Protocol for Transposed Transactions over HTTP
Internet-Draft
Intended status: Standards Track
Expires: 15 November 2026

M. Thomson
Mozilla

14 May 2026

(Potato) - Reverse HTTP
draft-thomson-ptth-potato-01

Abstract

This document defines (Potato), a suite of reversed versions of HTTP for origin servers.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://martinthomson.github.io/potato/draft-thomson-ptth-potato.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-thomson-ptth-potato/.

Discussion of this document takes place on the Protocol for Transposed Transactions over HTTP Working Group mailing list (mailto:ptth@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/ptth/. Subscribe at https://www.ietf.org/mailman/listinfo/ptth/.

Source for this draft and an issue tracker can be found at https://github.com/martinthomson/potato.

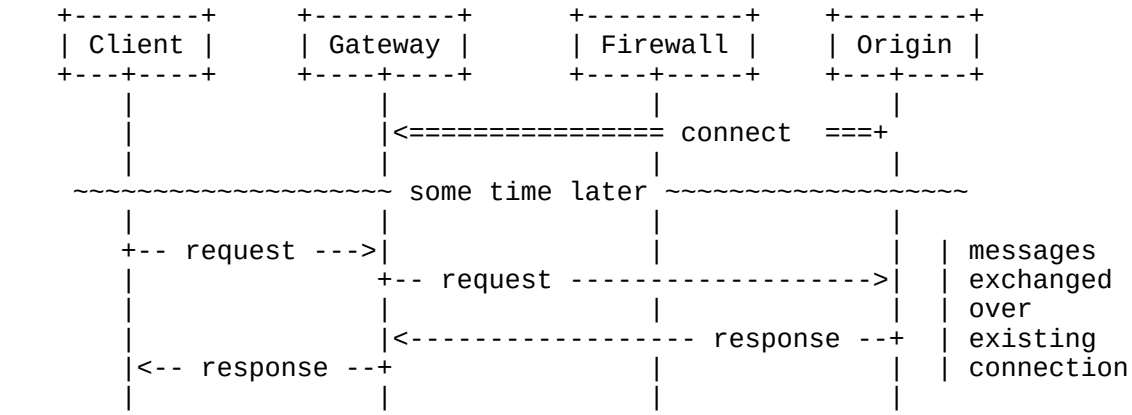
Status of This Memo

Table of Contents

1. Introduction

Operating an HTTP server that is accessable on the public internet creates an unmanageable risk of denial of service for many organizations, especially smaller operators. Content delivery networks (CDNs) are able to provide a measure of protection. This helps, but only to a degree. A CDN typically operates as a gateway, which imposes requirements on origin servers for reachability, security, and scalability that can be operationally challenging.

An alternative deployment model has emerged as a means of addressing this concern, where, rather than have the gateway initiate a connection to the origin, the origin connects to the gateway.



This arrangement has several advantages. It greatly simplifies the configuration of firewalls, which are better able to authorize outward-bound connections. It also changes the way that the origin scales up in the case where multiple server instances are needed.

This document defines alternative, "reversed", versions of HTTP [HTTP] protocols. These protocols are all nearly identical to their "forward" counterparts, with the exception that the client and server roles are inverted after completing the transport and TLS layer establishment.

This document defines reversed equivalents to HTTP/1.1 [HTTP/1.1], HTTP/2 [HTTP/2], and HTTP/3 [HTTP/3].

1.1. Comparative Notes

This is one of a set of different options in this space. Other designs include reverse tunnels [I-D.kazuho-ptth-ptth] and reverse HTTP transport [I-D.bt-httpbis-reverse-http].

These alternatives are all broadly capable of achieving the goal.

Key points of divergence exist around the use of tunnels (for reverse tunnels) and how different roles are authenticated and authorized.

There are some minor differences in use of terminology, which need to be cleared up.

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.

This document uses terminology from [HTTP].

Because having roles reversed can be especially confusing, this document tries to be consistent in its use of language.

The terms _client_ and _server_, when unqualified, refer to the HTTP client and HTTP server roles respectively.

In this document, a _gateway_ will generally be the HTTP client and an _origin server_ will generally be the HTTP server. These terms will be used where it makes sense.

Where it is necessary to identify a client or server role outside of the context of reversed HTTP, the term will be qualified. This typically will use the protocol where the role is used. For example, "TLS client role" or "acting as QUIC client".

3. Overview and Scope

This document describes how to establish reversed HTTP connections.

The overall approach is to allocate new Application-Layer Protocol Negotiation (ALPN) [ALPN] identifiers to each reversed HTTP version.

