Internet Engineering Task Force (IETF)
Request for Comments: 7711
Category: Standards Track
ISSN: 2070-1721

M. Miller
Cisco Systems, Inc.
P. Saint-Andre
&yet

November 2015

PKIX over Secure HTTP (POSH)

Abstract

Experience has shown that it is difficult to deploy proper PKIX certificates for Transport Layer Security (TLS) in multi-tenanted environments. As a result, domains hosted in such environments often deploy applications using certificates that identify the hosting service, not the hosted domain. Such deployments force end users and peer services to accept a certificate with an improper identifier, resulting in degraded security. This document defines methods that make it easier to deploy certificates for proper server identity checking in non-HTTP application protocols. Although these methods were developed for use in the Extensible Messaging and Presence Protocol (XMPP) as a Domain Name Association (DNA) prooftype, they might also be usable in other non-HTTP application protocols.

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 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7711.

Copyright Notice

Table of Contents

1. Introduction

We begin with a thought experiment.

Imagine that you work on the operations team of a hosting company that provides instances of the hypothetical "Secure Protocol for Internet Content Exchange" (SPICE) service for ten thousand different customer organizations. Each customer wants their instance to be identified by the customer's domain name (e.g., bar.example.com), not the hosting company's domain name (e.g., hosting.example.net).

In order to properly secure each customer's SPICE instance via Transport Layer Security (TLS) [RFC5246], you need to obtain and deploy PKIX certificates [RFC5280] containing identifiers such as bar.example.com, as explained in the "CertID" specification [RFC6125]. Unfortunately, you can't obtain and deploy such certificates because:

- Certification authorities won't issue such certificates to you because you work for the hosting company, not the customer organization.
- Customers won't obtain such certificates and then give them (plus the associated private keys) to you because their legal department is worried about liability.
- You don't want to install such certificates (plus the associated private keys) on your servers because your legal department is worried about liability, too.
- Even if your legal department is happy, this still means managing one certificate for each customer across the infrastructure, contributing to a large administrative load.

Given your inability to obtain and deploy public keys / certificates containing the right identifiers, your back-up approach has always been to use a certificate containing hosting.example.net as the identifier. However, more and more customers and end users are complaining about warning messages in user agents and the inherent security issues involved with taking a "leap of faith" to accept the identity mismatch between the source domain (bar.example.com) and the delegated domain (hosting.example.net) [RFC6125].

This situation is both insecure and unsustainable. You have investigated the possibility of using DNS Security [RFC4033] and DNS-Based Authentication of Named Entities (DANE) [RFC6698] to solve the problem. However, your customers and your operations team have told you that it will be several years before they will be able to

deploy DNSSEC and DANE for all of your customers (because of tooling updates, slow deployment of DNSSEC at some top-level domains, etc.). The product managers in your company are pushing you to find a method that can be deployed more quickly to overcome the lack of proper server identity checking for your hosted customers.

One possible approach that your team has investigated is to ask each customer to provide the public key / certificate for its SPICE service at a special HTTPS URI on their website ("https://bar.example.com/.well-known/posh/spice.json" is one possibility). This could be a public key that you generate for the customer, but because the customer hosts it via HTTPS, any user agent can find that public key and check it against the public key you provide during TLS negotiation for the SPICE service (as one added benefit, the customer never needs to hand you a private key). Alternatively, the customer can redirect requests for that special HTTPS URI to an HTTPS URI at your own website, thus making it explicit that they have delegated the SPICE service to you.

The approach sketched out above, called POSH ("PKIX over Secure HTTP"), is explained in the remainder of this document. Although this approach was developed for use in the Extensible Messaging and Presence Protocol (XMPP) as a prooftype for Domain Name Associations (DNA) [RFC7712], it might be usable by any non-HTTP application protocol.

2. Terminology

This document inherits security terminology from [RFC5280]. The terms "source domain", "delegated domain", "derived domain", and "reference identifier" are used as defined in the "CertID" specification [RFC6125].

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 [RFC2119].

