tetrate

White Paper

Zero Trust Architecture

by Zack Butcher

www.tetrate.io

v1.1

Background

Traditional datacenter network security architectures attempt to build strong perimeter walls around an edenic internal garden. This citadel and mote model has always had a fundamental weakness such that, when (not if) an invader penetrates the perimeter, they have the run of the whole garden. While not new, this weakness has been exacerbated by the twin trends of proliferation of entry points into and the expansion of workloads beyond the datacenter.

Zero trust network architectures offer a way forward to address the weaknesses of perimeter-based security by taking a stance that the network is inherently hostile; that safety behind the perimeter is an illusion and the barbarians have already crashed the gates.

While zero trust requires a significant rethink of the status quo, it is far from a lofty, unattainable goal. There are tools available right now to start implementing zero trust network architectures. And you can apply those tools and practices incrementally starting from where you are now so that you don't need to wholesale re-architect your entire network security infrastructure.

Traditional security model weaknesses

Perimeter security is weak for reasons similar to why modern armies abandoned large-scale fixed fortifications: once penetrated, the battle is lost; and the perimeter will be penetrated, eventually.

Perimeter security alone provides poor granularity of control. If all traffic inside the perimeter is trusted, a breach leaves everything inside vulnerable. While this may have been manageable when network services measured in the dozens and access could be strictly limited by physical location, the explosion of services to hundreds or thousands all talking to each other with the same level of access have made the current state of the art untenable—especially since one compromised service allows pivots to many others.

Over the years, business requirements have chipped away at the integrity of the perimeter. By necessity, holes have been punched through the firewall, leading to multiple exposed entry points and a proliferation of hard-to-manage firewall rules, leaving the perimeter more of a Maginot Line than a mote and castle wall.

In the face of the near-obliteration of the perimeter, newer efforts to improve on the perimeter security model like microsegmentation and software-defined networking have helped by decreasing the attack surface around services. But they, too, are only partial solutions that come at the expense of increased complexity and an explosion of configuration rules. Segmentation still offers poor granularity. For example, isolating a web server and database server may reduce the attack surface around those services, but the web server may support many applications and the vulnerabilities they may each introduce are still opaque.

Add the extension of services to the cloud and everything gets harder. The weaknesses inherent in traditional security models become untenable.

Tenets of Zero Trust

"*Trust is not the desired state; trust is the failure point you want to avoid.*" [1] — John Kindervag

Zero-trust is an approach—a way of thinking about network security—more than it is any particular architecture or implementation. It starts from an assumption that there are no safe places on the network. You should treat your datacenter, like it or not, as if all of its data and services are exposed to the public Internet.

In the zero trust model, unlike traditional perimeter security, reachability does not imply authorization. Zero trust seeks to shrink implicitly trusted zones around resources, ideally to zero. In a zero trust network, all access to resources should be:[2]

- **Authenticated** and dynamically **authorized**, not only at the network layer and the service-to-service layer, but also at the application layer. Network location does not imply trust. Service identity and end-user credentials are authenticated and dynamically authorized before any access is allowed.

- **Bounded in time**: authentication and authorization are bound to a short-lived session after which they must be re-established.

- **Bounded in space**: the perimeter of trust around a service should be as small as possible.

- **Encrypted**, both to prevent eavesdropping and to ensure messages are authentic and unaltered.

- **Observable**, so the integrity and security posture of all assets may be continuously monitored and policy enforcement continuously assured. Also, insights gained from observing should be fed back to improve policy.

Why is it better?

- Accessibility is not authorization. Unlike perimeter security, access to a service is not granted solely because that service is reachable. It must be explicitly authenticated and authorized as well.
- Authenticated and authorized workloads are protected from perimeter breaches.
- Bounding in time limits the risk of compromised credentials.
- Bounding in space allows for high granularity of policy enforcement.
- Dynamic policy enforcement ensures authorization policy is up to date.
- Encryption limits reconnaissance and provides for authenticity of communication.
- Fine-grained observability allows real-time assurance that policy is being enforced and post-facto auditability of how policy has been enforced historically plus the necessary data for troubleshooting and analysis.

[1] Kindervag, John. “Clarifying What Zero-Trust Is––and Is Not.” Palo Alto Networks Blog, 6 Aug. 2018, https://blog.paloaltonetworks.com/2018/08/clarifying-zero-trust-not

[2] Rose, Scott, et al. (2020) Zero Trust Architecture (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-207, August 2020. https://doi.org/10.6028/NIST.SP.800-207

A litmus test for a zero trust system is that applications deployed in it wouldn't have to change were they to be publicly exposed. If properly implemented, a zero trust security architecture would be as secure when operating exposed to the public Internet as it is behind a firewall.

