IETF
Internet-Draft
Intended status: Standards Track
Expires: May 20, 2020

K. Moriarty
Dell Technologies

November 17, 2019

ACME End User Client and Code Signing Certificates
draft-moriarty-acme-client-04

Abstract

Automated Certificate Management Environment (ACME) core protocol addresses the use case of web server certificates for TLS. This document extends the ACME protocol to support end user client, device client, and code signing certificates.

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 20, 2020.

Copyright Notice

Table of Contents

1. Introduction

ACME [RFC8555] is a mechanism for automating certificate management on the Internet. It enables administrative entities to prove effective control over resources like domain names, and automates the process of generating and issuing certificates.

The core ACME protocol defined challenge types specific to web server certificates with the possibility to create extensions, or additional challenge types for other use cases and certificate types. Client certificates, such as end user and code signing may also benefit from automated management to ease the deployment and maintenance of these certificate types, thus the definition of this extension defining challenge types specific to that usage.

2. Identity Proofing for Client Certificates

As with the TLS certificates defined in the core ACME document <xref target="RFC8555"/>, identity proofing for ACME issued end user client, device client, and code signing certificates is a separate process outside of the automation of ACME. Identity proofing may be an out-of-band process, if needed, and for this draft is likely tied to the credentials used for the defined challenge types.

Identity proofing for these certificate types present some challenges for process automation. NIST SP 800-63 r3 [NIST800-63r3] serves as guidance for identity proofing further detailed in NIST SP 800-63A [NIST800-63A] that may occur prior to the ability to automate certificate management via ACME or may obviate the need for it weighing end user privacy as a higher concern and allowing for credential issuance to be decoupled from identity proofing (IAL1). Using this guidance, a CA might select from the identity proofing levels to assert claims on the issued certificates as described in NIST SP 800-63 r3 [NIST800-63r3].

The certificate issuing CA may make this choice by certificate type issued. Once identity proofing has been performed, in cases where this is part of the process, and certificates have been issued, NIST SP 800-63 r3 [NIST800-63r3] includes recommendations for authentication or in the context of ACME, management of issuance for subsequent client, device, or code-signing certificates:

If federations and assertions are used for authorizing certificate issuance, NIST SP 800-63 C [NIST800-63C] may be referenced for guidance on levels of assurance.

Existing PKI certification authorities (CAs) tend to use a set of ad hoc protocols for certificate issuance and identity verification. For each certificate usage type, a basic process will be described to obtain an initial certificate and for the certificate renewal process. If higher assurance levels are desired, the guidance from NIST SP 800-63 r3 [NIST800-63r3] may be useful and out-of-band identity proofing options are possible options for pre-authorization challenges or notifications.

3. End User Client Certificates

A client certificate used to authenticate an end user may be used for mutual authentication in TLS, EAP-TLS, or messaging. The client certificate in this case may be stored in a browser, PKCS-#11 container, KMIP (possible, but just code signing and device client certificates in practice), or another key container. To obtain an end user client certificate, there are several possibilities to automate authentication of an identity credential intended to be tied to an end user.

[We need to determine if it is important in ACME to define an automated method that tests the identity or the user or to just have consistent credentials for the authentication challenges. The credentials may be distributed through an out-of-band method that involves identity proofing.]

[Several authentication options with identity proofing are intentionally provided for review and discussion by the ACME working group.]

A trusted federated service that ties the user to an email address with a reputation of the user attached to the email may be possible. One such example might be the use of a JWT signed OAuth token.

Risk based authentication used for identity proofing with red herring questions is a third option that could utilize public information on individuals to authenticate. This would be similar to the signup process used in some financial applications.

Existing credentials - for instance, FIDO. FIDO uses a public key pair and does not perform identity proofing. FIDO authentication provides a different key pair to each service using FIDO for authentication, which are generated at the client and registered by the server. This may require using the FIDO credentials from a specific service for authentication to gain ACME issued crededentials (not advised based on how FIDO credentials are supposed to be used). Are there instances where the same provider would issue both sets of credentials? You wouldn't want to expose your FIDO credentials to a different party, that's why each service has their own. Would you set up a mechanism to get FIDO credentials to then obtain a certificate? (What use cases would this be necessary? When do you need a certificate where you already have a specific public/private key pair?) This can be defined as an auth type, but should it be?

One-time password (OTP) authentication is a secure option. In cases where a higher assurance level is needed, OTP may be a good choice and many options exist today for OTP that could use an app on a phone for instance tied to an existing (or newly established) password. The OTP may be tied to an out-of-band process and may be associated with a username/password and other accounts.

