Network Working Group   R. Barnes
Internet-Draft   Mozilla
Intended status: Standards Track   S. Iyengar
Expires: May 3, 2018   Facebook
                                                        N. Sullivan
                                                        Cloudflare
                                                       E. Rescorla
                                                        RTFM, Inc.
                                                   October 30, 2017

Delegated Credentials for TLS
draft-rescorla-tls-subcerts-02

Abstract

   The organizational separation between the operator of a TLS server
   and the certificate authority that provides it credentials can cause
   problems, for example when it comes to reducing the lifetime of
   certificates or supporting new cryptographic algorithms. This
   document describes a mechanism to allow TLS server operators to
   create their own credential delegations without breaking
   compatibility with clients that do not support this specification.

Status of This Memo

   This Internet-Draft is submitted in full conformance with the
   provisions of BCP 78 and BCP 79.

   Internet-Drafts are working documents of the Internet Engineering
   Task Force (IETF). Note that other groups may also distribute
   working documents as Internet-Drafts. The list of current Internet-
   Drafts is at https://datatracker.ietf.org/drafts/current/.

   Internet-Drafts are draft documents valid for a maximum of six months
   and may be updated, replaced, or obsoleted by other documents at any
   time. It is inappropriate to use Internet-Drafts as reference
   material or to cite them other than as "work in progress."

   This Internet-Draft will expire on May 3, 2018.

Table of Contents

1. Introduction

Typically, a TLS server uses a certificate provided by some entity
other than the operator of the server (a "Certification Authority" or
CA) [RFC5246] [RFC5280]. This organizational separation makes the
TLS server operator dependent on the CA for some aspects of its
operations, for example:

o Whenever the server operator wants to deploy a new certificate, it
   has to interact with the CA.

o The server operator can only use TLS authentication schemes for
   which the CA will issue credentials.

These dependencies cause problems in practice. Server operators
often want to create short-lived certificates for servers in low-
trust zones such as CDNs or remote data centers. This allows server
operators to limit the exposure of keys in cases that they do not
realize a compromise has occurred. The risk inherent in cross-
organizational transactions makes it operationally infeasible to rely

on an external CA for such short-lived credentials. In contrast to
OCSP stapling, in which an operator could choose to talk to the CA
frequently to obtain stapled responses, failure to fetch an OCSP
stapled response results only in degraded performance, however
failure to fetch a potentially large number of short lived
certificates would result in the service not being available which
creates greater operational risk.

To remove these dependencies, this document proposes a limited
delegation mechanism that allows a TLS server operator to issue its
own credentials within the scope of a certificate issued by an
external CA. Because the above problems do not relate to the CAs
inherent function of validating possession of names, it is safe to
make such delegations as long as they only enable the recipient of
the delegation to speak for names that the CA has authorized. For
clarity, we will refer to the certificate issued by the CA as a
"certificate" and the one issued by the operator as a "Delegated
credential".

2. Solution Overview

A Delegated credential is a digitally signed data structure with the
following semantic fields:

o A validity interval

o A public key (with its associated algorithm)

The signature on the credential indicates a delegation from the
certificate which is issued to the TLS server operator. The key pair
used to sign a credential is presumed to be one whose public key is
contained in an X.509 certificate that associates one or more names
to the credential.

A TLS handshake that uses credentials differs from a normal handshake
in a few important ways:

o The client provides an extension in its ClientHello that indicates
   support for this mechanism.

o The server provides both the certificate chain terminating in its
   certificate as well as the credential.

o The client uses information in the server's certificate to verify
   the signature on the credential and verify that the server is
   asserting an expected identity.

o The client uses the public key in the credential as the server's
   working key for the TLS handshake.

Delegated credentials can be used either in TLS 1.3 or TLS 1.2.
Differences between the use of Delegated credentials in the protocols
are explicitly stated.

It was noted in [XPROT] that certificates in use by servers that
support outdated protocols such as SSLv2 can be used to forge
signatures for certificates that contain the keyEncipherment KeyUsage
([RFC5280] section 4.2.1.3) In order to prevent this type of cross-
protocol attack, we define a new DelegationUsage extension to X.509
which permits use of delegated credentials. Clients MUST NOT accept
delegated credentials associated with certificates without this
extension.

Credentials allow the server to terminate TLS connections on behalf
of the certificate owner. If a credential is stolen, there is no
mechanism for revoking it without revoking the certificate itself.
To limit the exposure of a delegation credential compromise, servers
MUST NOT issue credentials with a validity period longer than 7 days.
Clients MUST NOT accept credentials with longer validity periods.

2.1. Rationale

Delegated credentials present a better alternative from other
delegation mechanisms like proxy certificates [RFC3820] for several
reasons:

o There is no change needed to certificate validation at the PKI
   layer.

o X.509 semantics are very rich. This can cause unintended
   consequences if a service owner creates a proxy cert where the
   properties differ from the leaf certificate.

o Delegated credentials have very restricted semantics which should
   not conflict with X.509 semantics.

o Proxy certificates rely on the certificate path building process
   to establish a binding between the proxy certificate and the
   server certificate. Since the cert path building process is not
   cryptographically protected, it is possible that a proxy
   certificate could be bound to another certificate with the same
   public key, with different X.509 parameters. Delegated
   credentials, which rely on a cryptographic binding between the
   entire certificate and the Delegated credential, cannot.

o Delegated credentials allow signed messages to be bound to
   specific versions of TLS. This prevents them from being used for
   other protocols if a service owner allows multiple versions of
   TLS.

