Kubelet to Istio: Kubernetes Network Security Demystified

FOSDEM SPEED RUN

@sublimino and @controlplaneio
I’m:
- Andy
- Dev-like
- Sec-ish
- Ops-y
What is Network Security
Why do we need Network Security?

“Why everybody trying to break internet?”

@sublimino
Happy Path Application Design
How Applications Run in “Piratical Reality™”
How Kubernetes does it
Self Signed Certs. Always a bad thing?
Takeaway: Encrypt Everything Everywhere
What this talk is about

● Network Security 101
● Kubernetes API Components
● TLS, X.509, and Mutual Authentication
● CNI and Network Policies for Applications
● Bootstrapping Identity with SPIFFE
Network Security 101
Private & Trusted Communications

RISE OF THE HACKERS

Source: Carna Botnet
Human Communication: Trusted and Local

@sublimino
Human Communication: Untrusted and Local
Human Communication: Untrusted and Remote
In Internet Prehistory...
SSL/TLS – A brief history lesson

Secure Socket Layer (SSL) – Originated in Netscape Web browser

- SSL 1.0 (Never released)
- SSL 2.0 (1995) - Security Flaws
- SSL 3.0 (1996) - Obsolete

Transport Layer Security (TLS) – Standardized by IETF

- TLS 1.0 (1999) - Slight upgrade of SSL 3.0
- TLS 1.1 (2006)
- TLS 1.2 (2008)
- TLS 1.3

http://slideplayer.com/slide/11456904/
Is it really that simple?
Securing API Server Traffic
Securing API Server Traffic

kube-apiserver
...
--client-ca-file=/secret/authca.pem
...
--etcd-cafile=/secret/ca.pem
--etcd-certfile=/secret/cert.pem
--etcd-keyfile=/secret/key.pem
--experimental-encryption-provider-config=/secret/encryption.cfg
...
--kubelet-certificate-authority=/secret/ca.pem
--kubelet-client-certificate=/secret/cert.pem
--kubelet-client-key=/secret/key.pem
...
--oidc-ca-file=/secret/ca.pem
...
--service-account-key-file=/secret/service_account_key.pem
...
--tls-ca-file=/secret/ca.pem
--tls-cert-file=/secret/cert.pem
--tls-private-key-file=/secret/key.pem
--tls-sni-cert-key=/secret/cert.pem,/secret/key.pem:localhost
--tls-sni-cert-key=/secret/controller/cert.pem,/secret/controller/key.pem
...

@sublimino
Securing API Server Traffic
Securing API Server Traffic

Certificate Authority
Securing API Server Traffic

[Diagram showing the process of certificate issuance and verification, including Certificate Authority and Intermediate Certificate.]
Securing API Server Traffic

Certificate Authority

Leaf Certificate

Intermediate Certificate
TLS, X.509, and mutual authentication
Public Key Cryptography
Public Key Cryptography
Public Key Cryptography
Public Key Cryptography
Woah there, how does it work?
Woah there, how does it work?

- Symmetric encryption
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: make brute forcing more difficult using the properties of certain graphed shapes
Elliptic Curves

1. $P + Q + R = 0$
2. $P + Q + Q = 0$
3. $P + Q + 0 = 0$
4. $P + P + 0 = 0$
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: make brute forcing more difficult using the properties of certain graphed shapes
- Diffie-Hellman
Woah there, how does it work?

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: make brute forcing more difficult using the properties of certain graphed shapes
- Diffie-Hellman: a way to create a shared encryption key without ever communicating it publicly
Cryptography!

- Symmetric encryption: identical keys to lock and unlock
- Asymmetric encryption: different keys to lock and unlock
- Elliptic-curve cryptography: make brute forcing more difficult using the properties of certain graphed shapes
- Diffie-Hellman: a way to create a shared encryption key without ever communicating it publically
More Info


- https://howhttps.works/

by DNSimple
The Original Crypto!
The Original Crypto!

