NIST Special Publication 800
NIST SP 800-233 ipd

Service Mesh Proxy Models for Cloud-Native Applications

Initial Public Draft

Ramaswamy Chandramouli
Zack Butcher
James Callaghan

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-233.ipd

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

NIST Special Publication 800
NIST SP 800-233 ipd

Service Mesh Proxy Models for Cloud-Native Applications

Initial Public Draft

Ramaswamy Chandramouli
Computer Security Division
Information Technology Laboratory

Zack Butcher
Tetrate, Inc.

James Callaghan
control-plane.io, Inc.

This publication is available free of charge from:
https://doi.org/10.6028/NIST.SP.800-233.ipd

July 2024

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

Publication History
Approved by the NIST Editorial Review Board on YYYY-MM-DD [Will be added to final publication.]

How to Cite this NIST Technical Series Publication:
Chandramouli R, Butcher Z, Callaghan J (2024) Service Mesh Proxy Models for Cloud-Native Applications. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-233 ipd. https://doi.org/10.6028/NIST.SP.800-233.ipd

Author ORCID iDs
Ramaswamy Chandramouli: 0000-0002-7387-5858

Public Comment Period
July 19, 2024 – September 3, 2024

Submit Comments
sp800-233-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

Additional Information
Additional information about this publication is available at https://csrc.nist.gov/pubs/sp/800/233/ipd, including related content, potential updates, and document history.

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

Abstract

The service mesh has become the de-facto application services infrastructure for cloud-native applications. It enables the various runtime functions (network connectivity, access control etc.) of an application through proxies which thus form the data plane of the service mesh. Depending upon the distribution of the network layer functions (L4 & L7) and the granularity of association of the proxies to individual services/computing nodes, different proxy models or data plane architectures have emerged. The purpose of this document is to develop a threat profile for each of the data plane architectures through a detailed threat analysis in order to make recommendations for their applicability (usage) for cloud-native applications with different security risk profiles.

Keywords

proxy model; data plane architecture; service mesh; threat profile; cloud-native application.

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: sp800-233-comments@nist.gov

Table of Contents

Acknowledgments

The authors would like to express their thanks to Francesco Beltramini of control-plane.io for participating in our discussions and providing his valuable perspectives.

Executive Summary

Run-time services for Cloud-native applications, consisting of multiple loosely coupled components called microservices, are sometimes provided through a centralized infrastructure called a service mesh. These services include secure communication, service discovery, resiliency, and authorization of application communication. These services are mainly provided through *Proxies* that form the *data plane* of the service mesh, the layer that handles application traffic at runtime and enforces policy.

The functions that the proxies provide can be broadly categorized into two groups, based on the OSI model's network layer to which those functions pertain to. These groups are: Layer 4 ("L4") and Layer 7 ("L7"). In majority of deployments of service mesh in production environments today, all proxy functions (providing services in both L4 and L7 layers) are packed into a single proxy that is assigned to a single microservice. This service mesh proxy model is called a *sidecar proxy model* since the proxy is not only associated with a single service but is implemented to execute in the same network space as the service.

However, performance and resource considerations have led to the exploration of alternate proxy models which involve not only splitting up of L4 and L7 functions into different proxies (instead of a single proxy) but also the association or assignments of these proxies to either a single service or a group of services, thus enabling the proxies to be implemented at different locations - at the granularity of a node rather than at the level of services. Though different models are theoretically possible, we consider only those service mesh proxy models in the data plane implementation of commonly used service mesh offerings, at different stages.

We then consider a set of potential/likely threats to various proxy functions. Each of the threats may result in different types of exploits in different proxy models. This variation is due to several factors such as: attack surface (communication patterns to which a particular proxy is exposed), number of clients (services) served and OSI layer functions they provide (e.g., L7 functions are more complicated and likely to have more vulnerabilities than L4 functions). The two main contributions of this document are as follows:

1. The nature of exploits possible for each threat in each of the proxy models are characterized by assigning scores to the impact and likelihood of each of these threats in each of the proxy models or architectural patterns resulting in a threat profile associated with each architectural pattern or proxy model of service mesh.
2. Each threat profile inherently has a built-in set of security tradeoffs at an architectural level. The implications of these tradeoffs in meeting the requirements associated with security risk profile of different cloud-native applications are analyzed to make a broad set of recommendations towards specific architectural patterns that are appropriate for applications with different security risk profiles.

1. Introduction

The service mesh, an application service infrastructure is now an integral part of the overall application infrastructure of cloud-native applications, typically consisting of multiple loosely coupled services or microservices. The infrastructure services or functions provided by a service mesh during application runtime are provided by entities called proxies which constitute the data plane of the service mesh. In addition, the service mesh consists of another architectural component called the control plane which supports the functions of the data plane through interfaces to define configurations, inject software programs and provide security artifacts such as certificates.

Based on performance and security assurance data gained over the deployment of service mesh for the last several years, various configurations for proxies are being developed and tested. These configurations are based on the OSI layer functions they provide (see section 1.1 L4 and L7 functions of a proxy) and the granularity of association between a proxy and services and go by the name of proxy (implementation) models. Since proxies are the predominant entities of the data plane of a service mesh, these various proxy models are also called data plane architectures.

1.1. L4 and L7 Functions of Proxies

To understand proxy models, there are two aspects we should look at. They are:

Proxy Functions: The functions that a service mesh's proxies provide can be broadly categorized into two groups, based on the OSI model's layer [1] to which those functions pertain to. These groups are: Layer 4 ("L4") and Layer 7 ("L7"). The associated proxies are called L4 proxies and L7 proxies respectively.

Granularity of Association: A proxy can be associated with a single microservice instance, an entire service or it can be deployed to provide functions for a group of services. Depending upon the nature of this association, a proxy may execute within the same network space as the service, or it can execute at the same node where the group of services to which caters to run or in an independent node (dedicated to just proxies where no application services run).

The study of proxy functions (the first topic above) in turn requires us to go into fundamentals a little bit and look at what OSI's L4 and L7 layers are, from the network stack point of view and the specific network services provided by those layers.

The OSI model [1] is a useful abstraction for thinking about the functions required to serve an application over the network. It describes seven "layers", from the physical wires connecting two machines (Layer 1 – L1 – the physical layer) all the way up to the application itself (Layer 7 – L7 – the application layer). When facilitating the communication of cloud-native applications (e.g., two microservices making HTTP/REST calls to each other), we care primarily about layers 3, 4, and 7; A brief overview of the functionality of these layers are:

- Layer 3 ("L3"), the network layer, facilitates baseline connectivity between two workloads or service instances. In nearly all cases, the Internet Protocol (IP) is used as the layer 3 implementation.
- Layer 4 ("L4"), the transport layer, facilitates the reliable transmission of data between workloads on the network. It also includes capabilities like encryption. TCP and UDP are commonly used L4 implementations, where TLS (*transport layer security* – named after the OSI model) provides encryption.
- Layer 7 ("L7"), the application layer, which is where protocols like HTTP live – in user applications themselves (e.g., HTTP web servers, SSH servers).

With respect to the layers above, in cloud native environments, a service mesh's proxies are:

- Are agnostic to L3, so long as microservice instances can communicate at L3 and the proxy can communicate with the mesh's control plane.
- At Layer 4 (L4): connection establishment, management, and resiliency (e.g., connection-level retries); TLS (encryption in transit); application identity, authentication, and authorization; access policy based on network 5-tuple (source IP address and port, destination IP address and port, and transport protocol).
- At Layer 7 (L7): service discovery, request-level resiliency (e.g., retries, circuit breakers, outlier detection); and application observability.

What we have seen so far is one aspect of proxy model or data plane architecture – i.e., proxy functions. The other aspect as we alluded to earlier is the proxies' granularity of association to services.

1.2. Objective & Target Audience

This document will give a brief overview of the 4 data plane architectures (proxy models) being pursued by a range of service mesh implementations today. It will then develop threat profiles for different proxy models through a detailed threat analysis involving ten types of common threats. These threat profiles will be used to make a set of recommendations regarding their applicability (usage) for cloud-native applications with different security risk profiles. The target audience for these recommendations is:

- Infrastructure owners and platform/infrastructure engineers (and their team heads) building to build and deploy a secure run-time environment for applications by choosing the right architecture for their environment given the risk factors of the applications they'll be running (and the resulting security risk profile).
- Personnel in charge of infrastructure operations to familiarize them with the details of the various building blocks of the proxy models or data plane architectures (and their associated functions and interactions) to troubleshoot in the event of performance (availability) and security issues.

1.3. Relationship to Other NIST Documents

This document can be used as an adjunct to NIST Special Publication (SP) 800-204 series of publications [2,3,4,5], which offer guidance on providing security assurance for cloud-native applications integrated with a service mesh from the following perspectives: strategy, configuration, and development/deployment paradigm. However, this document focuses on the various configurations of the application service infrastructure elements (i.e., proxies) and the resulting architectures (i.e., data plane architecture of the service mesh) that have different security implications for the application that is hosted under each of these configurations.

