Seamless transparent encryption with BPF and Cilium

Linux Plumbers Conference 2019

John Fastabend, Cilium

cilium

Agenda

- Goals
- Cilium Kubernetes
  - Architecture
  - Challenges
- L3 Encryption
  - IPsec 101
  - Control Plane: Pre-Shared Keys, SPIFFE/Spire
  - Datapath:IPsec
  - Performance
- L7 Encryption
  - mTLS Istio/Envoy
  - mTLS Cilium/Envoy
  - mTLS Cilium/Envoy/Sockmap
- Pain points
  - L3
  - L7

Transparent Encryption:

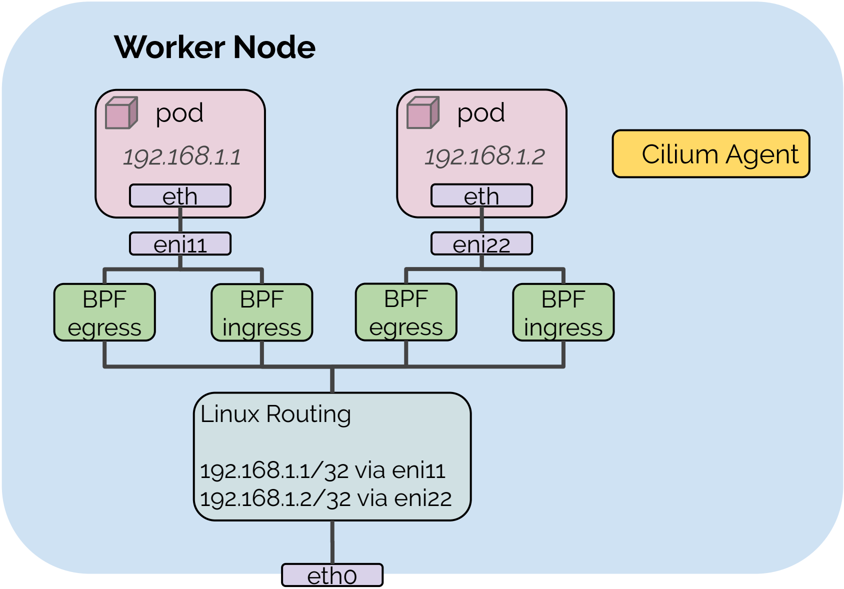
- Transparent:
  - Trust: Not required to “Trust” application to do encryption
  - Feasibility: Not possible to modify all applications we might encounter.
- Usability:
  - “--enable-encrypt”
  - Use existing deployment infrastructure, K8s, Helm, and Cilium
- Auditable:
  - What is encrypted, what is plaintext

Environment: Cilium Architecture

Cilium brings API-aware network security filtering to Linux container frameworks like **Docker** and **Kubernetes**. Using a new Linux kernel technology called **BPF**, Cilium provides a simple and efficient way to define and enforce both network-layer and application-layer security policies based on container/pod identity.

This talk: Transparent Encryption

Environment: Cilium Architecture



AWS-CNI:
- *Device plumbing*
- *IPAM (ENI)*
- *Routing*

Cilium
- *Load-balancing*
- *Network policy*
- *Encryption*
- *Multi-cluster*
- *Visibility*

Environment Challenges

- Dynamics
  - Clusters Added, Deleted
  - Nodes Added, Deleted
  - Pods Added, Deleted
  - Services Added, Deleted, Updated
  - Policies Added, Deleted, Updated

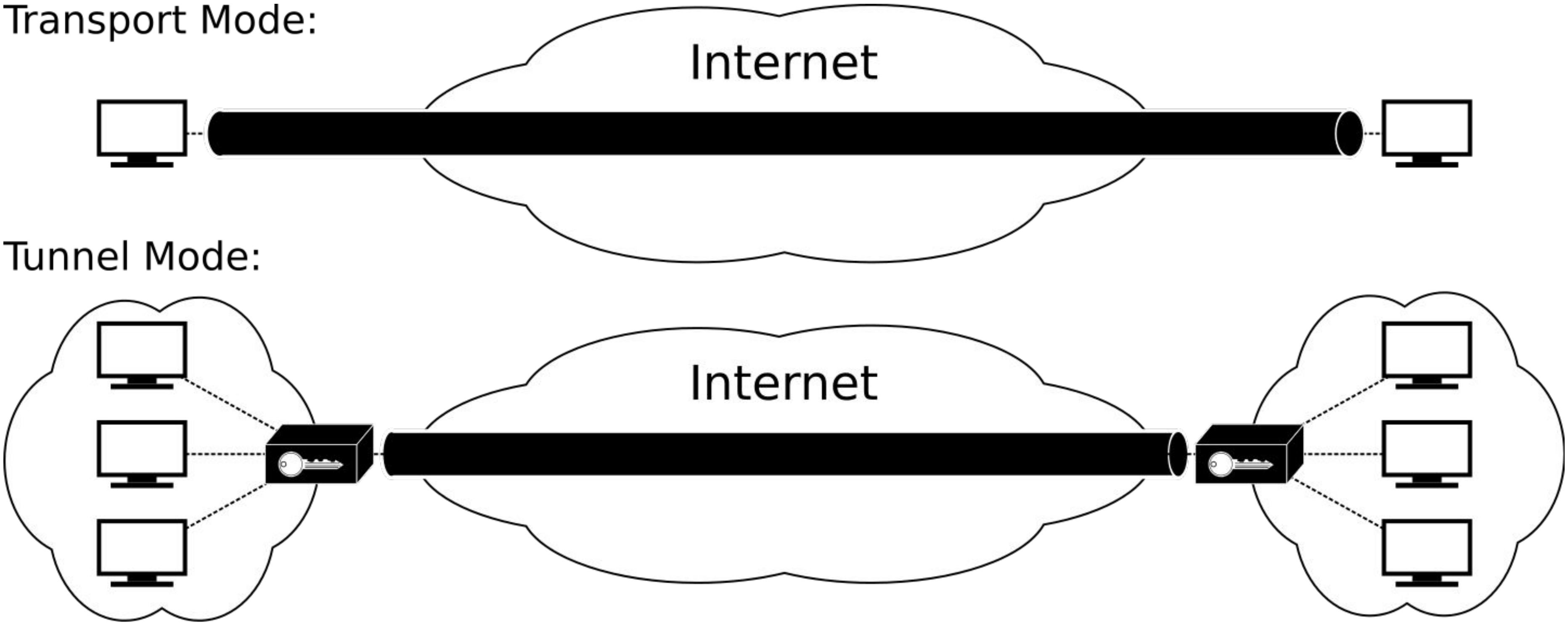
- Scale
  - Clusters: multiple is common, 50+
  - Nodes: 5k
  - Pods: 100k
  - Services: 1000+

