Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 26, 2016

M. Thomson
Mozilla

November 23, 2015

Voluntary Application Server Identification for Web Push
draft-thomson-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. 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

Copyright Notice

Table of Contents

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 more directly to a push service allows the push service to better distinguish between legitimate and bogus requests.

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.

1.1. 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].

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

2. Self-Identification

A push service that supports application server self-identification requests a client certificate from application servers. The client certificate is requested during the TLS [RFC5246] handshake.

An application server that does not have a client certificate offers no certificate in response; an application server that wishes to self-identify includes a certificate.

The certificate that the application server offers SHOULD be self-signed (see Section 3.2 of [RFC5280]). The certificate MAY contain an alternative name extension (Section 4.2.1.6 of [RFC5280]) that includes contact information. Of the available options, an email address using the "rfc822Name" form or an HTTP [RFC7230] (or HTTPS [RFC2818]) "uniformResourceIdentifier" SHOULD be included, though including other options are not prohibited.

The offered end-entity certificate or the public key it contains becomes an identifier for the application server. Push services are able to reduce the data they retain for an application server, either by extracting important information like the subject public key information (SPKI), by hashing, or a combination. Of course, a push service might choose to ignore the provided information.

To avoid requesting certificates from user agents, it might be necessary to serve requests from user agents and requests from application servers on different hostnames or port numbers.

2.1. Alternatives

Several options have been proposed, here are some of the more concrete options. Some options might even be better than the certificate-based option.

2.1.1. Application Tokens

In this model, the push service vends a token to each application server that requests it. That token is kept secret and used as the basis for authentication.

This doesn't address the need for contact information, but it addresses the need for a consistent application server identifier.

A Cookie [RFC6265] is a token of this nature. All the considerations regarding the use (and misuse) of HTTP cookies apply should this option be chosen. A mechanism that makes token theft more difficult, such as [I-D.ietf-tokbind-https] might be needed. However that suggests a separate option (see Section 2.1.4).

This information would be repeated with each request, but that overhead is greatly reduced by header compression [RFC7541] in HTTP/2 [RFC7540].

2.1.2. Contact Information Header Field

Contact information for an application server could be included in a header field, either directly (e.g., a From header field, Section 5.5.1 of [RFC7231]), or by reference (e.g., a new "contact" link relation [RFC5988] that identified a vCard [RFC6350]). Note that a From header field is limited to email addresses.

Like an application token Section 2.1.1, contact information would need to be repeated, though that cost is reduced with HTTP/2.

2.1.3. Request Signing

Signing of push message requests would allow the push service to attribute requests to an application server based on an asymmetric key. This could be done in any number of ways JWS [RFC7515] and HTTP signatures [I-D.cavage-http-signatures] being the most likely options. Note that the latter does not provide a means of conveying the signing key, which would be necessary for this application.

Request signing has several limitations:

o Deciding what to sign is challenging. Signing only the body of a message is not sufficient to prevent message replay attacks.

o Every message contains a signature, which can increase the load on a server signficantly. This is especially bad if a signature validation result is input to denial of service mitigation decision making.

2.1.4. Token Binding

The mechanism proposed in [I-D.ietf-tokbind-https] can be used to provide a stable identifier for application servers. This includes a signature over material that is exported from the underlying TLS connection. Importantly, this does not require a new signature for each request: the same signature is repeated for every request, HTTP/2 is again used to reduce the cost of the repeated information.

Token binding could be used independently of cookies. Consequently, an application server would not be required to accept and store cookies, though the push service would not be able to offload any state as a result.

3. IANA Considerations

This document has no IANA actions (yet).

4. Security Considerations

TLS 1.2 [RFC5246] does not provide any confidentiality protections for client certificates. A network attacker can therefore see the identification information that is provided by the application server. A push service MAY choose to offer confidentiality protection for application server identity by initiating TLS renegotiation immediately after establishing the TLS connection at the cost of some additional latency. Using TLS 1.3 [I-D.ietf-tls-tls13] provides confidentiality protection for this information without additional latency.

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.

5. References

5.1. Normative References

[I-D.ietf-webpush-protocol]
Thomson, M., Damaggio, E., and B. Raymor, "Generic Event Delivery Using HTTP Push", draft-ietf-webpush-protocol-01 (work in progress), October 2015.

[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>.

[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>.

[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>.

5.2. Informative References

[I-D.cavage-http-signatures]
Cavage, M. and M. Sporny, "Signing HTTP Messages", draft-cavage-http-signatures-05 (work in progress), October 2015.

[I-D.ietf-tls-tls13]
Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-10 (work in progress), October 2015.

[I-D.ietf-tokbind-https]
Popov, A., Nystrom, M., Balfanz, D., and A. Langley, "Token Binding over HTTP", draft-ietf-tokbind-https-02 (work in progress), October 2015.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC6265] Barth, A., "HTTP State Management Mechanism", RFC 6265, DOI 10.17487/RFC6265, April 2011, <http://www.rfc-editor.org/info/rfc6265>.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, DOI 10.17487/RFC6350, August 2011, <http://www.rfc-editor.org/info/rfc6350>.

[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>.

[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>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

[RFC7541] Peon, R. and H. Ruellan, "HPACK: Header Compression for HTTP/2", RFC 7541, DOI 10.17487/RFC7541, May 2015, <http://www.rfc-editor.org/info/rfc7541>.

Author's Address

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com