Abstract

A mechanism is described whereby a server can use client-selected shared cache.

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 May 3, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
1. Shared Caching for HTTPS

Shared caches allow an HTTP server to offload the responsibility for delivering certain content. Content in the shared cache can be accessed efficiently by multiple clients, saving the origin server from having to serve those requests and ensuring that clients receive responses to cached requests more quickly.

Proxy caching is the most common configuration for shared caching. A proxy cache is either explicitly configured by a client, discovered as a result of being automatically configured.

HTTPS [RFC2818] prevents the use of proxies by creating an authenticated end-to-end connection to the origin server or its gateway that is authenticated. This provides a critical protection against man-in-the-middle attacks, but it also prevents the proxy from acting as a shared cache.

Clients do not direct queries for "https" URIs to proxies. Clients configured with a proxy use the CONNECT pseudo-method (Section 4.3.6 of [RFC7231]) with any explicitly configured or discovered proxies to create an end-to-end tunnel. Transparent proxies are unable to intercept connections that are protected with TLS.

This document describes a method that enables shared caching for a limited set of "https" resources, as selected by the server. The server conditionally delegates the hosting of secure content to itself. This delegation includes a marker that signals permission for a client to send a request for an "https" resource via a proxy rather than insisting on an end-to-end TLS connection.
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 defined in [RFC2119].

This document uses the term "proxy cache" to refer to a proxy [RFC7230] that operates an HTTP cache [RFC7234].

2. Same-Host Secure Content Delegation

The secure content delegation mechanism defined in [SCD] is used to create a separate resource that contains encrypted and integrity protected content. To enable caching, the primary and secondary servers can be the same server.

A client that signals a willingness to support delegation is provided with a response that uses a proxy-enabled out-of-band encoding that behaves identically to the out-of-band encoding defined in [I-D.reschke-http-oob-encoding]. The out-of-band encoding identifies a secondary resource and implicitly indicates that the client is willing to use a proxy and that the server allows this use. The client is then able to request the secondary resource from a proxy cache rather than directly to the origin server.

In this document, the origin server is able to act in the role of the secondary server in [SCD]. However, all of the considerations that apply to having a secondary server host content apply instead to the proxy cache. Thus, integrity and confidentiality protections against the proxy cache are the primary consideration.

2.1. Signaling Presence of a Proxy

Without a clear signal from the client that a caching proxy is present, an origin server is unable to send a response with out-of-band encoding. A value of "out-of-band" in the Accept-Encoding header field only indicates willingness to use the secure content delegation mechanism.

A new "oobp" content encoding is defined. The "oobp" content encoding is identical to the "out-of-band" content encoding, with the following additional conditions:

- A client MUST NOT signal support for "oobp" content encoding unless it is using a proxy cache and it is willing to direct requests to that proxy.
o A server MUST NOT encode a response using the "oobp" content encoding unless it permits the request to be made to a proxy cache.

o The "oobp" content encoding MUST NOT be used to encode the contents of a request. The "out-of-band" content encoding is sufficient for that purpose.

Using a different content encoding name means that a resource using secure content delegation to a secondary server [SCD] does not inadvertently trigger a request via a proxy.

The security properties of delegation via a secondary server and via a caching proxy are similar only to the extent that a third party is involved. However, it might be the case that a secondary server has a stronger relationship with the primary server and additional constraints on its actions, such as contractual limitations. Such constraints might make it feasible to delegate to a secondary server selected by the primary server. A caching proxy might not be considered acceptable in the same way.

The "oobp" content encoding clearly indicates that the client is permitted to retrieve content from a proxy-cache.

Servers that use the "oobp" content encoding MUST include header fields for message integrity and encryption, such as the M-I header field [I-D.thomson-http-mice] or the Crypto-Key header field [I-D.ietf-httpbis-encryption-encoding]. Clients MUST NOT send a request via a proxy if these headers are not present. Absence of these header fields indicate an error on the part of the origin server, since integrity and confidentiality protection are mandatory.

2.2. Proxy Identification and Authentication

This mechanism does not work with a transparent caching proxy. Since the request is made over end-to-end HTTPS in the absence of a proxy, the feature will not be used unless the proxy is known to the client.

A proxy cache MUST therefore be expressly configured or discovered. This produces a name and possibly a port number for the proxy. The proxy MUST be contacted using HTTPS [RFC2818] and authenticated using the configured or discovered domain name.

Issue: What signal do we need from the proxy cache that it supports receiving requests with an "https://" scheme? Can we expect that a proxy cache will happily accept a request for an HTTPS URL? What if they ignore the scheme and send the request in the clear?
3. Performance Optimizations

As noted in [SCD], the secondary request required by out-of-band content encoding imposes a performance penalty. This can be mitigated by priming clients with information about the location and disposition of resources prior to the client making a request. A resource map described in [SCD] might be provided to clients to eliminate the latency involved in making requests of the origin server for resources that might be cached.

3.1. Proxy Cache Priming

A client that makes a request of an origin server via an unprimed proxy cache will suffer additional latency as a consequence of the cache having to make a request to the origin server.

The following options are possible:

- Clients can speculatively make requests to a proxy cache based on information it learns from a resource map, or from hints like the "prefetch" link relation [HINTS]. To avoid a potential waste of resources as a result of receiving complete responses, speculative requests might be limited to HEAD requests; alternatively, HTTP/2 [RFC7540] flow control might be used to allow only limited information to be sent.

- The origin server might provide the proxy cache with "prefetch" link relations in responses to requests for secondary resources. These link relations might identify other resources that the proxy might retrieve speculatively. This does not improve the latency of the initial request, but could improve subsequent requests.

4. Security Considerations

All the considerations of [SCD] apply. In particular, content that is distributed with the assistance of a proxy cache MUST include integrity and confidentiality protection. That means that the M-I header field [I-D.thomson-http-mice] and the Crypto-Key header field [I-D.ietf-httpbis-encryption-encoding] or equivalent information MUST be present in responses that include an out-of-band content encoding.

Clients that receive a response without the information necessary to ensure integrity and confidentiality protection against a proxy cache MUST NOT make a request to a proxy to retrieve that response. Clients could treat such a response as failed. Clients MAY then make the request directly to the origin server, or - if request can be safely retried - retry a request without the out-of-band token in the Accept-Encoding header field.
5. IANA Considerations

This document has no IANA actions. It should.

6. References

6.1. Normative References

[I-D.ietf-httpbis-encryption-encoding]

[I-D.thomson-http-mice]


6.2. Informative References


Authors’ Addresses

Martin Thomson
Mozilla
Email: martin.thomson@gmail.com

Goeran AP Eriksson
Ericsson
Email: goran.ap.eriksson@ericsson.com

Christer Holmberg
Ericsson
Email: christer.holmberg@ericsson.com