Proxies, TLS, and Injecting Security into Cloud Communication

Susan Hinrichs
shinrich@verizonmedia.com
Verizon Media

verizon media

ONES 2020

About me

Work with Edge Team at Verizon Media

- **Support Apache Traffic Server. Committer, PMC member**
- **Also work on various routing, networking tools**

Teach yoga in my spare time

Reach me at

- **shinrich@verizon.media, shinrich@apache.org**
- **https://www.linkedin.com/in/shinrich/**

verizon media

Goal of Talk

Present TLS scenarios we encountered while adapting services to hybrid cloud scenarios

Describe how we use a proxy (Apache Traffic Server) and Access Control system (Athenz) to secure new traffic patterns

- **Proxy supporting Application Delivery Network (ADN)**

Work performed with my colleagues at Verizon Media and members of the Apache Traffic Server and Athenz communities

- **Persia Aziz, Zeyuan Yu, and Alan Carroll, and Henry Avetisyan**

Apache Traffic Server: Web Cache and Proxy

Apache Traffic Server (ATS) has been used for many years. Primarily for the traditional web serving model.

> **Allows injection of business specific logic via “plugins” triggering on hooks during the lifecycle of a HTTP transaction and session.**

We have spent the past three years enhancing TLS configuration to better support our evolving requirements to support hybrid cloud and fine-grained mutual TLS based authentication

