IETF 112 DANCE

HTTPS use case

Topics

Client identity challenges with HTTPS and mTLS

TLS-native vs TLS-cooperative

Q&A

Problems

Web applications frequently live behind a TLS-offloading load balancer

Clients are typically consolidated to app- or org-specific PKI

...org must build/buy/maintain a private PKI

Allowing clients across multiple private PKI can open the door to

- Identifier collisions
- Impersonation

Problems

Org needs its own PKI (cost of labor + infrastructure or SaaS)

How to manage auth settings/certs/etc in load balancer?

- Integrate LB auth settings with application?
- Manage through infrastructure automation?

In-handshake DANE client authentication

Client signals DANE client auth intent and identifier in handshake

DNS query happens during handshake

If DANE client auth fails, TLS handshake fails

If auth is successful, dane_clientid passed to web app in HTTP header

Caveats:

- Need an allow list for client hosts/wildcards for abuse prevention
- Architectural challenges with placement of DANE comparison functionality

TLS-Cooperative DANE client authentication

TLS-terminating load balancer configured to accept any certificate

Client performs TLS client auth with load balancer

Load balancer forwards:

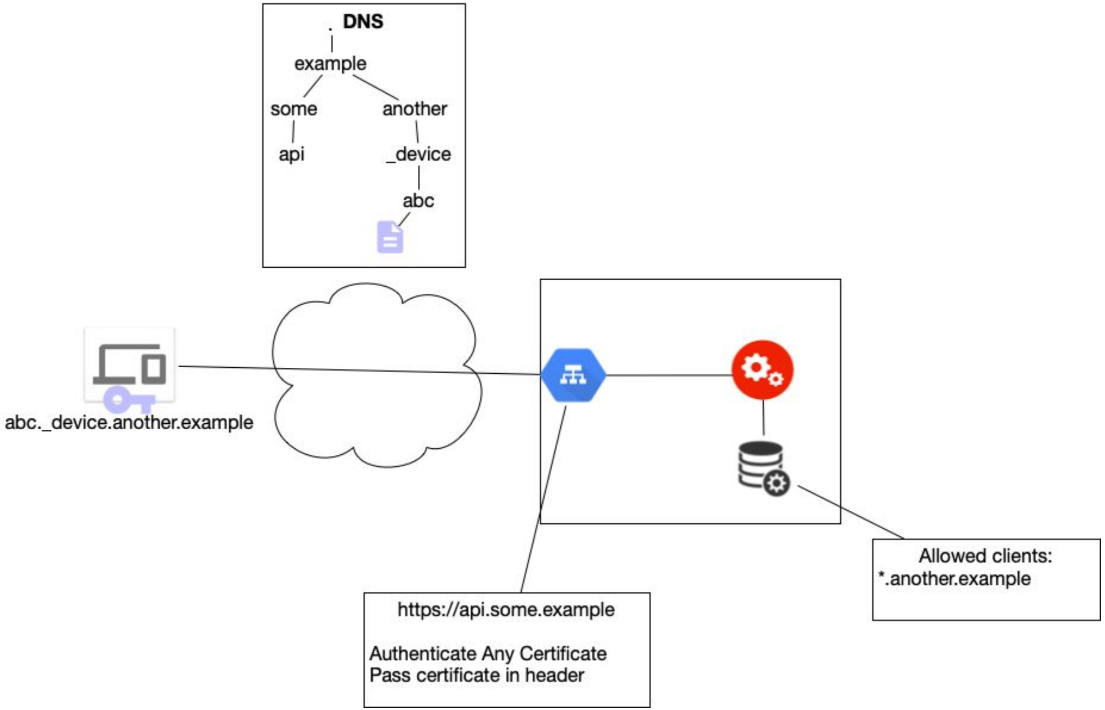
- Header containing certificate used in authentication
- Request body containing dane_clientid

Application server

- Compares dane_clientid to allowed client list
- Performs DANE TLSA query and comparison

