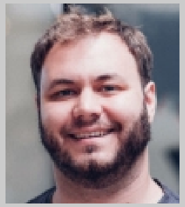
Improving Secure Pod-to-Pod Communication Using Trust Bundles


Ted Hahn, TCB Technologies, Inc.

Mark Hahn, Qualys.

tcbtech.com/cncf-kubetls

Achieving Mutual TLS - Secure Pod-to-Pod communication

"Every Kubernetes pod should include a SSL Certificate, verifying its identity."

This should be signed automatically, and be specific to each pod.

We have updated KubeTLS to modularize the certificate creation and improve the flow.

tcbtech.com/cncf-kubetls

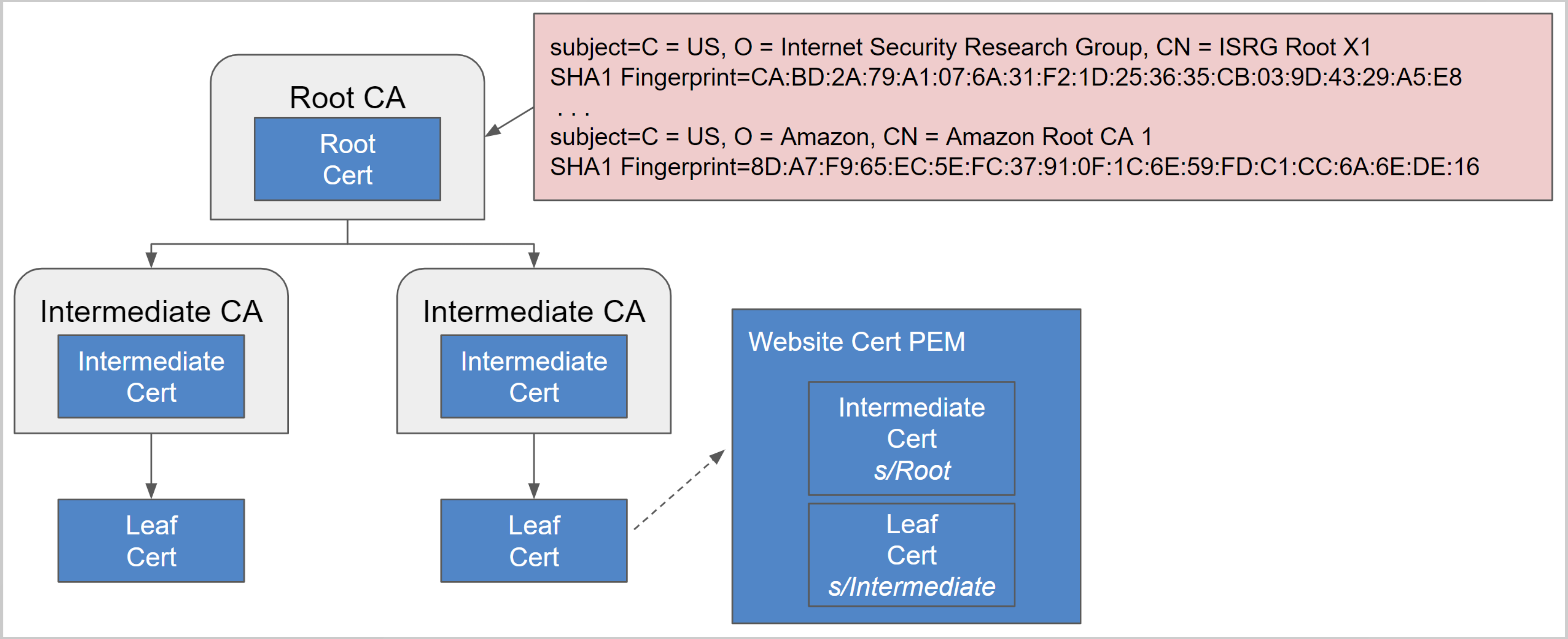
What is a TrustBundle?

Note: This is a forward looking feature, KEP-3257, that has yet to be implemented. We want to show some of the possible usecases.

A Trust Anchor, or "Root of trust", is a cryptographic entity that you trust implicity. Typically you express this as an X.509 certificate with the CA bit set. A Trust Bundle one or more trust anchors combined together, with the same implicit trust.

You already use a Trust Bundle - The Web Roots provided by your Base OS or by your Web Browser.