BLOCK CHAIN
TLS in Kubernetes

@sublimino
Certificate Path Validation

Certificate Authority

Leaf Certificate

Intermediate Certificate
X.509

Public key

Common name and organization

DNS Name

Dates valid

SSL Certificate

Issuer name and organization

Issuer signature

https://blog.cloudflare.com/content/images/2018/03/image8.png
X.509 RFC Format

Certificate ::= SEQUENCE {
  tbsCertificate TBSCertificate,
  signatureAlgorithm AlgorithmIdentifier,
  signatureValue BIT STRING }

TBSCertificate ::= SEQUENCE {
  version [0] EXPLICIT Version DEFAULT v1,
  serialNumber CertificateSerialNumber,
  signature AlgorithmIdentifier,
  issuer Name,
  validity Validity,
  subject Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
  subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
  extensions [3] EXPLICIT Extensions OPTIONAL,
    -- If present, version MUST be v3
}

Version ::= INTEGER { v1(0), v2(1), v3(2) }

CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
  notBefore Time,
  notAfter Time }

Time ::= CHOICE {
  utcTime UTCTime,
  generalTime GeneralizedTime }

UniqueIdentifier ::= BIT STRING

SubjectPublicKeyInfo ::= SEQUENCE {
  algorithm AlgorithmIdentifier,
  subjectPublicKey BIT STRING }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {
  extnID OBJECT IDENTIFIER,
  critical BOOLEAN DEFAULT FALSE,
  extnValue OCTET STRING
    -- contains the DER encoding of an ASN.1 value
    -- corresponding to the extension type identified
    -- by extnID
}

-----BEGIN CERTIFICATE-----
MIIC2jCCAkMCAg38MA0GCSqGSIb3DQEBBQUAMIGbMQswCQYDVQQGEwJKUDEOMAwGA1UE
CBMFVG9reW8xEDAOBgNVBAcTB0NodW8ta3UxETAPBgNVBAoTCEZyYW5rNEREMRgow
MRA0GCSqGSIb3DQEJARYUc3VwcG9ydEBmcnFuazuRkZC5jb20wHhcNMTIwODIyNzgw
NMTcwODIyMDUyNzQxWjBKMQswCQYDVQQGEwJKUDEOMAwGA1UECAwFVG9reW8xETAP
BgNVBAoMCEZyYW5rNEREMRgowMRA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKA0IBAQ
bMDIZaarys1a0YmUeVLCIqvzBkPJTSQsCopQQ9V8WuT252zzNzs68dVGh0dJK5J
NRQykpwexmnjPP0mvj7i8XgG379Tyl6Pc+Lw50eukjXj9ejS2ouDYdR2SM9BoVW
+FgxDu6BMzXhozw5EFnajFp7HL8kQC1I0Q0c79yuK13492rH6bzFsn2lfwW9yc7
7cP8ePcTeFptlFad+vxBhPzKeTQLHxK56h5Q5zeHIB5ySJJ7af2W8r4eTGYzbdRW2
4DDHCPhZAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAQl3s+21V6b5w40m+5vQFz7
8T9KynnmbJpTSi0+BM=
-----END CERTIFICATE-----
How to decode an X.509 Cert

$ openssl s_client -connect wikipedia.org:443
CONNECTED(00000003)
depth=2 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert High Assurance EV Root CA
verify return:1
depth=1 C = US, O = DigiCert Inc, OU = www.digicert.com, CN = DigiCert SHA2 High Assurance Server CA
verify return:1
depth=0 C = US, ST = California, L = San Francisco, O = "Wikimedia Foundation, Inc.", CN = *.wikipedia.org
verify return:1
---
Certificate chain
  0 s:/C=US/ST=California/L=San Francisco/O=Wikimedia Foundation, Inc./CN=*.wikipedia.org
  i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
  1 s:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert SHA2 High Assurance Server CA
  i:/C=US/O=DigiCert Inc/OU=www.digicert.com/CN=DigiCert High Assurance EV Root CA
---
Server certificate
-----BEGIN CERTIFICATE-----
MIIEfDCCB2SgAwIBAgIQCDCUYtH+pgrgur/174vFRTANBgkhk1G9w0BAQsFADBW
MQswCQYDVQQGEwJVUzEVMBMGA1UEChMMRGlnaUNlcnQgSW5jMRkwFwYDVQQLExB3
d3cuZGl1aWNIcnY29tMS8wLQYDVQQDEyZEawdpO29vdCBTSEEyIEhpZ2ggQXNz
dXJhbmNlIFNlcnVuBDTAEfW0xNSYmMjAwMDAwMDBaFw0xOTAxMjQxMjAwMDBa
...
X.509 Example Decoded Cert