When do I need it?

While every organization may benefit from adopting zero trust tenets, it's unrealistic to expect decades of infrastructure and business processes to transform wholesale to a new paradigm.

Specific pressures may prompt you to do so earlier rather than later. When your infrastructure spans disparate physical providers, e.g., split on-premises and cloud deployments or hybrid cloud deployments, the complexity and fragility of applying VPNs and NATs at scale over these extended networks may make it cost-effective and risk-efficient to apply zero trust networking principles to those deployments in the short term.

ZTA Components

NIST posits three logical components to implement dynamic authorization and authentication:

1) a **policy engine (PE)** responsible for determining authorization;

2) a **policy administrator (PA)** for establishing and/or shutting down the communication path to a resource based on results from the policy engine;

3) a **policy enforcement point (PEP)** that sits between the subject making a request and the destination resource that enables, monitors, and terminates the connection between them. [3]

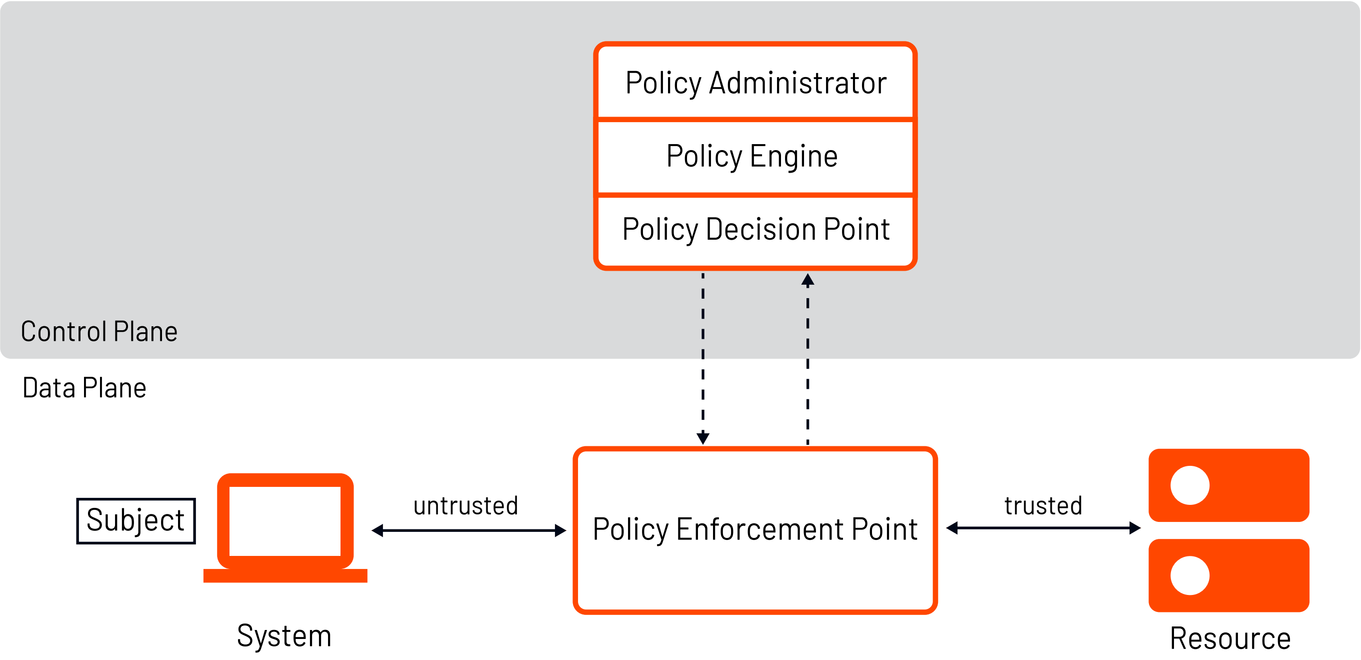

Figure 1

[3] Rose, Scott, et al. (2020) Zero Trust Architecture (National Institute of Standards and Technology, Gaithersburg, MD), NIST Special Publication (SP) 800-207, August 2020. https://doi.org/10.6028/NIST.SP.800-207

In this model, all workloads requested by the subject must have an identity that can be authenticated and authorized at the PEP. The policy decision point enforces policy over these identities and enforces authentication and authorization prior to allowing access. Here, authorization is based on fine-grained policy; reachability does not count as authorization. The PEP in the data plane allows for observability of systems at runtime and ensures continuous compliance and governance controls.

