Internet Engineering Task Force (IETF)
Request for Comments: 8470
Category: Standards Track
ISSN: 2070-1721

M. Thomson
Mozilla
M. Nottingham
Fastly
W. Tarreau
HAProxy Technologies
September 2018

Using Early Data in HTTP

Abstract

Using TLS early data creates an exposure to the possibility of a replay attack. This document defines mechanisms that allow clients to communicate with servers about HTTP requests that are sent in early data. Techniques are described that use these mechanisms to mitigate the risk of replay.

Status of this Memo

Copyright Notice

[1] https://www.rfc-editor.org/rfc/rfc7841.html#section-2
[2] https://www.rfc-editor.org/info/rfc8470
[3] https://trustee.ietf.org/license-info

Table of Contents

1. Introduction

TLS 1.3 [TLS13] introduces the concept of early data (also known as zero round-trip time (0-RTT) data). If the client has spoken to the same server recently, 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.

When used with HTTP [HTTP], early data allows clients to send requests immediately, thus avoiding the one or two round-trip delays 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 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. Early Data in HTTP

Conceptually, early data is concatenated with other application data to form a single stream. This can mean that requests are entirely contained within early data or that 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, the early data received on that TLS connection is known to not be a replayed copy of that data. 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.

TLS [TLS13] mandates the use of replay detection strategies that reduce the ability of an attacker to successfully replay early data. These anti-replay techniques reduce but don't completely eliminate the chance of data being replayed and ensure a fixed upper limit to the number of replays.

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 reject early data at the TLS layer. A server cannot selectively reject early data, so this results in all requests sent in early data being discarded.
2. The server can choose to delay processing of early data until after the TLS handshake completes. By deferring processing, it can ensure that only a successfully completed connection is used for the request(s) therein. This provides the server with some assurance that the early data was not replayed. If the server receives multiple requests in early data, it can determine whether to defer HTTP processing on a per-request basis.
3. The server can cause a client to retry individual requests and not use early data by responding with the 425 (Too Early) status code (Section 5.2) in cases where the risk of replay is judged too great.

All of these techniques are equally effective; a server can use the method that best suits it.

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). The primary risk associated with using early data is in the actions a server takes when processing a request; processing a duplicated request might result in duplicated effects and side effects. Appendix E.5 of [TLS13] also describes other effects produced by processing duplicated requests.

The request method's safety ([RFC7231], Section 4.2.1) is one way to determine this. However, some resources produce 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, origin servers MUST either reject early data or implement the techniques described in this document for ensuring that requests are not processed prior to TLS handshake completion.

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 any server instance might initiate processing prior to completion of the handshake, all server instances need to account for the possibility of replay of early data and how that 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 decide whether early data can be processed, 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 either 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 425 (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 requests that it might defer until the handshake completes.

4. Using Early Data in HTTP Clients

A client that wishes to use early data commences by 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 ([RFC7231], Section 4.2.1) in early data when it is available and MUST 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 were new. This could entail using a different negotiated protocol [ALPN] than the one optimistically used for the early data. Any requests sent in early data will need to be sent again, unless the client decides to abandon those requests.

Automatic retry creates the potential for a replay attack. An attacker intercepts a connection that uses early data and copies the early data to another server instance. The second server instance accepts and processes the early data, even though it will not complete the TLS handshake. The attacker then allows the original connection to complete. Even if the early data is detected as a duplicate and rejected, the first server instance might allow the connection to complete. If the client then retries requests that were sent in early data, the request will be 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 the latter would be in violation of requirements in Section 8 of [TLS13]).

Clients that use early data MUST retry requests upon receipt of a 425 (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” (see Section 5.1).

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 hop. Likewise, it is necessary to have some means of explicitly triggering a retry when early data is not desired. Finally, it is necessary to know whether the client will actually perform such a retry.

To meet these needs, two signaling mechanisms are defined:

- The Early-Data header field is included in requests that might have been forwarded by an intermediary prior to the completion of the TLS handshake with its client.
- The 425 (Too Early) status code is defined for a 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 that a client understands the 425 (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 prior to the completion of the TLS handshake with its client 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 use the Early-Data header field if the request might have been subject to a replay and might already have been forwarded by it or another instance (see Section 6.2).