Brief detour - Certificate Authorities

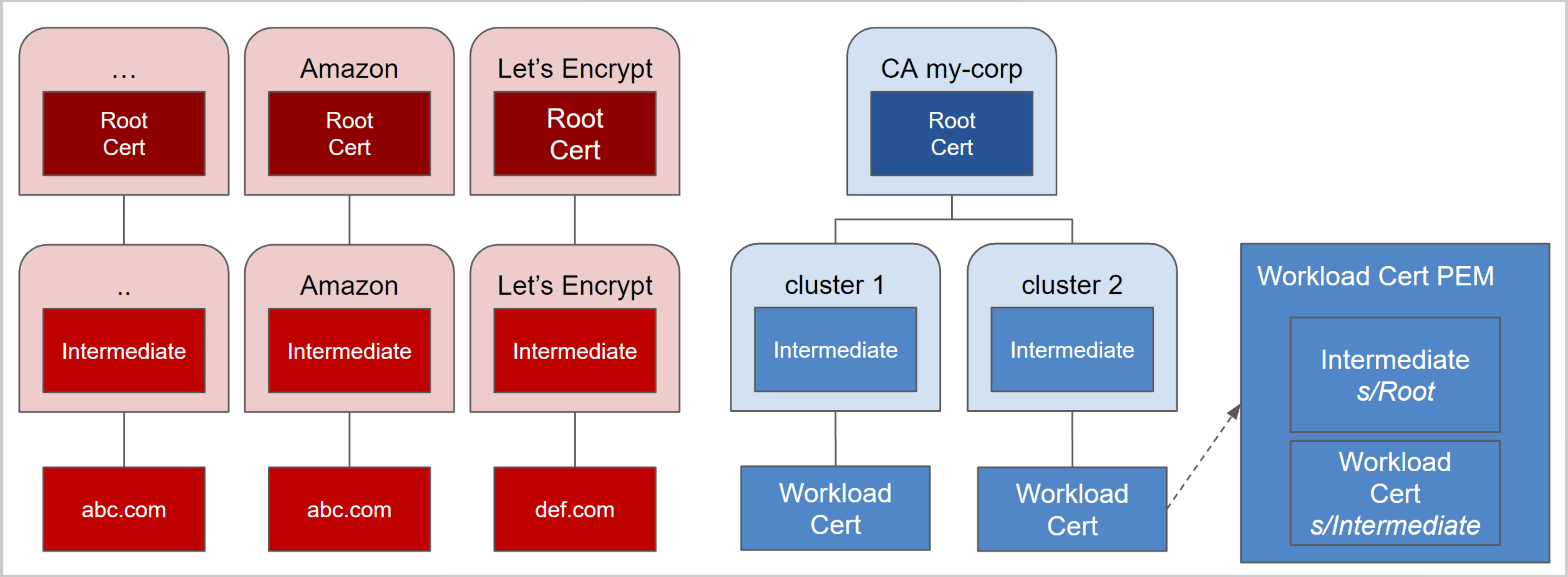


```
subject=C = US, O = Internet Security Research Group, CN = ISRG Root X1
SHA1 Fingerprint=CA:BD:2A:79:A1:07:6A:31:F2:1D:25:36:35:CB:03:9D:43:29:A5:E8
 . . .
subject=C = US, O = Amazon, CN = Amazon Root CA 1
SHA1 Fingerprint=8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
```

