NIST Special Publication
NIST SP 800-207A ipd

A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments

Initial Public Draft

Ramaswamy Chandramouli
Zack Butcher

This publication is available free of charge from:
https://doi.org/10.6028/NIST.800-207A.ipd

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Special Publication
NIST SP 800-207A ipd

A Zero Trust Architecture Model for Access Control in Cloud-Native Applications in Multi-Location Environments

Initial Public Draft

Ramaswamy Chandramouli
Computer Security Division
Information Technology Laboratory

Zack Butcher
Tetrate, Inc.

This publication is available free of charge from:
https://doi.org/10.6028/NIST.800-207A.ipd

April 2023

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

Certain commercial equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

Authority

This publication has been developed by NIST in accordance with its statutory responsibilities under the Federal Information Security Modernization Act (FISMA) of 2014, 44 U.S.C. § 3551 et seq., Public Law (P.L.) 113-283. NIST is responsible for developing information security standards and guidelines, including minimum requirements for federal information systems, but such standards and guidelines shall not apply to national security systems without the express approval of appropriate federal officials exercising policy authority over such systems. This guideline is consistent with the requirements of the Office of Management and Budget (OMB) Circular A-130.

Nothing in this publication should be taken to contradict the standards and guidelines made mandatory and binding on federal agencies by the Secretary of Commerce under statutory authority. Nor should these guidelines be interpreted as altering or superseding the existing authorities of the Secretary of Commerce, Director of the OMB, or any other federal official. This publication may be used by nongovernmental organizations on a voluntary basis and is not subject to copyright in the United States. Attribution would, however, be appreciated by NIST.

NIST Technical Series Policies

Copyright, Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History

Approved by the NIST Editorial Review Board on YYYY-MM-DD [Will be updated in final publication]

How to Cite this NIST Technical Series Publication:

Chandramouli R, Butcher Z (2023) A Zero-Trust Architecture Model for Access Control in Cloud Native Applications in Multi-Location Environments. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-207A ipd. https://doi.org/10.6028/NIST.800-207A.ipd

Author ORCID iDs

Ramaswamy Chandramouli: 0000-0002-7387-5858

Public Comment Period

April 18, 2023 – June 7, 2023

Submit Comments

sp800-207A-comments@nist.gov

National Institute of Standards and Technology
Attn: Computer Security Division, Information Technology Laboratory
100 Bureau Drive (Mail Stop 8930) Gaithersburg, MD 20899-8930

All comments are subject to release under the Freedom of Information Act (FOIA).

Abstract

One of the basic tenets of zero trust is to remove the implicit trust in users, services, and devices based only on their network location, affiliation, and ownership. NIST Special Publication 800-207 has laid out a comprehensive set of zero trust principles and referenced zero trust architectures (ZTA) for turning those concepts into reality. A key paradigm shift in ZTAs is the change in focus from security controls based on segmentation and isolation using network parameters (e.g., IP addresses, subnets, perimeter) to identities. From an application security point of view, this requires authentication and authorization policies based on application and service identities in addition to the underlying network parameters and user identities. This in turn requires a platform that consists of API gateways, sidecar proxies, and application identity infrastructures (e.g., SPIFFE) that can enforce those policies irrespective of the location of the services/applications, whether on-premises or on multiple clouds. The objective of this publication is to provide guidance for realizing an architecture that can enforce granular application-level policies while meeting the runtime requirements of ZTA for multi-cloud and hybrid environments.

Keywords

egress gateway; identity-tier policies; ingress gateway; microservices; multi-cloud; network-tier policies; service mesh; sidecar proxy; SPIFFE; transit gateway; zero trust; zero trust architecture.

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems. The Special Publication 800-series reports on ITL's research, guidelines, and outreach efforts in information system security, and its collaborative activities with industry, government, and academic organizations.

Call for Patent Claims

This public review includes a call for information on essential patent claims (claims whose use would be required for compliance with the guidance or requirements in this Information Technology Laboratory (ITL) draft publication). Such guidance and/or requirements may be directly stated in this ITL Publication or by reference to another publication. This call also includes disclosure, where known, of the existence of pending U.S. or foreign patent applications relating to this ITL draft publication and of any relevant unexpired U.S. or foreign patents.

ITL may require from the patent holder, or a party authorized to make assurances on its behalf, in written or electronic form, either:

a) assurance in the form of a general disclaimer to the effect that such party does not hold and does not currently intend holding any essential patent claim(s); or

b) assurance that a license to such essential patent claim(s) will be made available to applicants desiring to utilize the license for the purpose of complying with the guidance or requirements in this ITL draft publication either:

   i. under reasonable terms and conditions that are demonstrably free of any unfair discrimination; or

   ii. without compensation and under reasonable terms and conditions that are demonstrably free of any unfair discrimination.

Such assurance shall indicate that the patent holder (or third party authorized to make assurances on its behalf) will include in any documents transferring ownership of patents subject to the assurance, provisions sufficient to ensure that the commitments in the assurance are binding on the transferee, and that the transferee will similarly include appropriate provisions in the event of future transfers with the goal of binding each successor-in-interest.

The assurance shall also indicate that it is intended to be binding on successors-in-interest regardless of whether such provisions are included in the relevant transfer documents.

Such statements should be addressed to: sp800-207A-comments@nist.gov

Table of Contents

List of Figures

Acknowledgments

The author would like to express his thanks to Isabel Van Wyk of NIST for her detailed editorial review of the public comment version as well as the final publication.

Executive Summary

The principles of zero trust, as described in NIST Special Publication (SP) 800-207, have become the guiding markers for developing secure zero trust architecture. A well-established class of applications are cloud-native applications. The generally accepted characterization of a cloud native application includes the following:

- The application is made up of a set of loosely coupled components called microservices. Each of the microservices can be hosted on different physical or virtual machines (VMs) and even be geographically distributed (e.g., within several facilities that belong to the enterprise, such as the headquarters, branch offices, and in various cloud service provider environments).
- Any transaction involving the application may also involve one or more inter-service (microservice) calls across the network.
- A widespread feature (though not necessarily a requirement for cloud-native application) is the presence of a software platform called the service mesh that provides an integrated set of all application services (e.g., services discovery, networking connections, communication resilience, and security services like authentication and authorization).

