Stream: Internet Engineering Task Force (IETF)
RFC: 9447
Category: Standards Track
Published: September 2023
ISSN: 2070-1721
Authors: J. Peterson, *Neustar*; M. Barnes, *Neustar*; D. Hancock, *Somos*; C. Wendt, *Somos*

RFC 9447
Automated Certificate Management Environment (ACME) Challenges Using an Authority Token

Abstract

Some proposed extensions to the Automated Certificate Management Environment (ACME) rely on proving eligibility for certificates through consulting an external authority that issues a token according to a particular policy. This document specifies a generic Authority Token Challenge for ACME that supports subtype claims for different identifiers or namespaces that can be defined separately for specific applications.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc9447.

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 issuing certificates that attest control or ownership of those resources.

In some cases, proving effective control over an identifier requires an attestation from a third party who has authority over the resource, for example, an external policy administrator for a namespace other than the DNS application ACME was originally designed to support. In order to automate the process of issuing certificates for those resources, this specification defines a generic Authority Token Challenge that ACME servers can issue in order to require clients to return an Authority Token that authorizes a given issuance. The challenge contains a type indication that tells the client what sort of token it needs to acquire. It is expected that the Authority Token Challenge will be usable for a variety of identifier types. In particular, this document describes an architecture for Authority Tokens, defines a JSON Web Token (JWT) [RFC7519] Authority Token format along with a protocol for token acquisition, and shows how to integrate these tokens into an ACME challenge.

As a concrete example, [RFC9448] provides a mechanism that allows service providers to acquire certificates corresponding to a Service Provider Code (SPC) as defined in [RFC8226] by consulting an external authority responsible for those codes. Furthermore, Communications Service Providers (CSPs) can delegate authority over numbers to their customers, and those CSPs who support ACME can then help customers to acquire certificates for those numbering resources with ACME. This can permit number acquisition flows compatible with those shown in [RFC8396].

2. Requirements Language

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. ACME Authority Token Challenge

Proving that a device on the Internet has effective control over a non-Internet resource is not as straightforward as proving control over Internet resources, like a DNS zone or a web page. Provided that the issuer of identifiers in a namespace, or someone acting on the issuer's behalf, can implement a service that grants Authority Tokens to the people to whom it has issued identifiers, a generic token could be used as a response to an ACME challenge. This specification, therefore, defines an Authority Token issued by an authority over a namespace to an ACME client for delivery to an ACME server in response to a challenge. Authority over a hierarchical namespace can also be delegated so that delegates of a root authority can themselves act as Token Authorities for certain types of names.

This architecture assumes a trust relationship between certification authorities (CAs) and Token Authorities, i.e., that CAs are willing to accept the attestation of Token Authorities for particular types of identifiers as sufficient proof to issue a credential. It furthermore assumes that ACME clients have a relationship with Token Authorities, which permits them to authenticate and authorize the issuance of Authority Tokens to the proper entities. This ACME challenge has no applicability to identifiers or authorities where those pre-associations cannot be assumed.

The ACME Authority Token Challenge type, "tkauth-01", is here specified for use with the "TNAuthList" (Telephone Number Authentication List) ACME Identifier Type described in [RFC9448]; in order to use the "tkauth-01" Validation Method with an ACME Identifier Type other than "TNAuthList", that identifier type would need to be listed in a new registration in the ACME Validation Methods registry maintained by IANA. "tkauth-01" furthermore supports different token subtypes. The token subtype is determined by a new ACME challenge field, tkauth-type. An IANA registry is used to manage the values of tkauth-type (see Section 6.3). Additionally, this challenge type also has a new "token-authority" field to designate a location where a token can be acquired.

3.1. Token Type Requirements

IANA will maintain a registry of tkauth-types under a policy of Specification Required. In order to register a new tkauth-type, specifications must address the following requirements; in cases where a tkauth-type admits of its own subtypes, subtypes inherit these requirements.

While Authority Token types do not need to be specific to a namespace, every token must carry enough information for a CA to determine the name for which certificate issuance is authorized. Some types of Authority Token types might be reusable for a number of different namespaces; others might be specific to a particular type of name. Therefore, in defining tkauth-types, future specifications must indicate how a token conveys to the CA the name(s) that the Token Authority is attesting that the ACME client controls.

While nothing precludes use cases where an ACME client is itself a Token Authority, an ACME client will typically need a protocol to request and retrieve an Authority Token. The Token Authority will require certain information from an ACME client in order to ascertain that it is an authorized entity to request a certificate for a particular name. The protocols used to request an Authority Token **MUST** convey to the Token Authority the identifier type and value that will be used in the ACME challenge, as well as the binding (see Section 3.3), and those **MUST** be reflected in the Authority Token. A baseline mechanism for how the Token Authority authenticates and authorizes ACME clients to receive Authority Tokens is given in Section 5.

