Stream: Internet Engineering Task Force (IETF)
RFC: 8737
Category: Standards Track
Published: February 2020
ISSN: 2070-1721
Author: R.B. Shoemaker
ISRG

RFC 8737
Automated Certificate Management Environment (ACME) TLS Application-Layer Protocol Negotiation (ALPN) Challenge Extension

Abstract

This document specifies a new challenge for the Automated Certificate Management Environment (ACME) protocol that allows for domain control validation using TLS.

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/rfc8737.

Copyright Notice

Table of Contents

1. Introduction

The Automatic Certificate Management Environment (ACME) [RFC8555] specification describes methods for validating control of domain names via HTTP and DNS. Deployment experience has shown it is also useful to be able to validate domain control using the TLS layer alone. In particular, this allows hosting providers, Content Distribution Networks (CDNs), and TLS-terminating load balancers to validate domain control without modifying the HTTP handling behavior of their backends.

This document specifies a new TLS-based challenge type, tls-alpn-01. This challenge requires negotiating a new application-layer protocol using the TLS Application-Layer Protocol Negotiation (ALPN) Extension [RFC7301]. Because this protocol does not build on a pre-existing deployment base, the ability to complete tls-alpn-01 challenges requires changes by service providers, making it explicitly an opt-in process. Because service providers must proactively deploy new code in order to implement tls-alpn-01, we can specify stronger controls in that code, resulting in a stronger validation method.

2. 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.

3. TLS with Application-Layer Protocol Negotiation (TLS ALPN) Challenge

The TLS with Application-Layer Protocol Negotiation (TLS ALPN) validation method proves control over a domain name by requiring the ACME client to configure a TLS server to respond to specific connection attempts using the ALPN extension with identifying information. The ACME server validates control of the domain name by connecting to a TLS server at one of the addresses resolved for the domain name and verifying that a certificate with specific content is presented.

The tls-alpn-01 ACME challenge object has the following format:

type (required, string): The string "tls-alpn-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 as described in Section 5 of [RFC4648]. Trailing '=' padding characters **MUST** be stripped. See [RFC4086] for additional information on randomness requirements.

The client prepares for validation by constructing a self-signed certificate that **MUST** contain an acmeIdentifier extension and a subjectAlternativeName extension [RFC5280]. The subjectAlternativeName extension **MUST** contain a single dNSName entry where the value is the domain name being validated. The acmeIdentifier extension **MUST** contain the SHA-256 digest [FIPS180-4] of the key authorization [RFC8555] for the challenge. The acmeIdentifier extension **MUST** be critical so that the certificate isn't inadvertently used by non-ACME software.

The acmeIdentifier extension is identified by the id-pe-acmeIdentifier object identifier (OID) in the id-pe arc [RFC5280]:

```
id-pe-acmeIdentifier OBJECT IDENTIFIER ::=  { id-pe 31 }
```

The extension has the following ASN.1 [X.680] format :

```
Authorization ::= OCTET STRING (SIZE (32))
```

The extnValue of the id-pe-acmeIdentifier extension is the ASN.1 DER encoding [X.690] of the Authorization structure, which contains the SHA-256 digest of the key authorization for the challenge.

Once this certificate has been created, it **MUST** be provisioned such that it is returned during a TLS handshake where the "acme-tls/1" application-layer protocol has been negotiated and a Server Name Indication (SNI) extension [RFC6066] has been provided containing the domain name being validated.

A client responds by POSTing an empty JSON object ({}) to the challenge URL to acknowledge that the challenge is ready to be validated by the server. The base64url encoding of the protected headers and payload is described in Section 6.1 of [RFC8555].

```
POST /acme/authz/1234/1
Host: example.com
Content-Type: application/jose+json

{
  "protected": base64url({
    "alg": "ES256",
    "kid": "https://example.com/acme/acct/1",
    "nonce": "JHb54aT_KTXBWQOzGYkt9A",
    "url": "https://example.com/acme/authz/1234/1"
  }),
  "payload": base64url({}),
  "signature": "Q1bURgJoEslbD1c5...3pYdSMLio57mQNN4"
}
```

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

The server then verifies the client's control over the domain by verifying that the TLS server was configured as expected using the following steps:

1. The ACME server computes the expected SHA-256 digest of the key authorization.
2. The ACME server resolves the domain name being validated and chooses one of the IP addresses returned for validation (the server **MAY** validate against multiple addresses if more than one is returned).
3. The ACME server initiates a TLS connection to the chosen IP address. This connection **MUST** use TCP port 443. The ACME server **MUST** provide an ALPN extension with the single protocol name "acme-tls/1" and an SNI extension containing only the domain name being validated during the TLS handshake.
4. The ACME server verifies that during the TLS handshake the application-layer protocol "acme-tls/1" was successfully negotiated (and that the ALPN extension contained only the value "acme-tls/1") and that the certificate returned contains:
    - a subjectAltName extension containing the dNSName being validated and no other entries

- a critical acmeIdentifier extension containing the expected SHA-256 digest computed in step 1

The comparison of dNSNames **MUST** be case insensitive [RFC4343]. Note that as ACME doesn't support Unicode identifiers, all dNSNames **MUST** be encoded using the rules of [RFC3492].

If all of the above steps succeed, then the validation is successful. Otherwise, it fails.

4. acme-tls/1 Protocol Definition

The "acme-tls/1" protocol **MUST** only be used for validating ACME tls-alpn-01 challenges. The protocol consists of a TLS handshake in which the required validation information is transmitted. The "acme-tls/1" protocol does not carry application data. Once the handshake is completed, the client **MUST NOT** exchange any further data with the server and **MUST** immediately close the connection. While this protocol uses X.509 certificates, it does not use the authentication method described in [RFC5280] and, as such, does not require a valid signature on the provided certificate nor require the TLS handshake to complete successfully. An ACME server may wish to use an off-the-shelf TLS stack where it is not simple to allow these divergences in the protocol as defined. Because of this, an ACME server **MAY** choose to withhold authorization if either the certificate signature is invalid or the handshake doesn't fully complete.