The realization of a zero trust architecture for the above class of cloud-native applications requires a robust policy framework. In order to follow zero trust principles, the constituent polices in the framework should consider the following scenario:

- There should not be implicit trust in users, services, or devices based exclusively on their network location, affiliation, or ownership. Hence, policy definitions and associated security controls based on the segmentation or isolation of networks using network parameters (e.g., IP addresses, subnets, perimeter) are insufficient. These policies fall under the classification of network-tier policies.
- To ensure the presence of zero trust principles throughout the entire application, network-tier policies must be augmented with policies that establish trust in the identity of the various participating entities (e.g., users and services) irrespective of the location of the services or applications, whether on-premises or on multiple clouds.

This document provides guidance for realizing a zero trust architecture that can enforce granular application-level policies for cloud-native applications. The guidance is anchored in the following:

- A combination of network-tier and identity-tier policies
- The components of cloud-native applications that enable the definition and deployment of those policies, such as edge, ingress, sidecar, and egress gateways; the creation, issuance, and maintenance of service identities; the issuance of authentication and authorization tokens that carry user identities in the enterprise application infrastructure that encompasses multi-cloud and hybrid environments

1. Introduction

Zero trust (ZT) tenets or principles have been accepted as the guide markers for architecting all applications. There are several reasons why adherence to these tenets is critical for obtaining necessary security assurances, especially for cloud-native applications. The enterprise application environments for this class of applications is highly geographically distributed and span multiple cloud and on-premises environments (e.g., headquarters, enterprise-operated data centers, branch offices, etc.). Further, the user base consists of both remote and on-premises employees. These two features call for establishing trust in all of the data sources and computing services of the enterprise – irrespective of their location – through secure communication and the validation of access policies.

Apart from geographic distribution, another common feature of cloud-native applications is the presence of many microservices that are loosely coupled and collectively support business processes through extensive inter-service calls. This is augmented with an integrated infrastructure for providing all application services called the service mesh. These features emphasize the concept of identity for the various components of the application in the form of microservices as well as the users who access them through direct calls or clients (other services). This in turn highlights the critical need for authenticating these identities and for providing legitimate access on a per-session basis through a dynamic policy that takes the current status of the user, service, and requested asset into account.

The above requirements can only be met through a comprehensive policy framework. This document provides guidance for developing a policy framework that will form the foundation for realizing a zero trust architecture (ZTA) while incorporating zero trust principles into its design for cloud-native applications. The policy framework should also consist of a comprehensive set of policies that span all critical entities and resources in the application stack, including the network, network devices, users, and services.

1.1. Background – Zero Trust Principles and Zero Trust Architecture

A summary of the zero trust principles and the definition of a zero trust architecture, as described in NIST SP 800-207 [1], are:

- Zero trust is the term for an evolving set of cybersecurity paradigms that move defenses from static, network-based perimeters to focus on users, assets, and resources. It is a set of security primitives rather than a particular set of technologies. Zero trust assumes that there is no implicit trust granted to assets or user accounts based solely on their physical or network location (i.e., local area networks versus the internet) or on asset ownership (e.g., enterprise or personally owned). Zero trust focuses on protecting resources (e.g., assets, services, workflows, network accounts) rather than network segments, as the network location is no longer seen as the prime component to the security posture of the resource.
- A zero trust architecture uses zero trust principles to plan industrial and enterprise infrastructures and workflows.

NIST's guidance on zero trust also contains an abstract definition of zero trust architecture and gives general deployment models and use cases with which zero trust could improve an enterprise's overall information technology security posture.

1.2. Relationship to Other NIST Guidance Documents

Since the current document provides guidance for the realization of ZTA for cloud-native applications hosted in multiple locations (on-premises and multiple clouds) and the enforcement of ZT principles requires policies that are associated with various security services, it will be useful to refer to the following documents. These documents provide background information for the architecture of a microservices-based application with service mesh as well as guidance for configuring specific security services. The current document expands the reference environment to one where the IT application infrastructure of an enterprise spans multiple premises and multiple cloud provider locations as well as addresses the range of policies that are required for comprehensive security assurance.

- NIST SP 800-204A, *Building Secure Microservices-based Applications Using Service-Mesh Architecture* [2], provides deployment guidance for various security services (e.g., establishment of secure sessions, security monitoring, etc.) for a microservices-based application using a dedicated infrastructure (i.e., a service mesh) based on service proxies that operate independently of the application code.
- NIST SP 800-204B [3], *Attribute-based Access Control for Microservices-based Applications Using a Service Mesh*, provides deployment guidance for building an authentication and authorization framework within the service mesh that meets the security requirements. This may include establishing (1) zero trust by enabling mutual authentication in communication between any pair of services and (2) a robust access control mechanism based on an access control model (e.g., the attribute-based access control [ABAC] model) that can be used to express a wide set of policies and is scalable in terms of user base, objects (resources), and deployment environment.

1.3. Scope

The scope of this document includes:

- Identifying the requirements for realizing a ZTA for granular access control in microservices-based application platforms that include a service mesh infrastructure
- Identifying the infrastructural elements that should be part of the platform in order to configure and implement ZT principles
- Guidance for deploying a ZTA in the above platform and outlining the security assurances that the deployment can provide

1.4. Target Audience

This guidance is intended for security architects and infrastructure designers in organizations with a hybrid IT environment (consisting of both on-premises and multiple cloud-based applications) with a combination of legacy and microservices-based (i.e., cloud-native) applications with a built-in application services infrastructure, such as a service mesh.

