client authentication

•••

with TLS and HTTP/2

history

HTTP/1.1 “allowed” servers to use TLS renegotiation to authenticate clients

… server holds the request, renegotiates, then authorizes based on the client certificate

This is not possible in h2

… concurrency makes that design infeasible

… TLS renegotiation is prohibited

ignoring a problem doesn’t make it go away

People actually rely on this behaviour in HTTP/1.1

This is holding back deployments of h2

solution overview

Signal using an h2 setting that a client allows TLS renegotiation

If a request requires TLS-layer authentication, then

... provide an identifier in a **WAITING_FOR_AUTH** frame

... add client authentication at the TLS layer, referencing the identifier

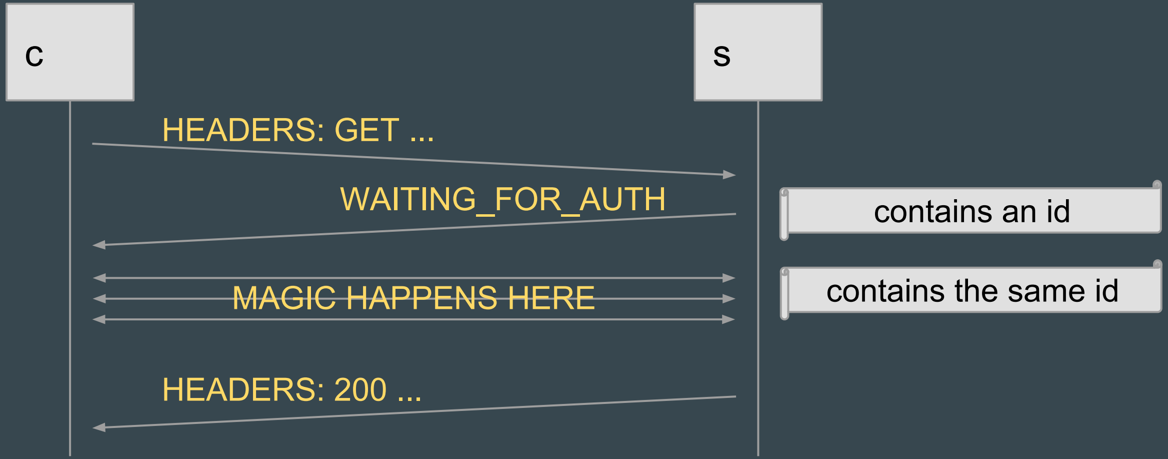
Multiple concurrent requests can await the same set of credentials

A single request can await multiple credentials

part 1.1 - WAITING_FOR_AUTH frame

WAITING_FOR_AUTH contains an opaque octet string: the identifier

It can be sent by a server when the client has an outstanding request



part 1.2 - tls 1.2 magic

In TLS 1.2, the magic is a server-initiated renegotiation

The identifier is carried in a **ClientHello** extension

```
enum {
    ..., application_context_id(EXTENSION-TBD), (65535)
} ExtensionType;

struct {
    opaque id<0..255>;
} ApplicationContextId;
```

part 1.3 - tls 1.3 magic

In TLS 1.3, there is no renegotiation

The server will send a TLS **CertificateRequest**, which will contain the identifier

Warning: This is not yet final in TLS 1.3, details are forthcoming

part 2 - setting negotiation

The setting **SETTINGS_REACTIVE_AUTH** defaults to 0

The client advertises **SETTINGS_REACTIVE_AUTH** = 1 to enable this

No setting, no play

don’t use this feature

adopt me