Oblivious HTTP Application Intermediation
Internet-Draft
Intended status: Informational
Expires: 28 January 2023

T. Pauly
Apple Inc.
T. Reddy
Nokia

27 July 2022

Discovery of Oblivious Services via Service Binding Records
draft-pauly-ohai-svcb-config-03

Abstract

This document defines a parameter that can be included in SVCB and HTTPS DNS resource records to denote that a service is accessible using Oblivious HTTP, by offering an Oblivious Gateway Resource through which to access the target. This document also defines a mechanism to learn the key configuration of the discovered Oblivious Gateway Resource.

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-pauly-ohai-svcb-config/.

Discussion of this document takes place on the Oblivious HTTP Application Intermediation Working Group mailing list (mailto:ohai@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ohai/.

Source for this draft and an issue tracker can be found at https://github.com/tfpauly/draft-ohai-svcb-config.

Status of This Memo

This Internet-Draft will expire on 28 January 2023.

Table of Contents

1. Introduction

Oblivious HTTP [OHTTP] allows clients to encrypt messages exchanged with an Oblivious Target Resource (target). The messages are encapsulated in encrypted messages to an Oblivious Gateway Resource (gateway), which gates access to the target. The gateway is access via an Oblivious Relay Resource (relay), which proxies the encapsulated messages to hide the identity of the client. Overall, this architecture is designed in such a way that the relay cannot inspect the contents of messages, and the gateway and target cannot discover the client's identity.

Since Oblivious HTTP deployments will often involve very specific coordination between clients, relays, and gateways, the key configuration can often be shared in a bespoke fashion. However, some deployments involve clients discovering oblivious targets and their assoicated gateways more dynamically. For example, a network may want to advertise a DNS resolver that is accessible over Oblivious HTTP and applies local network resolution policies via mechanisms like Discovery of Designated Resolvers ([DDR]). Clients can work with trusted relays to access these gateways.

This document defines a mechanism to advertise that an HTTP service supports Oblivious HTTP using DNS records, as a parameter that can be included in SVCB and HTTPS DNS resource records [SVCB] (Section 4). The presence of this parameter indicates that a service can act as an oblivious target and has an oblivious gateway that can provide access to the target.

The client learns the URI to use for the oblivious gateway using a well-known URI [WELLKNOWN], "oblivious-gateway", which is accessed on the oblivious target (Section 5).

This document also defines a way to fetch an oblivious gateway's key configuration from the oblivious gateway (Section 6).

This mechanism does not aid in the discovery of oblivious relays; relay configuration is out of scope for this document. Models in which this discovery mechanism is applicable are described in Section 3.

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. Applicability

There are multiple models in which the discovery mechanism defined in this document can be used.

* Upgrading non-oblivious HTTP to oblivious HTTP. In this model, the client intends to communicate with a specific target service, and prefers to use oblivious HTTP if it is available. The target service has an oblivious gateway that it offers to allow access using oblivious HTTP. Once the client learns about the oblivious gateway, it "upgrades" to using oblivious HTTP to access the target service.

* Discovering alternative oblivious HTTP services. In this model, the client has a default oblivious target service that it uses. For example, this may be a public DNS resolver that is accessible over oblivious HTTP. The client is willing to use alternative oblivious target services if they are discovered, which may provide more optimized or more relevant responses.

In both of these deployment models, the client is assumed to already know of an oblivious relay that it trusts and works with. This oblivious relay either needs to provide generic access to oblivious gateways, or provide a service to clients to allow them to check which gateways are accessible.

4. The oblivious SvcParamKey

The "oblivious" SvcParamKey (Section 8) is used to indicate that a service described in an SVCB record can be accessed as an oblivious target using an associated gateway. The service that is queried by the client hosts one or more target resources.

In order to access the service’s target resources obliviously, the client needs to send encapsulated messages to the gateway resource and the gateway’s key configuration (both of which can be retrieved using the method described in Section 6).

Both the presentation and wire format values for the "oblivious" parameter MUST be empty.

The "oblivious" parameter can be included in the mandatory parameter list to ensure that clients that do not support oblivious access do not try to use the service. Services that mark the oblivious parameter as mandatory can, therefore, indicate that the service might not be accessible in a non-oblivious fashion. Services that are intended to be accessed either obliviously or directly SHOULD NOT mark the "oblivious" parameter as mandatory. Note that since multiple SVCB responses can be provided for a single query, the oblivious and non-oblivious versions of a single service can have different SVCB records to support different names or properties.

The media type to use for encapsulated requests made to a target service depends on the scheme of the SVCB record. This document defines the interpretation for the "https" [SVCB] and "dns" [DNS-SVCB] schemes. Other schemes that want to use this parameter MUST define the interpretation and meaning of the configuration.

4.1. Use in HTTPS service records

For the "https" scheme, which uses the HTTPS RR type instead of SVCB, the presence of the "oblivious" parameter means that the target being described is an Oblivious HTTP service that is accessible using the default "message/bhttp" media type [OHTTP] [BINARY-HTTP].

For example, an HTTPS service record for svc.example.com that supports an oblivious gateway could look like this:

```
svc.example.com. 7200  IN HTTPS 1 . ( alpn=h2 oblivious )
```

A similar record for a service that only support oblivious connectivity could look like this:

```
svc.example.com. 7200  IN HTTPS 1 . ( mandatory=oblivious oblivious )
```

4.2. Use in DNS server SVCB records

For the "dns" scheme, as defined in [DNS-SVCB], the presence of the "oblivious" parameter means that the DNS server being described is an Oblivious DNS over HTTP (DoH) service. The default media type expected for use in Oblivious HTTP to DNS resolvers is "application/dns-message" [DOH].