Because the assignment of resources can change over time, demonstrations of authority must be regularly refreshed. Definitions of a tkauth-type **MUST** specify how they manage the freshness of authority assignments. Typically, a CA will expect a regular refreshing of the token.

3.2. Authority Token Scope

An Authority Token is used to answer a challenge from an ACME server, upon a request for the issuance of a certificate. It could be that the Authority Token is requested from the Token Authority after a challenge has been received, or it could be that the Authority Token was acquired prior to the initial ACME client request. A Token Authority could grant an Authority Token that has the exact same scope as the requested certificate to a client; alternatively, an Authority Token could attest to all of the resources that the client is eligible to receive certificates for, which could be a superset of the scope of the requested certificate.

For example, imagine a case where a Token Authority for DNS names knows that a client is eligible to receive certificates for "example.com" and "example.net". The client asks an ACME server for a certificate for "example.com", and the server directs the client to acquire an Authority Token from the Token Authority. When the client sends an acquisition request (see Section 5) to the Token Authority, the Token Authority could issue a token scoped just to "example.com" or a token that attests the client is eligible to receive certificates for both "example.com" or "example.net". The advantage of the latter is that if, at a later time (but one within the expiry of the token), the client wanted to acquire a certificate for "example.net", it would not have to return to the Token Authority, as the Token effectively pre-authorized the issuance of that certificate.

Applications of the Authority Token to different identifier types might require different scopes, so registrations of tkauth-types should be clear about if and how a scope greater than that of the requested certificate would be conveyed in a token.

3.3. Binding Challenges

Applications that use the Authority Token need a way to correlate tokens issued by a Token Authority with the proper ACME client to prevent replay or cut-and-paste attacks using a token issued for a different purpose. To mitigate this, Authority Tokens contain a binding signed by a Token Authority; an ACME server can use the binding to determine that a Token presented by a client was in fact granted by the Token Authority based on a request from the client and not from some other entity. It is **RECOMMENDED** that the ACME account fingerprint be used for this purpose.

Creating a binding from an Authority Token to a particular ACME account entails that the Token could be reused up until its expiry for multiple challenges issued by an ACME server. This might be a desirable property when using short-lived certificates, for example, in any cases where the ACME server issues challenges more frequently that an Authority Token can or should issue tokens or in cases where the Authority Token scope (see Section 3.2) is broad, so certificates with a more narrow scope may periodically be issued.

For some identifier types, it may be more appropriate to bind the Authority Token to a nonce specific to the challenge rather than to an ACME account fingerprint. Any specification of the use of the nonce or other factors for this purpose is left to the identifier type profile for the Authority Token.

Note that the fingerprint value in the client's JWT is reflected in the Authority Token returned by the Token Authority; the Token Authority has no requirement to validate that fingerprint. Were a fingerprint to be captured by an attacker that had its own account with the Token Authority, it could replay that fingerprint in its own JWT in order to receive an Authority Token with that fingerprint. However, were the attacker to present that Authority Token to an ACME service, the service would see the fingerprint does not match the attacker's ACME account fingerprint. So unless an attacker can compromise a target ACME account or gain similar privileges, the binding would be secure.

4. Authority Token Challenge tkauth-type Registration

This document specifies a tkauth-type of "atc", which contains a standard JWT [RFC7519] using a signature string defined by a JSON Web Signature (JWS) [RFC7515]. The "atc" tkauth-type **MAY** be used for any number of different ACME Identifier Types in the ACME challenge.

A new JWT claim, "atc", is defined below and lists the identifier type used in this Authority Token. The "atc" tkauth-type is restricted to the JWTs; if a non-JWT format is desired for the ACME Authority Token Challenge, a different tkauth-type should be specified and registered in the "ACME Authority Token Challenge Types" registry defined in Section 6.3.

For this ACME Authority Token usage of a JWT, it is **OPTIONAL** for the payload of the JWT to contain an "iss", indicating the Token Authority that generated the token if the "x5u" or "x5c" element in the header does not already convey that information; typically, this will be the same location that appeared in the "token-authority" field of the ACME challenge, when present. In order to satisfy the requirement for replay prevention, the JWT **MUST** contain a "jti" element and an "exp" claim; the "exp" claim manages token freshness. In addition to helping to manage replay, the "jti" provides a convenient way to reliably track when the same "atc" Authority Token is being used for multiple challenges over time within its set lifetime.

