

CiliumCon
NORTH AMERICA

Handing a Proxy the Keys to Your Pod: Inside Cilium's Sidecarless mTLS

Quang Nguyen, Microsoft · Robin Gögge, Isovalent at Cisco

Motivation



1. **Security**
   Encrypt traffic and authenticate workloads.

2. **Compliance**
   Ensure you meet requirements set by internal or external parties.

3. **Innovation**
   E.g. AI agents accessing other services and data autonomously.

Cilium Before ztunnel

CiliumCon NORTH AMERICA

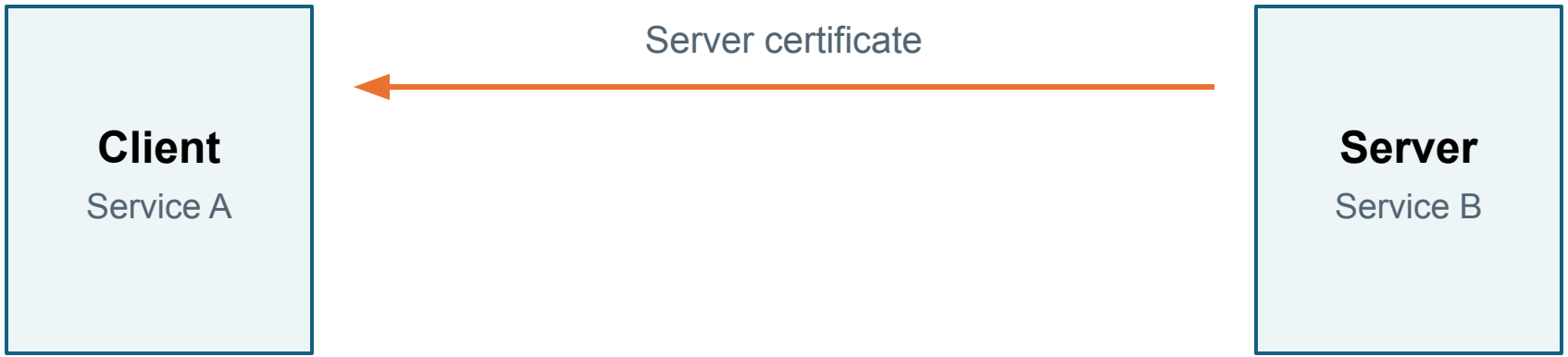
	Transparent encryption WireGuard / IPsec	Mutual Authentication Deprecated
Application traffic encryption	**Yes, between nodes**	**No**
Workload authentication	**No**	**Yes, via SPIFFE/SPIRE**
	IP-layer tunnels between nodes	Out-of-band TLS handshake between Cilium agents

Encryption and workload authentication were separate capabilities

What is mTLS?

TLS

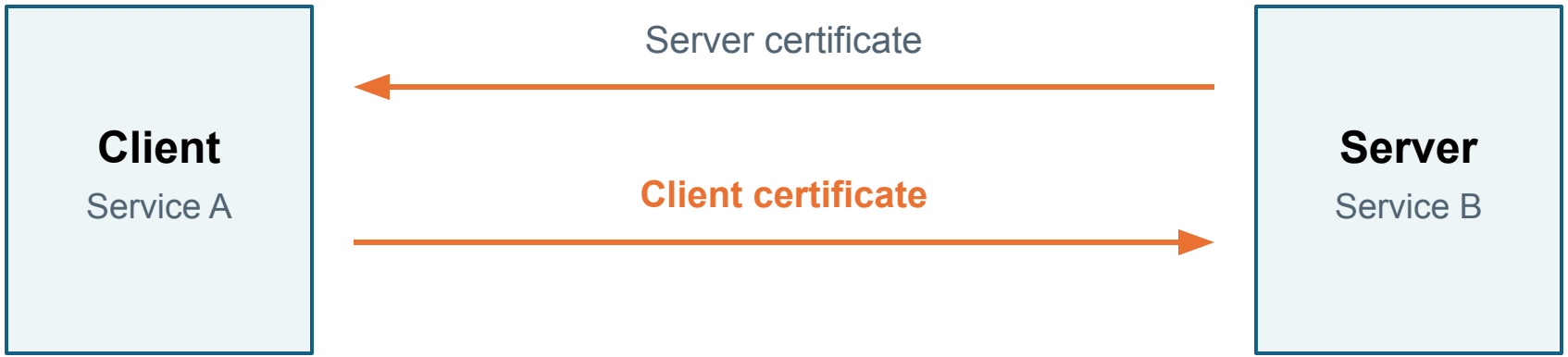
The client authenticates the server.



