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, with an indication of which Oblivious Gateway Resource to use to access the service (as an Oblivious Target Resource). This document also defines a mechanism to learn the key configuration of the related 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 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 https://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."
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 associated 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]. The presence of this parameter indicates that a service can act as an oblivious target, and indicates an oblivious gateway that can provide access to the target.

This document also defines a way to fetch an oblivious gateway’s key configuration by sending a request to the gateway (Section 4).

This mechanism does not aid in the discovery of oblivious relays; relay configuration is out of scope for this document.

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. The oblivious-gateway SvcParamKey

The "oblivious-gateway" SvcParamKey (Section 6) is used to indicate that a service described in an SVCB record can be accessed as an oblivious target using the specified gateway. The service that is queried by the client hosts one or more target resources. The gateway is a separate resource that is indicated by the SVCB record parameter, which allows oblivious access to any target resource hosted by the service described in the SVCB record.

In order to access the service’s target resources obliviously, the client needs to send encapsulated messages to the gateway resource using the gateway’s key configuration (which can be retrieved using the method described in Section 4).
The presentation format of the "oblivious-gateway" parameter is a comma-separated list of one or more gateway URIs. URIs MUST be encoded as escaped items if they include "," or "\", replacing these with "\," and "\\", respectively.

The wire format consists of one or more URIs encoded in UTF-8 [RFC3629], each prefixed by its length as a single octet, with these length-value pairs concatenated to form the SvcParamValue. These pairs MUST exactly fill the SvcParamValue; otherwise, the SvcParamValue is malformed.

The "oblivious-gateway" 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-gateway 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 with an oblivious gateway or directly SHOULD NOT mark the "oblivious-gateway" 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.

3.1. Use in HTTPS service records

For the "https" scheme, which uses the HTTPS RR type instead of SVCB, the presence of the "oblivious-gateway" parameter means that the target being described is an Oblivious HTTP service that uses 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-gateway=https://osvc.example.com/gateway )
```

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

```
svc.example.com. 7200  IN HTTPS 1 . ( mandatory=oblivious-gateway oblivious-gateway=https://osvc.example.com/gateway )
```
3.2. Use in DNS server SVCB records

For the "dns" scheme, as defined in [DNS-SVCB], the presence of the "oblivious-gateway" 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.

3.2.1. 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-gateway=https://odoh.example.net/gateway )
```

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 4, which can either be done directly, or via the relay or another proxy to avoid exposing client IP addresses.

Clients also need to ensure that they are not being targeted with unique key configurations that would reveal their identity. See Section 5 for more discussion.

3.2.2. 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 still need to ensure that they are not being targeted with unique key configurations that would reveal their identity. See Section 5 for more discussion.

4. Key Configuration Fetching

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

In order to fetch the key configuration of an oblivious gateway discovered in this manner, 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 receives the following record:

```
svc.example.com. 7200 IN HTTPS 1. (alpn=h2 oblivious-gateway=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 5 for more discussion of avoiding key targeting attacks.

5. 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 information. Use of encrypted DNS along with DNSSEC can 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 3.2.1 for more discussion, as well as the Security Considerations of [I-D.ietf-add-svcb-dns].

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 4), 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.

6. IANA Considerations

6.1. SVCB Service Parameter

IANA is requested to add the following entry to the SVCB Service Parameters registry ([SVCB]).
<table>
<thead>
<tr>
<th>Number</th>
<th>Name</th>
<th>Meaning</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>oblivious-gateway</td>
<td>Defines an oblivious HTTP gateway to use to access this resource</td>
<td>(This document)</td>
</tr>
</tbody>
</table>

Table 1

7. References

7.1. Normative References

[BINARY-HTTP]  

[DDR]  

[DNR]  

[DNS-SVCB]  

[DOH]  

[OHTTP]  
7.2. Informative References


Authors’ Addresses

Tommy Pauly
Apple Inc.
Email: tpauly@apple.com
Tirumaleswar Reddy
Akamai
Email: kondtir@gmail.com
Abstract

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

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

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 https://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 11 January 2023.
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 via the RateLimit fields defined in [RATELIMIT]. This is useful when such servers identify traffic anomalies or unexpected request volumes. The Oblivious relay 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 (rather than to the client).

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

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 [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. 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 MUST:

1. Identify the quota policy associated to 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.

