SOLO.IO

White paper

Achieve Compliance, Zero Trust with Istio Ambient Mesh

Understand how to use Zero Trust Architecture principles and Ambient Mesh to achieve compliance

Written by Christian Posta, VP, Global Field CTO, Solo.io

Contents

Introduction

With the erosion of perimeter-based security, usage of cloud-

based services, and the requirements for industry and federal-agency compliance, modernization efforts will necessitate the implementation of Zero Trust Architectures (ZTA). In recent years, a service mesh has been a powerful tool to implement the tenants of ZTA as we'll explore in more detail.

Istio is the most popular, mature, and widely deployed service mesh in the cloud-native ecosystem and is still pushing the boundaries of adoption and innovation. Istio, recently announced the general availability of 'ambient mode'.

This announcement is an advancement that delivers a production-ready Ambient mesh, a new approach to implementing service mesh using a sidecar-less data plane that focuses on ease of operations, enabling incremental adoption and separation of security boundaries for applications and mesh infrastructure. Ambient mesh maintains the properties of Zero Trust Networking and provides a flexible approach to help achieve Zero Trust Architecture (ZTA) adoption. Solo.io co-founded Istio and Ambient mesh and continues to be a lead contributor to the implementation in the open-source community.

In this white paper, we'll explore the forces of modernization and compliance pressures, how Zero Trust Architecture (ZTA) can help, and specifically how Istio ambient mesh lowers the barrier for establishing the properties necessary to achieve Zero Trust and compliance.

Solo.io co-founded Istio ambient mode and continues to be a lead contributor to this implementation in the open-source community.

Challenges with Industry Compliance

Finance, health, federal agencies and other organizations are responsible for treating their data safely, and various regulatory bodies have set baseline standards to which these organizations must adhere. Compliance requirements, such as Payment Card Industry Data Security Standard (PCI-DSS), among many others, call for protecting sensitive data, using secure networks, and implementing strong controls for how data is accessed. This becomes increasingly difficult to do across the modern enterprise.

Services and applications are deployed in various cloud islands where provider-specific controls like networking, computing, policy are used, but can cause friction with existing systems and controls.

Workloads are highly dynamic, constantly being scheduled, failing, restarting, and scaling automatically. Users need to understand the components that constitute a service and whether they include critical elements like sets of IP addresses and firewall configurations in a dynamic, heterogeneous, cross-cloud environment. How does traffic navigate these complex systems? How is security ensured? What Public Key Infrastructure (PKI) is in place? How is identity established? And how should the proliferation of diverse protocols—such as SOAP, REST, gRPC, GraphQL, Kafka, and MQ/JMS—be managed effectively for applications?

- Maintain a secure network
- Restrict sensitive data (card-holder, patient health, identifiable info, etc)
- Track vulnerabilities, patch/upgrade any known vulnerabilities
- Implement strong access control to sensitive data
- Monitor, track, and dynamically alter policy

Traditional modes of security, based on a secure perimeter, static deployments, and assumptions about ownership of assets/networks/ trust are no longer as ironclad as once-believed. Breaching a network firewall has been shown to have disastrous effects with unfettered lateral movement, exfiltration of plaintext data, and more.

Can Zero Trust Networking Help Here?

Zero Trust principles aim to eliminate trust inherently[2] in the system to limit lateral movement and data exposure in the event of a security compromise. Recently, the President of the United States signed an Executive Order[3] to begin implementing Zero Trust Architecture principles for federal agencies as outlined in an OMB memorandum[4] and published by NIST in 800-207[5]. Google began implementing Zero Trust networking[6] as a response to the highly distributed, dynamic, and vulnerable perimeter networking they established in the past.

As described in detail in NIST 800-207, the following are the tenants of Zero Trust Architecture:

- Everything (regardless of where it's deployed) is considered a "resource"
- All communication to resources is secured, regardless of location on the network
- All-access is authenticated and authorized
- Access is tracked, logged, audited, and can be dynamically revoked
- Access to resources is determined dynamically
- Access to resources is granted per session

ZTN is foundational to help **improve security posture and implement industry compliance**, especially when modernizing application infrastructure and architecture.