Additionally, this document uses the following terms:

POSH client: A client that uses the application service and that uses POSH to obtain material for verifying the service's identity.

POSH server: A server that hosts the application service and that uses POSH to provide material for verifying its identity.

3. Obtaining Verification Material

Server identity checking (see [RFC6125]) involves three different aspects:

1. A proof of the POSH server's identity (in PKIX, this takes the form of a PKIX end-entity certificate [RFC5280]).

2. Rules for checking the certificate (which vary by application protocol, although [RFC6125] attempts to harmonize those rules).

3. The material that a POSH client uses to verify the POSH server's identity or check the POSH server's proof (in PKIX, this takes the form of chaining the end-entity certificate back to a trusted root and performing all validity checks as described in [RFC5280], [RFC6125], and the relevant application protocol specification).

When POSH is used, the first two aspects remain the same: the POSH server proves its identity by presenting a PKIX certificate [RFC5280], and the certificate is checked according to the rules defined in the appropriate application protocol specification (such as [RFC6120] for XMPP). However, the POSH client obtains the material it will use to verify the server's proof by retrieving a JSON document [RFC7159] containing hashes of the PKIX certificate over HTTPS ([RFC7230] and [RFC2818]) from a well-known URI [RFC5785] at the source domain. POSH servers MUST use HTTPS. This means that the POSH client MUST verify the certificate of the HTTPS service at the source domain in order to securely "bootstrap" into the use of POSH; specifically, the rules of [RFC2818] apply to this "bootstrapping" step to provide a secure basis for all subsequent POSH operations.

A PKIX certificate is retrieved over secure HTTP in the following way:

1. The POSH client performs an HTTPS GET request at the source domain to the path "/.well-known/posh/{servicedesc}.json". The value of "{servicedesc}" is application-specific; see Section 8 of this document for more details. For example, if the application protocol is the hypothetical SPICE service, then "{servicedesc}" could be "spice"; thus, if an application client were to use POSH to verify an application server for the source domain "bar.example.com", the HTTPS GET request would be as follows:

```
GET /.well-known/posh/spice.json HTTP/1.1
Host: bar.example.com
```

2. The source domain HTTPS server responds in one of three ways:

   * If it possesses PKIX certificate information for the requested path, it responds as detailed in Section 3.1.

   * If it has a reference to where the PKIX certificate information can be obtained, it responds as detailed in Section 3.2.

   * If it does not have any PKIX certificate information or a reference to such information for the requested path, it responds with an HTTP 404 Not Found status code [RFC7231].

3.1. Source Domain Possesses PKIX Certificate Information

If the source domain HTTPS server possesses the certificate information, it responds to the HTTPS GET request with a success status code and the message body set to a JSON document [RFC7159]; the document is a "fingerprints document", i.e., a JSON object with the following members:

- A "fingerprints" member whose value is a JSON array of fingerprint descriptors (the member MUST include at least one fingerprint descriptor).

- An "expires" member whose value is a JSON number specifying the number of seconds after which the POSH client ought to consider the keying material to be stale (further explained under Section 6).

The JSON document returned MUST NOT contain a "url" member, as described in Section 3.2.

Each included fingerprint descriptor is a JSON object, where each member name is the textual name of a hash function (as listed in [HASH-NAMES]) and its associated value is the base64-encoded fingerprint hash generated using the named hash function (where the encoding adheres to the definition in Section 4 of [RFC4648] and where the padding bits are set to zero).

The fingerprint hash for a given hash algorithm is generated by performing the named hash function over the DER encoding of the PKIX X.509 certificate. (This implies that if the certificate expires or is revoked, the fingerprint value will be out of date.)

As an example of the fingerprint format, the "sha-256" and "sha-512" fingerprints are generated by performing the SHA-256 and SHA-512 hash functions, respectively, over the DER encoding of the PKIX certificate, as illustrated below. Note that for readability whitespace has been added to the content portion of the HTTP response shown below but is not reflected in the Content-Length.