In order for DNS servers to function as oblivious targets, their associated gateways need to be accessible via an oblivious relay. Encrypted DNS servers used with the discovery mechanisms described in this section can either be publicly accessible, or specific to a network. In general, only publicly accessible DNS servers will work as oblivious DNS servers, unless there is a coordinated deployment with an oblivious relay that is also hosted within a network.

4.2.1. Use with DDR

Clients can discover an oblivious DNS server configuration using DDR, by either querying _dns.resolver.arpa to a locally configured resolver or querying using the name of a resolver [DDR].

For example, a DoH service advertised over DDR can be annotated as supporting oblivious resolution using the following record:

```
_dns.resolver.arpa  7200  IN SVCB 1 doh.example.net (
     alpn=h2 dohpath=/dns-query{?dns} oblivious )
```

Clients still need to perform some verification of oblivious DNS servers, such as the TLS certificate check described in [DDR]. This certificate check can be done when looking up the configuration on the gateway as described in Section 6, which can either be done directly, or via the relay or another proxy to avoid exposing client IP addresses.

For the case of DoH servers, clients also need to ensure that they are not being targeted with unique DoH paths that would reveal their identity. See Section 7 for more discussion.

4.2.2. Use with DNR

The SvcParamKeys defined in this document also can be used with Discovery of Network-designated Resolvers (DNR) [DNR]. In this case, the oblivious configuration and path parameters can be included in DHCP and Router Advertisement messages.

While DNR does not require the same kind of verification as DDR, clients that learn about DoH servers still need to ensure that they are not being targeted with unique DoH paths that would reveal their identity. See Section 7 for more discussion.

5. Gateway Location

Clients that know a service is available as an oblivious target via discovery through the "oblivious" parameter in a SVCB or HTTPS record need to know the location of the associated oblivious gateway before sending oblivious requests.

By default, the oblivious gateway for an oblivious target is defined as a well-known resource ([WELLKNOWN]) on the target, "/.well-known/oblivious-gateway".

Commonly, servers will not want to actually operate the oblivious gateway on a well-known URI. In such cases, servers can use 3xx redirection responses (Section 15.4 of [HTTP]) to direct clients and relays to the correct location of the oblivious gateway.

Generally, the first request a client will make will be a GET request to discover the key configuration, described in Section 6. This initial request also provides a convenient way for clients to learn about the redirect from the well-known resource, if there is a redirect. When clients work with their oblivious relays to send oblivious requests to the gateway, clients can communicate this redirected gateway URI.

6. Key Configuration Fetching

Clients also need to know the key configuration of an oblivious gateway before sending oblivious requests.

In order to fetch the key configuration of an oblivious gateway discovered in the manner described in Section 5, the client issues a GET request to the URI of the gateway specifying the "application/ohttp-keys" ([OHTTP]) media type in the Accept header.

For example, if the client knows an oblivious gateway URI, "https://osvc.example.com/gateway", it could fetch the key configuration with the following request:

```
GET /gateway HTTP/1.1
Host: osvc.example.com
Accept: application/ohttp-keys
```

Oblivious gateways that coordinate with targets that advertise oblivious support SHOULD support GET requests for their key configuration in this manner, unless there is another out-of-band configuration model that is usable by clients. Gateways respond with their key configuration in the response body, with a content type of "application/ohttp-keys".

Clients can either fetch this key configuration directly, or do so via a proxy in order to avoid the server discovering information about the client’s identity. See Section 7 for more discussion of avoiding key targeting attacks.

7. Security and Privacy Considerations

Attackers on a network can remove SVCB information from cleartext DNS answers that are not protected by DNSSEC [DNSSEC]. This can effectively downgrade clients. However, since SVCB indications for oblivious support are just hints, a client can mitigate this by always checking for oblivious gateway configuration Section 6 on the well-known gateway location Section 5. Use of encrypted DNS along with DNSSEC can also be used as a mitigation.

When discovering designated oblivious DNS servers using this mechanism, clients need to ensure that the designation is trusted in lieu of being able to directly check the contents of the gateway server's TLS certificate. See Section 4.2.1 for more discussion, as well as the Security Considerations of [SVCBDNS].

For oblivious DoH servers, an attacker could use unique DoH path values to target or identify specific clients. Clients can mitigate such attacks in several ways. Some options include: only allow common DoH paths (such as the de-facto default "/dns-query{?dns}"); performing consistency checks by fetching the information about the resolver over multiple resolution paths; or coordinating with a trusted oblivious relay to validate that DoH paths are common across clients using the same gateway.

As discussed in [OHTTP], client requests using Oblivious HTTP can only be linked by recognizing the key configuration. In order to prevent unwanted linkability and tracking, clients using any key configuration discovery mechanism need to be concerned with attacks that target a specific user or population with a unique key configuration.

There are several approaches clients can use to mitigate key targeting attacks. [CONSISTENCY] provides an analysis of the options for ensuring the key configurations are consistent between different clients. Clients SHOULD employ some technique to mitigate key targeting attack. Oblivious gateways that are detected to use targeted key configurations per-client MUST NOT be used.

When clients fetch a gateway's configuration (Section 6), they can expose their identity in the form of an IP address if they do not connect via a proxy or some other IP-hiding mechanism. In some circumstances, this might not be a privacy concern, since revealing that a particular client IP address is preparing to use an Oblivious HTTP service can be expected. However, if a client is otherwise trying to obfuscate its IP address or location (and not merely decouple its specific requests from its IP address), or revealing its IP address will increase the risk of a key targeting attack (if a gateway service is trying to differentiate traffic across client IP addresses), a proxy or similar mechanism can be used to fetch the gateway's configuration.

8. IANA Considerations

8.1. SVCB Service Parameter

IANA is requested to add the following entry to the SVCB Service Parameters registry ([SVCB]).

