Network Working Group E. Kinnear
Internet-Draft Apple Inc.
Intended status: Standards Track P. McManus
Expires: May 4, 2020 Fastly
T. Pauly
C. Wood
Apple Inc.
November 01, 2019

Oblivious DNS Over HTTPS
draft-pauly-dprive-oblivious-doh-01

Abstract

This document describes an extension to DNS Over HTTPS (DoH) that allows hiding client IP addresses via proxying encrypted DNS transactions. This improves privacy of DNS operations by not allowing any one server entity to be aware of both the client IP address and the content of DNS queries and answers.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

DNS Over HTTPS (DoH) [RFC8484] defines a mechanism to allow DNS messages to be transmitted in encrypted HTTP messages. This provides improved confidentiality and authentication for DNS interactions in various circumstances.

While DoH can prevent eavesdroppers from directly reading the contents of DNS exchanges, it does not allow clients to send DNS queries and receive answers from servers without revealing their local IP address, and thus information about the identity or location of the client.

Proposals such as Oblivious DNS ([I-D.annee-dprive-oblivious-dns]) allow increased privacy by not allowing any single DNS server to be aware of both the client IP address and the message contents.

This document defines Oblivious DoH, an extension to DoH that allows for a proxied mode of resolution, in which DNS messages are encrypted in such a way that no DoH server can independently read both the client IP address and the DNS message contents.

This mechanism is intended to be used as one option for resolving privacy-sensitive content in the broader context of Adaptive DNS [I-D.pauly-dprive-adaptive-dns-privacy].

1.1. Specification of Requirements

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.

2. Terminology

This document defines the following terms:

Oblivious Proxy: A server that proxies encrypted client DNS queries to a resolution server that will be able to decrypt the query (the Oblivious Target). Oblivious DoH servers can function as proxies, but other non-resolver proxy servers could also be used.

Oblivious Target: A resolution server that receives encrypted client DNS queries and generates encrypted DNS responses transferred via an Oblivious Proxy.

3. Deployment Requirements

Oblivious DoH requires, at a minimum:

- Two DoH servers, where one can act as an Oblivious Proxy, and the other can act as an Oblivious Target.

- Public keys for encrypting DNS queries that are passed from a client through a proxy to a target (Section 6). These keys guarantee that only the intended Oblivious Target can decrypt client queries.

- Client ability to generate random [RFC4086] one-time-use symmetric keys to encrypt DNS responses. These symmetric keys ensure that only the client will be able to decrypt the response from the Oblivious Target. They are only used once to prevent the Oblivious Target from tracking clients based on keys.

The mechanism for discovering and provisioning the DoH URI Templates and public keys is via parameters added to DNS resource records. The mechanism for discovering the public key is decribed in Section 5. The mechanism for discovering a DoH URI Template is described in [I-D.pauly-dprive-adaptive-dns-privacy].

4. HTTP Exchange

Unlike direct resolution, oblivious hostname resolution over DoH involves three parties:

1. The Client, which generates queries.

2. The Oblivious Proxy, which is a resolution server that receives encrypted queries from the client and passes them on to another resolution server.

3. The Oblivious Target, which is a resolution server that receives proxied queries from the client via the Oblivious Proxy.

```
     --- [ Request encrypted with target public key ] -->
+---------+             +-----------+             +-----------+
| Client  +-------------> Oblivious +-------------> Oblivious |
|         <-------------+   Proxy   <-------------+  Target   |
+---------+             +-----------+             +-----------+
   <-- [ Response encrypted with client symmetric key ] ---
```

Figure 1: Obvlivious DoH Exchange

4.1. HTTP Request

Oblivious DoH queries are created by the Client, and sent to the Oblivious Proxy. Requests to the Oblivious Proxy indicate which DoH server to use as an Oblivious Target by specifying two variables: "targethost", which indicates the host name of the Oblivious Target server, and "targetpath", which indicates the path on which the Oblivious Target's DoH server is running. See Section 4.2 for an example request.

Oblivious DoH messages have no cache value since both requests and responses are encrypted using ephemeral key material. Clients SHOULD prefer using HTTP methods and headers that will prevent unhelpful cache storage of these exchanges (i.e., preferring POST instead of GET).

Clients MUST set the HTTP Content-Type header to "application/ oblivious-dns-message" to indicate that this request is an Oblivious

DoH query intended for proxying. Clients also SHOULD set this same value for the HTTP Accept header.