An intermediary MUST NOT remove this header field if it is present in a request. Early-Data MUST NOT appear in a Connection header field.

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 425 (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 header field and cannot be safely processed MUST be rejected using the 425 (Too Early) status code.

The Early-Data header field carries a single bit of information, and clients MUST include at most one instance. Multiple or invalid instances of the header field MUST be treated as equivalent to a single instance with a value of 1 by a server.

An Early-Data header field MUST NOT be included in responses or request trailers.

5.2. The 425 (Too Early) Status Code

A 425 (Too Early) status code indicates that the server is unwilling to risk processing a request that might be replayed.

User agents that send a request in early data are expected to retry the request when receiving a 425 (Too Early) response status code. A user agent SHOULD retry automatically, but any retries MUST NOT be sent in early data.

In all cases, an intermediary can forward a 425 (Too Early) status code. Intermediaries MUST forward a 425 (Too Early) status code if the request that it received and forwarded contained an Early-Data header field. Otherwise, an intermediary that receives a request in early data MAY automatically retry that request in response to a 425 (Too Early) status code, but it MUST wait for the TLS handshake to complete on the connection where it received the request.

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 425 status code unless one of these conditions is met.

The 425 (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. In particular, a request that is replayed might result in a different response, which might be observable from the length of protected data even if the content remains confidential.

6.1. Gateways and Early Data

A gateway MUST NOT forward requests that were received in early data unless it knows that the origin server it will forward to understands the Early-Data header field and will correctly generate a 425 (Too Early) status code. A gateway that is uncertain about origin server support for a given request SHOULD either delay forwarding the request until the TLS handshake with its client completes or send a 425 (Too Early) status code in response.

A gateway without at least one potential origin server that supports the Early-Data header field expends significant effort for what can at best be a modest performance benefit from enabling early data. If no origin server supports early data, it is more efficient to disable early data entirely.

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 (including gateways) need to agree and either reject the request or delay processing.

Disabling early data, delaying requests, or rejecting requests with the 425 (Too Early) status code are all equally good measures for mitigating replay attacks on requests that might be vulnerable to replay. Server instances can implement any of these measures and be considered consistent, even if different instances use different methods. Critically, this means that it is possible to employ different mitigations in reaction to other conditions, such as server load.

A server MUST NOT act on early data before the handshake completes if it and any other server instance could make a different decision about how to handle the same data.

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 425 (Too Early) status code often leads to clients retrying requests, which could result in increased load.

6.4. Out-of-Order Delivery

In protocols that deliver data out of order (such as QUIC [HQ]), early data can arrive after the handshake completes. A server MAY process requests received in early data after handshake completion only if it can rely on other instances correctly handling replays of the same requests.

7. IANA Considerations

This document registers the Early-Data header field in the “Permanent Message Header Field Names” registry located at <https://www.iana.org/assignments/message-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 425 (Too Early) status code in the “HTTP Status Codes” registry located at <https://www.iana.org/assignments/http-status-codes>.

Value:	425
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, <https://www.rfc-editor.org/info/rfc>.

[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, <https://www.rfc-editor.org/info/rfc>.

[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/rfc>.

[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, <https://www.rfc-editor.org/info/rfc>.

[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/rfc>.

[TLS13] 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/rfc>.

8.2. Informative References

[ALPN] Friedl, S., Popov, A., Langley, A., and E. Stephan, "Transport Layer Security (TLS) Application-Layer Protocol Negotiation Extension", RFC 7301, DOI 10.17487/RFC7301, July 2014, <https://www.rfc-editor.org/info/rfc>.

[HQ] Bishop, M., "Hypertext Transfer Protocol (HTTP) over QUIC", Work in Progress, draft-ietf-quic-http-14, August 2018.

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

Acknowledgments

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

Authors' Addresses

Martin Thomson
Mozilla
EMail: martin.thomson@gmail.com

Mark Nottingham
Fastly
EMail: mnot@mnot.net

Willy Tarreau
HAProxy Technologies
EMail: willy@haproxy.org