HTTP

Authority

Who can answer HTTP requests

What is authority in HTTP?

Authority determines...

> “[...] who has the right to respond authoritatively to requests that target the identified resource.”

— RFC 7230

What is an authority?

> authority = [userinfo "@"] host [":" port]
>
> — RFC 3986

> scheme, host, and port
>
> — RFC 6454

> If a subjectAltName extension of type dNSName is present, that MUST be used as the identity.
>
> — RFC 6454

Who decides?

Client decides whether to consider the server authoritative

Subtle differences in client requirements

e.g., which CAs are trusted, feature implementation

Servers only decide whether they can/should answer

About Caching

Caches provided non-authoritative responses to requests

Clients might accept a cached response as a substitute

http:// URIs (RFC 7230)

Whoever answers is authoritative

Transparent proxies, on-path attacks, DNS cache poisoning, route hijacking, can all subvert authority

… or maybe these are all valid authorities

https:// (RFC 2818, RFC 7230, and RFC 5280)

The ‘s’ in HTTPS creates a parallel system

https:// URIs come with an expectation of authentication

What “authentication” means is complex, but superficially:

> The server is expected to provide a certificate
> and cryptographic proof of control over a matching key

Servers that cannot be authenticated are not authoritative

Coalescing in HTTP/2 (RFC 7540)

HTTP/2 separates authority making a new TCP connection

If a server could be authoritative, why make a new connection?

Two conditions:

- The certificate must be good for the new name
- DNS must produce a matching IP for the new name

Alternative Services (RFC 7838)

Alternative services specify alternative resolution

Alt-Svc: h3="other.example:53"; quic=1

Think super CNAME; new information is used to find the endpoint

New protocol, new DNS QNAME, new port number

The service is still authenticated the same way (DNS, SNI and cert)

ORIGIN Frame (RFC 8336)

A server might not be authoritative for the names in its certificate

ORIGIN fixes that by advertising Origins that it wants to use

Contentiously, it removes DNS from authentication requirements

> “[…] clients MAY avoid consulting DNS to establish the connection's authority […]”
>
> Privacy gains, but security loss?

Secondary Certificates (…-secondary-certs)

Uses TLS exported authenticators to move certificates to HTTP

Aims to improve client certificate authentication

Enables more coalescing by letting servers add certificates

Uses ORIGIN for discovery

TLS Delegated Credentials (...-tls-subcerts)

Allows a certificate-holder to delegate their authority

Provides a short-lived certification of a key pair

The key is very tightly bound to the certificate

TLS encapsulates the entire process

Bonus: httpq://

Can a server on UDP be authoritative for https:// resources?

Can a new URI scheme deploy?

#include <ietf104/http>

Variations on a Theme

All variations on the same basic process:

find server, connect, check certificate

the connection provides authentication for exchanges

Surely that’s not all?

HTTP Signed Exchanges (…-signed-responses)

Authenticates content, not connections

Provides authenticity, not confidentiality

Why not a new URI scheme? (surely not shttp://)

Out-of-band Content Coding (…-oob-encoding)

Allows for providing the body of a message via alternative channels

Simpler integrity mechanisms to prevent transfer of authority

Similar characteristics to signed exchanges, but more limited

DANE for HTTP

DANE (RFC 6698) uses DNSSEC to carry authorized certificates

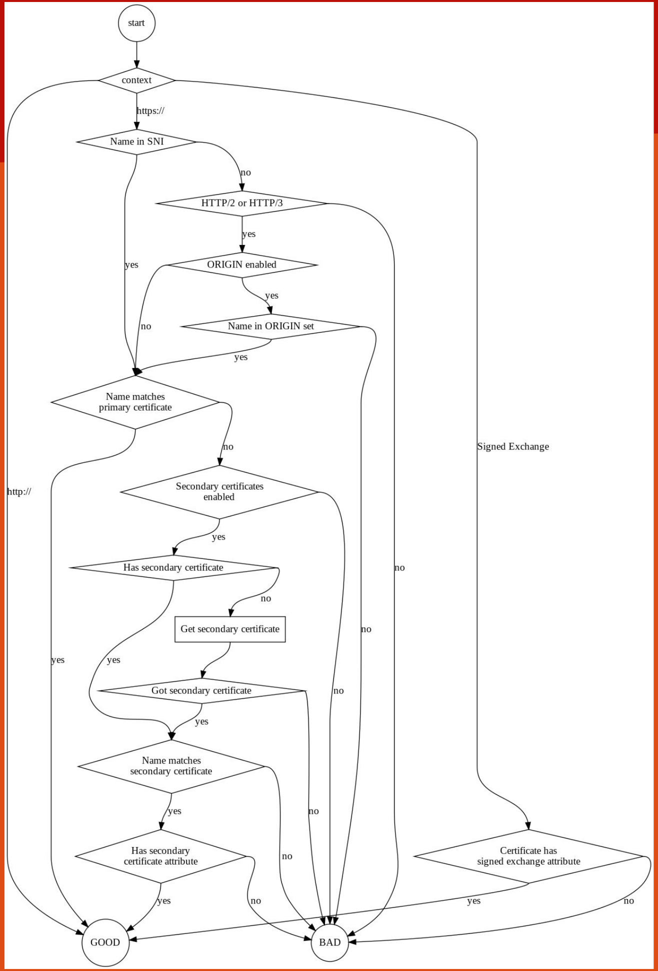
Assertions are made for specific services, not authorities:

e.g., _443._tcp.example.com

DANE is either supplemental to a PKI, or a replacement for it

A Picture

I tried, but it is just too complicated



Discussion Points

What comprises authority?

What is the role of DNS?

What about that QUIC thing?

Do we need to talk about MX records so we can all feel better?