Logical ZTN Architecture/Implementation

There are a few approaches to implementing ZTN as outlined in NIST 800-207. One approach relies on client- side agents that direct a client where to send requests for a particular resource and server-side gateways to gate the traffic. Another approach takes into account enclaves of services behind a next-generation gateway. Please see NIST 800-207 for more. A lot of the various models rely on a common set of components, however.

ZTA as described in NIST 800-207 uses the concepts of a decision engine (policy decision point or PDP) and a policy enforcement point (PEP) to dynamically determine whether or not access to a resource should be permitted as shown in Figure 1. ZTA also strives to reduce any “implicit trust zones” to be as small as possible. An implicit trust zone is a communication path to a resource after access has been granted. For a more detailed treatment on ZTA, please see NIST 800-207.

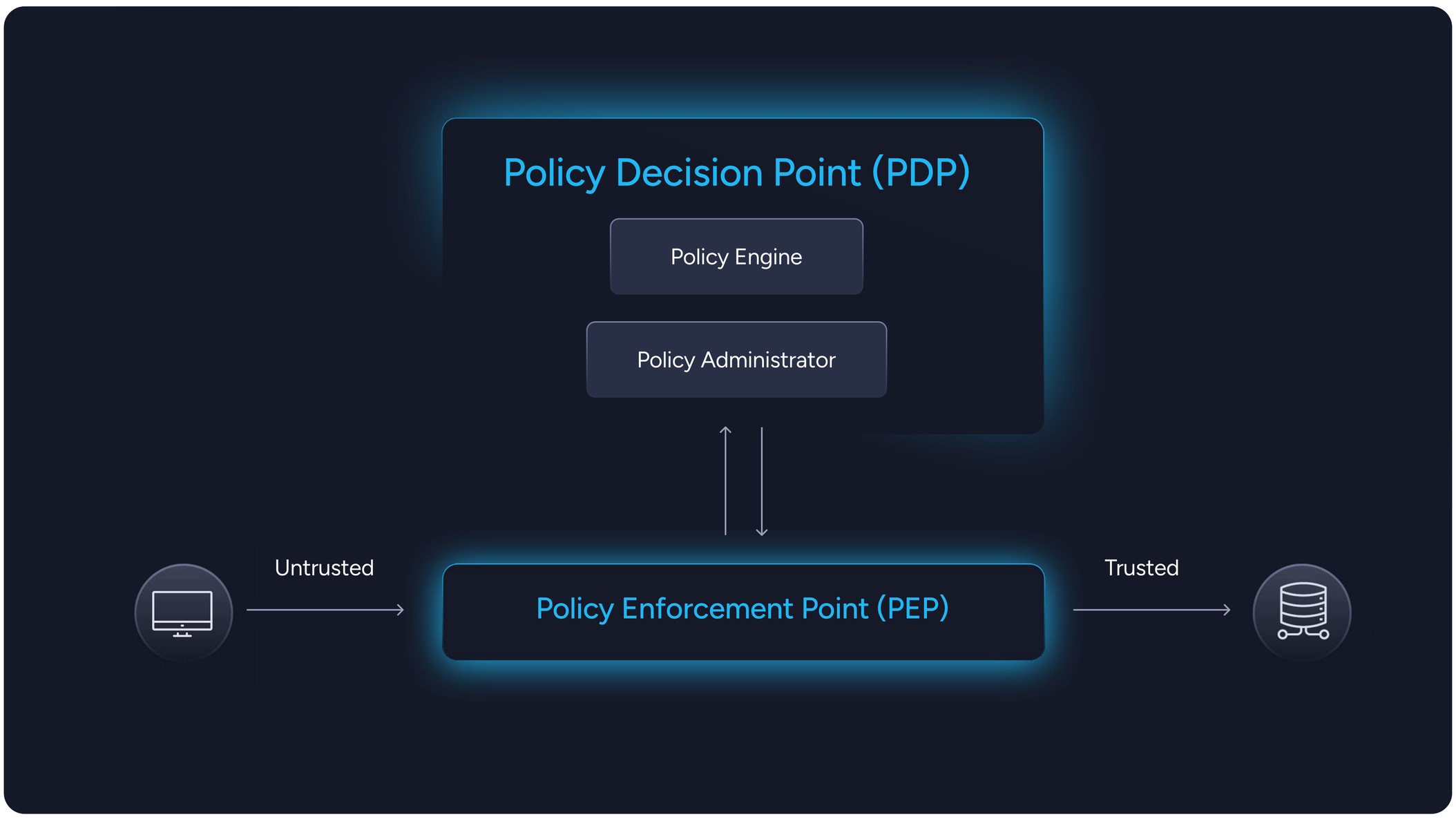

Figure 1: Logical components of ZTA include a policy enforcement point (PEP) and policy decision point (PDP)

