Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2016

M. Thomson
Mozilla

March 20, 2016

Message Encryption for Web Push
draft-ietf-webpush-encryption-02

Abstract

A message encryption scheme is described for the Web Push protocol. This scheme provides confidentiality and integrity for messages sent from an Application Server to a User Agent.

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/.

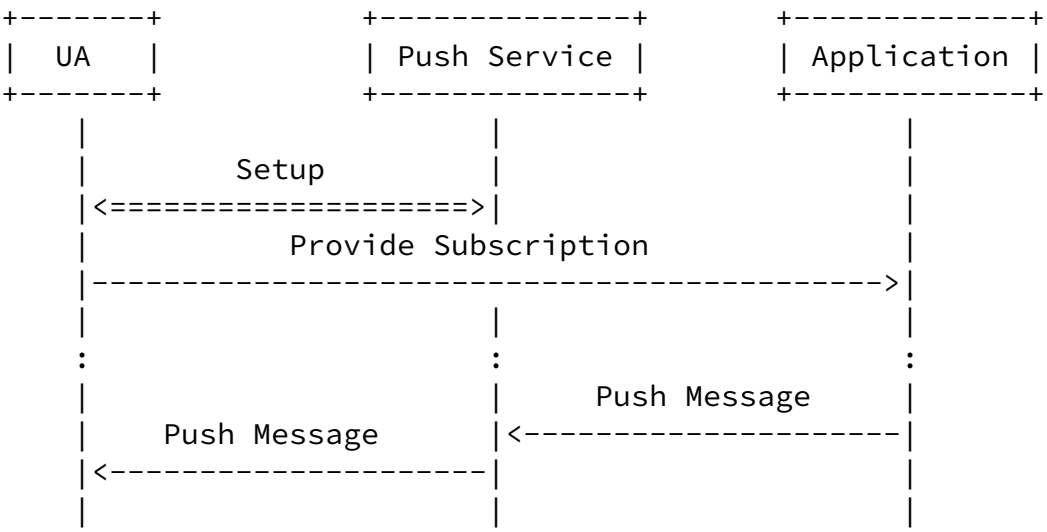
Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on September 21, 2016.

Table of Contents

1. Introduction

The Web Push protocol [I-D.ietf-webpush-protocol] is an intermediated protocol by necessity. Messages from an Application Server are delivered to a User Agent via a Push Service.



This document describes how messages sent using this protocol can be secured against inspection, modification and falsification by a Push

Service.

Web Push messages are the payload of an HTTP message [RFC7230]. These messages are encrypted using an encrypted content encoding [I-D.ietf-httpbis-encryption-encoding]. This document describes how

this content encoding is applied and describes a recommended key management scheme.

For efficiency reasons, multiple users of Web Push often share a central agent that aggregates push functionality. This agent can enforce the use of this encryption scheme by applications that use push messaging. An agent that only delivers messages that are properly encrypted strongly encourages the end-to-end protection of messages.

A web browser that implements the Web Push API [API] can enforce the use of encryption by forwarding only those messages that were properly encrypted.

1.1. Notational Conventions

The words "MUST", "MUST NOT", "SHOULD", and "MAY" are used in this document. It's not shouting, when they are capitalized, they have the special meaning described in [RFC2119].

2. Key Generation and Agreement

For each new subscription that the User Agent generates for an application, it also generates an asymmetric key pair for use in Diffie-Hellman (DH) [DH] or elliptic-curve Diffie-Hellman (ECDH) [ECDH]. The public key for this key pair can then be distributed by the application to the Application Server along with the URI of the subscription. The private key MUST remain secret.

This key pair is used with the Diffie-Hellman key exchange as described in Section 4.2 of [I-D.ietf-httpbis-encryption-encoding].

A User Agent MUST generate and provide a public key for the scheme described in Section 5.

The public key MUST be accompanied by a key identifier that can be

used in the "keyid" parameter to identify which key is in use. Key identifiers need only be unique within the context of a subscription.

2.1. Diffie-Hellman Group Information

As described in [I-D.ietf-httpbis-encryption-encoding], use of Diffie-Hellman for key agreement requires that the receiver provide clear information about its chosen group and the format for the "dh" parameter with each potential sender.

This document only describes a single ECDH group and point format, described in Section 5. A specification that defines alternative

groups or formats MUST provide a means of indicating precisely which group and format is in use for every public key that is provided.

2.2. Key Distribution

The application using the subscription distributes the key identifier and public key along with other subscription information, such as the subscription URI and expiration time.

The communication medium by which an application distributes the key identifier and public key MUST be confidentiality protected for the reasons described in [I-D.ietf-webpush-protocol]. Most applications that use push messaging have a pre-existing relationship with an Application Server. Any existing communication mechanism that is authenticated and provides confidentiality and integrity, such as HTTPS [RFC2818], is sufficient.

2.3. Push Message Authentication

To ensure that push messages are correctly authenticated, a symmetric authentication secret is added to the information generated by a User Agent. The authentication secret is mixed into the key derivation process described in [I-D.ietf-httpbis-encryption-encoding].

