and threat level
- **Automated Mapping and Verification:** AI- and ML-driven discovery of flows, along with policy-as-code pipelines, will facilitate the translation of business intent into SDP policies, continuous validation, and compliance auditing
- **Resilience and Decentralization:** Expect controller/fabric designs that reduce central chokepoints, using distributed consensus and self-healing overlays to improve reliability, support disconnected OT/edge use cases, and withstand large-scale disruptions. These resilient, distributed, smart-routing fabrics also maintain performance and availability while removing the need for traditional SD-WAN appliances, L4 load balancers, and other legacy intermediaries
- **Standardization and Interoperability:** Growing demand for cross-vendor interoperability and open standards (for policy expression, enforcement APIs, and identity claims) will push SDP out of proprietary silos and into an ecosystem where enterprises can mix and match

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Glossary

- CSA Glossary (main/primary)
- CSA SDP Glossary (Software Defined Perimeter)

Hair-Pin Latency: The additional one-way (sometimes quoted round-trip) delay that traffic accumulates when it is forced to exit a local segment or device, traverse an intermediate interface or gateway, and then loop straight back toward a destination that is still on - or very close to - the same side of the network or chassis. The interval is measured from the instant the signal enters the detour point to the instant it re-emerges headed home.

Useful References

Cloud Security Alliance (CSA). (2002). Integrating SDP and DNS: Enhanced Zero Trust Policy Enforcement

Cloud Security Alliance (CSA). (2002). Software-Defined Perimeter (SDP) Specification v2.0

Cloud Security Alliance (CSA). (2017). SDP for IaaS

Cloud Security Alliance (CSA). (2018). Software Defined Perimeter Glossary

Cloud Security Alliance (CSA). (2019). Software-Defined Perimeter as a DDoS Prevention Mechanism

Cloud Security Alliance (CSA). (2019). SDP Architecture Guide v2.0

NIST. (2020). NIST SP 800-207, Zero Trust Architecture

NIST. (2025). NIST SP 1800-35 Implementing a Zero Trust Architecture

NIST. (2025). NIST SP 800-215, Guide to a Secure Enterprise Network Landscape

Department of Defense (DoD). (2022). Zero Trust Reference Architecture, Version 2.0

Department of Defense (DoD). (2022). Cloud Native Access Point (CNAP) Reference Design (RD) Version 1.0

Mandiant. (2024). mWISE Conference Keynote Speaker - Jen Easterly, Director, Cybersecurity and Infrastructure Security Agency (CISA), *"We don't need more security products, we need more secure products"* (@16:11 of the video)

Conde, Daniel. (2017). Software-Defined Perimeters: An Architectural View of SDP. *IEEE*

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

OpenZiti. (2026). Open Source Zero Trust Network Overlay

OpenZiti & CSA. (2024). Zero Trust Networking for Complex Use Cases—CI/OT/IoT, Air-Gapped Networks and More (OpenZiti presentation at the Cloud Security Alliance)

OpenZiti & CSA. (2023). Open Source Zero Trust Networking and SDP with OpenZiti (OpenZiti presentation for the Cloud Security Alliance)

CipherDyne. (2024). Open Source SPA implementation- fwknop - fwknop-2.6.11

OpenNHP. (2026). Open Source NHP Implementation

NoiseProtocol.org. (2026). Noise Protocol Framework

Wikipedia. (2026). Identity-based cryptography (IBC)

GEEKFLARE. (2024, 21 December). 9 Best Software-Defined Perimeter Solutions for 2024

Comparitech. (2024, 12 December). The Best Software Defined Perimeter (SDP) Solutions

Packet Pushers. (2025, 23 September). Packet Protector, Ep79: Rethinking the Architecture of Microsegmentation

Hackathon References

Cloud Security Alliance (CSA). (2014, 22 October). Press Release 2014/10/22: Software hackathon closed, software-defined-perimeter prevails again

Cloud Security Alliance (CSA). (2014, 27 August). Press Release 2014/08/27: Hackathon on! Cloud Security Alliance challenges hackers to break its software-defined-perimeter SDP at CSA Congress 2014

https://www.prnewswire.com/news-releases/cloud-security-alliance-to-host-third-software-defined-perimeter-sdp-hackathon-top-prize-of-10000-available-300062025.html

Cloud Security Alliance (CSA). (2016, 31 March). Press Release 2016/03/31: Cloud Security Alliance releases results of software-defined-perimeter hackathon

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Appendix 1: SDP Deployment Models

SDP v3.0 supports multiple deployment models to accommodate diverse enterprise environments, system types, and operational constraints. These models build on earlier SDP concepts while extending them to support non-human identities, cross-domain systems, and dynamic, software-defined architectures.

Unlike earlier SDP approaches that were tightly coupled to SPA workflows or gateway-centric enforcement, SDP v3.0 defines deployment models in terms of identity roles and policy enforcement placement, rather than specific protocols or network topologies.

Across all SDP v3.0 deployment models:

- Access is deny-by-default
- Authentication and authorization occur before connectivity is established
- Resources expose no inbound network surface
- Enforcement occurs at a Policy Enforcement Point (PEP) appropriate to the environment
- The underlying network is treated as untrusted

In all deployment models, policy decisions are centralized in the SDP Controller, while enforcement occurs at the Gateway or Accepting Host.

SDP Deployment Model	Description	Example Use Case
Initiator-to-Gateway	An Initiating Entity (IE) connects to an SDP Gateway acting as a PEP in front of one or more protected resources. The gateway authenticates the initiator, enforces policy, and brokers access without exposing backend systems.	Secure access to legacy or mixed on-prem applications; environments where protected systems cannot host SDP components.
Initiator-to-Resource	The IE connects directly to a protected resource hosting its own SDP enforcement component. Connectivity is established only after identity and policy validation, without intermediate gateways.	Developer access to cloud services; direct workload or agent access to APIs or databases.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Resource-to-Resource	Two protected resources communicate using mutual authentication and policy enforcement at both ends. Policies constrain which resources may communicate and under what conditions.	Microservices communication; data replication; internal APIs; AI pipeline stages
Initiator-to-Resource-to-Initiator	Two initiating entities communicate through an SDP-protected rendezvous or mediation service. Network locations remain hidden, and access is granted only after both identities are authorized.	Secure collaboration, remote support, peer coordination, agent-to-agent communication
Initiator-to-Gateway-to-Initiator	An SDP Gateway mediates peer connectivity between two initiating entities, enforcing policy and enabling secure peer-to-peer communication without exposing either endpoint.	Cross-organization collaboration; users on restricted or NATed networks
Gateway-to-Gateway	SDP Gateways at different sites establish policy-controlled connectivity between groups of resources. Only explicitly authorized identities, services, or flows are exposed.	Site-to-site connectivity; OT-to-IT integration; data center to cloud connectivity

Table A1.1: SDP Deployment Models

Enforcement Placement in SDP v3.0

In SDP v3.0, policy enforcement placement is flexible and does not define the trust model. Enforcement may be implemented as:

- **Application-embedded components** for cloud-native services, agents, and APIs
- **Host-based agents or sidecars** for workloads and endpoints
- **Gateways** for aggregation points or legacy systems
- **Hardware-enforced edges** for unmanaged or constrained devices

© Copyright 2026, Cloud Security Alliance. All rights reserved.

The same SDP v3.0 deployment model may be realized using different enforcement placements depending on system capabilities and operational requirements.

Key Characteristics of SDP v3.0 Models

All SDP v3.0 deployment models share the following characteristics:

- Identity-based access control independent of IP addressing
- Explicit authorization per resource or interaction
- Zero inbound exposure and outbound-only connectivity
- Mutual authentication and encrypted communications
- Microsegmentation by policy, limiting lateral movement
- Auditability based on identity, policy decision, and outcome

These deployment models preserve the architectural flexibility of SDP while extending it to support modern enterprise environments, including cloud-native applications, OT systems, multi-domain architectures, and emerging agent-based workloads.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Appendix 2: SPA Details

Below is the format of the SPA packet as defined in the SDP v2.0 specification.