```
RateLimit-Limit: 100
RateLimit-Policy: 10;w=1, 100;w=60;ohttp-target=1
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. This parameter is OPTIONAL.
```

The "ohttp-target" parameter has the following syntax:

```
ohttp-target = sf-integer
```
Its value MUST be an Integer (Section 3.3.1 of [STRUCTURED-FIELDS]) and indicates whether the quota policy is applicable to all the clients that are serviced by the Oblivious relay or applicable only to a specific client. The "ohttp-target" parameter MUST have one of the following values:

1: Indicates that RateLimit fields are applicable to all the clients that are serviced by the same Oblivious relay.

2: Indicates that RateLimit fields are applicable only to the offending client. For example, this value is used if the client is attacking the server (e.g., the client is using an abnormal header that matches an attack pattern).

The Oblivious relay does not immediately act to rate-limit the traffic from the client but starts maintaining a count of responses to the client with "ohttp-target" parameter value set to "2" marked as "potential malicious requests" and responses without the parameter marked as "legitimate requests".

The Oblivious relay can rate-limit requests from the offending client for a certain duration only when the client has a high ratio of "potential malicious requests" to "legitimate requests". In other words, the Oblivious relay will rate-limit requests from a client if the target has seen an attack pattern in multiple requests from that same client. A malicious client sends malformed HTTP requests, whereas a benign client sends valid HTTP requests. The malformed HTTP requests are linkable whereas the valid HTTP requests are not linkable. Most importantly, the target will not be able to partition the anonymity set of legitimate clients.

Other values MUST cause the parameter to be ignored.

The "ohttp-target" parameter MUST NOT appear more than once in a quota policy. If the parameter is malformed or its value is invalid, it MUST be ignored, and the receiving Oblivious relay 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 receiving RateLimit fields providing Oblivious Relay Feedback will do the following:

1. It MUST remove the RateLimit fields from the response, since they are not intended to be forwarded to clients.
2. It MAY add a new set of RateLimit fields that are intended to be forwarded to a client.

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, so that the Oblivious 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 failing to reshape traffic from a specific client or 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 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 resource to convey the likeliness that an HTTP request is malicious. 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.

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 a malformed request and uses the source IP address to identify that it was an encapsulated request decapsulated by an oblivious gateway resource. The Oblivious target resource generates a 400 response and adds the RateLimit fields along with the "ohttp-target" quota policy parameter. 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.
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".
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=2; attack-severity="high";
comment="abnormal header matching a WAF rule"
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" header is defined in this specification (Section 9.2.1). Its purpose is to signal which HTTP headers will be removed by the Oblivious gateway.

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 the header fields signaled in the associated request and removes those headers from the HTTP response. The Oblivious gateway then encapulates 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 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 using Augmented Backus-Naur Form (ABNF) of the header field, using the grammar defined in [RFC5234] and the rules defined in Section 5 of [RFC9110]. The field values of the header field conform to the same rules.

```
Ohai-Outside-Encap = header-field *( OWS "|" OWS header-field)
header-field = token
```

Figure 4: Ohttp-Outside-Encap Header Syntax

Optional white space (OWS) is used as defined in Section 5.6.3 of [RFC9110].
An example is illustrated below:

```
Ohttp-Outside-Encap: RateLimit-Limit|RateLimit-Remaining|RateLimit-Reset|RateLimit-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. Client and Oblivous Proxy Collusion

While Oblivious HTTP relies upon an Oblivious relay to prevent leaking the client identity to the Oblivious resources, it might be the case that the Oblivious relay 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.

8.2. Attack Categories

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 as an attacker. This might be too many requests from a single client, too many requests from the clients behind the same oblivious relay or too many requests from all clients on the Internet.
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   |
| 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 and Brandon Williams for the discussion and comments.