The authentication secret ensures that exposure or leakage of the DH public key - which, as a public key, is not necessarily treated as a secret - does not enable an adversary to generate valid push messages.

A User Agent MUST generate and provide a hard to guess sequence of octets that is used for authentication of push messages. This SHOULD be generated by a cryptographically strong random number generator [RFC4086] and be at least 16 octets long.

3. Message Encryption

An Application Server that has the public key, group and format information plus the authentication secret can encrypt a message for the User Agent.

3.1. Key Derivation

The Application Server generates a new DH or ECDH key pair in the same group as the value generated by the User Agent.

From the newly generated key pair, the Application Server performs a DH or ECDH computation with the public key provided by the User Agent to find the input keying material for key derivation. The

Application Server then generates 16 octets of salt that is unique to the message. A random [RFC4086] salt is acceptable.

Web push uses the authentication secret defined in Section 4.3 of [I-D.ietf-httpbis-encryption-encoding]. This authentication secret (see Section 2.3) is generated by the user agent and shared with the application server.

3.2. Push Message Content Encryption

The Application Server then encrypts the payload. Header fields are populated with base64url encoded [RFC7515] values:

- the salt is added to the "salt" parameter of the Encryption header field; and
- the public key for its DH or ECDH key pair is placed in the "dh" parameter of the Crypto-Key header field.

An application server MUST encrypt a push message with a single record. This allows for a minimal receiver implementation that handles a single record. If the message is 4096 octets or longer,

the "rs" parameter MUST be set to a value that is longer than the encrypted push message length.

Note that a push service is not required to support more than 4096 octets of payload body, which equates to 4080 octets of cleartext, so the "rs" parameter can be omitted for messages that fit within this limit.

An application server MUST NOT use other content encodings for push messages. In particular, content encodings that compress could result in leaking of push message contents. The Content-Encoding header field therefore has exactly one value, which is "aesgcm128". Multiple "aesgcm128" values are not permitted.

An application server MUST include exactly one entry in each of the Encryption and Crypto-Key header fields. This allows the "keyid" parameter to be omitted from both header fields.

An application server MUST NOT include an "aesgcm128" parameter in the Encryption header field.

4. Message Decryption

A User Agent decrypts messages are decrypted as described in [I-D.ietf-httpbis-encryption-encoding]. The authentication secret described in Section 3.1 is used in key derivation.

Note that the value of the "keyid" parameter is used to identify the correct share, if there are multiple values for the Crypto-Key header field.

A receiver is not required to support multiple records. Such a receiver MUST check that the record size is large enough to contain the entire payload body in a single record. The "rs" parameter MUST NOT be exactly equal to the length of the payload body minus the length of the authentication tag (16 octets); that length indicates that the message has been truncated.

5. Mandatory Group and Public Key Format

User Agents MUST expose an elliptic curve Diffie-Hellman share on the P-256 curve [FIPS186].

Public keys, such as are encoded into the "dh" parameter, MUST be in the form of an uncompressed point as described in [X.692] (that is, a 65 octet sequence that starts with a 0x04 octet).

The label for this curve is the string "P-256" encoded in ASCII (that is, the octet sequence 0x50, 0x2d, 0x32, 0x35, 0x36).

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

The security considerations of [I-D.ietf-httpbis-encryption-encoding] describe the limitations of the content encoding. In particular, any HTTP header fields are not protected by the content encoding scheme. A User Agent MUST consider HTTP header fields to have come from the Push Service. An application on the User Agent that uses information from header fields to alter their processing of a push message is exposed to a risk of attack by the Push Service.

The timing and length of communication cannot be hidden from the Push Service. While an outside observer might see individual messages intermixed with each other, the Push Service will see what Application Server is talking to which User Agent, and the subscription that is used. Additionally, the length of messages could be revealed unless the padding provided by the content encoding scheme is used to obscure length.

8. References

8.1. Normative References

[DH] Diffie, W. and M. Hellman, "New Directions in Cryptography", IEEE Transactions on Information Theory, V.IT-22 n.6 , June 1977.

[ECDH] SECG, "Elliptic Curve Cryptography", SEC 1 , 2000, <http://www.secg.org/>.

[FIPS186] National Institute of Standards and Technology (NIST), "Digital Signature Standard (DSS)", NIST PUB 186-4 , July 2013.

[I-D.ietf-httpbis-encryption-encoding] Thomson, M., "Encrypted Content-Encoding for HTTP", draft-ietf-httpbis-encryption-encoding-01 (work in progress), March 2016.

[I-D.ietf-webpush-protocol] Thomson, M., Damaggio, E., and B. Raymor, "Generic Event Delivery Using HTTP Push", draft-ietf-webpush-protocol-03 (work in progress), February 2016.

[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>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[X.692] ANSI, "Public Key Cryptography For The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62 , 1998.

8.2. Informative References

[API] van Ouwerkerk, M. and M. Thomson, "Web Push API", 2015, <https://w3c.github.io/push-api/>.

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

[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>.

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

Author's Address

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com