Ciphertext	**Nonce**	16-byte random data field prevents replay attack
Ciphertext	**Timestamp**	Prevents servicing outdated SPA packets
Ciphertext	**Message Type**	Most common: Service Access Request Possibly deprecated: access request, NAT access request, gateway command message.
Ciphertext	**Message String**	Source IP address to allow service ID(s) to open Note: Gateway knows which port to open and, if applicable, where to forward the connection.
Ciphertext	**Optional Fields**	Might be used to request tunneling of service traffic.
Ciphertext	**Digest**	Before encryption, this SHA256 hash is calculated over the ciphertext portion of the message and then used by the server to verify message integrity after a successful message decryption.
Cleartext	**HMAC**	Calculated over all fields above, the algorithm choices are MD5, SHA1, SHA256 (default), SHA384, and SHA512.

Table A2.1: SPA Packet Format

A suggested improvement to the above is the addition of a clear-text Client ID, which would enable more efficient handling of incoming packets. An effort has been made to design a binary SPA format and to create an RFC document describing it. These efforts are still ongoing.

SPA is most effective when sent as a single UDP packet. However, this is impractical in some use cases, as network environments may block some or all outgoing UDP packets. In such cases, SPA packets can be sent over TCP connections. This technically violates SPA's “single packet” nature but is sometimes necessary as a practical consideration.

The connecting machine can send an SPA packet, but another device can as well. An example is when a mobile device sends an SPA packet on behalf of a desktop computer. In some scenarios, this is also a reasonable workaround for an environment that blocks UDP packets.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Appendix 3: NHP Details

NHP: Zero Trust Implementation at the OSI Session Layer

The Network Infrastructure Hiding Protocol (NHP) is designed as a session-layer (the 5th layer) protocol in the OSI network model to implement Zero Trust principles. According to the OSI model, the session layer is responsible for establishing connections and controlling dialogue. Implementing Zero Trust at the session layer offers significant benefits:

- **Mitigate Vulnerability Risk:** The openness of TCP/IP protocols leads to a "trust by default" connection model, allowing anyone to connect to a server port that provides services. Attackers exploit this openness to target server vulnerabilities. The NHP protocol implements the Zero Trust principle "never trust, always verify" by enforcing "deny-all" rules on the server side by default, allowing only authorized hosts to establish connections. This effectively mitigates the exploitation of vulnerabilities, particularly zero-day exploits
- **Mitigate Phishing Attacks:** DNS hijacking poses a serious threat to internet security and is used to facilitate phishing, steal sensitive information, or spread malware. The NHP can function as an encrypted DNS resolution service to mitigate this problem. When the NHP-Agent on the client side sends a knock request to the NHP-Server component with the identifier (e.g., the domain name) of the protected resource, the NHP-Server returns the IP address and port number of the protected resource if the NHP-Agent is successfully authenticated. Since NHP communication is encrypted and mutually verified, the risk of DNS hijacking is effectively mitigated
- **Mitigate DDoS Attacks:** As mentioned above, a client cannot obtain the IP address and port number of protected resources without authentication. If the protected resources are distributed across multiple locations, the NHP server may return different IP addresses to different clients, making DDoS attacks significantly more difficult and expensive
- **Attack Attribution:** The TCP/IP protocol suite is IP-based. With NHP, the connection model becomes identity-based (ID). The identity of the connection initiator must be authenticated before the connection is established, making attacks much more auditable

NHP has been implemented as open-source software named OpenNHP. It can be used as a stand-alone solution to protect network resources (such as devices, servers, and data) or as an embedded component/library in a multi-layer Zero Trust product, such as SDP. The integration of NHP and SDP is detailed in a later section of this chapter.

V5 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

NHP Architecture and Components

The NIST ZTA standard serves as the inspiration for the NHP architecture. Its goal is to ensure that protected resources remain invisible to unauthorized entities and only accessible to authorized subjects through continuous verification. The architecture leverages modern cryptographic algorithms and frameworks to ensure security and trustworthiness against sophisticated attacks. Unlike SPA, the NHP architecture decouples authentication and access control, improving performance and scalability. The NHP architecture is an extensible framework that allows seamless integration with SDP for SPA replacement and interoperability with other protocols (e.g., DNS, FIDO, OAuth). The diagram below illustrates the core components of the NHP architecture, along with other interacting components.

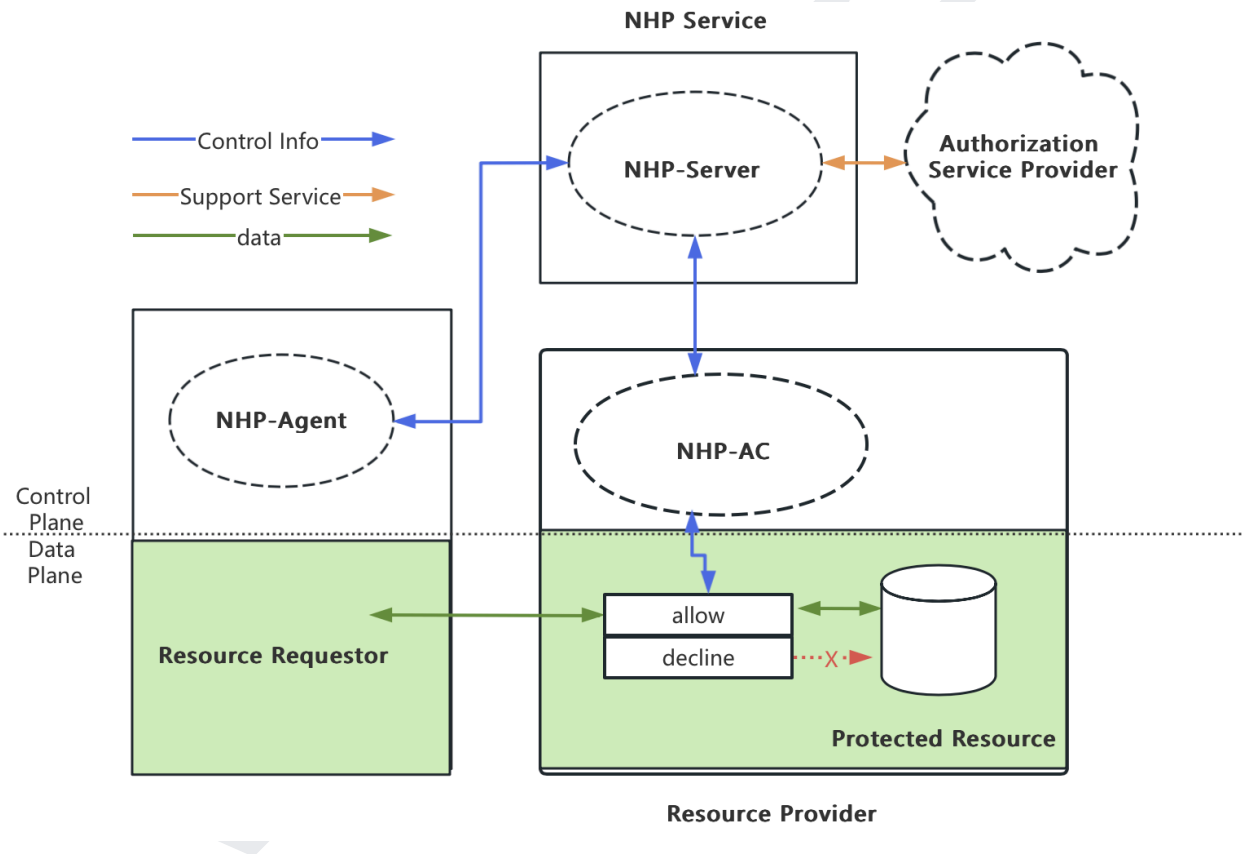


Figure A3.1: Core Components of NHP Architecture

NHP Core Components

- **NHP-Agent:** A client-side component that initiates communication and requests access to protected resources by sending knock requests to the NHP-Server component. It can exist as a client, SDK, browser, application, or other similar entity