2.2. Related Work

Many of the use cases for delegated credentials can also be addressed
using purely server-side mechanisms that do not require changes to
client behavior (e.g., LURK [I-D.mglt-lurk-tls-requirements]). These
mechanisms, however, incur per-transaction latency, since the front-
end server has to interact with a back-end server that holds a
private key. The mechanism proposed in this document allows the
delegation to be done off-line, with no per-transaction latency. The
figure below compares the message flows for these two mechanisms with
TLS 1.3 [I-D.ietf-tls-tls13].

LURK:

```
Client              Front-End              Back-End
   |----ClientHello--->|                       |
   |<---ServerHello----|                       |
   |<---Certificate----|                       |
   |                   |<-------LURK------->|
   |<---CertVerify-----|                       |
   |        ...        |                       |
```

Delegated credentials:

```
Client              Front-End              Back-End
   |                   |<---Cred Provision--|
   |----ClientHello--->|                       |
   |<---ServerHello----|                       |
   |<---Certificate----|                       |
   |<---CertVerify-----|                       |
```

These two classes of mechanism can be complementary. A server could
use credentials for clients that support them, while using LURK to
support legacy clients.

It is possible to address the short-lived certificate concerns above
by automating certificate issuance, e.g., with ACME
[I-D.ietf-acme-acme]. In addition to requiring frequent
operationally-critical interactions with an external party, this
makes the server operator dependent on the CA's willingness to issue
certificates with sufficiently short lifetimes. It also fails to
address the issues with algorithm support. Nonetheless, existing

automated issuance APIs like ACME may be useful for provisioning
credentials, within an operator network.

3. Client and Server behavior

This document defines the following extension code point.

```
enum {
  ...
  delegated_credential(TBD),
  (65535)
} ExtensionType;
```

A client which supports this document SHALL send an empty
"delegated_credential" extension in its ClientHello.

If the extension is present, the server MAY send a
DelegatedCredential extension. If the extension is not present, the
server MUST NOT send a credential. A credential MUST NOT be provided
unless a Certificate message is also sent.

When negotiating TLS 1.3, and using Delegated credentials, the server
MUST send the DelegatedCredential as an extension in the
CertificateEntry of its end entity certificate. When negotiating TLS
1.2, the DelegatedCredential MUST be sent as an extension in the
ServerHello.

The DelegatedCredential contains a signature from the public key in
the end-entity certificate using a signature algorithm advertised by
the client in the "signature_algorithms" extension. Additionally,
the credential's public key MUST be of a type that enables at least
one of the supported signature algorithms. A Delegated credential
MUST NOT be negotiated by the server if its signature is not
compatible with any of the supported signature algorithms or the
credential's public key is not usable with the supported signature
algorithms of the client, even if the client advertises support for
delegated credentials.

On receiving a credential and a certificate chain, the client
validates the certificate chain and matches the end-entity
certificate to the server's expected identity following its normal
procedures. It then takes the following additional steps:

o Verify that the current time is within the validity interval of
   the credential.

o Use the public key in the server's end-entity certificate to
   verify the signature on the credential.

   o Use the public key in the credential to verify a signature
      provided in the handshake. That is the CertificateVerify message
      in TLS 1.3 or ServerKeyExchange in 1.2.

   o Verify that the certificate has the correct extensions that allow
      the use of Delegated credentials.

   Clients that receive Delegated credentials that are valid for more
   than 7 days MUST terminate the connection with an "illegal_parameter"
   alert.

4. Delegated Credentials

   While X.509 forbids end-entity certificates from being used as
   issuers for other certificates, it is perfectly fine to use them to
   issue other signed objects as long as the certificate contains the
   digitalSignature key usage (RFC5280 section 4.2.1.3). We define a
   new signed object format that would encode only the semantics that
   are needed for this application.

```
struct {
  uint32 validTime;
  opaque publicKey<0..2^16-1>;
} DelegatedCredentialParams;

struct {
  DelegatedCredentialParams cred;
  SignatureScheme scheme;
  opaque signature<0..2^16-1>;
} DelegatedCredential;
```

   validTime: Relative time in seconds from the beginning of the
      certificate's notBefore value after which the Delegated Credential
      is no longer valid.

   publicKey: The Delegated Credential's public key which is an encoded
      SubjectPublicKeyInfo [RFC5280].

   scheme: The Signature algorithm and scheme used to sign the
      Delegated credential.

   signature: The signature over the credential with the end-entity
      certificate's public key, using the scheme.

   The DelegatedCredential structure is similar to the CertificateVerify
   structure in TLS 1.3. Since the SignatureScheme defined in TLS 1.3,
   TLS 1.2 clients should translate the scheme into an appropriate group
   and signature algorithm to perform validation.