Certificate:

Data:

   Version: 3 (0x2)
   Signature Algorithm: sha256WithRSAEncryption
   Issuer: C=BE, O=GlobalSign nv-sa, CN=GlobalSign Organization Validation CA - SHA256 - G2
   Validity
      Not Before: Nov 21 08:00:00 2016 GMT
      Not After : Nov 22 07:59:59 2017 GMT
   Subject: C=US, ST=California, L=San Francisco, O=Wikimedia Foundation, Inc., CN=*.wikipedia.org
   Subject Public Key Info:
      Public Key Algorithm: id-ecPublicKey
      pub:
         9d:3b:ef:d5:c1
      ASN1 OID: prime256v1
      NIST CURVE: P-256
      X509v3 extensions:
         X509v3 Key Usage: critical
         Digital Signature, Key Agreement
         Authority Information Access:
            CA Issuers - URI:http://secure.globalsign.com/cacert/gsorganizationvalsha2g2r1.crt
            OCSP - URI:http://ocsp2.globalsign.com/gsorganizationvalsha2g2

X509v3 Certificate Policies:

   Policy: 1.3.6.1.4.1.4146.1.20
   CPS: https://www.globalsign.com/repository/
   Policy: 2.23.140.1.2.2

X509v3 Basic Constraints:
   CA:FALSE
X509v3 CRL Distribution Points:
   Full Name:
      URI:http://crl.globalsign.com/gsorganizationvalsha2g2.crl

X509v3 Subject Alternative Name:
   DNS:*.wikipedia.org, DNS:*.m.mediawiki.org, DNS:*.m.wikibooks.org, ...

X509v3 Extended Key Usage:
   TLS Web Server Authentication, TLS Web Client Authentication

X509v3 Subject Key Identifier:

Signature Algorithm: sha256WithRSAEncryption
   ...

Self Signed Certs aka Signing Your Own Homework
One-Way (Traditional) TLS Handshake

SSL authentication handshake messages
Mutual TLS Handshake (mTLS)
Private & Trusted Communications
Securing API Server Traffic
Don't we trust our networks and firewalls?
BeyondCorp
Zero Trust Networking
Zero Trust API Server?
What could possibly go wrong?
GAME OVER
What could possibly go wrong?
Game Over?
GAME OVER

Would you like to continue?
Continuous (Kubernetes) Security

Slides / @sublimino

@sublimino
Application Layer
Containers and Traditional Network Security?
https://medium.com/google-cloud/understanding-kubernetes-networking-services-f0cb48e4cc82
Kubernetes NetworkPolicy: default deny

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
podSelector:
```

https://github.com/ahmetb/kubernetes-network-policy-recipes
Kubernetes NetworkPolicy: default deny

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
spec:
  podSelector:
    - "*"
```

Illegal syntax, but represents what it actually does (effectively a wildcard)

https://github.com/ahmetb/kubernetes-network-policy-recipes
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: foo-deny-external-egress
spec:
  podSelector:
    matchLabels:
      app: foo
  policyTypes:
  - Egress
  egress:
  - ports:
    - port: 53
      protocol: UDP
    - port: 53
      protocol: TCP
  - to:
    - namespaceSelector: {}/

https://github.com/ahmetb/kubernetes-network-policy-recipes
thockin (Tim Hockin) 27 days ago

I really don't think we want to impose DNS refreshing on implementations of NetworkPolicy without a bunch of REALLY REALLY good use cases that just CAN NOT be solved any other way. Do we have such use cases?

thockin (Tim Hockin) closed this 27 days ago

https://github.com/kubernetes/kubernetes/issues/56901
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: foo-deny-external-egress
spec:
  podSelector:
    dnsName: control-plane.io
  policyTypes:
    - Egress
  egress:
    - ports:
      - port: 53
        protocol: UDP
      - port: 53
        protocol: TCP
    - to:
      - namespaceSelector: {}
netassert - cloud native network testing