Number	Name	Meaning	Reference
TBD	oblivious	Denotes that a service operates an oblivious HTTP target	(This document)

Table 1

8.2. Well-Known URI

IANA is requested to add one new entry in the "Well-Known URIs" registry [WELLKNOWN].

URI suffix: oblivious-gateway

Change controller: IETF

Specification document: This document

Status: permanent

Related information: N/A

9. References

9.1. Normative References

[BINARY-HTTP] Thomson, M. and C. A. Wood, "Binary Representation of HTTP Messages", Work in Progress, Internet-Draft, draft-ietf-httpbis-binary-message-06, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpbis-binary-message-06>.

[DDR] Pauly, T., Kinnear, E., Wood, C. A., McManus, P., and T. Jensen, "Discovery of Designated Resolvers", Work in Progress, Internet-Draft, draft-ietf-add-ddr-08, 5 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-add-ddr-08>.

[DNR] Boucadair, M., Reddy, T., Wing, D., Cook, N., and T. Jensen, "DHCP and Router Advertisement Options for the Discovery of Network-designated Resolvers (DNR)", Work in Progress, Internet-Draft, draft-ietf-add-dnr-12, 24 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-add-dnr-12>.

[DNS-SVCB] Schwartz, B., "Service Binding Mapping for DNS Servers", Work in Progress, Internet-Draft, draft-ietf-add-svcb-dns-06, 5 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-add-svcb-dns-06>.

[DOH] 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>.

[HTTP] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[OHTTP] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-02, 10 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ohai-ohttp-02>.

[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>.

[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>.

[SVCB] Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-10, 24 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-svcb-https-10>.

[WELLKNOWN] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/rfc/rfc8615>.

9.2. Informative References

[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>.

[DNSSEC] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <https://www.rfc-editor.org/rfc/rfc4033>.

[SVCBDNS] Schwartz, B., "Service Binding Mapping for DNS Servers", Work in Progress, Internet-Draft, draft-ietf-add-svcb-dns-06, 5 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-add-svcb-dns-06>.

Authors' Addresses

Tommy Pauly
Apple Inc.
Email: tpauly@apple.com

Tirumaleswar Reddy
Nokia
Email: kondtir@gmail.com

ohai
Internet-Draft
Intended status: Standards Track
Expires: 15 November 2023

T. Reddy
Nokia
D. Wing
Citrix
M. Boucadair
Orange
R. Polli
Team Digitale, Italian Government

14 May 2023

Oblivious Relay Feedback
draft-rdb-ohai-feedback-to-proxy-09

Abstract

Servers often rate-limit incoming requests, for example, rate-limit based upon the source IP address to provide equitable service to clients. However, oblivious HTTP removes the ability for the server to distinguish amongst clients so the server can only rate-limit traffic from an Oblivious Relay Resource. This harms all clients behind that Oblivious Relay Resource.

This specification enables a server to convey rate-limit information to an Oblivious Relay Resource, which can use it to apply rate-limit policies on clients. Cooperating Oblivious Relay Resources can thus provide more equitable service to their distinguishable clients without impacting on all clients behind that Oblivious Relay Resource.

Status of This Memo

Table of Contents

1. Introduction

Oblivious HTTP [OHTTP] requires three parties to exchange HTTP messages: the client, the relay, and the target (formally, the Oblivious Gateway Resource and Oblivious Target Resource). Oblivious HTTP enables a client to send requests to a target in such a way that the target cannot tell whether two requests came from the same client, and the relay cannot see the contents of the requests.

Since clients are located behind a relay, a target cannot distinguish between well-behaving and malicious clients: an unexpected behavior from one or more clients can then impact on all the intermediated clients, as described in Section 8.2.1 of [OHTTP]. This can be problematic when the target implements rate-limiting policies based on an information masked by the intermediary, such as the source IP address.

This document defines a mechanism that allows Oblivious Gateway and Target Resource to provide rate-limit information to an Oblivious Relay Resource via the RateLimit fields defined in [RATELIMIT]. This is useful when such servers identify traffic anomalies or unexpected request volumes. An Oblivious Relay Resource can then use this information to apply rate-limit policies on clients.

While [RATELIMIT] provides enough information to generic clients to shape their request policy and avoid being throttled out, this specification allows an Oblivious Gateway and Target Resource to indicate their RateLimit information is intended for the Oblivious Relay Resource (rather than to the client).

How an Oblivious Relay Resource can use this information to avoid being throttled out or shape its request policy is outside the scope of this specification.

The proposed mechanism does not address on purpose the attack of an offending client attacking the server (e.g., the client is using an abnormal header that matches an attack pattern) because the application of the rate-limit can potentially allow the target to take advantage of the differential treatment applied by the relay to de-anonymize the client.

2. Terminology

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.

The terms "content", "receiver", "request", and "response" are to be interpreted as described in [HTTP].

The terms "Encapsulated request", "Encapsulated response", "Oblivious Relay Resource", "Oblivious Gateway Resource", "Oblivious Target Resource", and "Client" are to be interpreted as described in Section 2.2 of [OHTTP].

The collective term "Oblivious resource" indicates either an "Oblivious Gateway Resource" or an "Oblivious Target Resource".

The terms "quota policy", "service limit", "expiring limit", and "RateLimit fields" are to be interpreted as described in [RATELIMIT].

This document uses the Integer type from [STRUCTURED-FIELDS].

3. Providing RateLimit Information to an Oblivious Proxy

An Oblivious resource that uses RateLimit fields [RATELIMIT] to return service limit information MAY add the "ohttp-target" quota policy parameter defined in Section 4 to signal to the receiver that the associated quota policy is intended for an Oblivious Relay Resource. For example, when an Oblivious target identifies a high frequency or high volume anomalies in the HTTP requests it would include the "ohttp-target" parameter.

The term "Oblivious Relay Feedback" denotes both the mechanism described in this specification and the complete set of RateLimit fields together with the "ohttp-target" parameter.