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **NHP-Server:** A server-side component that processes and validates knock requests, performs authentication and security policy validation, and manages the NHP-AC components to dynamically grant access to authorized NHP-Agents.
- **NHP-AC:** This module enforces access policies and ensures the network invisibility of protected resources, typically a server program or application. It enforces the default deny-all security policy and ensures the protected resource's network invisibility, usually residing on the same host as the protected resource. It is responsible for granting access to authorized NHP-Agents, denying access to unauthorized NHP-Agents, and performing actions based on commands returned by the NHP-Server

Components That Interact with NHP

- **Protected Resources:** The resource provider is responsible for protecting these resources, including API interfaces, application servers, gateways, routers, and network devices. In the SDP scenario, the protected resources are the SDP Gateway and Controller
- **Authorization Service Provider (ASP):** This provider validates access policies and provides the actual access addresses of protected resources. In the SDP scenario, the ASP may be the SDP Controller

NHP Workflow

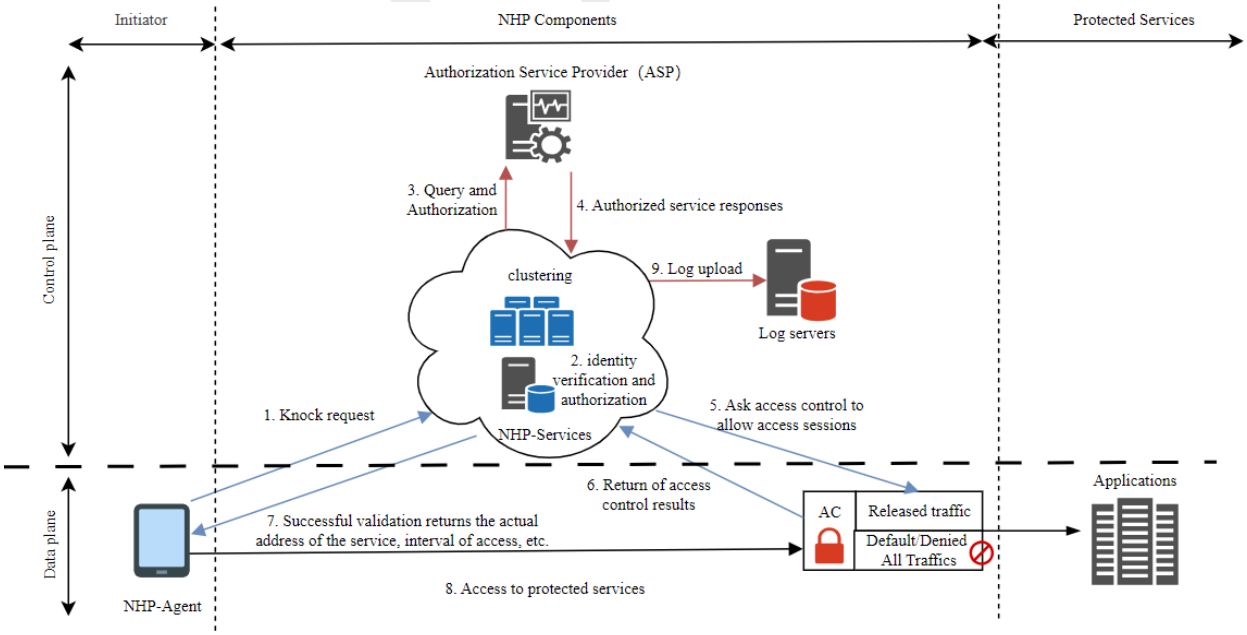


Figure A3.2: NHP Workflow

© Copyright 2026, Cloud Security Alliance. All rights reserved.

1. Before the NHP-Agent accesses the protected resource, a knock request is sent to the NHP-Server.
2. The NHP-Server processes the request and retrieves the NHP-Agent's information, which includes the NHP-Agent's identity, device information, the ASP requesting access, and the target resource information.
3. The NHP-Server aggregates the agent information, locates the corresponding ASP server (usually the IAM system) based on the ASP information, and sends a query.
4. The ASP server performs authentication and authorization based on the agent's information, authorizes the target resources the NHP-Agent can access, and replies with the target resource's actual IP address and port, along with other authorization information (e.g., a token).
5. If the NHP-Server query in step 3 is successful, the NHP-Server will send an open-door request to the target resource's corresponding NHP-AC.
6. Upon receiving the NHP-Server's request, the NHP-AC must verify that the requested resource matches the protected resource. It then opens a connection channel from the NHP-Agent to the protected resource and returns the validity period of the open-door session to the NHP-Server.
7. Once the NHP-Server confirms the open-door session succeeded, it responds to the NHP-Agent with the protected resource's actual address and the session duration.
8. From this point, the NHP-Agent can access the requested resource. Before this open-door session, the NHP-Agent must initiate another knock request and repeat steps 1-7 to renew access to the requested resource.
9. The logs generated in the preceding steps are uploaded to the regulator for compliance auditing. (Optional)

Cryptographic Algorithms and Frameworks

Cryptographic algorithms are fundamental to the NHP protocol, ensuring security and trustworthiness. The key algorithms utilized in NHP are listed below:

- **Elliptic Curve Cryptography (ECC):** ECC is a public-key cryptography method renowned for its strong security and excellent performance. For the same level of security, ECC's key length is significantly shorter than RSA's. When performing identity verification for NHP messages, transmitting a random ECC key (32 or 64 bytes) for ECDH exchange incurs lower computational overhead and consumes less bandwidth than transmitting an RSA2048 message signature (256 bytes) for signature verification. As ECC continues to gain popularity, TLS v1.3 now supports ECC, further enhancing its adoption in secure communications
- **Noise Protocol Framework:** The Noise Protocol is a key exchange protocol that combines random ECC key pairs with the ECDH algorithm. The framework implicitly implements identity verification for both parties and generates a symmetric key for message encryption and decryption. Introducing a random key pair in each interaction is particularly suitable for secure applications, such as NHP, which involve rotating encryption and decryption keys for every

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

session. The Noise Protocol has been proven to be extraordinarily secure and fast in popular applications such as WhatsApp and WireGuard

- **Identity-Based Cryptography (IBC):** IBC represents a new paradigm in cryptography. Although PKI key distribution systems have been widely used, the increasing number of users and devices in Zero Trust security architectures makes the key distribution process for each user device and subsequent management processes increasingly complex. Traditional PKI-based key distribution and verification systems are unlikely to meet the high requirements for accessibility, robustness, and performance in Zero Trust security architectures. In contrast, IBC can address this issue by simplifying the key distribution and verification process. The IBC algorithm uses a standard public key matrix, allowing easily readable character combinations to serve as unique identifiers in place of public keys for transmission and computation. This significantly reduces the complexity of the key center's public key management and distribution processes

Comparison Between SPA and NHP

	SPA	NHP	NHP Advantages
Architecture	The SPA packet-decryption and user/device-authentication component is coupled with the network access control component on the SPA server.	The NHP-Server (packet-decryption and user/device-authentication component) and NHP-AC (access-control component) are decoupled. NHP-Server can be deployed on separate hosts and supports horizontal scaling.	• **Performance:** The resource-consuming component, NHP-server, is separated from the protected server • **Scalability:** NHP-Server can be deployed in distributed or clustered mode • **Security:** The IP address of the protected server is not visible to the client unless the authentication succeeds
Communication	Single direction	Bi-direction	Better reliability with the status notification of access control.
Cryptographic Framework	PKI/Shared Secrets	IBC + Noise Framework	• **Security:** Proven secure key exchange mechanism to mitigate the MITM threats • **Low cost:** Efficient key distribution for a Zero Trust model • **Performance:** High-performance

© Copyright 2026, Cloud Security Alliance. All rights reserved.

			encryption/decryption
Capability of Hiding Network Infrastructure	Only server ports	Domain, IP, and ports	Enhanced defense against various attack vectors (e.g., vulnerabilities, DNS hijack, and DDoS attacks).
Extensibility	No, only for SDP	All-purpose	Support any scenario that needs service darkening.
Interoperability	Not available	Customizable	NHP can seamlessly integrate with existing protocols (e.g., DNS, FIDO).