Google's Beyond Corp uses the approach of an "access proxy" as its policy enforcement point (PEP) with an access-control engine that can dynamically determine access to a resource (policy decision point, PDP) based on the client device, location, time of the day, and other attributes that go into a "trust score". See Figure 2.

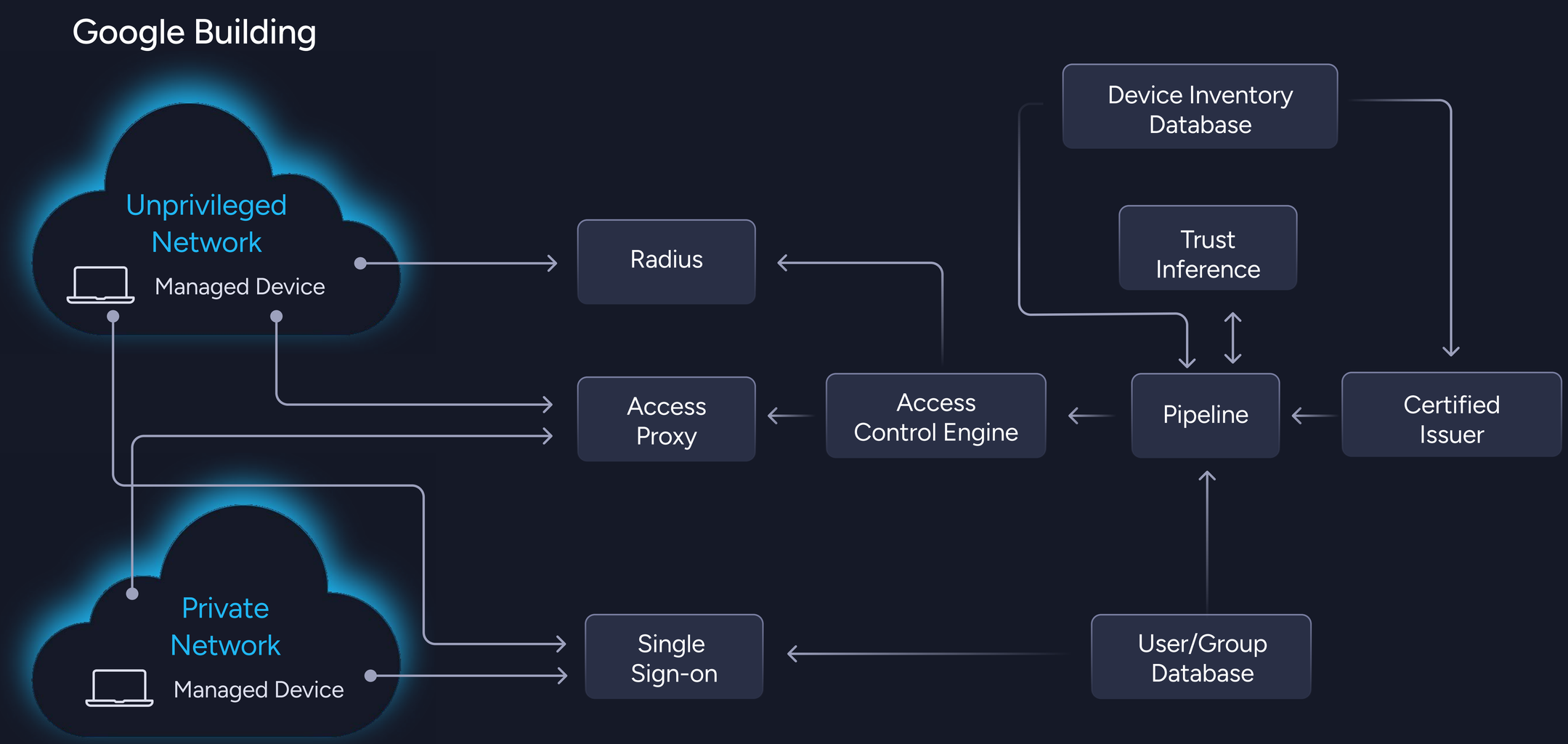

Figure 2: Google's BeyondCorp paper describes their Zero Trust Architecture using an access proxy (PEP) and access control engine (PDP)

