Bootstrapped TLS Authentication
draft-friel-tls-eap-dpp-05

Abstract

   This document defines a TLS extension that enables a server to prove
   to a client that it has knowledge of the public key of a key pair
   where the client has knowledge of the private key of the key pair.
   Unlike standard TLS key exchanges, the public key is never exchanged
   in TLS protocol messages. Proof of knowledge of the public key is
   used by the client to bootstrap trust in the server. The use case
   outlined in this document is to establish trust in an EAP server.

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 27 November 2022.

Copyright Notice

Table of Contents

[1](#). Introduction

   On-boarding of devices with no, or limited, user interface can be
   difficult. Typically, a credential is needed to access the network
   and network connectivity is needed to obtain a credential. This
   poses a catch-22.

   If trust in the integrity of a device's public key can be obtained in
   an out-of-band fashion, a device can be authenticated and provisioned
   with a usable credential for network access. While this
   authentication can be strong, the device's authentication of the
   network is somewhat weaker. [[duckling](#)] presents a functional
   security model to address this asymmetry.

   There are on-boarding protocols, such as [[DPP](#)], to address this use
   case but they have drawbacks. [[DPP](#)] for instance does not support
   wired network access. This document describes an on-boarding
   protocol, which we refer to as TLS Proof of Knowledge or TLS-POK.

1.1. Terminology

The following terminology is used throughout this document.

*   BSK: Bootstrap Key which is an elliptic curve public private key
    pair.

*   DPP: Device Provisioning Protocol

*   EPSK: External Pre-Shared Key

*   PSK: Pre-Shared Key

1.2. Bootstrap Key Pair

The mechanism for on-boarding of devices defined in this document
relies on bootstrap key pairs. A client device has an associated
elliptic curve (EC) bootstrap key pair (BSK). The BSK may be static
and baked into device firmware at manufacturing time, or may be
dynamic and generated at on-boarding time by the device. If the BSK
public key, specifically the ASN.1 SEQUENCE SubjectPublicKeyInfo from
[RFC5280], can be shared in a trustworthy manner with a TLS server, a
form of "origin entity authentication" (the step from which all
subsequent authentication proceeds) can be obtained.

The exact mechanism by which the server gains knowledge of the BSK
public key is out of scope of this specification, but possible
mechanisms include scanning a QR code to obtain a base64 encoding of
the ASN.1-formatted public key or uploading of a Bill of Materials
(BOM) which includes the public key. If the QR code is physically
attached to the client device, or the BOM is associated with the
device, the assumption is that the public key obtained in this
bootstrapping method belongs to the client. In this model, physical
possession of the device implies legitimate ownership.

The server may have knowledge of multiple BSK public keys
corresponding to multiple devices, and existing TLS mechanisms are
leveraged that enable the server to identity a specific bootstrap
public key corresponding to a specific device.

Using the process defined herein, the client proves to the server
that it has possession of the private analog to its public
bootstrapping key. Provided that the mechanism in which the server
obtained the BSK public key is trustworthy, a commensurate amount of
authenticity of the resulting connection can be obtained. The server
also proves that it knows the client's public key which, if the
client does not gratuitously expose its public key, can be used to
obtain a modicum of correctness, that the client is connecting to the
correct network (see [duckling]).

1.3. Alignment with Wi-Fi Alliance Device Provisioning Profile

The definition of the BSK public key aligns with that given in [DPP].
This, for example, enables the QR code format as defined in [DPP] to
be reused for TLS-POK. Therefore, a device that supports both wired
LAN and Wi-Fi LAN connections can have a single QR code printed on
its label, and the bootstrap key can be used for DPP if the device
bootstraps against a Wi-Fi network, or TLS-POK if the device
bootstraps against a wired network. Similarly, a common bootstrap
public key format could be imported in a BOM into a server that
handles devices connecting over both wired and Wi-Fi networks.

Any bootstrapping method defined for, or used by, [DPP] is compatible
with TLS-POK.

2. Bootstrapping in TLS 1.3

Bootstrapping in TLS 1.3 leverages Certificate-Based Authentication
with an External Pre-Shared Key [RFC8773]. The External PSK (EPSK)
is derived from the BSK public key, and the EPSK is imported using
[I-D.ietf-tls-external-psk-importer]. This BSK MUST be from a
cryptosystem suitable for doing ECDSA.

The TLS PSK handshake gives the client proof that the server knows
the BSK public key. Certificate based authentication of the client
by the server is carried out using the BSK, giving the server proof
that the client knows the BSK private key. This satisfies the proof
of ownership requirements outlined in Section 1.