One consideration is to understand if the use case could just use FIDO and not create anything new (ACME client certificates). FIDO provides a mechanism to have unique public key pair based access for client authentication to web sites and they are working on non-web. Identity proofing is intentionally decoupled from authentication in this model as that is in line with NIST 800-63r3 recommendations for privacy protections of the user. The credential in this case is authenticated and would be consistent for it's use, but the identity proofing for that credential is not performed. Obviously, identity proofing is more important for some services, like financial applications where tying the user to the identity for access to financial information is important. However, is automated identity proofing important for any user certificate or should it remain

decoupled where it could be automated by a service offering or is there a need for a standardized mechanism to support it for user certificates?

Three methods for ACME client authentication, not identity proofing, are proposed in the Challenge Type Section.

4. CodeSigning Certificates

The process to retrieve a code signing certificate is similar to that of a web server certificate, with differences primarily in the CSR request and the resulting certificate properties. [The storage and access of a code signing certificate must be protected and is typically done through hardware, a hardware security module (HSM), which likely has a PKCS#11 interface. A code signing certificate may either be a standard one or an extended validation (EV) certificate.]

For automation purposes, the process described in this document will follow the standard process and any out-of-band preprocessing can increase the level of the issued certificate if the CA offers such options and has additional identity proofing mechanisms (in band or out-of-band).

Strict vetting processes are necessary for many code signing certificates to provide a high assurance on the signer. In some cases, issuance of a standard CodeSigning certificate will be appropriate and no additional "challenges" [RFC8555 Section 8] will be necessary. In this case, the standard option could be automated very similar to Web server certificates with the only changes being in the CSR properties. However, this may not apply to all scenarios, such as those requiring EV certificates with the possibility for required out-of-band initial authentication and identity proofing.

EV code signing certificates have a distinct set of requirements from EV web certificates. In particular, they don't have associated domain names, nor is CAA checking done. The code signing certificate links a public key to an organization, not a domain. CAs may chose different methods to enable the use of ACME for EV code signing certificates. The intent of this work is to provide additional authentication challenge types that may enable their automation process.

Organization validation is required for standard code signing certificates from most issuers. The CSR is used to identify the organization from the included domain name in the request. The resulting certificate, however, instead contains the organization's name and for EV certificates, other identifying information for the organization. For EV certificates, this could require that the

domain is registered with the Certificate Authority provider, listed in CAA [RFC6844], and administrators for the account are named with provided portal access for certificate issuance and management options.

While ACME allows for the client to directly establish an account with a CA, an initial out-of-band process for this step may assist with the additional requirements for EV certificates and assurance levels typically required for code signing certificates. For standard certificates, with a recommendation for additional vetting through extended challenge options to enable ACME to establish the account directly. In cases where code signing certificates are used heavily for an organization, having the portal access replaced with ACME authenticated client access with extra challenges for authentication may be an option to automate the functionality.

[For standard certificates, is it worth defining SMS and email for the challenge? Obviously, EV needs more, so a few choices are suggested in this revision.]

To improve the vetting process, ACME's optional use of CAA [RFC6844] with the Directory "meta" data "caaIdentities" ([RFC8555] Section 9.7.6) assists with the validation that a CA may have issue certificates for any particular domain and is RECOMMENDED for use with code signing certificates for this additional level of validation checking on issued certificates.

CAA helps as anyone verifying a certificate used for code signing can verify that the CA used has been authorized to issue certificates for that organization. CSR requests for code signing certificates typically contain a Common Name (CN) using a domain name that is replaced with the organization name to have the expected details displayed in the resulting certificate. Since this work flow already occurs, there is a path to automation and validation via an existing ACME type, "dns".

As noted in RFC8555, "the external account binding feature (see Section 7.3.4) can allow an ACME account to use authorizations that have been granted to an external, non-ACME account. This allows ACME to address issuance scenarios that cannot yet be fully automated, such as the issuance of "Extended Validation" certificates."

The ACME challenge object, [RFC8555] Section 7.1.5 is RECOMMENDED for use for Pre-authorization ([RFC8555] Section 7.4.1). Additional challenge types are added to provide higher levels of security for this issuance verification step. The use of OTP, FIDO credentials (public/private key pairs), or validation from a certificate issued at account setup time are defined in Section 8. Pre-Authoriziation.

Questions for reviewers:

[Is there interest to set a specific or default challenge object for CodeSigning Certificates? Or should this be left to individual CAs to decide and differentiate? The current challenge types defined in RFC8555 include HTTPS (provisioning HTTP resources) and DNS (provisioning a TXT resource record). Use of DNS may be possible, but the HTTP resource doesn't necessarily make sense. Since the process to retrieve an EV CodeSigning certificate usually requires proof of the organization and validation from one of 2 named administrators, some other challenge type like public/private key pairs or OTP may be needed as defined challenge types. An organization may want to tie this contact to a role rather than a person and that consideration should be made in the design as well as implementation by organizations.]