The origin server acts in the client role for establishing the transport and security layers of a connection to the gateway, which provides a server for those layers. That is, for HTTP/1.1 and HTTP/2, the origin server is the TCP and TLS client and the gateway is the TCP and TLS server. For HTTP/3, the origin server is the QUIC client and the gateway is the QUIC server.

If the gateway selects the reversed HTTP version in the protocol negotiation, the corresponding HTTP version is conducted with the usual roles reversed. The gateway becomes the HTTP client and the origin server becomes the HTTP server. No other changes are made to the protocol.

The document explores some of the implications of this design, including authorization (Section 6 and Section 5), and interactions with some protocol features. However, there are numerous factors

that are relevant when choosing an appropriate server deployment architecture. This document will address only those that are relevant to the design of the protocol and essential to its secure and correct operation.

This document does not define a protocol that allows both endpoints to make requests over the same connection. Separate connections can be used if requests need to be exchanged in both directions. Alternatively, tunnels (e.g., CONNECT or variants [RFC9298]) within a regular HTTP connection and a variant can use that tunnel. Finally, a connection could also be established and a tunnel with variant inside, but the role inversions and the authentication and authorization that are involved could be more confusing than helpful.

4. Reversed HTTP

This section defines the basic operation of for each major HTTP version.

All versions of require special handling for questions of server authority. Authorizing the origin server is discussed in Section 5. Similarly the means by which a gateway authorizes an origin server are discussed in Section 5.

Most HTTP features, including extensions to specific HTTP versions, are unaffected by the reversal of the protocol.

Only those features that depend on lower protocols layers that also depend on assumptions about how the the TLS or transport-layer role relates to the HTTP client or server role. Such features cannot be used without additional profiling.

The features that cannot be used with include: the Client-Cert HTTP field [RFC9440].

A small adjustment to the operation of the HTTP/3 Datagram extension [HTTP-DGRAM] is described in Section 4.3.

4.1. for HTTP/1.1

A reversed version of HTTP/1.1 [HTTP/1.1] is identified by the ALPN label "ph1".

To negotiate the use of for HTTP/1.1, an origin server advertises the "ph1" token in its TLS handshake using ALPN [ALPN] and a gateway selects that token.

Once a TLS connection is established, the origin server (as a TLS client) becomes the HTTP server and the gateway (as TLS server) becomes the HTTP client. Messages sent by the gateway are HTTP requests and messages from the origin server are HTTP responses.

HTTP/1.1 depends only on an undifferentiated stream of bytes. No special considerations apply to this version.

4.2. for HTTP/2

A reversed version of HTTP/2 [HTTP/2] is identified by the ALPN label "ph2".

To negotiate the use of for HTTP/2, an origin server advertises the "ph2" token in its TLS handshake using ALPN [ALPN] and a gateway selects that token.

Once a TLS connection is established, the origin server (as a TLS client) becomes the HTTP/2 server and the gateway (as TLS server) becomes the HTTP/2 client. Each send a preface (Section 3.4 of [HTTP/2]) and establishes an HTTP connection.

HTTP/2 depends only on an undifferentiated stream of bytes. No changes are necessary to the base protocol.

In operation, the gateway (as HTTP/2 client) initiates requests on odd-numbered streams, just like a regular HTTP/2 connection. Any push promises from the origin server (as HTTP server) are sent on even-numbered streams.

4.3. for HTTP/3

A reversed version of HTTP/3 [HTTP/3] is identified by the ALPN label "ph3".

To negotiate the use of for HTTP/3, an origin server advertises the "ph3" token in its QUIC handshake using ALPN [ALPN] and a gateway selects that token.

Once a QUIC connection is established, the origin server (as a QUIC client) becomes the HTTP/3 server and the gateway (as QUIC server) becomes the HTTP/3 client.

Unlike HTTP/2, the streaming layer that HTTP/3 uses is provided by QUIC. Therefore, the identifiers given to streams change. This should not be visible to the HTTP/3 layer; the HTTP/3 design generally does not depend on stream identifiers being in any particular form.

One exception to this is the Quarter Stream ID concept introduced in Section 2.1 of [HTTP-DGRAM]. Because requests in reversed HTTP/3 are made on bi-directional, server-initiated streams, those identifiers end with two bits (0b01); see Section 2.1 of [QUIC]. To construct a Quarter Stream ID from a stream identifier, any remainder is discarded. To construct a stream identifier from a Quarter Stream ID in reversed HTTP/3, the value is multiplied by 4, then the stream type (0b01) is added. This adjustment only requires an understanding of the type of QUIC stream that the HTTP client uses to make requests, which works for regular HTTP/3 as well.