https://trafficserver.apache.org/

Athenz: a Role Based Access Control (RBAC) System

Athenz is a open source RBAC system that leverages TLS mutual authentication

Contributed to open source by Yahoo/Verizon Media

Athenz has been the basis for much of our Verizon Media authentication work

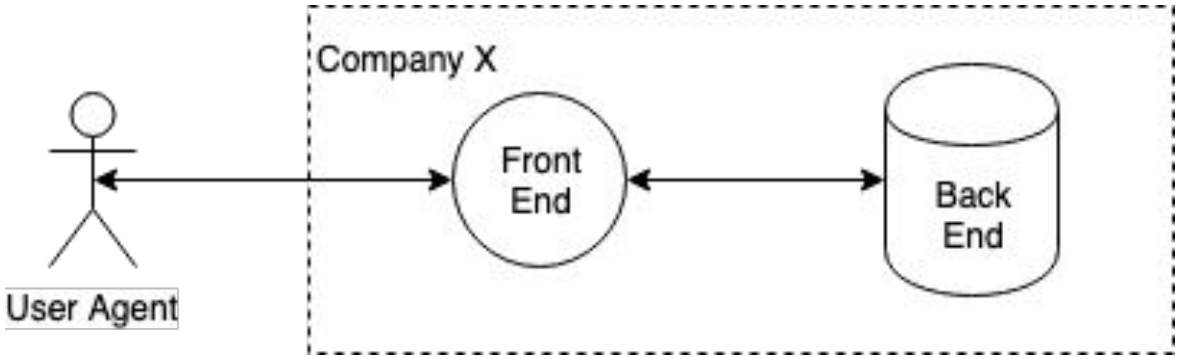
- https://github.com/yahoo/athenz

Motivation

verizon media

The Old Days, Corporate Data Centers

Dedicated resources to a single organization

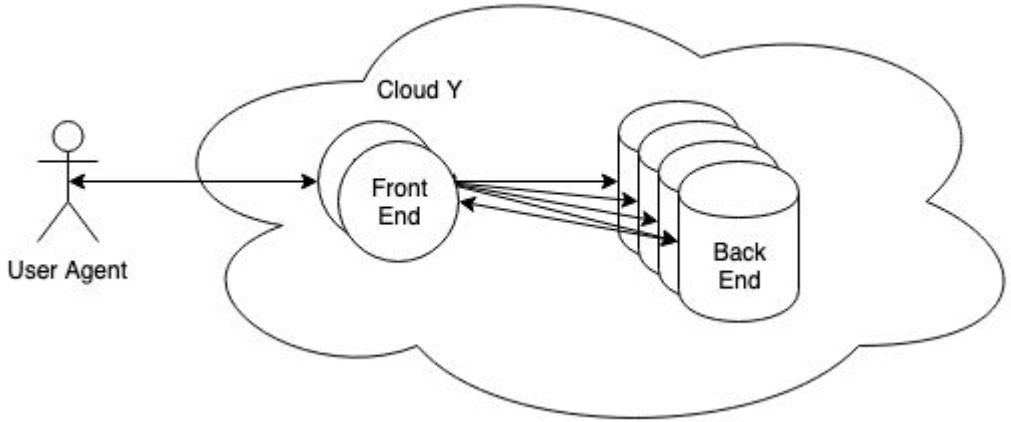


Recent Past, Public Cloud

Each organization rents virtual resources from a 3rd party cloud service to run company apps

No employee investment in machine care and feeding. Quick to scale up and down.

But expensive for large, dedicated resource usage

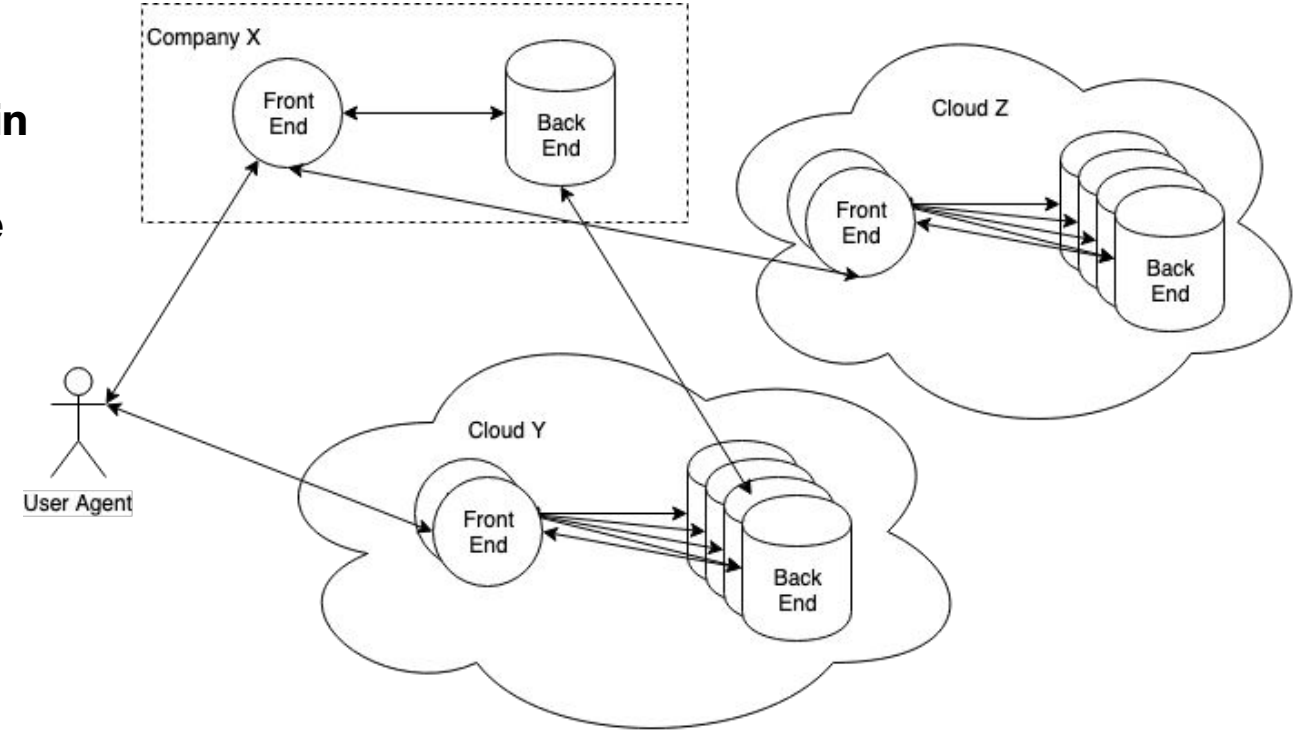


verizon media

Now, Hybrid Cloud

Run applications in combination of public and private resources

Dynamic and complex trust relationships



verizon media

TLS Everywhere

Network encryption is no longer optional

- Obviously must encrypt across the Internet
- Even within the same location pressure to encrypt. - Big brother is watching