The JWT payload **MUST** also contain a new JWT claim, "atc", for Authority Token Challenge, which contains three mandatory elements in a JSON map: the ATC identifier type ("tktype"), the identifier value ("tkvalue"), and the binding ("fingerprint"). The use of "tktype" is restricted to the values in the "ACME Identifier Types" registry, as defined by [RFC8555]. The identifier type and value are those given in the ACME challenge and conveyed to the Token Authority by the ACME client. For the purposes of the "atc" tkauth-type, the binding "fingerprint" is assumed to be a fingerprint of the ACME credential for the account used to request the certificate, but the specification of how the binding is generated is left to the identifier type profile for the Authority Token (see Section 3.3). The "tkvalue" indicates the scope of the authority that the token and its semantics are outside the scope of this document, as they will be specified by the "tkvalue" identifier in a separate specification.

Following the example of [RFC9448], the "tktype" identifier type could be the TNAuthList (as defined in [RFC8226]), which would be the value for the "tkvalue" element that the Token Authority is attesting. Practically speaking, that scope may comprise a list of Service Provider Code elements, telephone number range elements, and/or individual telephone numbers. So for example:

```
    {
 "protected": base64url({
     "typ":"JWT",
  "alg":"ES256",
  "x5u":"https://authority.example.org/cert"
    }),
 "payload": base64url({
     "iss":"https://authority.example.org/authz",
  "exp":1300819380,
  "jti":"id6098364921",
  "atc":{"tktype":"TnAuthList","tkvalue":"F83n2a...avn27DN3==",
  "fingerprint":"SHA256 56:3E:CF:AE:83:CA:4D:15:B0:29:FF:1B:71:D3:
 BA:B9:19:81:F8:50:9B:DF:4A:D4:39:72:E2:B1:F0:B9:38:E3"}
    }),
  "signature": "9cbg5JO1Gf5YLjjz...SpkUfcdPai9uVYYQ"
    }
```

Optionally, the "atc" claim may contain a fourth boolean element, "ca". If set to "true", the "ca" element indicates that the Token Authority is granting permission to issue a certification authority certificate rather than an end-entity certificate for the names in question. This permits subordinate delegations from the issued certificate (using [RFC9115] or similar mechanisms). If the "ca" element is absent, the Token Authority is explicitly withholding permission. The "atc" object in the example above would then look like:

```
"atc":{"tktype":"TnAuthList","tkvalue":"F83n2a...avn27DN3==",
"ca":true,"fingerprint":"SHA256 56:3E:CF:AE:83:CA:4D:15:B0:29:FF:1B:
71:D3:BA:B9:19:81:F8:50:9B:DF:4A:D4:39:72:E2:B1:F0:B9:38:E3"} }
```

Specifications of "tktype" identifier types may define additional optional "atc" elements.

5. Acquiring a Token

The acquisition of an Authority Token requires a network interface, apart from potential use cases where the entity that acts as an ACME client itself also acts as a Token Authority trusted by the ACME server. Implementations compliant with this specification **MUST** support an HTTPS interface for Authority Token acquisition as described below, though other interfaces **MAY** be supported as well.

5.1. Basic REST Interface

In order to request an Authority Token from a Token Authority, a client sends a HTTPS POST request [RFC9110]. This specification assumes that Token Authority URIs are known to clients through preexisting business relationships and that the credentials and related authentication and authorization for Authority Token acquisition are encompassed in that relationship. Different services may organize their web resources in domain-specific ways, but the resource locator should specify the account of the client, an identifier for the service provider, and finally a locator for the token.

```
POST /at/account/:id/token HTTP/1.1
Host: authority.example.com
Content-Type: application/json
```

Note that ":id" here is a placeholder for an actual account identifier. The body of the POST request **MUST** contain the Authority Token Challenge element (the key "atc", colon, and its value) that the client is requesting the Token Authority generate. In this way, the client proposes the scope of the Authority Token it would like to receive from the Token Authority.

In common use cases, the "tkvalue" in this request is asking that the Token Authority issue a token that attests the entire scope of authority to which the client is entitled. The client may also request an Authority Token with some subset of its own authority via the "tkvalue" element in the Authority Token Challenge object. The way that "tkvalue" is defined will necessarily be specific to the identifier type. For the TNAuthList identifier type, for example, an object requesting an Authority Token could request authority for only a single telephone number in a way that is defined in the TNAuthList specification.

Finally, the JSON object **MAY** also contain an optional boolean element, "ca", which signifies that the client is requesting that the Token Authority issue an Authority Token with the "ca" flag set, as described in Section 4.

After an HTTPS-level challenge (e.g., a 401 HTTP response code) to verify the identity of the client and subsequently making an authorization decision about whether the client should receive an Authority Token with the requested scope, then in the success case, the Token Authority **MUST** return a 200 OK with a body of type "application/json" containing the Authority Token.

A full example of "atc" token acquisition using the HTTP interface, with the "tktype" of "TNAuthList", is given in Section 5.5 of [RFC9448].

