Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 14 August 2023
10 February 2023

O. Friel
Cisco
D. Harkins
Hewlett-Packard Enterprise

Bootstrapped TLS Authentication with Proof of Knowledge (TLS-POK)
draft-ietf-emu-bootstrapped-tls-02

Abstract

This document defines a mechanism that enables a bootstrapping device to establish trust and mutually authenticate against a network. Bootstrapping devices have a public private key pair, and this mechanism enables a network server to prove to the device that it knows the public key, and the device to prove to the server that it knows the private key. The mechanism leverages existing DPP and TLS standards and can be used in an EAP exchange.

Status of This Memo

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 a device has a public / private keypair, and 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.

Device on-boarding protocols such as the Device Provisioning Profile [DPP], also referred to as Wi-Fi Easy Connect, address this use case but they have drawbacks. [DPP] for instance does not support wired network access, and does not specify how the device's DPP keypair can

be used in a TLS handshake. This document describes an on-boarding protocol that can be used for wired network access, which we refer to as TLS Proof of Knowledge or TLS-POK.

This document does not address the problem of Wi-Fi network discovery, where a bootstrapping device detects multiple different Wi-Fi networks and needs a more robust and scalable mechanism than simple round-robin to determine the correct network to attach to. DPP addresses this issue. Thus, the intention is that DPP is the recommended mechanism for bootstrapping against Wi-Fi networks, and TLS-POK is the recommended mechanism for bootstrapping against wired networks.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The following terminology is used throughout this document.

* 802.1X: IEEE Port-Based Network Access Control

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

* DPP: Device Provisioning Protocol [DPP]

* EAP: Extensible Authentication Protocol [RFC3748]

* EPSK: External Pre-Shared Key

* EST: Enrollment over Secure Transport [RFC7030]

* PSK: Pre-Shared Key

* TEAP: Tunnel Extensible Authentication Protocol [RFC7170]

1.2. Bootstrapping Overview

A bootstrapping device holds a public / private key pair which we refer to as a Bootstrap Key (BSK). The private key of the BSK is known only by the device. The public key of the BSK is known by the device, is known by the owner or holder of the device, and is provisioned on the network by the network operator. In order to establish trust and mutually authenticate, the network proves to the

device that it knows the public part of the BSK, and the device proves to the network that it knows the private part of the BSK. Once this trust has been established during bootstrapping, the network can provision the device with a credential that it uses for subsequent network access. More details on the BSK are given in Section 2.

1.3. EAP Network Access

Enterprise deployments typically require an [IEEE802.1X]/EAP-based authentication to obtain network access. Protocols like Enrollment over Secure Transport (EST) [RFC7030] can be used to enroll devices into a Certification Authority to allow them to authenticate using 802.1X/EAP. This creates a Catch-22 where a certificate is needed for network access and network access is needed to obtain certificate.

Devices whose BSK public key can been obtained in an out-of-band fashion and provisioned on the network can perform an EAP-TLS-based exchange, for instance Tunnel Extensible Authentication Protocol (TEAP) [RFC7170], and authenticate the TLS exchange using the bootstrapping mechanisms defined in Section 3. 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.

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 key of its BSK. 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 BSK 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]).

2.1. 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, or dynamically display a single QR code on a display, 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 into 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.

3. 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 [RFC9258]. This BSK MUST be from a cryptosystem suitable for doing ECDSA. As the BSK is an ASN.1 SEQUENCE SubjectPublicKeyInfo, the client presents a raw public key certificate as specified in Using Raw Public Keys in TLS and DTLS [RFC7250].

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.

3.1. External PSK Derivation

An [RFC9258] EPSK is made up 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 [RFC9258] 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 [RFC9258].

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.

3.2. TLS 1.3 Handshake Details

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 [RFC9258]. The client MUST also include the [RFC7250] "client_certificate_type" extension in the ClientHello and MUST specify type of RawPublicKey.

Upon receipt of the ClientHello, the server looks up the client's EPSK key in its database using the mechanisms documented in [RFC9258]. The ImportedIdentity context value of "tls13-bsk" informs the server that the mechanisms specified in this document for deriving the EPSK and executing the TLS handshake MUST be used. If no match is found, the server MUST 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 MUST 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 server MUST send a CertificateRequest message and client MUST authenticate with a raw public key (its BSK) per [RFC7250]. The BSK is always an elliptic curve key pair, therefore the type of the client's Certificate MUST be ECDSA and MUST contain the client's BSK public key as a DER-encoded ASN.1 subjectPublicKeyInfo SEQUENCE.

Note that the client MUST NOT share its BSK public key with the server until after the client has completed processing of the ServerHello and verified the TLS key schedule. The PSK proof has completed at this stage, and the server has proven to the client that is knows the BSK public key, and it is therefore safe for the client to send the BSK public key to the server in the Certificate message. If the PSK verification step fails when processing the ServerHello, the client terminates the TLS handshake and the BSK public key MUST NOT be shared with the server.

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

4. Using TLS Bootstrapping in EAP

Upon "link up", an Authenticator on an 802.1X-protected port will issue an EAP Identity 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@eap-dpp.arpa" 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-dpp.arpa) ->

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

       EAP-Response/
       EAP-Type=TEAP
       (TLS client_hello with
        tls_cert_with_extern_psk
        and pre_shared_key) ->

                          .
                          .
                          .
```

Both client and server have derived the EPSK and associated [RFC9258] ImportedIdentity from the BSK as described in Section 3.1. When the client starts the TLS exchange in the EAP transaction, it includes the ImportedIdentity structure in the pre_shared_key extension in the ClientHello. When the server received the ClientHello, it extracts the ImportedIdentity and looks up the EPSK and BSK public key. As previously mentioned in Section 2, the exact mechanism by which the server has gained knowledge of or been provisioned with the BSK public key is outside the scope of this document.

The server continues with the TLS handshake and uses the EPSK to prove that it knows the BSK public key. When the client replies with its Certificate, CertificateVerify and Finished messages, the server MUST ensure that the public key in the Certificate message matches the BSK public key.

Once the TLS handshake completes, the client and server have established mutual trust. The server can then proceed to provision a credential onto the client using, for example, the mechanisms outlined in [RFC7170].

The client can then use this provisioned credential for subsequent network authentication. The BSK is only used during bootstrap, and it not used for any subsequent network access.

5. IANA Considerations

This document registers the following arpa domain:

eap-dpp.arpa

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 key.

Trust on the part of the client is based on successful completion of the TLS 1.3 handshake using the EPSK derived from the BSK. This proves to the client that the server knows its BSK public key. In addition, the client assumes that knowledge of its BSK public key is not widely disseminated and therefore any server that proves knowledge of its BSK public key is the appropriate server 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

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

[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>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[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>.

[RFC9258] Benjamin, D. and C. A. Wood, "Importing External Pre-Shared Keys (PSKs) for TLS 1.3", RFC 9258, DOI 10.17487/RFC9258, July 2022, <https://www.rfc-editor.org/info/rfc9258>.

7.2. Informative References

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

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

[IEEE802.1X] IEEE, "Port-Based Network Access Control", 2010.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <https://www.rfc-editor.org/info/rfc3748>.

[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