11. References

11.1. Normative References


11.2. Informative References


Authors’ Addresses

Tirumaleswar Reddy
Akamai
Embassy Golf Link Business Park
Bangalore 560071
Karnataka
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
Abstract

The assurances provided by Oblivious HTTP depend on the client’s ability to verify that it is using the same Gateway, Target, 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

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 https://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 2 January 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Oblivious HTTP [I-D.ietf-ohai-ohttp] identifies four parties to each exchange: the Client, Relay, Gateway and Target. When used properly, Oblivious HTTP enables the Client to send requests to the Target in such a way that the Target and Gateway cannot tell whether two requests came from the same Client, and the Relay cannot see the contents of the requests.
The Target and Gateway are tightly coupled, as the Gateway can see and modify all cleartext data to and from the Target. For ease of description, we will refer to the Target and Gateway collectively (including the URI and KeyConfig for the Gateway and the URI of the Target) as the "Service".

Oblivious HTTP’s threat model assumes that at least one of the Relay and the Service is acting properly, i.e. complying with the protocol and keeping certain information confidential. If either Relay or Service 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 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 Service, regardless of their prior activity. Otherwise, the encrypted request identifies the Client to the Service.

4. (optional) The Gateway must 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.

This specification defines behaviors for the Client, Relay, and Service that achieve preconditions 2-4. (This specification does not address precondition 1.)

This specification assumes that the Service is identified by an Access Description [I-D.schwartz-masque-access-descriptions], which we call the "Service Description". For this specification to meet its goals, the Service Description's URL must have been distributed to clients in a globally consistent fashion. For example, the Service Description URL 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 Service Description URL into a fresh Service Description without losing the privacy guarantees of Oblivious HTTP.
In principle, Services could achieve a similar effect by distributing their Service Descriptions directly through this 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 combines elements of the "Direct Discovery", "Single Proxy Discovery", and "Independent Verification" strategies defined in [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: one to the Relay, and one through the Relay to the Service Description Host using CONNECT-UDP. (The Service Description Host, Gateway, and Target are most commonly expected to be a single origin.) The Relay will forward the first request to the Service Description Host if the response is not in cache.

```
+--------+       +-------+       +-------------+
|        |<=====>|       |<----->|   Service   |
| Client |       | Relay |       | Description |
|        |<=====================>|    Host     |
+--------+       +-------+       +-------------+
```

Figure 1: Overview of Key-Consistency Double-Check

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

4. Requirements

4.1. Oblivious Service

The Oblivious Service MUST publish an Access Description [I-D.schwartz-masque-access-descriptions] containing the "ohttp.gateway" key, e.g.:
This Access Description is called the Service Description, and its origin is called the Service Description Host. This origin MUST support HTTP/3 [RFC9114], so that it can be accessed via the proxy’s CONNECT-UDP service (see Section 4.2).

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

If the Service 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 Relay

The Oblivious Relay MUST also provide CONNECT-UDP service [I-D.ietf-masque-connect-udp], and SHOULD also offer DNS over HTTPS [RFC8484], to enable the use of HTTPS records [SVCB] with CONNECT-UDP. This corresponds to an Access Description that includes the "ohttp.relay", "udp", and "dns" keys:
The Oblivious Relay MUST allow use of the GET method to retrieve small JSON responses, and SHOULD make ample cache space available in order to avoid eviction of Service Descriptions. The Relay SHOULD share cache state among all clients, to ensure that they use the same Service Descriptions for each Oblivious Service. 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.

Oblivious Relays MUST preserve the ETag response header on cached responses, and MUST add an Age header ([RFC9111], Section 5.1) to all proxied responses. Oblivious Relays 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]). Oblivious Relays also MUST NOT accept PUSH_PROMISE frames from the target.

Relays 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 Relays that are not intended for general-purpose proxy usage MAY impose strict transfer limits or rate limits on HTTP CONNECT and CONNECT-UDP usage.
If the Relay offers a DNS over HTTPS resolver, it MUST NOT enable EDNS Client Subnet support [RFC7871].