1.4. Document Structure

This document is organized as follows:

- Section 2 provides a list of typical capabilities of the data plane of the service mesh under three headings (security, observability and traffic management) and the corresponding L4 and L7 proxy functions implemented under those capabilities.
- Section 3 provides a brief overview of the four architectural patterns called the proxy models or data plane architectures.
- Section 4 discusses proxy model threat scenarios and gives a roadmap of the threat analysis methodology adopted in this document for evaluating the threat profile score for the four data plane architectures.
- Section 5 provides a detailed threat analysis for the four data plane architectures by assigning scores to impact and likelihood factors associated with each threat and using them to arrive at the overall threat score.
- Section 6 provides the recommendations for the applicability (usage) of each of the 4 data plane architectures for cloud-native applications of different security risk profiles based on their security requirements.
- Section 7 provides the summary and conclusions.

2. Typical Service Mesh Data Plane Capabilities and Associated Proxy Functions

Since examining the security tradeoffs of the proxy models (data plane architectures) is part of our methodology in this document, we have to look at implementations of the various capabilities (under the umbrella of Security, Observability and Network Traffic Management) that result as L4 and L7 functions in proxies. To arrive at the totality of proxy functions, we need to analyze for each capability, which category (L4 vs L7) it falls in to, and the granularity of the function that it provides at L4 and L7 levels.

Table 1 - Security Capabilities

Capability	L4 Function(s)	L7 Function(s)
Service-to-service authentication	SPIFFE, via mTLS certs. Control plane issues a short-lived X.509 encoding the pod's service account identity.	N/A—service identity in a service mesh is usually based on TLS only.
Service-to-service authorization	Network-based authorization, plus identity-based policy, e.g.: *A* can accept inbound calls from only "10.2.0.0/16"; *A* can call *B.*	Full policy, e.g.: *A* can GET /foo on *B* only with valid end-user credentials containing the READ scope.
End-user authentication	N/A—we can't apply per-user settings.	Local authentication of JWTs, support for remote authentication via OAuth and OIDC flows.
End-user authorization	N/A—see above.	Service-to-service policies can be extended to require end user credentials with specific scopes, issuers, principal, audiences, etc—but it cannot be used for full user-to-resource access control. Full user-to-resource access should be implemented using external authorization.
Mesh proxy's External Authorization API (ext_authz)	Cannot perform any per-request policy; ext_authz API is only configurable for L7 traffic.	Enforce per-request policy with decisions from an external service, e.g., OPA.

Table 2 - Observability Capabilities

Capability	L4 Function(s)	L7 Function(s)
Logging	Basic network information: network 5-tuple, bytes sent/received, etc.	Full request metadata logging, in addition to basic network information.
Tracing	Not today; possible eventually, with HBONE.	Mesh proxy participates in distributed tracing.
Metrics	TCP only (bytes sent/received, number of packets, etc).	L7 RED metrics: rate of requests, rate of errors, request duration (latency).

Table 3 – Traffic Management Capabilities

Capability	L4 Function(s)	L7 Function(s)
Load balancing	Connection level only. See TCP traffic shifting task.	Per request, enabling e.g. canary deployments, gRPC traffic, etc. See HTTP traffic shifting task.
Circuit breaking	TCP only.	HTTP settings in addition to TCP.
Outlier detection	On connection establishment/failure.	On request success/failure.
Rate limiting	Rate limit on L4 connection data only, on connection establishment, with global and local rate limiting options.	Rate limit on L7 request metadata, per request.
Timeouts	Connection establishment only (connection keep-alive is configured via circuit breaking settings).	Per request.
Retries	Retry connection establishment	Retry per request failure.
Fault Injection	N/A—fault injection cannot be configured on TCP connections.	Full application and connection level faults (timeouts, delays, specific response codes).
Traffic Mirroring	N/A—HTTP only	Percentage-based mirroring of requests to multiple backends.

It's important to note that L7 functions carried out by proxies are much more complex than L4 functions as the latter are carried out in lower layers of OSI stack involving protocols such as IP and TCP. For example, parsing a TCP stream for L4 functionality requires simply decoding a fixed set of bytes as integers (the packet header), while handling HTTP requests for L7 functionality requires decoding HTTP headers including complex string parsing and compression with variable amounts of data. Further, that data dealt with in an L7 function is user-supplied (i.e., can be controlled by an attacker), while the TCP data at L4 is typically system-supplied as part of routing a request to your infrastructure – there's less room to embed malicious data without breaking the system itself. As one case study, the proxy Envoy is used as the data plane by several service mesh implementations: historically, majority of Envoy vulnerabilities have been in L7-function-related code compared to L4-function-related code.

3. Proxy Models (Data plane Architectures) in Service Mesh Implementations

As we had briefly seen before, different data plane architectures or proxy models in service mesh are a consequence of the following parameters.

- Delineation of L4 and L7 functions
- Nature of association of a proxy to service instances (1:1 or 1:N)

In this section, an overview of the building blocks of different data plane architectures is undertaken to facilitate the threat analysis that follows in section 5. Before we list the different data plane architectures (also called different iterations of service mesh implementations) that have been commonly implemented, it is in order to look at as to why these different architectures were necessitated in the first place. These iterations were driven by the adoption of mesh across a variety of use cases, necessitating different tradeoffs in terms of performance, reliability, and security across a variety of organizations with different application risk profiles. It must be mentioned, however, that in spite of these different operating scenarios, the first model listed here, i.e., the sidecar model, has been the primary predominant method of delivering the capabilities of service mesh for several years.

The various alternate data plane architectures, including the one with widespread deployment at present, are:

- “L4 and L7 per Service Instance” - Side-car Model (DPA-1)
- “Shared L4 – L7 per Service” (DPA-2) - A shared L4 proxy per node, i.e., shared among all applications that execute on the same physical host, with L7 proxies dedicated per service account or namespace.
- “Shared L4 and L7” (DPA-3) - A shared L4 *and* L7 proxy per node, i.e., shared among all applications on the same physical host.
- “L4 and L7 within Application (gRPC proxy-less model)” (DPA-4) – Both L4 and L7 functions instead of being implemented in stand-alone proxies are part of the application server itself, e.g., frameworks such as gRPC, Java Spring etc.

It must be mentioned that though the last architectural pattern does not have distinct entities such as proxies, all the service mesh capabilities delivered by proxies are enabled by the frameworks mentioned above.

3.1. L4 and L7 Proxy per Service Instance – Sidecar Model (DPA-1)

The first and most common service mesh data plane architecture today dedicates a proxy that has the capability to implements both L4 and L7 functions for each application (service) instance. This is also called a "sidecar model" since the proxy sits beside every instance of every service. The security model here is simple: the proxy holds one identity (for the service it's deployed beside) and resides in the same trust domain as the application (in Kubernetes, it exists in the same pod; on a VM, it's deployed in the same VM as the service itself). The service and the proxy communicate with each other through the "local host interface" instead of through a network socket. However, the proxy itself presents a larger attack surface than the service because it implements the complex L7 functions. An example of a data plane architecture is the one that is implemented in the Istio service mesh with an envoy proxy deployed per pod that performs both L4 and L7 functions.

A schematic diagram of this architecture is shown in Figure 1.

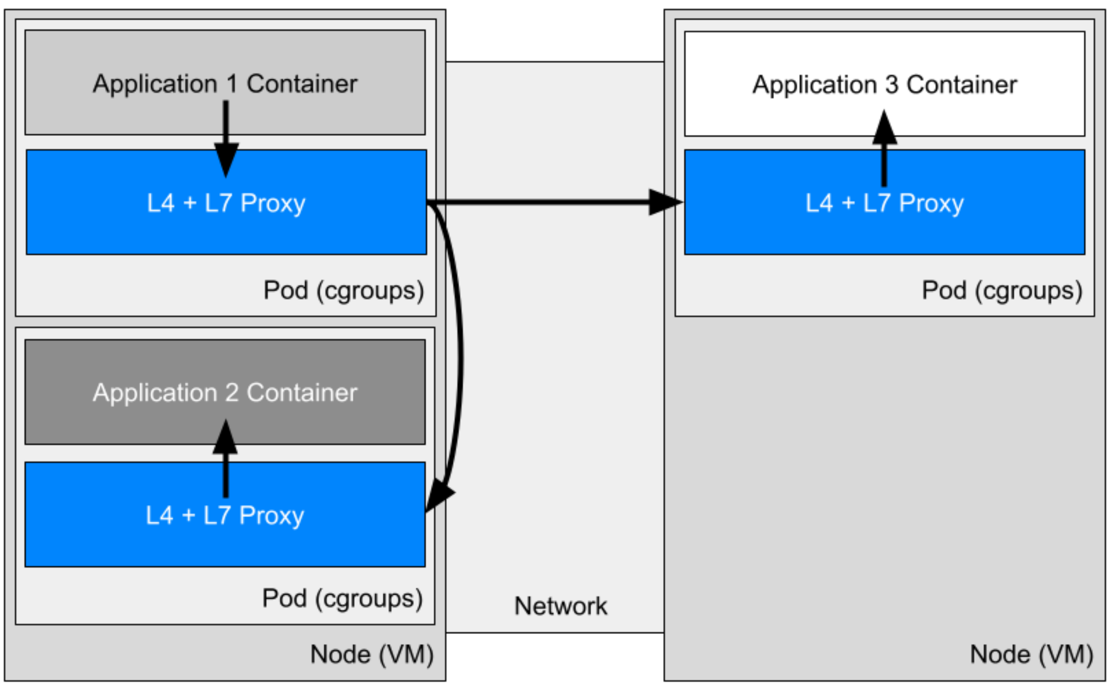

Figure 1 – L4 and L7 Proxy per Service Instance (Side Car Model) (DPA-1)