- netassert - network security testing for DevSecOps workflows
  https://github.com/controlplaneio/netassert

```
host:
  localhost:
  bitbucket.com:
    - 22
  control-plane.io:
  github.com:
    - 22
```
netassert - cloud native network testing

```yaml
k8s: # used for Kubernetes pods
deployment: # only deployments currently supported
test-frontend: # pod name, defaults to `default` namespace
test-microservice: 80 # `test-microservice` is the DNS name of the target service
test-database: -80 # should not be able to access port 80 of `test-database`

new-namespace:test-microservice: # `new-namespace` is the namespace name
test-database.new-namespace: 80 # longer DNS names can be used for other namespaces
test-frontend.default: 80

default:test-database:
test-frontend.default.svc.cluster.local: 80 # full DNS names can be used
test-microservice.default.svc.cluster.local: -80
control-plane.io: 443 # we can check remote services too

https://github.com/controlplaneio/netassert
```
TAP version 13
ok 1 - localhost TCP:30731 closed
ok 2 - localhost UDP:1234 closed
ok 3 - localhost TCP:22 open
ok 4 - binarysludge.com TCP:443 open
ok 5 - localhost TCP:999 closed
ok 6 - control-plane.io TCP:443 open
ok 7 - localhost UDP:555 closed
ok 8 - control-plane.io TCP:80 open
ok 9 - binarysludge.com TCP:22 open
ok 10 - binarysludge.com TCP:80 open
ok 11 - 8.8.8.8 UDP:53 open
ok 12 - google.co.uk TCP:443 open
ok 13 - binarysludge.com TCP:81 open
ok 14 - 8.8.4.4 UDP:53 open

1..14
# tests 14
# pass 14
# fall 0
Cloud Native Dynamic Firewalls

- Network Policy recipes - https://github.com/ahmetb/kubernetes-network-policy-recipes
- NeuVector Container Firewall - https://neuvector.com/products/
Applications: CNI and Network Policy
# Applications: CNI and Network Policy

<table>
<thead>
<tr>
<th>Provider</th>
<th>Network Model</th>
<th>Route Distribution</th>
<th>Network Policies</th>
<th>Mesh</th>
<th>External Datasstore</th>
<th>Encryption</th>
<th>Ingress/Egress Policies</th>
<th>Commercial Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Calico</td>
<td>Layer 3</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Etcd</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Canal</td>
<td>Layer 2 vxlan</td>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
<td>Etcd</td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>flannel</td>
<td>vxlan</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>kopeio-networking</td>
<td>Layer 2 vxlan</td>
<td>N/A</td>
<td>No</td>
<td>No</td>
<td>None</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>kube-router</td>
<td>Layer 3</td>
<td>BGP</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>romana</td>
<td>Layer 3</td>
<td>OSPF</td>
<td>Yes</td>
<td>No</td>
<td>Etdc</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>WeaveNet</td>
<td>Layer 2 vxlan</td>
<td>N/A</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

[Choosing a CNI Provider](#)
Bootstrapping identity with SPIFFE
Attestation Example: Kubernetes

/proc/[pid]/cgroup
Who am I?

You are spiffe://acme.com/fe

And here is your short-lived key to prove it to others.
SPIFFE ID

spiffe://acme.com/billing/payments

Trust Domain  Workload Identifier

@sublimino
SPIFFE Verifiable Identity Document

spiffe://acme.com/billing/payments

Typically short-lived

Today only one form of SVID (X509-SVID). Other document types under consideration (including JWT-SVID)
X.509 RFC Format

Certificate ::= SEQUENCE {
    tbsCertificate TBSCertificate,
    signatureAlgorithm AlgorithmIdentifier,
    signatureValue   BIT STRING  }

TBSCertificate ::= SEQUENCE {
    version         [0] EXPLICIT Version DEFAULT v1,
    serialNumber    CertificateSerialNumber,
    signature       AlgorithmIdentifier,
    issuer          Name,
    validity        Validity,
    subject         Name,
    subjectPublicKeyInfo SubjectPublicKeyInfo,
    issuerUniqueID  [1] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
    subjectUniqueID [2] IMPLICIT UniqueIdentifier OPTIONAL,
    -- If present, version MUST be v2 or v3
    extensions      [3] EXPLICIT Extensions OPTIONAL
    -- If present, version MUST be v3
}

