Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 10, 2022

O. Friel
Cisco
D. Harkins
Hewlett-Packard Enterprise

July 09, 2021

Bootstrapped TLS Authentication
draft-friel-tls-eap-dpp-03

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.

Table of Contents

1. Introduction 2
1.1. Bootstrap Key Pair 2
1.2. Alignment with Wi-Fi Alliance Device Provisioning Profile 3
2. Bootstrapping in TLS 1.3 4
2.1. Bootstrap Extended PSK 4
2.2. Changes to TLS 1.3 Handshake 5
3. Using TLS Bootstrapping in EAP 6
4. Summary of Work 7
5. IANA Considerations 7
6. Security Considerations 7
7. References 8
7.1. Normative References 8
7.2. Informative References 9
Authors' Addresses 9

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. 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) key pair. The key pair 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 this 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 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 upload of a Bill of Materials (BOM). 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 bootstrap public keys corresponding to multiple devices, and TLS extensions are defined in this document that enable the server to identity a specific bootstrap public key correspinding 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 bootstrapping 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]).

1.2. Alignment with Wi-Fi Alliance Device Provisioning Profile

The definition of the boostrap 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

The bootstrapping modifications introduce an extension to identify a "bootstrapping" key which is converted into an external PSK and used directly in the TLS 1.3 handshake. This key MUST be from a cryptosystem suitable for doing ECDSA.

2.1. Bootstrap Extended PSK

This document defines the "bskey" extended PSK type by expanding on the work in [extensible-psks].

```
 enum {
   bskey(TBD), (255)
 } ExtendedPskIdentityType;
```

A bskey PSK is a varient of an external PSK which, in this case, is derived from a public key.

The PSKIdentity of a bskey extended PSK is encoded with a string derived from the DER-encoded ASN.1 subjectPublicKeyInfo representation of the bootstrapping public key.

```
 struct {
   opaque identity<1..2^32-1>
 } BootstrapPSKIdentity;
```

Both the bskey PSK and the BootstrapPSKIdentity are computed using [RFC5869] with the hash algorithm from the ciphersuite:

```
bskeypsk = HKDF-Expand(HKDF-Extract(<>, bskey),
                       "tls13-extended-psk-bskey", L)
identity = HKDF-Expand(HKDF-Extract(<>, bskey),
                       "tls13-psk-identity-bskey", L)
```

where:

- <> is a NULL salt
- bskey is the DER-encoded ASN.1 subjectPublicKeyInfo representation of the bootstrapping key
- L is the length of the digest of the underlying hash algorithm

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], and identifies the bootstrapping key using the BootstrapPSKIdentity extension. The server looks up the client's bootstrapping key in its database by checking the hash of each entry with the value received in the ClientHello. If no match is found, the server SHALL terminate the TLS handshake with an alert.

[[TODO: should we define an explicit unknown_bsk_identity alert, similar to unknown_psk_identity]]

If the server found the matching bootstrap key, it generates the bskeypsk and includes the "tls_cert_with_extern_psk" extension in the ServerHello message. When these extensions have been successfully negotiated, the TLS 1.3 key schedule SHALL include both the bskeypsk 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 bootstrapping key) per [RFC7250]. The bootstrapping key is always an elliptic curve public key, therefore the ClientCertTypeExtension SHALL always indicate RawPublicKey and the type of the client's Certificate SHALL be ECDSA and contain the client's bootstrapping key as a DER-encoded ASN.1 subjectPublicKeyInfo SEQUENCE.

[[DISCUSS: since the bskey identity is being negotiated we already know what the client cert type will be, the ClientCertTypeExtension is superfluous. Should it be removed from this spec?]]

When the server processes the client's Certificate it MUST ensure that it is identical to the bootstrapping public key that it used to generate an external PSK and PSKIdentifier 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.

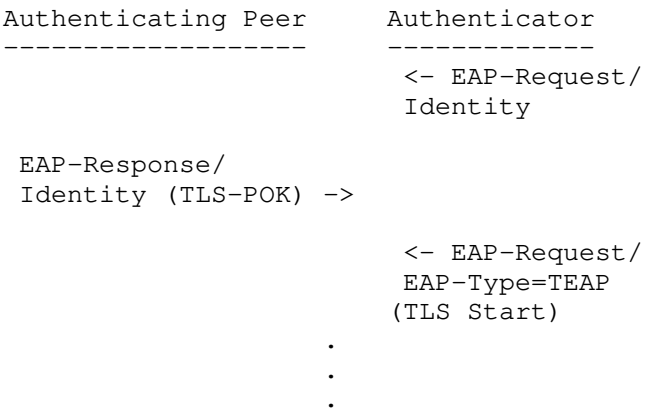
Figure 1: TLS 1.3 TLS-POK Handshake

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.



4. Summary of Work

[TODO: agree with WG chairs where this work lives and where it should be documented.]

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, but it may be more appropriate to merge into an existing document.

5. IANA Considerations

IANA will allocated an ExtensionPSKIdentityType for the bskey type from the TLS 1.3 repository created by [extensible-psks] and replace TBD in this document with that number.

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 it is the appropriate device from which to receive provisioning, for instance via [RFC7170].

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

[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.

[extensible-psks] Wood, C. and R. Anderson, "Extensible Pre-Shared Key Types for TLS", n.d., <https://chris-wood.github.io/draft-tls-extensible-psks/draft-group-tls-extensible-psks.html>.

[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>.

[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>.

[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