Cilium: Transparent Encryption

- **L3:** IPsec
  - Cilium -- v1.5+
  - https://docs.cilium.io/en/latest/gettingstarted/encryption/

- **L7:** mTLS
  - Istio/Envoy
  - Istio/Envoy Cilium Accelerated -- v1.4+
  - Istio/Envoy Cilium kTLS -- v1.7 (next release targeted)
  - Cilium/Envoy -- prototype

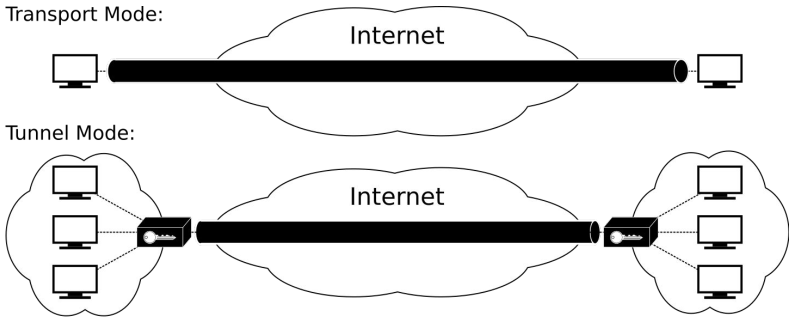
L3 Encryption: IPsec 101



https://en.wikipedia.org/wiki/File:Ipsec-modes.svg

L3 Encryption: IPsec 101

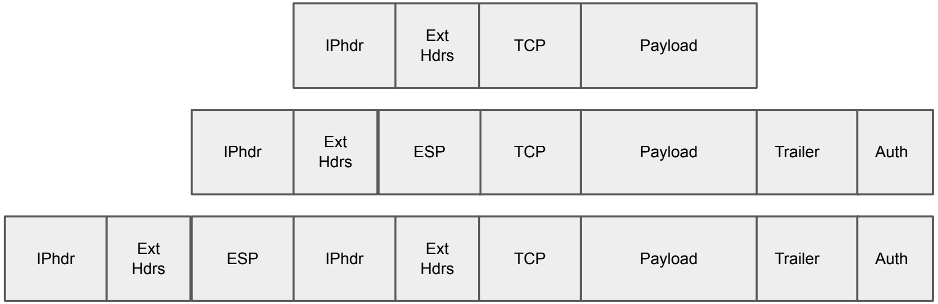
			IPhdr	Ext Hdrs	TCP	Payload				Original Packet
		IPhdr	Ext Hdrs	ESP	TCP	Payload	Trailer	Auth		Transport Mode IPsec
IPhdr	Ext Hdrs	ESP	IPhdr	Ext Hdrs	TCP	Payload	Trailer	Auth		Tunnel Mode IPsec

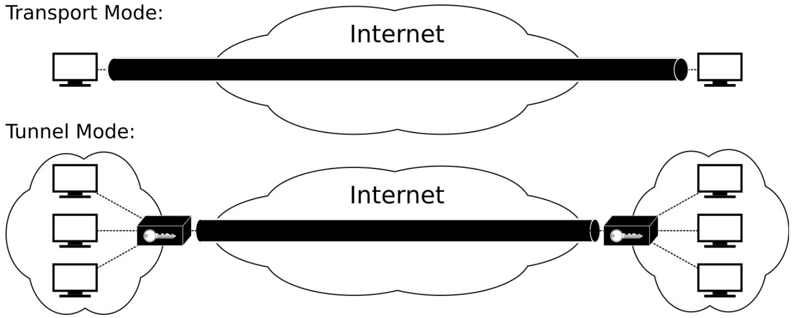


L3 Encryption: IPsec 101

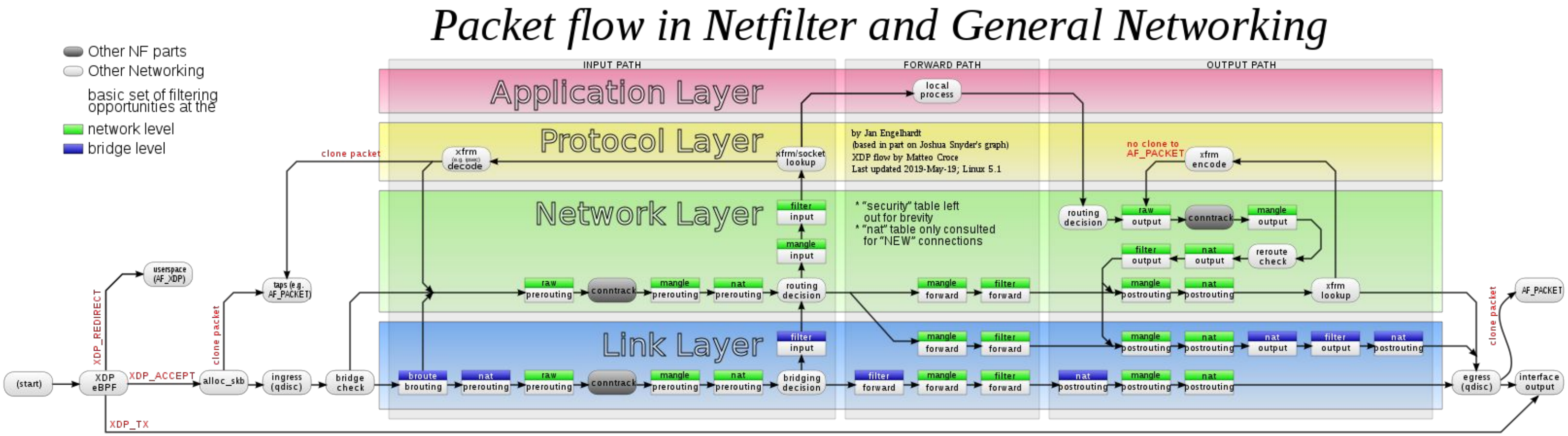
```
$ ip x p
src 10.26.38.249/16 dst 10.250.178.152/16
    dir out priority 0
    mark 0x3e00/0xff00
    tmpl src 10.26.38.249 dst 10.250.178.152
        proto esp spi 0x00000003 reqid 1 mode tunnel
```

```
$ ip x s
src 10.26.38.249 dst 10.250.178.152
    proto esp spi 0x00000003 reqid 1 mode tunnel
    replay-window 0
    aead rfc4106(gcm(aes)) * 128
    anti-replay context: seq 0x0, oseq 0x0, bitmap 0x00000000
    sel src 0.0.0.0/0 dst 0.0.0.0/0
```