TLS encrypts application traffic and detects tampering.

Mutual TLS

The server also authenticates the client.



TLS encrypts application traffic and detects tampering.

Why ztunnel?

Benefits

- Out of the box transparent mTLS L4 proxy
- Promising performance
- Combining existing CNCF projects, be "Better Together"

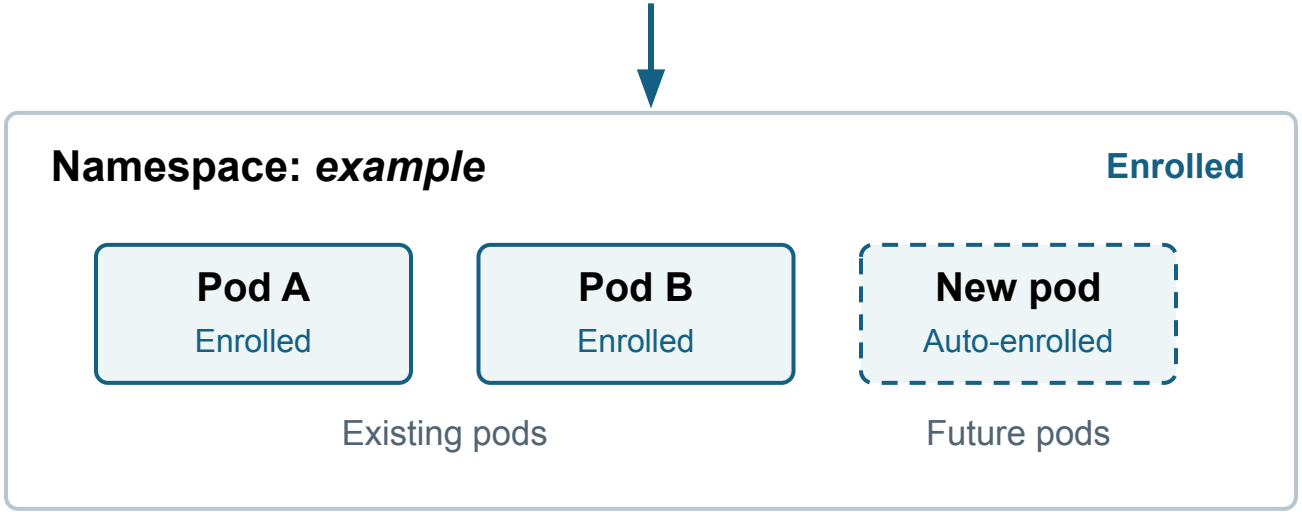
Limitations



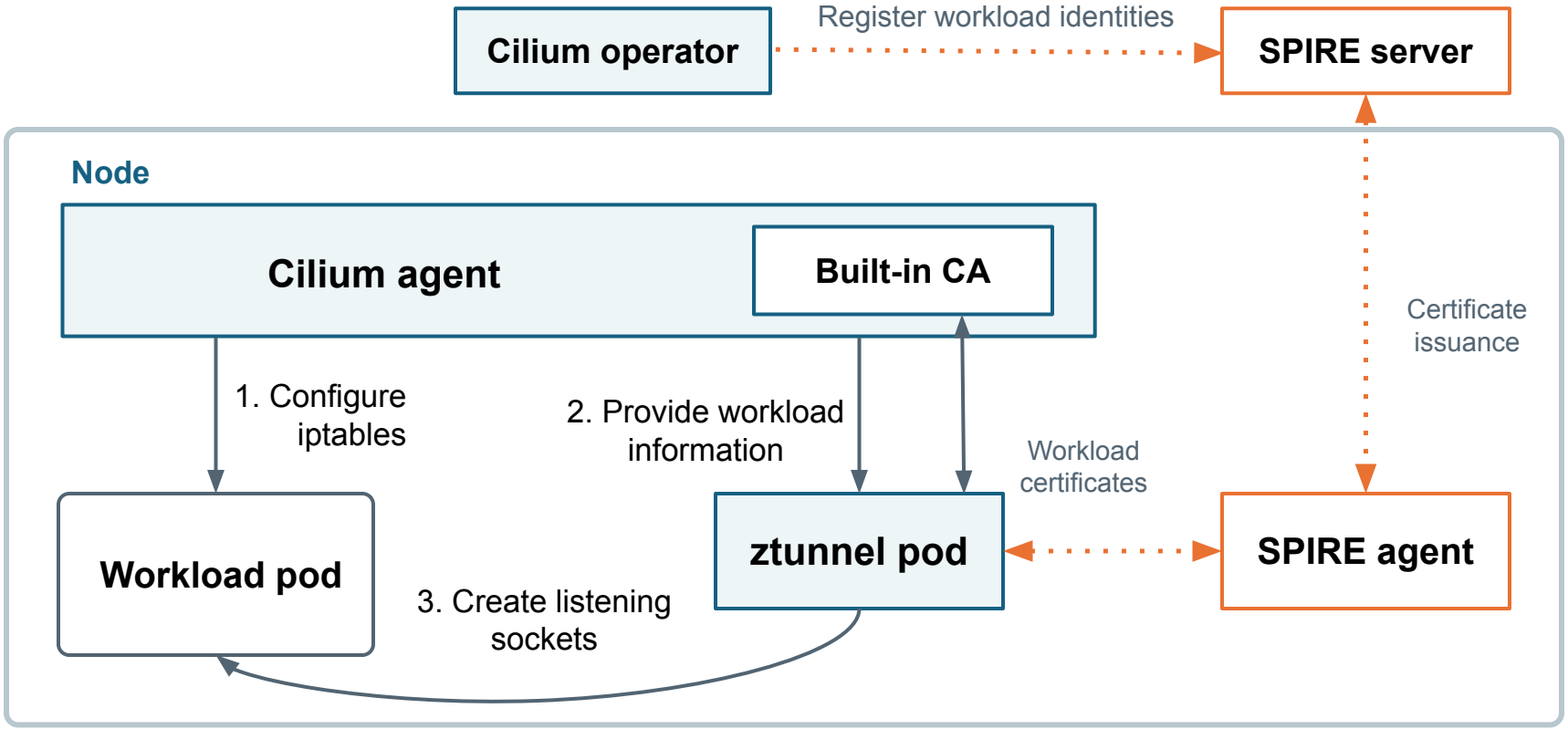
- TCP only
- Upgrade Downtime
- Limited use of CiliumNetworkPolicy

Enroll ztunnel Workloads


```
kubectl label namespace example io.cilium/mtls-enabled=true
```



