Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: November 13, 2016

Y. Sheffer
Intuit

May 12, 2016

Delegating TLS Certificates to a CDN
draft-sheffer-lurk-cert-delegation-00

Abstract

An organization that owns web content often prefers to delegate hosting of this content to a Content Delivery Network (CDN). To serve HTTP content securely, it needs to be protected with TLS. This document proposes a way for the CDN to request constrained certificates so that it can serve web content on behalf of the content owner, without having the owner's long term certificate.

Status of This Memo

This Internet-Draft will expire on November 13, 2016.

Copyright Notice

Table of Contents

1. Introduction

Content owners frequently prefer a Content Delivery Network (CDN) to host their content. CDNs typically have very large networks, and are designed to serve content to a global audience with a high aggregate bandwidth.

To protect this traffic, the CDN uses HTTPS and presents a certificate that usually bears the content owner's name. However, many content owners balk at sharing their long-term private keys with another organization.

This document proposes a way for the CDN to obtain short-term credentials (an end-entity certificate along with the associated private key), allowing the content owner to revoke this authority at short notice.

We note that there are other solutions to this problem:

- The CDN could contact the content owner on each TLS handshake and have the content owner take part in completing the TLS handshake. Such a solution is described in e.g. [I-D.cairns-tls-session-key-interface].

- We could extend ACME [I-D.ietf-acme-acme] by allowing the content owner to share an authorization "ticket" with the CDN, the CDN then using it to obtain short-term certificates directly from the ACME server. This alternative is possibly easier to deploy than the one described in this document, but it would require a non-trivial change to the ACME protocol.

- The current proposal has the content owner generate the certificate's private key, although the best practice would have the CDN generate it and create a Certificate Signing Request (CSR). Note however that it would be difficult for the content owner to validate the correctness of a CSR, potentially allowing a malicious CDN to obtain fraudulent certificates.

1.1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Overview

We define the interaction between the CDN and the content owner, where the CDN requests a short-term certificate periodically, and the content owner obtains it on the CDN's behalf and returns it to the CDN.

We expect the content owner to use the ACME protocol to obtain a short-term certificate, but this is not strictly required by the protocol.

2.1. Advantages

- Compared with solutions that require the CDN to have the content owner sign each handshake, this solution does not require the content owner to set up its own scalable infrastructure.

- Moreover, the need to scale the content owner's web service could result in the content owner ending up by sharing the private keys with the CDN and abdicating its responsibility for its own security.

3. LURK Operations

This section lists the REST APIs that the content owner needs to provide to the CDN.

3.1. Request a Certificate

```
POST /.well-known/lurk/certificate/1234 HTTP/1.1
Content-Type: application/json

{
    "password":"fb2831d6607124286a7b439f2f09793a"
}
```

There is no negotiation of key type (RSA or ECDSA), key length or validity dates, and the client and server must coordinate these details in advance. Similarly, the server MUST be able to determine the FQDN to be included in the certificate based on the authenticated client's identity.

The URI contains a request ID, which MAY be sequential or generated randomly by the client.

The given password MUST be randomly generated and SHOULD have at least 128-bits of entropy.

The server responds with one of:

- A "200 OK" status code, and response body containing a PKCS #12 [RFC7292] structure (private key and certificate), with the content type: "application/x-pkcs12". The structure is protected by the given password.

- A "201 Accepted" status code if the certificate is not yet ready. The CDN should poll the content owner periodically (see below), but not more often than once every 5 seconds.

- Other responses if the request is not acceptable or not allowed.

3.2. Poll for a Certificate

```
GET /.well-known/lurk/certificate/1234 HTTP/1.1
```

The server responds with one of:

- A "200 OK" status code, and response body containing the PKCS #12 response, with the content type: "application/x-pkcs12".

- A "204 No Content" status code if the certificate is not yet ready.

- Other responses if the request is not acceptable or not allowed.

Access to these resources MUST be protected by TLS.

Both requests MUST be authenticated, using one of the following methods:

- Mutual TLS authentication with a client certificate. This is the RECOMMENDED option.
- TLS with preshared secret authentication or TLS-SRP.
- TLS with HTTP-Basic or Digest authentication.

The client cannot assume that the sever will cache the certificate beyond a few seconds after it is first fetched.

4. Security Considerations

This section presents additional considerations beyond those strictly required by the protocol.

4.1. Certificate Details

- It is RECOMMENDED to restrict the certificate's scope as much as possible. Specifically, the certificate request SHOULD specify restrictive Key Usage.
- The certificate SHOULD NOT be for a wildcard DN.
- The RECOMMENDED validity period for certificates provisioned using this mechanism is 3 days, and the certificate SHOULD be valid immediately when it is fetched.

4.2. Revocation

When the content owner decides it no longer trusts the CDN, the content owner MUST:

- Revoke any extant short-term certificates already handed to the CDN. This implies that all such certificates MUST be logged.
- Immediately block the certificate issuance operations described above.

4.3. Restricting CDNs to the Delegation Mechanism

Currently there are no standard methods for the content owner to ensure that the CDN cannot issue a certificate through mechanisms other than the one described here, for the URLs under the CDN's

control. The best solution currently being worked on would consist of several related configuration steps:

- Make sure that the CDN cannot modify the DNS records for the domain. Typically this would mean that the content owner establishes a CNAME resource record from a subdomain into a CDN-managed domain.

- Restrict certificate issuance for the domain to specific CAs that comply with ACME. This assumes universal deployment of CAA [RFC6844] by CAs, which is not the case yet.

- Deploy ACME-specific methods to restrict issuance to a specific authorization key which is controlled by the content owner [I-D.landau-acme-caa].

This solution is recommended in general, even if an alternative to the mechanism described here (e.g. [I-D.cairns-tls-session-key-interface]) is used.

5. References

5.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7292] Moriarty, K., Ed., Nystrom, M., Parkinson, S., Rusch, A., and M. Scott, "PKCS #12: Personal Information Exchange Syntax v1.1", RFC 7292, DOI 10.17487/RFC7292, July 2014, <http://www.rfc-editor.org/info/rfc7292>.

5.2. Informative References

[I-D.cairns-tls-session-key-interface]
Cairns, K., Mattsson, J., Skog, R., and D. Migault, "Session Key Interface (SKI) for TLS and DTLS", draft-cairns-tls-session-key-interface-01 (work in progress), October 2015.

[I-D.ietf-acme-acme]
Barnes, R., Hoffman-Andrews, J., and J. Kasten, "Automatic Certificate Management Environment (ACME)", draft-ietf-acme-acme-02 (work in progress), March 2016.

[I-D.landau-acme-caa]
Landau, H., "ACME Account Key Binding via CAA Records", draft-landau-acme-caa-00 (work in progress), April 2016.

[RFC6844] Hallam-Baker, P. and R. Stradling, "DNS Certification Authority Authorization (CAA) Resource Record", RFC 6844, DOI 10.17487/RFC6844, January 2013, <http://www.rfc-editor.org/info/rfc6844>.

Appendix A. Document History

A.1. draft-sheffer-lurk-cert-delegation-00

Initial version.

Author's Address

Yaron Sheffer
Intuit

EMail: yaronf.ietf@gmail.com