XMPP Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 8, 2014

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

February 4, 2014

PKIX over Secure HTTP (POSH)
draft-ietf-xmpp-posh-00

Abstract

Experience has shown that it is extremely difficult to deploy proper PKIX certificates for TLS in multi-tenanted environments, since certification authorities will not issue certificates for hosted domains to hosting services, hosted domains do not want hosting services to hold their private keys, and hosting services wish to avoid liability for holding those keys. As a result, domains hosted in multi-tenanted environments often deploy non-HTTP applications such as email and instant messaging 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 obvious security implications. This document defines two methods that make it easier to deploy certificates for proper server identity checking in non-HTTP application protocols. The first method enables the TLS client associated with a user agent or peer application server to obtain the end-entity certificate of a hosted domain over secure HTTP as an alternative to standard PKIX techniques. The second method enables a hosted domain to securely delegate a non-HTTP application to a hosting service using redirects provided by HTTPS itself or by a pointer in a file served over HTTPS at the hosted domain. While this approach is developed for use in the Extensible Messaging and Presence Protocol (XMPP) as a Domain Name Association prooftype, it can be applied to any non-HTTP application protocol.

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 http://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 August 8, 2014.

Table of Contents

1. Introduction

We start with a thought experiment.

Imagine that you work on the operations team of a hosting company that provides the "foo" service (or email or instant messaging or social networking service) for ten thousand different customer

organizations. Each customer wants their service to be identified by the customer's domain name (e.g., foo.example.com), not the hosting company's domain name (e.g., hosting.example.net).

In order to properly secure each customer's "foo" service via Transport Layer Security (TLS) [RFC5246], you need to obtain PKIX certificates [RFC5280] containing identifiers such as foo.example.com, as explained in the "CertID" specification [RFC6125]. Unfortunately, you can't obtain 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 anyway because your legal department is worried about liability, too.

Given your inability to deploy public keys / certificates containing the right identifiers, your back-up approach was always 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 (foo.example.com) and the delegated domain (hosting.example.net).

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 they will not be able to deploy DNSSEC and DANE for several years at least. 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 is to ask each customer to provide the public key / certificate for the "foo" service at a special HTTPS URI on their website ("https://foo.example.com/.well-known/posh.foo.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 "foo" 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 "foo" service to you.

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

2. Discussion Venue

The discussion venue for this document is the posh@ietf.org mailing list; visit https://www.ietf.org/mailman/listinfo/posh for subscription information and discussion archives.

3. Terminology

This document inherits security terminology from [RFC5280]. The terms "source domain", "derived domain", "reference identifier", and "presented 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].

4. Obtaining Verification Materials

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

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

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

3. The materials that a TLS client uses to verify the TLS server's identity or check the TLS 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 TLS server proves it 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 TLS client obtains the materials it will use to verify the server's proof by retrieving a JSON Web Key (JWK) set [JOSE-JWK] over HTTPS ([RFC2616] and [RFC2818]) from a well-known URI [RFC5785].

The process for retrieving a PKIX certificate over secure HTTP is as follows.

1. The TLS client performs an HTTPS GET at the source domain to the path "/.well-known/posh.{servicedesc}.json". The value of "{servicedesc}" is application-specific; see Section 9 of this document for more details. For example, if the application protocol is some hypothetical "Foo" service, then "{servicedesc}" could be "foo"; thus if a Foo client were to use POSH to verify a Foo server for the domain "foo.example.com", the HTTPS GET request would be as follows:

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

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

   * If it possesses a PKIX certificate for the requested path, it responds as detailed in Section 4.1.

   * If it has a reference to where the PKIX certificate can be obtained, it responds as detailed in Section 4.2.

   * If it does not have any PKIX certificate for the requested path, it responds with a client error status code (e.g., 404).