To know whether the RateLimit fields provides Oblivious Relay Feedback (see Section 3.1), an Oblivious Relay Resource MUST:

1. Identify the quota policy associated with the expiring limit.

2. Check whether the "ohttp-target" parameter is present and its syntax is correct.

In the example shown in Figure 1, the expiring limit value is "100", so the associated quota policy is the second one. This quota policy includes the "ohttp-target" parameter: this indicates that the RateLimit fields are intended for an Oblivious Relay Resource.

```
RateLimit-Limit: 100
RateLimit-Policy: 10;w=1, 100;w=60;ohttp-target
RateLimit-Remaining: 8
RateLimit-Reset: 15
```

Figure 1: An Example of Oblivious Proxy Feedback.

4. The ohttp-target Quota Policy Parameter

4.1. ohttp-target Parameter

The following quota policy parameter is defined for the RateLimit-Policy field [RATELIMIT]:

ohttp-target: Indicates that the associated quota policy provides Oblivious Relay Feedback and its value is empty. This parameter is OPTIONAL. It indicates that RateLimit fields are applicable to all the clients that are serviced by the same Oblivious Relay Resource. In order to offer better privacy, a relay SHOULD rate-limit all of the clients it is serving provided it will continue to forward a high volume of messages from a large number of clients even after applying the rate-limiting. A relay MUST NOT rate-limit a client or a subset of clients it is serving based on the RateLimit information from the server.

The "ohttp-target" parameter MUST NOT appear more than once in a quota policy. If the parameter is malformed or has a value, it MUST be ignored, and the receiving Oblivious Relay Resource MUST NOT attempt to fix neither the parameter nor its value. That is, the RateLimit fields must not be considered as providing Oblivious Relay Feedback.

4.2. Processing the ohttp-target Parameter

An Oblivious Relay Resource receiving RateLimit fields providing Oblivious Relay Feedback will do the following:

* It MUST remove all the RateLimit fields from the response, since they are not intended to be forwarded to clients.

An Oblivious gateway resource receiving RateLimit fields providing Oblivious Relay Feedback MUST proceed as follows:

1. Remove the RateLimit fields from the HTTP response, since they are not intended to be forwarded to the client. It, then, encapsulates the HTTP response.

2. Add the above RateLimit fields to the response containing the Encapsulated Response sent to the Oblivious Relay Resource, so that the relay can access them.

If the RateLimit fields along with the "ohttp-target" parameter are generated by the Oblivious gateway resource before removing the protection (including being unable to remove the encapsulation for any reason)(Section 6.2 of [OHTTP]), it will result in the RateLimit fields added in the response being sent without protection in response to a POST request from a client.

While this specification does not mandate specific traffic shaping actions for Oblivious proxies in addition to the ones indicated in [RATELIMIT], an Oblivious Relay Resource failing to reshape traffic from all the clients according to the received Oblivious Relay Feedback can experience different levels of service denial by the Oblivious gateway and target resources. There is no explicit mechanism for an Oblivious Relay Resource to indicate to the server that the rate-limit information was processed or was ignored.

5. The 'attack-severity' Quota Policy Parameter

The following quota policy parameter is defined for the RateLimit-Policy field defined in [RATELIMIT]:

attack-severity: Is used by the Oblivious Gateway Resource or Target Resource to convey the severity of the attack. This parameter is OPTIONAL.

attack-severity = sf-string

Note that sf-string is defined in Section 3.3.3 of [STRUCTURED-FIELDS].

The value of the "attack-severity" parameter is a String (Section 3.3.3 of [RFC8941]) that takes one of the values defined in [SEVERITY]. This parameter MUST NOT appear more than once in a quota policy. If the parameter is malformed or its value is invalid, the parameter MUST be ignored, and the relays MUST NOT attempt to fix neither the parameter nor the value.

The severity of the attack may be used as an operational alert to the Oblivious Relay Resource's security operation team and to automatically trigger the rate-limit mitigation action accordingly. Irrespective of the security level at which a rate-limit is triggered, the relay must follow the rules discussed in Section 4.1 to offer privacy to the clients it is serving. The severity of the attack can also be used to identify a "ramp-up" attack versus a "blast" attack.

6. Use of The ohttp-target Quota Policy Parameters: An Example

The example depicted in Figure 2 illustrates the use of the "ohttp-target" parameter. An oblivious target resource receives unexpected request volumes and uses the source IP address to identify that these were Encapsulated Requests decapsulated by an oblivious gateway resource. The Oblivious target resource adds the RateLimit fields along with the "ohttp-target" quota policy parameter to the HTTP response. The oblivious gateway resource proceeds as follows:

1. Copy the RateLimit fields from the original response.

2. Remove them from the original response before encapsulating it.

3. Generate a single 200 response containing the encapsulated response for the Oblivious Relay Resource along with the copied RateLimit fields.

```
+----+            +----------+       +----------+    +----------+
| C  |            | Relay    |       | Gateway  |    | Target   |
|    |            | Resource |       | Resource |    | Resource |
+-+--+            +----+-----+       +-----+----+    +-----+----+
  |                    |                   |                |
  | Encapsulated       |                   |                |
  +------------------->|                   |                |
  |  Request           |                   |                |
  |                    | Encapsulated      |                |
  |                    +------------------>|                |
  |                    |  Request          |                |
  |                    |                   | Request        | .----------------.
  |                    |                   +-------------->| | Identify       |
  |                    |                   |               +-+ unexpected     |
  |                    |                   |               | | request volumes|
  |                    |                   |  2xx response | '----------------'
  |                    |                   <--------------+
  |                    | 200 response with |                |
  |                    | RateLimit-Limit,  |                |
  |                    | RateLimit-Policy  |                |
  |                    | fields and the    |                |
  |                    | ohttp-target      |                |
  |                    | parameter         |                |
  |                    <------------------+                |
.--------------------. | Encapsulated 2xx  |                |
| Process            | |    response       |                |
| ohttp-target       +-+                   |                |
| and rate-limit     | |                   |                |
| requests from      | |                   |                |
| clients            | |                   |                |
'--------------------' |                   |                |
  |                    |                   |                |
  | Encapsulated 2xx   |                   |                |
  <--------------------+                   |                |
  |     response       |                   |                |
  |                    |                   |                |
```

