Workgroup: ohai
Internet-Draft:
draft-schwartz-ohai-consistency-doublecheck-00
Published: 7 April 2022
Intended Status: Standards Track
Expires: 9 October 2022
Authors: B. M. Schwartz
Google LLC

Key Consistency for Oblivious HTTP by Double-Checking

Abstract

The assurances provided by Oblivious HTTP depend on the client's ability to verify that it is using the same Request Resource and KeyConfig as many other users. This specification defines a protocol to enable this verification.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-schwartz-ohai-consistency-doublecheck/.

Source for this draft and an issue tracker can be found at https://github.com/bemasc/access-services.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Oblivious HTTP [I-D.ietf-ohai-ohttp] presumes at least three parties to each exchange: the client, the proxy, and the target (formally, the Oblivious Request Resource). When used properly, Oblivious HTTP enables the client to send requests to the target in such a way that the target cannot tell whether two requests came from the same client and the proxy cannot see the contents of the requests.

Oblivious HTTP's threat model assumes that at least one of the proxy and the target is acting properly, i.e. complying with the protocol and keeping certain information confidential. If either proxy or target misbehaves, the only effect must be a denial of service.

In order for these security guarantees to hold, several preconditions must be met:

1. The client must be one of many users who might be using the proxy. Otherwise, use of the proxy reveals the user's identity to the target.

2. The client must hold an authentic KeyConfig for the target. Otherwise, they could be speaking to the proxy, impersonating the target.

3. All users of this proxy must be equally likely to use this URI and KeyConfig for this target, regardless of their prior activity. Otherwise, the encrypted request identifies the user to the target.

4. (optional) The target must not learn the IP addresses of the clients, collectively. Otherwise, the target might be able to deanonymize requests by correlating them with external information about the clients.

This specification defines behaviors for the client, proxy, and target that achieve preconditions 2-4. (This specification does not address precondition 1.)

This draft is an instantiation of the "Single Proxy Discovery" architecture for key consistency, defined in Section 4.2 of [I-D.wood-key-consistency].

2. Conventions and Definitions

The key words "**MUST**", "**MUST NOT**", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Overview

In the Key Consistency Double-Check procedure, the Client emits two requests: one to the Proxy, and one through the Proxy to the Target. The Proxy will forward the first request to the Target if the response is not in cache.

```
+--------+       +-------+      +--------+
|        |<=====>|       |<---->|        |
| Client |       | Proxy |      | Target |
|        |<====================>|        |
+--------+       +-------+      +--------+
```

Figure 1: Overview of Key-Consistency Double-Check

The proxy caches the response, ensuring that all clients share it during its freshness lifetime. The client checks this against the authenticated response from the Target, preventing forgeries.

4. Requirements

4.1. Oblivious Request Resource

The Oblivious Request Resource **MUST** publish an Access Description [I-D.schwartz-masque-access-descriptions] over HTTP/3 containing the ohttp.request key, e.g.:

```
{
  "ohttp": {
    "request": {
      "uri": "https://example.com/ohttp/",
      "key": "(KeyConfig in Base64)"
    }
  }
}
```

The Oblivious Request Resource **MUST** include a "strong validator" ETag (Section 2 of [RFC7232]) in any response to a GET request for this access description, and **MUST** support the "If-Match" HTTP request header (Section 3 of [RFC7232]). The response **MUST** indicate "Cache-Control: public, no-transform, s-maxage=(...), immutable" [I-D.ietf-httpbis-cache][RFC8246]. For efficiency reasons, the max age **SHOULD** be at least 60 seconds, and preferably much longer.

If this Access Description changes, and the resource receives a request whose "If-Match" header identifies a previously served version that has not yet expired, it **MUST** return a success response containing the previous version. This response **MAY** indicate "Cache-Control: private".

4.2. Oblivious Proxy

The Oblivious Proxy **MUST** publish an Access Description that includes the ohttp.proxy and udp keys, indicating support for CONNECT-UDP [I-D.ietf-masque-connect-udp]. It **SHOULD** also contain the dns key, indicating support for DNS over HTTPS [RFC8484], to enable the use of HTTPS records with CONNECT-UDP.

```
{
  "dns": {
    "template": "https://doh.example.com/dns-query{?dns}",
  },
  "udp": {
    "template":
        "https://proxy.example.org/masque{?target_host,target_port}"
  },
  "ohttp": {
    "proxy": {
      "template": "https://proxy.example.org/ohttp{?request_uri}"
    }
  }
}
```

Figure 2: Example Proxy Access Description

The Oblivious Proxy Resources **MUST** allow use of the GET method to retrieve small JSON responses, and **SHOULD** make ample cache space available in order to cache Access Descriptions. Each proxy instance (as defined by its external-facing network interface) **MUST** share cache state among all clients to ensure that they use the same Access Descriptions for each Oblivious Request Resource.

