ACME Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 1, 2018

R. Shoemaker
ISRG

May 30, 2018

ACME TLS ALPN Challenge Extension
draft-ietf-acme-tls-alpn-01

Abstract

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

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 December 1, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction

The Automatic Certificate Management Environment (ACME) [I-D.ietf-acme-acme] standard specifies 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, CDNs, and TLS-terminating load balancers to validate domain control without modifying the HTTP handling behavior of their backends. This separation of layers can improve security and usability of ACME validation.

Early ACME drafts specified two TLS-based challenge types: TLS-SNI-01 and TLS-SNI-02. These methods were removed because they relied on assumptions about the deployed base of HTTPS hosting providers that proved to be incorrect. Those incorrect assumptions weakened the security of those methods and are discussed in the "Design Rationale" appendix.

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 no existing software implements this protocol, the ability to fulfill TLS-ALPN-01 challenges is effectively opt-in. A service provider must proactively deploy new code in order to implement TLS-ALPN-01, so we can specify stronger controls in that code, resulting in a stronger validation method.

2. Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

3. TLS with Application Level Protocol Negotiation (TLS ALPN) Challenge

The TLS with Application Level Protocol Negotiation (TLS ALPN) validation method proves control over a domain name by requiring the client to configure a TLS server to respond to specific connection attempts utilizing 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.

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, including padding characters ("=").

```
GET /acme/authz/1234/1 HTTP/1.1
Host: example.com

HTTP/1.1 200 OK
{
  "type": "tls-alpn-01",
  "url": "https://example.com/acme/authz/1234/1",
  "status": "pending",
  "token": "evaGxfADs6pSRb2LAv9IZf17Dt3juxGJ-PCt92wr-oA"
}
```

The client prepares for validation by constructing a self-signed certificate which MUST contain a acmeValidation-v1 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 acmeValidation-v1 extension MUST contain the SHA-256 digest [FIPS180-4] of the key authorization [I-D.ietf-acme-acme] for the challenge. The acmeValidation extension MUST be critical so that the certificate isn't inadvertently used by non-ACME software.

```
id-pe-acmeIdentifier OBJECT IDENTIFIER ::=  { id-pe 30 }

id-pe-acmeIdentifier-v1 OBJECT IDENTIFIER ::=  { id-pe-acmeIdentifier 1 }

acmeValidation-v1 ::= OCTET STRING (SIZE (32))
```

Once this certificate has been created it MUST be provisioned such that it is returned during a TLS handshake that contains a ALPN extension containing the value "acme-tls/1" and a SNI extension containing the domain name being validated.

A client responds with an empty object ({}) to acknowledge that the challenge is ready to be validated by the server.

```
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 a response 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. Compute the expected SHA-256 digest of the expected key authorization.

2. Resolve the domain name being validated and choose one of the IP addresses returned for validation (the server MAY validate against multiple addresses if more than one is returned, but this is not required).

3. Initiate a TLS connection with the chosen IP address, this connection MUST use TCP port 443. The ClientHello that initiates the handshake MUST contain a ALPN extension with the single

protocol name "acme-tls/1" and a Server Name Indication [RFC6066] extension containing the domain name being validated.

4. Verify that the ServerHello contains a ALPN extension containing the value "acme-tls/1" and that the certificate returned contains a subjectAltName extension containing the dNSName being validated and no other entries and a critical acmeValidation extension containing the 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 [RFC3492] rules.

If all of the above steps succeed then the validation is successful, otherwise it fails. Once the TLS handshake has been completed the connection MUST be immediately closed and no further data should be exchanged.

3.1. 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. Once the handshake is completed the client MUST NOT exchange any further data with the server and MUST immediately close the connection.

4. Security Considerations

The design of this challenges relies on some assumptions centered around how a server behaves during validation.

The first assumption is that when a server is being used to serve content for multiple DNS names from a single IP address that 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 server should not allow a TLS request using a SNI value for Host A to be served by User B or Host B to be served by User A. If the 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.

5. IANA Considerations

5.1. SMI Security for PKIX Certificate Extension OID

Within the SMI-numbers registry, the "SMI Security for PKIX Certificate Extension (1.3.6.1.5.5.7.1)" table is to be updated to add the following entry:

Decimal	Description	References
30	id-pe-acmeIdentifier	RFC XXXX

5.2. ALPN Protocol ID

Within the Transport Layer Security (TLS) Extensions registry, the "Application-Layer Protocol Negotiation (ALPN) Protocol IDs" table is to be updated to add the following entry:

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

5.3. ACME Validation Method

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

Label	Identifier Type	Reference
tls-alpn-01	dns	RFC XXXX

6. Appendix: Design Rationale

The TLS ALPN challenge exists to replace the TLS SNI challenge defined in the early ACME drafts. This 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.

A security issue was discovered in the TLS SNI challenge 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 controlled (i.e. if User A registered Host A and User B registered Host B with a service provider that User A wouldn't be able to respond to SNI traffic for Host B). This turns out not to be a security property provided by a number of large service providers. Because of this users were able to respond to SNI traffic for the SNI names used by the TLS SNI challenge validation process. This meant that if User A and User B had registered Host A and Host B respectively User A would be able to claim the SNI name for a validation for Host B and when the validation connection was made that User A would be able to answer, proving control of Host B.

7. Acknowledgements

The author would like to thank all those whom have 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, and Martin Thomson and especially Frans Rosen who discovered the vulnerability in the TLS SNI method which necessitated the writing of this specication.

8. Normative References

[FIPS180-4] Department of Commerce, National., "NIST FIPS 180-4, Secure Hash Standard", March 2012, <http://csrc.nist.gov/publications/fips/fips180-4/fips-180-4.pdf>.

[I-D.ietf-acme-acme] Barnes, R., Hoffman-Andrews, J., McCarney, D., and J. Kasten, "Automatic Certificate Management Environment (ACME)", draft-ietf-acme-acme-12 (work in progress), April 2018.

[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>.

[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>.

[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>.

Author's Address

Roland Bracewell Shoemaker
Internet Security Research Group

Email: roland@letsencrypt.org