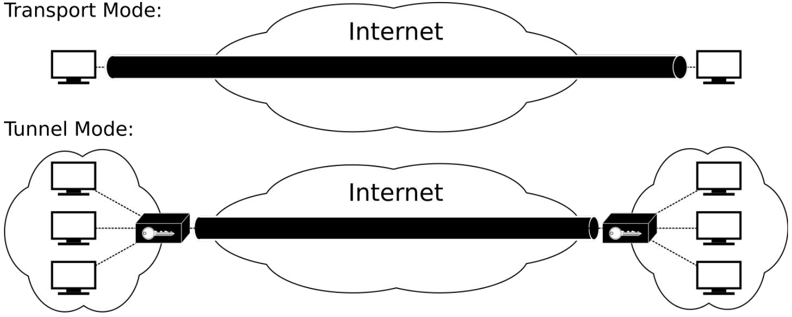
L3 Encryption: IPsec 101



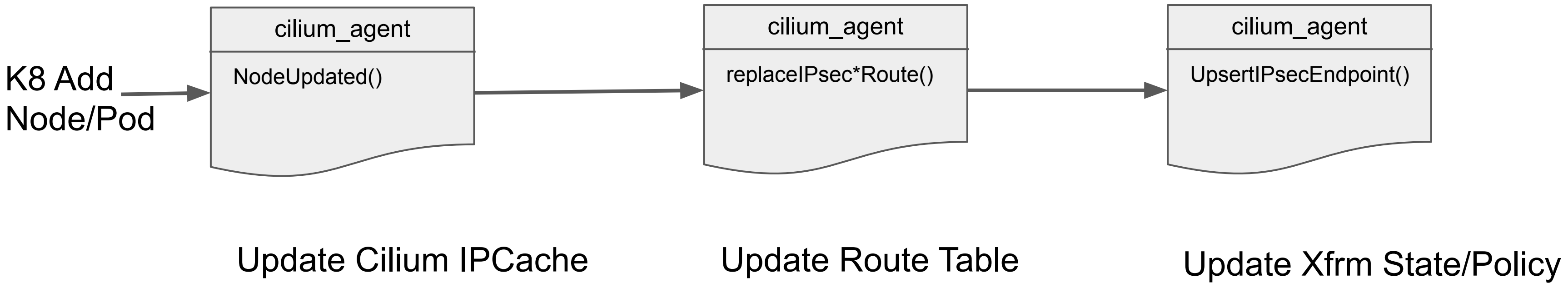
IPhdr	Ext Hdrs	TCP	Payload

IPhdr	Ext Hdrs	ESP	TCP	Payload	Trailer	Auth

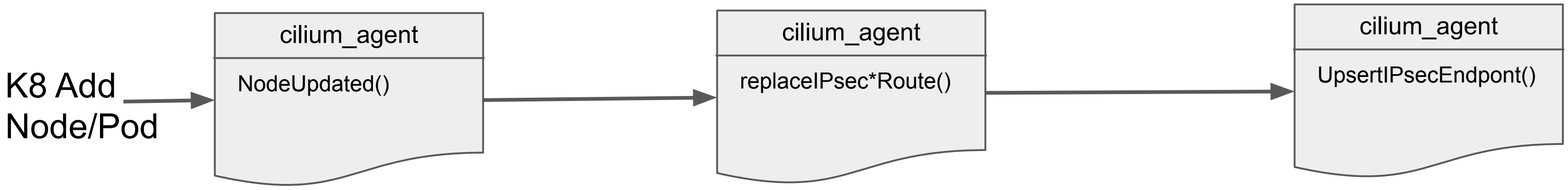
IPhdr	Ext Hdrs	ESP	IPhdr	Ext Hdrs	TCP	Payload	Trailer	Auth



L3 Encryption: Cilium Control Plane



L3 Encryption: Cilium Control Plane



Update Cilium IPCache

BPF map -- cilium_ipcache

```
IP                        ID    Key    NodeIP
------------------------  ----- -----  --------------
10.250.186.168/32         100   3      192.168.86.250
```

Update Route Table

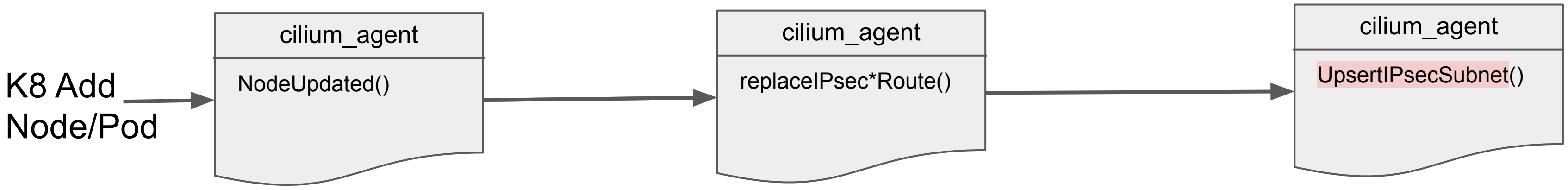
```
# L3 Encryption Routing Table 200
$ ip r s t 200
10.250.0.0/16 dev cilium_host mtu 1423
192.168.86.250 dev cilium_host
        proto cilium mtu 1423
local 192.168.86.26 dev eno1
        proto cilium

$ ip rule
1:    from all fwmark 0xd00/0xf00 lookup 200
1:    from all fwmark 0xe00/0xf00 lookup 200
```