Implementation

Because zero trust is less of a blueprint and more of a design philosophy, there are potentially many ways to implement a zero trust architecture. As the founders and implementers of service mesh and next-generation access control (NGAC) technologies, we posit that a service mesh coupled with NGAC provides the best foundation on which to build your zero trust architecture network.

A service mesh provides the important primitives that you need:

- Centrally-managed policy authorship
- Distributed policy enforcement points—PEPs are co-deployed with the resource access points (RAPs)
- Built-in support for workload identity based on a runtime identity rather than network location
- Built-in support for application-level authentication and authorization of end users, allowing for global and consistent policy enforcement for every application in the mesh
- Encryption for data on the wire
- Built-in observability

The mesh provides operational assurances you can use when you deploy your authentication and authorization systems to make them safer and easier to manage. We can readily redraw NIST's logical architecture represented in Fig. 1 in terms of the components available today in a service mesh:

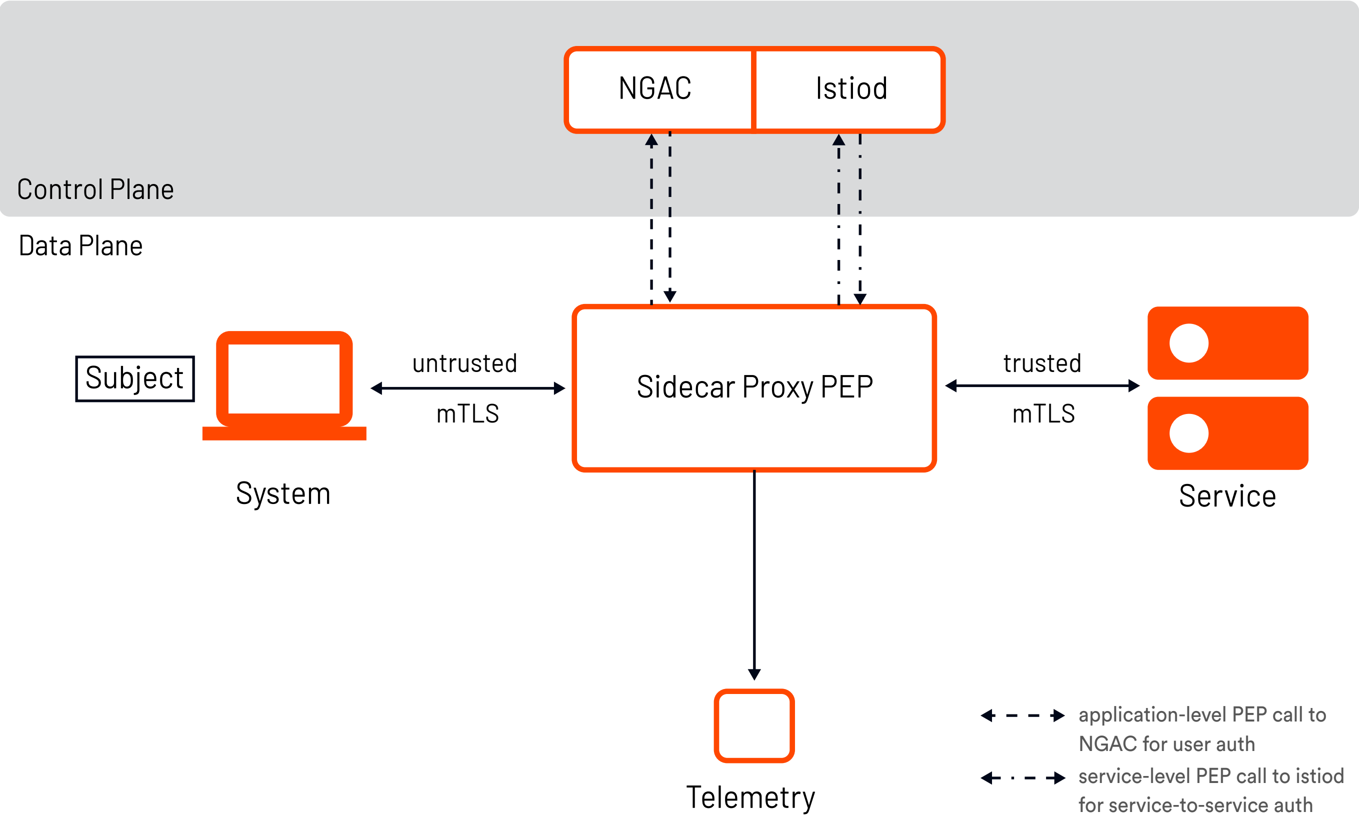
NGAC
Istiod
Control Plane
Data Plane
Subject
untrusted
mTLS
Sidecar Proxy PEP
trusted
mTLS
System
Service
Telemetry
application-level PEP call to NGAC for user auth
service-level PEP call to istiod for service-to-service auth