1.5. Organization of This Document

The organization of this document is as follows:

- Section 2 describes a modern enterprise cloud-native application platform that includes a dedicated infrastructure for providing all application services as well as a management plane when the application spans both on-premises and multiple cloud service provider locations.
- Section 3 introduces the basic concepts of a policy framework for ZTA for the platform described in the previous section in terms of drivers and design requirements. It also provides an analysis of identity-based policies and introduces the concept of multi-tier policies.
- Section 4 describes the implementation approach for deploying multi-tier policies for two enterprise application infrastructure scenarios by outlining the roles of the service mesh, the functional components involved, and the advantages of identity-tier policies, which provide service-level segmentation and play a critical role in the security assurance of an application ecosystem to conform to zero trust principles or tenets.
- Section 5 provides a summary and conclusion.

2. The Enterprise Cloud-Native Platform and its Components

An enterprise cloud-native platform is increasingly made up of microservices that are implemented as containers and hosted on a container orchestration platform. In addition, it has a dedicated infrastructure layer called a service mesh, which provides a comprehensive set of application services (e.g., network connectivity, network resilience, observability, and security). The application services provided by a service mesh are enabled by the following:

- A built-in infrastructure for (a) providing service identities, (b) service discovery, and (c) external policy-based authorization engines based on Next Generation Access Control (NGAC), Attribute-based Access Control (ABAC), and Open Policy Agent (OPA)
- Code for performing network-related functions (e.g., traffic routing) and for ensuring network resiliency through functions such as retries, timeouts, blue-green deployments, and circuit breaking
- Code for ensuring application integrity and confidentiality through service-to-service and user-to-resource authentications and authorizations

More details on the container orchestration platform with an integrated service mesh can be found in [2], and an access control implementation in that platform is described extensively in [3].

In the modern enterprise, the platform described above is present in both on-premises data centers and multiple cloud service locations. Assuming that a service mesh instance is deployed for managing a single cluster that consists of the above platforms, there will be multiple clusters spread over multiple on-premises sites and multiple availability zones in different clouds. Consequently, there will be multiple service mesh instances.

Each service mesh instance has two main logical components: 1) a control plane that implements the APIs needed to define various configurations and policies that govern access between various microservices in that cluster and 2) a data plane that enforces those policies at runtime. However, a uniform set of policies is also needed to govern access between any pair of microservices or services in the enterprise irrespective of their location or the service mesh instance of which they are a part. This requires a global control plane that can define a uniform set of policies applicable to the entire set of services that operate in the enterprise and disseminate them to the control planes of the individual service mesh instances.

It is technically possible to have a single service mesh control plane instance (i.e., single service mesh instance) that manages multiple clusters spanning multiple environments (i.e., on-premises and on clouds). However, this architecture may make the multiple clusters a single failure domain and potentially defeat the very purpose of designing a multi-cluster configuration (i.e., availability). Thus, running a service mesh control plane instance for each cluster isolates the failure domain and improves availability and scalability. Further, providing the required underlying network connectivity to facilitate every workload (since each workload or application instance has an associated sidecar proxy that forms the data plane) to communicate with a single control plane instance is untenable in most enterprise environments and impossible in many government ones (e.g., air-gapped systems).

2.1. Enterprise Infrastructure Layer

The global control plane forms an integral part of the enterprise infrastructure layer. The management plane that contains the various interfaces is hosted within the global control plane. The roles of the global control plane and the management plane are as follows:

- The global control plane can be leveraged to perform the functions of individual control planes at the enterprise level rather than at the cluster level (e.g., issuing identities to all services in the enterprise by leveraging the enterprise PKI system).
- The management plane provides the human-computer interfaces (e.g., user interfaces, such as command line interfaces and APIs) that enable enterprise-level systems to work by encoding organizational processes related to the usage of various tools (e.g., policy definition and evaluation tools, telemetry tools, etc.) at the lower layer.

In short, the management plane enables the definition and deployment of consistent and uniform policies for all services throughout the enterprise. In addition to the global control plane and management plane, the enterprise infrastructure for a ZTA consists of local control planes (associated with service mesh instances) and a set of various types of proxies that form part of their respective data planes. The proxies act as the policy enforcement points (PEPs) and have three types:

1. Ingress proxies enforce policies for entering user or service requests from client applications that originate outside of the cluster into any service within the cluster.
2. Side-car proxies enforce policies between intra-cluster services.
3. Egress proxies enforce policies for requests that emanate from any service within the cluster to an external application that is outside of the cluster.

Figure 1 shows a schematic diagram of the entire infrastructure layer for uniform (enterprise-wide) policy deployment for realizing a ZTA:

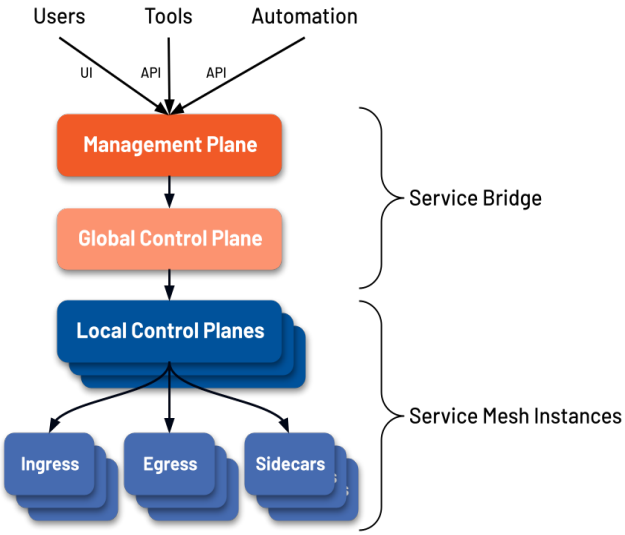

Fig. 1. Enterprise infrastructure layer for uniform policy deployment

3. Designing a Policy Framework for ZTA for Cloud-Native Application Environments

Based on the set of zero trust principles and some strawman ZTAs provided in [1], the following driver assumptions were formulated for realizing a ZTA for an enterprise cloud-native application environment (i.e., a set of microservices in various clusters with each cluster managed by a service mesh and augmented with an enterprise-level infrastructure that consists of a global control plane and management plane). These driver assumptions are:

- Trust can no longer be based on a network perimeter as perimeters can always be breached.
- Policies have to be defined based on the assumption that the attacker is already inside of the corporate network.
- All access decisions have to rely on least-privilege, per-request, and context-based principles and on identities associated with users, services, and devices. This results in a form of runtime isolation for applications, which this document refers to as "identity-based segmentation."