3.2. Shared L4 – L7 per Service Model (DPA-2)

In this architecture, there is a shared L4 proxy per node, i.e., shared among all service instances that execute on the same physical host, with L7 proxies dedicated per service account. This is also called "ambient mode". A variation in this architecture is to dedicate a L7 proxy for an entire namespace. This is not desirable from a security viewpoint based on the same reasons we recommend against shared service account for entire namespace [2] and hence not considered for threat analysis in this document. An example of implementation of this data plane architecture is the Istio Ambient where the per node L4 proxy is called Ztunnel proxy and per service account L7 proxy is called Waypoint proxy [6,7,10,11,13].

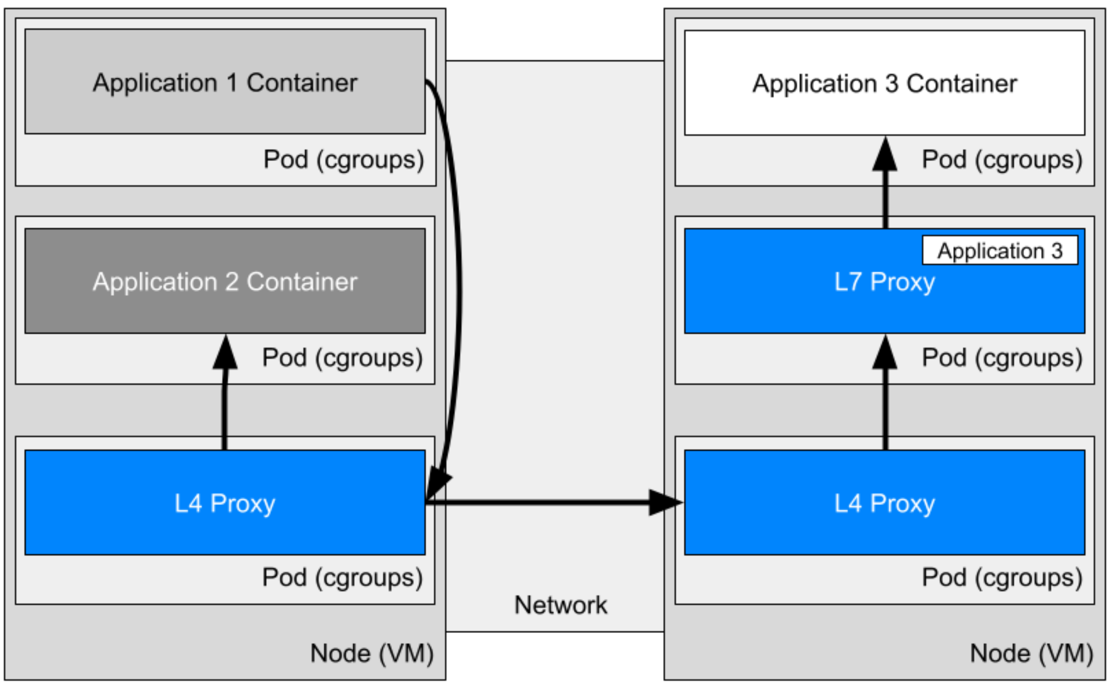

Figure 2 – Shared L4 - L7 per Service Model (DPA-2)

3.3. Shared L4 and L7 Model (DPA-3)

In this architecture, the L4 and L7 functions are implemented on a per node basis. There is a shared L7 proxy per node, i.e., shared among all service instances that execute on the same physical host and provides L7 functions for all services in that node. However, the L4 functions such as traffic routing can be performed not by proxies but by in-kernel programs (e.g., eBPF programs) or the mesh proxy. An example of this data plane architecture is the Cilium service mesh which deploys the Envoy proxy as L7 proxy based on its CiliumEnvoyConfig specification [8,9,12].

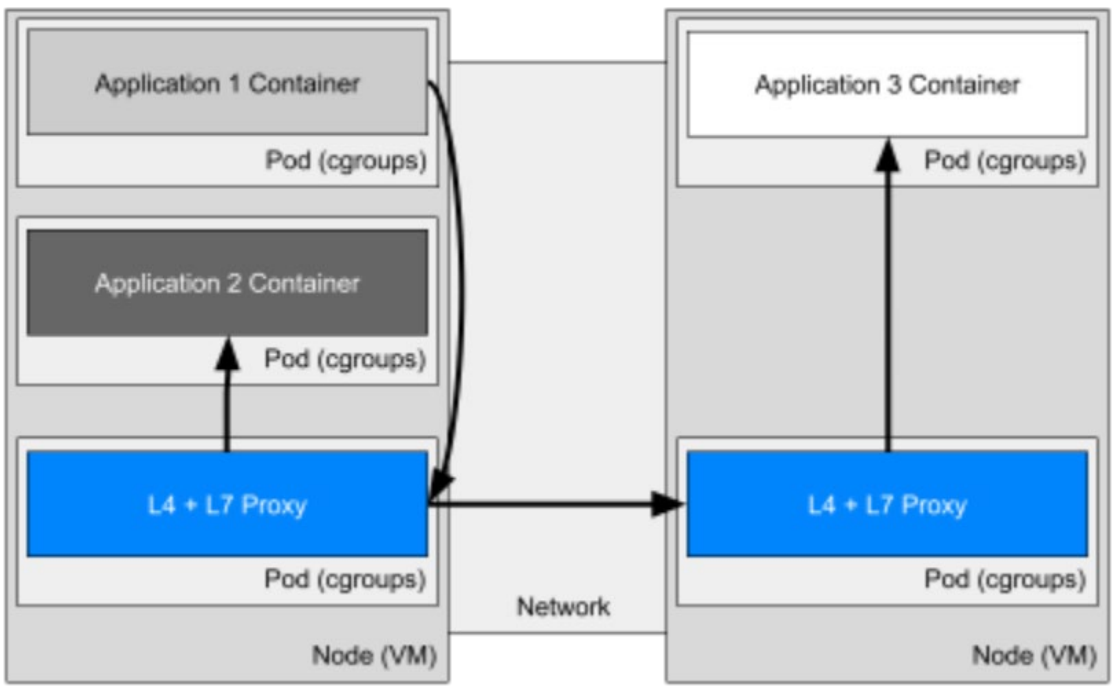

Figure 3 – Shared L4 - L7 Model (DPA-3)

3.4. L4 and L7 Part of the Application Model (DPA-4)

This is a data plane architecture that does not have any proxies. The service mesh control plane dynamically configures proxies using a set of discovery APIs collectively known as xDS APIs. The gRPC client library for applications provides extensive support for the xDS APIs. Leveraging this feature, the service mesh control plane can program L4 and L7 functions into this library in the service container. These gRPC libraries can then provide the L4 and L7 functionality (in general all policy enforcements) to the workloads or service instances to which they are integrated with, thus replicating the exact services which the L4 and L7 proxies provide to those workloads [14].