In any of these models, the components used to implement ZTA are the same:

Something (PEP) is used to gate access to a resource and enforce any policies associated with communication to the target resource	A component used to dynamically decide whether that communication is allowed based on attributes of the caller, request, and circumstances of the access

Through this decision and enforcement mechanism, we can secure access to just about any resource and eliminate any implicit trust in the network architecture. This security posture would also provide the foundation for requirements set forth for industry compliance.

Ambient mesh deployment architecture actually looks a lot closer to the logical diagrams of ZTA composed of Policy Decision Points (PDP) and Policy Enforcement Points (PEP) deployed as resource gateways.

Implementing ZTN To Help Achieve Compliance With Ambient Mesh

A service mesh can help provide the foundations of Zero Trust networking by transparently enabling and enforcing authentication and authorization through mTLS and strong identity. A service mesh can also use the properties of an application's request to enforce access control based on the end-user that originates the request.

Typically a service mesh requires teams to deploy the policy enforcement point (PEP) as a sidecar proxy that runs co-located with an application instance.

This can lead to an unnecessary infrastructure-to- application coupling which introduces friction for platform operations. The sidecar approach also leads to unnecessary resource allocation to run a sidecar per workload instance.

Ambient Mesh implements an alternative data plane that does not require injecting/co-deploying a sidecar or PEP with each instance of the application without trading off the properties of Zero Trust networking. In fact, ambient mesh deployment architecture actually looks a lot closer to the logical diagrams of ZTA composed of Policy Decision Points (PDP) and Policy Enforcement Points (PEP) deployed as resource gateways. Ambient mesh separates the mesh functionality into two separate, but composable, layers that handle different concerns and can be adopted incrementally. The first layer is responsible for establishing the foundation for security in the mesh and handles only L4 traffic. This is called the "secure overlay layer". Another layer, that is built to handle more complex L7 capabilities, is called the waypoint proxy layer and handles more complex L7 policy enforcement (ie, becomes the PEP).

The benefits of running as two different layers and outside of the applications (ie, without a sidecar) are primarily around operations, however, performance, security, and resource usage/cost can also be improved. Let's dig into these two layers.

Secure Transport Layer

The secure transport layer of ambient mesh is responsible for establishing secure connections using a strong identity between mesh workloads. This allows application traffic to be encrypted transparently, authenticated using mTLS, and apply L4 network authorization policy. It does this by leveraging a CNI plugin and a component called the ztunnel. The ztunnel specifically handles collecting L4 telemetry, opening connections, and establishing mTLS with workload identity cryptography.

The ztunnel gets deployed as a DaemonSet on Kubernetes and can be implemented with dedicated L4 technology. See Figure 3. For example, in Istio open-source, we are building a version of the ztunnel in Rust while at Solo.io we are building this component with eBPF. When deployed per node, the ztunnel becomes shared among all of the workloads that run on the respective node. In many ways, the ztunnel component becomes an extension of the underlying network or CNI.