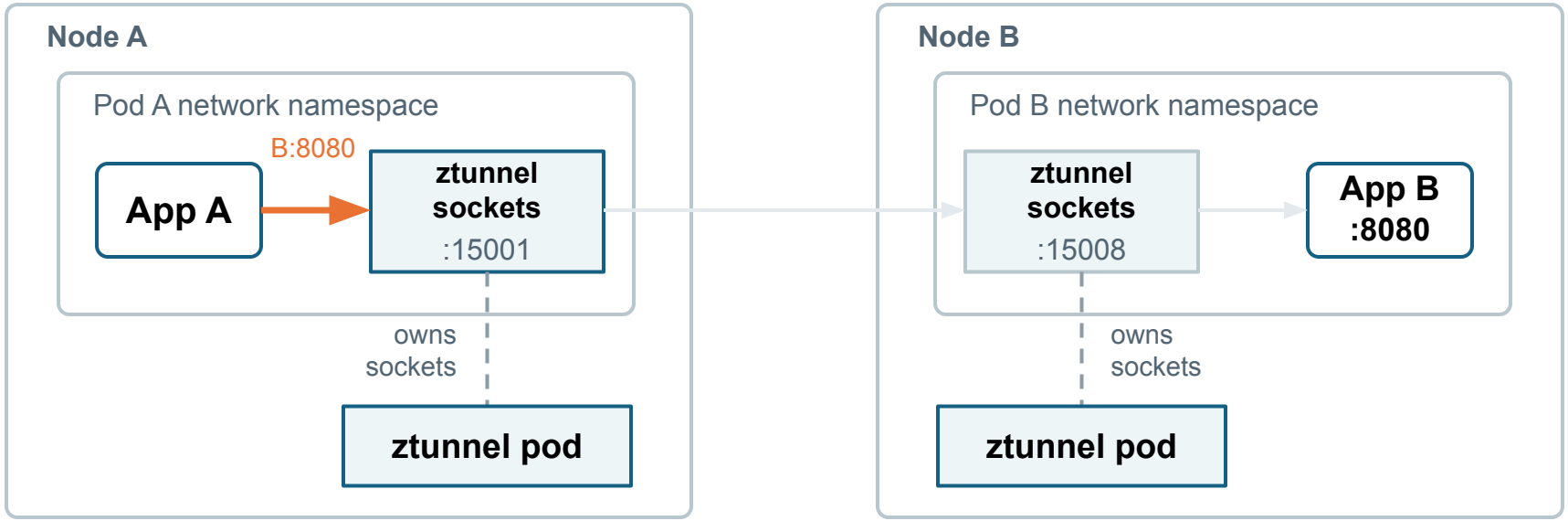
Cilium Control Plane Integration



Cilium Datapath Integration

CiliumCon NORTH AMERICA

1. Redirect the application connection

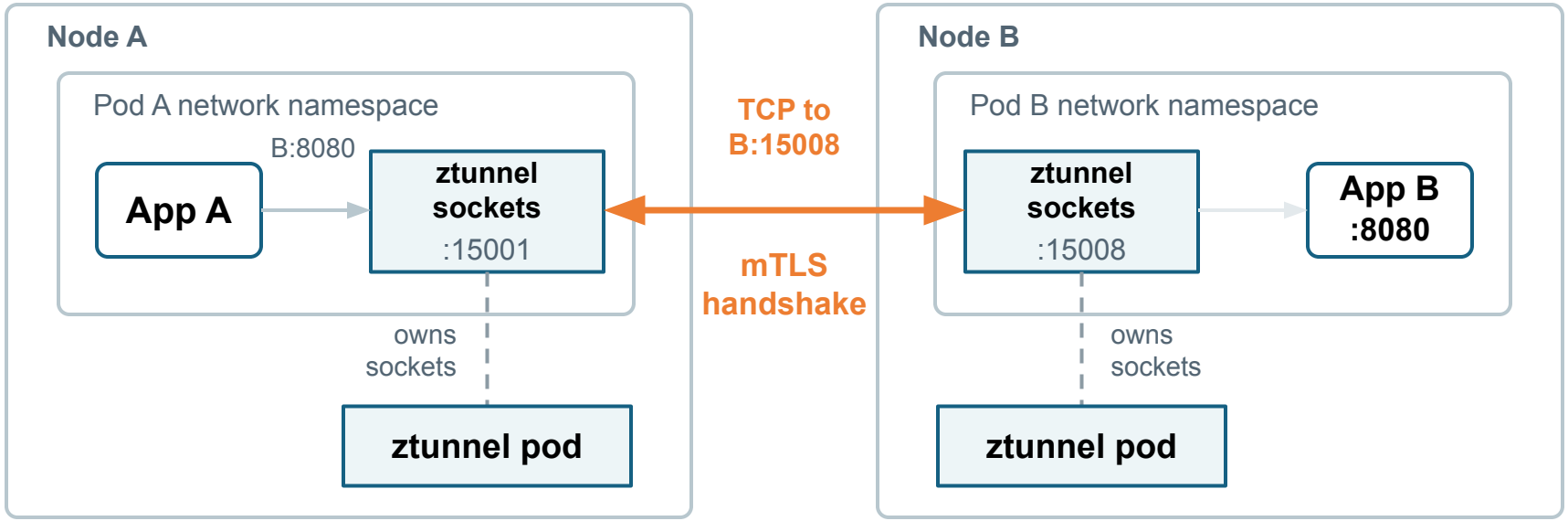


iptables redirects to :15001. ztunnel stores the original target B:8080.