Authentication must become stronger

- Just checking IP address is not sufficient (if it ever really was)
- Service client and servers can be in different locations
- Service endpoints may move on the next deployment cycle

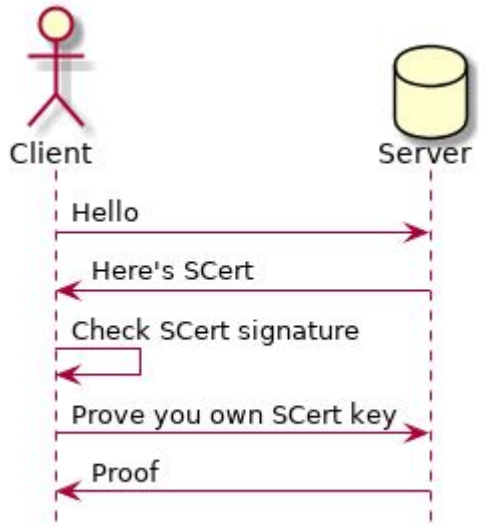
Injecting TLS Authentication

verizon media

Classic Client-Server TLS Authentication Model

Only server proves identity via certificate

Client authentication (if any) is through mechanism independent of TLS, e.g. password or 2FA

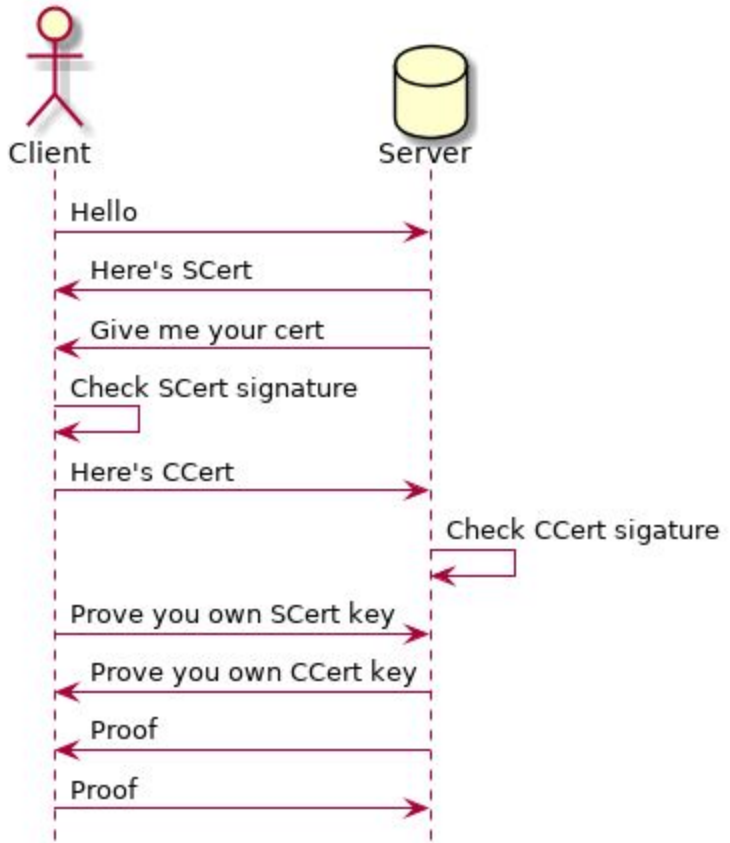


Mutual TLS Authentication Model

Client provides certificate too

Both sides verify the other's certificate

Athenz provides a Role Based Access Control (RBAC) system on top of mutual TLS authentication