4.3. Client

The Client is assumed to know an "https" URI of the relevant Service Description. Before using that Service Description, it MUST perform the following "double-check" procedure:

1. Send a GET request to the Oblivious Relay’s template (ohttp.proxy.template) with request_uri set to the Service Description URI.

2. Record the response (A).

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

4. Establish a CONNECT-UDP tunnel through the proxy to the Service Description URI’s origin.

5. Fetch the Service Description URI through this tunnel, 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 Service 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 Service Description, 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.

5. Example: Oblivious DoH

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

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

The following exchanges then occur between the client and the proxy:
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 = /config.json
if-match = ABCD1234

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

Having successfully fetched the 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.

The client can now use the KeyConfig in this Service Description to reach the Oblivious DoH server, by converting DNS-over-HTTPS requests into Binary HTTP requests for "https://doh.example.com/dns-query", encrypting them to ohttp.gateway.key, and POSTing the encrypted requests to "https://relay.example.org/ohttp?request_uri=https%3A%2F%2Fexample.com%2Fohttp%2F".

6. Performance Implications

6.1. Latency

Suppose that the Client-Relay Round-Trip Time (RTT) is A, and the Relay-Service Description Host RTT is B. Suppose additionally that the Client has a persistent connection to the Relay that is already running. Then the procedure described in Section 4.3 requires:

* A for the GET request to the Relay
  - +B if the requested Service Description is not in cache
  - +B if the Relay does not have a TLS session ticket for the Service Description Host
* A for the CONNECT-UDP request to the Relay
* A + B for the QUIC handshake to the Service Description Host
* A + B for the GET request to the Service Description Host

This is a total of 4A + 4B in the worst case. However, clients can reduce the latency by issuing the requests to the Relay in parallel, and by using CONNECT-UDP’s "false start" support. The Service Description Host 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 Service Description 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 Relay and Service will have locally cached Service Descriptions with the same expiration time. When this entry expires, all active clients will send refresh GET requests to the proxy at their next request. Relays SHOULD use "request coalescing" to avoid duplicate cache-refresh requests to the target.

If the Service Description has changed, these clients will initiate GET requests through the Relay to the Service Description Host to double-check the new contents. Relays and Service Description Hosts 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. Forgery

A malicious Relay could attempt to learn the contents of the Oblivious HTTP request by forging a Service Description containing its own KeyConfig. This is prevented by the client’s requirement that the KeyConfig be served to it by the Service Description Host over HTTPS (Section 4.3).

7.1.2. Deanonymization

A malicious Service could attempt to link multiple Oblivious HTTP requests together by issuing each Client a unique, persistent KeyConfig. This attack is prevented by the Client’s requirement that the KeyConfig be fresh according to the Relay’s cache (Section 4.3).

A malicious Service could attempt to rotate its entry in the Relay’s cache in several ways:

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

* By also acting as a Client and sending requests designed to replace the Service Description 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 Relay’s obligation to to respect these directives strictly (Section 4.2).
- By filling the cache with new entries, causing its previous Service Description to be evicted. Section 4.2 describes some possible mitigations.

A malicious Service could attempt to link different requests for the Service Description, in order to link the Oblivious HTTP requests that follow shortly after. This is prevented by fully isolating each request (Section 4.3), and by disabling EDNS Client Subnet (Section 4.2).

7.1.3. 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 Relay and Service 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 Service. If a large fraction of the Clients are malicious, they could conspire to flood the Relay’s cache with entries that seem popular, leading to rapid eviction of the malicious Service’s Service Descriptions. Similar concerns apply if a malicious Service can compel naive Clients to fetch a very large number of Service Descriptions.

A Client’s requests for the Service Description may become linkable if they have distinctive QUIC Initials, HTTP/3 Settings, RTT, or other protocol features observable through the Relay. This specification does not offer specific mitigations for protocol fingerprinting.

8. IANA Considerations

No IANA action is requested.

9. References

9.1. Normative References
[I-D.ietf-masque-connect-udp]

[I-D.ietf-ohai-ohttp]

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


9.2.  Informative References


Acknowledgments

TODO acknowledge.

Author’s Address

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