4.1. Source Domain Possesses PKIX Certificate

If the source domain HTTPS server possesses the certificate information, it responds to the HTTPS GET with a success status code and the message body set to a JSON Web Key (JWK) set [JOSE-JWK]. The JWK set MUST contain at least one JWK object, and MUST contain an "expires" field whose value is the number of seconds after which the TLS client ought to consider the key information to be stale (further explained under Section 7).

Each included JWK object MUST possess the following information:

- The "kty" field set to the appropriate key type used for TLS connections (e.g., "RSA" for a certificate using an RSA key).

- The required public parameters for the key type (e.g., "n" and "e" for a certificate using an RSA key).

- The "x5t" field set to the certificate thumbprint, as described in section 3.6 of [JOSE-JWK].

Each JWK object MUST NOT possess the private parameters for the key type (e.g., "d", "p", "q" for a certificate using an RSA key).

Each JWK object MAY possess other parameters as desired by application servers (e.g., the "x5c" field containing the entire X.509 certificate chain, as per section 3.7 of [JOSE-JWK]).

The following example illustrates the usage described above.

Example Content Response

```
HTTP/1.1 200 OK
Content-Type: application/jwk-set+json
Content-Length: 2785

{
  "keys": [
    {
      "kty":"RSA",
      "kid":"c8fb8b80-1193-11e3-b2b1-835742119fe8",
      "n":"ANxwssdcU3LbODErec3owrwUhlzjtuskAn8rAcBMRPImn5xA
           JRX-1T5g2D7MTozWWFk4TlpgzAR5slvM0tc35qAI9I0Cqk4Z
           LChQrYsWuY7alTrnNXdusHUYc6Eq89DZaH2knTcp57wAXzJP
           IG_tpBi5F7ck9LVRvRjybix0HJ7i4YrL-GeLuSgrjO4-GDcX
           Ip8oV0FMKZH-NoMfUITlWYl_JcX1D0WUAiuAnvWtD4Kh_qMJ
           U6FZuupZGHqPdc3vrXtp27LWgxzxjFa9qnOU6y53vCCJXLLI
           5sy2fCwEDzLJqh2T6UItIzjrSUZMIsK8r2pXkroI0uYuNn3W
           y-jAzK8",
      "e":"AQAB",
      "x5t":"UpjRI_A3afKE8_AIeTZ5o1dECTY"
    }
  ],
  "expires": 604800
}
```

The "expires" value is a hint regarding the expiration of the keying materials. If no "expires" field is included, a TLS client SHOULD consider these verification materials invalid. See Section 7 for how to reconcile this "expires" field with the reference's "expires" field.

4.2. Source Domain References PKIX Certificate

If the source domain HTTPS server has a reference to the certificate information, it responds to the HTTPS GET with a JSON document. The document MUST contain a "url" field whose value is the HTTPS URL where TLS clients can obtain the actual JWK set, and MUST contain an "expires" field whose value is the number of seconds after which the TLS client ought to consider the delegation to be stale (further explained under Section 7).