6. IANA Considerations

6.1. ACME Validation Method Registration

IANA has added a new ACME Validation Method (per [RFC8555]) in the "ACME Validation Methods" subregistry of the "Automated Certificate Management Environment (ACME) Protocol" registry group as follows:

Label: tkauth-01

Identifier Type: TNAuthList

ACME: Y

Reference: RFC 9447

6.2. JSON Web Token Claim Registration

IANA has added a new claim in the "JSON Web Token Claims" registry, as defined in [RFC7519], as follows:

Claim name: atc

Claim Description: Authority Token Challenge

Change Controller: IETF

Specification document(s): RFC 9447

6.3. Creation of ACME Authority Token Challenge Types Registry

IANA has created a new registry for "ACME Authority Token Challenge Types" as used in these challenges, under a policy of Specification Required and following the requirements in Section 3.1, with three columns: Label, Description, and Reference. The initial content of the registry is as follows:

Label: atc (as defined in Section 4)

Description: JSON Web Token (JWT) challenge type

Reference: RFC 9447

7. Security Considerations

Per the guidance in [RFC8555], ACME transactions **MUST** use TLS, and similarly, the HTTPS REST transactions used to request and acquire Authority Tokens **MUST** use TLS. These measures are intended to prevent the capture of Authority Tokens by eavesdroppers. A preexisting trust relationship between the HTTPS REST client and the Token Authority must also exist in order for the parties to meaningfully authenticate one another. The security considerations of [RFC8555] apply to the use of the mechanism in this specification. Implementations should follow the best practices identified in [RFC8725].

As described in Section 3.2, an Authority Token can either have a scope that attests all of the resources that a client is eligible to receive certificates for or potentially a more limited scope that is intended to capture only those resources for which a client will receive a certificate from a particular certification authority. Any certification authority that sees an Authority Token can learn information about the resources a client can claim. In cases where this incurs a privacy risk, Authority Token scopes should be limited to only the resources that will be attested by the requested ACME certificate.

In cases where a tkauth-type, as defined in Section 4, admits of its own subtypes, the security of features like binding challenges (see Section 3.3) will depend on the subtype specification.

The capture of Authority Tokens by an adversary could enable an attacker to acquire a certificate from a CA. Therefore, all Authority Tokens **MUST** contain a field that identifies to the CA which ACME client requested the token from the Token Authority; here, that is the fingerprint specified in Section 4. All Authority Tokens must specify an expiry (of the token itself as proof for a CA, as opposed to the expiry of the name), and for some applications, it may make sense for that expiry to be quite short. ACME services relying on Authority Tokens **SHOULD NOT** issue certificates with a longer expiry than the expiry of the Authority Token. Any protocol used to retrieve Authority Tokens from a Token Authority **MUST** use confidentiality to prevent eavesdroppers from acquiring an Authority Token. The details of this protocol are out of the scope of this specification.

This document only specifies SHA256 for the fingerprint hash. However, the syntax of the fingerprint object would permit other keys if, due to concerns about algorithmic agility, a more robust algorithm were required at a future time. Future specifications can define new keys for the fingerprint object as needed.

8. References

8.1. 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>.

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <https://www.rfc-editor.org/info/rfc7515>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

[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>.

[RFC8725] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", BCP 225, RFC 8725, DOI 10.17487/RFC8725, February 2020, <https://www.rfc-editor.org/info/rfc8725>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[RFC9448] Wendt, C., Hancock, D., Barnes, M., and J. Peterson, "TNAuthList Profile of Automated Certificate Management Environment (ACME) Authority Token", RFC 9448, DOI 10.17487/RFC9448, September 2023, <https://www.rfc-editor.org/info/rfc9448>.

8.2. Informative References

[RFC8226] Peterson, J. and S. Turner, "Secure Telephone Identity Credentials: Certificates", RFC 8226, DOI 10.17487/RFC8226, February 2018, <https://www.rfc-editor.org/info/rfc8226>.

[RFC8396] Peterson, J. and T. McGarry, "Managing, Ordering, Distributing, Exposing, and Registering Telephone Numbers (MODERN): Problem Statement, Use Cases, and Framework", RFC 8396, DOI 10.17487/RFC8396, July 2018, <https://www.rfc-editor.org/info/rfc8396>.

[RFC9115] Sheffer, Y., López, D., Pastor Perales, A., and T. Fossati, "An Automatic Certificate Management Environment (ACME) Profile for Generating Delegated Certificates", RFC 9115, DOI 10.17487/RFC9115, September 2021, <https://www.rfc-editor.org/info/rfc9115>.

Acknowledgements

We would like to Roman Danyliw and Ben Kaduk for contributions to this problem statement and framework.

Authors' Addresses

Jon Peterson
Neustar, Inc.
Email: jon.peterson@team.neustar

Mary Barnes
Neustar, Inc.
Email: mary.ietf.barnes@gmail.com

David Hancock
Somos
Email: davidhancock.ietf@gmail.com

Chris Wendt
Somos
Email: chris-ietf@chriswendt.net