Oblivious Proxies **MUST** preserve the ETag response header on cached responses, and **MUST** add an Age header ([I-D.ietf-httpbis-cache-19], Section 5.1) to all proxied responses. Oblivious Proxies **MUST** respect the "Cache-Control: immutable" directive, never revalidating these cached entries, and **MUST NOT** accept PUSH_PROMISE frames from the target.

Proxies **SHOULD** employ defenses against malicious attempts to fill the cache. Some possible defenses include:

* Rate-limiting each client's use of GET requests.

* Prioritizing preservation of cache entries that have been served to many clients, if eviction is required.

Oblivious Proxies that are not intended for general-purpose proxy usage **MAY** impose strict transfer limits or rate limits on HTTP CONNECT and CONNECT-UDP usage.

4.3. Client

The Client is assumed to know an "https" URI of an Oblivious Request Resource's Access Description. To use that Request Resource, it **MUST** perform the following "double-check" procedure:

1. Send a GET request to the Oblivious Proxy's template with request_uri set to the Access Description URI.

2. Record the response (A).

3. Check that response A's "Cache-Control" values indicates "public" and "immutable".

4. Fetch the Access Description URI from its origin via CONNECT-UDP, with "If-Match" set to response A's ETag.

5. Record the response (B).

6. Check that responses A and B were successful and the contents are identical, otherwise fail.

This procedure ensures that the Access Description is authentic and will be shared by all users of this proxy. Once response A or B expires, the client **MUST** refresh it before continuing to use this Access Description, and **MUST** repeat the "double-check" process if either response changes.

5. Example: Oblivious DoH

In this example, the client has been configured with an Oblivious DoH server and an Oblivious Proxy. The Oblivious DoH server is identified by an Access Description at "https://doh.example.com/config.json" with the following contents:

```
{
  "dns": {
    "template": "https://doh.example.com/dns-query{?dns}",
  },
  "ohttp": {
    "request": {
      "uri": "https://example.com/ohttp/",
      "key": "(KeyConfig in Base64)"
    }
  }
}
```

The Oblivious Proxy is identified as "proxy.example.org", which implies an Access Description at "https://proxy.example.org/.well-known/access-services". This resource's contents are:


```
{
  "dns": {
    "template": "https://proxy.example.org/dns-query{?dns}",
  },
  "udp": {
    "template":
        "https://proxy.example.org/masque{?target_host,target_port}"
  },
  "ohttp": {
    "proxy": {
      "template": "https://proxy.example.org/ohttp{?request_uri}"
    }
  }
}
```


The following exchanges then occur between the client and the proxy:

```
HEADERS
:method = GET
:scheme = https
:authority = proxy.example.org
:path = /ohttp?request_uri=https%3A%2F%2Fdoh.example.com%2Fconfig.json
accept: application/json

                              HEADERS
                              :status = 200
                              cache-control: public, immutable, \
                                  no-transform, s-maxage=86400
                              age: 80000
                              etag: ABCD1234
                              content-type: application/json
                              [Access Description contents here]

HEADERS
:method = CONNECT
:protocol = connect-udp
:scheme = https
:authority = proxy.example.org
:path = /masque?target_host=doh.example.com,target_port=443
capsule-protocol = ?1

                              HEADERS
                              :status = 200
                              capsule-protocol = ?1
```

The client now has a CONNECT-UDP tunnel to doh.example.com, over which it performs the following request using HTTP/3:

```
HEADERS
:method = GET
:scheme = https
:authority = doh.example.com
:path = /config.json
if-match = ABCD1234

                              HEADERS
                              :status = 200
                              cache-control: public, immutable, \
                                  no-transform, s-maxage=86400
                              etag: ABCD1234
                              content-type: application/json
                              [Access Description contents here]
```

Having successfully fetched the Access Description from both locations, the client confirms that:

* The responses are identical.

* The cache-control response from the proxy contained the "public" and "immutable" directives.