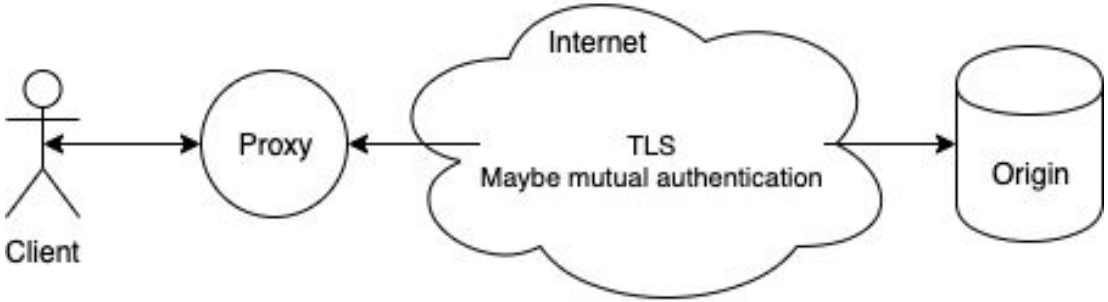


Retrofitting Clients for TLS

Process must connect to service using TLS

Client is not capable of TLS or maybe not capable of mutual TLS

- **Put a proxy like ATS in front of the client**
- **Configure ATS to provide the same certificate for all outgoing service requests.**
- **What if client only should mutual TLS to some services?**



Fine Grained Client Certificate Selection

Via conf_remap.so in remap.config

- **Specify override for proxy.config.ssl.client.cert.filename**
- `map https://bank.yahoo.com/pci https://pci.server.com/ @plugin=conf_remap.so @param=proxy.config.ssl.client.cert.filename=pci.pem`

Via sni.yaml

- **For FQDN Traffic Server requests to origin, use client_cert option to override default client certificate.**
- ```
  sni:
    -fqdn: supersensitive.example.com
     client_cert: supersensitive.pem
  ```