The signature of the DelegatedCredential is computed over the concatenation of:

1. A string that consists of octet 32 (0x20) repeated 64 times.

2. The context string "TLS, server delegated credentials".

3. A single 0 byte which serves as the separator

4. Big endian serialized 2 bytes ProtocolVersion of the negotiated TLS version, defined by TLS.

5. DER encoded X.509 certificate used to sign the DelegatedCredential.

6. Big endian serialized 2 byte SignatureScheme scheme.

7. The DelegatedCredentialParams structure.

This signature has a few desirable properties:

o It is bound to the certificate that signed it.

o It is bound to the protocol version that is negotiated. This is intended to avoid cross-protocol attacks with signing oracles.

The code changes to create and verify Delegated credentials would be localized to the TLS stack, which has the advantage of avoiding changes to security-critical and often delicate PKI code (though of course moves that complexity to the TLS stack).

4.1. Certificate Requirements

We define a new X.509 extension, DelegationUsage to be used in the certificate when the certificate permits the usage of Delegated Credentials. When this extension is not present the client MUST not accept a Delegated Credential even if it is negotiated by the server. When it is present, the client MUST follow the validation procedure.

id-ce-delegationUsage OBJECT IDENTIFIER ::= { TBD }

DelegationUsage ::= BIT STRING { allowed (0) }

Conforming CAs MUST mark this extension as non-critical. This would allow the certificate to be used by service owners for clients that do not support certificate delegation as well and not need to obtain two certificates.

5. IANA Considerations

   TBD

6. Security Considerations

6.1. Security of delegated private key

   Delegated credentials limit the exposure of the TLS private key by
   limiting its validity. An attacker who compromises the private key
   of a delegated credential can act as a man in the middle until the
   delegate credential expires, however they cannot create new delegated
   credentials. Thus delegated credentials should not be used to send a
   delegation to an untrusted party, but is meant to be used between
   parties that have some trust relationship with each other. The
   secrecy of the delegated private key is thus important and several
   access control mechanisms SHOULD be used to protect it such as file
   system controls, physical security or hardware security modules.

6.2. Revocation of delegated credentials

   Delegated credentials do not provide any additional form of early
   revocation. Since it is short lived, the expiry of the delegated
   credential would revoke the credential. Revocation of the long term
   private key that signs the delegated credential also implictly
   revokes the delegated credential.

6.3. Privacy considerations

   Delegated credentials can be valid for 7 days and it is much easier
   for a service to create delegated credential than a certificate
   signed by a CA. A service could determine the client time and clock
   skew by creating several delegated credentials with different expiry
   timestamps and observing whether the client would accept it. Client
   time could be unique and thus privacy sensitive clients, such as
   browsers in incognito mode, who do not trust the service might not
   want to advertise support for delegated credentials or limit the
   number of probes that a server can perform.

7. Acknowledgements

   Thanks to Kyle Nekritz, Anirudh Ramachandran, Benjamin Kaduk, Kazuho
   Oku, Daniel Kahn Gillmor for their discussions, ideas, and bugs
   they've found.

8. References

8.1. Normative References

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S.,
              Housley, R., and W. Polk, "Internet X.509 Public Key
              Infrastructure Certificate and Certificate Revocation List
              (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008,
              <https://www.rfc-editor.org/info/rfc5280>.

8.2. Informative References

   [I-D.ietf-acme-acme]
              Barnes, R., Hoffman-Andrews, J., and J. Kasten, "Automatic
              Certificate Management Environment (ACME)", draft-ietf-
              acme-acme-07 (work in progress), June 2017.

   [I-D.ietf-tls-tls13]
              Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", draft-ietf-tls-tls13-21 (work in progress),
              July 2017.

   [I-D.mglt-lurk-tls-requirements]
              Migault, D. and K. Ma, "Authentication Model and Security
              Requirements for the TLS/DTLS Content Provider Edge Server
              Split Use Case", draft-mglt-lurk-tls-requirements-00 (work
              in progress), January 2016.

   [RFC3820] Tuecke, S., Welch, V., Engert, D., Pearlman, L., and M.
              Thompson, "Internet X.509 Public Key Infrastructure (PKI)
              Proxy Certificate Profile", RFC 3820,
              DOI 10.17487/RFC3820, June 2004,
              <https://www.rfc-editor.org/info/rfc3820>.

   [XPROT]   Jager, T., Schwenk, J., and J. Somorovsky, "On the
              Security of TLS 1.3 and QUIC Against Weaknesses in PKCS#1
              v1.5 Encryption", Proceedings of the 22nd ACM SIGSAC
              Conference on Computer and Communications Security , 2015.

Authors' Addresses

   Richard Barnes
   Mozilla

   Email: rlb@ipv.sx

   Subodh Iyengar
   Facebook

   Email: subodh@fb.com

   Nick Sullivan
   Cloudflare

   Email: nick@cloudflare.com

   Eric Rescorla
   RTFM, Inc.

   Email: ekr@rtfm.com