Envoy OCSP Stapling

Stephan Zuercher <zuercher@gmail.com>, May 2020

Problem

Certificate Revocation Lists (CRLs) and Online Certificate Status Protocol (OCSP) checks require a third party, namely the Certificate Authority (CA), to provide an online service to enable certificate validation. Browsers may choose a hard-fail strategy and respond to failing CRL or OCSP requests by blocking website access. Thus, the availability of the CA's services becomes a factor in the availability for every website using a certificate issued by the CA. Because users become inured to the resulting warnings, they are likely to ignore them. Therefore, browsers may instead adopt a soft-fail strategy such that the inability to check the certificate is handled by assuming the certificate remains valid. While normally true, the soft-fail strategy allows an attack where certificate verification requests are blocked by the attacker, resulting in a failure to detect a revoked certificate. In addition, these requests to the CA result in slower connections, especially to infrequently visited sites. For these reasons, some browsers depend only on CRLs distributed periodically and eschew network-based checks altogether.

A solution to this problem is known as OCSP Stapling. OCSP stapling uses an extension to the Transport Layer Security (TLS) handshake to pass an OCSP response from the TLS server to the client. The OCSP response provides proof-of-validity during the TLS handshakes, which in turn enables improved security and performance.

The remainder of this document describes the requirements and implementation details for an OCSP stapling implementation in Envoy.

Requirements

1. Envoy must be capable of serving stapled OCSP responses during TLS negotiation with downstream clients.
2. OCSP responses must be delivered to Envoy via either the control plane or from local disks.
3. OCSP responses must be validated to determine that they match the corresponding TLS certificate and are not expired.

Non-Requirements

1. OCSP response validation on upstream TLS connections. Note that Requirement 3 likely constitutes the bulk of the code necessary to perform this function, so it shouldn't be difficult to implement at a later date..

2. Envoy will not directly make OCSP requests to CAs. The implementation proposed here must not preclude a future enhancement that allows Envoy to transmit OCSP requests to a CA, cache the response locally, and use it for OCSP stapling.

Implementation

Configuration

The following discusses the Envoy v3 API. All proposed fields will be promoted to the v4 API in the appropriate locations.

OCSP responses are specific to a TLS certificate, so the OCSP response is stored with the certificate. Indeed, the Envoy control plane anticipates this in `envoy.extensions.transport_sockets.tls.v3.TlsCertificate` with an unimplemented `ocsp_staple` field of type `envoy.config.core.v3.DataSource`. This allows the OCSP response to be provided inline or via a file on the local filesystem. The field is ignored for upstream TLS contexts.

The referenced OCSP response must be for a single certificate only. The response must indicate that the OCSP request succeeded. If provided in a file or as inline bytes, the response must be a DER-encoded binary OCSP response as in RFC 6960, Appendix B. If provided as an inline string, the DER-encoded binary OCSP response should be base64 encoded. No header or footer is specified, because no standard PEM encoding label appears to be defined for OCSP responses.

Because it may be impossible to obtain a valid, unexpired OCSP response in all cases, additional configuration is necessary to control how Envoy responds to a missing, invalid, or expired OCSP response. To that end, a new field is added to `envoy.extensions.transport_sockets.tls.v3.DownstreamTlsContext` with an enumeration type:

```
enum OcspStaplePolicy {
      // OCSP responses are optional. Invalid or expired OCSP
      // responses are rejected at config time. If an OCSP
      // response expires before a config update, it is no
      // longer stapled.
      SKIP_STAPLING_IF_EXPIRED = 0;

      // OCSP responses are required on all certificates for a
      // configuration to be deemed valid. If an OCSP response
      // expires before a config update, subsequent TLS
      // handshakes for that certificate are aborted.
```

```
      STAPLING_REQUIRED = 1;

      // OCSP responses are optional. Invalid or expired OCSP
      // responses are rejected at config time. If an OCSP
      // responses expires before a config update, subsequent
      // TLS handshakes for that certificate are aborted.
      REJECT_CONNECTION_ON_EXPIRED = 2;
  }
  OcspStaplePolicy ocsp_staple_policy = 8
      [(validate.rules).enum = {defined_only: true}];
```

Validation

If the TLS certificate is marked “must-staple” (see RFC 7633) or if Envoy’s OCSP stapling policy is set to `STAPLING_REQUIRED`, the OCSP response becomes a required configuration field and its absence will cause a validation failure.

Envoy must validate that the OCSP response, if present, is well-formed, matches the correct TLS certificate, indicates success, and has not expired (as of configuration load time).

Implementation

During configuration, Envoy creates a BoringSSL `SSL_CTX` for each TLS certificate. OCSP responses will be assigned to the `SSL_CTX` instances at that time. Later, during the TLS handshake, Envoy selects one of these pre-existing `SSL_CTX` instances based on the client by matching clients that support ECDSA with ECDSA certificates. If the OCSP response has expired at this point, Envoy will abort the TLS handshake by returning `ssl_select_cert_error` if required by the OCSP stapling policy or if the certificate is marked as must-staple. On success, the handshake is resumed and BoringSSL will automatically staple the OCSP response if the client has indicated support.

In order to keep the logic around skipping expired OCSP response stapling vs rejecting connections simple, it may make sense to update the `Envoy::Extensions::TransportSockets::Tls::ServerContextImpl` when the response expires (e.g. using a timer). It should be possible to simply flag certificates that should reject connections due to expired OCSP responses using the same mechanism and thereby avoid checking the system time during the certificate selection callback.

Runtime Flags

Envoy must provide the runtime flags to disable the following parts of OCSP stapling:

- Disable requiring OCSP responses for “must-staple” certificates.
- Disable expiration validation on the OCSP response at configuration load time.

- Disable expiration validation on the OCSP response during connection.

Stats and Administration

Envoy will report the following stats:

- Gauges:
    - Seconds until the first OCSP response expires. This is similar to the existing “days until the first certificate expires” statistic currently reported.
- Counters:
    - OCSP stapled request. By querying `SSL_get_tlsext_status_type`, Envoy can count TLS handshakes that support a stapled OCSP response.
    - OCSP stapled response counter. This is inferred when clients report OCSP stapling support and Envoy configures an OCSP response.
    - OCSP staple omitted. Counts OCSP responses omitted due to an expired OCSP response, per the OCSP stapling policy.
    - OCSP staple failed. Counts TLS handshakes terminated due to an expired OCSP response, per the OCSP stapling policy.

The Envoy admin endpoints will be modified to report expiration info via `envoy.admin.v3.Certificates`.