> Note that the same Quarter Stream ID adjustment is safe to use in a regular HTTP/3 implementation of HTTP datagrams.

Any extension that makes similar assumptions about the structure of stream identifiers can be adjusted in a similar manner.

5. Authenticating and Authorizing Origin Servers

Arrangements between gateways and origin servers are often privately arranged. Therefore, how a gateway chooses to authorize an origin server does not necessarily benefit from having a single, standard mechanism.

This section describes a few options for origin server authentication. None of these approaches is mandated, as it is expected that private arrangements will be used in most deployments.

5.1. TLS Client Certificates

TLS client certificates can be used to authenticate TLS clients. That authentication could be used as the basis of authorization.

The choice of how to authenticate a client certificate is complex. Multiple options are possible, each with different operational and deployment properties:

* The gateway could use the same public key infrastructure (PKI) as the certificate that the gateway itself offers (see Section 6).

* A completely separate PKI could be used. This includes a PKI managed by the gateway.

* Self-signed certificates could be mapped to specific authorizations.

This option might introduce some challenges in implementation or deployment. Because the TLS connection that is established to the gateway uses the same endpoint as other clients, the TLS server software would need to make its protocol selection before deciding to request a client certificate.

5.2. HTTP Request

Once a connection is established, the gateway could make a request to a pre-arranged resource.

This request could be used to retrieve a shared secret that authorizes the origin server.

```
GET /some/prearranged/resource HTTP/1.1
Host: origin.example

HTTP/1.1 200 OK

{some prearranged secret}
```

However, this is trivially vulnerable to impersonation attack if an adversary is able to capture the secret.

A marginally more complex and more secure exchange might involve the origin server producing a signed object that covers a challenge produced by the gateway.

TODO: Should this document define a protocol?

5.3. Authorization for Limited Path Scope

A gateway does not need to authorize an origin server to serve all requests for an origin. A gateway might direct only a subset of URLs to a specific origin.

Nor does a gateway need to limit its authorization to a single origin. A gateway could send requests for multiple origins, varying host or port as configured.

The choice of which requests are forwarded to an origin server is a matter for gateway policy. That policy might be guided by the choice of credential presented by the origin server.

If a HTTP request is used to authorize the origin server, the response might include information that guides the gateway in determining which requests are forwarded over the connection. This could be as simple as having the origin server list path prefixes that it can serve or it could be bound to the credentials that are offered.

This might be used to enable different deployment architectures where the one logical origin server is distributed across multiple instances, with different paths being served by different hosts.

This approach is also compatible with a gateway that makes regular HTTP connections to some origin server nodes. A gateway might forward requests for the one origin to origin servers that use both regular and reversed HTTP based on whatever policy it chooses.

6. Authenticating Gateways

An origin server MUST authenticate a gateway unless an alternative means of authentication is privately arranged.

In all variants, when the the origin server establishes a TLS connection to the gateway, it confirms the identity of the gateway according to the rules of the respective, non-, HTTP version.

The rules in Section 4.3 of [HTTP] regarding how clients determine whether a server can answer a request are applied in reverse. That is, an origin server that cannot successfully determine that a gateway is authoritative for resources MUST NOT answer requests from that gateway.

Because the server and client roles are reversed, some methods for expanding the scope that a server can claim authority for are invalidated. Mechanisms like the ORIGIN frame [ORIGIN] remain valid as a means of limiting scope.

These requirements do not exist to provide protection against impersonation, which is generally the reason to require that a server establish authority. They exist to protect things like the confidentiality of responses.

TODO: For discovery, should this use generic SVCB records, HTTPS records, or a new RR type? It seems like HTTPS will work just fine for this.

7. TLS Early Data

A gateway can use TLS session resumption to make establishing connections more efficient. This includes the possibility of enabling TLS early data.

Unlike in regular HTTP where a client can use TLS early data to make a request, reversed HTTP gives the server an opportunity to speak first. This has little value, as an HTTP server generally has to wait for requests.

This provides similar properties to the first flight of server application data in a TLS 1.3 connection without early data. There, the server is able to send first.

Though that data has no replay risk, the data that a server might send in any HTTP version without a request is unlikely to produce side effects that might be a problem if replayed.

An origin can use early data to reduce the latency of settings or pre-populate header compression state. These both only apply to HTTP/2 and HTTP/3. These uses can avoid any replay attack risk, as these do not cause side effects beyond the connection state.

