httpbis M.
Thomson
Internet-Draft
Mozilla
Intended status: Standards Track M.
Nottingham
Expires: February 10, 2018
true
W.
Tarreau
HAProxy
Technologies
August 09,
2017

Using Early Data in HTTP
draft-thomson-http-replay-01

Abstract

This document explains the risks of using early data for HTTP and describes techniques for reducing them. In particular, it defines a mechanism that enables clients to communicate with servers about early data, to assure correct operation.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on February 10, 2018.

include Simplified BSD License text as described in Section 4.e of

Table of Contents

1. Introduction

TLS 1.3 [TLS13] introduces the concept of early data (also known as zero round trip data or 0-RTT data). Early data allows a client to send data to a server in the first round trip of a connection, without waiting for the TLS handshake to complete if the client has spoken to the same server recently.

When used with HTTP [HTTP], early data allows clients to send requests immediately, avoiding the one or two round trip delay

needed
   for the TLS handshake. This is a significant performance
   enhancement; however, it has significant limitations.

   The primary risk of using early data is that an attacker might
   capture and replay the request(s) it contains. TLS [TLS13]
describes
   techniques that can be used to reduce the likelihood that an
attacker
   can successfully replay a request, but these techniques can be
   difficult to deploy, and still leave some possibility of a
successful
   attack.

   Note that this is different from automated or user-initiated
retries;
   replays are initiated by an attacker without the awareness of the
   client.

To help mitigate the risk of replays in HTTP, this document gives an overview of techniques for controlling these risks in servers, and defines requirements for clients when sending requests in early data.

The advice in this document also applies to use of 0-RTT in HTTP over QUIC [HQ].

1.1. Conventions and Definitions

The words "MUST", "MUST NOT", "SHOULD", and "MAY" are used in this document. It's not shouting; when they are capitalized, they have the special meaning defined in [RFC2119].

2. Early Data in HTTP

Conceptually, early data is concatenated with other application to form a single stream. This can mean that requests are entirely contained within early data, or only part of a request is early. In a multiplexed protocol, like HTTP/2 [RFC7540] or HTTP/QUIC [HQ], multiple requests might be partially delivered in early data.

The model that this document assumes is that once the TLS handshake completes, early data is not replayed. However, it is important to note that this does not mean that early data will not be or has not been replayed on another connection.

3. Supporting Early Data in HTTP Servers

A server decides whether or not to offer a client the ability to send early data on future connections when sending the TLS session ticket.

When a server enables early data, there are a number of techniques it can use to mitigate the risks of replay:

1. The server can choose whether it will process early data before the TLS handshake completes. By deferring processing, it can ensure that only a successfully completed connection is used for the request(s) therein. Assuming that a replayed ClientHello will not result in additional connections being made by the client, this provides the server with some assurance that the early data was not replayed.

2. If the server receives multiple requests in early data, it can determine whether to defer HTTP processing on a per-request basis. This may require buffering the responses to preserve ordering in HTTP/1.1.

3. The server can cause a client to retry a request and not use early data by responding with the 4NN (Too Early) status code (Section 5.2), in cases where the risk of replay is judged too great.

4. Finally, TLS [TLS13] describes several mitigation strategies that reduce the ability of an attacker to successfully replay early data. Servers are strongly encouraged to implement these techniques, but to also recognize that they are imperfect. These anti-replay techniques can reduce the number of replays that will be successful from being essentially unbounded to a fixed value.

For a given request, the level of tolerance to replay risk is specific to the resource it operates upon (and therefore only known to the origin server). In general, if processing a request does not have state-changing side effects, the consequences of replay are not significant.

The request method's safety ([RFC7231], Section 4.2.1) is one way to determine this. However, some resources do elect to associate side effects with safe methods, so this cannot be universally relied upon.

It is RECOMMENDED that origin servers allow resources to explicitly configure whether early data is appropriate in requests. Absent such explicit information, they SHOULD mitigate against early data in requests that have unsafe methods, using the techniques outlined above.