Figure 2

The transparent nature of a service mesh allows for incremental adoption without a wholesale reimplementation of your entire security infrastructure and business processes. The decoupling of application, deployment, and security concerns afforded by the mesh means you can start to build a zero trust architecture on your existing infrastructure without disrupting your business processes or your application delivery lifecycles.

Case Study: DoD Platform One

"*I don't think you can succeed today, honestly, at any kind of meaningful scale without a service mesh; maybe in 2018, but not in 2020 and beyond.*" [5]

— Nicolas M. Chaillan, U.S. Air Force Chief Software Officer

The Department of Defense, under the auspices of USAF Chief Software Officer Nicolas M. Chaillan, has revolutionized the way it develops and operates software. Platform One, the team led by Chaillan to develop the DevSecOps practice across the DoD, provides multiple enterprise services to bring “automated software tools, services and standards to DoD programs so that warfighters can create, deploy, and operate software applications in a secure, flexible, and interoperable manner.”[6] Those services include their DevSecOps Platform (DSOP), a collection of approved, hardened CNCF-compliant Kubernetes distributions, along with Istio, infrastructure as code playbooks, and hardened containers.

[4] Chaillan, Nicholas M. “Istio Success Story: USAF CSO Nicolas Chaillan Discusses Istio and DevSecOps Adoption with Tetrate,” YouTube, 21 Oct. 2020, https://www.youtube.com/watch?v=bv9OIl_-v5k&t=75s

[5] DoD Enterprise DevSecOps Initiative (DSOP). Office of the Chief Software Officer, U.S Air Force, 2021, https://software.af.mil/dsop/

According to Chaillan, “having a centralized, government-furnished DevSecOps stack that teams can come and use is game-changing.” Where software update cycles used to span years, the DoD is now pushing code “every day, multiple times a day… **saving an average of 12 to 18 months** for every five years of initial planned time per program.”

Istio is a primary pillar of their architecture, providing a rich set of service mesh capabilities and, especially, enabling a zero trust model. When asked why they use a service mesh instead of just ingress controllers, Chaillan cited not just that mTLS encryption in transit is available to every application by default, but that “once you move to microservices and containers, you have to manage east-west traffic and that's completely different from north-south… you need to **make sure lateral movement is limited**. You don't want a bad actor to get access to one container and be able to… laterally move to others.” In addition to SSO and mTLS, Platform One’s architecture uses Istio to enforce east-west whitelisting and provide policy enforcement points between containers.

The mesh takes policy enforcement out of the application stack and moves it transparently to the sidecar proxies. Platform One was able to consolidate multiple “snowflake” application-level SSO and encryption implementations built independently by different app teams into one, hardened single sign-on and authorization library that can be used by all applications, enterprise-wide. This takes the burden of building security uniquely into each app from development teams. It also dramatically reduces vulnerability risk by standardizing on a single, well-vetted implementation.

Says Chaillan, “if you don't use a service mesh, you end up having to do it per language, per microservice. And now you're tightly coupled. And, let's say, you need to update your encryption bits, now you have to update all your containers where you could just update your sidecar service mesh and now you're decoupled. That alone is worth using a service mesh.”

Conclusion

The perimeter security model and its incremental successors are too brittle and complex to serve modern application development and deployment practices. The way apps are built now requires a dynamic, flexible security solution and one that can be both centrally managed and universally available for all app development teams. Zero trust architectures provide much-needed improvements in security at both the network and application level and a service mesh provides the most powerful, dynamic, and flexible way to implement zero trust.

By dveploying globally-managed policy enforcement points between all services and all applications, a service mesh provides shim points to insert zero trust capabilities like SSO, mTLS, and dynamic authorization. By offloading security responsibilities from individual applications globally to the mesh, it’s possible for organizations to adopt zero trust principles incrementally, without rewriting applications or upending existing processes.