2.1. External PSK Derivation

An [I-D.ietf-tls-external-psk-importer] EPSK is made of of the tuple
of (Base Key, External Identity, Hash). The EPSK is derived from the
BSK public key using [RFC5869] with the hash algorithm from the
ciphersuite:

```
epsk   = HKDF-Expand(HKDF-Extract(<>, bskey),
                      "tls13-imported-bsk", L)
epskid = HKDF-Expand(HKDF-Extract(<>, bskey),
                      "tls13-bspsk-identity", L)
```
where:
  - epsk is the EPSK Base Key
  - epskid is the EPSK External Identity
  - <> is a NULL salt
  - bskey is the DER-encoded ASN.1 subjectPublicKeyInfo
    representation of the BSK public key
  - L is the length of the digest of the underlying hash
    algorithm

The [I-D.ietf-tls-external-psk-importer] ImportedIdentity structure
is defined as:

```
struct {
   opaque external_identity<1...2^16-1>;
   opaque context<0..2^16-1>;
   uint16 target_protocol;
   uint16 target_kdf;
} ImportedIdentity;
```

and is created using the following values:

```
external_identity = epskid
context = "tls13-bsk"
target_protocol = TLS1.3(0x0304)
target_kdf = HKDF_SHA256(0x0001)
```

The EPSK and ImportedIdentity are used in the TLS handshake as
specified in [I-D.ietf-tls-external-psk-importer].

A performance versus storage tradeoff a server can choose is to
precompute the identity of every bootstrapped key with every hash
algorithm that it uses in TLS and use that to quickly lookup the
bootstrap key and generate the PSK. Servers that choose not to
employ this optimization will have to do a runtime check with every
bootstrap key it holds against the identity the client provides.

2.2. Changes to TLS 1.3 Handshake

The client includes the "tls_cert_with_extern_psk" extension in the
ClientHello, per [RFC8773]. The client identifies the BSK by
inserting the serialized content of ImportedIdentity into the
PskIdentity.identity in the PSK extension, per
[I-D.ietf-tls-external-psk-importer]. The server looks up the
client's EPSK key in its database using the mechanisms documented in

[I-D.ietf-tls-external-psk-importer]. If no match is found, the
server SHALL terminate the TLS handshake with an alert.

If the server found the matching BSK, it includes the
"tls_cert_with_extern_psk" extension in the ServerHello message, and
the corresponding EPSK identity in the "pre_shared_key" extension.
When these extensions have been successfully negotiated, the TLS 1.3
key schedule SHALL include both the EPSK in the Early Secret
derivation and an (EC)DHE shared secret value in the Handshake Secret
derivation.

After successful negotiation of these extensions, the full TLS 1.3
handshake is performed with the additional caveat that the client
authenticates with a raw public key (its BSK) per [RFC7250]. The BSK
is always an elliptic curve key pair, therefore the
ClientCertTypeExtension SHALL always indicate RawPublicKey and the
type of the client's Certificate SHALL be ECDSA and contain the
client's BSK public key as a DER-encoded ASN.1 subjectPublicKeyInfo
SEQUENCE.

When the server processes the client's Certificate it MUST ensure
that it is identical to the BSK public key that it used to generate
the EPSK and ImportedIdentity for this handshake.

When clients use the [duckling] form of authentication, they MAY
forgo the checking of the server's certificate in the
CertificateVerify and rely on the integrity of the bootstrapping
method employed to distribute its key in order to validate trust in
the authenticated TLS connection.

The handshake is shown in Figure 1.

```
         Client                                          Server
         --------                                        --------
         ClientHello
         + cert_with_extern_psk
         + client_cert_type=RawPublicKey
         + key_share
         + pre_shared_key         -------->
                                                     ServerHello
                                          + cert_with_extern_psk
                                    + client_cert_type=RawPublicKey
                                                     + key_share
                                                 + pre_shared_key
                                            {EncryptedExtensions}
                                             {CertificateRequest}
                                                    {Certificate}
                                              {CertificateVerify}
                                  <--------             {Finished}
         {Certificate}
         {CertificateVerify}
         {Finished}               -------->
         [Application Data]       <------->    [Application Data]
```

                 Figure 1: TLS 1.3 TLS-POK Handshake

3. Using TLS Bootstrapping in EAP

   Enterprise deployments typically require an 802.1X/EAP-based
   authentication to obtain network access. Protocols like [RFC7030]
   can be used to enroll devices into a Certification Authority to allow
   them to authenticate using 802.1X/EAP. But this creates a Catch-22
   where a certificate is needed for network access and network access
   is needed to obtain certificate.

   Devices whose bootstrapping key can been obtained in an out-of-band
   fashion can perform an EAP-TLS-based exchange, for instance
   [RFC7170], and authenticate the TLS exchange using the bootstrapping
   extensions defined in Section 2. This network connectivity can then
   be used to perform an enrollment protocol (such as provided by
   [RFC7170]) to obtain a credential for subsequent network connectivity
   and certificate lifecycle maintenance.