Failed authentication signalled by HTTP 401/403 instead of TLS failure

Configuration

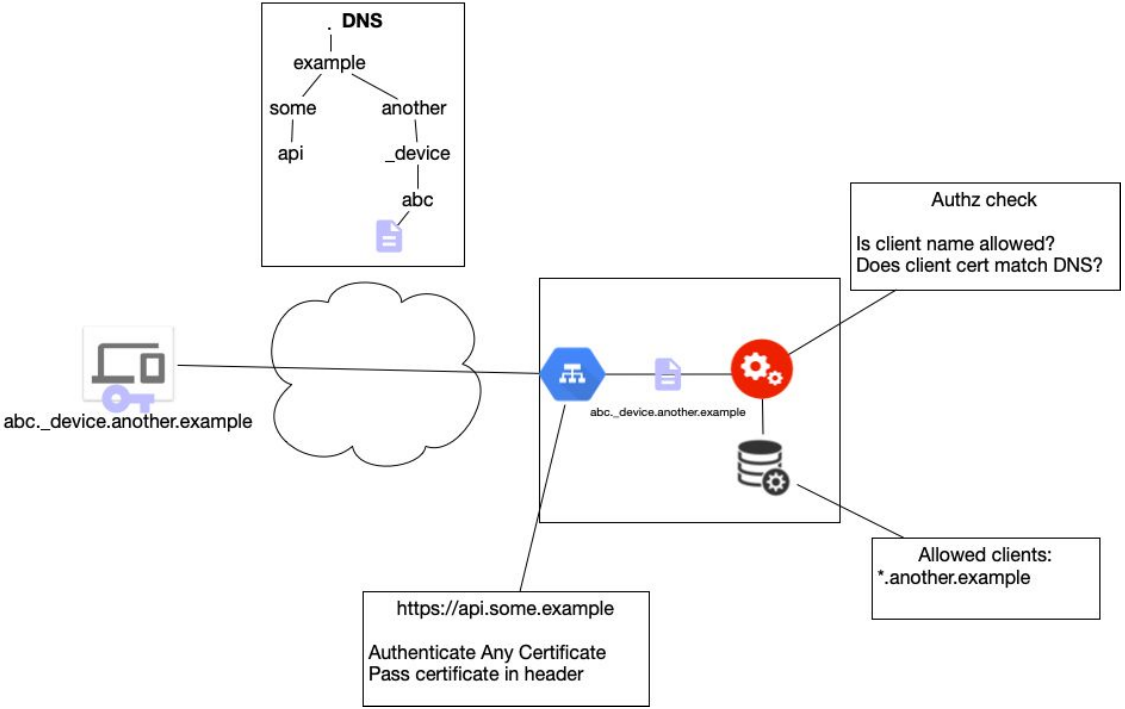


https://api.some.example

Authenticate Any Certificate
Pass certificate in header

Allowed clients:
*.another.example

TLS authentication



DNS

Authz check

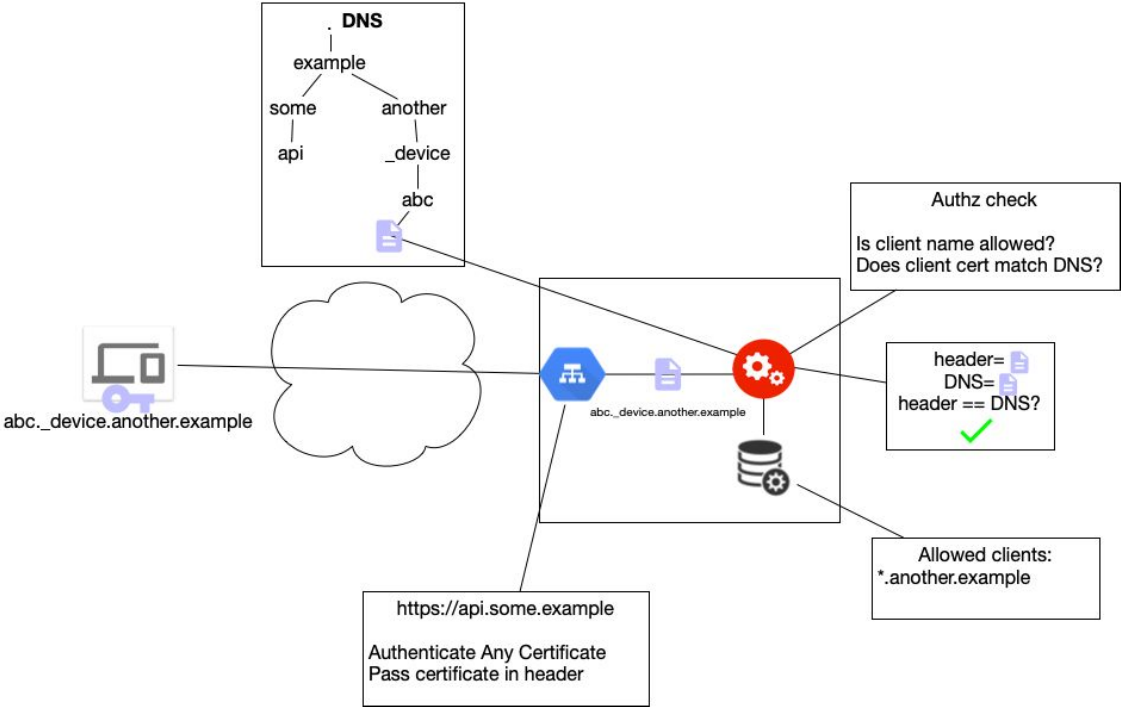
Is client name allowed?
Does client cert match DNS?

Allowed clients:
*.another.example

https://api.some.example

Authenticate Any Certificate
Pass certificate in header

DANE alignment check



Q&A