Example Fingerprints Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 195

{
  "fingerprints": [
    {
      "sha-256": "4/mggdlVx8A3pvHAWW5sD+qJyMtUHgiRuPjVC48N0XQ=",
      "sha-512": "25N+1hB2Vo42l9lSGqw+n3BKFhDHsyork8ou+D9B43TXeJ
                  1J81mdQEDqm39oR/EHkPBDDG1y5+AG94Kec0xVqA=="
    }
  ],
  "expires": 604800
}
```

The "expires" value is a hint regarding the expiration of the keying material. It MUST be a non-negative integer. If the "expires" member has a value of 0 (zero), a POSH client MUST consider the verification material to be invalid. See Section 6 for how to reconcile this "expires" member with the reference's "expires" member.

To indicate alternate PKIX certificates (such as when an existing certificate will soon expire), the returned fingerprints member MAY contain multiple fingerprint descriptors. The fingerprints SHOULD be ordered with the most relevant certificate first as determined by the application service operator (e.g., the renewed certificate), followed by the next most relevant certificate (e.g., the certificate soonest to expire). Here is an example (note that whitespace is added for readability):

```
{
  "fingerprints": [
    {
      "sha-256": "4/mggdlVx8A3pvHAWW5sD+qJyMtUHgiRuPjVC48N0XQ",
      "sha-512": "25N+1hB2Vo42l9lSGqw+n3BKFhDHsyork8ou+D9B43TXe
                  J1J81mdQEDqm39oR/EHkPBDDG1y5+AG94Kec0xVqA=="
    },
    {
      "sha-256": "otyLADSKjRDjVpj8X7/hmCAD5C7Qe+PedcmYV7cUncE=",
      "sha-512": "MbBD+ausTGJisEXKSynROWrMfHP2xvBnmI79Pr/KXnDyLN
                  +13Jof8/Uq9fj5HZG8Rk1E2fclcivpGdijUsvHRg=="
    }
  ],
  "expires": 806400
}
```

Matching on any of these fingerprints is acceptable.

Rolling over from one hosting provider to another is best handled by updating the relevant SRV records, not primarily by updating the POSH documents themselves.

3.2. Source Domain References PKIX Certificate

If the source domain HTTPS server has a reference to the certificate information, it responds to the HTTPS GET request with a success status code and message body set to a JSON document. The document is a "reference document", i.e., a JSON object with the following members:

- A "url" member whose value is a JSON string specifying the HTTPS URI where POSH clients can obtain the actual certificate information. The URI can be a well-known POSH URI as described in Section 8, but it need not be. (For historical reasons, the member name is "url", not "uri".)

- An "expires" member whose value is a JSON number specifying the number of seconds after which the POSH client ought to consider the delegation to be stale (further explained under Section 6).

Example Reference Response

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 82

{
  "url":"https://hosting.example.net/.well-known/posh/spice.json",
  "expires":86400
}
```

In order to process a reference response, the client performs an HTTPS GET request for the URI specified in the "url" member value. The HTTPS server for the URI to which the client has been referred responds to the request with a JSON document containing fingerprints as described in Section 3.1. The document retrieved from the location specified by the "url" member MUST NOT itself be a reference document (i.e., containing a "url" member instead of a "fingerprints" member), in order to prevent circular delegations.

Note: See Section 10 for discussion about HTTPS redirects.

The "expires" value is a hint regarding the expiration of the source domain's delegation of service to the delegated domain. It MUST be a non-negative integer. If the "expires" member has a value of 0 (zero), a POSH client MUST consider the delegation invalid. See Section 6 for guidelines about reconciling this "expires" member with the "expires" member of the fingerprints document.

3.3. Performing Verification

The POSH client compares the PKIX information presented by the POSH server against each fingerprint descriptor object in the POSH fingerprints document, until a match is found using the hash functions that the client supports, or until the collection of POSH verification material is exhausted. If none of the fingerprint descriptor objects match the POSH server PKIX information, the POSH client SHOULD reject the connection (however, the POSH client might still accept the connection if other verification methods are successful, such as DANE [RFC6698]).