Update Xfrm State/Policy

```
$ ip x p
src 10.26.0.0/16
dst 10.250.0.0/16
    dir out priority 0
    mark 0x3e00/0xff00
$ ip s p ...
```

L3 Encryption: Cilium Control Plane



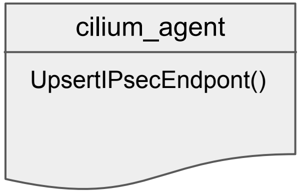
Update Cilium IPCache

BPF map -- cilium_ipcache

IP	ID	Key	NodeIP
10.250.186.168/32	100	3	192.168.86.250

- Subnet Based Encryption
  Requires "knowing" subnets a priori
  - Pros:
    - Reduces Xfrm rule space
  - Cons:
    - Requires subnet allocation upfront
    - Per node fidelity unless combined with per endpoint strategy.

L3 Encryption: Cilium Keys PreShared



PreShared Keys with Kubernetes secrets

- Generate secret key

```
kubectl create -n kube-system secret generic cilium-ipsec-keys \
--from-literal=keys="3 rfc4106(gcm(aes)) $(echo $(dd if=/dev/urandom count=20 bs=1 2> /dev/null |
        xxd -p -c 64)) 128"
```

- Mount secret in cilium-agent
- Cilium-agent will read key indexed by key

Key Rotations

- Update secret
- [Cilium 1.6 rolling restart agent notify hooks PR for 1.7]
- During roll-out may have multiple keys in-use

L3 Encryption: Cilium Keys SPIFFE

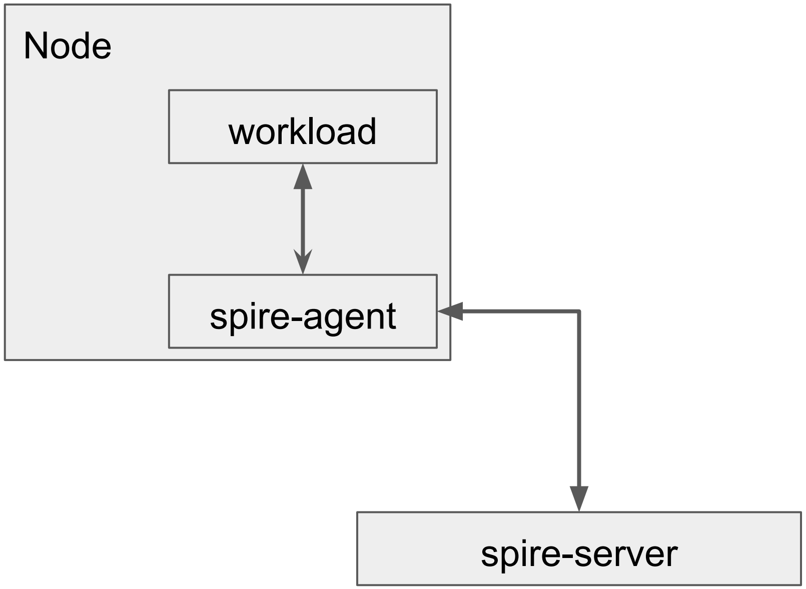
SPIFFE, the Secure Production Identity Framework For Everyone, provides a secure identity, in the form of a specially crafted X.509 certificate, to every workload in a modern production environment.

SPIRE: SPIFFE Runtime Environment

https://spiffe.io/

https://github.com/spiffe/spire

L3 Encryption: Cilium Keys SPIRE



Spire Server:

- Manages/Issues x509 identities and keys
- Node Attestations for agents
- Upstream CA integration

Spire Agent:

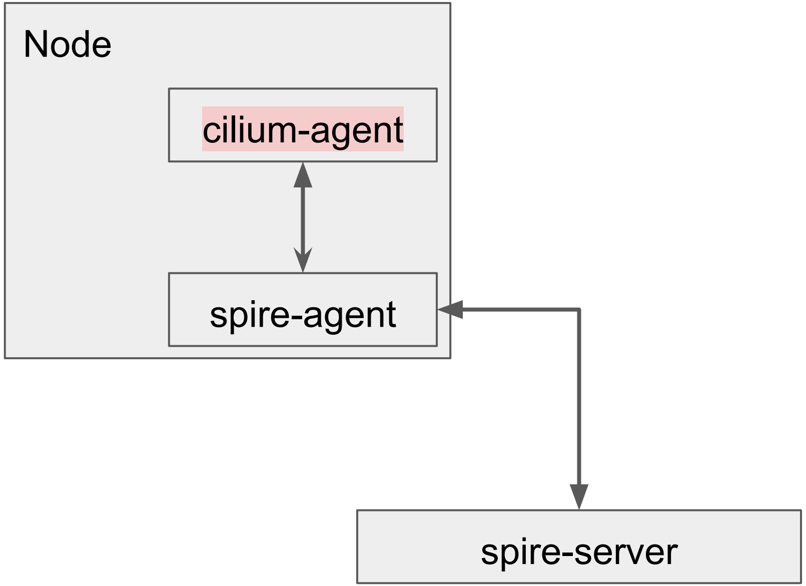
- Node Attestation
- Worker Attestation
- Fetch certificates and keys from server
- Expose workload API

Workload API:

- Fetch X509 Cert + Private Key
- Validate X509 Cert
- Watch Updates

https://spiffe.io/spire/overview/

L3 Encryption: Cilium Keys SPIRE



Cilium-agent:

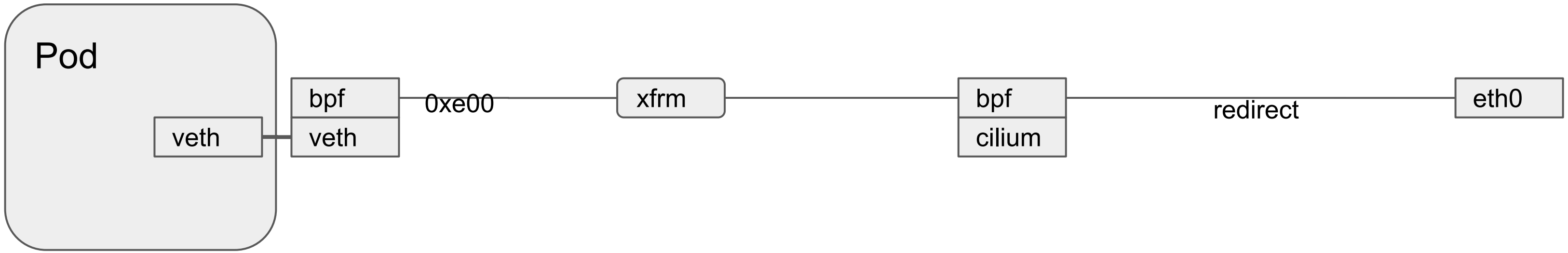
- Fetch X509 Cert + Private Key
- Generate shared Key
- Watch Updates

https://spiffe.io/spire/overview/

L3 Encryption: Cilium BPF Datapath

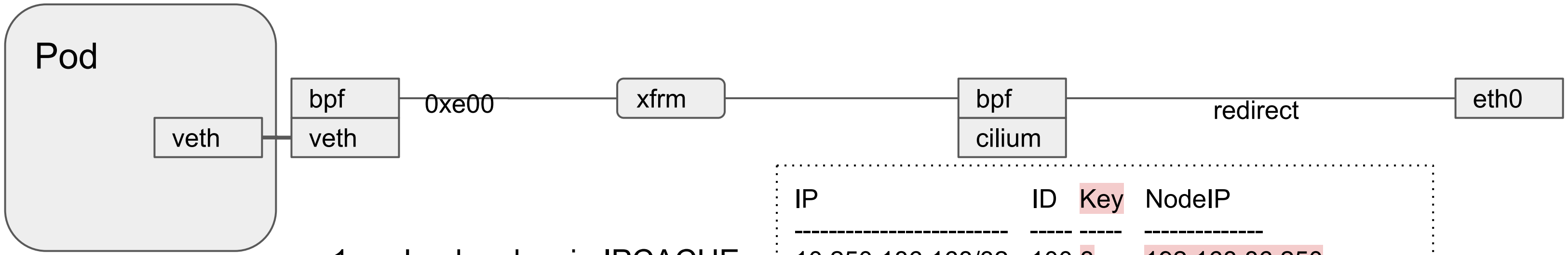
L3 Encryption: Cilium BPF Datapath

- ❑ Pod -> Pod
- ❑ Pod -> Node
- ❑ Pod -> Host Networking



L3 Encryption: Cilium BPF Datapath

- [ ] Pod -> Pod
- [ ] Pod -> Node
- [ ] Pod -> Host Networking



1. Lookup key in IPCACHE
2. skb->mark = key | 0xe00
   Encode IP metadata
3. TC_ACT_OK
4. xfrm encrypt

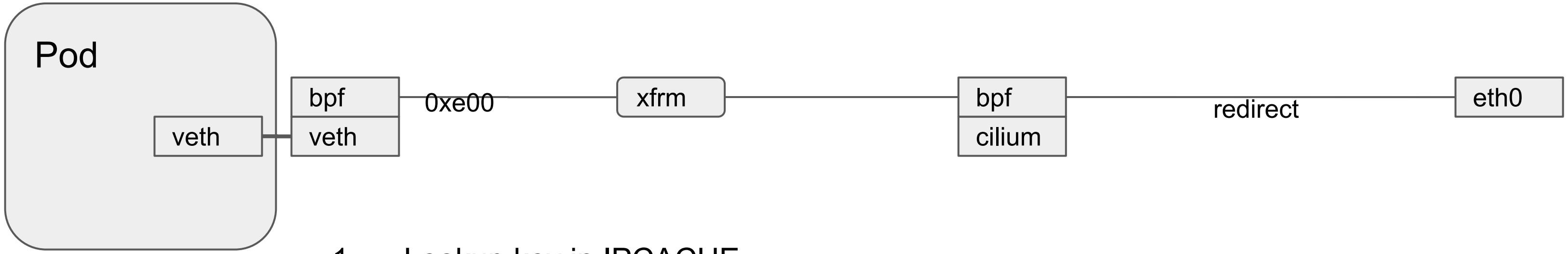
IP	ID	Key	NodeIP
10.250.186.168/32	100	3	192.168.86.250

```
$ ip x p
src 10.26.0.0/16
dst 10.250.0.0/16
    dir out priority 0
    mark 0x3e00/0xff00
```

```
$ ip r s
192.168.86.250 dev cilium_host  proto 192  mtu 1423
```

L3 Encryption: Cilium BPF Datapath

- ❑ Pod -> Pod
- ❑ Pod -> Node
- ❑ Pod -> Host Networking

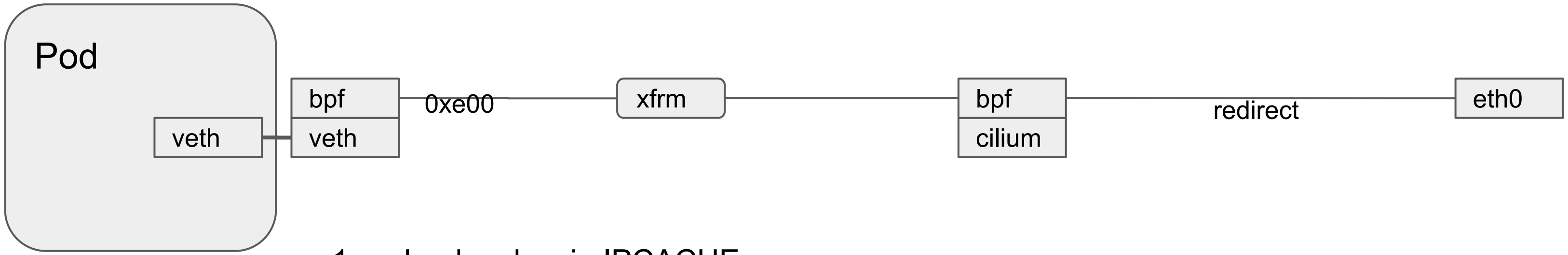


1. Lookup key in IPCACHE
2. skb->mark = key | 0xe00
   Encode IP metadata
3. TC_ACT_OK
4. xfrm encrypt

```
IF ESP THEN
   IF SUBNET_POOL THEN
       IP_REWRITE
   FIB_LOOKUP()
   MAC_REWRITE
   REDIRECT()
```

