Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 3, 2016

M. Thomson
Mozilla

July 2, 2015

Message Encryption for Web Push
draft-thomson-webpush-encryption-01

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.

Table of Contents

1. Introduction

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

```
 +-------+           +--------------+       +-------------+
 |  UA   |           | Push Service |       | Application |
 +-------+           +--------------+       +-------------+
     |                      |                      |
     |        Setup         |                      |
     |<====================>|                      |
     |           Provide Subscription              |
     |-------------------------------------------->|
     |                      |                      |
     :                      :                      :
     |                      |     Push Message     |
     |    Push Message      |<---------------------|
     |<---------------------|                      |
     |                      |                      |
```

This document describes how messages sent using this protocol can be secured against inspection or modification 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.thomson-http-encryption]. 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.thomson-http-encryption].

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.thomson-http-encryption], use of Diffie-Hellman for key agreement requires that the receiver provide clear information about it's 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.thomson-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.

3. Message Encryption

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

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 shared secret. The Application Server then generates 16 octets of salt that is unique to the message. A random [RFC4086] salt is acceptable. These values are used to calculate the content encryption key as defined in Section 3.2 of [I-D.thomson-http-encryption].

The Application Server then encrypts the payload. Header fields are populated with URL-safe base-64 encoded [RFC4648] values:

- the "keyid" from the User Agent is added to both the Encryption-Key and Encryption header fields;
- 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 Encryption-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 long, or longer, this means that 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.

4. Message Decryption

A User Agent decrypts messages are decrypted as described in [I-D.thomson-http-encryption]. The value of the "keyid" parameter is used to identify the correct key pair, if there is more than one possible value for the corresponding subscription.

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 that enforce encryption 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].

6. IANA Considerations

This document has no IANA actions.

7. Security Considerations

The security considerations of [I-D.thomson-http-encryption] 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 they are talking about. 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.thomson-http-encryption] Thomson, M., "Encrypted Content-Encoding for HTTP", draft-thomson-http-encryption-01 (work in progress), July 2015.

[I-D.thomson-webpush-protocol] Thomson, M., Damaggio, E., and B. Raymor, "Generic Event Delivery Using HTTP Push", draft-thomson-webpush-protocol-00 (work in progress), April 2015.

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

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

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

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

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[RFC7230] Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, June 2014.

Author's Address

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com