verizon media
  ```

Fine grained Server Certificate Verification

Ideally strongly verify all certificates provided by the origins

- **Validating both signature and the requested name is in the certificate**

Not all origins, may behave as we like.

- **Can disable name or signature checking on a per origin basis**
- **Via sni.yaml or conf_remap**

Can operate in permissive mode during initial testing deployment. Log failures but don't fail.

Also plugin verification hook `TS_SSL_VERIFY_SERVER_HOOK` to inject more deployment-specific verification.

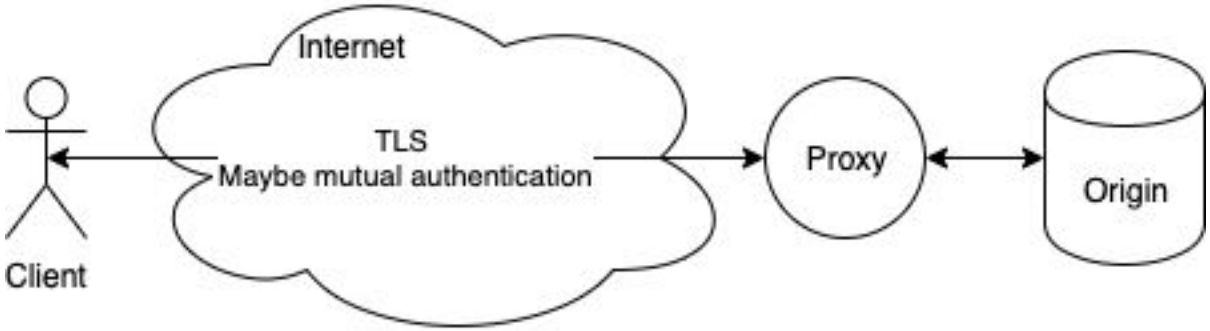
verizon media

Retrofitting Server for TLS

Server must provide a TLS interface for clients

Server is not capable of TLS or maybe not capable of requiring certificates from clients or doing adequate certificate verification

- **Put a proxy like Traffic Server in front of the server**
- **Configure Traffic Server to require certificates from all clients**
- **What if only some client requests require mutual TLS?**



verizon media

Using SNI to Drive TLS requirements

In Client Hello, user agent can set Server Name Indication (SNI) in the TLS Extensions

- **Most modern clients do this**

Server can adjust requirements for TLS handshake based on SNI value

- **Which HTTP protocols to offer via ALPN**
- **Which TLS protocols allow**
- **Whether to require a client certificate**
- **Which ciphers to offer**

Fine Grained Client Requirements

Within the sni.yaml file, can configure TLS server requirements based in the presented SNI

- `verify_client` Specify whether to request or require a client certificate
- `valid_tls_versions_in` Specify the set of TLS protocols to accept, e.g. don't offer TLS1.0 and TLS1.1 to very sensitive domains
- `http2` Add or remove HTTP/2 for the set of ALPN protocols
- `tunnel_route` Don't terminate the TLS connection on the proxy. Instead forward on the TLS handshake to the specified origin.
- `forward_route` Terminate TLS connection on proxy and blind tunnel contents to origin.

Augmenting Client Certificate Verification

The core ATS will validate the offered client certificate is signed by a certificate in the specified CA certificate list

Can build a plugin to hook on `TS_SSL_VERIFY_CLIENT_HOOK`

From there, plugin can access offered client certificate and make additional checks to possibly invalidate the TLS authentication

- **client allow list plugin - Will fail TLS handshake if certificate does not contain a name for the configured list of good names**
- **Could write a plugin to check for specific serial numbers or any other field in the certificate**

Misbehaving Clients

verizon media

What If User Agent Does Not Play By The Rules?

When making a request from Browser or curl, at least the first request on the TLS connection we assume SNI == Host header value

Host header value along with path is used by the remap rule to determine where to send transaction.

There is nothing intrinsic technically to force SNI == Host header value. Easy enough to use the curl command to violate that assertion

