Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: October 9, 2016

M. Thomson
Mozilla
P. Beverloo
Google

April 7, 2016

Voluntary Application Server Identification for Web Push
draft-ietf-webpush-vapid-00

Abstract

An application server can voluntarily identify itself to a push service using the described technique. This identification information can be used by the push service to attribute requests that are made by the same application server to a single entity. This can used to reduce the secrecy for push subscription URLs by being able to restrict subscriptions to a specific application server. An application server is further able include additional information the operator of a push service can use to contact the operator of the application server.

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 October 9, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
1.1. Voluntary Identification 3
1.2. Notational Conventions 3
2. Application Server Self-Identification 4
2.1. Application Server Contact Information 4
2.2. Example 4
3. WebPush Authentication Scheme 5
4. Public Key Representation 6
5. Subscription Restriction 6
5.1. Creating a Restricted Push Subscription 7
5.2. Using Restricted Subscriptions 7
6. Security Considerations 8
7. IANA Considerations 9
7.1. WebPush Authentication Scheme 9
7.2. p256ecdsa Parameter for Crypto-Key Header Field 9
8. Acknowledgements 9
9. References 9
9.1. Normative References 9
9.2. Informative References 10
Authors' Addresses 11

1. Introduction

The Web Push protocol [I-D.ietf-webpush-protocol] describes how an application server is able to request that a push service deliver a push message to a user agent.

As a consequence of the expected deployment architecture, there is no basis for an application server to be known to a push service prior to requesting delivery of a push message. By the same measure, requesting the creation of a subscription for push message receipts has no prior authentication. Requiring that the push service be able to authenticate application servers places an unwanted constraint on the interactions between user agents and application servers, who are the ultimate users of a push service. That constraint would also

degrade the privacy-preserving properties the protocol provides. For these reasons, [I-D.ietf-webpush-protocol] does not define a mandatory system for authentication of application servers.

An unfortunate consequence of this design is that a push service is exposed to a greater risk of denial of service attack. While requests from application servers can be indirectly attributed to user agents, this is not always efficient or even sufficient. Providing more information about the application server directly to a push service allows the push service to better distinguish between legitimate and bogus requests.

Additionally, this design also relies on endpoint secrecy as any application server in possession of the endpoint is able to send messages, albeit without payloads. In situations where usage of a subscription can be limited to a single application server, the ability to associate a subscription with the application server could reduce the impact of a data leak.

1.1. Voluntary Identification

This document describes a system whereby an application server can volunteer information about itself to a push service. At a minimum, this provides a stable identity for the application server, though this could also include contact information, such as an email address.

A consistent identity can be used by a push service to establish behavioral expectations for an application server. Significant deviations from an established norm can then be used to trigger exception handling procedures.

Voluntarily-provided contact information can be used to contact an application server operator in the case of exceptional situations.

Experience with push service deployment has shown that software errors or unusual circumstances can cause large increases in push message volume. Contacting the operator of the application server has proven to be valuable.

Even in the absence of usable contact information, an application server that has a well-established reputation might be given preference over an unidentified application server when choosing whether to discard a push message.

1.2. 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].

The terms "push message", "push service", "push subscription", "application server", and "user agent" are used as defined in [I-D.ietf-webpush-protocol].

2. Application Server Self-Identification

Application servers SHOULD generate and maintain a signing key pair usable with elliptic curve digital signature (ECDSA) over the P-256 curve [FIPS186]. Use of this key when sending push messages establishes a continuous identity for the application server.

When requesting delivery of a push message, the application includes a JSON Web Token (JWT) [RFC7519], signed using its signing key. The token includes a number of claims as follows:

- An "aud" (Audience) claim in the token MUST include the unicode serialization of the origin (Section 6.1 of [RFC6454]) of the push resource URL. This binds the token to a specific push service. This ensures that the token is reusable for all push resource URLs that share the same origin.

- An "exp" (Expiry) claim MUST be included with the time after which the token expires. This limits the time that a token over which a token is valid. An "exp" claim MUST NOT be more than 24 hours from the time of the request.

This JWT is included in an Authorization header field, using an auth-scheme of "WebPush". A push service MAY reject a request with a 403 (Forbidden) status code [RFC7235] if the JWT signature or its claims

are invalid.

The JWT MUST use a JSON Web Signature (JWS) [RFC7515]. The signature MUST use ECDSA on the NIST P-256 curve [FIPS186], that is "ES256" [RFC7518].

2.1. Application Server Contact Information

If the application server wishes to provide contact details it MAY include an "sub" (Subject) claim in the JWT. The "sub" claim SHOULD include a contact URI for the application server as either a "mailto:" (email) [RFC6068] or an "https:" [RFC2818] URI.