L3 Encryption: Cilium BPF Datapath

- ❏ Pod -> Pod
- ❏ Pod -> Node
- ❏ Pod -> Host Networking

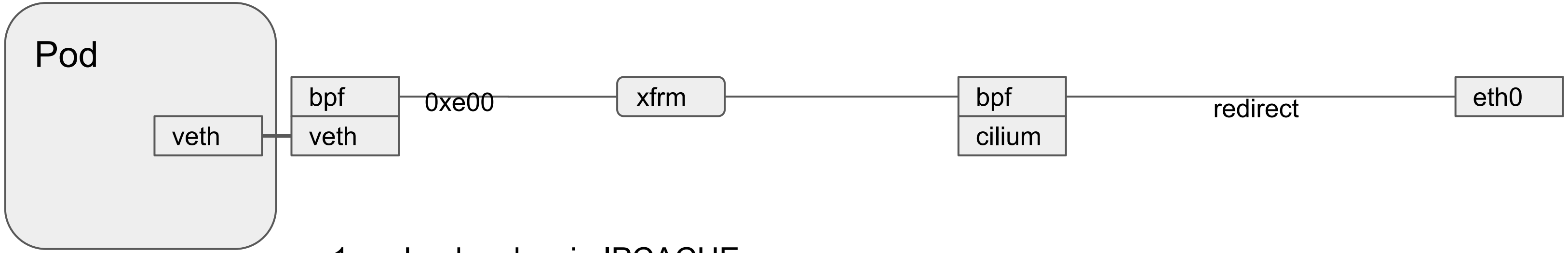


1. Lookup key in IPCACHE
2. skb->mark = key | 0xe00
   Encode IP metadata
3. TC_ACT_OK
4. xfrm encrypt

```
IF ESP THEN
  IF SUBNET_POOL THEN
     IP_REWRITE
  FIB_LOOKUP()
  MAC_REWRITE
  REDIRECT()
```

L3 Encryption: Cilium BPF Datapath

- ❏ Pod -> Pod
- ❏ Pod -> Node
- ❏ Pod -> Host Networking



1. Lookup key in IPCACHE

2. skb->mark = key | 0xe00
   Encode IP metadata

3. TC_ACT_OK

4. xfrm encrypt

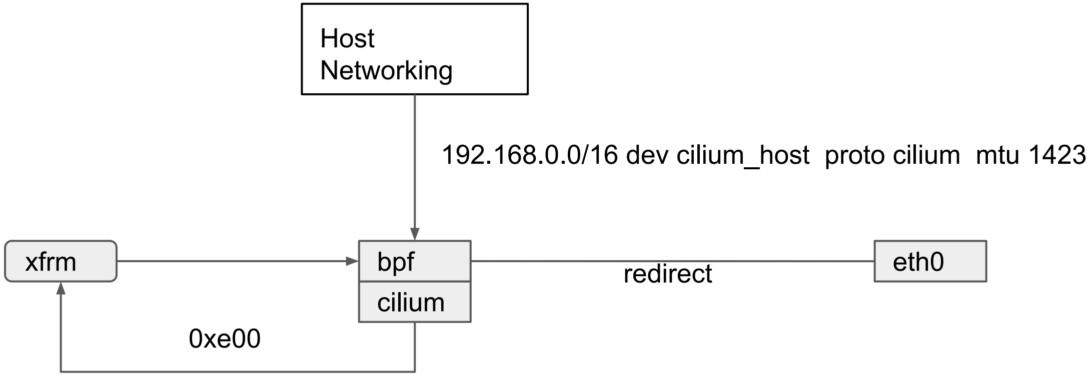
```
IF ESP THEN
  IF SUBNET_POOL THEN
      IP_REWRITE
  FIB_LOOKUP()
  MAC_REWRITE
  REDIRECT()
```

REDIRECT_TUNNEL()

Tunnel Case: VXLAN, Geneve

L3 Encryption: Cilium BPF Datapath

- ❏ Node -> Pod
- ❏ Node -> Node
- ❏ Node -> Host Networking



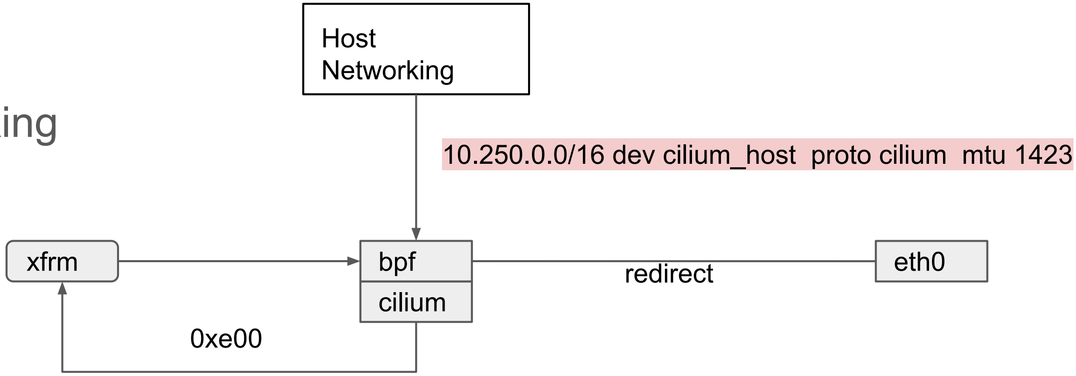
Pass #1
1. Lookup key in IPCACHE
2. skb->mark = key | 0xe00
   Encode IP metadata
3. TC_ACT_OK
4. xfrm encrypt

Pass #2

```
IF ESP THEN
  IF SUBNET_POOL THEN
      IP_REWRITE
  FIB_LOOKUP()
  MAC_REWRITE
  REDIRECT()
```

L3 Encryption: Cilium BPF Datapath

- ❑ Host Networking -> Pod
- ❑ Host Networking -> Node
- ❑ Host Networking -> Host Networking