The client can now use the KeyConfig in this Access Description to reach the Oblivious DoH server, by forming Binary HTTP requests for "https://doh.example.com/dns-query" and delivering the encapsulated requests to "https://example.com/ohttp/" via the proxy.

6. Security Considerations

6.1. In scope

A malicious proxy could attempt to learn the contents of the oblivious request by forging an Access Description containing its own KeyConfig. This is prevented by the client's requirement that the KeyConfig be served to it by the configured origin over HTTPS (Section 4.3).

A malicious target could attempt to link multiple requests together by issuing each user a unique, persistent KeyConfig. This attack is prevented by the client's requirement that the KeyConfig be fresh according to the proxy's cache (Section 4.3).

A malicious target could attempt to rotate its entry in the proxy's cache in several ways:

* Using HTTP PUSH_PROMISE frames. This attack is prevented by disabling PUSH_PROMISE at the proxy (Section 4.2).

* By also acting as a client and sending requests designed to replace the Access Description in the cache before it expires:

  - By sending requests with a "Cache-Control: no-cache" or similar directive. This is prevented by the response's "Cache-Control: public, immutable" directives, which are verified by the client (Section 4.3).

  - By filling the cache with new entries, causing its previous Access Description to be evicted. Section 4.2 describes some possible mitigations.

A malicious client could use the proxy to send abusive traffic to any destination on the internet. Abuse concerns can be mitigated by imposing a rate limit at the proxy (Section 4.2).

6.2. Out of scope

This specification assumes that the client starts with identities of the proxy and target that are authentic and widely shared. If these identities are inauthentic, or are unique to the user, then the security goals of this specification are not achieved.

This specification assumes that at most a small fraction of clients are acting on behalf of a malicious target. If a large fraction of the clients are malicious, they could conspire to flood the proxy cache with entries that seem popular, leading to rapid eviction of the malicious target's Access Descriptions. Similar concerns apply if a malicious target can compel naive clients to fetch a very large number of Access Descriptions.

7. IANA Considerations

IANA is requested to open a Specification Required registry entitled "HTTP Access Service Descriptors", with the following initial contents:

Key	Specification
dns	(This document)
udp	(This document)
ip	(This document)
ohttp	(This document)

Table 1

IANA is requested to add the following entry to the "Well-Known URIs" registry

URI Suffix	Change Controller	Reference	Status	Related Information
access-services	IETF	(This document)	permanent	Sub-registry at (link)

Table 2

8. References

8.1. Normative References

[I-D.ietf-httpbis-cache] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Caching", Work in Progress, Internet-Draft, draft-ietf-httpbis-cache-19, 12 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cache-19>.

[I-D.ietf-httpbis-cache-19] Fielding, R. T., Nottingham, M., and J. Reschke, "HTTP Caching", Work in Progress, Internet-Draft, draft-ietf-httpbis-cache-19, 12 September 2021, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-cache-19>.

[I-D.ietf-masque-connect-udp]
Schinazi, D., "UDP Proxying Support for HTTP", Work in Progress, Internet-Draft, draft-ietf-masque-connect-udp-08, 21 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-masque-connect-udp-08>.

[I-D.ietf-ohai-ohttp] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-01, 15 February 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ohai-ohttp-01>.

[I-D.schwartz-masque-access-descriptions]
Schwartz, B. M., "HTTP Access Service Description Objects", Work in Progress, Internet-Draft, draft-schwartz-masque-access-descriptions-00, 7 April 2022, <https://datatracker.ietf.org/doc/html/draft-schwartz-masque-access-descriptions-00>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <https://www.rfc-editor.org/rfc/rfc7232>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8246] McManus, P., "HTTP Immutable Responses", RFC 8246, DOI 10.17487/RFC8246, September 2017, <https://www.rfc-editor.org/rfc/rfc8246>.

[RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <https://www.rfc-editor.org/rfc/rfc8484>.

8.2. Informative References

[I-D.wood-key-consistency] Davidson, A., Finkel, M., Thomson, M., and C. A. Wood, "Key Consistency and Discovery", Work in Progress, Internet-Draft, draft-wood-key-consistency-02, 4 March 2022, <https://datatracker.ietf.org/doc/html/draft-wood-key-consistency-02>.

Acknowledgments

TODO acknowledge.

Author's Address

Benjamin M. Schwartz
Google LLC

Email: bemasc@google.com