4. Secure Delegation

The delegation from the source domain to the delegated domain can be considered secure if the credentials offered by the POSH server match the verification material obtained by the client, regardless of how the material was obtained.

5. Order of Operations

In order for the POSH client to perform verification of reference identifiers without potentially compromising data, POSH operations MUST be complete before any application-layer data is exchanged for the source domain. In cases where the POSH client initiates an application-layer connection, the client SHOULD perform all POSH retrievals before initiating a connection (naturally, this is not possible in cases where the POSH client receives instead of initiates an application-layer connection). For application protocols that use DNS SRV (including queries for TLSA records in concert with SRV records as described in [RFC7673]), the POSH operations ideally ought to be done in parallel with resolving the SRV records and the addresses of any targets, similar to the "Happy Eyeballs" approach for IPv4 and IPv6 [RFC6555].

The following diagram illustrates the possession flow:

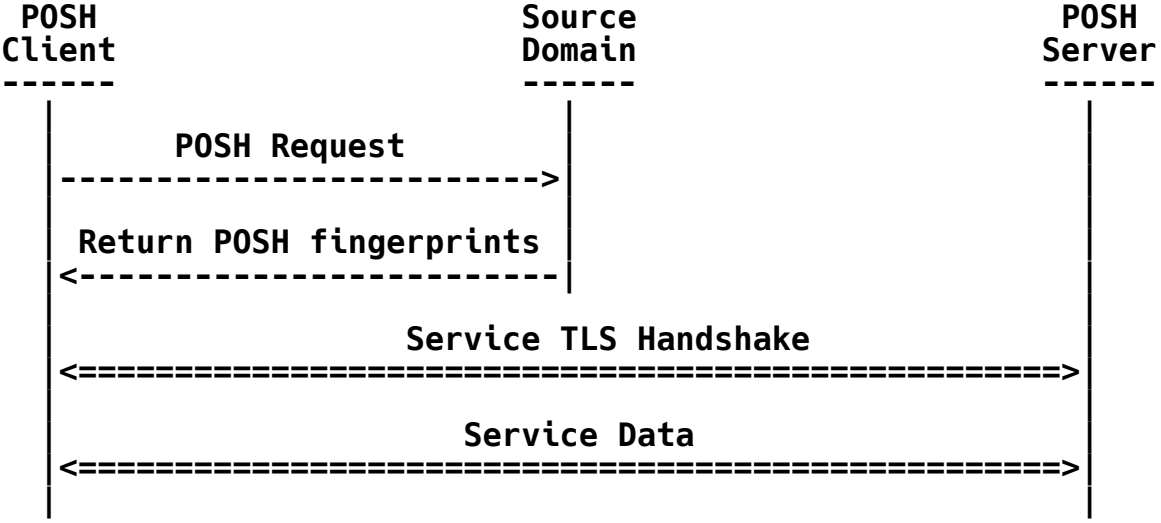

Figure 1: Order of Events for Possession Flow

While the following diagram illustrates the reference flow:

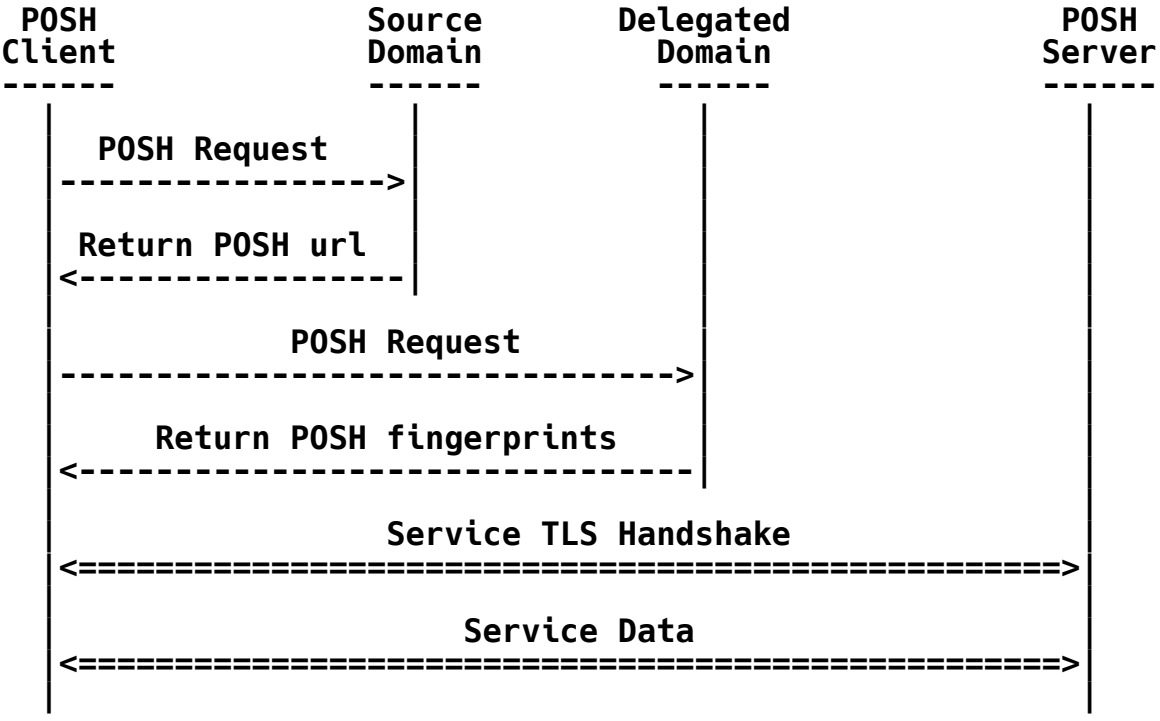

Figure 2: Order of Events for Reference Flow

6. Caching Results

The POSH client MUST NOT cache results (reference or fingerprints) indefinitely. If the source domain returns a reference, the POSH client MUST use the lower of the two "expires" values when determining how long to cache results (i.e., if the reference "expires" value is lower than the fingerprints "expires" value, honor the reference "expires" value). Once the POSH client considers the results stale, it needs to perform the entire POSH operation again, starting with the HTTPS GET request to the source domain. The POSH client MAY use a lower value than any provided in the "expires" member(s), or not cache results at all.

The foregoing considerations apply to the handling of the "expires" values in POSH documents; naturally, a POSH client MUST NOT consider an expired PKIX certificate to be valid, in accordance with [RFC5280].

The POSH client SHOULD NOT rely on HTTP caching mechanisms, instead using the expiration hints provided in the POSH reference document or fingerprints document. To that end, the HTTPS servers for source domains and derived domains SHOULD specify a 'Cache-Control' header indicating a very short duration (e.g., max-age=60) or "no-cache" to indicate that the response (redirect, reference, or fingerprints) is not appropriate to cache at the HTTP layer.

7. Guidance for Server Operators

POSH is intended to ease the operational burden of securing application services, especially in multi-tenanted environments. It does so by obviating the need to obtain certificates for hosted domains, so that an operator can obtain a certificate only for its hosting service (naturally, this certificate needs to be valid according to [RFC5280] and contain the proper identifier(s) in accordance with [RFC6125] and the relevant application protocol specification).

However, in order to use POSH, an operator does need to coordinate with its customers so that the appropriate POSH documents are provided via HTTPS at a well-known URI at each customer's domain (i.e., at the source domain), thus ensuring delegation to the operator's hosting service (i.e., the delegated domain). Because correct hosting of the POSH document at the source domain is essential for successful functioning of the POSH "chain", errors at the source domain will result in authentication problems, certificate warnings, and other operational issues.

Furthermore, if the POSH document is a reference document instead of a fingerprints document, the operational burden is further decreased because the operator does not need to provision its customers with updated POSH documents when the certificate for the delegated domain expires or is replaced.

8. Guidance for Protocol Authors

Protocols that use POSH are expected to register with the "POSH Service Names" registry defined under Section 9.2.