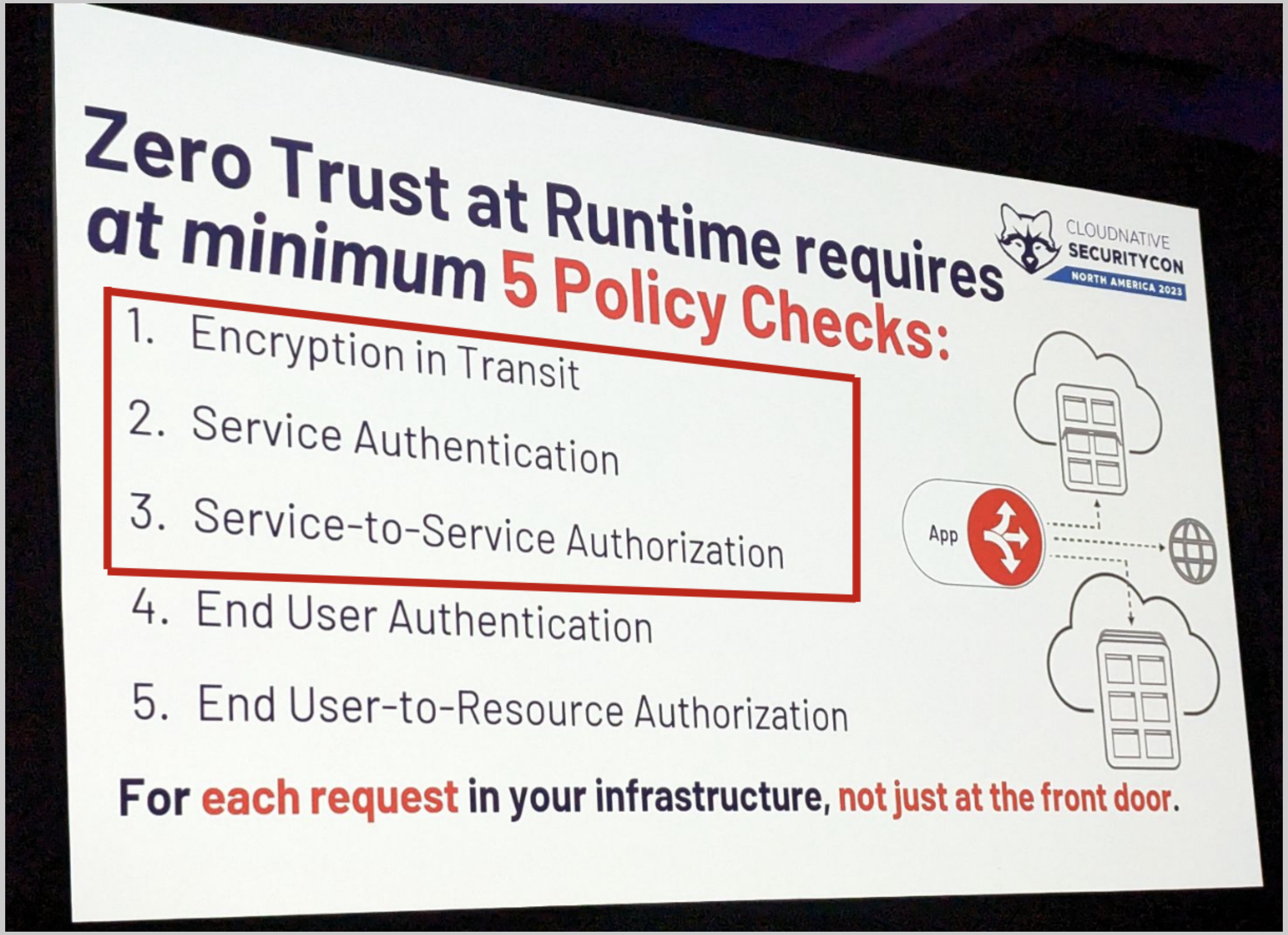
How to use TrustBundles?

- TrustBundles implment small scale scope of trust
- "The Web Trust Bundle" is one of the topmost layers of most docker images
- Docker Images should be small
- ClusterTrustBundles can be mounted like ConfigMaps, replacing "The Web Trust Bundle"
- Rapid Updates to TrustBundles

Limiting Trust Scope



Limiting Trust from KeyNote


CLOUDNATIVE
SECURITYCON
NORTH AMERICA 2023
App

Zero Trust at Runtime requires at minimum 5 Policy Checks:

1. Encryption in Transit
2. Service Authentication
3. Service-to-Service Authorization
4. End User Authentication
5. End User-to-Resource Authorization

For each request in your infrastructure, not just at the front door.

What is KubeTLS

KubeTLS is about automatically injecting certificates that provide workload identity into every pod and every containter in a cluster.

These Certificates provide Privacy, Authentication and Authorization.

These Certificates work with TrustBundles to assist in mutual system identification.

Secure Networking on Kubernetes

- The complicated way: Sidecars or CNI with network policies
  - Sidecars add latency, and a management layer separate from the applications
  - Network policies add that management layer separate from the applications
- The native way: All application containers natively support TLS
  - Applications are in charge of security
  - Applications implement business specific security logic

KubeTLS assists with implementing TLS Everywhere

- Using TLS natively everywhere eliminates various attack vectors by encrypting all traffic inside the applcation:
  - Privacy is provided by TLS
  - Authentication is provided by mTLS using the TrustBundle.
  - Authorization is provided by inspection of the client provided certificate.