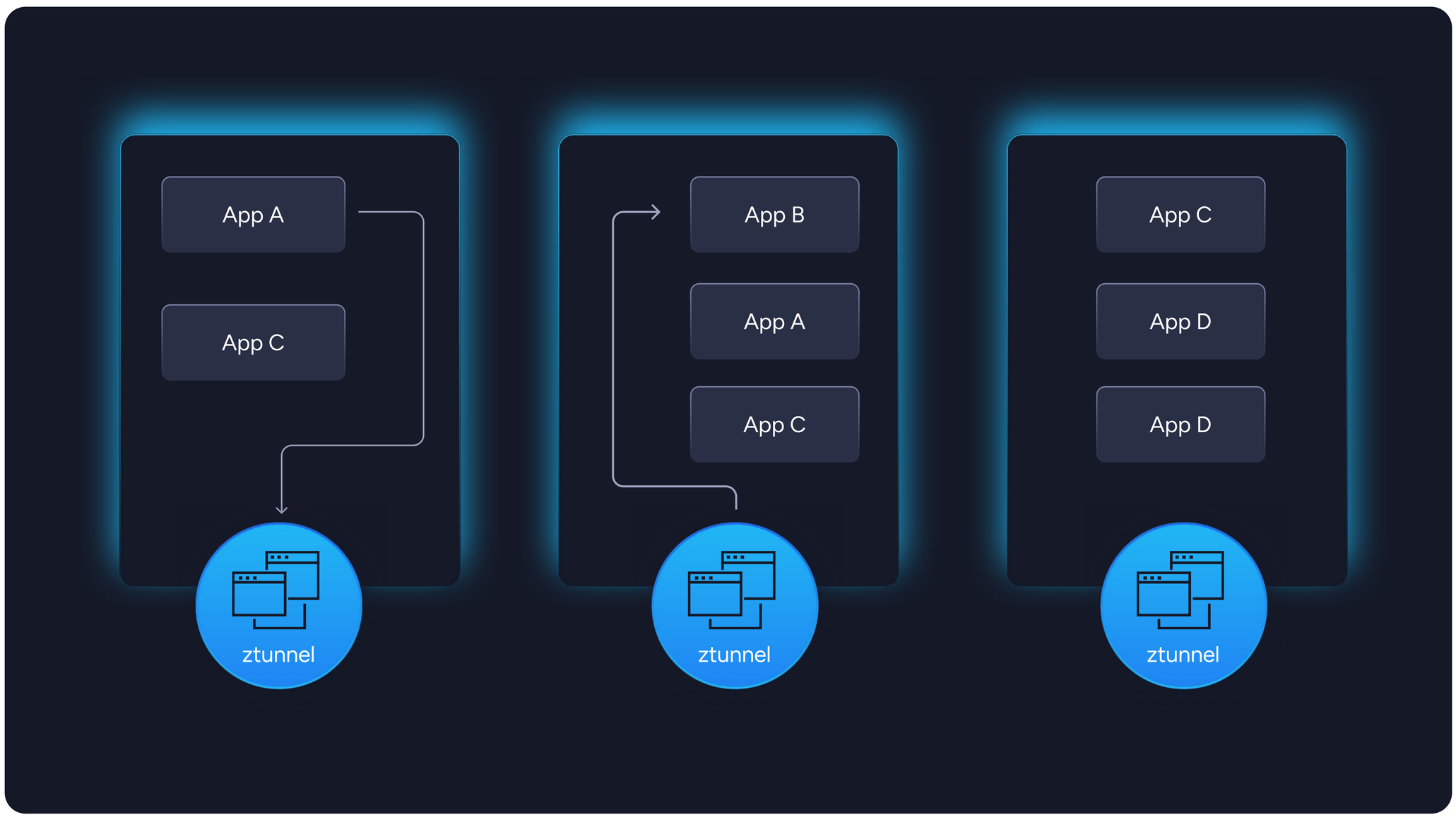

Figure 3: The secure overlay layer uses the ztunnel component to provide the foundations of Zero Trust networking

We can improve the security and patching lifecycle of the platform with predictable, low-risk upgrades.

By not running the data plane co-located with the application asone would in the sidecar model, we reduce the attack surface of the mesh data plane. In a scenario where the application is compromised, an attacker will not have access to the tokens, keys, and certificate material that represents a strong identity in the service mesh. This separation also plays a role in reducing friction for upgrades and data plane patching.

By simplifying the process to perform CVE patches for the data plane, and decoupling the infrastructure from the application workloads, we can improve the security and patching lifecycle of the platform with predictable, low-risk upgrades.

In this model, the secure overlay layer plays the role of securing, encrypting, and establishing authentication and simple authorization policies for the traffic in the mesh. To enforce more sophisticated network policies on the application traffic, we need to get access to the request and do this with the L7 waypoint proxy layer.