2.2. Example

An application server requests the delivery of a push message as described in [I-D.ietf-webpush-protocol]. If the application server wishes to self-identify, it includes an Authorization header field with credentials that use the "WebPush" authentication scheme

Section 3 and a Crypto-Key header field that includes its public key Section 4.

```
POST /p/JzLQ3raZJfFBR0aqvOMsLrt54w4rJUsV HTTP/1.1
Host: push.example.net
Push-Receipt: https://push.example.net/r/3ZtI4YVNBnUUZhuoChl6omU
Content-Type: text/plain;charset=utf8
Content-Length: 36
Authorization: Bearer
    eyJ0eXAiOiJKV1QiLCJhbGciOiJFUzI1NiJ9.eyJhdWQiOiJodHRwczovL3B
    1c2guZXhhbXBsZS5uZXQiLCJleHAiOjE0NTM1MjM3NjgsInN1YiI6Im1haWx
    0bzpwdXNoQGV4YW1wbGUuY29tIn0.i3CYb7t4xfxCDquptFOepC9GAu_HLGk
    MlMuCGSK2rpiUfnK9ojFwDXb1JrErtmysazNjjvW2L9OkSSHzvoD1oA
Crypto-Key: p256ecdsa=BA1Hxzyi1RUM1b5wjxsn7nGxAszw2u61m164i3MrAIxH
                      F6YK5h4SDYic-dRuU_RCPCfA5aq9ojSwk5Y2EmClBPs

iChYuI3jMzt3ir20P8r_jgRR-dSuN182x7iB
```

Figure 1: Requesting Push Message Delivery with JWT

Note that the header fields shown in Figure 1 don't include line wrapping. Extra whitespace is added to meet formatting constraints.

This equates to a JWT with the header and body shown in Figure 2. This JWT would be valid until 2016-01-21T01:53:25Z [RFC3339].

```
header = {"typ":"JWT","alg":"ES256"}
body = { "aud":"https://push.example.net",
         "exp":1453341205,
         "sub":"mailto:push@example.com" }
```

Figure 2: Example JWT Header and Body

Issue: The first part of the JWT is effectively fixed. Would be it acceptable to require that that segment is omitted from the header field?

3. WebPush Authentication Scheme

A new "WebPush" HTTP authentication scheme [RFC7235] is defined. This authentication scheme carries a signed JWT, as described in Section 2.

This authentication scheme is for origin-server authentication only. Therefore, this authentication scheme MUST NOT be used with The Proxy-Authenticate or Proxy-Authorization header fields.

This authentication scheme does not require a challenge. Clients are able to generate the Authorization header field without any additional information from a server. Therefore, a challenge for this authentication scheme MUST NOT be sent in a WWW-Authenticate header field.

All unknown or unsupported parameters to "WebPush" authentication credentials MUST be ignored. The "realm" parameter is ignored for this authentication scheme.

4. Public Key Representation

In order for the push service to be able to validate the JWT, it needs to learn the public key of the application server. A "p256ecdsa" parameter is defined for the Crypto-Key header field

[I-D.ietf-httpbis-encryption-encoding] to carry this information.

The "p256ecdsa" parameter includes an elliptic curve digital signature algorithm (ECDSA) public key [FIPS186] in uncompressed form [X9.62] that is encoded using the URL- and filename-safe variant of base-64 [RFC4648] with padding removed.

Note that with push message encryption [I-D.ietf-webpush-encryption], this results in two values in the Crypto-Key header field, one with the a "p256dh" key and another with a "p256ecdsa" key.

Some implementations permit the same P-256 key to be used for signing and key exchange. An application server MUST select a different private key for the key exchange (i.e., "p256dh") and signing (i.e., "p256ecdsa"). Though a push service is not obligated to check either parameter, it SHOULD reject push messages that have identical values for these parameters with a 400 (Bad Request) status code.

Editor's Note: JWK [RFC7517] seems like the obvious choice here. However, JWK doesn't define a compact representation for public keys, which complicates the representation of JWK in a header field.

5. Subscription Restriction

The public key of the application server serves as a stable identifier for the server. This key can be used to restrict a push subscription to a specific application server.

Subscription restriction reduces the reliance on endpoint secrecy by requiring proof of possession to be demonstrated by an application server when requesting delivery of a push message. This provides an

additional level of protection against leaking of the details of the push subscription.

5.1. Creating a Restricted Push Subscription

The user agent includes the public key of the application server when requesting the creation of a push subscription. This restricts use of the resulting subscription to application servers that are able to

provide proof of possession for the corresponding private key.

