Gateway API TLS Use Cases

This document attempts to round up TLS use cases for Gateway API and talk about how they should be handled, for both common and uncommon use cases.

Please be courteous and don't edit other people's use cases, either comment or add a suggestion.

The format is this:

Heading describing use case

Short explanation of what the use case is about, and who owns the TLS config.

Comments
- [github tag of comment owner] Opinions about how the use case should/is solved.

We should consider what the impact of the roles that we defined in the Gateway API: infrastructure op, cluster admin, application dev and how they impact the use cases below:
- Who is able to control/create resources that control sensitive things such as Secrets, TLS settings to be used
    - Can a Service force a proxy to use an insecure protocol version (e.g. SSLv2) when the proxy infrastructure is configured to not allow the protocol be used.
    - Can a Service replace the CA used for validation if the proxy infrastructure (e.g. Gateway) is configured with a different CA?
- What the semantics of what is represented/expected behaviour:
    - Should the configuration include things that also are part of a negotiated protocol (e.g TLS crypto negotiation).
    -

1. Termination of TLS for HTTP Routing

For this, a Gateway owner has a TLS keypair, stored in a Kubernetes Secret, that they want to use to terminate TLS, and then route the unencrypted HTTP traffic *based on HTTP Properties* (like path, method, headers, and so on).

Comments:
- [youngnick] This is served by the TLS stanza in the listener, with a HTTPRoute to direct the traffic. It's expected that the TLSMode be Terminate, and the Protocol be either TLS or HTTPS (it doesn't matter either way)

2. HTTPS Passthrough

For this, a service owner is running a workload that expects encrypted traffic to show up at the workload pods. The encryption is handled between the client and the workload pod, and the Gateway is not expected to inspect the traffic aside from using the SNI header for routing.
Comments:

- [youngnick] This is what TLSRoute is for. Multiple TLSRoutes can be attached to a TLS listener on port 443, with the hostnames field on the TLSRoute used to discriminate between them. The TLS Mode should be passthrough, and the Protocol either HTTPS or TLS. Although https://gateway-api.sigs.k8s.io/concepts/api-overview/#route-summary-table suggests that "Terminated" is possible with TLSRoute, I think that's more confusing than helpful. I'll talk more about why you might use that in another use case, but in this one, it's just misleading.

3. Termination of TLS for non-HTTP TCP streams

In this case, service owners want to have TCP streams arrive at their workload, and either the service owner or the Gateway owner (if they are different) wants the TCP stream arriving at the Gateway to be encrypted. The keypairs need to be owned by the Gateway owner or accessible by them.

Comments:

- [youngnick] This is the TCPRoute attached to a Listener with a TLS stanza use case. The Protocol should be either TLS or TCP, and the TLS mode should be Terminate.

4. Simple HTTPS Termination and Re-encryption

Here, the Gateway owner wants to terminate TLS at the outside of the Gateway, and the workload running *inside the targeted pods* is expecting TLS-encoded traffic to arrive at its ports. (There's a good chance that this will use a self-signed CA). This could be because there's a policy mandating that all wire traffic is encrypted, and there's no other capability to do it, because the Service owner wants to, or some other reason. But the critical part is that the *workload owner* is *managing their own TLS*, or to put it another way, the settings control the *Gateway*'s connection *to the backend service*, not the *backend's TLS configuration.*

Comments:

- [candita] This has become the experimental BackendTLSPolicy in v1.0.0 of GatewayAPI.
- [youngnick] This specific use case is what originally had me start thinking about what turned into GEP 1282, BackendProperties. Right now, there is no way at all to

describe what is, in the absence of a service mesh or other relatively-magic automatic TLS infrastructure, a common operation.
- [markmc] with the use cases laid out this way, this one seems purely additive to “Termination of TLS for HTTP Routing” - the route owner additionally wants to modify the stream to encapsulate it in TLS. Have ever considered modelling this as a filter?
- [howardjohn] I don't know if we want another section, but it equally valid to do "HTTP in, HTTPS out". An egress gateway, for example, would be a great example of this.
- [youngnick] Another setting to keep in mind here is controlling the client certificate used for the Gateway -> backend connection. It probably won’t be sufficient to have a single client certificate across the entire Gateway, so we will need a way to configure this on the same basis that we configure “should I encrypt to the backend”.
- [youngnick] TODO Discussion about this in meeting re magic TLS vs configured.

5. Controlling TLS versions or ciphersuites used

Here, someone (usually either a Gateway owner or someone who can tell the Gateway owner what to do, but sometimes a Route owner) is mandating what TLS version or ciphersuite can be used. Perhaps the organization has security requirements that a particular TLS version doesn’t match, or perhaps the organization has legacy stuff that doesn’t support a newer TLS version, and they need a way to tell more modern proxy implementations (that often disable TLS 1.1, for example), that they want old versions or ciphers enabled.

Comments:
- [youngnick] There’s currently no way to represent this in the API, but my previous experience with Contour makes me reluctant to only have one place to configure this. For Contour, we ended up with being able to configure this at the route level (inside a HTTPProxy), with a global default or override. This eventual design was the genesis of the defaults and overrides in Policy attachment. The key features of what we ended up with are:
    - Sometimes Route owners want to specify this information
    - *Forcing* Route owners to specify this information on *every* Route is annoying, so some form of defaulting is very useful.
    - Sometimes Gateway owners (or GatewayClass owners) want to be able to *force* certain minimum or maximum TLS versions, or certain cipher suites.

  For those cases, a per-Route or per-backend setting with a common Policy attachment default or override would make the most sense to me. The per-backend settings are one of the things that GEP 1282 was targeting with BackendProperties.

6. Controlling certificates used

Sometimes there are global certificates or shared certificates which are configured to take the place of individual certificates in order to complete a TLS transaction. For example, the GatewayClass could be configured by the infra-admin role to supply a default global

certificate to be used for certain workloads that pertain to infra operations. Or a GatewayClass that is configured for a certain customer that needs a global certificate but not individual certificates for specific Gateway workloads. In most cases this would be used to provide an easy installation process for specific cloud platforms, or applications, with the intention of updating a default certificate before exposing the cluster as production-ready.

Comments:

- [candita] There's currently no way (that I know of) to express this in the API, but the Gateway certificateRefs could be configured with referenceGrants in the target namespace to allow the certificate to be used by other objects. ParametersReference in the GatewayClass is a group-value-kind struct that could be used to configure almost anything for the GatewayClass. It's not clear how widespread this is a use case, but it bears a mention.

7. Client Certificate settings

For TLS use cases where the handshake is also used to confirm the clients identity (this is "mutual TLS", but I'm reluctant to use that phrasing because mTLS has *very specific* meanings in Service mesh contexts), it's useful to be able to specify some settings for the client certificates as well as the serving certificates.

*Note that this is for **external** clients, that are connecting to the **Listener**, not for connections from the **Gateway** to **backends.***

Comments:

- [youngnick] This is particularly useful for JWT use cases, and can make doing some types of authentication and authorization easier.