Figure 2: An Example of Ratelimit Feedback to Proxy

The response that is generated by the Oblivious gateway resource is depicted in Figure 3. This response includes an unregistered, informative "comment" quota policy parameter providing the rationale for the "attack- severity".

```
=============== NOTE: '\' line wrapping per RFC 8792 ================

  HTTP/1.1 200 OK
  Date: Wed, 27 March 2022 04:45:07 GMT
  Cache-Control: private, no-store
  RateLimit-Limit: 10
  RateLimit-Policy: 10;ohttp-target;attack-severity="high";\
comment="Bandwidth Limit Exceeded"
  Content-Type: message/ohttp-res
  Content-Length: 38 <content is the encapsulated 400 response>
  ...encrypted content...
```

Figure 3: Example of a Response

7. Ohttp-Outside-Encap Header

The "Ohttp-Outside-Encap" HTTP header field is defined in this specification (Section 9.2.1). Its purpose is to signal which HTTP headers will be removed by the Oblivious gateway. It is intended only for use in requests to an Oblivious target.

When an Oblivious gateway resource sends an HTTP request to an Oblivious target, it adds the "Ohttp-Outside-Encap" header to indicate which headers will be removed from the response.

On receipt of an HTTP response from the Oblivious target resource, the Oblivious gateway resource copies those header fields and values, and it then removes them from the HTTP response. The Oblivious gateway then encapsulates the HTTP response. The Oblivious gateway resource adds the copied header fields and values to the response containing the Encapsulated Response, so that the Oblivious Relay Resource can access and act on them.

The "Ohttp-Outside-Encap" header is useful in deployments where the Oblivious gateway resource and Oblivious target resource are managed by separate entities.

Figure 4 describes the syntax. The syntax of this header field conforms to [RFC8941]. It is a List ([RFC8941], Section 3.1):

```
Ohai-Outside-Encap =  sf-list
```

Figure 4: Ohttp-Outside-Encap Header Syntax

An example is illustrated below:

```
Ohttp-Outside-Encap: RateLimit-Limit,RateLimit-Remaining,RateLimit-Reset,RateLimi
t-Policy
```

8. Security Considerations

The security considerations for the Oblivious HTTP protocol (Section 8 of [OHTTP]) as well as the ones for RateLimit fields (Section 6 of [RATELIMIT]) apply. The following sub-sections discuss security considerations specific to this specification.

8.1. Avoid Correlation

If the Oblivious Relay Resource attempts to divide the rate limit fairly among the active clients, the timing pattern of requests can possibly be strongly correlated by the to gateway to de-anonymize clients. As discussed in Section 5 of [OHTTP], the relay can delay requests before forwarding them to migitate the attack as it will likely increase the anonymity set into which each request is attributed.

8.2. Client and Oblivous Proxy Collusion

While Oblivious HTTP relies upon an Oblivious Relay Resource to prevent leaking the client identity to the Oblivious resources, it might be the case that the Oblivious Relay Resource colludes with clients in attacking Oblivious resources. RateLimit fields might disclose operational capacity information useful to design denial of service attacks or to circumvent defensive measures put in place by the Oblivious resources (Section 6.2 of [RATELIMIT]). The Oblivious target and gateway resources SHOULD convey Oblivious Relay Feedback only to trusted Oblivious proxies.

8.3. Attack Categories and Mitigation

Attacks against the Oblivious Gateway and Target Resources can be classified into three primary categories:

1. A client deliberately sends a malformed Encapsulated Request causing decryption failure or decryption overload failure on the oblivious gateway resource. This causes the oblivious gateway resource to send an error status code back to the oblivious relay.

2. A client deliberately sends an HTTP request that causes an HTTP error on the oblivious target resource. This might be a malformed HTTP request, or request for a missing resource.

3. A botnet performing an application layer denial of service attack (e.g., HTTP flood) against an Oblivious resource. Because each bot in a botnet makes seemingly legitimate network requests the traffic may appear "normal" in origin, nonetheless as a whole it not only can saturate the Oblivious resources, but also makes appear the Oblivious Relay Resource as an attacker. There might be too many requests from a single client, too many requests from the clients behind the same Oblivious Relay Resource, or too many requests from all clients on the Internet.

Typicially, a DDoS attack mitigation service (DMS) would be used to analyze and filter application-level DDoS attacks. DMSes use rate-limiting as one of the techniques to mitigate the impact of DDoS attacks. A DMS could trigger the rate-limiting only when the capacity of the mitigation service is exceeded (or exceeding a threshold). However, some DMSes may use rate-limiting as a first line of defense against attacks, while other DMSes may use it as a secondary measure when other techniques (e.g., traffic filtering) are not sufficient. A DMS can leverage the mechanism defined in this document to avoid rate-limiting traffic from the Oblivious Relay Resource. If the relay fails to act, it can potentially experience different levels of service denial by the DMS.

9. IANA Considerations

9.1. RateLimit Parameter Value Registrations

This specification requests IANA to add the following parameters to the "Hypertext Transfer Protocol (HTTP) RateLimit Parameters" registry defined in [RATELIMIT].

Field Name	Parameter Name	Description	Specification
RateLimit-Policy	ohttp-target	ohttp ratelimit	Section 3 of this document
RateLimit-Policy	attack-severity	ohttp ratelimit	Section 5 of this document

9.2. Registration of new HTTP Header Field

9.2.1. Ohttp-Outside-Encap Header

This section describes a header field for registration in the Permanent Message Header Field Registry [RFC3864].