A request might be sent partially in early data with the remainder of the request being sent after the handshake completes. This does not necessarily affect handling of that request; what matters is when the server starts acting upon the contents of a request. Any time a server might initiate processing prior to completion of the handshake needs to consider how a possible replay of early data could affect that processing (see also Section 6.2).

A server can partially process requests that are incomplete. Parsing header fields - without acting on the values - and determining request routing is likely to be safe from side-effects, but other actions might not be.

Intermediary servers do not have sufficient information to make this determination, so Section 5.2 describes a way for the origin to

signal to them that a particular request isn't appropriate for early data. Intermediaries that accept early data MUST implement that mechanism.

Note that a server cannot choose to selectively reject early data at the TLS layer. TLS only permits a server to accept all early data, or none of it. Once a server has decided to accept early data, it MUST process all requests in early data, even if the server rejects the request by sending a 4NN (Too Early) response.

A server can limit the amount of early data with the "max_early_data_size" field of the "early_data" TLS extension. This can be used to avoid committing an arbitrary amount of memory for deferred requests. A server SHOULD ensure that when it accepts early data, it can defer processing of requests until after the TLS handshake completes.

4. Using Early Data in HTTP Clients

A client that wishes to use early data commences sending HTTP requests immediately after sending the TLS ClientHello.

By their nature, clients have control over whether a given request is sent in early data - thereby giving the client control over risk of replay. Absent other information, clients MAY send requests with safe HTTP methods (see [RFC7231], Section 4.2.1) in early data when it is available, and SHOULD NOT send unsafe methods (or methods whose safety is not known) in early data.

If the server rejects early data at the TLS layer, a client MUST start sending again as though the connection was new. For HTTP/2, this means re-sending the connection preface. Any requests sent in early data MUST be sent again, unless the client decides to abandon those requests.

This automatic retry exposes the request to a potential replay attack. An attacker sends early data to one server instance that accepts and processes the early data, but allows that connection to proceed no further. The attacker then forwards the same messages from the client to another server instance that will reject early data. The client the retries the request, resulting in the request being processed twice. Replays are also possible if there are multiple server instances that will accept early data, or if the same server accepts early data multiple times (though this would be in violation of requirements in TLS).

Clients that use early data MUST retry requests upon receipt of a 4NN (Too Early) status code; see Section 5.2.

An intermediary MUST NOT use early data when forwarding a request unless early data was used on a previous hop, or it knows that the

request can be retried safely without consequences (typically, using

out-of-band configuration). Absent better information, that means that an intermediary can only use early data if the request either arrived in early data or arrived with the "Early-Data" header field set to "1".

5. Extensions for Early Data in HTTP

Because HTTP requests can span multiple "hops", it is necessary to explicitly communicate whether a request has been sent in early data on a previous connection. Likewise, some means of explicitly triggering a retry when early data is not desirable is necessary. Finally, it is necessary to know whether the client will actually perform such a retry.

To meet these needs, two signalling mechanisms are defined:

- The "Early-Data" header field is included in requests that are received in early data.

- The 4NN (Too Early) status code is defined for an server to indicate that a request could not be processed due to the consequences of a possible replay attack.

They are designed to enable better coordination of the use of early data between the user agent and origin server, and also when a gateway (also "reverse proxy", "Content Delivery Network", or "surrogate") is present.

Gateways typically don't have specific information about whether a given request can be processed safely when it is sent in early data. In many cases, only the origin server has the necessary information to decide whether the risk of replay is acceptable. These extensions allow coordination between a gateway and its origin server.

5.1. The Early-Data Header Field

The "Early-Data" request header field indicates that the request has been conveyed in early data, and additionally indicates that a client understands the 4NN (Too Early) status code.

It has just one valid value: "1". Its syntax is defined by the following ABNF [ABNF]:

```
Early-Data = "1"
```

For example:

```
GET /resource HTTP/1.0
Host: example.com
Early-Data: 1
```

An intermediary that forwards a request received in TLS early data MUST send it with the "Early-Data" header field set to "1" (i.e., it adds it if not present in the request).

An intermediary MUST NOT remove this header field if it is present in a request.