Unpacking what is KubeTLS

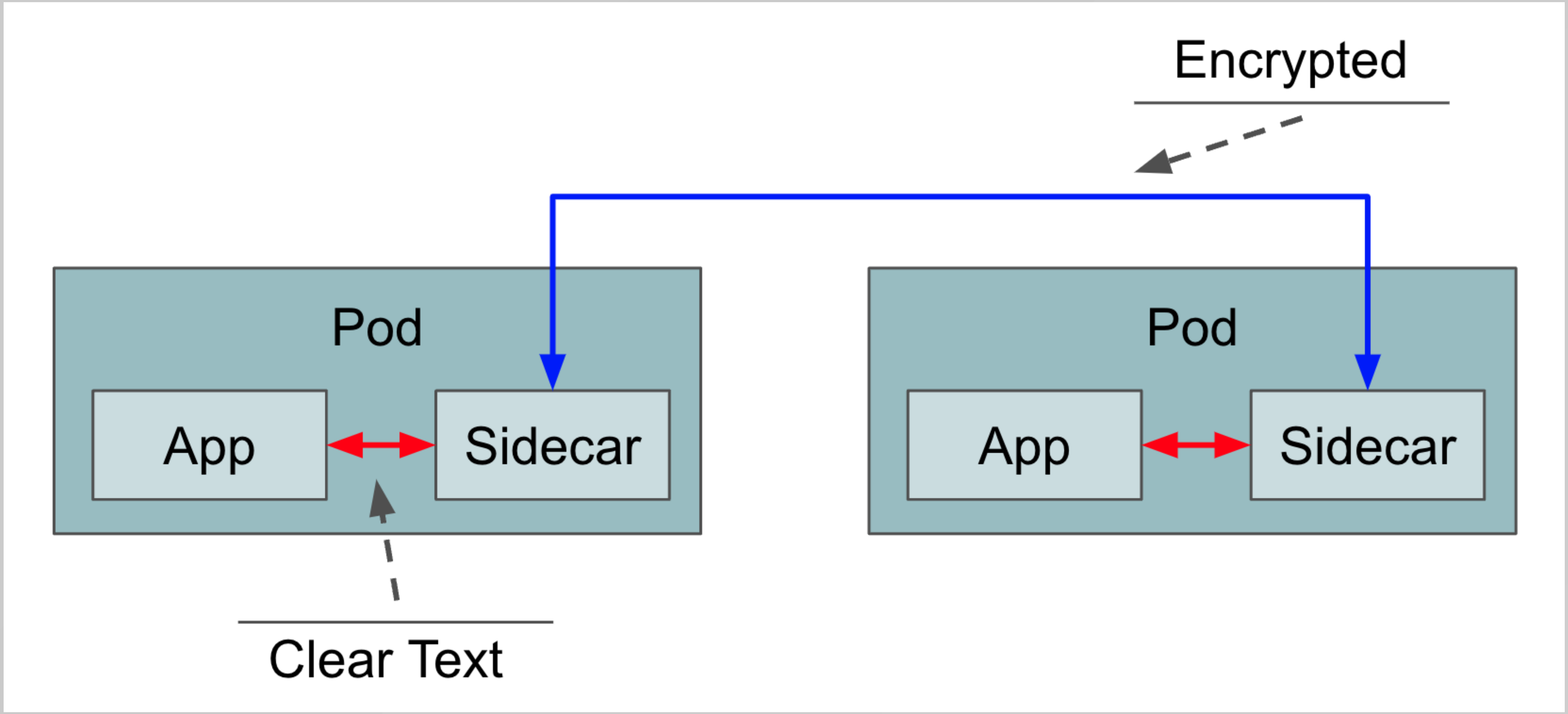
- KubeTLS is an admission controller using `MutatingWebhookConfiguration` object.
- KubeTLS provides three files by modifying the pod on admission -
  - A Private Key
  - A Matching Certificate, signed by our trust bundle
  - The trust bundle itself (since KEP-3257 is TBD)
- These are the building blocks of a TLS native architecture.