<u>Pass #1</u>

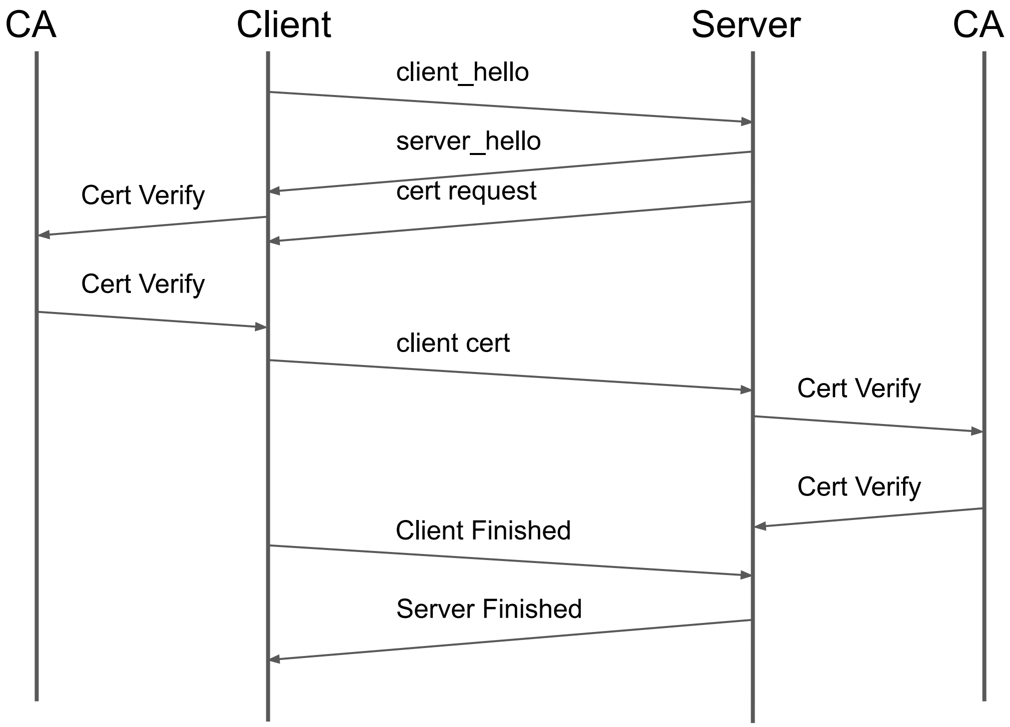
1. Lookup key in IPCACHE
2. skb->mark = key | 0xe00
   Encode IP metadata
3. TC_ACT_OK
4. xfrm encrypt

<u>Pass #2</u>

IF ESP THEN
  IF SUBNET_POOL THEN
    IP_REWRITE
  FIB_LOOKUP()
  MAC_REWRITE
  REDIRECT()

L7 Encryption:

L7 Encryption: Mutual TLS in 1 slide!



Client_hello: Initiates TLS
Cipher suites, keys, extensions, ...

Server hello: Response to hello
Cipher suites, keys, extensions, ..

Cert Request: The “m” server certificate request

Client Cert: Client providing certificate

Client Finished: Ready to send application data.

Server Finished: Ready to send application data.

https://tools.ietf.org/html/rfc8446

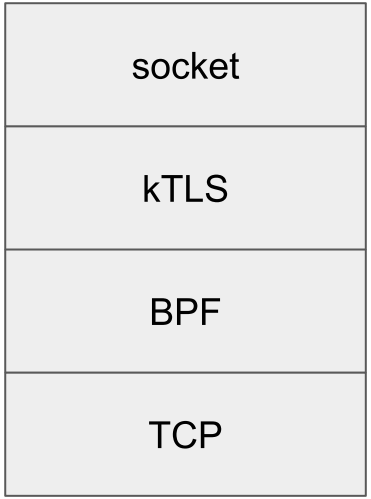
L7 Encryption: kTLS/Sockmap

Socket: OpenSSL kTLS enabled

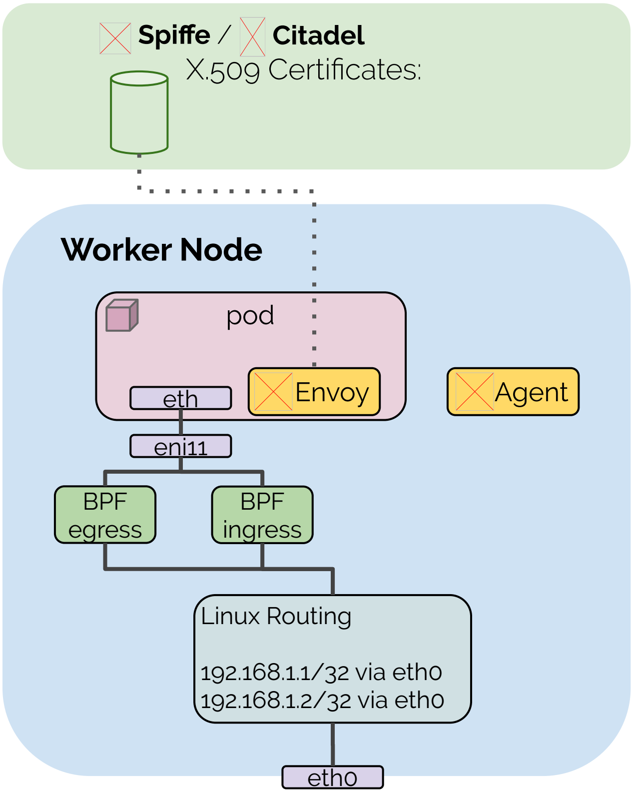
BPF: Sockmap BPF attached to socket enforces policy