Table A3.1: SPA and NHP Comparison

NHP Integration with SDP

The NHP is an enhanced replacement for the SPA component in the SDP architecture. Network hiding was initially introduced in SDP 1.0 using SPA to protect infrastructure components. By replacing SPA with the NHP, organizations can significantly boost the security of their SDP deployments.

Integration Overview

By incorporating the NHP, organizations can reduce their attack surface by rendering SDP components such as Controllers and Gateways invisible to unauthorized users. This integration ensures that even the SDP infrastructure remains concealed from potential attackers, enhancing security against network-based threats.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

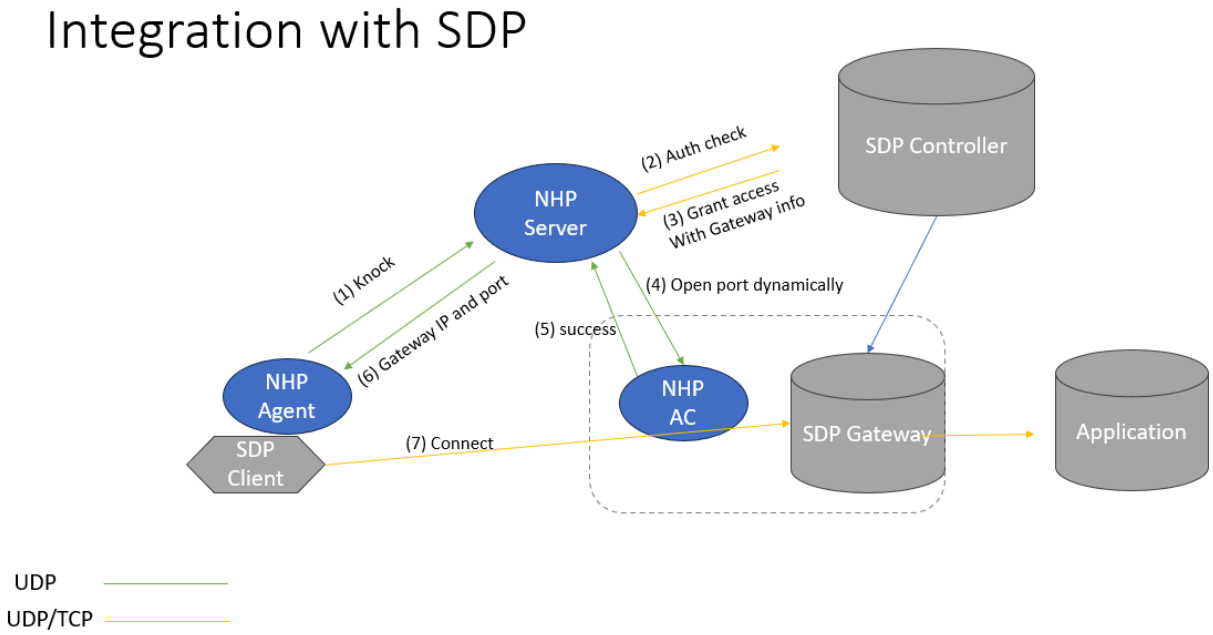


Figure A3.3: NHP Integration with SDP

Integration Process

1. **Knock Request Initiation:** The NHP-Agent, integrated with the SDP Client on the Initiating Host (IH), prepares a secure "knock" request. This request includes the IH's identity information, device posture, and requested resource identifiers. Modern cryptographic methods, such as the Noise Protocol Framework, encrypt the request to ensure confidentiality and integrity. The knock request is sent to the NHP-Server over a secure channel. This initial communication does not expose any network infrastructure details, maintaining the invisibility of SDP components.
2. **Authentication Check:** Upon receiving the knock request, the NHP-Server decrypts and validates the message using IBC. The NHP-Server verifies the IH's identity and device posture against its records. It checks for compliance with security policies, such as device health, location, and time constraints. The NHP-Server collaborates with the SDP Controller (acting as the ASP) to validate access policies. The SDP Controller determines whether the IH can access the requested resources according to predefined policies.
3. **Access Grant:** If authentication and authorization are successful, the SDP Controller grants access. The decision includes details on which resources and services the IH is permitted to access. The SDP Controller provides the NHP-Server with the necessary SDP Gateway information, such as IP addresses and port numbers.
4. **Port Opening Instructions:** The NHP-Server sends a secure instruction to the NHP-AC module on the SDP Gateway. This instruction includes details about the IH, such as its identity and the specific ports or services to be opened. The NHP-AC dynamically updates firewall rules or access control lists to allow the IH's IP address to connect to the specified ports. This dynamic process occurs in real time to minimize the open window during which ports remain open.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

5. **Confirmation:** The NHP-AC confirms to the NHP-Server that the necessary ports have been opened successfully. It may provide additional information, such as the validity period for the opened ports or any session-specific identifiers. The confirmation ensures that the IH can proceed to the next step, knowing that access has been provisioned.
6. **Gateway Information Relay:** The NHP-Server sends the SDP Gateway's IP address, port information, and any necessary session parameters back to the NHP-Agent on the IH. This message is securely transmitted, maintaining confidentiality and integrity. The NHP-Agent updates the SDP Client with the received information, preparing it to establish a connection with the SDP Gateway.
7. **SDP Connection Establishment:** The SDP Client on the IH initiates a connection with the SDP Gateway using the provided IP address and port. Mutual TLS (mTLS) authentication ensures that both the IH and the SDP Gateway are authenticated. Upon a successful mTLS handshake, the standard SDP workflow proceeds. The IH gains access to the authorized resources through a secure, encrypted tunnel. Throughout the session, access remains governed by policies defined by the SDP Controller. Any policy changes or detected anomalies can result in session termination.

By integrating the NHP with SDP, organizations enhance the invisibility of their network infrastructure, making it significantly more difficult for unauthorized users to discover or target SDP components.

Benefits of Integrating NHP with SDP

- **Enhanced Security**
  - **Infrastructure Hiding:** The NHP employs advanced cryptographic techniques to conceal SDP components from unauthorized users, significantly reducing the risk of network reconnaissance
  - **Authenticate-Before-Connect:** Adds an extra layer of authentication before any connection is established
- **Improved Resilience to Attacks**
  - **DDoS Mitigation:** The NHP reduces the potential vectors for DDoS attacks against SDP components by ensuring that only authenticated clients can initiate communication
  - **Zero-Day Protection:** Hiding SDP components minimizes exposure to potential zero-day vulnerabilities
- **Scalability and Performance**
  - **Decoupled Architecture:** The NHP separates authentication and access-control components, enabling scalable deployment without significantly impacting SDP performance
  - **Efficient Key Management:** Modern cryptographic methods, such as IBC, simplify key distribution and management, enhancing security and performance
- **Compliance and Monitoring**

V5 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

- **Enhanced Logging:** The NHP provides detailed logs that can be integrated with existing SIEM systems for comprehensive monitoring
- **Simplified Compliance Reporting:** Improved audit trails facilitate adherence to security standards and regulations

Implementation Considerations

When integrating NHP into your SDP deployment, consider the following:

- **Component Placement:** Deploy the NHP-Server in a secure, centralized location to efficiently handle authentication requests before they reach SDP components
- **Synchronization:** Ensure the NHP's authentication processes are seamlessly integrated with SDP workflows to maintain a cohesive security posture
- **Key Management:** Employ automated key management solutions to synchronize cryptographic keys and certificates between the NHP and SDP components
- **Logging and Monitoring:** Integrate NHP logs with existing SDP and SIEM systems for comprehensive monitoring and incident response
- **Performance Impact:** Monitor the effect of NHP on SDP performance and adjust configurations as necessary to optimize performance

