MAZU: Zero Trust Architecture for Service Mesh Control Planes

William & Mary
CHARTERED 1693

Aashutosh Poudel
apoudel01@wm.edu

Pankaj Niroula
pniroula@wm.edu

Collin MacDonald
cmacdonald01@wm.edu

Lily Gloudemans
algloudemans@wm.edu

Stephen Herwig
smherwig@wm.edu

/etc/lab
Extending Trust in Computing Lab

Background

Microservices: Modern software architecture where applications are split into small, independent services that communicate through APIs - enabling elastic scaling and fault isolation

Service Mesh: Infrastructure layer that manages communication between microservices, handling security and reliability without modifying the services themselves

Zero Trust Networking: Security model that requires authentication and authorization for all service-to-service communication, treating every request as potentially hostile regardless of its origin

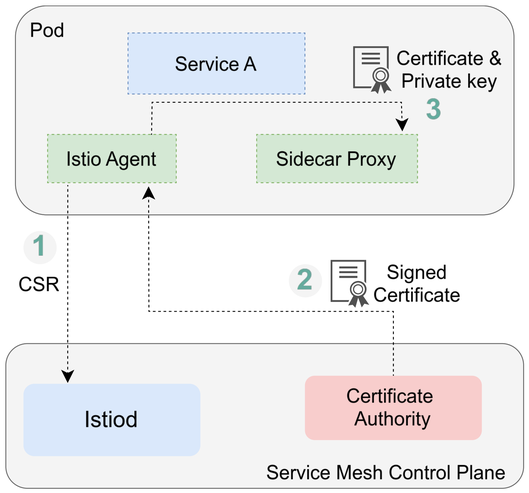


Figure 1: Identity and certificate management in Istio Service Mesh

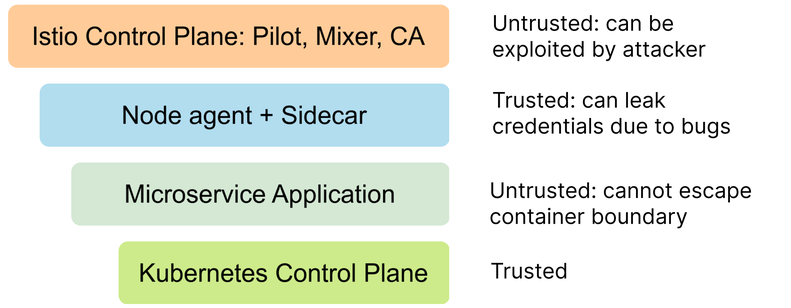
Problems

- Secure service-to-service communication in a Service Mesh is based on TLS certificates issued by mesh-local CA
- Cloud providers offering managed Service Mesh platforms have full control over security infrastructure, including the CA
- A compromised CA (resulting from misconfigured, vulnerable software, or insider threats in cloud provider) allows attackers to impersonate services and ex-filtrate unauthorized information

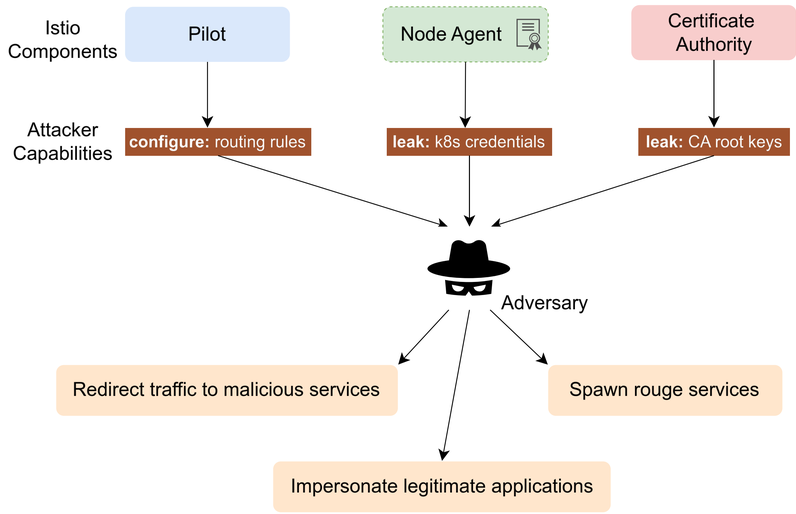
Research Questions

Is it possible to reduce trust in the service mesh's control plane while maintaining microservice compatibility and performance?