Cilium Datapath Integration

CiliumCon NORTH AMERICA

2. Establish mTLS between workloads

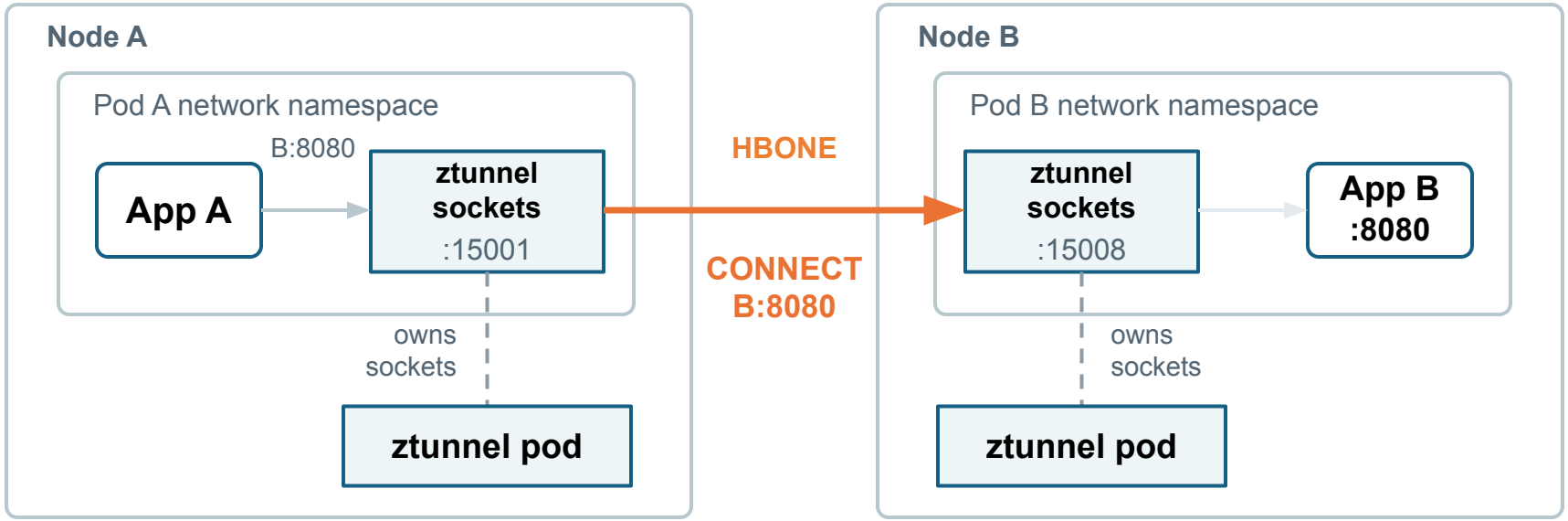


Each proxy authenticates using the workload's identity.