Applicability Across SDP Deployment Models

The NHP can enhance security in various SDP deployment models:

- **Client-to-Gateway Model:** The NHP ensures that the SDP Gateway remains hidden until authentication succeeds, adding an extra layer of security before the standard SDP connection is established
- **Client-to-Server Model:** Adds an extra layer of protection by authenticating access to individual servers
- **Server-to-Server Model:** Secures server-to-server communications by ensuring only authenticated servers can connect
- **Complex Topologies:** In models such as Client-to-Server-to-Client, the NHP can authenticate multiple points along the communication chain. For instance, the NHP can authenticate multiple points in peer-to-peer applications or client-mediated interactions, ensuring secure communication channels throughout the network

Open Source Project - OpenNHP

The NHP has been implemented as an open-source project named OpenNHP. You can find the repository at: https://github.com/OpenNHP/opennhp.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

This project aims to provide a reference implementation of the NHP, allowing the community to:

1. Understand the practical aspects of implementing the NHP.
2. Contribute to the development and improvement of the protocol.
3. Test the NHP in various network environments.
4. Integrate the NHP into the existing security infrastructure.
5. Audit the NHP implementation.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Appendix 4: OpenZiti/NetFoundry Details

OpenZiti is the world's most used and widely integrated open-source secure networking platform. OpenZiti provides both Zero Trust security and overlay networking as pure open-source software. The project provides all the necessary components to implement a Zero Trust overlay network and offers the tools required to integrate Zero Trust security into your existing solutions. The OpenZiti project believes that Zero Trust principles shouldn't stop at your network; they should extend to your application.

Implementing OpenZiti offers several critical advantages from a security perspective:

- **Making Conventional Network Threats Useless:** Traditionally, applications must listen on the underlying network (e.g., host OS network, LAN, WAN) and thus can be attacked by IP-based tooling. If using OpenZiti's app-embedded ZTN techniques, the application no longer needs to listen on the underlay network. It's unattackable via conventional IP-based tooling
- **Mitigate Vulnerability Risk:** Traditional TCP/IP protocols follow a "trust by default" connection model, allowing connections from any client to server ports offering services, leaving systems vulnerable to attack. OpenZiti implements the Zero Trust principle of "never trust, always verify" by enforcing a deny-all policy by default. Only authenticated and authorized clients are allowed to establish connections, which are always made outbound from the source/destination to the fabric (or PEP). This makes external network attacks (e.g., zero-day exploits, CVEs, DDoS, credential stuffing) impossible at both the source and destination, while attacks against the fabric are significantly more difficult
- **Mitigate Phishing Attacks:** DNS hijacking is a well-known attack vector used in phishing and malware distribution. OpenZiti provides a private DNS that does not require compliance with top-level domains. All traffic on the OpenZiti overlay network uses this private DNS and Ziti's system of identity to route traffic. IP and DNS are only used for routing traffic, not for the overlay. This eliminates the risk of DNS hijacking
- **Mitigate DDoS Attacks:** Clients cannot obtain the IP addresses and port numbers of protected resources without prior authentication. Combined with the outbound connectivity, OpenZiti allows resources to remain obscured, making it impossible for attackers to launch DDoS attacks. While attackers could launch a DDoS attack against the fabric, OpenZiti has built resiliency, HA, and intelligent routing into the controllers and routers, and an eBPF firewall has been developed to make DDoS attacks close to useless
- **Attack Attribution:** Unlike the traditional IP-based connection model of TCP/IP, OpenZiti shifts to an identity-based model. Every connection requires the client's identity to be authenticated before the connection is established, enhancing the traceability and auditability of actions performed on protected resources. Operators can see exactly which identity is accessing which services, for how long, at what time, and how much data is being transferred

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

OpenZiti Architecture and Components

The OpenZiti architecture ensures that protected resources remain hidden from unauthorized users and are only accessible after continuous authentication and verification. This architecture is designed to support modern cryptographic techniques, providing scalability and flexibility for advanced Zero Trust environments.

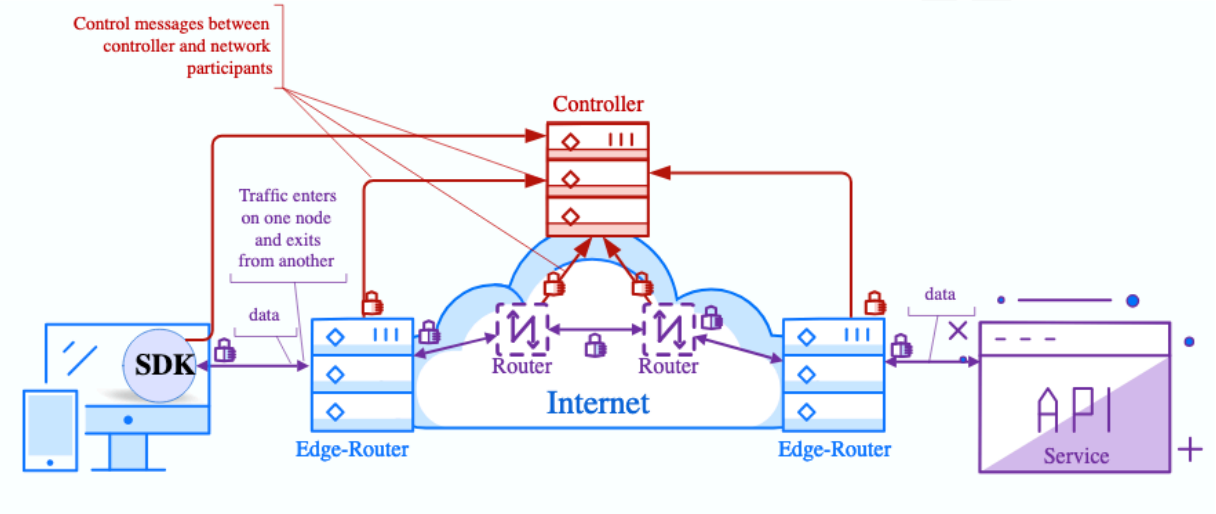


Figure A4.1: An Example of an OpenZiti Overlay Network

In OpenZiti, you will find:

- A Zero Trust overlay mesh network composed of a controller, edge routers, and fabric routers
- Numerous SDKs make it easy to embed Zero Trust principles directly into your application
- Tunneling applications providing Zero Trust access to those applications (Zero Trust cannot be embedded into them directly)
- Zero Trust principles such as authorization before connection, continual authorization, and least privilege access

OpenZiti makes it easy to embed Zero Trust. You can have Zero Trust, high-performance, programmable networking directly into your app. With OpenZiti, you can have Zero Trust, high-performance networking over any internet connection without VPNs!

Ready to deploy your first network? Jump right in, feet first, and follow along with one of our quickstart guides!

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Let's explore these components. We will describe them in two parts: the software components deployed, and then the logical constructs.