Upon receiving a request that contains a "application/oblivious-dns-message" Content-Type, the DoH server looks for the "targethost" and "targetpath" variables. If the variables are not present, then it is the target of the query, and it can decrypt the query (Section 7). If the variables are present, then the DoH server is acting as an Oblivious Proxy. If it is a proxy, it is expected to send the request on to the Oblivious Target using the URI template constructed as "https://targethost/targetpath".

4.2. HTTP Request Example

The following example shows how a client requests that an Oblivious Proxy, "dnsproxy.example.net", forwards an encrypted message to "dnstarget.example.net". The URI template for the Oblivious Proxy is "https://dnsproxy.example.net/dns-query{?targethost,targetpath}". The URI template for the Oblivious Target is "https://dnstarget.example.net/dns-query".

```
:method = POST
:scheme = https
:authority = dnsproxy.example.net
:path = /dns-query?targethost=dnstarget.example.net&targetpath=/dns-query
accept = application/oblivious-dns-message
cache-control = no-cache, no-store
content-type = application/oblivious-dns-message
content-length = 106

<Bytes containing the encrypted payload for an Oblivious DNS query>
```

The Oblivious Proxy then sends the following request on to the Oblivious Target:

```
:method = POST
:scheme = https
:authority = dnstarget.example.net
:path = /dns-query
accept = application/oblivious-dns-message
cache-control = no-cache, no-store
content-type = application/oblivious-dns-message
content-length = 106

<Bytes containing the encrypted payload for an Oblivious DNS query>
```

4.3. HTTP Response

The response to an Oblivious DoH query is generated by the Oblivious Target. It MUST set the Content-Type HTTP header to "application/oblivious-dns-message" for all successful responses. The body of the response contains a DNS message that is encrypted with the client's symmetric key (Section 7).

All other aspects of the HTTP response and error handling are inherited from standard DoH.

4.4. HTTP Response Example

The following example shows a response that can be sent from an Oblivious Target to a client via an Oblivious Proxy.

```
:status = 200
content-type = application/oblivious-dns-message
content-length = 154

<Bytes containing the encrypted payload for an Oblivious DNS response>
```

5. Public Key Discovery

In order to use a DoH server as an Oblivious Target, the client must know a public key to use for encrypting its queries. This key can be discovered using the SVCB or HTTPSSVC record type ([I-D.nygren-dnsop-svcb-httpssvc]) for a name owned by the server.

The Service Binding key name is "odohkey" (Section 9). If present, this key/value pair contains the public key to use when encrypting Oblivious DoH messages that will be targeted at a DoH server. The format of the key is defined in (Section 6).

Clients MUST only use keys that were retrieved from records protected by DNSSEC [RFC4033] to encrypt messages to an Oblivious Target.

6. Oblivious DoH Public Key Format

An Oblivious DNS public key is a structure encoded, using TLS-style encoding [RFC8446], as follows:

```
struct {
   uint16 kem_id;
   uint16 kdf_id;
   uint16 aead_id;
   opaque public_key<1..2^16-1>;
} ObliviousDNSKey;
```

It contains the information needed to encrypt a message under ObliviousDNSKey.public_key such that only the owner of the corresponding private key can decrypt the message. The values for ObliviousDNSKey.kem_id, ObliviousDNSKey.kdf_id, and ObliviousDNSKey.aead_id are described in [I-D.irtf-cfrg-hpke] Section 7. For convenience, let Identifier(ObliviousDNSKey) be defined as the SHA256 value of ObliviousDNSKey serialized.

7. Oblivious DoH Message Format

There are two types of Oblivious DoH messages: Queries (0x01) and Responses (0x02). Both are encoded as follows:

```
struct {
   uint8  message_type;
   opaque key_id<0..2^16-1>;
   opaque encrypted_message<1..2^16-1>;
} ObliviousDNSMessage;
```

ObliviousDNSMessage.message_type = 0x01 for Query messages and ObliviousDNSMessage.message_type = 0x02 for Response messages. ObliviousDNSMessage.key_id contains the identifier of the corresponding ObliviousDNSKey key. ObliviousDNSMessage.encrypted_message contains an encrypted message for the Oblivious Target (for Query messages) or client (for Response messages). The following sections describe how these meessage bodies are constructed.

7.1. Oblivious Queries

Oblivious DoH Query messages must carry the following information:

1. A symmetric key under which the DNS response will be encrypted. The AEAD algorithm used for the client's response key is the one associated with the server's public key.