This public key is then added to the request to create a push subscription as described in Section 4. The Crypto-Key header field includes exactly one public key. For example:

```
POST /subscribe/ HTTP/1.1
Host: push.example.net
Crypto-Key: p256ecdsa=BBa22H8qaZ-iDMH9izb4qE72puwyvfjH2RxoQr5oiS4b
                      KImoRwJm5xK9hLrbfIik20g31z8MpLFMCMr8y2cu6gY
```

Figure 3: Example Subscribe Request

An application might use the Web Push API [API] to include this information. For example, the API might permit an application to provide a public key as part of a new field on the "PushSubscriptionOptions" dictionary.

Editor's Note: Allowing the inclusion of multiple keys when creating a subscription would allow a subscription to be associated with multiple application servers or application server instances. This might be more flexible, but it also would require more state to be maintained by the push service for each subscription.

5.2. Using Restricted Subscriptions

When a push subscription has been associated with an application server, the request for push message delivery MUST include proof of possession for the associated private key or token that was used when creating the push subscription.

A push service MUST reject a message that omits mandatory credentials with a 401 (Unauthorized) status code. A push service MAY reject a message that includes invalid credentials with a 403 (Forbidden) status code. Credentials are invalid if:

- either the authentication credentials or public key are not included in the request,

- the signature on the JWT cannot be successfully verified using the

included public key,

- the current time is later than the time identified in the "exp" (Expiry) claim or more than 24 hours before the expiry time,

- the origin of the push resource is not included in the "aud" (Audience) claim, or

- the public key used to sign the JWT doesn't match the one that was included in the creation of the push message.

A push subscription that is not restricted to a particular key MAY still validate a token that is present, except for the last check. A push service MAY then reject a request if the token is found to be invalid.

Editor's Note: In theory, since the push service was given a public key, the push message request could omit the public key. On balance, requiring the key keeps things simple and it allows push services to compress the public key (by hashing it, for example). In any case, the relatively minor space savings aren't particularly important on the connection between the application server and push service.

A push service does not need to forward the JWT or public key to the user agent when delivering the push message.

6. Security Considerations

This authentication scheme is vulnerable to replay attacks if an attacker can acquire a valid JWT. Applying narrow limits to the period over which a replayable token can be reused limits the potential value of a stolen token to an attacker and can increase the difficulty of stealing a token.

An application server might offer falsified contact information. A push service operator therefore cannot use the presence of unvalidated contact information as input to any security-critical decision-making process.

Validation of a signature on the JWT requires a non-trivial amount of computation. For something that might be used to identify legitimate requests under denial of service attack conditions, this is not ideal. Application servers are therefore encouraged to reuse a JWT, which permits the push service to cache the results of signature validation.

7. IANA Considerations

7.1. WebPush Authentication Scheme

This registers the "WebPush" authentication scheme in the "Hypertext Transfer Protocol (HTTP) Authentication Scheme Registry" established in [RFC7235].

Authentication Scheme Name: WebPush

Pointer to specification text: Section 3 of this document

Notes: This scheme is origin-server only and does not define a challenge.

7.2. p256ecdsa Parameter for Crypto-Key Header Field

This registers a "p256ecdsa" parameter for the Crypto-Key header field in the "Hypertext Transfer Protocol (HTTP) Crypto-Key Parameters" established in [I-D.ietf-httpbis-encryption-encoding].

Parameter Name: p256ecdsa

Purpose: Conveys a public key for that is used to generate an ECDSA signature.

Reference: Section 4 of this document

8. Acknowledgements

This document would have been much worse than it currently is if not for the contributions of Benjamin Bangert, Chris Karlof, Costin Manolache, and others.

9. References

9.1. Normative References

[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-04 (work in progress), March 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>.

[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>.

[RFC6068] Duerst, M., Masinter, L., and J. Zawinski, "The 'mailto' URI Scheme", RFC 6068, DOI 10.17487/RFC6068, October 2010, <http://www.rfc-editor.org/info/rfc6068>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <http://www.rfc-editor.org/info/rfc6454>.

[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>.

[RFC7518] Jones, M., "JSON Web Algorithms (JWA)", RFC 7518, DOI 10.17487/RFC7518, May 2015, <http://www.rfc-editor.org/info/rfc7518>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <http://www.rfc-editor.org/info/rfc7519>.

[X9.62] ANSI, "Public Key Cryptography For The Financial Services

Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)", ANSI X9.62 , 1998.

9.2. Informative References

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

[I-D.ietf-webpush-encryption] Thomson, M., "Message Encryption for Web Push", draft-ietf-webpush-encryption-02 (work in progress), March 2016.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <http://www.rfc-editor.org/info/rfc7235>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

Authors' Addresses

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

Peter Beverloo
Google

Email: beverloo@google.com