Software Components

- **Controller:** The controller is the central function of the network. The controller provides the configuration plane. It is responsible for configuring services and serves as the central point for managing the identities of users, devices, and network nodes. Lastly, but critically, the controller is responsible for authentication and authorization for every connection in the network. The controller must be configured with a public key infrastructure (PKI). The configured PKI is used to create secure, mutually authenticated TLS (mTLS) network connections between any two network endpoints. The controller does not provide its own PKI. For the controller to sign a certificate signing request (CSR), it must be configured with a key and a certificate for signing. The ziti pki command can generate a PKI. The controller also supports the use of a third-party PKI if the network operator has an existing PKI they wish to reuse. Utilizing a third-party CA shifts the burden of obtaining and distributing properly signed certificates to the network operator, but for sophisticated customers, this might make overall network management easier. The controller uses a local Bolt-based database to store the information required to manage the network. The controller's TLS server uses SNI to select the correct certificate to present when multiple certificates are available. Ziti clients use ALPN to negotiate a connection to the control plane (ziti-ctrl) or the REST APIs (h2, http/1.1). See the Controller Deployment Guide
- **Routers:** The fundamental building blocks of the network. These routers securely and reliably deliver traffic from one network node to the destination. Ziti Routers are linked together to form a mesh network. This mesh is constantly being monitored for latency, and the fastest paths are used to route traffic to the destination. The monitoring also enables active failover to ensure a reliable network connection, even in the event of a node failure. The router is the entry point for client connections into the network. The router and the controller are responsible for authenticating and authorizing clients. See the Router Deployment Guide
- **Edge Clients:** Connecting to the network requires an edge client. Edge Clients are designed to work with both brownfield and greenfield applications. If the solution being developed includes new software, Ziti SDKs support multiple languages and runtimes to provide fast, reliable, and secure connectivity. These SDKs provide the necessary capabilities to connect to the network securely and are designed to be easily integrated into the target application. When adding secure connectivity to an existing solution, OpenZiti provides specialized Edge Clients, called tunnelers, that provide seamless, secure connectivity without changes to the target application. Read more about clients
- **BrowZer:** A set of optional components that facilitate bootstrapping trust in a web browser without requiring client-side installation. This means there is no need to install an extension in web browsers or one of the mobile or desktop clients (tunnelers). It enables the automatic embedding of Zero Trust networking into a web application, thereby transforming a web browser (e.g., Chrome, Brave, Edge) into a full-fledged client. The only software users need is the ubiquitous

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

browser they use daily. Also noteworthy is that BrowZer places no burden on web application developers to first instrument or otherwise modify the web application to enable secure remote access. BrowZer enables operating a web app licensed from a 3rd party and protecting it without altering it. Similarly, if a web app can be modified but is not desirable to do so, BrowZer allows OpenZiti to protect those apps as well. The BrowZer Bootstrapper automatically performs the necessary instrumentation of the web application on the fly, as it is being loaded from the web server to the user's browser. It must be configured to enable BrowZer in a given network. For information on how to add BrowZer to a network, follow the BrowZer quickstart guide

Logical Components

Once the network is established and deployed, the next step is to configure the software-powered network. The three main concepts for configuring a network are Identities, Services, and Policies:

- **Identities:** Identities represent individual endpoints in the network that can establish connectivity. All connections within the network are mutually authenticated using X.509 certificates. Every Identity is mapped to a given certificate's signature. Edge clients present this certificate when initiating network connections. The network uses the presented certificate to authorize the client and enumerate the services for which the Identity is authorized. Read more about identities and authentication
- **Services:** A service encapsulates the definition of any resource that a client could access on a traditional network. A service is defined by a strong, extensible identity rather than by an expression of an underlying concept. This means that services defined on a network have an almost limitless "namespace" available for identifying services. A service is defined by a name and/or a certificate rather than a DNS name or an IP address (underlay concepts). Services also declare a node to which traffic exiting the network must be sent before exiting. The node where traffic enters the network may be the same node where traffic exits. Alternatively, traffic may need to traverse the network routers to reach the exit node. Specifying the node is all the end-user needs to do; the network handles the rest. Read more about services
- **Policies:** Policies control how Identities, Services, and Edge Routers interact. To use a service, the identity must be granted access. Additionally, since all access to a service is routed through one more edge router, both the service and the identity must be granted access to the same edge router or a set of edge routers
    - **Role Attributes:** Entities such as identities, services, and edge routers can be explicitly added to policies by ID or name. Entities can also be tagged with role attributes. Role attributes are simple strings like sales, Boston, us-employees, or support. The administrator decides their meaning. Policies can include entities by specifying a set of role attributes to match
    - **Service Policies:** Encapsulate the mapping between identities and services in a software-powered network. In the simplest terms, Service Policies comprise a set of services and a set of identities. Adding a service to a Service Policy grants the identities

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

in that Service Policy access to the specified service. Similarly, adding an identity to a Service Policy grants that identity access to the services mapped by that Service Policy. Service Policies control both which identities may dial a service (use the service) and which identities may bind a service (provide or host the service). Each Service Policy may either grant dial or bind access, but not both

- **Edge Router Policies:** Manage the mapping between identities and edge routers. Edge router policies are a group of edge routers and a group of identities. Adding a router to an Edge Router Policy grants the identities in that Edge Router Policy access to the specified edge router. Similarly, adding an identity to an edge router policy grants that identity access to the edge routers mapped in that policy
- **Service Edge Router Policies:** These policies manage the mapping between services and edge routers. Service edge router policies are collections of edge routers and services. Adding an edge router to a Service Edge Router Policy will grant the services in that Service Edge Router Policy access to the given edge router. Similarly, adding a service to a Service Edge router policy grants that service access to the edge routers mapped by that policy. Read more about authorization

Connection Security

Ziti networks use robust modern cryptography and security mechanisms. Each component in a Ziti network uses security technology tailored to its role and use cases.

Here are the different types of connections:

- **Control:** Connections between a controller and a router that manage network state. Secured via mTLS 1.2+
- **Link:** Connections between routers to form a mesh network. Secured via mTLS 1.2+
- **Edge:** Multiplexed connections between an SDK and an edge router that carries service connections. Secured via mTLS 1.2+
- **Controller APIs:** Connections between a controller and an SDK (or other software) via HTTPS and/or secure websockets. Secured via TLS, mTLS 1.2+
- **Service:** Connections carried by an edge connection that establishes communication between a client and a host. Application data is end-to-end encrypted using libsodium public-key cryptography

Below is a diagram illustrating the connections between the `control`, `link`, `edge`, and `controller APIs`. The `service` connections exist within an `edge` connection and are pictured in more detail in the second diagram.

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

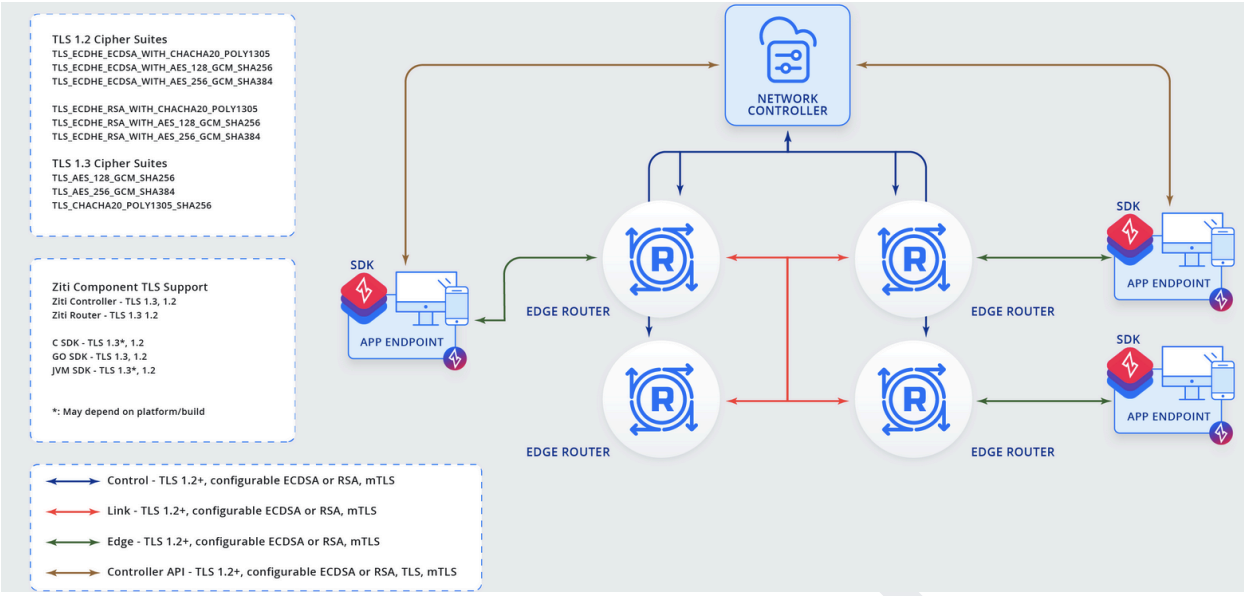


Figure A4.2: Ziti Network Connection Security

Connections between SDKs and Edge Routers are called `edge` connections. `Edge` connections are multiplexed and carry multiple `service` connections. Each connection is dedicated to a specific service and secured with end-to-end encryption to transport application/service data securely between the two intended parties only.