Waypoint Proxy Layer

Layer 7 service mesh capabilities are implemented with waypoint proxies in the ambient mesh architecture as shown in Figure 4. This data plane fully parses the connection into requests and can apply policies based on properties like headers and credentials found in the request. Layer 7 functionality includes things like:

- HTTP 1.x, 2, or 3
- Advanced load balancing
- Request routing or mirroring
- Fault injection
- Request retries
- JWT token validation
- HTTP aware circuit breaking

Waypoint proxies are deployed per workload identity (or per service account in Kubernetes) and can be scaled independently depending on the request load to an individual workload as shown in Figure 4. You can think of these waypoint proxies as individual gateways or policy enforcement points (PEPs) per workload type, as shown in Figure 5.

We made a deliberate choice with the ambient mesh architecture to not co-locate multiple workload identities on the same PEP. Although ambient can be configured to share multiple workloads on a single waypoint proxy, by default, the waypoint proxy PEPs are deployed per service account.

A big reason why we don't want to co-locate multiple workload identities in a single waypoint proxy is the security blast radius. If we evaluate the number of CVEs in a complex but powerful proxy like Envoy proxy, we find that most of the CVEs are in L7 capabilities of the proxy. By avoiding sharing identities on a shared L7 waypoint proxy, we can avoid a single identity or workload compromise extending to others.single identity or workload compromise extending to others.