ACME servers that implement "acme-tls/1" **MUST** only negotiate TLS 1.2 [RFC5246] or higher when connecting to clients for validation.

5. Security Considerations

The design of this challenge relies on some assumptions centered around how an HTTPS server behaves during validation.

The first assumption is that when an HTTPS server is being used to serve content for multiple DNS names from a single IP address, it properly segregates control of those names to the users that own them. This means that if User A registers Host A and User B registers Host B, the HTTPS server should not allow a TLS request using an SNI value for Host A to be served by User B or a TLS connection with a server_name extension identifying Host B to be answered by User A. If the HTTPS server allows User B to serve this request, it allows them to illegitimately validate control of Host A to the ACME server.

The second assumption is that a server will not violate [RFC7301] by blindly agreeing to use the "acme-tls/1" protocol without actually understanding it.

To further mitigate the risk of users claiming domain names used by other users on the same infrastructure hosting providers, CDNs, and other service providers **SHOULD NOT** allow users to provide their own certificates for the TLS ALPN validation process. If providers wish to implement TLS ALPN validation, they **SHOULD** only generate certificates used for validation themselves and not expose this functionality to users.

The extensions to the ACME protocol described in this document build upon the Security Considerations and threat model defined in Section 10.1 of [RFC8555].

6. IANA Considerations

6.1. SMI Security for PKIX Certificate Extension OID

Within the "Structure of Management Information (SMI) Numbers (MIB Module Registrations)" registry, the following entry has been added to the "SMI Security for PKIX Certificate Extension" (1.3.6.1.5.5.7.1) table.

Decimal	Description	References
31	id-pe-acmeIdentifier	RFC 8737

Table 1

6.2. ALPN Protocol ID

Within the "Transport Layer Security (TLS) Extensions" registry, the following entry has been added to the "TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs" table.

Protocol	Identification Sequence	Reference
acme-tls/1	0x61 0x63 0x6d 0x65 0x2d 0x74 0x6c 0x73 0x2f 0x31 ("acme-tls/1")	RFC 8737

Table 2

6.3. ACME Validation Method

Within the "Automated Certificate Management Environment (ACME) Protocol" registry, the following entry has been added to the "ACME Validation Methods" registry.

Label	Identifier Type	ACME	Reference
tls-alpn-01	dns	Y	RFC 8737

Table 3

7. Normative References

[FIPS180-4] National Institute of Standards and Technology (NIST), "Secure Hash Standard (SHS)", FIPS PUB 180-4, DOI 10.6028/NIST.FIPS.180-4, August 2015, <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf>.

[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>.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <https://www.rfc-editor.org/info/rfc3492>.

[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/info/rfc4086>.

[RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <https://www.rfc-editor.org/info/rfc4343>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[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>.

[RFC6066] Eastlake 3rd, D., "Transport Layer Security (TLS) Extensions: Extension Definitions", RFC 6066, DOI 10.17487/RFC6066, January 2011, <https://www.rfc-editor.org/info/rfc6066>.

[RFC7301] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc7301>.

[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>.

[X.680] ITU-T, "Information technology -- Abstract Syntax Notation One (ASN.1): Specification of basic notation", ITU-T Recommendation X.680, ISO/IEC 8824-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.680-201508-I/en>.

[X.690] ITU-T, "Information Technology -- ASN.1 encoding rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)", ITU-T Recommendation X.690, ISO/IEC 8825-1:2015, August 2015, <https://www.itu.int/rec/T-REC-X.690-201508-I/en>.

Appendix A. Design Rationale

The TLS ALPN challenge exists to iterate on the TLS SNI challenge defined in the early ACME drafts. The TLS SNI challenge was convenient for service providers who were either operating large TLS-layer load balancing systems at which they wanted to perform validation or running servers fronting large numbers of DNS names from a single host as it allowed validation purely within the TLS layer. The value provided by the TLS SNI challenge was considered large enough that this document was written in order to provide a new challenge type that addressed the existing security concerns.

A security issue in the TLS SNI challenge was discovered by Frans Rosen, which allowed users of various service providers to illegitimately validate control of the DNS names of other users of the provider. When the TLS SNI challenge was designed, it was assumed that a user would only be able to respond to TLS traffic via SNI for domain names they had registered with a service provider (i.e., if a user registered 'a.example', they would only be able to respond to SNI requests for 'a.example' and not for SNI requests for 'b.example'). It turns out that a number of large service providers do not honor this property. Because of this, users were able to respond to SNI requests for the names used by the TLS SNI challenge validation process. This meant that (1) if User A and User B had registered Host A and Host B, respectively, User A would be able to claim the constructed SNI challenge name for Host B, and (2) when the validation connection was made, User A would be able to answer, thereby proving 'control' of Host B. As the SNI name used was a subdomain of the domain name being validated, rather than the domain name itself, it was likely to not already be registered with the service provider for traffic routing, making it much easier for a hijack to occur.

Acknowledgments

The author would like to thank all those that provided design insights and editorial review of this document, including Richard Barnes, Ryan Hurst, Adam Langley, Ryan Sleevi, Jacob Hoffman-Andrews, Daniel McCarney, Marcin Walas, Martin Thomson, and especially Frans Rosen, who discovered the vulnerability in the TLS SNI method that necessitated the writing of this specification.

Author's Address

Roland Bracewell Shoemaker
Internet Security Research Group
Email: roland@letsencrypt.org