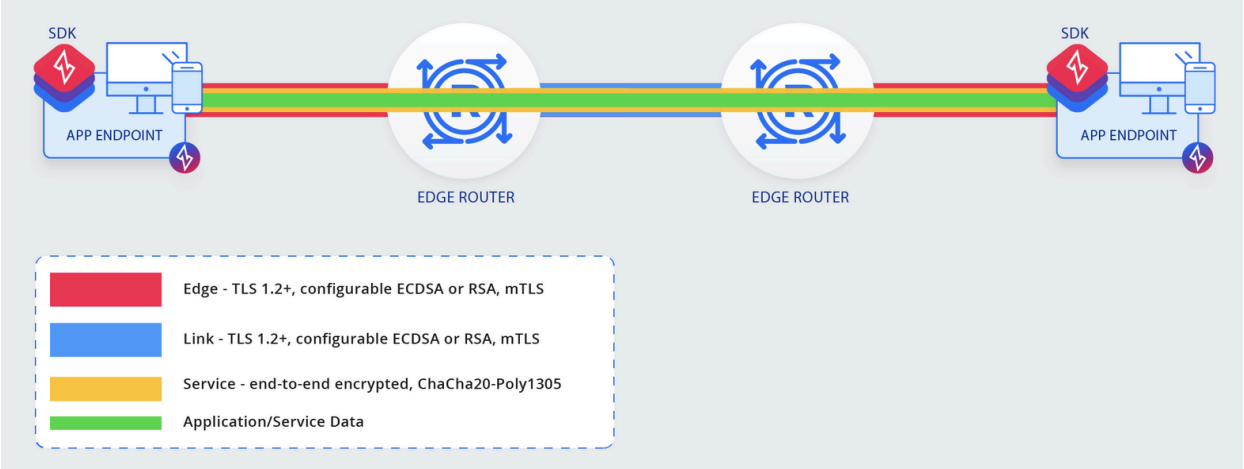


Figure A4.3: Ziti Network Edge Connections

Control and Link Connection Details

Routers work in concert with a controller to establish a mesh network of `links` among routers. Routers coordinate with a controller over a `control` connection. These connections are initialized and maintained while the network is in operation. `Control` and `link` connections are always authenticated using mTLS.

© Copyright 2026, Cloud Security Alliance. All rights reserved.

The certificates that enable mTLS for control connections are exchanged during the router enrollment process. The control connection between controllers and Edge Routers continuously updates routers with the proper certificate information for all other routers in the mesh.

Controller API Connection Details

Controller APIs allow clients (SDKs or otherwise) to interact with a network. The edge management API is used for configuration and maintenance. The edge client API allows clients to authenticate, discover services, request Sessions, discover Edge Routers, and perform basic self-maintenance.

Access to the APIs requires authentication, which returns an API Session to the client. An API Session is required to make edge connections.

Edge Connection Details

Edge connections are made between SDKs and Edge Routers. They require the following:

- a valid API Session represented by a token
- a valid x509 certificate associated with the supplied API Session
- a target Edge Router

An API Session is obtained during authentication with the edge client or management APIs. It is represented as a token. The x509 certificate used to establish the mTLS connection may be either the certificate used during authentication (if applicable) or an API Session Certificate.

Once established, edge connections will be allowed, and Session tokens will be used to establish service connections. When a Session is created, a list of valid Edge Routers is included in the response.

Service Connection Details

Service connections represent an SDK that has connected to a service (dial) or is hosting a service (bind). To establish a service connection of either type, the following is required:

- An edge connection to an edge router that has the correct policies in place to support the target service and intent (dial/bind)
- A Session for the target service and intent (dial/bind)

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Sessions are issued by the controller's edge client API. A valid Sessions token must be included with dial and bind requests. Edge Routers continuously validate API Session tokens. If valid, the Edge Router will facilitate connecting the client to a service or registering the client as a host.

Should a Session become invalid at any point, any existing service connection established with the invalidated Session will be terminated. Any service that attempts to re-establish a connection with the invalidated Session will be refused.

Comparison Between SPA and OpenZiti

	SPA	OpenZiti	OpenZiti Advantages
Architecture	The SPA packet-decryption and user/device-authentication component is coupled with the network access control component on the SPA server.	The OpenZiti controller (which handles identity and policy management) is decoupled from the protected resources. It can be deployed independently for distributed and horizontal scaling. OpenZiti also includes a decoupled, highly available, and smart routing data plane for high performance.	• **Performance:** Better E2E performance and reliability for apps • **Scalability:** Both control and data planes are scalable and resilient • **Security:** The protected server is accessible to the client only if the authentication/authorization is successful
Communication	Single direction	Bi-directional, with no concept of client or server. Any endpoint can access or host a service.	It can be used for any use case while providing better reliability with the status notification of access control.
Cryptographic Framework	PKI	• PKI can work with any external provider to replace primary	• **Security:** Proven secure key exchange mechanism to

v3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

		or augment secondary • mTLS for link encryption • E2E encryption from source edge to destination edge	mitigate the MITM threats • **Low cost:** Efficient key distribution for the Zero Trust model • **Performance:** High-performance encryption/decryption
Capability of Hiding the Network Infrastructure	Only server ports	Private DNS provides complete obfuscation, allowing you to mask the actual destination domain/IP address and ports. All metadata in flight is encrypted except the next hop IP, which is synthesized to look like HTTPS/443.	Enhanced defense and obfuscation against various attack vectors (e.g., vulnerabilities, DNS hijack, and DDoS attacks).
Extensibility	No, only for SDP	OpenZiti can support any use case while being software and API driven.	Support any scenario that needs service darkening, including IT, OT, IoT, and more.
Interoperability	Not available	Customizable and agnostic to protocols, external identity, and OS platforms.	OpenZiti can seamlessly operate with existing technologies and be built into any application.

Table A4.1: SPA and OpenZiti Comparison

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Appendix 5: SDP Market Survey

Below are our survey results that focus on (1) what solutions are out there, (2) how closely they hew to the five SDP principles, (3) which modern use-case domains they truly cover, and (4) where the gaps still lie. It is dated to mid-2025.

While several public-facing analyst surveys, such as Gartner's Market Guide for ZTNA, Forrester's Zero Trust Wave, and GigaOm's Radar for ZTNA, offer valid comparisons of ZTNA and Security Service Edge (SSE) vendors, they do not explicitly evaluate solutions against the core principles of SDP. This is primarily due to a scope mismatch. These reports tend to focus on remote-access usability, vendor reach, or SSE bundling rather than on architectural purity or service-layer cloaking. As a result, key SDP pillars such as SPA-based invisibility, mTLS enforcement, or embedded application security often go unmeasured. Additionally, most of these reports are behind analyst paywalls or require vendor reprints for public access, which limits transparency and the ability to benchmark solutions specifically for SDP conformance.