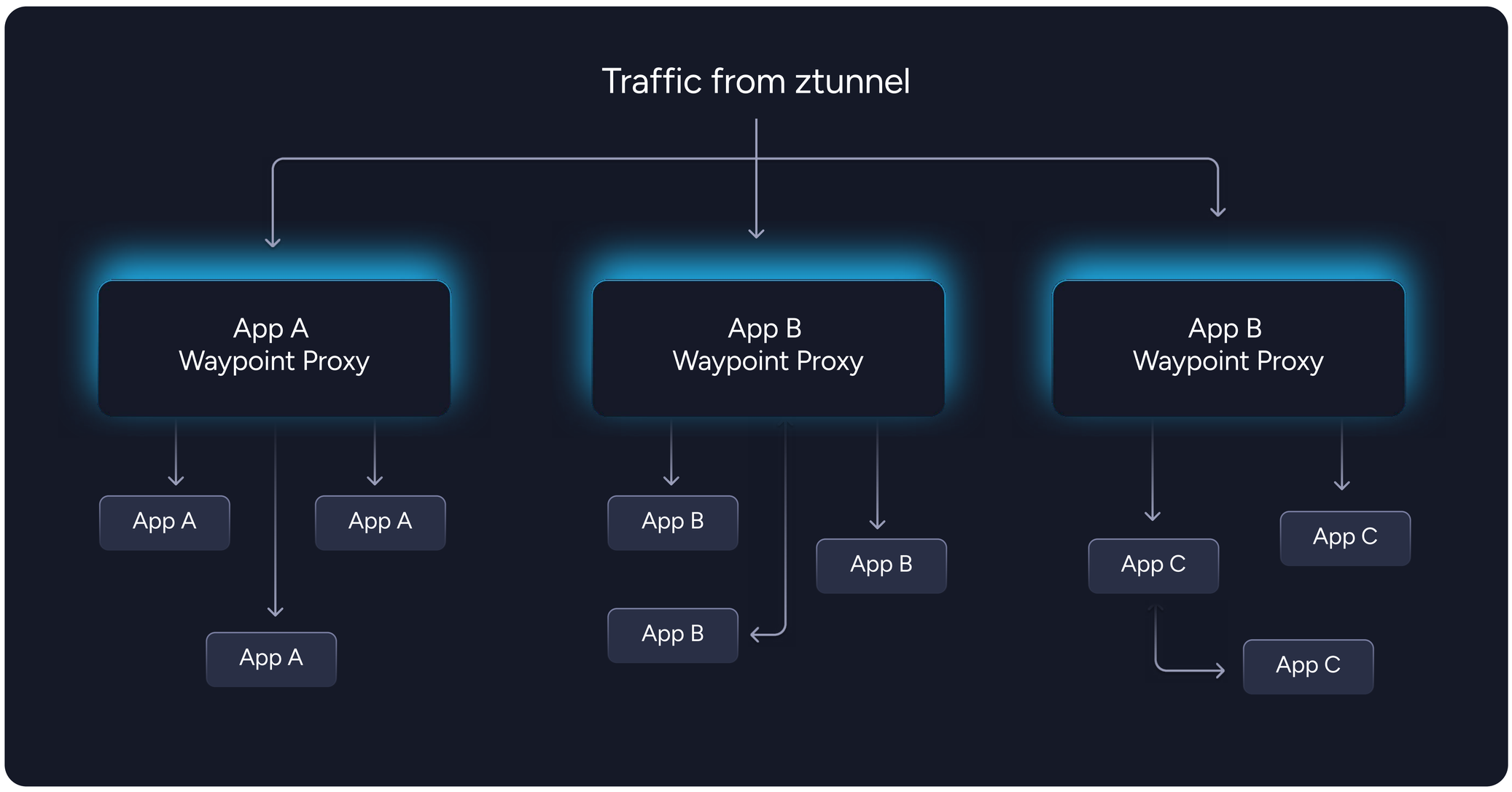

Figure 4: The waypoint proxy is deployed per service account/workload identity and can be thought of as a "gateway per workload" or Policy Enforcement Point (PEP)

The Policy Decision Point in the ambient model will be similar to what we see in the sidecar approach; each PEP can be configured to apply attribute-based access policy (ABAC) through the Istio control plane or can be configured to communicate with an external policy engine (like an ExtAuthz server, Open Policy Engine/OPA, or something similar).

Waypoint proxies get deployed by namespace owners, platform owners, or automation. When a waypoint proxy is deployed, and a corresponding L7 policy is configured for a destination represented by the waypoint proxy, the secure transport layer will route the connection to the correct L7 waypoint proxy as shown in Figure 5.

With Solo.io's Gloo Mesh, we automate away a lot of the deployment toil of where a waypoint proxy runs and how it scales. Gloo Mesh, our service-mesh management product built around Istio, provides deep automation and simplification around Zero Trust including support for ambient mesh.