The above driver assumptions provide the design requirements for a ZTA as follows:

- No single component or function is sufficient to implement ZTA. Rather, they must collectively enforce zero trust principles across all applications in the infrastructure.
- ZTA component functions should be clearly articulated, including their interrelationships and workflows.

- The enforcement infrastructure that implements the security controls (mainly consisting of PEPs) should satisfy the properties of a security kernel – always invoked (non-bypassable), verifiable, and independent of the application code.
- The core tenant or primary function of ZTA at runtime is implementing an identity-based segmentation of applications that leverages the enforcement infrastructure.

3.1. Functional Components of Identity-Based Segmentation Policies for ZTA

The following policy checks should be implemented at runtime through the deployment of identity-tier policies in order to realize identity-based segmentation:

- ID-SEG-REC-1: Encrypted connection between service endpoints – Service endpoints can be located in different subnets, different availability zones or regions in a cloud provider environment, in different clouds, or on-premises. Wherever they are located, communication between any two should be encrypted to ensure eavesdropping protection and message authenticity.
- ID-SEG-REC-2: Service authentication – Each service should present a short-lived cryptographically verifiable identity to other services that is authenticated per connection and reauthenticated regularly.

  Note on the above recommendation: In an ideal situation, services would be authenticated for each service request. Since this is highly disruptive from the point of view of application transaction response, this authentication is accomplished at the connection level via mutual TLS (mTLS) when a service makes an initial connection establishment as part of its inter-service call. This authentication is not performed again in subsequent calls. However, the security of this operation is ensured by not allowing the connections to be very long (usually as long as the TTL of the service's identity certificate or as short as 15-30 minutes, depending on the configuration).
- ID-SEG-REC-3: Service to service authorization – Services should leverage runtime service identity (ID-SEG-REC-2) to enforce granular policies and have the capability to call external authorization services if the mesh level proxies are insufficient to enforce dynamic authorization policies.
- ID-SEG-REC-4: End-user authentication – Since all application requests are triggered by user actions, a robust identity management system is required to assign and maintain user identities and enforce robust protocols with phishing-resistant multi-factor (MFA) authentication. This system should be used to issue a cryptographically verifiable runtime token that represents the user principal to the rest of the infrastructure (e.g., a JSON Web Token [JWT]), and services should authenticate the credential at each hop.

  Note on the above recommendation: Authenticating the user in session at every hop is impractical at scale. Therefore, NIST recommends using short-lived end-user credentials (e.g., OAuth 2.0 tokens) for external users and exchanging them for a locally authenticatable token, like a JWT, that is authenticated at each hop.

- ID-SEG-REC-5: End-user to resource authorization – As part of each service access request, the system must ensure that the authenticated end user principal (ID-SEG-REC-4) is authorized to act on the resources designated in the request. This authorization may be performed by the application itself or checked locally (e.g., by checking against a set of claims in a JWT) or externally against an authorization system's policy decision point. Enforcing end user authorization via the service mesh's sidecar PEP is particularly effective [3].

Context for the application of these policy recommendations and the improved security assurance that emanates from their deployment and enforcement are explained in [2] and [3].

3.2. Shortcomings of Identity-Based Segmentation Policies for Enterprise ZTA

While identity-based segmentation is powerful, purely identity-based policies cannot currently be adopted due to the following scenarios:

- Identity-based segmentation policies can include access scenarios that cover all origins, such as users, services, and all target resources that consist of services and data. However, enterprise scenarios that involve both on-premises and cloud-based applications require identification of the location of those resources using network parameters. Purely identity-based enforcement should by augmented by other factors (e.g., network location) to evaluate risk when performing context-based authorization.
- A subset of identity-based segmentation policies (i.e., service identity-based) can be difficult to administer since service identity assignments are often based on specific domains, which makes consistent policy deployment difficult across on-premises systems, cloud-based systems, and different compute runtimes. However, this is mitigated by adopting consistent service names across the infrastructure using the concept of a universal identity domain, as recommended in SM-DR11 of [2].
- Having network-level policies alone requires high maintenance due to the continuous changes to their location parameters as containers and virtualized workloads are frequently migrated for availability and performance reasons (e.g., migration to different VMs or to a different pod in containerized applications).

Network-oriented policies cannot be completely eliminated given current compliance requirements and regulations. However, relaxing requirements at the network level in exchange for introducing more descriptive policy at the identity level could lead to an improved overall security posture compared to network-oriented security alone.

3.3. Multi-Tier Policies for Enterprise ZTA

A successful enterprise ZTA requires multi-tier policies:

- Network-tier policies – Allowed communication between enterprise network elements (e.g., firewall rules, which are relatively static)

- Identity-tier policies – Access scope for services and resources based on service and user identities (e.g., dynamic application-to-application communication rules based on identities through a dedicated infrastructure layer, such as user identity provided by an enterprise IAM provider and service identity provided by a standard-based Secure Production Identity Framework for Everyone [SPIFFE] server [4])

Multi-tier policies can be implemented realistically and are non-disruptive to current compliance practices. Other tiers of policy also exist. For example, in the context of the service mesh, there are "application-tier" policies, which apply to the application payload itself. These include coarse-grained WAF rules, fine-grained rules like Spring Cloud Gateway payload validation, and the validation of request semantics via tools like the Open Policy Agent (OPA). Many can even be enforced by a service mesh, but those policies are beyond the scope of this document.

The difficulty with having all network-tier policies is that policies expressed through firewall rules have to be continuously changed, depending on the application pair behind those firewalls. The flexibility in having multi-tier policies is that network-tier policies can be relatively static while identity-tier policies higher up in the stack (e.g., service to service) can be dynamic, as illustrated in **Fig. 2**.