Our opinions

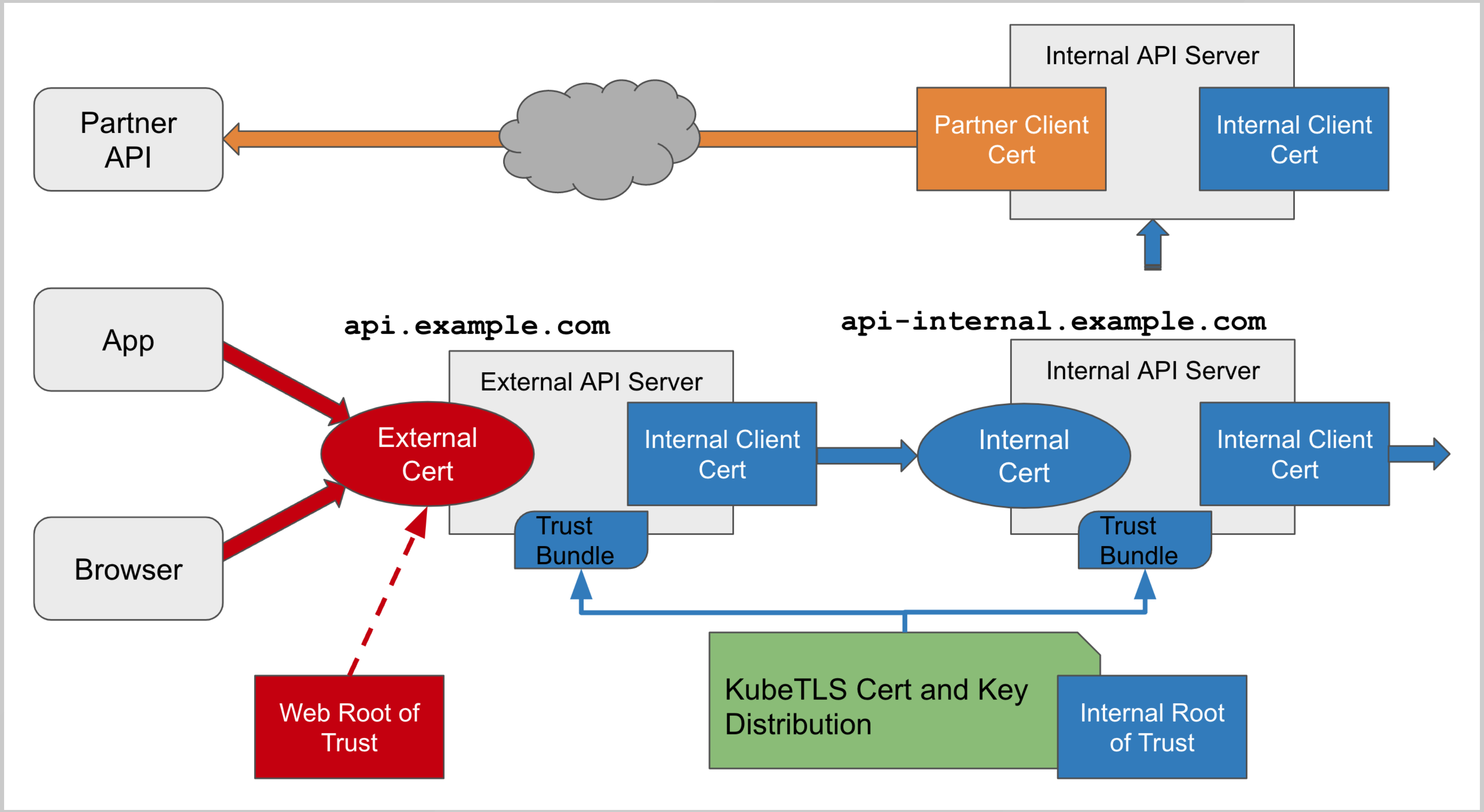
We have strong opinions on TLS

- Your applications should speak only TLS
- Your internal applications should not use the web root of trust
- Your need to design your organization's zones of trust
- Your internal applications should use an internal CA
- Your applications should present a client certificate
- You should validate the client certificate as caller identity