ACME provides an option for notification of the operator via email or SMS upon issuance/renewal of a certificate after the domain has been validated as owned by the requestor. This option is RECOMMENDED due to the security considerations of code signing certificates as a way to limit or reduce the possibility of a third party gaining access to a code signing certificate inappropriately. Development of additional challenge types is included in this document to support this for pre-authorization, which would better match the security considerations for this certificate type. Additional types may be added if agreed upon by the working group.

Since DNS is used to identify the organization in the request, the identifier "type" ([RFC8555]Section 7.4) is set to dns, not requiring any additions to the ACME protocol for this type of certificate. The distinction lies in the CSR, where the values are set to request a CodeSigning certificate for a client certificate. [Question: Is it helpful to define an identifier for the administrator or for the developer to distinguish the certificate type in ACME and not just the CSR?]

KeyUsage (DigitalSignature) and ExtendedKeyUsage (CodeSigning) in the CSR MUST be set to the correct values for the CA to see the request is for a Code Signing certificate. The Enhanced Key Usage SHOULD be set to show this is a client certificate., using OID "1.3.6.1.5.5.7.3.2". The CN MUST be set to the expected registered domain with the CA account.

An advantage of ACME is the ability to automate rollover to allow for easy management of short expiry times on certificates. The lifetime of CodeSigning certificates is typically a year or two, but automation could allow for shorter expiry times becoming feasible. However, lifetimes are less of an issue for code signing certificates

than other certificate types. however there is a legitmate case for "one signature per certificate." Automation might be helpful in this case if patches or software updates were frequent or to minimize the knowledge needed for the organization using this method.

Automation of storage to an HSM, which typically requires authentication is intentionally left out-of-scope.

5. Pre-authorization

Additional challenge types are defined here for the verification of administrators at an organization requesting CodeSigning certificates. SMS and email are listed as possible in RFC8555 and may be used singularly or in combination as the ACME protocol allows for multiple pre-authorization challenges to be issued. Additional pre-authorization types are defined that provide a higher level of assurance to authorize a request.

6. Challenge Types

The challenge types defined in the following subsections are to authenticate individuals or holders of specific pre-issued credentials (users acting in roles for an organization). The challenge types can be used to obtain end user certificate types or as a pre-authorization challenges with certificate types such as the Code Signing Certificate. Please note that the pre-authorization challenge is also coupled with the account certificate in ACME for verification. The process for obtaining EV Code Signing Certificates typically requires authorization from one or more individuals in a role for the organization. The use of pre-issued secure credentials, at an assurance level appropriate for the certificate type being issued, provides a way to automate the issuance and renewal process.

6.1. One Time Password (OTP)

There are numerous one time password technologies with slight variations between implementations. The response to the challenge is entered in the provided URL, offering flexibility to those using this challenge type to acomodate the specific requirements of their solution. Looking at 2 OTP solutions, the challenge response is provided via a tool or app without any user interaction of information required from the server to generate the challenge. The 2 solutions that operate in this manner include SecureID and Duo Security. If a challenge is required to generate the response to be provided in the URL, the token can supply the challenge.

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

token (required, string): A random value that uniquely identifies the challenge. OTP types and input vary between technologies. The token value will match the type expected for the pre-issued OTP credential. The user will be able to supply a response in the provided URL from this challenge. It MUST NOT contain any characters outside the base64url alphabet and MUST NOT include base64 padding characters ("=").

```
{
  "type": "otp-01",
  "url": "https://example.com/acme/chall/WrV_H87EyD3",
  "status": "pending",
  "token": "challenge"
}
```

6.1.1. HMAC-Based One-Time Password (HOTP)

HOTP([RFC4226]) describes an algorithm for the generation of time-based password values.

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

token (required, string): The HOTP value. This SHOULD be the 6 digit representation.

```
 {
   "type": "hotp-01",
   "url": "https://example.com/acme/chall/WrV_H87EyD3",
   "status": "pending",
   "token": "123456"
 }
```

6.1.2. Time-Based One-Time Password (TOTP)

TOTP([RFC6238]) describes an algorithm for the generation of time-based password values, an extension from HOTP.

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

token (required, string): The TOTP value. This SHOULD be the 6 digit representation.

```
 {
   "type": "totp-01",
   "url": "https://example.com/acme/chall/WrV_H87EyD3",
   "status": "pending",
   "token": "123456"
 }
```

6.1.3. Generic One Time Password (OTP)

There are numerous other one time password technologies with slight variations between implementations. The response to the challenge is entered in the provided URL, offering flexibility to those using this challenge type to acomodate the specific requirements of their solution. Looking at 2 OTP solutions, the challenge response is provided via a tool or app without any user interaction of information required from the server to generate the challenge. The 2 solutions that operate in this manner include SecureID and Duo Security. If a challenge is required to generate the response to be provided in the URL, the token can supply the challenge.