```
curl -H 'host: secure.com' https://insecure.com/sensitive-data
```

SNI == insecure.com, Host header value == secure.com

Cannot solely rely on SNI for policy decisions

Could have dedicated proxies that always ask for client cert. SNI is not part of the decision making process.

- May be other SNI-tied actions that need to be addressed

Could fail any transaction where Host header value != connection's SNI

- May be too onerous. If the SNI is not used for policy selection, doesn't matter.
  With browser connection/session reuse name differences may be innocuous.

What we ended up with

The mechanism

- **In transaction processing, use Host header value and see if that value would have selected a security-sensitive SNI policy.**
- **If so, fail transaction (return 403) if SNI != Host header value**

Controlled by proxy.config.host_sni_policy to disable, monitor, or enable this mechanism.

Another way of thinking about it

A draft RFC was brought up in the open source PR.

Secondary Certificate Authentication in HTTP/2

Rather than failing the transaction, attempt to request additional client certificates on the fly. More friendly and better able to use existing connections.

- Supporting implementations exist for HTTP/1.1
- Need updated implementation for HTTP/2
- We're keeping an eye on this work and may move there in the future

Secure Edge: Proxy TLS

verizon media

Proxying TLS via the Secure Edge

Much like retrofitting TLS Server. However, motivation is addressing client in one cloud and server in another cloud (e.g. hybrid cloud case).

Don’t want to expose cloud servers directly.

- **May not trust security of client/server implementations**
- **May find it more security to enforce security standards on a fixed set of Edge machines.**

Three options

- **TLS Delegation**
- **Tunnel**
- **Bridge**

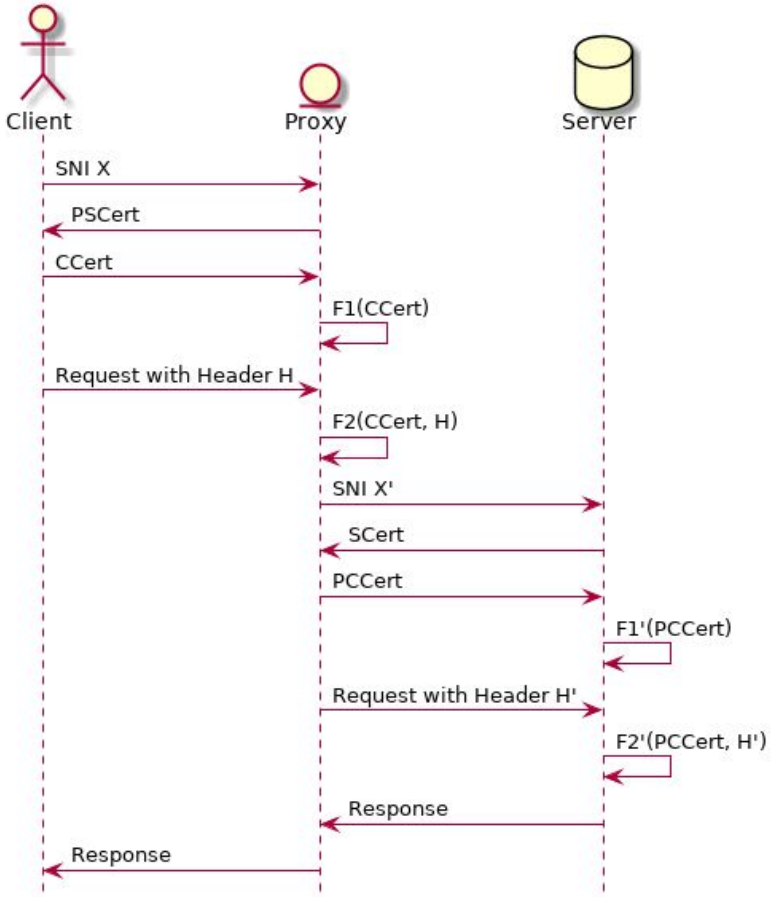
verizon media

TLS Delegation

Proxy validates CCert and authentication header H

Proxy presents PCCert and header H' with information about the client's authentication status

Must protect against malicious user presenting bogus H' directly to Server



verizon media

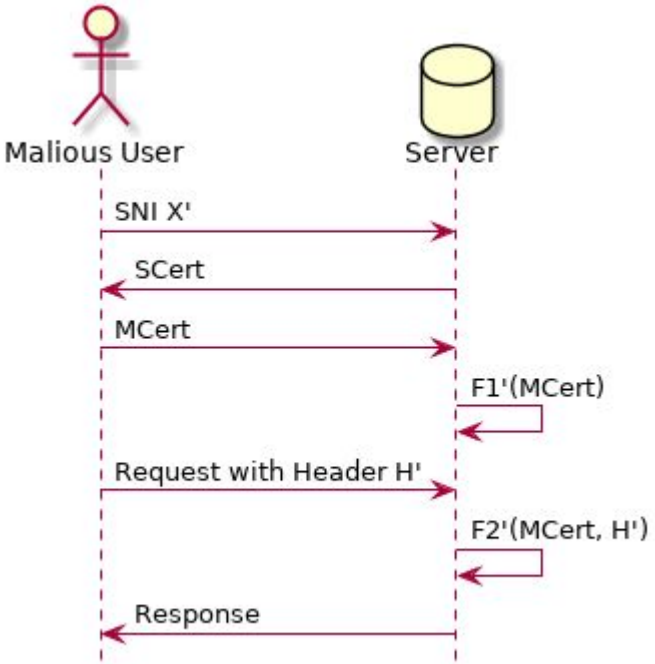
Attacking TLS Delegation

Request header H' only check if initial mutual TLS passes

If Malicious User can present and verify a certificate trusted by Server, all is lost.

Buying a cert signed by DigiCert or Symantec could be a problem

Server must trust a limited set of CA



verizon media

TLS Tunnel

Proxy does not terminate TLS connection

Looks at SNI and decides where to forward the TLS bytes

- **Actual TLS termination is end-to-end**

Proxy just acts as a SNI-based router in this case