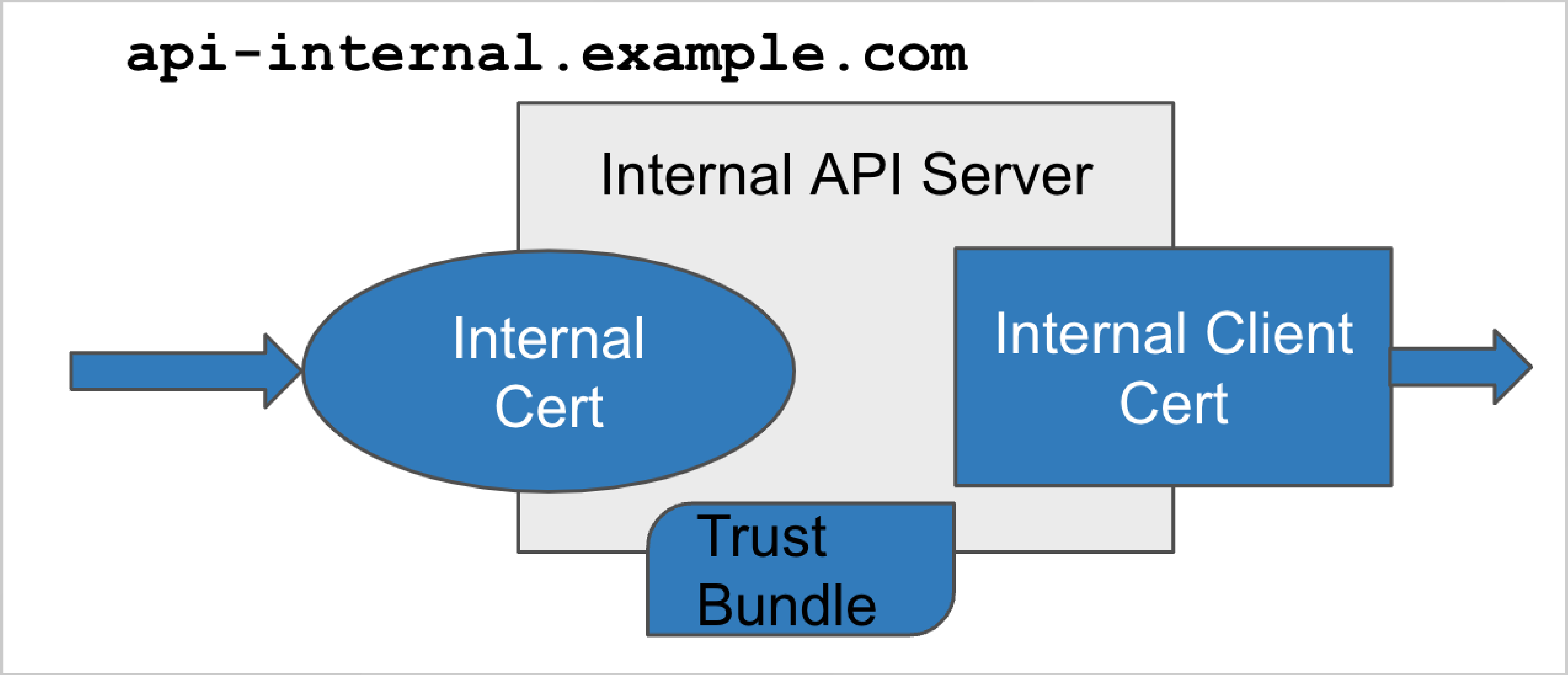
What not to do



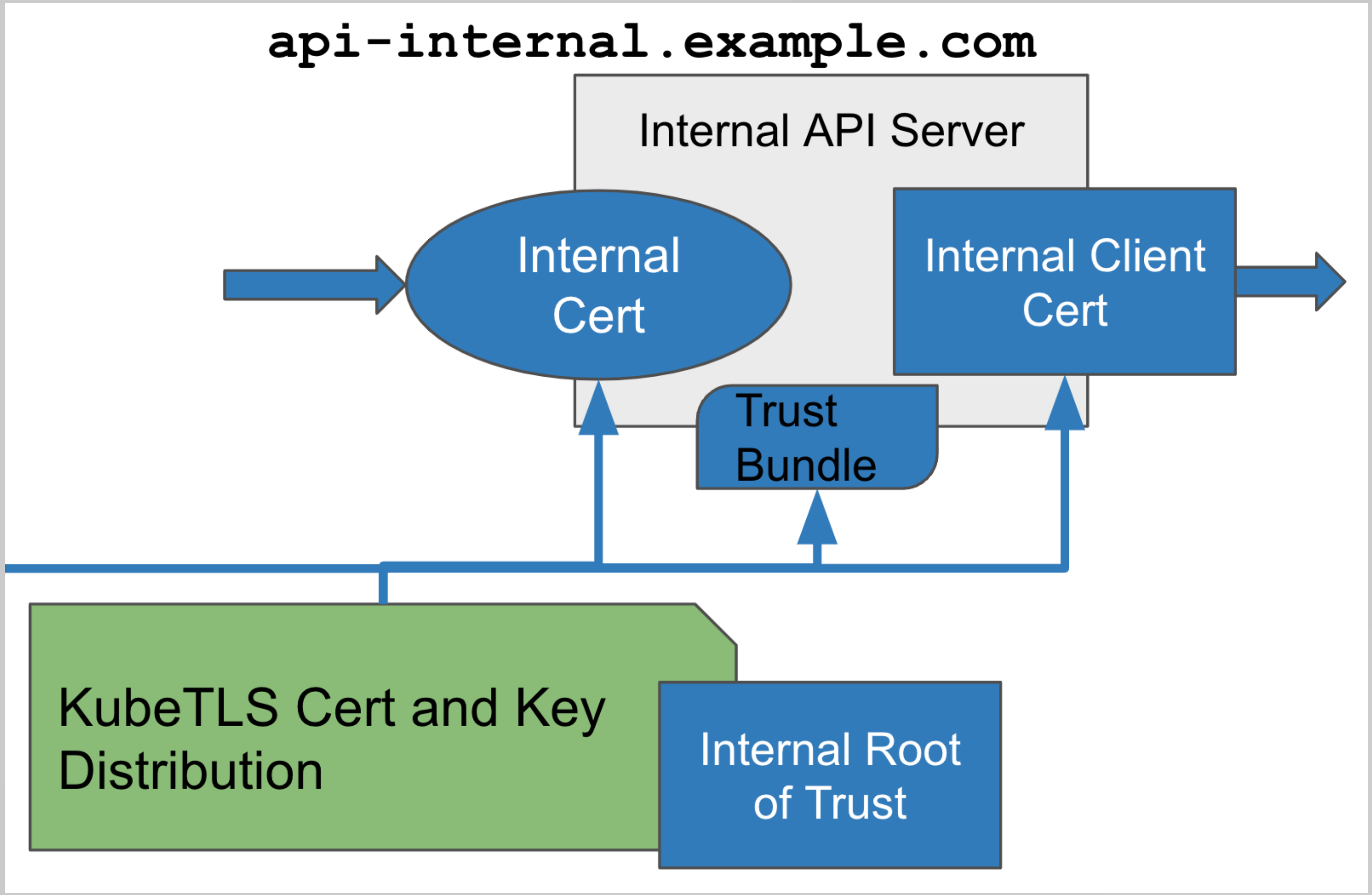
Our Opinion


Internal API Server
Partner API
Partner Client Cert
Internal Client Cert
App
api.example.com
api-internal.example.com
External API Server
Internal API Server
External Cert
Internal Client Cert
Internal Cert
Internal Client Cert
Browser
Trust Bundle
Trust Bundle
Web Root of Trust
KubeTLS Cert and Key Distribution
Internal Root of Trust