2. A DNS query message which the client wishes to resolve.

3. Padding of arbitrary length which MUST contain all zeros.

The key and message are encoded using the following structure:

```
struct {
   opaque dns_message<1..2^16-1>;
   opaque response_seed[32];
   opaque padding<0..2^16-1>;
} ObliviousDNSQueryBody;
```

Let M be a DNS message a client wishes to protect with Oblivious DoH. When sending an Oblivious DoH Query for resolving M to an Oblivious Target with ObliviousDNSKey key pk, a client does the following:

1. Generate a random response seed of length 32 octets according to the guidelines in [RFC4086].

2. Create an ObliviousDNSQueryBody structure, carrying the message M, response_seed, and padding, to produce Q_plain.

3. Unmarshal pk.public_key to produce a public key pkR of type pk.kem_id.

4. Compute the encrypted message as Q_encrypted = encrypt_query_body(pkR, key_id, Q_plain). key_id is defined as Identifier(pk).

5. Output a ObliviousDNSMessage message Q where Q.message_type = 0x01, Q.key_id carries Identifier(pk), and Q.encrypted_message = Q_encrypted.

The client then sends Q to the Oblivious Proxy according to Section 4.1.

```
def encrypt_query_body(pkR, key_id, Q_plain):
  enc, context = SetupBaseI(pkR, "odns-query")
  aad = 0x01 || key_id
  ct = context.Seal(aad, Q_plain)
  Q_encrypted = enc || ct
  return Q_encrypted
```

7.2. Oblivious Responses

An Oblivious DoH Response message carries the DNS response (dns_message) along with padding. This message is encrypted with the client's chosen response key.

```
struct {
   opaque dns_message<1..2^16-1>;
   opaque padding<0..2^16-1>;
} ObliviousDNSResponseBody;
```

Targets that receive a Query message Q decrypt and process it as follows:

1. Look up the ObliviousDNSKey according to Q.key_id. If no such key exists, the Target MAY discard the query. Otherwise, let skR be the private key corresponding to this public key, or one

chosen for trial decryption, and pk be the corresponding ObliviousDNSKey.

2. Compute Q_plain, error = decrypt_query_body(skR, Q.key_id, Q.encrypted_message).

3. If no error was returned, and Q_plain.padding is valid (all zeros), resolve Q_plain.dns_message as needed, yielding a DNS message M.

4. Create an ObliviousDNSResponseBody structure, carrying the message M and padding, to produce R_plain.

5. Compute akey, anonce = derive_keys(Q_plain). (See definition for derive_keys below. Hash, Expand, Extract, Nn, and Nk are functions and parameters bound to the target's HPKE public key.)

6. Compute R_encrypted = encrypt_response_body(R_plain, akey, anonce). (See definition for encrypt_response_body below. The key_id field used for encryption is empty, yielding 0x0000 as part of the AAD.)

7. Output a ObliviousDNSMessage message R where R.message_type = 0x02, R.key_id = nil, and R.encrypted_message = R_encrypted.

```
def derive_keys(Q_plain):
  context = Hash(Q_plain.dns_message)
  key = Expand(Q_plain.response_seed, concat("odoh key", context), Nk)
  nonce = Expand(Q_plain.response_seed, concat("odoh nonce", context), Nn)
  return key, nonce

   def decrypt_query_body(skR, key_id, Q_encrypted):
     enc || ct = Q_encrypted
     dec, context = SetupBaseR(skR, "odns-query")
     aad = 0x01 || key_id
     Q_plain, error = context.Open(aad, ct)
     return Q_plain, error

   def encrypt_response_body(R_plain, akey, anonce):
     aad = 0x02 || 0x0000 // 0x0000 represents a 0-length KeyId
     R_encrypted = Seal(akey, anonce, aad, R_plain)
     return R_encrypted
```

The Target then sends R to the Proxy according to Section 4.3.

The Proxy forwards the message R without modification back to the client as the HTTP response to the client's original HTTP request.

Once the client receives the response, it can use its known response_seed to derive the decryption key and nonce, decrypt R.encrypted_message using decrypt_response_body (defined below), and produce R_plain. Clients MUST validate R_plain.padding (as all zeros) before using R_plain.dns_message.

```
def decrypt_response_body(R_encrypted):
  aad = 0x02 || 0x0000
  R_plain = Open(response_key, 0^Nn, aad, R_encrypted)
  return R_plain
```

8. Security Considerations

DISCLAIMER: this is a work in progress draft and has not yet seen significant security analysis.