Cilium Datapath Integration

CiliumCon NORTH AMERICA

3. Carry the original destination in HBONE

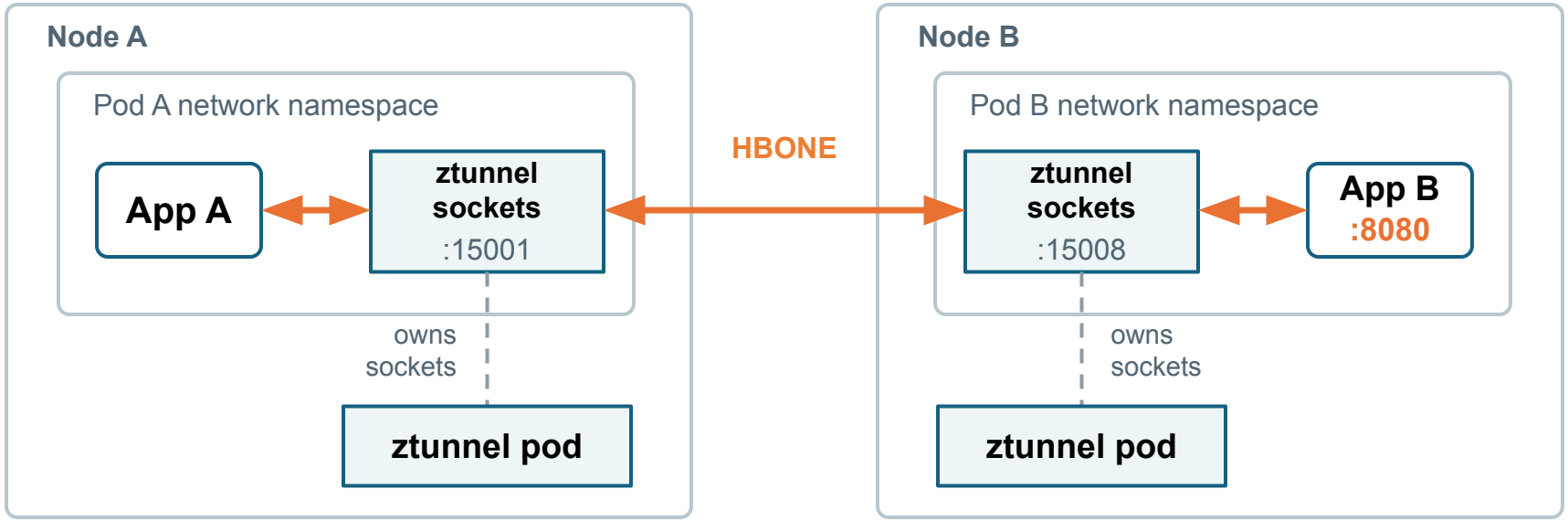


HTTP/2 CONNECT carries B:8080 inside the encrypted connection.

Cilium Datapath Integration

CiliumCon NORTH AMERICA

4. Connect to the destination application



The destination app receives TCP on :8080. Replies follow the same path.

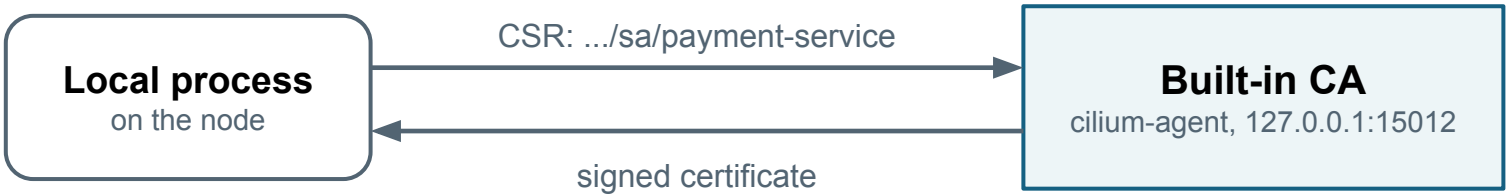
Workload Identity

CiliumCon NORTH AMERICA

`spiffe://cluster.local/ns/shop/sa/payment-service`

trust domain | namespace | service account

An SVID, the workload certificate, carries this name. The name is not secret.



✓ **CSR signature is valid**
The caller holds the private key.

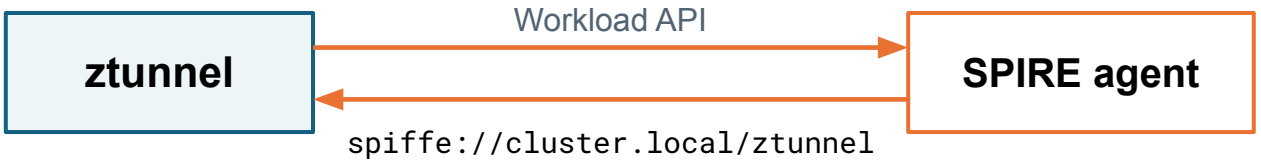
✓ **payment-service runs on this node**
Co-location, not proof of the caller.

Neither check proves that the caller is payment-service.