Version ::= INTEGER  {  v1(0), v2(1), v3(2) }

CertificateSerialNumber ::= INTEGER

Validity ::= SEQUENCE {
    notBefore  Time,
    notAfter   Time }

Time ::= CHOICE {
    utcTime  UTCTime,
    generalTime GeneralizedTime }

UniqueIdentifier ::= BIT STRING

SubjectPublicKeyInfo ::= SEQUENCE {
    algorithm AlgorithmIdentifier,
    subjectPublicKey BIT STRING  }

Extensions ::= SEQUENCE SIZE (1..MAX) OF Extension

Extension ::= SEQUENCE {
    extnID OBJECT IDENTIFIER,
    critical BOOLEAN DEFAULT FALSE,
    extnValue OCTET STRING
    -- contains the DER encoding of an ASN.1 value
    -- corresponding to the extension type identified
    -- by extnID
}

https://github.com/spiffe/spiffe/blob/master/standards/X509-SVID.md
#appendix-a-x509-field-reference
Certificate Path Validation

Certificate Authority

Leaf Certificate

Intermediate Certificate
SPIFFE Runtime Environment

spiffe://acme.com/billing/payments

selector: aws:sg:sg-edcd9784

selector: k8s:ns:payments

selector: k8s:sa:pay-svc

selector: docker:image-id:442ca9
**SPIRE**

*Secure Introduction to other services*
- mTLS
- JWTs
- gRPC

*Identity for proxy services*

*Simplify deployment of distributed systems*

**Workload attestor plug-ins**
- Node

**Workload API**
- Identity for proxy services

**Core**

**Workload API**

**Workload attestor plug-ins**
- Linux
- OS X
- Kubernetes

**Node attestor plug-ins**
- Azure
- HSM providers
- OS X
- YubiKey
- Mesosphere
- GCP
- AWS
- Kerberos
- Windows
- Join Token

**Platform**
What SPIFFE is *not*

- **Authorization** (however it provides identities upon which authorization schemes can be deployed)
- **Transport level security** (however SVIDs can be used to facilitate things like TLS or JWT signing)
Using SPIFFE in TLS Certificates

https://www.slideshare.net/MattBaldwin3/istio-cloud-native-online-series-intro-to-istio-security
Istio and SPIFFE

@sublimino

https://www.slideshare.net/MattBaldwin3/istio-cloud-native-online-series-intro-to-istio-security
Recap
End to End Encryption

- TLS on API Server Components
- SPIFFE to identify application workloads
- Istio CA to issue TLS certificates to application workloads
- Envoy to proxy application’s HTTPS traffic across the Istio service mesh
Takeaway: Encrypt Everything Everywhere

- Encrypt
Takeaway: Encrypt Everything Everywhere

- Encrypt
- Encrypt Everything
Takeaway: Encrypt Everything Everywhere

- Encrypt
- Encrypt Everything
- Encrypt Everything Everywhere
ALERT: OBLIGATORY HIRING NOTIFICATION

- Hacking cloud native systems (from London)
I HAVE NO IDEA WHAT I'M DOING
OBLIGATORY HIRING NOTIFICATION

- Hacking cloud native systems (from London)
- Working on difficult problems
  - Secure infra and app delivery and provenance
  - Continuous security patterns
  - Supply chain security
  - Third party code risk
HIRING THAT NEW INTERN HAS BEEN A CAT-ASTROPHE.
OBLIGATORY HIRING NOTIFICATION

● Hacking cloud native systems (from London)
● Working on difficult problems
  ○ Secure application delivery and provenance
  ○ Continuous security patterns
  ○ Supply chain security
  ○ Third party code risk
● Good pay, no monoculture, generous benefits, travel
● Engaged minds wanted, on-the-job training
● Infrastructure and build engineers, DevOps, SRE, hackers, security engineers
OBLIGATORY HIRING NOTIFICATION

- Hacking cloud native systems (from London)
- Working on difficult problems
  - Secure application delivery and provenance
  - Continuous security patterns
  - Supply chain security
  - Third party code risk
- Good pay, no monoculture, generous benefits, travel
- Only prereq is an engaged mind
- Infrastructure engineers, DevOps, SRE, hackers, security engineers
Conclusion

● Network Security is important
● TLS, X.509, and Network Policies keep us safe
● Cloud Native applications have more security primitives than ever before
● Istio and SPIFFE give you wings
● Encrypt Everything Everywhere