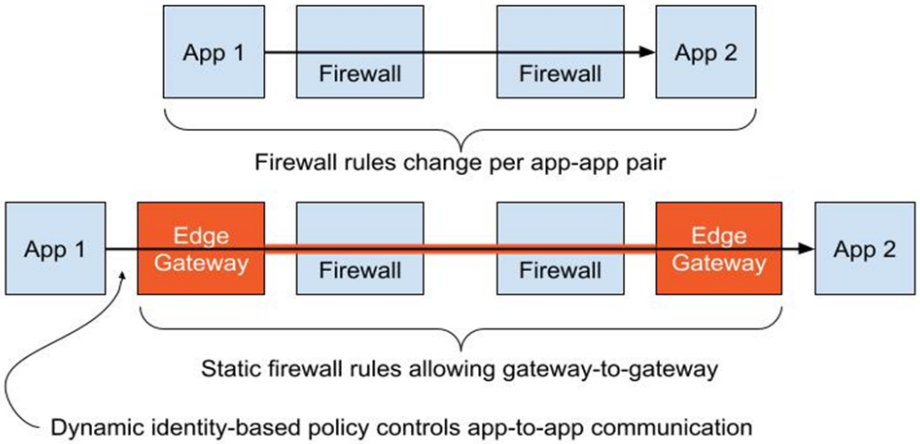

Fig. 2. Flexibility provided by multi-tier policies

Implementing identity-tier policies is also a more agile process that allows for new policy capabilities, such as writing policy in terms of identity and application-level action and verb. For example, a network-tier policy would describe the subnets that contain application instances of the client being allowed to call the subnet on a specific port. In contrast, an identity-tier policy would allow the client application identity to communicate with the server application identity via HTTPS on port 443 and execute only the GET method on the /public path. The full range of policies that an enterprise ZTA implemented via a service mesh can enable is outlined in [2] and [3].

Implementing multi-tier policies by relaxing network-tier policies (e.g., by allowing communication across a set of gateways) while introducing identity-tier policies with advanced layer seven controls results in a better overall security posture than either a purely identity-tier or purely network-tier approach.

4. Implementing Multi-Tier Policies for ZTA for Cloud-Native Application Environments

This section will consider the implementation of multi-tier policies for realizing an enterprise ZTA using a reference enterprise scenario in which an enterprise hosts microservices applications in several clusters. Each cluster is serviced with a service mesh instance, and clusters are spread out both on-premises and in multiple clouds.

Section 4.1 outlines a simple application infrastructure scenario, and Section 4.2 presents a sample set of associated policies that are relevant for that context. Section 4.3 shows how the same set of policies can be defined and deployed for a realistic application infrastructure scenario in which the incoming traffic comes through a DMZ.

4.1. Reference Application Infrastructure Scenario

Consider an application infrastructure of an enterprise where the application topology spans a cloud and on-premises environment. The applications are implemented as microservices with a service mesh instance for each cluster. Hence, a sidecar proxy is associated with each service. At the entry and exit points of each cluster are ingress and egress gateways, respectively. The same data plane (e.g., open-source Envoy) can be used to implement both the sidecar proxy and the transit gateways.

Next, consider establishing policies for a scenario that involves two services – Service 1 and Service 2 – that reside in clusters in a cloud and on-premises, respectively. Service 1 in the cloud cluster can interact with services outside of the cluster through an egress gateway. Similarly, all services that attempt to access Service 2 from outside of the cluster have to go through an ingress gateway. All traffic coming out of the cloud has to go through an outbound firewall, and all traffic coming on-premises have to come through an inbound firewall. The paired egress-ingress proxies and the firewall rules that allow them connectivity are collectively referred to as a "transit gateway." The network location for the two services are each designated by a subnet address. The application topology and policies described so far is shown in Fig. 3.