For POSH-using protocols that rely on DNS SRV records [RFC2782], the service name SHOULD be the same as the DNS SRV "Service". As an example, the POSH service name for XMPP server-to-server connections would be "xmpp-server" because [RFC6120] registers a DNS SRV "Service" of "xmpp-server". One example of the resulting well-known URI would be "https://example.com/.well-known/posh/xmpp-server.json".

For other POSH-using protocols, the service name MAY be any unique string or identifier for the protocol; for example, it might be a service name registered with the IANA in accordance with [RFC6335], or it might be an unregistered name. As an example, the well-known URI for the hypothetical SPICE application might be "spice".

9. IANA Considerations

9.1. Well-Known URI

IANA has registered "posh" in the "Well-Known URIs" registry as defined by [RFC5785]. The completed template follows.

URI suffix: posh

Change controller: IETF

Specification: RFC 7711 (this document)

Related information: The suffix "posh" is expected to be followed by an additional path component consisting of a service name (say, "spice") and a file extension of ".json", resulting in a full path of, for instance, "/.well-known/posh/spice.json". Registration of service names shall be requested by developers of the relevant application protocols.

9.2. POSH Service Names

IANA has established the "POSH Service Names" registry within the "Uniform Resource Identifier (URI) Schemes" group of registries.

The IANA registration policy [RFC5226] is Expert Review or IETF Review (this was chosen instead of the more liberal policy of First Come First Served to help ensure that POSH services are defined in ways that are consistent with this specification). One or more Designated Experts are to be appointed by the IESG or their delegate.

Registration requests are to be sent to the posh@ietf.org mailing list for review and comment, with an appropriate subject (e.g., "Request for POSH service name: example").

Before a period of 14 days has passed, the Designated Expert(s) will either approve or deny the registration request, communicating this decision both to the review list and to IANA. Denials should include an explanation and, if applicable, suggestions as to how to make the request successful. Registration requests that are undetermined for a period longer than 21 days can be brought to the IESG's attention (using the iesg@iesg.org mailing list) for resolution.

9.2.1. Registration Template

Service name: The name requested, relative to "/.well-known/posh/"; e.g., a service name of "example" would result in a well-known URI such as "https://example.com/.well-known/posh/example.json".

Change controller: For Standards Track RFCs, state "IETF". In all other cases, provide the name and email address of the responsible party. Other details (e.g., postal address or website URI) may also be included.

Definition and usage: A brief description that defines the service name and mentions where and how it is used (e.g., in the context of a particular application protocol).

Specification: Optionally, reference to a document that specifies the service or application protocol that uses the service name, preferably including a URI that can be used to retrieve a copy of the document. An indication of the relevant sections may also be included but is not required.

10. Security Considerations

This document supplements but does not supersede the security considerations provided in specifications for application protocols that decide to use POSH (e.g., [RFC6120] and [RFC6125] for XMPP). Specifically, the security of requests and responses sent via HTTPS depends on checking the identity of the HTTP server in accordance with [RFC2818] as well as following the most modern best practices for TLS as specified in [RFC7525]. Additionally, the security of POSH can benefit from other HTTP-hardening protocols, such as HTTP Strict Transport Security (HSTS) [RFC6797] and key pinning [RFC7469], especially if the POSH client shares some information with a common HTTPS implementation (e.g., a platform-default web browser).

Note well that POSH is used by a POSH client to obtain the public key of a POSH server to which it might connect for a particular application protocol such as IMAP or XMPP. POSH does not enable a hosted domain to transfer private keys to a hosting service via HTTPS. POSH also does not enable a POSH server to engage in certificate enrollment with a certification authority via HTTPS, as is done in Enrollment over Secure Transport [RFC7030].

A web server at the source domain might redirect an HTTPS request to another HTTPS URI. The location provided in the redirect response MUST specify an HTTPS URI. Source domains SHOULD use only temporary redirect mechanisms, such as HTTP status codes 302 (Found) and 307 (Temporary Redirect) [RFC7231]. Clients MAY treat any redirect as

temporary, ignoring the specific semantics for 301 (Moved Permanently) [RFC7231] and 308 (Permanent Redirect) [RFC7538]. To protect against circular references, it is RECOMMENDED that POSH clients follow no more than 10 redirects, although applications or implementations can require that fewer redirects be followed.