The architecture diagram of the gRPC proxyless data plane architecture for a service mesh is given below:

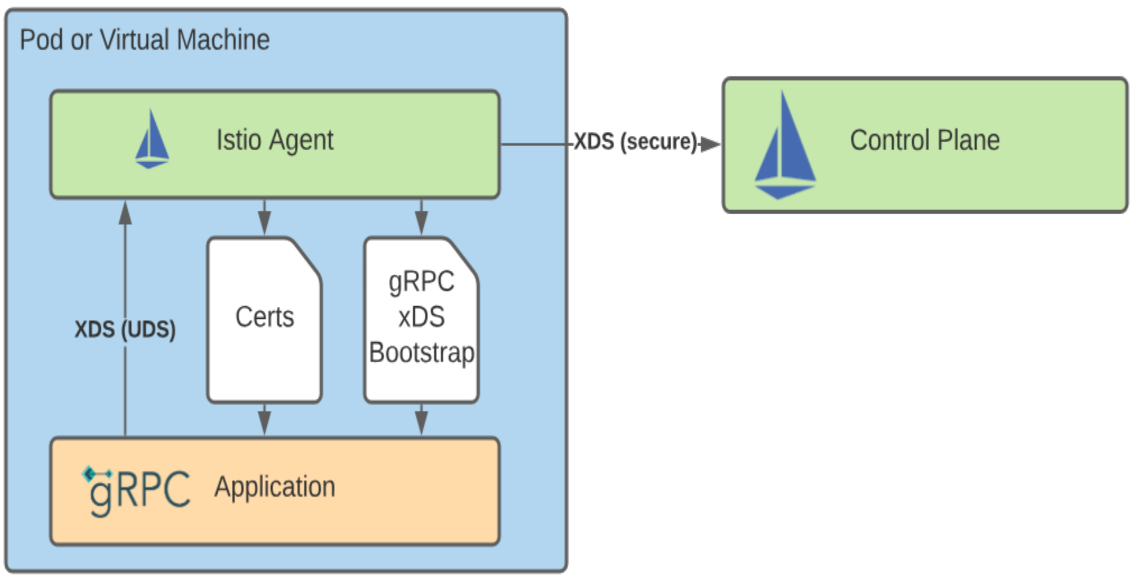

Figure 4 – L4 and L7 Part of the Application Model (gRPC proxyless Model) (DPA-4)

4. Data Plane Architectures Threat Scenarios and Analysis Methodology

We are studying service mesh deployed in a Kubernetes cluster. Assumption that no human can directly access the cluster, achieved via k8s RBAC; only interaction with the cluster is via CI/CD system controlled declarative configuration in a version-controlled repository with a multi-step approval process to change that configuration (including the approval of each change by at minimum one other human).

We start by identifying a variety of access by external threat actors, internal threat actors, and malicious co-tenants.

External threat actors include:

- Compromised workload (application) container, e.g., via a supply chain attack
- Compromised node L4 proxy or CNI
- Compromised node L7 proxy
- Compromised node with limited privileged access, e.g., a container breakout
- Root compromise of node, e.g., a container breakout chained with exploitation of a privilege escalation vulnerability.
- Network access to the Kubernetes API server

Internal threat actors include:

- Cluster admins, who have wide-ranging rights to view the cluster and approve changes to the version-controlled repository; they may even have direct access to the Kubernetes cluster, e.g., via a break-glass debugging account – such super-accounts should generate detailed audit records of their usage.
- Application developers, who can build images and approve configuration that goes into the cluster.
- Infrastructure engineers, who have permission to deploy and configure the mesh – again, gated by the version-controlled repository's approval process.
- Compromised network infrastructure between nodes, e.g., un-encrypted cross-data center communication

Finally, malicious co-tenants – in general k8s is not a hard multi-tenant system and we recommend isolating tenants from each other with stronger boundaries. In this context, a malicious co-tenant would fall into one of the internal threat actor personas above.

In the context of these threat actors, we introduce the following threats as a minimum set to consider in your environment as they relate to the service mesh:

1. Compromised L4 proxy
2. Compromise of the Application Container
3. Compromise of Business Data

4. Compromised L7 proxy
5. Compromise of shared L7 Proxy
6. Outdated Client Libraries in Applications
7. Denial of Service
8. Resource Consumption
9. Privileged L4 Proxy
10. Bypassing Traffic Interception

In the next section three we will evaluate the impact of these threats on the components of the data plane architecture for each of the four that we have taken up for consideration in this document.

4.1. Threat analysis Methodology

We first identify 10 potential threats to the components that make up the four architectural patterns for the proxy model or data plane architecture. For each threat, we describe how the functionality of each component of the architecture is adversely affected by the threat and then rate the impact and likelihood of their occurrence, justifying each rating. We have chosen three values for ratings - *low, medium, high*. The values assigned to these ratings are relative to other data plane architectures and are not absolute values based on a metric. For example, the assignment of the rating value “High” for the likelihood parameter for a threat does not imply that the threat is highly likely in all situations; it means that this threat is likeliest to be executable against that architecture *relative to the other architectures under discussion*.

For each threat and architecture, we evaluate the *impact (I)* of the exploitation of that threat along with the *likelihood (L)* of that threat being exploited. As we already stated, for both parameters we give a rating of low, medium, and high which we translate to numeric scores 1, 2, and 3 respectively. By multiplying these together, *I * L*, we can get a indication of how important that threat is and therefore the necessity to mitigate that threat relative to other architectures under discussion. Summing up the values of this indicator for all 10 potential threats, we obtain an indication of the threat profile for that architectural pattern.

For those threats whose impact and likelihood are same irrespective of the architecture – in other words, the threats are agnostic to the architecture, we assign a score of 1 for impact and 1 for likelihood due the fact that we stated earlier - these scores are relative scores and not absolute scores.

5. Detailed Threat Analysis for Data Plane Architectures

In this section, we analyze the various potential proxy-functions targeted threats (both for L4 & L7 proxies or the libraries implementing the associated functions), the relevant proxy function that is impacted, the degree of impact, the likelihood of the threat occurring for each of the data plane architectures discussed in sections 3.1 to 3.4.

Recapping from Section 4, the 10 threats with their identifiers added that are considered for analysis in this section are:

Compromised L4 proxy (TR-1)

Compromise of the Application Container (TR-2)

Compromise of Business Data (TR-3)

Compromised L7 proxy (TR-4)

Compromise of shared L7 Proxy (TR-5)

Outdated Client Libraries in Applications (TR-6)

Denial of Service (TR-7)

Resource Consumption (TR-8)

Privileged L4 Proxy (TR-9)

Data plane (Service Mesh) Bypassed (TR-10)

The organization of this section is as follows:

Section 5.1 will analyze each threat for the “L4 and L7 Proxy per Service Instance – Sidecar Model (DPA-1)” and come up with the overall threat score.

Section 5.2 will analyze each threat for the “Shared L4 - L7 per Service Model (DPA-2)” and come up with the overall threat score.

Section 5.3 will analyze each threat for the “Shared L4 - L7 Model (DPA-3)” and come up with the overall threat score.

Section 5.4 will analyze each threat for the “L4 and L7 Part of the Application Model (gRPC proxyless Model) (DPA-4)” and come up with the overall threat score.

5.1. Threat Analysis for L4 and L7 Proxy per Service Instance – Sidecar Model (DPA-1)

Each of the threats to the data plane of the service mesh is denoted using the mnemonic TR-x where TR stands for threat and x for the threat sequence number.

5.1.1. Compromised L4 Proxy (TR-1)

<u>Threat Description</u>: Compromised L4 proxy (or L4 functions in the case of sidecar proxy with combined L4 and L7 functions) leads to leaked identities for every workload (service) running on the node.

<u>Proxy Function Impacted</u>: Sidecar proxies negotiate mTLS connections (for communicating with any other service) on behalf of only the single workload it is associated with. In order to compromise key material and identity documents (threat targets) for multiple workloads, multiple proxy (sidecar) instances would need to be compromised.

<u>Impact Score=1</u>: Because of the nature of impact discussed above (i.e., Single workload / single identity being affected), this threat is assigned the impact score of 1.

<u>Likelihood Score=2</u>: Code relating to L7 functions is present to be exploited, if it can be triggered. In a pure L4 proxying case it *should* not be triggerable, but this relies on correct configuration from users and the service mesh implementation.

5.1.2. **Compromised Application Container (TR-2)**

<u>Threat Description</u>: Compromised application container (e.g., via a supply chain attack – during development phase) leads to takeover of identity associated with that application.

<u>Proxy Function Impacted</u>: Proxies run in the same network space (same pod in Kubernetes environment) as the application container, meaning that a compromise of the application container (hosting the service instance) can easily lead to a compromise of any key material (full access to key material pertaining to the identity of the service) possessed by the proxy.

<u>Impact Score=2</u>: Because of the nature of impact discussed above (i.e., Single workload / single identity being affected), this threat is assigned the impact score of 2. Even though only a single identity is compromised, like TR-1, this has a higher impact score as the application itself must be updated. A compromised proxy can be remediated without requiring the application itself to be updated, so there’s a higher chance a central team can successfully remediate a compromise without involving application teams.

<u>Likelihood Score=1</u>: Same regardless of Architecture.

5.1.3. **Compromise of Business Data (TR-3)**

<u>Threat Description</u>: Compromised identity is used to pivot through the infrastructure, in order to compromise the confidentiality, integrity or availability of business data.

<u>Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)</u>: Same regardless of architecture -- this is the fundamental risk of identity-based policy and is why we need to practice the principle of least privilege (PoLP). The telemetry provided by the service mesh (regardless of architecture) is invaluable for understanding communicating in your system and creating accurate access policies (thereby implementing PoLP)

5.1.4. **Compromised L7 Proxy (TR-4)**

Threat Description*:* Vulnerability in L7 processing stack of the service mesh proxy. As L7 processing is inherently more complex, there is a higher probability for vulnerabilities to arise in this part of the stack, as supported by historical CVE data.

Proxy Function Impacted*:* No separation between L4 and L7 processing. It can be argued that any exploitable vulnerability in a sidecar proxy can lead to the compromise of all identities in the mesh, however as this would involve more individual proxy instances being compromised, it may be more difficult for an attacker to accomplish this feat undetected.