SPIRE Broker API

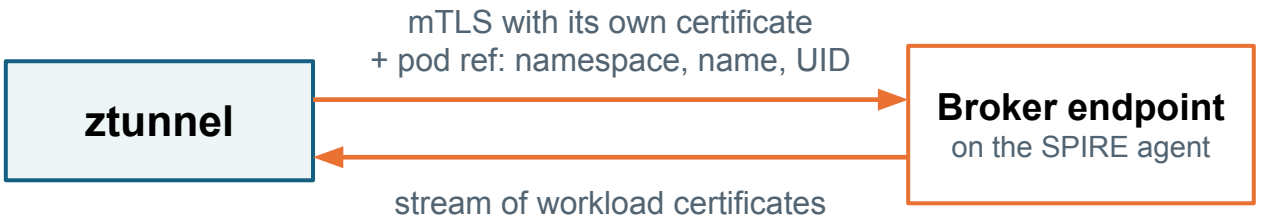
CiliumCon NORTH AMERICA

1. ztunnel gets its own identity



SPIRE attests the calling process.

2. ztunnel asks for workload certificates



Allowed broker?

RBAC for this pod?

The caller is the proxy. The target is the pod.

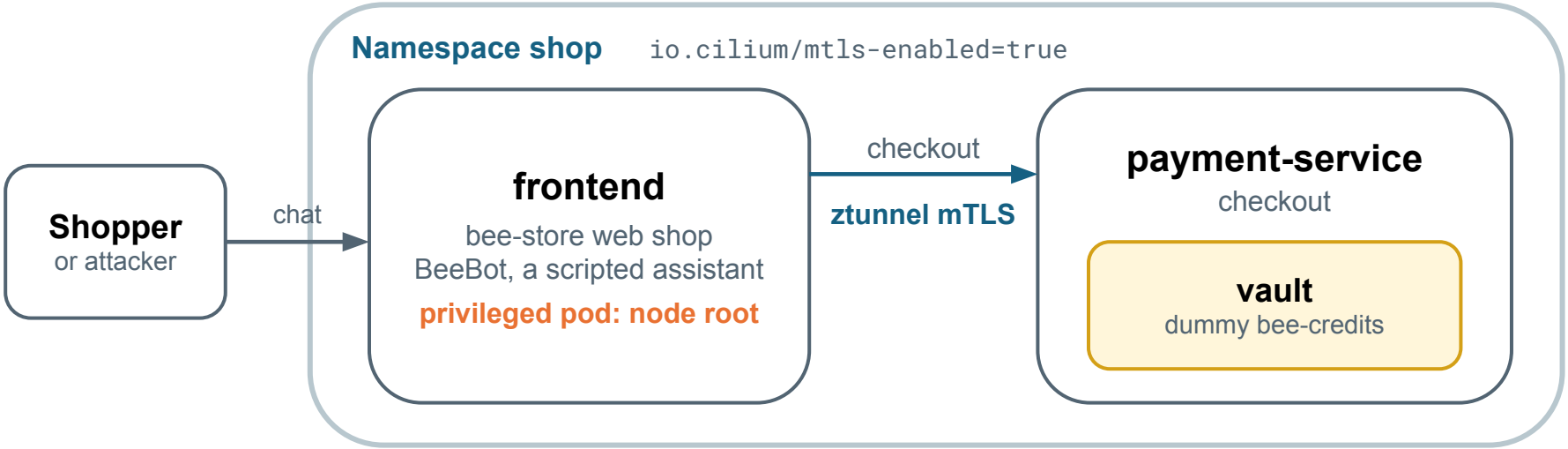
Naming a pod does not authorize the caller.

Demo

Demo Setup: The Bee Store

CiliumCon NORTH AMERICA

bee-store sells bees: Buzzy the Bumblebee, Queen Bee-yoncé, Barry B. Benson, and more.



Attacker's goal: **get payment-service's identity, then open the vault.**



Demo Assumptions

CiliumCon NORTH AMERICA

1. **I'm not great at security, so agent2shell works.**
   BeeBot runs any shell command you ask nicely. Scripted, not a real jailbreak.