Header field name
Ohttp-Outside-Encap

Applicable protocol
http

Status
Standard

Author/Change controller
IETF

Specification document(s)
RFC XXXX

Related information
This header field is only used for Oblivious HTTP.

10. Acknowledgements

Thanks to Lucas Pardue, Rich Salz, Martin Thomson, Christopher A. Wood, Ben Schwartz, Ted Hardie and Brandon Williams for the discussion and comments.

11. References

11.1. Normative References

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

[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>.

[RATELIMIT] Polli, R. and A. M. Ruiz, "RateLimit Fields for HTTP", Work in Progress, Internet-Draft, draft-ietf-httpapi-ratelimit-headers-05, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpapi-ratelimit-headers-05>.

[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/info/rfc2119>.

[RFC3864] Klyne, G., Nottingham, M., and J. Mogul, "Registration Procedures for Message Header Fields", BCP 90, RFC 3864, DOI 10.17487/RFC3864, September 2004, <https://www.rfc-editor.org/info/rfc3864>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[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/info/rfc8174>.

[RFC8941] Nottingham, M. and P. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/info/rfc8941>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/info/rfc9110>.

[STRUCTURED-FIELDS] Nottingham, M. and P-H. Kamp, "Structured Field Values for HTTP", RFC 8941, DOI 10.17487/RFC8941, February 2021, <https://www.rfc-editor.org/rfc/rfc8941>.

11.2. Informative References

[SEVERITY] IANA, "Incident Object Description Exchange Format v2 (IODEF)", <https://www.iana.org/assignments/iodef2/iodef2.xhtml#businessimpact-severity>.

Authors' Addresses

Tirumaleswar Reddy
Nokia
India
Email: kondtir@gmail.com

Dan Wing
Citrix Systems, Inc.
4988 Great America Pkwy
Santa Clara, CA 95054
United States of America
Email: danwing@gmail.com

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Roberto Polli
Team Digitale, Italian Government
Email: robipolli@gmail.com

ohai
Internet-Draft
Intended status: Standards Track
Expires: 22 April 2023

B. M. Schwartz
Google LLC

19 October 2022

Key Consistency by Double-Checking via a Semi-Trusted Proxy
draft-schwartz-ohai-consistency-doublecheck-03

Abstract

Several recent IETF privacy protocols require clients to acquire bootstrap information for a service in a way that guarantees both authenticity and consistency, e.g., encrypting to the same key as many other users. This specification defines a procedure for transferring arbitrary HTTP resources in a manner that provides these guarantees. The procedure relies on access to a semi-trusted HTTP proxy, under the same security assumptions as an Oblivious HTTP Relay.

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] enables an HTTP client to send requests to a target in such a way that no single party can determine both the identity of the client and the contents of its request. Oblivious HTTP identifies four parties to each exchange: the Client, Relay, Gateway and Target.

In order for Oblivious HTTP's privacy guarantees to hold, several preconditions must be met:

1. The Client must be one of many users who might be using the Relay. Otherwise, use of the Relay reveals the user's identity to the Gateway.

2. The Client must hold an authentic KeyConfig for the Gateway. Otherwise, the Client could be speaking to the Relay, impersonating the Gateway.

3. All users of the Relay must be equally likely to use this Gateway, KeyConfig, and Target, regardless of their prior activity. Otherwise, the encrypted request identifies the Client to the Gateway.

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

The Privacy Pass protocol [I-D.ietf-privacypass-protocol] allows a Client to retrieve tokens from an Issuer, and present them to an Origin, in such a way that the Origin can verify the validity of the tokens but cannot identify the Client, even if it colludes with the Issuer. Privacy Pass requires a similar set of preconditions for its privacy guarantees:

1. The Client's transport to the Origin must not reveal the client's identity (e.g. via the client IP address).

2. The Client must hold an authentic Issuer Directory Object. Otherwise, issuance will fail (resulting in a denial of service, not a privacy violation).

3. All Clients with the same transport metadata must be using the same Issuer Directory Object for this Issuer. Otherwise, these factors together uniquely identify the client.

4. (optional) The Origin should not learn the IP addresses of the Clients collectively, as above.

Pre-requisite 1 on this list can be achieved by employing a shared transport proxy, on the assumption that at least one of the proxy or the Origin is non-malicious. (This is the same assumption that an Oblivious HTTP Client makes about its Relay).

This specification assumes that a shared, "semi-trusted" proxy is available (fulfilling precondition 1 on each list). It defines a three-party protocol for a Client, Proxy, and Origin to fetch an HTTP resource in a manner that ensures authenticity and consistency among Clients of a single Proxy. When used to initialize Oblivious HTTP or Privacy Pass, it guarantees preconditions 2, 3, and optionally 4.

The input to this protocol is a Desired Resource URI: an "https" URI on the Origin that is assumed to have been distributed to Clients in a globally consistent fashion. For example, this URI might be the default value of a software setting, or it might be published on a third party's website. This specification allows Clients to convert the static, long-lived Desired Resource URI into a fresh copy of that resource, i.e. to fetch the resource.

In principle, the Desired Resource could have been distributed directly through the assumed globally consistent channel. However, these ad hoc publication channels may not be fast enough to support frequent updates (e.g., key rotations), especially if updates require user intervention.

This draft is an instantiation of the Shared Proxy with Key Confirmation strategy defined in Section 4.3 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 HTTP GET requests for the Desired Resource. One uses the Proxy as an HTTP request proxy (terminating TLS and HTTP), and the other optionally uses it as a transport proxy (using TLS end-to-end with the Origin). The Proxy will forward the first request to the Origin only if the response is not in cache.

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

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 Origin, preventing forgeries.