Impact Score=1: A single workload is impacted (either leaking credentials, or becoming unavailable due to DoS, depending on the type of L7 attack). The same exploit could be used against all sidecars in the mesh with applications opting in to L7 behavior, resulting in compromise of all identities (Impact 3); in practice this requires many more events than any other architecture, increasing our likelihood of detecting and responding to the event in a timely manner.

Likelihood Score=1*:* Full L7 capability is available in the proxy, meaning a relatively large attack surface is exposed; in practice for the service mesh use case, however, it tends to be the HTTP processing that is targeted. If the application is using L7 mesh capabilities, they would be vulnerable to exploit.

5.1.5. **Compromise of Shared L7 Proxy (TR-5)**

Threat Description*:* Co-tenant exploits L7 traffic processing vulnerability in shared proxy, to affect the confidentiality, integrity or availability of traffic to/from another workload running on the same node.

Proxy Function Impacted*:* Because the proxy is dedicated per application, impact on availability is limited to the resource constraints imposed by the scheduling system (e.g. Kubernetes). Confidentiality is impacted the same as if another application itself is compromised -- i.e. containers provide some guarantee, micro-VMs provide a stronger degree of isolation, full blown VMs the strongest.

Impact Score=1: For noisy neighbors – other L7 proxies on the same host that are compromised – Impact limited by underlying scheduling and resource constraint system (e.g. k8s, VM sizing, etc). Identical across all architectures: for a shared ingress gateway, all services exposed on that gateway would be impacted (Impact 2); for a shared egress gateway, all services utilizing the egress gateway are impacted (Impact 3; typically only a single deployment of egress gateways is used).

Likelihood Score=1*:* The sidecar itself is not a shared proxy – by its nature it is dedicated to an individual application. In this case TR-5 refers to both noisy neighbors, other proxies on the same node causing a denial of service, as well as shared ingress or egress gateways. Noisy neighbors are mitigated based on the degree of isolation of the host (container vs micro-VM vs

VM). Likelihood of exploiting a shared L7 ingress or egress gateway is the same across all architectures.

5.1.6. **Outdated Client Libraries in Applications (TR-6)**

Threat Description: Client libraries are not updated frequently or consistently across the estate of microservices, leading to potential vulnerabilities and weaknesses that can be exploited.

Proxy Function Impacted: The proxy's Infrastructure code is decoupled from application code.

Impact Score=1: The mesh infrastructure is separate from the application itself, therefore it's not impacted by application vulnerabilities directly. Instead, a compromised app would use the (functioning) mesh to hijack the application's identity (see threat on compromised app container, compromised identity). Some application vulnerabilities can be mitigated via policies enforced by the mesh, for example: mesh enforced WAF policy can help mitigate an app vulnerability like Log4j while the organization is patching applications.

Likelihood Score=1: Same regardless of architecture.

5.1.7. **Denial of Service (TR-7)**

Threat Description: Conventional Denial of Service threat.

Proxy Function Impacted: Because the proxy is per app instance, a DoS needs to be executed per app. Because the proxy shares resources with the app, a DoS on the mesh data plane directly competes for resources with the app instance itself. The overall blast radius of the DoS is as strong as the underlying isolation mechanism protecting workloads (pods) from each other (VMs, micro-VMs, containers, etc.).

Impact Score=1: Single instance of a single app

Likelihood Score=1: L4 and L7 code is able to be exploited; however the attack must be executed across each instance of the target (there's not a central resource to target to achieve a DoS, other than a shared ingress gateway which is identical across all architectures under discussion).

5.1.8. **Resource Consumption (TR-8)**

Threat Description: Overall resource consumption by the data plane of the service mesh infrastructure.

Proxy Function Impacted: Because sidecars are a separate process and are dedicated per app, they have the worst overall resource consumption:

configuration that's identical across all apps must be held by the data plane per app, and can't be shared.

static overhead of the sidecar data plane implementation itself (e.g. constant RAM usage, constant CPU overhead, and so on) is duplicated per app instance, and can't be amortized over all apps on the node

In part this isolation is what allows sidecars to have lower impact and likelihood across many of the other threats identified here.

Impact Score=3: Highest resource usage of all options, though good configuration can help mitigate the impact (even then, in well-configured environments sidecars will consume the most resource out of all available options).

Likelihood Score=3*:* It is challenging to configure sidecars correctly to minimize configuration and reduce overhead. Some specific implementations do better jobs than others due to engineer tradeoffs (e.g. lazily loading configuration the first time an app needs it, vs eagerly pushing all configuration ahead of use) but overall it's easiest to land in a situation with the most resource utilization with a sidecar architecture.

5.1.9. **Privileged L4 Proxy (TR-9)**

Threat Description*:* Service mesh implementation requires L4 component (e.g., deployed as a DaemonSet on a Kubernetes cluster) to run with an overprivileged security context (e.g., Privileged Pod)

Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)*:* Same regardless of architecture -- in the per-node case this is usually encapsulated as a container network interface (CNI) provider which runs in a privileged context by default. In the sidecar case, privilege is only needed at startup to establish traffic interception rules; depending on the implementation (e.g., Kubernetes init containers) this can ensure that the privileged user is not run alongside the application but only during initialization. In all cases, typically CAP_NET_ADMIN is the only privilege required for mesh data plane functionality.

5.1.10. **Data Plane (Service Mesh) Bypassed (TR-10)**

Threat Description*:* Traffic is sent directly to a workload, bypassing mesh functionality and authorization policies.

Proxy Function Impacted*:* Easiest to bypass of all the available models, from app choosing not to use sidecar to container-local bypasses/configurations.

Impact Score=2: An app is exposed without mesh security controls.

Likelihood Score=2*:* Because the proxy runs in user space in the same cgroups as the application, there are a variety of attacks available that are not relevant/applicable to other implementations.

Cumulative Threat Score: (computed based on the methodology of Section 4.1) = 23

5.2. Threat Analysis for Shared L4 – L7 per Service Model (DPA-2)

5.2.1. Compromised L4 Proxy (TR-1)

<u>Threat Description</u>: Compromised L4 proxy (or L4 functions in the case of sidecar proxy with combined L4 and L7 functions) leads to leaked identities for every workload (service) running on the node.

<u>Proxy Function Impacted</u>: The L4 proxy has access to all the keys associated with the workloads running on the node.

<u>Impact Score=3</u>: Identities of all workloads (services) on the node are compromised

<u>Likelihood Score=1</u>: only code delivering L4 functions is present. This minimal code footprint and functionality presents the lowest attack surface of all options.

5.2.2. Compromised Application Container (TR-2)

<u>Threat Description</u>: Compromised application container (e.g., via a supply chain attack – during development phase) leads to takeover of identity associated with that application.

<u>Proxy Function Impacted</u>: Data plane components are not located in the same pod as workload containers, so a compromised workload does not necessarily lead to the access of keys / secrets.

<u>Impact Score=1</u>: Single workload / single identity. No direct access to underlying key material.

<u>Likelihood Score=2</u>: Same regardless of architecture

5.2.3. Compromise of Business Data (TR-3)

<u>Threat Description</u>: Threat Description: identity is used to pivot through the infrastructure, in order to compromise the confidentiality, integrity or availability of business data.

<u>Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)</u>: Same regardless of architecture -- this is the fundamental risk of identity-based policy and is why we need to practice the principle of least privilege (PoLP). The telemetry provided by the service mesh (regardless of architecture) is invaluable for understanding communicating in your system and creating accurate access policies (thereby implementing PoLP)

5.2.4. Compromised L7 Proxy (TR-4)

<u>Threat Description</u>: Vulnerability in L7 processing stack of the service mesh proxy. As L7 processing is inherently more complex, there is a higher probability for vulnerabilities to arise in this part of the stack, as supported by historical CVE data.

Proxy Function Impacted: This topology allows 'less complex' L4 capabilities, e.g. mTLS, to be adopted, with L7 processing only occurring if there is a strict requirement for it. Each service account has its own dedicated L7 proxy.

Impact Score=2: A single set of workloads is impacted (DoS) / single identity leaked. In the event of a DoS, it's much easier to make all workloads unavailable compared to the sidecar model because the mesh's L7 processing is centralized into L7 "middle proxies". We need to DoS this smaller number of middle proxies, vs needing to DoS every instance of the app in the sidecar/library cases.

Likelihood Score=1: same as sidecar / same argument around potentially impacting all workloads using L7 capabilities – see Section 5.1.4.

5.2.5. Compromise of Shared L7 Proxy (TR-5)

Threat Description: Co-tenant exploits L7 traffic processing vulnerability in shared proxy, to affect the confidentiality, integrity or availability of traffic to/from another workload running on the same node.

Proxy Function Impacted: By limiting the per-node functionality to L4 processing, the attack surface is significantly reduced.