Threat Model



Example Attack Paths



The compromise of a CA gives attackers significant advantages since CAs hold extensive trust in the security system. To reduce this risk, we can limit our reliance on CAs by replacing them with a decentralized public trust system.

Building blocks: RBE

Registration Based Encryption (RBE) helps achieve the notion of decentralized trust using a public key curator.

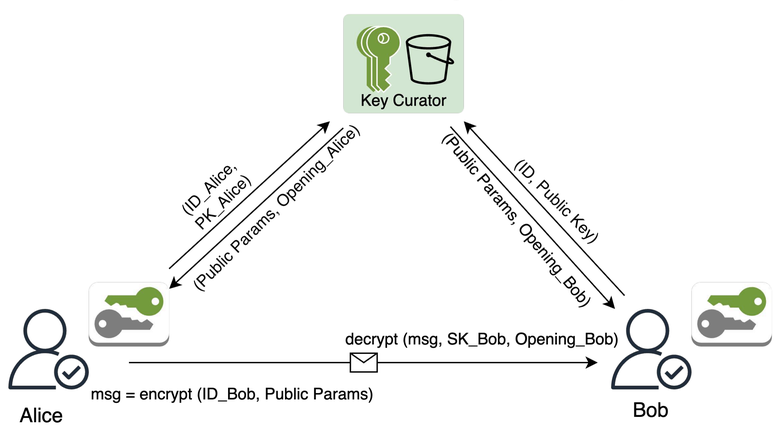


Figure 2: How RBE works

Mazu Design

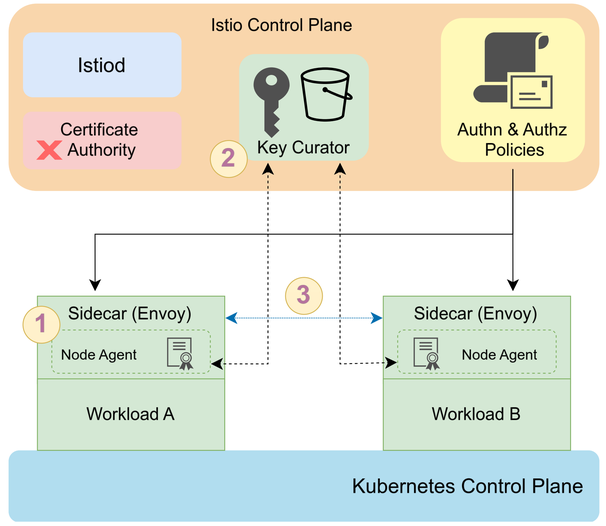


Figure 3: Mazu's Implementation on Istio Service Mesh

1. **Workload Identity and Certificate:** Node agents register *workload IDs* (hash of Kubernetes-signed *admin token*) with the key curator and generate self-signed certificates containing both the *ID* and *admin token*.
2. **Key Curator:** Key Curator accepts node agent registration requests and responds to queries for updated public parameters.
3. **mTLS Certificate Validation:** During mTLS setup, the Envoy sidecar accepts the connection:
   a. if *admin token* is valid confirming authorized node deployment
   b. if the *workload ID* is registered with the Key Curator

Preliminary Evaluations

Mazu adds only 0.17ms latency with 16 and 64 concurrent connections compared to mTLS Istio.

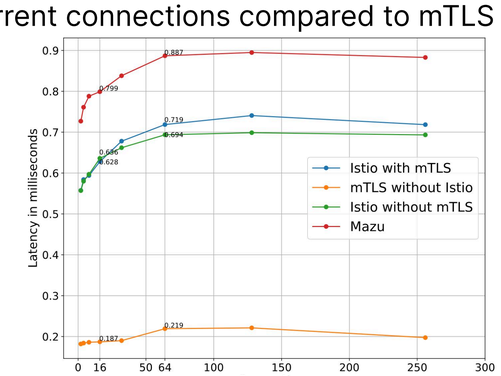


Figure 4: Latency vs connections at p90, 1000 qps over 120 sec

References

- S. Garg, M. Hajiabadi, M. Mahmoody, and A. Rahimi, "Registration-Based Encryption: Removing Private-Key Generator from IBE," Lecture Notes in Computer Science, pp. 689–718, 2018, doi: https://doi.org/10.1007/978-3-030-03807-6_25.
- N. Glaeser, Dimitris Kolonelos, Giulio Malavolta, and A. Rahimi, "Efficient Registration-Based Encryption," Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, pp. 1065–1079, Nov. 2023, doi: https://doi.org/10.1145/3576915.3616596.