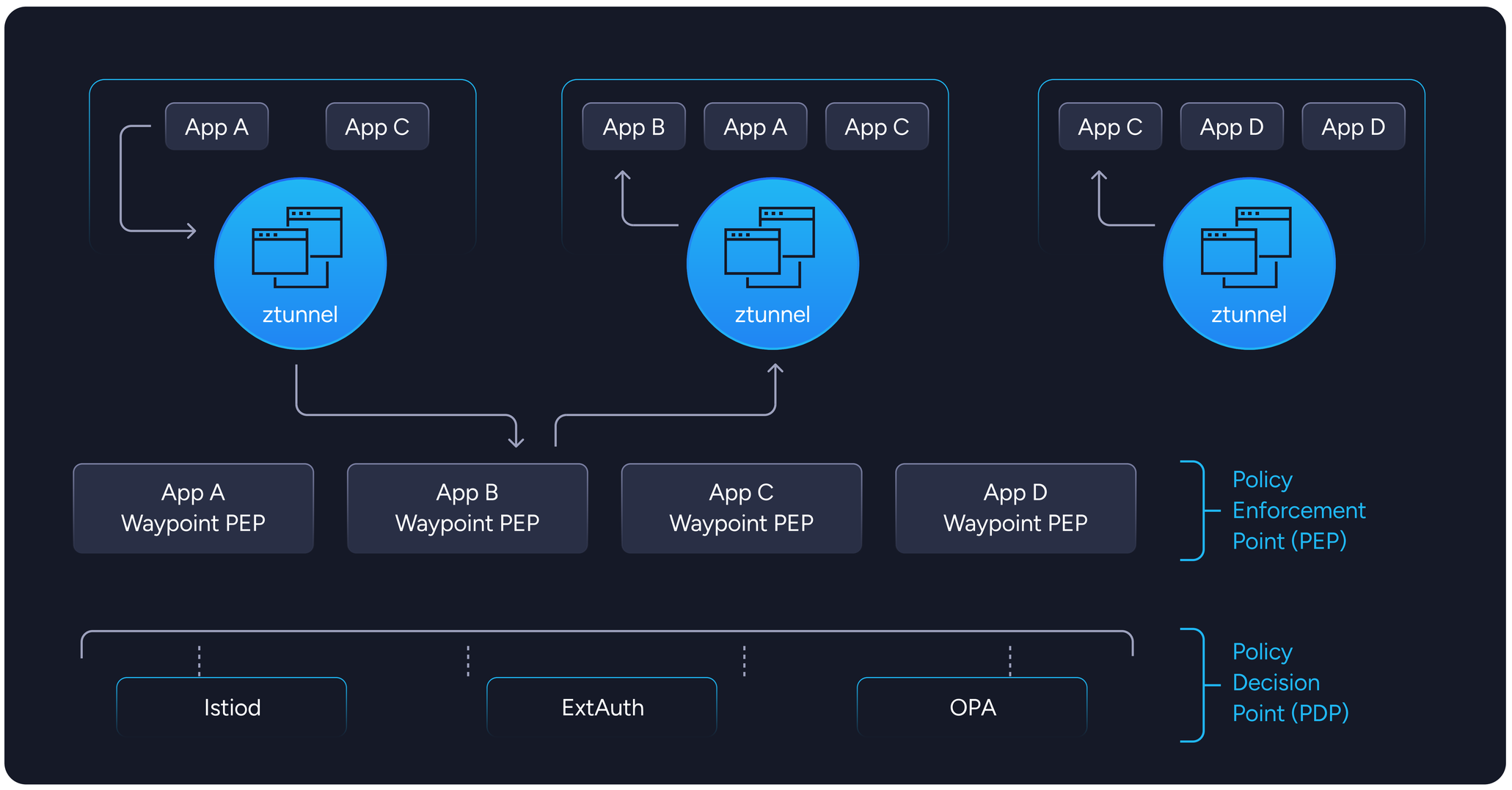

Figure 5: L7 Policy Enforcement Points will rely on a Policy Decision Point to determine whether to allow traffic to a workload/resourceEnforcement Point (PEP)

The characteristics of tenancy for Layer 7 are similar in the ambient mesh to a sidecar deployment. L7 capabilities are not shared for multiple identities in a single L7 Proxy. App A will have its own waypoint proxy proxies, App B will have its own, and so on. Configuration or extensions (plugins, extensions, etc) that are specific for a particular workload can be isolated from other workloads by not sharing these waypoint proxies across multiple workloads.

References:

1 https://istio.io/latest/blog/2022/introducing-ambient-mesh/

2 https://www.paloaltonetworks. com/blog/2017/11/trust-is-a- vulnerability/

3 https://www.whitehouse.gov/ briefing-room/presidential- actions/2021/05/12/executive-order-on-improving- the-nations-cybersecurity/

4 https://www.whitehouse. gov/wp-content/uploads/2022/01/M-22-09.pdf

5 https://csrc.nist. gov/publications/detail/sp/ 800-207/final

6 https://cloud.google.com/ beyondcorp

Conclusion

Ambient mode is a powerful new sidecar-less data plane for Istio that can be used to more easily and transparently enable a Zero Trust Architecture to satisfy security and compliance requirements. A quick recap of the benefits of this model:

- Fine-grained application and user-level authorization policies
- Workload-specific identity without relying on brittle network-location-based identity
- Secure and authenticated traffic through mTLS without the applications having to know about it
- Flexible and dynamic policy enforcement that can be driven by the Istio control plane or external policy engines or policy decision points (PDPs)
- A concrete implementation model that more closely resembles the logical architectures defining ZTA
- Supports hybrid workloads
- Observe and track all access for later tuning or audit

As Solo.io is a co-founder of the ambient mesh sidecar-less architecture and leads the development upstream in the Istio community, we are uniquely positioned to help our customers adopt this architecture for production security and compliance requirements. Please reach out to us to talk with an expert.

For more information

visit www.solo.io/products/gloo-mesh

SOLO.IO

contact@solo.io

www.solo.io

About Solo.io

Solo.io, the application networking company, delivers API infrastructure from the edge to service mesh, helping enterprises adopt, secure, and operate innovative cloud native technologies. APIs drive microservices and cloud native technologies, forming the foundation for developers, partners, and customers to interact with application services quickly, effectively, and securely. Solo.io brings developer and operations tooling to manage and federate security and traffic control and tie together the integration points to enable and observe the application network.