Oblivious DoH aims to keep knowledge of the true query origin and its contents known to only clients. In particular, it assumes a Dolev-Yao style attacker which can observe all client queries, including those forwarded by oblivious proxies, and does not collude with target resolvers. (Indeed, if a target colludes with the network attacker, then said attacker can learn the true query origin and its contents.) Oblivious DoH aims to achieve the following confidentiality goals in the presence of this attacker:

1. Queries and answers are known only to clients and targets in possession of the corresponding response key and HPKE keying material. In particular, proxies know the origin and destination of an oblivious query, yet do not know the plaintext query. Likewise, targets know only the oblivious query origin, i.e., the proxy, and the plaintext query. Only the client knows both the plaintext query contents and destination.

2. Target resolvers cannot link queries from the same client in the absence of unique per-client keys.

Traffic analysis mitigations are outside the scope of this document. In particular, this document does not recommend padding lengths for ObliviousDNSQueryBody and ObliviousDNSResponseBody messages. Implementations SHOULD follow the guidance for choosing padding length in [RFC8467].

Oblivious DoH security does not depend on proxy and target indistinguishability. Specifically, an on-path attacker could determine whether a connection a specific endpoint is used for oblivious or direct DoH queries. However, this has no effect on confidentiality goals listed above.

8.1. Denial of Service

Malicious clients (or proxies) may send bogus Oblivious DoH queries to targets as a Denial-of-Service (DoS) attack. Target servers may throttle processing requests if such an event occurs.

Malicious targets or proxies may send bogus answers in response to Oblivious DoH queries. Response decryption failure is a signal that either the proxy or target is misbehaving. Clients can choose to stop using one or both of these servers in the event of such failure.

9. IANA Considerations

9.1. Oblivious DoH Message Media Type

This document registers a new media type, "application/oblivious-dns-message".

Type name: application

Subtype name: oblivious-dns-message

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: This is a binary format, containing encrypted DNS requests and responses, as defined in this document.

Security considerations: See this document. The content is an encrypted DNS message, and not executable code.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof.

Published specification: This document.

Applications that use this media type: This media type is intended to be used by clients wishing to hide their DNS queries when using DNS over HTTPS.

Additional information: None

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: None

Author: IETF

Change controller: IETF

9.2. Oblivious DoH Public Key DNS Parameter

This document defines one new key to be added to the Service Binding (SVCB) Parameter Registry [I-D.nygren-dnsop-svcb-httpssvc].

Name: odohkey

SvcParamKey: TBD

Meaning: Public key used to encrypt messages in Oblivious DoH

Reference: This document.

10. Acknowledgments

This work is inspired by Oblivious DNS [I-D.annee-dprive-oblivious-dns]. Thanks to all of the authors of that document. Thanks to Frederic Jacobs, Elliot Briggs, Paul Schmitt, Brian Swander, and Tommy Jensen for the feedback and input.

11. References

11.1. Normative References

[I-D.irtf-cfrg-hpke]
Barnes, R. and K. Bhargavan, "Hybrid Public Key Encryption", draft-irtf-cfrg-hpke-00 (work in progress), July 2019.

[I-D.nygren-dnsop-svcb-httpssvc]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)", draft-nygren-dnsop-svcb-httpssvc-00 (work in progress), September 2019.

[I-D.pauly-dprive-adaptive-dns-privacy]
Kinnear, E., Pauly, T., Wood, C., and P. McManus, "Adaptive DNS: Improving Privacy of Name Resolution", draft-pauly-dprive-adaptive-dns-privacy-00 (work in progress), October 2019.

[RFC4033] 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/info/rfc4033>.

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <https://www.rfc-editor.org/info/rfc4086>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <https://www.rfc-editor.org/info/rfc8467>.

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

11.2. Informative References

[I-D.annee-dprive-oblivious-dns] Edmundson, A., Schmitt, P., Feamster, N., and A. Mankin, "Oblivious DNS - Strong Privacy for DNS Queries", draft-annee-dprive-oblivious-dns-00 (work in progress), July 2018.

[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>.

[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>.

Authors' Addresses

Eric Kinnear
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: ekinnear@apple.com

Patrick McManus
Fastly

Email: mcmanus@ducksong.com

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: tpauly@apple.com

Chris Wood
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: cawood@apple.com