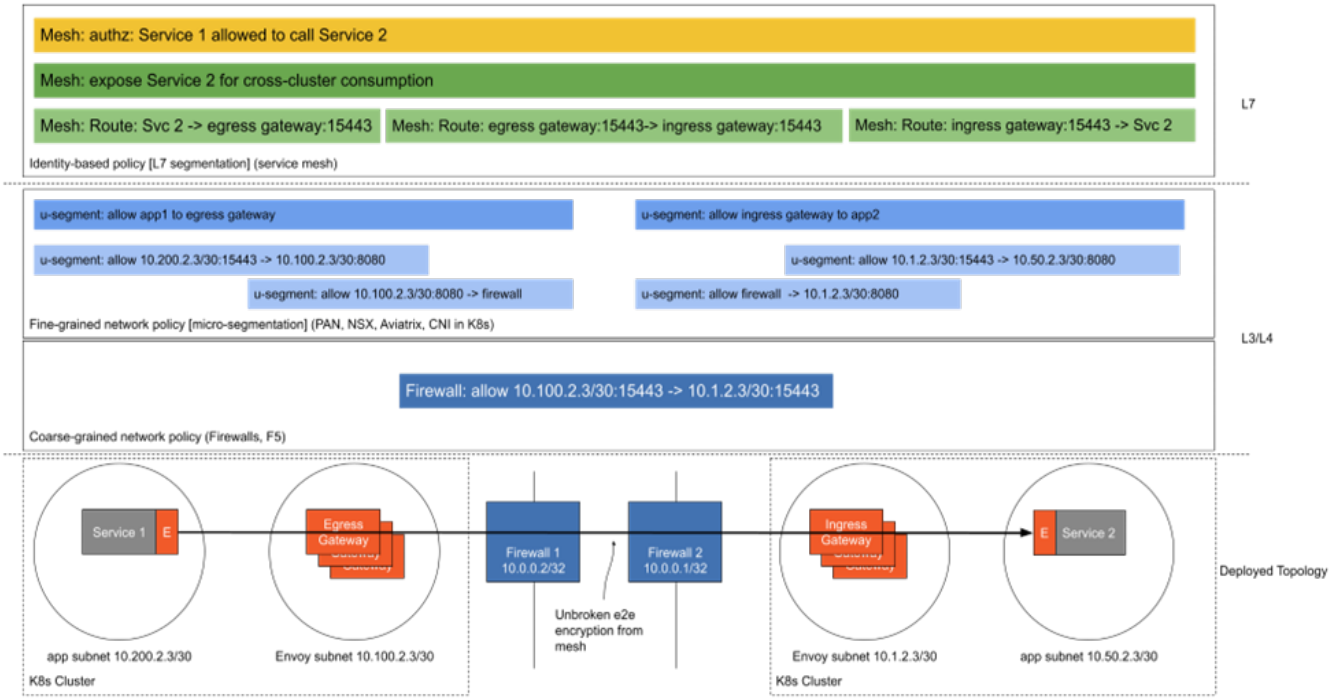

Fig. 3. Multi-tier Policies for a Hybrid Application Environment

4.2. Role of the Service Mesh in Policy Deployment, Enforcement, and Updates

The service mesh has a unique role within the overall policy life cycle activities of policy definition, deployment, enforcement, and update. As already stated, the service mesh is a dedicated infrastructure that provides all application services, including security controls like secure communication and application-level access control. These services are only possible if there are also policies to enforce them during application runtime.

Based on the discussion of the control plane in previous sections, it should clear that this component of the service mesh provides access to the interfaces of various policy definition tools through which policies can be defined and updated. Thus, the control plane of the service mesh acts as the *policy administration point*, while the underlying policy tools become the *policy decision point*. In addition, the control plane also enables those policies to be distributed to the various proxies described in the previous section. Once distributed, these proxies intercept all traffic in and out of the applications, where it acts as a universal *policy enforcement point*. This allows the service mesh – which centrally manages a fleet of the applications' proxies – to become the modern cloud-native *security kernel* [3].

The proxies – especially the sidecars – can enforce security and traffic policies and generate telemetry data to allow operators to *close the loop* on policy changes by authoring a change, observing its effect on the runtime, and making additional changes as needed in a real-time feedback control loop. In other words, the mesh provides the needed capabilities to implement the runtime controls and achieve a zero trust posture.

4.3. Policy Deployment for Reference Application Infrastructure

Connectivity (between network elements) and access policies (between service instances) are network-tier policies and identity-tier policies, respectively.

Consider the following example set of policies that contain a combination of network-tier and identity-tier policies. Network-tier policies can be further categorized into coarse-grained and fine-grained policies.

- Coarse-grained network-tier policies – These perimeter control policies are informally called firewall rules and are mostly static as they specify:
    - The network location of the egress gateway from which the network edge element at the exit point of a cloud network (e.g., outbound firewall, such as the one at the edge of a cloud) can receive traffic
    - The network location of the ingress gateway to which the incoming traffic that lands at the entry point of the on-premises network edge (e.g., inbound firewall at the entry point to an on-premises network) should be routed:

Firewall: allow 10.100.2.3/30 15443 to 10.1.2.3/30:15443

- Fine-grained network-tier policies – These microsegmentation policies specify the pathways for traffic flowing into and out of the services located within the network subnets at the cloud location or on-premises location.
    - Specify the path on which the outbound traffic from a service or an application (e.g., app 1) can flow. The elements in the path that are specified include the egress gateway at the edge of the cluster and, subsequently, the outbound firewall for the network (cloud network in this example).
    - Specify the path for the inbound traffic into the on-premises network to reach the target application. The elements in the path start from the inbound firewall at the edge of the on-premises network to the ingress gateway in an on-premises cluster to the network subnet where the target service is located.
    - Notably, they specify how traffic can flow “east-west”(i.e., inside of the perimeter). This is in contrast to coarse-grained policies, which specify how traffic can flow “north-south” (i.e., from an external to internal network).
- Identity-tier policies – These are also called mesh-level policies as they are deployed and enforced at the data plane of the service mesh in the reference platform. In the context of the application infrastructure and the example policies that cover traffic flows from Service 1 to Service 2, these are:

```
                selector:
    matchLabels:
      app: service-2
  action: ALLOW
  rules:
  - from:
    - source:
        principals: ["cluster.local/ns/service-1/sa/service-1"]
    to:
    - operation:
        ports: ["443"]
        methods: ["GET"]
        paths: ["/public"]
```

Fig. 4. An Istio Authorization Policy that allows Service 1 to Service 2 on port 443 but only allows it to execute the GET HTTP verb on the "/public" path

This simple example shows some of the advanced capabilities that identity-tier policies can achieve by limiting access based on the application request context. Specifically, this policy limits the application actions to a single HTTP verb on a specific path, but much more sophisticated policy can be implemented as well. See [2] and [3] for detailed overviews.

4.4. Another Application Infrastructure Scenario

Consider another common application scenario in which there is an internal (i.e., within a cluster in the enterprise data center) three-tier application. This application is accessed from outside (through a mobile app or website) through a DMZ. This scenario consists of edge gateways present in the DMZ – an ingress gateway and an egress gateway at the entrance and exit points to and from the data center with firewalls at either side of the gateways. Each of the services that represent the front end and back end (application logic) of the three-tier application have to have a sidecar proxy to enforce policies that pertain to inter-service call requests. This scenario requires the definition and deployment of a combination of network-tier and identity-tier policies that span the various types of gateways and the sidecar proxies. Deploying policies at these multiple locations requires an enterprise-level infrastructure that plays the role of a global control plane, as described in Section 2.1. This is designated as a central coordination infrastructure, as shown in **Fig. 5**.