kTLS: kernel implements TLS after initial handshake

TCP: Normal TCP stack

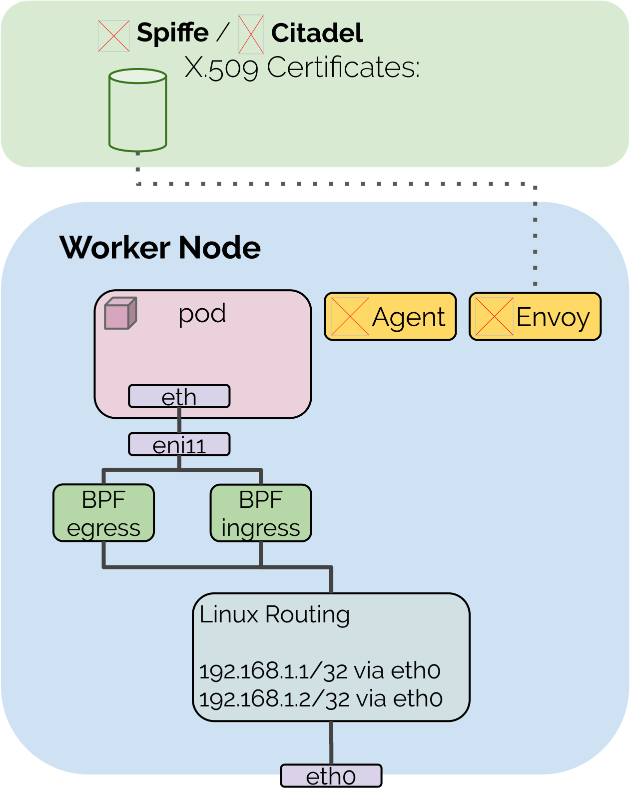


L7 Encryption: mTLS Istio/Envoy



- Spiffe Envoy Agent: Glues Envoy into Spiffe
  - Fetches initial certificates and keys
  - Watches for any updates
- Per HTTP request JWT tokens
  - Fetch per request
  - Validate on response
- Envoy sidecar per Pod
  - Scales with number of Pods
  - 100k pods vs 5k nodes in scaling tests
- 2x stack trips introducing latency

L7 Encryption: mTLS Cilium Istio/Envoy



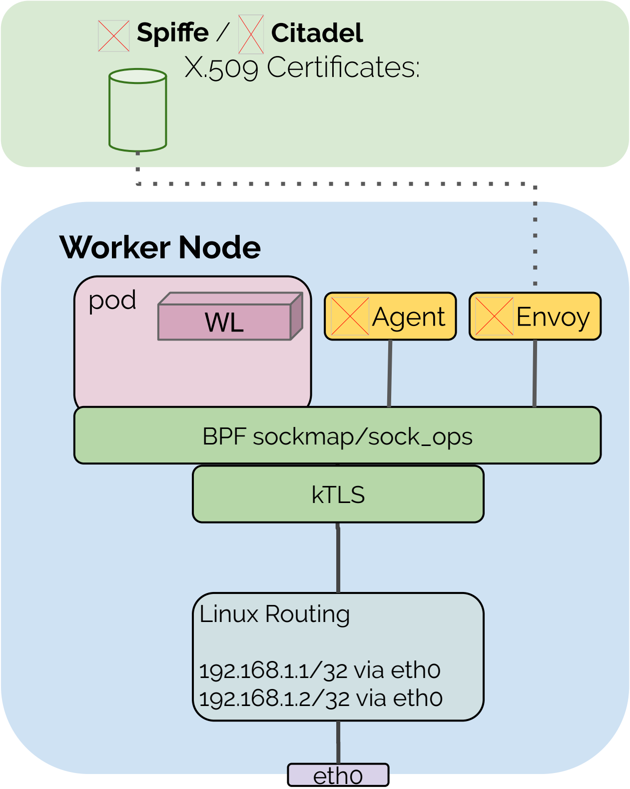
✓ Spiffe Envoy Agent: Glues Envoy into Spiffe
- Fetches initial certificates and keys
- Watches for any updates

✓ Per HTTP request JWT tokens
- Fetch per request
- Validate on response

○ Envoy sidecar per ~~Pod~~ Node
- Scales with number of ~~Pod~~ Node
- 100k pods vs 5k nodes in scaling tests

✓ 2x stack trips introducing latency

L7 Encryption: mTLS Cilium Istio/Envoy/Sockmap



- ✓ Spiffe Envoy Agent: Glues Envoy into Spiffe
  - Fetches initial certificates and keys
  - Watches for any updates
- ✓ Per HTTP request JWT tokens
  - Fetch per request
  - Validate on response
- ○ Envoy sidecar per ~~Pod~~ Node
  - Scales with number of ~~Pod~~ Node
  - 100k pods vs 5k nodes in scaling tests
- ~~○ 2x stack trips introducing latency~~
- ○ kTLS support for sendfile, etc.

Pain Points: L3

- L3 Traversal to use XFRM stack
  - Complexity stack may drop, route rules hit
  - Performance extra pass through L3, L2, tc, veth, etc.
- IPsec fields limiting
  - Only IP, no wildcard destination IP
  - For workload encryption arbitrary field matching (BPF) is useful

- **Solution:** BPF encryption engine
  - BPF Encryption Map: (Arbitrary key) -> (encryption state)
  - Hash table O(1) lookup scales
  - Performance better but still cost of encryption
- **Offload:** Encryption Offload dev bindings
  - BPF “knows” outgoing interface despite possible extra programs and/or hops enroute
  - BPF Encryption Map: (Encryption state) include dev binding

Pain Points: L7 kTLS/Sockmap

- kTLS/Sockmap missing pieces:
  - Receive hook missing: Allow for redirect on receive after encryption
  - PerfRing support for streaming events to userspace
- kTLS offload breaks BPF policy
  - TBD, add BPF hooks to device offload paths
- OpenSSL distributed with kTLS enabled
- Other SSL library support, BoringSSL

Solutions: Couple feature additions and **distributions** start to push kTLS enabled SSL libs when available.

comment: Any BoringSSL or other library developers want to help? Come find me afterwards.

Pain Points: Key Management

- Service Keys
  - Complexity increases if keys are managed manually via secrets. Works best if existing infrastructure in place.
  - SPIRE workload API “fetch” identifies agent “workload” with a X509 and key.

- Key Rotation and Management
  - Automate key creation, issue open on github.com/cilium/cilium
  - Key rotation currently automatic if user kicks it by supplying new key

BPF

- ✓ Instruction Limits
- ✓ Loops
- ✓ BTF
- ✓ Socket Lookup
- ✓ Socket Memory

More work still but moving quickly!!!

Thank you!

More Information:

Slack: https://cilium.io/slack

GitHub: https://github.com/cilium/cilium

Docs: https://docs.cilium.io/

Twitter: @ciliumproject