Impact Score=1: The application workload itself is unaffected, only the proxy – which is a separate deployment. As long as the L7 proxy is not shared with the compromised application, there is no impact.

Likelihood Score=1: As likely as the previous entry.

5.2.6. Outdated Client Libraries in Applications (TR-6)

Threat Description: Client libraries are not updated frequently or consistently across the estate of microservices, leading to potential vulnerabilities and weaknesses that can be exploited.

Proxy Function Impacted: Infrastructure code decoupled from application code.

Impact Score=1: Same as the sidecar model, DPA-1 – see 5.1.6.

Likelihood Score=1: Same regardless of architecture.

5.2.7. Denial of Service (TR-7)

Threat Description: Conventional Denial of Service threat.

Proxy Function Impacted: A DoS executed at L4 has the same impact as the centralized per-node model because the L4 process is centralized per node: all apps on the node are impacted.

A DoS executed at L7 impacts all app instances of the target app, since a (set of) dedicated L7 proxy(-ies) is deployed per app. The number of proxies implementing L7 functionality is typically (far) less than the number of application instances making them an easier target for DoS than "every instance of the target app".

Impact Score=2: Every instance of the target app. An L4 DoS would impact all application instances on the target host.

Likelihood Score=2*:* The L4 proxy is deployed once per node, so it presents a better target for DoS than DPA-1 or DPA-4; this is mitigated somewhat by the simplified functionality of an L4 proxy compared to a combined L4+L7 proxy.

The L7 proxy is shared by multiple instances of the same application, it presents an easier DoS target than the application itself. Therefore it is more likely than the sidecar model, DPA-1.

5.2.8. **Resource Consumption (TR-8)**

Threat Description*:* Overall resource consumption by the data plane of the service mesh infrastructure.

Proxy Function Impacted*:* The shared L4 proxy typically has a much lower memory (RAM) footprint, as well as lower CPU usage overall due to a lower rate of change of config, less config overall, and less responsibility than a combined L4+L7 sidecar proxy, DPA-1. For the service mesh's data plane, L7 processing is typically the dominating CPU cost, followed by encryption.

L7 proxies are shared by all instances of the same application, deployed as a few traditional "reverse proxies" per app. This results in much lower resource consumption for L7 processing than the sidecar model (DPA-1). Overall DPA-2 uses more resources than the shared per node model (DPA-3), but substantially less than the sidecar (DPA-1). This is due primarily to reduced overhead -- e.g., an app with 50 instances requires 50 sidecars, but might be served with 5 shared L7 proxies (or less).

Impact Score=2: DPA-3 achieves a good middle ground: lower consumption than sidecar *and* easier to achieve than sidecar (DPA-1); but not as low as all shared (DPA-3) or all in app (DPA-4).

Likelihood Score=1*:* Easy to achieve low resource usage.

5.2.9. **Privileged L4 Proxy (TR-9)**

Threat Description*:* Service mesh implementation requires L4 component (e.g. deployed as a DaemonSet on a Kubernetes cluster) to run with an overprivileged security context (e.g. Privileged Pod).

Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)*:* Same regardless of architecture -- in the per-node case this is usually encapsulated as a container network interface (CNI) provider which runs in a privileged context by default. In the sidecar case, privilege is only needed at startup to establish traffic interception rules; depending on the implementation (e.g., Kubernetes init containers) this can ensure that the privileged user is not run alongside the application but only during initialization. In all cases, typically CAP_NET_ADMIN is the only privilege required for mesh data plane functionality.

5.2.10. **Data Plane (Service Mesh) Bypassed (TR-10)**

Threat Description: Traffic is sent directly to a workload, bypassing mesh functionality and authorization policies.

Proxy Function Impacted: Part of the goal of moving enforcement out of the app context and into a shared context is to use stronger primitives to ensure the non-bypass-ability of the mesh data plane. In general, with a per-node L4 setup, sending traffic to an individual app instance on the node should not be achievable (e.g. similar to [but not necessarily implemented as] a host-level VPN requiring workloads to be part of the VPN overlay to connect).

L7 proxies are deployed independently from the applications they represent, which requires special configuration in the mesh to ensure they're routed through, making bypassability easier than other models. Impact of missing L7 policy can be significant. (In other models we rely on 0 or 1 things to ensure traffic is directed to the correct policy enforcement point; in this model we rely on 2 things [traffic interception, mesh configuration to route via middle proxies] to ensure traffic is subject to the correct PEPs)

Impact Score=2: An app is exposed without mesh security controls.

Likelihood Score=2: L4 controls are by-design built to mitigate this; L7 controls are easier to bypass compared to sidecar model.

Cumulative Threat Score: (computed based on the methodology of Section 4.1) = 22

5.3. Threat Analysis for Shared L4 and L7 Model (DPA-3)

5.3.1. **Compromised L4 Proxy (TR-1)**

Threat Description: Compromised L4 proxy (or L4 functions in the case of sidecar proxy with combined L4 and L7 functions) leads to leaked identities for every workload (service) running on the node.

Proxy Function Impacted: The L4 proxy has access to all the keys associated with the workloads running on the node.

Impact Score=3: All identities on node

Likelihood Score=3: L7 code may be enabled for another server (not yours) which can be exploited to affect all apps on the host

5.3.2. **Compromised Application Container (TR-2)**

Threat Description: Compromised application container (e.g., via a supply chain attack – during development phase) leads to takeover of identity associated with that application.

Proxy Function Impacted: Data plane components are not located in the same pod as workload containers, so a compromised workload does not necessarily lead to the access of keys / secrets.

Impact Score=1: Single workload / single identity. No direct access to underlying key material.

Likelihood Score=2*:* Same regardless of architecture.

5.3.3. **Compromise of Business Data (TR-3)**

Threat Description*:* Threat Description: identity is used to pivot through the infrastructure, in order to compromise the confidentiality, integrity or availability of business data.

Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)*:* Same regardless of architecture -- this is the fundamental risk of identity-based policy and is why we need to practice the principle of least privilege (PoLP). The telemetry provided by the service mesh (regardless of architecture) is invaluable for understanding communicating in your system and creating accurate access policies (thereby implementing PoLP)

5.3.4. **Compromised L7 Proxy (TR-4)**

Threat Description*:* Vulnerability in L7 processing stack of the service mesh proxy. As L7 processing is inherently more complex, there is a higher probability for vulnerabilities to arise in this part of the stack, as supported by historical CVE data.

Proxy Function Impacted*:* This topology allows 'less complex' L4 capabilities, e.g., mTLS, to be adopted, with L7 processing only occurring if there is a strict requirement for it. Blast radius of a proxy compromise affects all workloads on the node. That means that its failure represents a shared fate outage, and as a shared resource it’s susceptible to denial of service attacks.

Impact Score=3: L7 capability is shared across all applications on the node, so if even a single application's configuration causes the proxy to become susceptible to failure then all applications on the node can be attacked (either a credential leak or denial of service, depending on the attack).

Likelihood Score=2*:* For a given app using L7 capabilities, as likely as the sidecar model. However, because workloads that are only doing L4 are susceptible to attack if they share the same node (which under the sidecar model, DPA-1, would have been safe), likelihood is higher.

5.3.5. **Compromise of Shared L7 Proxy (TR-5)**

Threat Description*:* Co-tenant exploits L7 traffic processing vulnerability in shared proxy, to affect the confidentiality, integrity, or availability of traffic to/from another workload running on the same node.

Proxy Function Impacted*:* A single proxy instance does not provide an inherently multi-tenant setup. Hence security concerns arise when combining complex processing rules for L7 traffic from multiple unconstrained tenants in a shared instance. In this configuration, L7 processing of multiple co-tenants' traffic is performed within one process, with no memory protection or isolation benefits that could be gained by containerizing L7 functionality per workload

Impact Score=3: All workloads on the node are impacted.

<u>Likelihood Score=2</u>: See section 5.3.4 above – a compromise is as likely as the sidecar model (DPA-1), but applications that would not be susceptible to attack under DPA-1 *are* susceptible under this model, DPA-3.

5.3.6. Outdated Client Libraries in Applications (TR-6)

<u>Threat Description</u>: Client libraries are not updated frequently or consistently across the estate of microservices, leading to potential vulnerabilities and weaknesses that can be exploited.

<u>Proxy Function Impacted</u>: Infrastructure code decoupled from application code.

<u>Impact Score=1</u>: Same as the sidecar model, DPA-1 – see 5.1.6.

<u>Likelihood Score=1</u>: Same regardless of architecture.

5.3.7. Denial of Service (TR-7)

<u>Threat Description</u>: Conventional Denial of Service threat.

<u>Proxy Function Impacted</u>: Because processing for all app instances on the node is shared, and a single proxy instance is not inherently multi-tenant (provides no controls wrt resource utilization across independent backends and clients), the blast radius of DoS on the mesh data plane is every app on the node.

<u>Impact Score=3</u>: All workloads on the node.

<u>Likelihood Score=2</u>: If *any* app configuration triggers exploitable paths in the shared proxy, *all* apps on the node suffer.