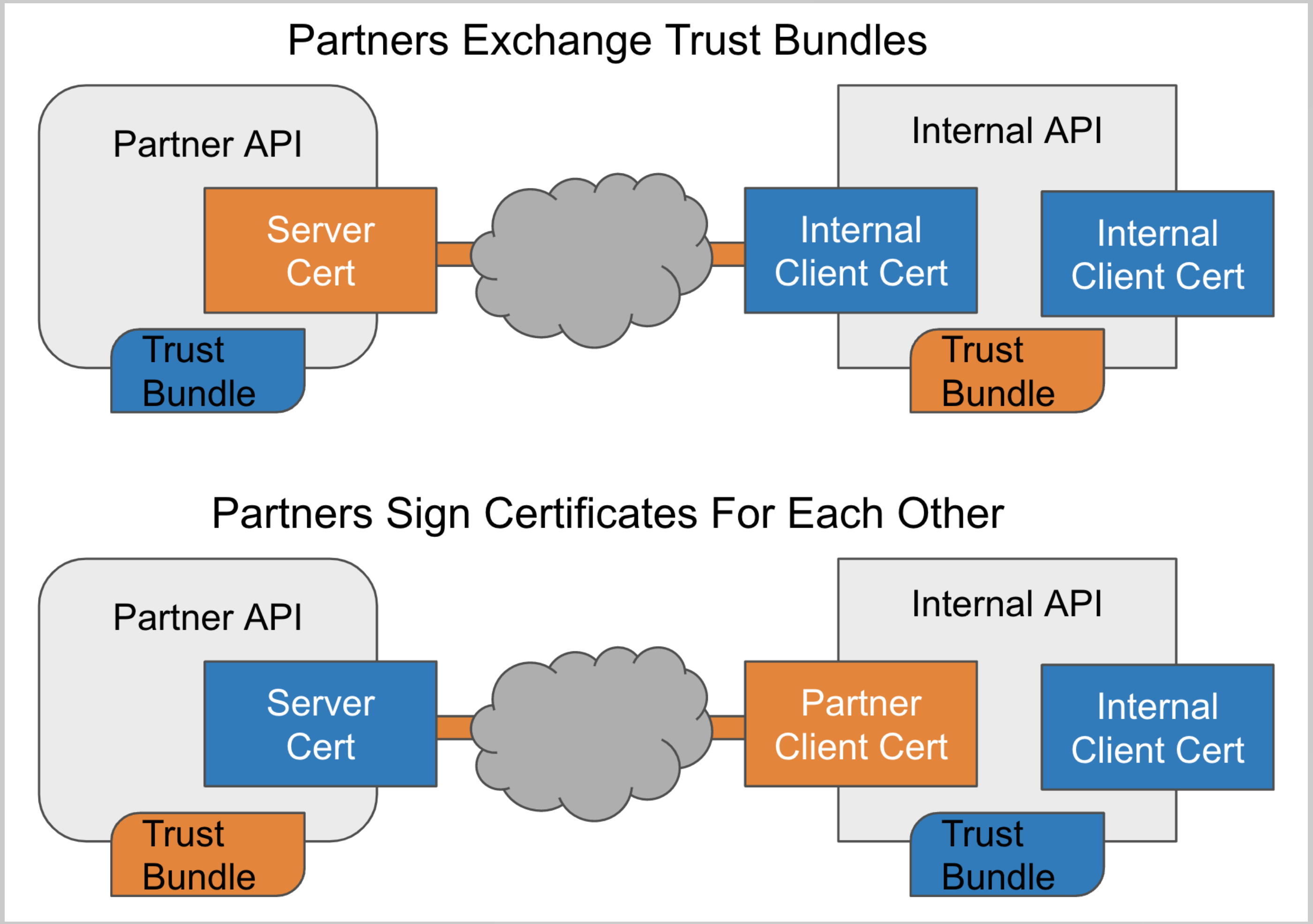
Internal API Server



KubeTLS Certificate Distribution



Our Opinion on Partner Trust



KubeTLS Certificate Creation Details

- When a pod is created KubeTLS's admission controller is called
- KubeTLS looks up container's service (via metadata)
- KubeTLS will create and approve a certificate signing request (csr)
- The certificate manager will do magic to create a cert
- KubeTLS creates a secret with:
    - private key, the service cert, the root CA cert
- KubeTLS attaches the secret to all containers in the pod
- KubeTLS responds to the pod adminssion web call

KubeTLS X.509 Key Fields

- Common Name
  - Name of the pod
- Subject Alternative Name (SAN)
  - DNS with the name of the services
  - URI with the SPIFFE SVID
- Key usage and extended key usage
  - Identified as Web server and client
- Issuer (aka signer) identification

Demo Time

Demo time, open the command line
And let you out into the shell
Demo time, turn all of the servers on
Over every pod and every service

Demo time, one last call for changes
So, finish your commit or pr
Demo time, you don't have to log out
But you can't stay here
. . .
(apologies to Semisonic)

Wrap up

- Having certificates automatically populated is easy
- Developers should know/learn how to use mTLS
- KubeTLS is "identity policy" agnostic
- Once eveybody is establishing authentication through mTLS we can move to establishing authorization through mTLS

Future Directions

- Private keys should be generated on nodes
- KubeTLS will move to a CSI model
- This should be built into Kubernetes

Repository

https://gitlab.com/gauntletwizard_net/kubetls

Presentation Slides

tcbtech.com/cncf-kubetls

Session Feedback link:

Additional Notes

The following slides and notes are useful.

- Auto mount service account tokens
  - https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/

KubeTLS Admission Webhook

```yaml
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- admissionReviewVersions:
  - v1beta1
  clientConfig:
    caBundle: <<X.509 Cert (base64 encoded)>>
    service:
      name: kubetls
      namespace: kubetls
  name: kubetls.gauntletwizard.net
  rules:
  - apiGroups:
    - ""
    apiVersions:
    - v1
    operations:
    - CREATE
    resources:
    - pods
...
```