```
sni:
  -fqdn: supersensitive.example.com
   tunnel_route: supersensitive.origin.example.com
```

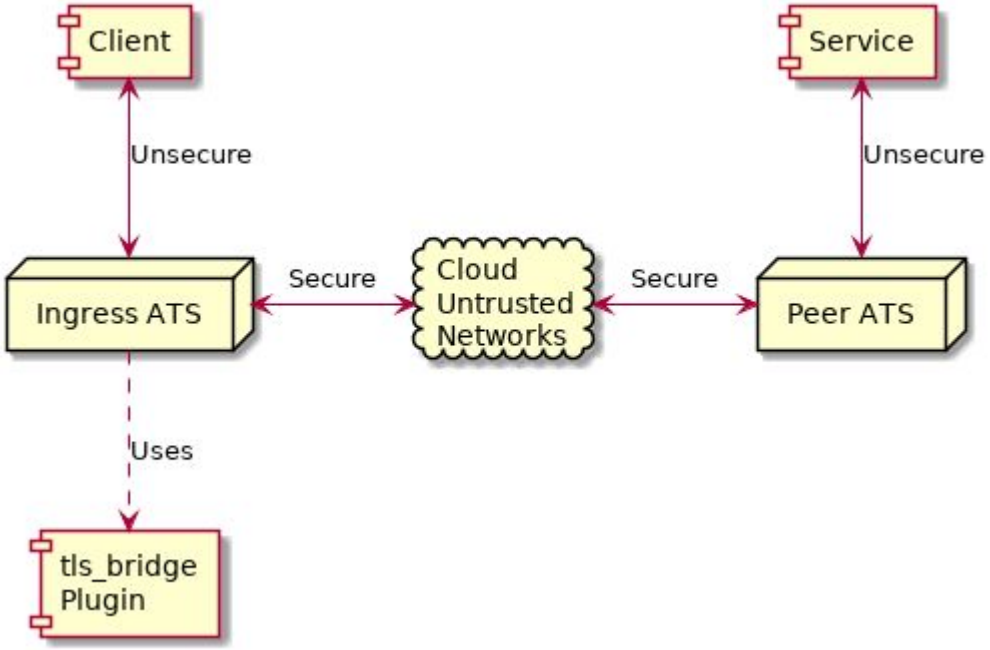
TLS Bridge

Avoid decryption on Proxy, but make some guarantees on strength of TLS connection independent of the user agent and/or server stack

Build a separate Tunnel between gateways. Double tunnel.

https://docs.trafficserver.apache.org/en/latest/developer-guide/plugins/example-plugins/tls_bridge.en.html

TLS Bridge



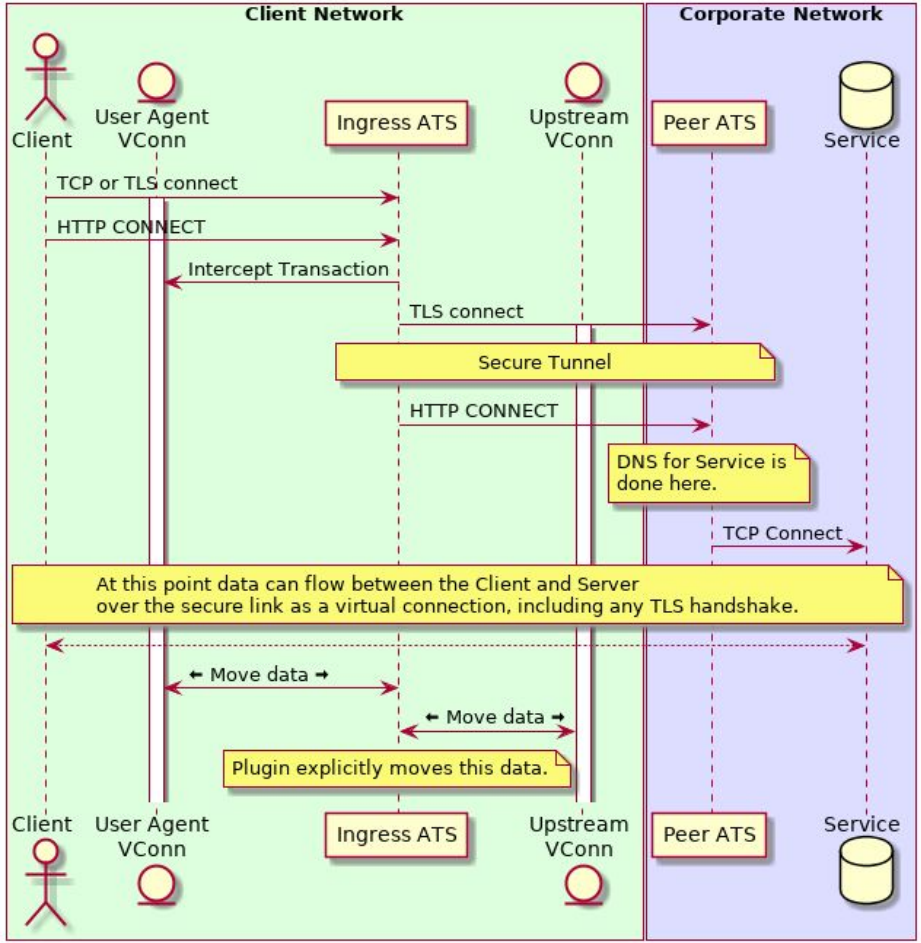
verizon media

TLS Bridge

Client connects to Ingress ATS

Proxy negotiates new TLS to Peer ATS

Proxies blind tunnel original traffic between client and service



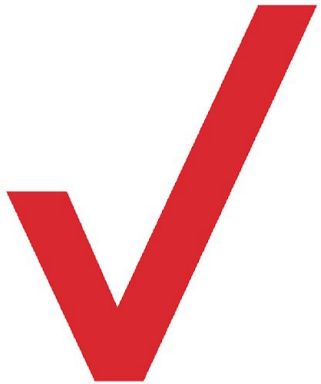
verizon media

Q&A

shinrich@verizonmedia.com, shinrich@apache.org

Slides at https://github.com/shinrich/presos/blob/master/apacheconna19.pdf

verizon media



Dynamic Certificate Support

verizon media

More, Shorter-Lived Certificates = Deployment Headaches

With client certificate authentication, the Proxy must handle many certificates

- Potentially one for each service

Ongoing process to reduce the lifetime of each certificate

- Moving from years or months to weeks or days

Complicates proxy configuration

- Does proxy need to be explicitly reloaded to pick up changes?
- How can we verify that the changes have been correctly applied?

Improving Support for Many Short-lived Certificates

Certificate reporting tool

https://docs.trafficserver.apache.org/en/latest/admin-guide/plugins/cert_reporting_tool.en.html

Hot loading certificates

https://docs.trafficserver.apache.org/en/latest/developer-guide/api/functions/TSSslClientCertUpdate.en.html

https://docs.trafficserver.apache.org/en/latest/developer-guide/api/functions/TSSslServerCertUpdate.en.html

Certificate Reporting Tool

Responds to message sent to plugin via the traffic_ctl utility

Dumps state of the currently loaded certificates (either client certificates or server certificates) to cert_reporting_tool.log file.

- **Each entry contains: Time, lookup name, subject name, all subject alternate names (SAN), serial number, and validity time.**

Looking at the dumps, the operator can verify whether the expected certificates were loaded.

Hot Loading Certificates

New Plugin APIs have been added in Traffic Server to update client or server certificates without a full configuration reload

We have been experimenting with a plugin that takes a message of the name of certificate files to reload

The operator can create a script using inotify or other knowledge of file updates to notify traffic server to reload specific certificates and keys