5.3.8. Resource Consumption (TR-8)

<u>Threat Description</u>: Overall resource consumption by the data plane of the service mesh infrastructure.

<u>Proxy Function Impacted</u>: Because *all* functionality is shared at the node level, DPA-3 has the most opportunity for deduplication -- therefore reduction in resource usage. Configuration like service discovery need only be sent a single time to each node, rather than to each and every app instance. Overall this means the lowest rate of change and least data transferred, as well as a lower runtime footprint (RAM, CPU).

Note some implementations don't fully de-dupe configuration (for a variety of reasons, both due to implementation and as a security measure to provide some degree of isolation), so consume RAM more similarly to a sidecar case than might otherwise appear.

<u>Impact Score=1</u>: lowest overall resource utilization of all available architectures

<u>Likelihood Score=1</u>: easiest to achieve low resource utilization

5.3.9. **Privileged L4 Proxy (TR-9)**

***Threat Description**:* Service mesh implementation requires L4 component (e.g. deployed as a Daemon Set on a Kubernetes cluster) to run with an overprivileged security context (e.g. Privileged Pod).

***Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)**:* Same regardless of architecture -- in the per-node case this is usually encapsulated as a container network interface (CNI) provider which runs in a privileged context by default. In the sidecar case, privilege is only needed at startup to establish traffic interception rules; depending on the implementation (e.g., Kubernetes init containers) this can ensure that the privileged user is not run alongside the application but only during initialization. In all cases, typically CAP_NET_ADMIN is the only privilege required for mesh data plane functionality.

5.3.10. **Data Plane (Service Mesh) Bypassed (TR-10)**

***Threat Description**:* Traffic is sent directly to a workload, bypassing mesh functionality and authorization policies.

***Proxy Function Impacted**:* Part of the goal of moving enforcement out of the app context and into a shared context is to use stronger primitives to ensure the non-bypass-ability of the mesh data plane. In general, with a per-node setup, sending traffic to an individual app instance on the node should not be achievable (e.g., similar to [but not necessarily implemented as] a host-level VPN requiring workloads to be part of the VPN overlay to connect).

Impact Score=3: All applications on the node are exposed without mesh security controls.

***Likelihood Score=1**:* By design built to mitigate this kind of bypass

Cumulative Threat Score: (computed based on the methodology of Section 4.1) = 37

5.4. **Threat Analysis for L4 and L7 within Application Model (gRPC proxyless Model (DPA-4))**

5.4.1. **Compromised L4 Proxy (TR-1)**

Threat Description: Compromised L4 proxy (or L4 functions in the case of sidecar proxy with combined L4 and L7 functions) leads to leaked identities for every workload (service) running on the node.

Proxy Function Impacted: mTLS connections are negotiated by the client library inside of the application, with a single identity (the application's). In order to compromise key material and identity documents for multiple workloads, multiple application instances would need to be compromised.

Impact Score=1: Single workload / single identity.

<u>Likelihood Score=2</u>: Large surface area if something goes wrong, since we're inside the application's context. Therefore, this is as likely or slightly more likely than DPA-1.

5.4.2. **Compromised Application Container (TR-2)**

<u>Threat Description</u>*:* Compromised application container (e.g., via a supply chain attack – during development phase) leads to takeover of identity associated with that application.

<u>Proxy Function Impacted</u>*:* Compromising the application *is* compromising the mesh in this case; full access to any key material used by the application -- including the mesh identity -- is achievable.

<u>Impact Score=2</u>: Single workload / single identity. Full access to key material used by that application.

<u>Likelihood Score=2</u>*:* Same regardless of architecture.

5.4.3. **Compromise of Business Data (TR-3)**

<u>Threat Description</u>*:* Threat Description: identity is used to pivot through the infrastructure, in order to compromise the confidentiality, integrity or availability of business data.

<u>Proxy Function Impacted & Impact Score (=1) & Likelihood Score (=1)</u>*:* Same regardless of architecture -- this is the fundamental risk of identity-based policy and is why we need to practice the principle of least privilege (PoLP). The telemetry provided by the service mesh (regardless of architecture) is invaluable for understanding communicating in your system and creating accurate access policies (thereby implementing PoLP)

5.4.4. **Compromised L7 Proxy (TR-4)**

<u>Threat Description</u>*:* Vulnerability in L7 processing stack of the service mesh proxy. As L7 processing is inherently more complex, there is a higher probability for vulnerabilities to arise in this part of the stack, as supported by historical CVE data.

<u>Proxy Function Impacted</u>*:* Compromising the L7 processing stack results in compromising the entire application, resulting in more risk of compromise beyond runtime identity and DoS for other users.

<u>Impact Score=3</u>: The application itself is compromised, including non-mesh credentials (e.g. `truncate table users;`) that are not available if only the proxy is compromised.

<u>Likelihood Score=3</u>*:* L7 processing code *is* the application, and as a result the surface area is much larger.

5.4.5. **Compromise of Shared L7 Proxy (TR-5)**

***Threat Description**:* Co-tenant exploits L7 traffic processing vulnerability in shared proxy, to affect the confidentiality, integrity or availability of traffic to/from another workload running on the same node.

***Proxy Function Impacted**:* L7 processing is entirely isolated by whatever mechanisms isolate applications themselves (containers, micro-VMs, VMs, etc). Impact is limited by the strength of that boundary.

Impact Score=1: See section 5.1.5.

***Likelihood Score=1**:* As likely as any other application compromise.

5.4.6. **Outdated Client Libraries in Applications (TR-6)**

***Threat Description**:* Client libraries are not updated frequently or consistently across the estate of microservices, leading to potential vulnerabilities and weaknesses that can be exploited.

***Proxy Function Impacted**:* Infrastructure concerns are embedded within application code. Challenges can arise when enforcing consistency in versions between microservices etc.

Impact Score=3: The mesh functionality itself is part of the application, therefore bad application updates mean bad mesh updates. This means vulnerabilities stick around for longer. By the same token, since the mesh is part of the app, a vulnerability in the app *is* a vulnerability in the mesh data plane.

***Likelihood Score=2**:* Depends on frequency of update -- if applications can be updated quickly (i.e., on the order of minutes to hours), likelihood is low. If applications take on the order of weeks to months to update, likelihood is high. In the realm of days-to-update we have a middle ground of risk that's likely acceptable to most organizations. However cross-cutting concerns like mesh data plane, which are critical to the org's overall security posture, should be patched as soon as possible.

5.4.7. **Denial of Service (TR-7)**

***Threat Description**:* Conventional Denial of Service threat.

***Proxy Function Impacted**:* A DoS of the mesh data plane (L4 *or* L7) is a DoS of the application itself. In all other respects, it's very similar to the sidecar.

Impact Score=1: Single instance of single app. See 5.1.7 – an attack could be repeated across all applications.

***Likelihood Score=2**:* Not just mesh data plane functionality is susceptible to DoS, but application code/functionality itself.

5.4.8. Resource Consumption (TR-8)

***Threat Description**:* Overall resource consumption by the data plane of the service mesh infrastructure.

***Proxy Function Impacted**:* Because it's built into the app, resources devoted to mesh data plane functionality are *very* low. The only reason resource utilization overall winds up being higher than the shared L4/L7 model (DPA-3) is because some duplication of configuration and processing needs to happen since configuration needs to be pushed to every application instance.

Impact Score=2: Potentially lower resource usage on a per-app basis than any other model, but likely higher in aggregate because we can't share any resources or configuration across data plane instances.

***Likelihood Score=1**:* Easy to achieve low resource usage.

5.4.9 Privileged L4 Proxy (TR-9)

***Threat Description**:* Service mesh implementation requires L4 component (e.g., deployed as a Daemon Set on a Kubernetes cluster) to run with an overprivileged security context (e.g., Privileged Pod).

***Proxy Function Impacted & Impact Score (=0) & Likelihood Score (=0)**:* Mesh data plane functionality runs in the application context without any special privileges -- it's the same as the app itself. No special capabilities or permissions are required to intercept traffic or implement policy enforcement.

5.4.10 Data Plane (Service Mesh) Bypassed (TR-10)

***Threat Description**:* Traffic is sent directly to a workload, bypassing mesh functionality and authorization policies.

***Proxy Function Impacted**:* App *is* the enforcement point, there is no bypassing.

Impact Score=1: The app is exposed in a degraded state or without some controls.

***Likelihood Score=1**:* By nature of RPC frameworks and in-process enforcement, mesh data plane policy should not be bypassable.

Cumulative Threat Score: (computed based on the methodology of Section 4.1) = 28

6. Recommendations Based on Application Security Risk Profile

While the ratings or scores for the impact and likelihood parameters for different threats in different data plane architectures are dictated by the number of service instances affected, the risk profiles associated with applications are determined by the criticality of the entire application with respect to the business process it supports.