4. Requirements

4.1. Origin

The Desired Resource MUST include a "strong validator" ETag (Section 2 of [RFC7232]) in any response to a GET request, 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" [RFC9111][RFC8246]. For efficiency reasons, the max age SHOULD be at least 60 seconds, and preferably much longer.

If the Desired Resource changes, and the Origin receives a request for the resource 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. Proxy

The Proxy MUST offer HTTP request proxy capability, and SHOULD also offer TCP proxy, UDP proxy, and encrypted DNS resolution capabilities. (An associated encrypted DNS resolver enables reliable use of HTTPS records [SVCB], improves metadata confidentiality, and allows EDNS Client Subnet to be disabled reliably.) For example, a proxy in the recommended configuration could be described by the following Access Service Description

[I-D.schwartz-masque-access-descriptions]:

```
{
  "http": {
    "template": "https://relay.example.org/http{?target_uri}"
  },
  "tcp": {
    "template":
        "https://proxy.example.org/tcp{?target_host,tcp_port}"
  },
  "udp": {
    "template":
        "https://proxy.example.org/masque{?target_host,target_port}"
  },
  "dns": {
    "template": "https://doh.example.com/dns-query{?dns}",
  }
}
```

Figure 2: Example Proxy Access Service Description

The Proxy MUST allow use of the GET method to proxy small responses, and SHOULD make ample cache space available in order to avoid eviction of Desired Resources. The proxy SHOULD share cache state among all clients, to ensure that they observe the same resource contents. If the cache must be partitioned for architectural or performance reasons, operators SHOULD keep the number of users in each partition as large as possible.

Proxies MUST preserve the ETag response header on cached responses, and MUST add an Age header ([RFC9111], Section 5.1) to all proxied responses. Proxies MUST respect the "Cache-Control: immutable" directive, and MUST NOT revalidate fresh immutable cache entries in response to any incoming requests. (Note that this is different from the general recommendation in Section 2.1 of [RFC8246]). Proxies also 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.

Proxies that are not intended for general-purpose use MAY impose strict transfer limits or rate limits on transport proxy usage.

If the Proxy offers an Encrypted DNS service, it MUST NOT enable EDNS Client Subnet support [RFC7871].

4.3. Client

The Client is assumed to know the "https" URI of the Desired Resource. To retrieve that resource, it MUST perform the following "double-check" procedure:

1. Send a GET request for the Desired Resource URL via the Proxy's HTTP request proxy function.

2. Record the response (A).

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

4. Establish a transport tunnel through the Proxy to the Origin (OPTIONAL).

5. Fetch the Desired Resource (through this tunnel if available), using a GET request with "If-Match" set to response A's ETag.

6. Record the response (B).

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

This procedure ensures that the retrieved copy of the Desired Resource is authentic and is identical to the one held by any other users of this proxy. Once response A or B expires, the client MUST refresh it before continuing to use this resource, and MUST repeat the "double-check" process if either response changes.

Clients MUST perform each fetch to the Origin (step 4) as a fully isolated request. Any state related to this Origin (e.g., cached DNS records, CONNECT-UDP tunnels, QUIC transport state, TLS session tickets, HTTP cookies) MUST NOT be shared with prior or subsequent requests.

4.4. Negative responses

If the Desired Resource does not exist, the Double Check requirements apply without modification. The Cache-Control headers ensure that the negative response (e.g., HTTP 404) is cacheable regardless of its status code. If Double Check succeeds with a negative response, the Client can be confident that any other Clients of this proxy are holding the same negative response.

5. Example: Oblivious DoH

In this example, the client has been configured with a Proxy via the following Access Service Description:

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

The client has recently been instructed to use an Encrypted DNS server identified as "doh.example.com" that might support Oblivious DoH. To discover and access this Oblivious DoH service, the client attempts to retrieve two Desired Resources on the Origin "https://doh.example.com":

* The DNS Server's Access Service Description, at "/.well-known/access-services" (Section 5 of [I-D.schwartz-masque-access-descriptions]).

  - This conveys the DoH URI template associated with this origin.

* The Oblivious Gateway KeyConfig, at "/.well-known/oblivious-gateway" (Section 5 of [I-D.pauly-ohai-svcb-config]).

  - This conveys the public keys to use for Oblivious HTTP.

To prevent client-targeting attacks, the client retrieves both of these resources via DoubleCheck. The HTTP requests for the Access Service Description appears as follows:

```
HEADERS
:method = GET
:scheme = https
:authority = relay.example.org
:path = /http?target_uri=
    https%3A%2F%2Fdoh.example.com%2F.well-known%2Faccess-services
accept: application/access-services+json
```

```
HEADERS
:status = 200
cache-control: public, immutable, \
    no-transform, s-maxage=86400
age: 80000
etag: ABCD1234
content-type: application/access-services+json
{
  "dns": {
    "template":
        "https://doh.example.com/foo{?dns}"
  }
}
```

```
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 GET request using HTTP/3:

```
HEADERS
:method = GET
:scheme = https
:authority = doh.example.com
:path = /.well-known/access-services
accept: application/access-services+json
if-match = ABCD1234
```