   Upon "link up", an Authenticator on an 802.1X-protected port will
   issue an EAP Identify request to the newly connected peer. For
   unprovisioned devices that desire to take advantage of TLS-POK, there
   is no initial realm in which to construct an NAI (see [RFC4282]) so
   the initial EAP Identity response SHOULD contain simply the name
   "TLS-POK" in order to indicate to the Authenticator that an EAP
   method that supports TLS-POK SHOULD be started.

```
      Authenticating Peer      Authenticator
      -------------------      -------------
                                <- EAP-Request/
                                Identity

       EAP-Response/
       Identity (TLS-POK) ->

                                <- EAP-Request/
                                EAP-Type=TEAP
                               (TLS Start)
                         .
                         .
                         .
```

[4](#). Summary of Work

The protocol outlined here can be broadly broken up into 4 distinct
areas:

*   TLS extensions to transport the bootstrap public key identifier

*   Use of the TLS 1.3 extension for certificate-based authentication
    with an external PSK

*   The client's use of a raw public key in its certificate

*   TEAP extensions to leverage the new TLS-POK handshake for trust
    establishment

This document captures all 4 areas.

[5](#). IANA Considerations

None.

[6](#). Security Considerations

Bootstrap and trust establishment by the TLS server is based on proof
of knowledge of the client's bootstrap public key, a non-public
datum. The TLS server obtains proof that the client knows its
bootstrap public key and, in addition, also possesses its
corresponding private analog.

Trust on the part of the client is based on validation of the server
certificate and the TLS 1.3 handshake. In addition, the client
assumes that knowledge of its public bootstrapping key is not widely
disseminated and therefore any device that proves knowledge of its

bootstrapping key is the appropriate device from which to receive
provisioning, for instance via [RFC7170]. [duckling] describes a
security model for this type of "imprinting".

An attack on the bootstrapping method which substitutes the public
key of a corrupted device for the public key of an honest device can
result in the TLS sever on-boarding and trusting the corrupted
device.

If an adversary has knowledge of the bootstrap public key, the
adversary may be able to make the client bootstrap against the
adversary's network. For example, if an adversary intercepts and
scans QR labels on clients, and the adversary can force the client to
connect to its server, then the adversary can complete the TLS-POK
handshake with the client and the client will connect to the
adversary's server. Since physical possession implies ownership,
there is nothing to prevent a stolen device from being on-boarded.

7. References

7.1. Normative References

[I-D.ietf-tls-external-psk-importer]
          Benjamin, D. and C. A. Wood, "Importing External PSKs for
          TLS", Work in Progress, Internet-Draft, draft-ietf-tls-
          external-psk-importer-08, 22 April 2022,
          <https://www.ietf.org/archive/id/draft-ietf-tls-external-
          psk-importer-08.txt>.

[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>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand
          Key Derivation Function (HKDF)", RFC 5869,
          DOI 10.17487/RFC5869, May 2010,
          <https://www.rfc-editor.org/info/rfc5869>.

[RFC7250] Wouters, P., Ed., Tschofenig, H., Ed., Gilmore, J.,
          Weiler, S., and T. Kivinen, "Using Raw Public Keys in
          Transport Layer Security (TLS) and Datagram Transport
          Layer Security (DTLS)", RFC 7250, DOI 10.17487/RFC7250,
          June 2014, <https://www.rfc-editor.org/info/rfc7250>.

   [RFC8773] Housley, R., "TLS 1.3 Extension for Certificate-Based
              Authentication with an External Pre-Shared Key", RFC 8773,
              DOI 10.17487/RFC8773, March 2020,
              <https://www.rfc-editor.org/info/rfc8773>.

7.2. Informative References

   [DPP]   Wi-Fi Alliance, "Device Provisioning Profile", 2020.

   [duckling] Stajano, F. and E. Rescorla, "The Ressurecting Ducking:
              Security Issues for Ad-Hoc Wireless Networks", 1999.

   [RFC4282] Aboba, B., Beadles, M., Arkko, J., and P. Eronen, "The
              Network Access Identifier", RFC 4282,
              DOI 10.17487/RFC4282, December 2005,
              <https://www.rfc-editor.org/info/rfc4282>.

   [RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed.,
              "Enrollment over Secure Transport", RFC 7030,
              DOI 10.17487/RFC7030, October 2013,
              <https://www.rfc-editor.org/info/rfc7030>.

   [RFC7170] Zhou, H., Cam-Winget, N., Salowey, J., and S. Hanna,
              "Tunnel Extensible Authentication Protocol (TEAP) Version
              1", RFC 7170, DOI 10.17487/RFC7170, May 2014,
              <https://www.rfc-editor.org/info/rfc7170>.

Authors' Addresses

   Owen Friel
   Cisco
   Email: ofriel@cisco.com

   Dan Harkins
   Hewlett-Packard Enterprise
   Email: daniel.harkins@hpe.com