In arriving at the threat profile for each of the architectural patterns considered in section 5, please recall that we observed that for some threats, the impact and likelihood parameters are the same irrespective of the proxy model or data plane architecture. The ratings assigned to these parameters are as we already stated, are relative ratings, and hence both parameters are assigned the rating 1, resulting in the overall threat rating of 1 for those threats. The threats that come under this category are:

- Compromise of Business Data (TR-3)
- Privileged L4 Proxy (TR-9)

Hence, we have to ignore the threat ratings of the above listed threats and dwell into the consideration of threat ratings for the other remaining threats. While considering the remaining threats, we must ignore those threats that have no direct security implications but may have performance implications. The only threat that comes under this category is:

- Resource Consumption (TR-8)

6.1. Cloud-Native Applications with Low Risk Profile

The service mesh capability requirements for this class of application are as follows:

- LOW-REQ1: Service-to-Service authorization (Service A can call Service B) is based on network location/parameter (e.g., subnet) and authorization at the granularity of the called service method, calling user and per call request are not required.
- LOW-REQ2: Logging and Metrics need to be captured only at the level of network parameters (e.g., Source/Destination TCP address) and not at the level of per call request.
- LOW-REQ3: All traffic management capabilities such as load balancing, rate limiting etc. need to be enforced at the network connection level and not at the per call request.

Examination of the above capabilities reveals that these essentially involve network transport/network level data, that can be all provided by proxy's L4 functions and hence by L4 proxies. Hence the following are recommendations for this class of application.

Recommendations:

1. Since all requirements can be met by L4 proxies or L4 functions built into the libraries, all four data plane architectures can be theoretically used.
2. Since neither method-level nor per call request handling is required, thus eliminating all L7 functions, data plane architectures that deploy a L7 proxy per service instance (side-

car model (DPA-1) expose an unnecessary attack surface. Therefore, either of the two models with a shared L4 proxy (DPA-2 and DPA-3) is recommended. gRPC proxy-less model (DPA-4) is also usable for this class of applications, though it does expose a larger attack surface than DPA-2 or DPA-3.

6.2. Cloud-Native Applications with Medium Risk Profile

The service mesh capability requirements for this class of application are as follows:

- MEDIUM-REQ1: In addition to Service-to-Service authorization at the level of service, a full authorization policy at the method level (Service A can execute GET on B's Billing method with valid end user credentials containing the READ scope) is required.
- MEDIUM-REQ2: Logging and Metrics data need to be captured not only at the level of network parameters (e.g., Source/Destination TCP address) but also some metadata such as the called service and method.
- MEDIUM-REQ3: All traffic management capabilities such as load balancing, rate limiting etc. can be enforced at the network connection level (as in low risk profile case) and not at the per call request or per method level.

Examination of the above capabilities reveals that these essentially involve not only network transport/network level data (all L4 functions), but also some L7 functions (not all) such as authenticating user identities not only locally from tokens (e.g., Jason Web Tokens (JWT)) but also remotely using standardized protocols such as OAuth and OIDC. Hence use of L7 proxies with some limited functionality is mandatory. Hence the following are recommendations for this class of application.

Recommendations:

1. Just like for applications with low risk profile, all four data plane architectures can be theoretically used.
2. Since L7 functions are limited, it is not essential to dedicate a L7 proxy for each service. Hence, data plane architectures that deploy a L7 proxy for each service (side-car model (DPA-1)) may end up consuming more resources than other models for limited additional assurance. On the other hand, as previously discussed, L7 code is where most exploitable vulnerabilities lie. Hence shared L4-L7 model (DPA-3) is not desirable since the shared L7 component introduces risk for all services that share the same physical host. Therefore, the shared L4 -- L7 per service model (DPA-2) is likely the best mix of resource utilization and risk. gRPC proxy-less model (DPA-4) with inclusion of libraries for L4 functions and limited L7 functions is also recommended, with similar risk but even less resource utilization than DPA-2 in most cases.

6.3. Cloud-Native Applications with High Risk Profile

The service mesh capability requirements for this class of application are as follows:

- HIGH-REQ1: In addition to: (a) Service-to-Service authorization at the level of service and, (b) a full authorization policy at the method level (Service A can execute GET on B's Billing method with valid end user credentials containing the READ scope), a full user to resource level access control is required. The last requirement necessitates the proxy making an external authorization call for each request.
- HIGH-REQ2: Logging and Metrics meta data relating to a request must be captured – rate of requests, rate of positive outcomes, processing time for each request etc.
- HIGH-REQ3: All traffic management capabilities are required at the request level and should involve application layer parameters in addition to those at the network connection level.

Examination of the above capabilities reveals that a complete suite of L7 functions is required.

Recommendations:

1. Just like for applications with low risk and medium risk profiles, all four data plane architectures can be theoretically used.
2. However, based on the requirements, this class of applications belong to Highly critical applications, which require a great degree of isolation, where any compromise, if it occurs should be limited to only one service instances and not multiple service instances. Hence, data plane architectures that deploy a L7 proxy for each service (side-car model (DPA-1)) is most applicable. A shared L7 proxy per Service (like DPA-2) can be an acceptable tradeoff for some organizations, provided they have other mechanisms for mitigating shared-fate failures of all instances of the service that the shared service mesh L7 proxy brings (e.g., mitigating a denial-of-service attack via L3 controls outside the mesh). However, tightly integrating both L4 & L7 functions with the service instance provides a greater degree of isolation and hence the former data plane architecture (DPA-1) is highly recommended.

7. Summary and Conclusions

Microservices-based applications implemented using containers & VMs and sometimes spanning on-premises and multiple clouds go by the name of cloud-native applications. In instances where a centralized service infrastructure is beneficial to the overall security of this class of applications, this need is met by a service mesh.

Service mesh implementations are characterized by the type of configurations of entities called proxies which are the engines that enable various capabilities during application runtimes - such as policy enforcement (including access control), network connectivity (including establishment of secure sessions), performance monitoring (through collection of data for computing various metrics) etc. The proxies thus form the data plane of the service mesh, and a particular configuration of proxies is called a proxy model or a data plane architecture.

The first and still the widely prevalent deployment of the proxy model is the side car model where a single proxy that provides functions both at the L4 and L7 level is associated with a service instance. Performance, resource consumption and specific security needs for different cloud-native applications have led to exploration of alternate proxy models.

In this document, we performed a detailed threat analysis of these alternate proxy models (including the ones that provide the needed security functions without proxies) by identifying ten common threats and provided recommendations for their use in cloud-native applications with different security risk profiles.

References

[1] Wikipedia (2024) OSI Model. Available at https://en.wikipedia.org/wiki/OSI_model

[2] Chandramouli R, Butcher Z (2020) Building Secure Microservices-based Applications Using Service-Mesh Architecture. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-204A. https://doi.org/10.6028/NIST.SP.800-204A

[3] Chandramouli R, Butcher Z, Aradhna C (2021) Attribute-based Access Control for Microservices-based Applications using a Service Mesh. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-204B. https://doi.org/10.6028/NIST.SP.800-204B

[4] Chandramouli R (2022) Implementation of DevSecOps for a Microservices-based Application with Service Mesh. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) NIST SP 800-204C. https://doi.org/10.6028/NIST.SP.800-204C

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

[6] Jackson E, Kohavi Y, Pettit J, Posta C (2022) Ambient Mesh Security Deep Dive. (Istio) Available at https://istio.io/latest/blog/2022/ambient-security/

[7] Howard J, Jackson EJ, Kohavi Y, Levine I, Pettit J, Sun L (2022) Introducing Ambient Mesh. (Istio) Available at https://istio.io/latest/blog/2022/introducing-ambient-mesh/#what-about-security

[8] Turner M (2022) eBPF and Sidecars - Getting the Most Performance and Resiliency out of the Service Mesh. (Tetrate) Available at https://tetrate.io/blog/ebpf-and-sidecars-getting-the-most-performance-and-resiliency-out-of-the-service-mesh/

[9] Graf T (2021) How eBPF will solve Service Mesh - Goodbye Sidecars. (Isovalent) Available at https://isovalent.com/blog/post/2021-12-08-ebpf-servicemesh/

[10] Song J (2022) Transparent Traffic Intercepting and Routing in the L4 Network of Istio Ambient Mesh. (Tetrate) Available at https://tetrate.io/blog/transparent-traffic-intercepting-and-routing-in-the-l4-network-of-istio-ambient-mesh/

[11] Song J (2022) L7 Traffic Path in Ambient Mesh. (Tetrate) Available at https://tetrate.io/blog/l7-traffic-path-in-ambient-mesh/

[12] Cilium (2024) Threat Model — Cilium 1.15.6 documentation. (Cilium) Available at https://docs.cilium.io/en/stable/security/threat-model/

[13] Istio (2024) Ambient mode overview: ztunnel. Available at https://istio.io/latest/docs/ambient/overview/#ztunnel

[14] Landow S (2021) gRPC Proxyless Service Mesh. (Istio) Available at https://istio.io/v1.15/blog/2021/proxyless-grpc/