```
HEADERS
:status = 200
cache-control: public, immutable, \
    no-transform, s-maxage=86400
etag: ABCD1234
content-type: application/access-services+json
{
  "dns": {
    "template":
        "https://doh.example.com/foo{?dns}"
  }
}
```

Having successfully fetched the DoH Service 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.

This concludes the DoubleCheck procedure. The Oblivious Gateway KeyConfig is retrieved by a similar procedure. These two DoubleCheck procedures can run in parallel, and MAY share a single transport proxy tunnel for efficiency.

Once the client has double-checked the DoH Service Description and the KeyConfig, it can use the Oblivious DoH service by forming DNS-over-HTTPS requests for "https://doh.example.com/foo{?dns}" as Binary HTTP requests, encrypting them to the KeyConfig, and POSTing the ciphertext to "https://relay.example.org/http?target_uri=https%3A%2F%2Fdoh.example.com%2F.well-known%2Foblivious-gateway".

6. Performance Implications

6.1. Latency

Suppose that the Client-Proxy Round-Trip Time (RTT) is A, and the Proxy-Origin RTT is B. Suppose additionally that the Client has a persistent connection to the Proxy that is already running. Then the procedure described in Section 4.3, with a CONNECT-UDP transport proxy, requires:

* A for the GET request to the Proxy
  - +B if the Desired Resource is not in cache
  - +B if the Proxy does not have a TLS session ticket for the Origin
* A for the CONNECT-UDP request to the Proxy
* A + B for a QUIC handshake to the Origin
* A + B for the GET request to the Origin

This is a total of 4A + 4B in the worst case. However, clients can reduce the latency by issuing the requests to the Proxy in parallel, and by using CONNECT-UDP's "false start" support. The Origin can also optimize performance, by issuing long-lived TLS session tickets. With these optimizations, the expected total time is 2A + 2B.

This procedure only needs to be repeated if the Desired Resource has expired. To enable regular key rotation and operational adjustments, a cache lifetime of 24 hours may be suitable. Clients MAY perform this procedure in advance of an expiration to avoid a delay.

6.2. Thundering Herds

All clients of the same Proxy and Desired Resource will have locally cached copies with the same expiration time. When this copy expires, all active clients will send refresh GET requests to the Proxy at their next request. Proxies SHOULD use "request coalescing" to avoid duplicate cache-refresh requests to the Origin.

If the Desired Resource has changed, these clients will all initiate GET requests to the Origin (via transport proxy if applicable) to double-check the new contents. Proxies and Origins MAY use an HTTP 503 response with a "Retry-After" header to manage load spikes.

7. Security Considerations

7.1. In scope

7.1.1. Authenticity

A malicious Proxy could attempt to forge the Desired Resource by replacing the response body (e.g., returning an Oblivious HTTP KeyConfig containing a Public Key controlled by the Relay). This is prevented by the Client's requirement that the Desired Resource be served to it by the Origin over HTTPS (Section 4.3).

7.1.2. Consistency

A malicious Origin could attempt to break the consistency guarantee by issuing each Client a unique, persistent variant of the Desired Resource. This attack is prevented by the Client's requirement that the resource be fresh according to the Proxy's cache (Section 4.3).

A malicious Origin 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 Desired Resource in the cache before it expires:

  - By sending GET 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), and by the Proxy's obligation to respect these directives strictly (Section 4.2).

  - By filling the cache with new entries, causing the cached copy of the resource to be evicted. Section 4.2 describes some possible mitigations.

7.1.3. Temporal Correlation Attacks

A malicious Origin could attempt to identify or link different requests for the Desired Resource, in order to link proxied requests that follow shortly after. If a transport proxy is used (as recommended), this is prevented by fully isolating each request (Section 4.3), and by disabling EDNS Client Subnet (Section 4.2).

7.1.4. Abusive traffic

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).

7.2. Out of scope

This specification assumes that the Client starts with identities of the Proxy and the Desired Resource that are authentic and widely shared. If these identities are inauthentic, or are unique to the client, 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 Origin. If a large fraction of the Clients are malicious, they could conspire to flood the Proxy's cache with entries that seem popular, leading to rapid eviction of any cached resources. Similar concerns apply if a malicious Origin can compel naive Clients to fetch a very large number of distinct resources through the Proxy.

Even when a transport proxy is used, the Client's requests for the Desired Resource may become linkable if they have distinctive TLS ClientHellos, QUIC Initials, HTTP/3 Settings, RTT, or other protocol features observable through the transport proxy. This specification does not offer specific mitigations for protocol fingerprinting.

8. IANA Considerations

No IANA action is requested.

9. References

9.1. Normative References

[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>.

[RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <https://www.rfc-editor.org/rfc/rfc7871>.

[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>.

[RFC9111] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Caching", STD 98, RFC 9111, DOI 10.17487/RFC9111, June 2022, <https://www.rfc-editor.org/rfc/rfc9111>.

9.2. Informative References

[I-D.ietf-ohai-ohttp]
Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-05, 26 September 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-ohai-ohttp-05>.

[I-D.ietf-privacypass-protocol]
Celi, S., Davidson, A., Faz-Hernández, A., Valdez, S., and C. A. Wood, "Privacy Pass Issuance Protocol", Work in Progress, Internet-Draft, draft-ietf-privacypass-protocol-06, 6 July 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-privacypass-protocol-06>.

[I-D.pauly-ohai-svcb-config]
Pauly, T. and T. Reddy.K, "Discovery of Oblivious Services via Service Binding Records", Work in Progress, Internet-Draft, draft-pauly-ohai-svcb-config-03, 27 July 2022, <https://datatracker.ietf.org/doc/html/draft-pauly-ohai-svcb-config-03>.

[I-D.schwartz-masque-access-descriptions]
Schwartz, B. M., "HTTP Access Service Description Objects", Work in Progress, Internet-Draft, draft-schwartz-masque-access-descriptions-03, 18 October 2022, <https://datatracker.ietf.org/doc/html/draft-schwartz-masque-access-descriptions-03>.

[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-03, 17 August 2022, <https://datatracker.ietf.org/doc/html/draft-wood-key-consistency-03>.

[SVCB] Schwartz, B. M., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPS RRs)", Work in Progress, Internet-Draft, draft-ietf-dnsop-svcb-https-11, 11 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-dnsop-svcb-https-11>.

Acknowledgments

TODO acknowledge.

Author's Address

Benjamin M. Schwartz
Google LLC
Email: bemasc@google.com