The "Early-Data" header field is not intended for use by user agents (that is, the original initiator of a request). Sending a request in early data implies that the client understands this specification and is willing to retry a request in response to a 4NN (Too Early) status code. A user agent that sends a request in early data does not need to include the "Early-Data" header field.

A server cannot make a request that contains the Early-Data header field safe for processing by waiting for the handshake to complete. A request that is marked with Early-Data was sent in early data on a previous hop. Requests that contain the Early-Data field and cannot be safely processed MUST be rejected using the 4NN (Too Early) status code.

5.2. The 4NN (Too Early) Status Code

A 4NN (Too Early) status code indicates that the server is unwilling to risk processing a request that might be replayed.

Clients (user-agents and intermediaries) that sent the request in early data MUST automatically retry the request when receiving a 4NN (Too Early) response status code. Such retries MUST NOT be sent in early data, and SHOULD NOT be sent if the TLS handshake on the original connection does not successfully complete.

Intermediaries that receive a 4NN (Too Early) status code MAY automatically retry requests after allowing the handshake to complete unless the original request contained the "Early-Data" header field when it was received. Otherwise, an intermediary MUST forward the 4NN (Too Early) status code.

The server cannot assume that a client is able to retry a request unless the request is received in early data or the "Early-Data" header field is set to "1". A server SHOULD NOT emit the 4NN status code unless one of these conditions is met.

The 4NN (Too Early) status code is not cacheable by default. Its payload is not the representation of any identified resource.

6. Security Considerations

Using early data exposes a client to the risk that their request is replayed. A retried or replayed request can produce different side effects on the server. In addition to those side effects, replays and retries might be used for traffic analysis to recover information about requests or the resources those requests target.

6.1. Gateways and Early Data

A gateway that forwards requests that were received in early data MUST only do so if it knows that the server that receives those requests understands the "Early-Data" header field and will correctly generate a 4NN (Too Early) status code. A gateway that isn't certain about server support SHOULD either delay forwarding the request until the TLS handshake completes, or send a 4NN (Too Early) status code in response. A gateway that is uncertain about whether an origin server supports the "Early-Data" header field SHOULD disable early data.

6.2. Consistent Handling of Early Data

Consistent treatment of a request that arrives in - or partially in - early data is critical to avoiding inappropriate processing of replayed requests. If a request is not safe to process before the TLS handshake completes, then all instances of the server need to agree and either reject the request or delay processing.

6.3. Denial of Service

Accepting early data exposes a server to potential denial of service through the replay of requests that are expensive to handle. A server that is under load SHOULD prefer rejecting TLS early data as a whole rather than accepting early data and selectively processing requests. Generating a 503 (Service Unavailable) or 4NN (Too Early) status code often leads to clients retrying requests, which could result in increased load.

7. IANA Considerations

This document registers the "Early-Data" header field in the "Message

Headers" registry [HEADERS].

Header field name: Early-Data

Applicable protocol: http

Status: standard

Author/Change controller: IETF

Specification document(s): This document

Related information: (empty)

This document registers the 4NN (Too Early) status code in the "Hypertext Transfer Protocol (HTTP) Status Code" registry established in [RFC7231].

Value: 4NN

Description: Too Early

Reference: This document

8. References

8.1. Normative References

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

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

[HTTP] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <http://www.rfc-editor.org/info/rfc7230>.

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

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[TLS13] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", draft-ietf-tls-tls13-21 (work in progress), July 2017.

8.2. Informative References

[HQ] Bishop, M., "Hypertext Transfer Protocol (HTTP) over QUIC", draft-ietf-quic-http-04 (work in progress), June 2017.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <http://www.rfc-editor.org/info/rfc7540>.

Appendix A. Acknowledgments

This document was not easy to produce. The following people made substantial contributions to the quality and completeness of the document: Subodh Iyengar, Benjamin Kaduk, Ilari Liusavaara, Kazuho Oku, and Victor Vasiliev.

Authors' Addresses

Martin Thomson
Mozilla

Email: martin.thomson@gmail.com

Mark Nottingham
true

Email: mnot@mnot.net

Willy Tarreau
HAProxy Technologies

Email: willy@haproxy.org