Example Reference Response

```
HTTP/1.1 200 Ok
Content-Type: application/json
Content-Length: 78

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

The client performs an HTTPS GET for the URL specified in the "url" field value. The HTTPS server for the URI to which the client has been redirected responds to the request with a JWK set. The content retrieved from the "url" location MUST NOT itself be a reference (i.e., containing a "url" fields instead of a "keys" field), in order to prevent circular delegations.

> Note: The JSON document returned by the source domain HTTPS server MUST contain either a reference or a JWK-set, but MUST NOT contain both.
>
> 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. If no "expires" field is included, a TLS client SHOULD consider the delegation invalid. See Section 7 for guidelines about reconciling this "expires" field with the JWK-set's "expires" field.

4.3. Performing Verification

The TLS client compares the PKIX information obtained from the TLS server against each JWK object in the POSH results, until a match is found or the collection of POSH verification materials is exhausted. If none of the JWK objects match the TLS server PKIX information, the TLS client SHOULD reject the connection (the TLS client might still accept the connection if other verification schemes are successful).

The TLS client SHOULD compare the fingerprint of the PKIX certificate from the TLS server against the "x5t" field of the JWK object (note the "x5t" field is the base64url encoding of the fingerprint).

The TLS client MAY verify the certificate chain provided in the "x5c" field of the JWK object (if present), but it MUST NOT implicitly consider the final certificate in the "x5c" field to be a trust anchor itself; the TLS client only uses the end entity certificate information for verification.

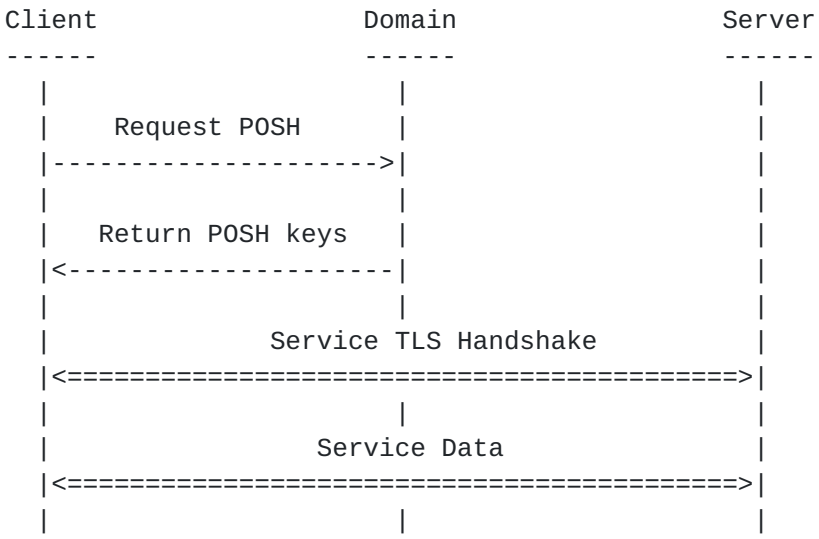
5. Secure Delegation

The delegation from the source domain to the delegated domain can be considered secure if the certificate offered by the TLS server matches the POSH certificate, regardless of how the POSH certificates are obtained.

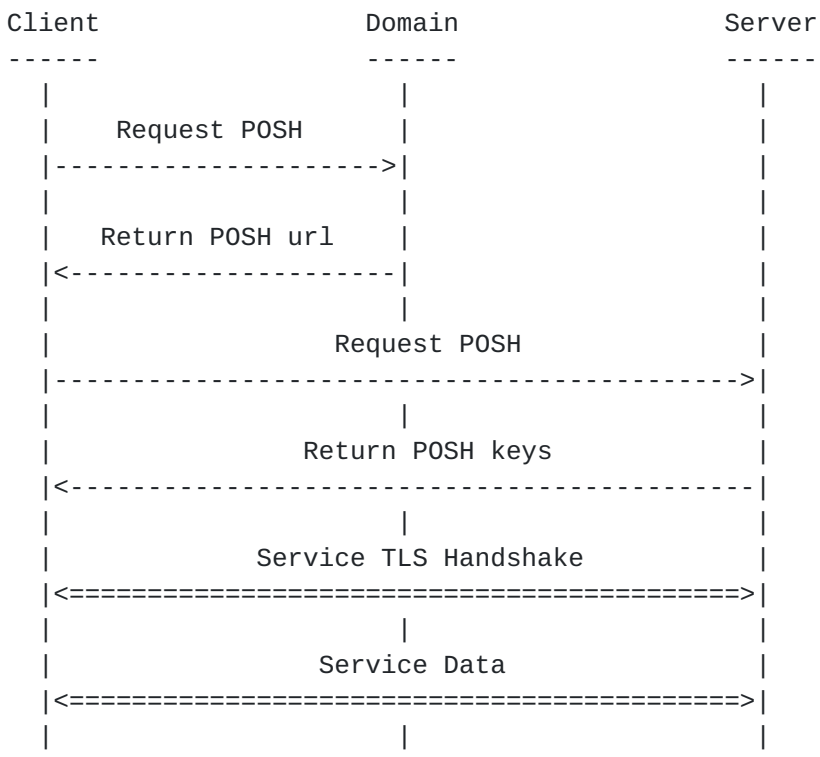
6. Order of Operations

In order for the TLS client to perform verification of reference identifiers without potentially compromising data, POSH processes MUST be complete before any application-level data is exchanged for the source domain. The TLS client SHOULD perform all POSH retrievals before opening any socket connections to the application protocol server. For application protocols that use DNS SRV, the POSH processes 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:



While the following diagram illustrates the reference flow:



7. Caching Results

The TLS client MUST NOT cache results (reference or JWK-set) indefinitely. If the source domain returns a reference, the TLS 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 JWK-set "expires" value, honor the reference "expires" value). Once the TLS client considers the results stale, it SHOULD perform the entire POSH process again starting with the HTTPS GET to the source domain. The TLS client MAY use a lower value than any provided in the "expires" field(s), or not cache results at all.

The TLS client SHOULD NOT rely on HTTP caching mechanisms, instead using the expiration hints provided in the POSH reference or JWK-set documents. To that end, the HTTPS servers for source 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 content) is not appropriate to cache at the HTTP level.

8. Alternates and Roll-over

To indicate alternate PKIX certificates (such as when an existing certificate will soon expire), the returned JWK set MAY contain multiple JWK objects. The JWK set 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:


```
{
  "keys":[
    {
      "kty": "RSA",
      "kid": "cfc0ca70-1193-11e3-b2b1-835742119fe8",
      "n":   "AM-ktWkQ8btj_HEdAA6kOpzJGgoHNZsJmxjh_PifpgAUfQeq
              MO_YBR100IdJZRzJfULyhRwn9bikCq87WToxgPWOnd3sH3qT
              YiAcIR5S6tBbsyp6WYmwM1yuC0vLCo6SoDzdK1SvkQKM3QWk
              0GFNU4l4qXYAMxaSw83i6yv5DBVbST7E92vS6Gq_4pgI26l1
              0JhybZuTEVPRUCG6pTKAXQpLxmjJ5oG9M91RP17nsuQeE7Ng
              0Ap4BBn5hocojkfthwgbX4lqBMecpBAnky5jn6slmzS_rL-L
              w-_8hUldaTPD9MHlHPrvcsRV5uw8wK5MB6QyfS6wF4b0Kj2T
              vYceNlE",
      "e":   "AQAB",
      "x5t": "Ae0sLVtm78VT-mQXJQop-ENOM6o"
    },
    {
      "kty": "RSA",
      "kid": "dbc28570-1193-11e3-b2b1-835742119fe8",
      "n":   "AM-ktWkQ8btj_HEdAA6kOpzJGgoHNZsJmxjh_PifpgAUfQeq
              MO_YBR100IdJZRzJfULyhRwn9bikCq87WToxgPWOnd3sH3qT
              YiAcIR5S6tBbsyp6WYmwM1yuC0vLCo6SoDzdK1SvkQKM3QWk
              0GFNU4l4qXYAMxaSw83i6yv5DBVbST7E92vS6Gq_4pgI26l1
              0JhybZuTEVPRUCG6pTKAXQpLxmjJ5oG9M91RP17nsuQeE7Ng
              0Ap4BBn5hocojkfthwgbX4lqBMecpBAnky5jn6slmzS_rL-L
              w-_8hUldaTPD9MHlHPrvcsRV5uw8wK5MB6QyfS6wF4b0Kj2T
              vYceNlE",
      "e":   "AQAB",
      "x5t": "lYZC2n9TBpOaUsBclEIacQTKToA"
    }
  ]
}
```

9. IANA Considerations

This document registers a well-known URI [RFC5785] for protocols that use POSH. The completed template follows.

URI suffix: posh.

Change controller: IETF

Specification document: [[this document]]

Related information: Because the "posh." string is merely a prefix, protocols that use POSH need to register particular URIs that are prefixed with the "posh." string.

Note that the registered URI is "posh." (with a trailing dot). This is merely a prefix to be placed at the front of well-known URIs [RFC5785] registered by protocols that use POSH, which themselves are responsible for the relevant registrations with the IANA. The URIs registered by such protocols SHOULD match the URI template [RFC6570] path "/.well-known/posh.{servicedesc}.json"; that is, begin with "posh." and end with ".json" (indicating a media type of application/json [RFC4627] or application/jwk-set+json [JOSE-JWK]).

For POSH-using protocols that rely on DNS SRV records [RFC2782], the "{servicedesc}" part of the well-known URI SHOULD be "{service}.{proto}", where the "{service}" is the DNS SRV "Service" prepended by the underscore character "_" and the "{proto}" is the DNS SRV "Proto" also prepended by the underscore character "_". As an example, the well-known URI for XMPP server-to-server connections would be "posh._xmpp-server._tcp.json" since XMPP [RFC6120] registers a service name of "xmpp-server" and uses TCP as the underlying transport protocol.

For other POSH-using protocols, the "{servicedesc}" part of the well-known URI can be any unique string or identifier for the protocol, which might be a service name registered with the IANA in accordance with [RFC6335] or which might be an unregistered name. As an example, the well-known URI for the mythical "Foo" service could be "posh.foo.json".