The methodology used combined:

- **Tier**
  - **5 / 5** = satisfies all five core SDP principles natively
  - **4 / 5** = one yellow box
  - **≤ 3 / 5** = two or more gaps
- **Use-Case Buckets** (drawn from the Sample Vendor Implementation and SDP Scenarios sections)
  - **1 DC-Remote:** cloaking classic datacenter workloads from remote/users
  - **2 Hybrid:** hybrid and multi-cloud workload meshes
  - **3 OT/ICS:** operational technology and industrial control flows
  - **4 Edge/IoT:** highly distributed, resource-constrained fleets
  - **5 Service by Design/Embedded:** product or application-embedded Secure by Design overlays
- **Legend:** ✅ = production-ready; 🟡 = possible with add-ons/refactoring; ❌ = not a current focus
-

Platform	Tier	1 DC-Remote	2 Hybrid	3 OT/ICS	4 Edge/ IoT	5 Secure by Design/ Embedded	Primary Limitations
AppGate SDP	5/5	✅	✅	🟡 (requires agentless gateway)	🟡	🟡	SPA only; no in-app SDKs
OpenZiti/NetF	4/5	✅	✅	✅	✅	✅ (full	Dynamic

© Copyright 2026, Cloud Security Alliance. All rights reserved.

oundry						SDK)	revocation is still admin/API-drive n
Google BeyondCorp En terprise	4/5	✅	✅	🟡	🟡	❌	Public IAP FQDN discoverable (no SPA)
Zscaler ZPA	3/5	✅	✅	❌	❌	❌	Final hop may lack mTLS; no packet-drop cloaking
Palo Alto Prisma Access (ZTNA 2.0)	3/5	✅	✅	❌	❌	❌	mTLS optional; no SPA layer
Akamai EAA	3/5	✅	✅	❌	❌	❌	Mutual-auth add-on only; proxy FQDN reachable
Cloudflare Access	3/5	✅	✅	❌	🟡 (with Cloudflare Workers)	❌	No SPA; posture optional; edge PoP dependency
IBM Security Verify (ZTNA)	3/5	✅	✅	❌	❌	❌	Reverse proxies remain routable; revocation lag
Perimeter 81	2.5/5	✅	🟡	❌	❌	❌	Fixed gateways are discoverable; device posture is basic
Cisco Duo Beyond	2.5/5	✅	🟡	❌	❌	❌	Network masking relies on firewalls, not SPA
Tailscale/ ZeroTier	2/5	🟡	🟡	🟡	✅	❌	WireGuard keys only; no app-layer policy engine
GoodAccess/T wingate (SMB)	2/5	✅	🟡	❌	❌	❌	Limited posture; coarse policy; SaaS only

Table A5.1: SDP Market Survey Results

© Copyright 2026, Cloud Security Alliance. All rights reserved.

How to Interpret This Information

- Who is "closest" to pure SDP?
  - AppGate and OpenZiti/NetFoundry
  - BeyondCorp misses only SPA-grade invisibility but also only works as a proxy for "client-server" and admits they still need to use VPN for edge use cases (See BeyondCorp and the long tail of Zero Trust)
- Mainstream ZTNA gateways (Zscaler, Palo Alto, Akamai, Cloudflare, and IBM) excel at remote and hybrid access but still expose at least one routable endpoint and often treat mTLS/posture as optional
- Developer-centric overlays (Tailscale, ZeroTier) excel at lightweight IoT or ad hoc tunnels but lack application-centric policy and stealth features, such as SPA/packet-drop protection
- SMB SaaS offerings (Perimeter 81, GoodAccess, and Twingate) offer trade depth for simplicity, fine for VPN replacement but not for OT or product-embedded scenarios

At the very top of the purity ladder sit **AppGate SDP** and **OpenZiti/NetFoundry**. AppGate follows the CSA SDP reference model almost line by line; every service is cloaked behind an SPA knock, all tunnels are mutually authenticated with mTLS, and entitlements are recalculated in real-time, so a compromised device is cut off mid-flow.

NetFoundry achieves the same level of invisibility by dropping unsolicited packets in its updated NetFoundry Edge Router (which uses eBPF), and then layers certificate-bound identities and per-service policies on top. The only remaining yellow box for NetFoundry is policy revocation, which still relies on an API call rather than an always-on risk engine.

Google BeyondCorp Enterprise is the lone runner-up. It enforces mTLS and live device posture on every request, yet its Identity-Aware Proxy remains publicly reachable, and Google itself concedes that anything outside the client-server web model (e.g., edge gateways, OT controllers, contractor laptops) still needs a fallback VPN.

A step down are the mainstream **ZTNA gateways, Zscaler ZPA, Palo Alto Prisma Access, Akamai EAA, Cloudflare Access,** and **IBM Security Verify**. These services excel at brokering remote and multi-cloud traffic, with connectors that call out from private networks, dynamically pushed least privilege policies, and global points of presence to ensure low latency. The trade-off is depth. Most leave at least one routable endpoint exposed (usually the cloud proxy), and mTLS or device-posture checks are optional rather than mandatory, leaving insiders or compromised hosts with a foothold if admins skip the hardening steps.

Developer-centric overlays such as **Tailscale** and **ZeroTier** solve a different problem: they let engineers spin up WireGuard-based meshes in minutes, perfect for prototypes, IoT pilots, or quick "jump-box-free"

V5 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.

troubleshooting. Because everything is key-based and peer-to-peer, small edge devices can run them with ease. However, the focus is on network plumbing, not application security; there is no SPA-style stealth, no per-service entitlement engine, and no posture-driven session revocation. If you need granular, auditable policies or must pass a PCI or NERC audit, you'll need to bolt on extra controls.

Finally, the SMB-oriented SaaS VPN replacements—**Perimeter 81, GoodAccess, Twingate, and similar**—optimize for simplicity and price. In a few clicks, you get SSO-integrated tunnels, basic role-based rules, and traffic logs good enough for small compliance checkboxes. What you lose is sophistication: post-connect risk scoring, OT-grade offline operation, SDKs for product-embedded security, and, above all, any mechanism to render workloads truly dark to the internet. They're excellent for retiring legacy VPN concentrators, but ill-suited to high-assurance industrial or secure-by-design software projects.

Where Each Platform Fits Today

Use-Case Lens	Observations and Examples
Datacenter Cloaking	All listed vendors can front legacy DC apps; true "darkness" (no banner, no open port) is delivered only by AppGate and OpenZiti/NetFoundry.
Hybrid/Multi-Cloud	NetFoundry's SDK-inside-container pattern and Google's IAP model remove network hair-pinning; most others rely on regional PoPs or fixed connectors.
OT and ICS	NetFoundry offers agentless "tap-and-wrap" gateways, and AppGate publishes Purdue-L3 designs; ZTNA clouds generally shy away from Level-1/2 latency and offline-mode needs.
Edge and IoT	OpenZiti's SDK and Tailscale's WireGuard mesh enable constrained devices to self-dial; however, large cloud PoP solutions struggle with intermittent 4G/LoRa connections.
Secure by Design/Embedded	Only OpenZiti ships language SDKs to embed SDP directly into applications today; BeyondCorp, Zscaler, and others depend on external proxies, so security remains bolt-on rather than baked-in.

Table A5.2: Platform Use Case Observations and Examples

© Copyright 2026, Cloud Security Alliance. All rights reserved.

Typical Limitations

Limitation	Who Does it Still Affect	Why it Matters
No SPA or Kernel-Drop Cloak	Most ZTNA clouds	Assets remain discoverable via port scans and can still be DDoS-probed.
mTLS Optional/Last Hop Plain TLS	Zscaler, Palo Alto, Akamai, Cloudflare	Breaks the "mutual authentication" pillar; insider spoofing is possible on the final segment.
Static, Ticket-Based Policy	SMB offerings, dev overlays	Lacks risk-adaptive revocation; ignores device posture drift.
PoP Reliance/Hair-Pin Latency	Cloud-first brokers	Adds hops for east-west or OT traffic; PoP outage = access outage.
No SDK/Embedded Route	All but NetFoundry	Cannot satisfy CISA's Secure by Design principles for software supply-chain use cases.

Table A5.3: Platform Limitations

V3 DRAFT

© Copyright 2026, Cloud Security Alliance. All rights reserved.