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

token (required, string): A random value that uniquely identifies the challenge. OTP types and input vary between technologies. The token value will match the type expected for the pre-issued OTP credential. The user will be able to supply a response in the provided URL from this challenge. It MUST NOT contain any characters outside the base64url alphabet and MUST NOT include base64 padding characters ("=").

```
{
  "type": "otp-01",
  "url": "https://example.com/acme/chall/WrV_H87EyD3",
  "status": "pending",
  "token": "challenge"
}
```

6.2. Public Key Cryptography

Certificates may be pre-issued and stored according to assurance level requirements for the purpose of identifying a user's identity. If a higher assurance level is needed for a user serving in a specific role or for that individual, it is possible for identity proofing to occur in person using identifiers acceptable for the specified process and the private key stored appropriately for the required assurance level. PKCS#11 software or hardware tokens are both possible options. This model assumes that there may be multiple authorized users with different certificates that can be used for the authorization or pre-authentication challenge. As such, the user first provides the digital signature, so the account management can determine if one of the acceptable certificates was used to digitally sign the token.

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

token (required, string): A random value that uniquely identifies the challenge. The token for a certificate authentication challenge includes a value for the recipeint to digitally sign using their private key and post to the provided URL. The ACME server then uses the digitally signed content to verify that the challenge was signed using authorized credentials (certificate issued and authorized for this challenge type). It MUST NOT contain any characters outside the base64url alphabet and MUST NOT include base64 padding characters ("=").

```
{
  "type": "cert-01",
  "url": "https://example.com/acme/chall/WrV_H87EyD3",
  "status": "pending",
  "token": "Some challenge to digitally sign"
}
```

6.3. WebAuthn or Public/Private Key Pairs

W3C's WebAuthn uses raw public/private key pairs that are issued specific to a service. If WebAuthn or public/private key pairs (PPKP) are selected as the challenge type, the account and credential issuance will have to occur prior to use of this challenge type. The WebAuthn or public/private key pair credentials would be specific to the certificate management account and would be created by the client, then registered with the service as occurs with normal WebAuthn regisration of credentials. As with normal WebAuthn and public/private key pairs, the token or challenge is digitally signed to prove possession of the private key.

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

token (required, string): A random value that uniquely identifies the challenge. This challenge will operate much in the same way as the certificate challenge as the operations are largely the same. The user will be able to supply a response in the provided URL from this challenge. It MUST NOT contain any characters outside the base64url alphabet and MUST NOT include base64 padding characters ("=").

```
{
  "type": "ppkp-01",
  "url": "https://example.com/acme/chall/WrV_H87EyD3",
  "status": "pending",
  "token": "Some challenge to sign"
}
```

7. Security Considerations

This will likely be full of considerations and is TBD for this revision until challenge types are settled.

8. IANA Considerations

This memo includes no request to IANA, yet.

9. Contributors

Thank you to reviewers and contributors who helped to improve this document. Thank you to Thomas Peterson who added the one-time password types, HOTP and TOTP. Thank you to Tim Hollebeek for your early review and added text specific to EV certificate issuance and one time use code signing certificates. Thank you to Andrei Popov and Deb Cooley for your reviews and suggestions made in -04.

10. References

10.1. 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>.

[RFC4226] M'Raihi, D., Bellare, M., Hoornaert, F., Naccache, D., and O. Ranen, "HOTP: An HMAC-Based One-Time Password Algorithm", RFC 4226, DOI 10.17487/RFC4226, December 2005, <https://www.rfc-editor.org/info/rfc4226>.

[RFC6238] M'Raihi, D., Machani, S., Pei, M., and J. Rydell, "TOTP: Time-Based One-Time Password Algorithm", RFC 6238, DOI 10.17487/RFC6238, May 2011, <https://www.rfc-editor.org/info/rfc6238>.

[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>.

[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>.

[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/info/rfc8555>.

10.2. Informative References

[I-D.ietf-acme-ip]
Shoemaker, R., "ACME IP Identifier Validation Extension", draft-ietf-acme-ip-08 (work in progress), October 2019.

10.3. URL References

[NIST800-63A]
US National Institute of Standards and Technology, "https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63a.pdf".

[NIST800-63B]
US National Institute of Standards and Technology, "https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf".

[NIST800-63C]
US National Institute of Standards and Technology, "https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63c.pdf".

[NIST800-63r3]
US National Institute of Standards and Technology, "https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63-3.pdf".

Appendix A. Change Log

Note to RFC Editor: if this document does not obsolete an existing RFC, please remove this appendix before publication as an RFC.

02 draft added subsections contributed from Thomas Peterson on HOTP and TOTP.

Appendix B. Open Issues

Note to RFC Editor: please remove this appendix before publication as an RFC.

Author's Address

Kathleen M. Moriarty
Dell Technologies
176 South Street
Hopkinton
US

EMail: Kathleen.Moriarty@dell.com