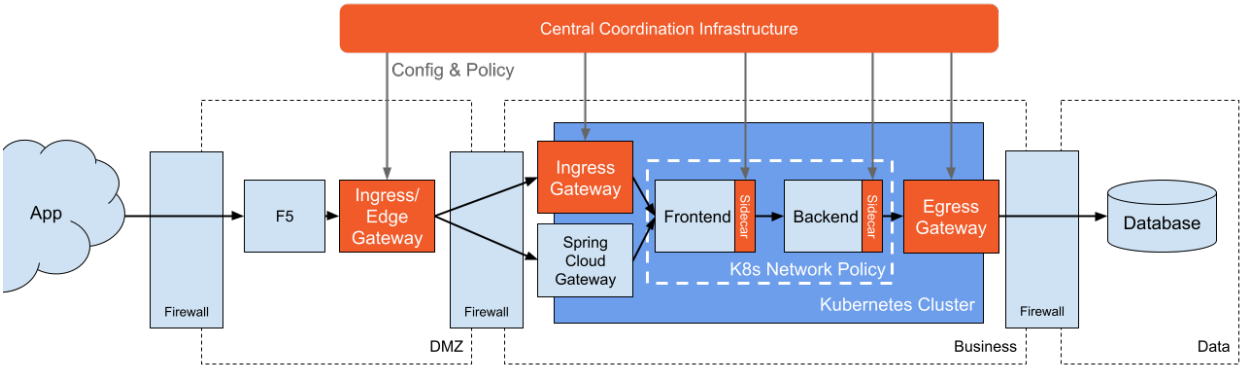

Fig. 5. Policy Deployment for a Three-tier Application

4.5. Functional Roles of Application Infrastructure Elements in Enforcing Policies

This section will show the functionality of each of the application infrastructure elements involved in the policies (e.g., firewalls, gateways, sidecar, transit, and edge proxies) in detail. Since the functionality of firewalls that take part in this context for coarse network-tier policies are well known, this section will focus on the functionality of gateways that take part in fine-grained and identity-tier policies:

- *Sidecar* – Beside each application instance to intercept all traffic into and out of the application and handles "east-west" internal communication between services in the infrastructure. This is the primary use case of the service mesh.
- *Ingress gateway* – Controls how applications in the cluster are exposed outside (e.g., managing what names, certificates, ports, protocols, and application endpoints are served to the world outside of the cluster). Think of this as the service mesh control plane that manages a traditional reverse proxy similar to Spring Cloud Gateway, NGINX, or HAProxy.
- *Egress gateway* – Controls how applications in the cluster communicate with the outside world. This can be used for traditional egress filtering and logging, like a Squid proxy, but can also implement identity-based policy for what is allowed to call out and perform *credential exchange*, or presenting a set of credentials (e.g., an mTLS certificate for a partner API), on behalf of the application so that the application does not need to handle them (e.g., communicating via mTLS with the partner API). Think of this as a next-generation identity-aware Squid proxy.
- *Edge gateway* – Accepts external traffic before the ingress gateway and performs fine-grained load balancing across clusters or sites. It is used to terminate external traffic, enable infrastructure-level failover, deploy blue-green clusters, and facilitate ingress-gateway-per-team deployments without requiring each of those teams to have publicly routable ingress gateways. Think of this as a modern software-based local traffic manager, like F5, that can apply policy per-request rather than per-connection.

4.6. Comparison of Identity-Tier and Network-Tier Policies

While network-tier policies are necessary for geographically distributed application infrastructures and for meeting the compliance requirements of regulators, having a combination of network-tier and identity-tier policies allows for some relaxation of the network-tier policies as any unauthorized traffic flow due to an overlooked network element in the path can be addressed through flexible service identity-tier policies. In order to appreciate the need for the coexistence of both policy tiers, it is necessary to know the characteristics of both tiers of policies. This layering of policies, whose strictness can be tuned per organizational needs at each tier, provides agility and operational ease over status quo perimeter-based models while enhancing the overall security posture of the organization.

4.6.1. Approaches for Deployment and the Limitations of Network-Tier Policies

In this approach, applications and service resources with similar security requirements are grouped into a unique segment, and firewall rules are created to block or allow communication with each group or segment [5]. The segments are created using network layer abstractions (e.g., VLAN IDs or some other tagging approaches), while policies are defined using network address constructs (e.g., IP addresses and ports). Policies apply to subnets (e.g., VLANs) rather than to individual hosts. The assignment of applications to a particular segment can be based on different criteria, such as "all applications with similar security requirements" or "all tiers (web front end, application logic servers, and database servers) associated with a particular application should run in a single segment."

Each segment is protected by gateway devices, such as intelligent switches and routers or next-generation firewalls, which should have the capacity to react and adapt in response to the threats and changes in the application workflows. Segmentation gateways monitor traffic, stop threats, and enforce granular access across east-west traffic (rarely for north-south traffic) within on-premises data centers or cloud regions. The main difficulty with this approach is in mapping the applications' security requirements-based segments to corresponding network segments. Another difficulty is change management. The mapping between applications and network identities that are being statically maintained has to continuously be kept in sync with the operational scenario in which the application's network locations are continuously changing due to performance and security.

More modern cloud-native deployments that leverage techniques like container network interface-driven network policy are good improvements because they provide identity-tier style policies (i.e., policy in terms of identities and non-network-oriented nouns) while implementing that policy at the network layer (e.g., via eBPF policy or BGP propagation rules). These are a strong upgrade from traditional microsegmentation because they tend to result in finer-grained policies that are easier for the organization to manage over time. However, they typically lack the ability to apply policy per request in the context of the application, which is needed to achieve identity-based segmentation.

4.6.2. Prerequisites for the Deployment of Identity-Tier Policies

Identity-tier policies use contextual, application-driven identifiers (e.g., "order processing front-end service can communicate with inventory back-end service") instead of network parameters (e.g., "permit calls from 192.168.10.x subnet to 10.0.0.31"). The identifiers assigned to services at runtime are cryptographic identities, which are used for mutual authentication and authorization during each service request and response.