For a gateway, the first flight of application data can be used to make requests. However, this data is sent prior to receiving client certificates if those are used to authorize the origin server; see Section 5.1. This flight might be used to initiate a request to authorize the origin server, such as the option described in Section 5.2.

8. Scalability, Availability, and Connection Management

A gateway that is configured to make connections to an origin server is able to make connections as needed. The origin server might deploy a load balancer to manage inbound connections. When roles are reversed, having the origin server be responsible for connection establishment can mean that the gateway cannot assume that it is able to make new connections as demand increases.

This can mean that origin servers have a greater control over their availability and service scaling. Where an origin server that might need to scale up to handle increased demand might otherwise need to arrange for load balancing infrastructure, that becomes something that the gateway assumes more responsibilty for.

As a passive participant, a gateway is not able to act if an origin server goes offline. The origin server is responsible for establishing connections when it becomes available.

This can present scaling challenges as a gateway cannot make additional connections on the assumption that origin servers will scale to meet increased demand. Gateways can attempt to make additional requests over existing connections, but origin servers can use protocol features designed to limit resource commitment -- such as stream limits and flow control -- to manage load.

This document does not define any mechanism that might allow the gateway to communicate changes in demand for capacity to origin server instances. Defining mechanisms to help with adding or removing capacity on demand is a matter for future work.

The use of reversed HTTP/1.1 presents particular difficulty for connection management, which is borne by the origin server. The lack of any concurrency features in that protocol means that origin servers that use for HTTP/1.1 will need to manage a pool of connections if the gateway needs to handle requests with any amount of concurrency or volume. Consequently, a choice to use it is inadvisable except for very specific conditions.

9. Security Considerations

Using changes the denial of service profile for origin servers that rely on it. Such servers are able to use different tools to manage their reachability.

A gateway becomes a more central part of managing load, but the gateway no longer has direct control over connection establishment. This alters how gateways and origin servers need to be configured to handle scaling, availability, and connections; see Section 8 for details.

Other than these changes, the security considerations of HTTP and the specific HTTP version in use apply in full.

10. IANA Considerations

TODO: Register ALPN labels.

11. References

11.1. Normative 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/rfc/rfc7301>.

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

[HTTP-DGRAM] Schinazi, D. and L. Pardue, "HTTP Datagrams and the Capsule Protocol", RFC 9297, DOI 10.17487/RFC9297, August 2022, <https://www.rfc-editor.org/rfc/rfc9297>.

[HTTP/1.1] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/rfc/rfc9112>.

[HTTP/2] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/rfc/rfc9113>.

[HTTP/3] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

[QUIC] Iyengar, J., Ed. and M. Thomson, Ed., "QUIC: A UDP-Based Multiplexed and Secure Transport", RFC 9000, DOI 10.17487/RFC9000, May 2021, <https://www.rfc-editor.org/rfc/rfc9000>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

11.2. Informative References

[I-D.bt-httpbis-reverse-http] Schwartz, B. M., Reddy.K, T., Boucadair, M., and P. S. Tiesel, "Reverse HTTP Transport", Work in Progress, Internet-Draft, draft-bt-httpbis-reverse-http-01, 23 October 2023, <https://datatracker.ietf.org/doc/html/draft-bt-httpbis-reverse-http-01>.

[I-D.kazuho-ptth-ptth]
Oku, K., "Protocol for Transposed Transactions over HTTP", Work in Progress, Internet-Draft, draft-kazuho-ptth-ptth-00, 25 September 2025, <https://datatracker.ietf.org/doc/html/draft-kazuho-ptth-ptth-00>.

[ORIGIN] Nottingham, M. and E. Nygren, "The ORIGIN HTTP/2 Frame", RFC 8336, DOI 10.17487/RFC8336, March 2018, <https://www.rfc-editor.org/rfc/rfc8336>.

[RFC9298] Schinazi, D., "Proxying UDP in HTTP", RFC 9298, DOI 10.17487/RFC9298, August 2022, <https://www.rfc-editor.org/rfc/rfc9298>.

[RFC9440] Campbell, B. and M. Bishop, "Client-Cert HTTP Header Field", RFC 9440, DOI 10.17487/RFC9440, July 2023, <https://www.rfc-editor.org/rfc/rfc9440>.

Acknowledgments

This draft is based on discussions with many people. Both 奥 一穂 (Kazuho Oku) and Andrew McGregor both made helpful contributions.

Author’s Address

Martin Thomson
Mozilla
Email: mt@lowentropy.net