2. **I handed BeeBot the keys, too.**
   Its pod has node root, so it reaches the built-in CA and the SPIRE agent socket.

3. **The vault only opens for payment-service.**
   Pretend it's hard-coded. Ours checks nothing, and identity rules are future work.

So, can BeeBot get payment-service's identity?

Same Attack, Two CAs

CiliumCon NORTH AMERICA

BeeBot makes an unsafe tool call. The attacker gets a frontend shell, then node root.

ACT 1

Built-in CA

Ask the CA for payment-service's certificate.

Watch for: **certificate issued**

ACT 2

SPIRE

Ask SPIRE for payment-service by name, then ask the broker.

Watch for: **no identity issued**

Root has other paths. We test one: the local certificate issuer.

Prepared clusters. Act 1: development build. Act 2: Cilium 1.20 with a local broker-enabled ztunnel. Neither is stock Cilium 1.19.

Demo Results

CiliumCon NORTH AMERICA

Act 1 · Built-in CA

```
x509.CertificateRequest{
  URIs: ".../sa/payment-service",
}
```

✗ Signed. The attack worked.

ztunnel then saw the attacker as payment-service.

Act 2 · SPIRE

```
JWTSVIDRequest{
  spiffe_id: ".../sa/payment-service",
}
```

✓ Refused. The attack failed.

`PermissionDenied: no identity issued`

Second try on the broker path: refused at TLS.

The built-in CA trusts the name in the request.
SPIRE attests the caller.

Future Work

CiliumCon NORTH AMERICA

IN PROGRESS

SPIRE as the CA

With ca.type=spire, SPIRE signs every workload certificate.

Today the built-in CA signs in every mode.

NEXT

L4 and L7 policy

Policy for the traffic inside the tunnel.

Today L4 policy only sees the HBONE port 15008.

POSSIBLE

Authorization by identity

Allow or deny traffic by SPIFFE identity.

Today Cilium sends ztunnel no authorization rules.

POSSIBLE

Kata VM support

Each pod runs in its own VM. The keys stay outside, in ztunnel.

Today Kata's default tc redirect skips the iptables capture. Kata 4.1 added an experimental L3 mode that the capture can see.

Upstream Contributions

CiliumCon NORTH AMERICA

Project	Change	By	Status
spiffe/spiffe	Broker API specification (#340)	@arndt-s	**Published**
spiffe/spire	Broker endpoint (#6915)	@matheuscscp	**Released in v1.15.2**
rust-spiffe	Broker API bindings (#389)	**@nddq**	**Merged, spiffe 0.19.0**
istio/ztunnel	Per-workload certificate cache (#2067)	**@nddq**	**Merged**
istio/ztunnel	DER certificates (#2131); Broker client next	**@nddq**	**In review**
cilium/cilium	Chart: run ztunnel with SPIRE (#45369)	**@nddq**	**Open**

Goal: SPIRE support in upstream istio/ztunnel, replacing the Delegated Identity path in the cilium/ztunnel fork.

Credit: @MikeZappa87 reviewed the Broker API spec, pushed for its merge, and drafted the ztunnel Broker client (istio/ztunnel#1936) that our work builds on.

CiliumCon
NORTH AMERICA

Thank You!

Docs
docs.cilium.io/en/stable/security/network/encryption-ztunnel

Code
github.com/cilium/cilium/tree/main/pkg/ztunnel

Spec
github.com/spiffe/spiffe → standards/SPIFFE_Broker_API.md

Handing a Proxy the Keys to Your Pod: Inside Cilium's Sidecarless mTLS

Session feedback

Backup

Which traffic is encrypted

CiliumCon NORTH AMERICA

Origin	Destination	Encrypted
Enrolled pod	Enrolled pod, same node or remote	**Yes**
Enrolled pod	Non-enrolled pod	**No**
Non-enrolled pod	Enrolled pod	**No**
Enrolled pod	Node	**No**
Node	Enrolled pod	**No**

Node = the host, or a pod in the host network namespace. Nodes cannot be enrolled. Traffic that is not encrypted still flows.

Source: cilium/cilium#49410 (docs update, in review)