Deploying identity-tier policies requires a standardized infrastructure for creating, issuing, and maintaining tamper-proof service identities. Some of the components of this infrastructure are outlined below and are also discussed in [5]:

- Creation of application identity: The fundamental requirement to enable this is the assignment of a unique identity to each application or service, just like how each user carries a unique identity (e.g., userid). Prior to the era of cloud-based applications, application requests were validated based on the IP subnet or IP address from which they originated. Since ubiquitous access and multi-clouds have eliminated the concept of network perimeters, authentication and authorization based on those parameters are neither feasible nor scalable. Further, the presence of proxies, network address translations, dynamic infrastructures (e.g., migration of applications between VMs), and load balancers make it impossible for the called application to know the IP address of the calling application in order to make authentication or authorization decisions. A unique application identity is required.
- Establishment of trust in application identity: The created application (workload or service) identity should not be subject to spoofing and should be continuously verifiable. An example of workload identity is a SPIFFE ID [4], which is a string that uniquely and specifically identifies a workload and is encoded as a Uniform Resource Identifier (URI). The SPIFFE ID is carried in a cryptographically verifiable document called a SPIFFE Verifiable Identity Document (SVID). SPIFFE supports multiple SVID formats, but the most commonly used is an X.509 certificate.
- Discovery of application resources: There should be a robust and secure method for discovering all of the application dependencies consumed over the network (e.g., services, SaaS endpoints, network appliances, etc.). This capability is enabled through an authenticated service registry.

These allowable flows can be based on either (a) the structure of the application (i.e., "the front end of application 1 can call the back end of application 1") or (b) a legitimate business transaction (e.g., "order processing application can call the shipping application"). Often, organizations do not know all of the allowable service requests in their infrastructure. However, the observability capabilities of the infrastructure (e.g., the metrics provided by the service mesh) can be leveraged to build a view of "requests made today." From that view, the organization can begin to create fine-grained policies for allowable service requests. Utilizing this observe-and-lock-down methodology builds the organizational processes required to maintain the life cycle of these policies over time.

4.6.3. Advantages of Identity-Tier Policies

Policies based on service and application identities do not use any infrastructure-related variables (e.g., IP addresses, subnets), so they are environment-agnostic and provide the freedom for the services and applications to be migrated to different environments and still maintain the same policies. In other words, there can be a consistent set of policies across cloud providers and on-premises because the policy follows the application rather than the network.

- Identity-tier policies enable the automated testing of policies. Policies that are independent of infrastructure can be tested by merely exercising the application and observing the outcomes (e.g., trace the sequence of service calls and requests or responses instead of configuring the infrastructure correctly for test runs).
- Identity-tier policies enable "policy as code" (PaC). With the availability of tools for the declarative specification of policies through PaC, identity-tier policies can be defined and implemented by incorporating the code into automated workflows, such as CI/CD pipelines.
- Identity-tier policies enable granular (fine-grained) access control by providing visibility into application call sequences/interdependencies and data flows through request-level tracking, which enables the enforcement of security policies for application traffic that is both north-south and east-west, irrespective of the environment (e.g., corporate data center or cloud infrastructure).

Additional advantages include:

- Write once, enforce everywhere – This means that policy can span environments and topologies (i.e., write a policy once, and enforce it everywhere) rather than bespoke policies per environment.
- Human-readable primitives – The written policies use human-understandable primitives (e.g., "service A can call service B") rather than network-oriented primitives (e.g., "10.1.2.3/30 is allowed to call 10.100.2.3/30 on port 8080"). This context is critical since the lack of context for rules is a key reason for the lack of agility around traditional network policy.
- Contextual intent is codified in a single policy – There is a single policy, not a set of policies that need to be pieced together to understand their intent. A human can read a policy like "the front-end service is allowed to call 'GET /foo' on the back-end service" and understand the access that the policy intends to convey even if, for example, the front end is deployed in the cloud and the back end is deployed on-premises. It is significantly harder to read a set of network peering and firewall rules that allow communication across the DMZ for a set of subnets and understand the access that the policy intends to convey. In turn, this means it is harder to write the wrong policy and easier for a human to understand when a policy is incorrect.

Identity-tier policies enable only valid network traffic between the various component services of the application due to the mutual authentication and authorization of the service identities, thus enabling the goals of ZTNA to be met.

5. Summary and Conclusions

This document provides guidance for realizing a ZTA for cloud-native application platforms (microservices with a service mesh infrastructure) in the context of an enterprise environment in which applications are hosted in multi-cluster and multi-cloud deployments. A ZTA consists of deployment artifacts that enforce zero trust principles, which is only possible with robust, flexible, scalable, and granular policies that cover all enterprise resources. A policy framework that consists of network-tier and identity-tier policies to meet these goals has been proposed in this document.

The artifacts needed for the definition, deployment, and enforcement of these policies have been discussed along with examples of network-tier policies and identity-tier policies. The applicability of these policies in modern enterprise application infrastructures is also illustrated. Finally, the policies that belong to the two tiers are compared in terms of their advantages and limitations, and the critical role of identity-tier policies for realizing a ZTA in the context of modern cloud-native application infrastructures is emphasized.

765 ## References

766 [1] Rose S, Borchert O, Mitchell S, Connelly S (2020) Zero Trust Architecture. (National
767 Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP)
768 NIST SP 800-207. https://doi.org/10.6028/NIST.SP.800-207
769 [2] Chandramouli R, Butcher Z (2020) Building Secure Microservices-based Applications
770 Using Service-Mesh Architecture. (National Institute of Standards and Technology,
771 Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-204A.
772 https://doi.org/10.6028/NIST.SP.800-204A
773 [3] Chandramouli R, Butcher Z, Aradhna C (2021) Attribute-based Access Control for
774 Microservices-based Applications using a Service Mesh. (National Institute of Standards
775 and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-204B.
776 https://doi.org/10.6028/NIST.SP.800-204B
777 [4] SPIFFE (2022) *SPIFFE Concepts*. Available at https://spiffe.io/docs/latest/spiffe-
778 about/spiffe-concepts/
779 [5] Chandramouli R (2022) Guide to a Secure Enterprise Network Landscape. (National
780 Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP)
781 NIST SP 800-215. https://doi.org/10.6028/NIST.SP.800-215
782