Hash function agility is an important quality to ensure secure operations in the face of attacks against the fingerprints obtained within verification material. Because POSH verification material is relatively short-lived compared to long-lived credentials such as PKIX end-entity certificates (at least as typically deployed), entities that deploy POSH are advised to swap out POSH documents if the hash functions are found to be subject to practical attacks [RFC4270].

11. References

11.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <http://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, <http://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, <http://www.rfc-editor.org/info/rfc5280>.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, DOI 10.17487/RFC5785, April 2010, <http://www.rfc-editor.org/info/rfc5785>.

[RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <http://www.rfc-editor.org/info/rfc6125>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7230] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

[RFC7231] Fielding, R., Ed., and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

11.2. Informative References

[HASH-NAMES] "Hash Function Textual Names", <http://www.iana.org/assignments/hash-function-text-names>.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <http://www.rfc-editor.org/info/rfc2782>.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <http://www.rfc-editor.org/info/rfc4033>.

[RFC4270] Hoffman, P. and B. Schneier, "Attacks on Cryptographic Hashes in Internet Protocols", RFC 4270, DOI 10.17487/RFC4270, November 2005, <http://www.rfc-editor.org/info/rfc4270>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120, March 2011, <http://www.rfc-editor.org/info/rfc6120>.

[RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S. Cheshire, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", BCP 165, RFC 6335, DOI 10.17487/RFC6335, August 2011, <http://www.rfc-editor.org/info/rfc6335>.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <http://www.rfc-editor.org/info/rfc6555>.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <http://www.rfc-editor.org/info/rfc6698>.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTP Strict Transport Security (HSTS)", RFC 6797, DOI 10.17487/RFC6797, November 2012, <http://www.rfc-editor.org/info/rfc6797>.

[RFC7030] Pritikin, M., Ed., Yee, P., Ed., and D. Harkins, Ed., "Enrollment over Secure Transport", RFC 7030, DOI 10.17487/RFC7030, October 2013, <http://www.rfc-editor.org/info/rfc7030>.

[RFC7469] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", RFC 7469, DOI 10.17487/RFC7469, April 2015, <http://www.rfc-editor.org/info/rfc7469>.

[RFC7538] Reschke, J., "The Hypertext Transfer Protocol Status Code 308 (Permanent Redirect)", RFC 7538, DOI 10.17487/RFC7538, April 2015, <http://www.rfc-editor.org/info/rfc7538>.

[RFC7673] Finch, T., Miller, M., and P. Saint-Andre, "Using DNS-Based Authentication of Named Entities (DANE) TLSA Records with SRV Records", RFC 7673, DOI 10.17487/RFC7673, October 2015, <http://www.rfc-editor.org/info/rfc7673>.

[RFC7712] Saint-Andre, P., Miller, M., and P. Hancke, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", RFC 7712, DOI 10.17487/RFC7712, November 2015, <http://www.rfc-editor.org/info/rfc7712>.

Acknowledgements

Thanks to Thijs Alkemade, Philipp Hancke, Joe Hildebrand, and Tobias Markmann for their implementation feedback, and to Dave Cridland, Chris Newton, Max Pritikin, and Joe Salowey for their input on the specification.

During IESG review, Stephen Farrell, Barry Leiba, and Kathleen Moriarty provided helpful input that resulted in improvements in the document.

Thanks also to Dave Cridland as document shepherd, Joe Hildebrand as working group chair, and Ben Campbell as area director.

Peter Saint-Andre wishes to acknowledge Cisco Systems, Inc., for employing him during his work on earlier draft versions of this document.

Authors' Addresses

Matthew Miller
Cisco Systems, Inc.
1899 Wynkoop Street, Suite 600
Denver, CO 80202
United States

Email: mamille2@cisco.com

Peter Saint-Andre
&yet

Email: peter@andyet.com
URI: https://andyet.com/