Workgroup: ACME Working Group
Internet-Draft:
draft-bweeks-acme-device-attest-00
Published: 17 May 2022
Intended Status: Standards Track
Expires: 18 November 2022
Authors: B. Weeks
Google

Automated Certificate Management Environment (ACME) Device Attestation Extension

Abstract

This document specifies new identifiers and a challenge for the Automated Certificate Management Environment (ACME) protocol which allows validating the identity of a device using attestation.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Automatic Certificate Management Environment (ACME) [RFC8555] standard specifies methods for validating control over identifiers, such as domain names. It is also useful to be able to validate properties of the device requesting the certificate, such as the identity of the device and if the certificate key is protected by a secure cryptoprocessor.

Many operating systems and device vendors offer functionality enabling a device to generate a cryptographic attestation of their identity, such as:

*Android Key Attestation

*Chrome OS Verified Access

*Trusted Platform Module

Using ACME and device attestation to issue client certificates for enterprise PKI is anticipated to be the most common use case. The following variances to the ACME specification are described in this document:

*Addition of permanent-identifier and hardware-module identifier types.

*Addition of the device-attest-01 challenge type to prove control of the permanent-identifier and hardware-module identifier types.

*The challenge response payload contains a serialized WebAuthn attestation statement format instead of an empty JSON object ({}).

- Accounts and external account binding being used as a mechanism to pre-authenticate requests to an enterprise CA.

2. Conventions and Definitions

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.

3. Permanent Identifier

A new identifier type, "permanent-identifier" is introduced to represent the identity of a device assigned by the manufacturer, typically a serial number. The name of this identifier type was chosen to align with [RFC4043], it does not prescribe the lifetime of the identifier, which is at the discretion of the Assigner Authority.

The identity along with the assigning organization can be included in the Subject Alternate Name Extension using the PermanentIdentifier form described in [RFC4043].

The server **MAY** allow the client to include this identifier in the certificate signing request (CSR). Alternatively if the server wishes to only issue privacy-preserving certificates, it **MAY** reject CSRs containing a PermanentIdentifier in the subjectAltName extension.

4. Hardware Module

A new identifier type, "hardware-module" is introduced to represent the identity of the secure cryptoprocessor, if any, that generated the certificate key.

(TODO describe the certificate representation)

If the server includes HardwareModule in the subjectAltName extension the CA **MUST** verify that the certificate key was generated on the secure cryptoprocessor with the asserted identity and type. The key **MUST NOT** be able to be exported from the cryptoprocessor.

If the server wishes to issue privacy-preserving certificates, it **MAY** omit HardwareModule from the subjectAltName extension.

5. Device Attestation Challenge

The client can prove control over a permanent identifier of a device by providing an attestation statement containing the identifier of the device.

The device-attest-01 ACME challenge object has the following format:

type (required, string): The string "device-attest-01".

token (required, string): A random value that uniquely identifies the challenge. This value **MUST** have at least 128 bits of entropy. It **MUST NOT** contain any characters outside the base64url alphabet, including padding characters ("="). See [RFC4086] for additional information on randomness requirements.

```
{
  "type": "device-attest-01",
  "url": "https://example.com/acme/chall/Rg5dV14Gh1Q",
  "status": "pending",
  "token": "evaGxfADs6pSRb2LAv9IZf17Dt3juxGJ-PCt92wr-oA"
}
```

A client fulfills this challenge by constructing a key authorization ([RFC4086] Section 8.1) from the "token" value provided in the challenge and the client's account key. The client then generates an WebAuthn attestation object using the key authorization as the challenge.

This specification borrows the WebAuthn *attestation object* representation as described in Section 6.5.4 of [WebAuthn] for encapsulating attestation formats with these modification:

- The key authorization is used to form *attToBeSigned*. This replaces the concatenation of *authenticatorData* and *clientDataHash*. *attToBeSigned* is hashed using an algorithm specified by the attestation format.

- The *authData* field is unused and should be omitted.

A client responds with the response object containing the WebAuthn attestation object in the "attObj" field to acknowledge that the challenge can be validated by the server.

On receiving a response, the server constructs and stores the key authorization from the challenge "token" value and the current client account key.

To validate a device attestation challenge, the server performs the following steps:

1. Perform the verification proceedures described in Section 6 of [WebAuthn].

2. Verify that key authorization conveyed by *attToBeSigned* matches the key authorization stored by the server.

```
POST /acme/chall/Rg5dV14Gh1Q
Host: example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/evOfKhNU60wg",
    "nonce": "SS2sSl1PtspvFZ08kNtzKd",
    "url": "https://example.com/acme/chall/Rg5dV14Gh1Q"
  }),
  "payload": base64url({
    "attObj": base64url(/* WebAuthn attestation object */),
  }),
  "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

6. Security Considerations

TODO Security

7. IANA Considerations

7.1. ACME Identifier Types

The "ACME Validation Methods" registry is to be updated to include the following entry:

Label	Reference
permanent-identifier	RFC XXXX
hardware-module	RFC XXXX

Table 1

7.2. ACME Validation Method

The "ACME Validation Methods" registry is to be updated to include the following entry:

Label	Identifier Type	Reference
device-attest-01	permanent-identifier	RFC XXXX

Table 2

8. 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/rfc/rfc2119>.

[RFC4043] Pinkas, D. and T. Gindin, "Internet X.509 Public Key Infrastructure Permanent Identifier", RFC 4043, DOI 10.17487/RFC4043, May 2005, <https://www.rfc-editor.org/rfc/rfc4043>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/rfc/rfc4086>.

[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/rfc/rfc8174>.

[RFC8555] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", RFC 8555, DOI 10.17487/RFC8555, March 2019, <https://www.rfc-editor.org/rfc/rfc8555>.

[WebAuthn] Hodges, J., Jones, J., Jones, M. B., Kumar, A., and E. Lundberg, "Web Authentication: An API for accessing Public Key Credentials Level 2", April 2021, <https://www.w3.org/TR/webauthn-2/>.

Appendix A. Enterprise PKI

ACME was originally envisioned for issuing certificates in the Web PKI, however this extension will primarily be useful in enterprise PKI. The subsection below covers some operational considerations for an ACME-based enterprise CA.

A.1. External Account Binding

An enterprise CA likely only wants to receive requests from authorized devices. It is **RECOMMENDED** that the server require a value for the "externalAccountBinding" field to be present in "newAccount" requests.

If an enterprise CA desires to limit the number of certificates that can be requested with a given account, including limiting an account to a single certificate. After the desired number of certificates have been issued to an account, the server **MAY** revoke the account as described in Section 7.1.2 of [RFC8555].

Acknowledgments

TODO acknowledge.

Author's Address

Brandon Weeks
Google

Email: bweeks@google.com