Note: As explained in [RFC5785], the IANA registration policy [RFC5226] for well-known URIs is Specification 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]. Additionally, the security of POSH can benefit from other HTTP hardening protocols, such as HSTS [RFC6797] and key pinning [KEYPIN], especially if the TLS client shares some information with a common HTTPS implementation (e.g., platform-default web browser).

Note well that POSH is used by a TLS client to obtain the public key of a TLS 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 TLS 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 URL. The location provided in the redirect response MUST specify an HTTPS URL. Source domains SHOULD use only temporary redirect mechanisms, such as HTTP status codes 302 (Found) and 307 (Temporary Redirect). Clients MAY treat any redirect as temporary, ignoring the specific semantics for 301 (Moved Permanently) and 308 (Permanent Redirect) [HTTP-STATUS-308]. To protect against circular references, clients MUST NOT follow an infinite number of redirects. It is RECOMMENDED that clients follow no more than 10 redirects, although applications or implementations can require that fewer redirects be followed.

11. References

11.1. Normative References

[JOSE-JWK] Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key-20 (work in progress), January 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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, May 2008.

[RFC5785] Nottingham, M. and E. Hammer-Lahav, "Defining Well-Known Uniform Resource Identifiers (URIs)", RFC 5785, April 2010.

[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, March 2011.

11.2. Informative References

[HTTP-STATUS-308]
Reschke, J., "The Hypertext Transfer Protocol (HTTP) Status Code 308 (Permanent Redirect)", draft-reschke-http-status-308-07 (work in progress), March 2012.

[KEYPIN] Evans, C., Palmer, C., and R. Sleevi, "Public Key Pinning Extension for HTTP", draft-ietf-websec-key-pinning-09 (work in progress), November 2013.

[XMPP-DNA]
Saint-Andre, P. and M. Miller, "Domain Name Associations (DNA) in the Extensible Messaging and Presence Protocol (XMPP)", draft-ietf-xmpp-dna-05 (work in progress), February 2014.

[RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, May 2005.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[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, August 2011.

[RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, April 2012.

[RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, March 2012.

[RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, August 2012.

[RFC6797] Hodges, J., Jackson, C., and A. Barth, "HTTPS Strict Transport Security (HSTS)", RFC 6797, November 2012.

[RFC7030] Pritikin, M., Yee, P., and D. Harkins, "Enrollment over Secure Transport", RFC 7030, October 2013.

Appendix A. Acknowledgements

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

Authors' Addresses

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

Email: mamille2@cisco.com

Peter Saint-Andre
&yet

Email: ietf@stpeter.im