A WebSocket handshake proposal using Upgrade and CONNECT masking
draft-cridland-hybi-upgrade-connect-00

Abstract

This document proposes a handshake design for WebSockets based on an Upgrade with a first-frame of a pseudo-request CONNECT.

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 June 10, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Table of Contents

1. Conventions used in this document .................. 3
2. Introduction ........................................... 3
3. The Upgrade+CONNECT proposal ...................... 3
4. IANA Considerations .................................... 4
5. Security Considerations .............................. 4
6. Acknowledgements .................................... 5
7. References ............................................ 5
   7.1. Normative References ............................... 5
   7.2. Informative References ............................. 5
Author’s Address ........................................ 5
1. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [KEYWORDS].

Formal Syntax are to be considered normative, and are specified using [ABNF]. Where a formal syntax and the prose are in conflict, the formal syntax takes precedence.

This document uses two terms not well-defined elsewhere: Firstly, the websocket metadata is the data exchanged during the setup of the websocket proper, and secondly the websocket payload is the data exchanged within websocket frames. The frames as a whole make up the websocket data flow.

2. Introduction

The pure CONNECT based handshake outlined in [CONNECT] suffers from three problems. Firstly, it is not conceptually HTTP - it deliberately breaks that model in order to avoid manipulation by transparent proxies. Secondly, it is either reliant on IP-based routing, or else breaks semantic transparency of intermediaries. Thirdly, the protocol cannot be routed internally through a server without knowledge of the WebSocket spec, thus causing tight coupling within the server.

However, many of its properties are very good, in particular it successfully defeats the vast majority of inadvertent attempts to treat the websocket payload data as HTTP request/response flows. Since this incorrect interpretation by intermediaries is known to provide an attack vector for cache poisoning, this is a highly desirable feature.

On the other hand, the Upgrade+Hello proposal provides good fallback capability (being based on the HTTP Upgrade mechanism which itself has good fallback), and provides good feedback to applications that the WebSocket setup has completed.

3. The Upgrade+CONNECT proposal

This proposal uses features from both handshake proposals, in order to provide an HTTP compliant handshake with good failure properties and fallback capability.

In order to setup a WebSocket, a client first sends a suitable
fallback request along with the following additional header fields set:
Sec-WebSocket-Nonce  A random string selected by the browser (ie, not the Javascript). This string MUST contain no whitespace.
Sec-Websocket-Metadata  Metadata, base64 encoded.
Connection  This MUST be "Upgrade".
Upgrade  This MUST be Websocket.

Other header fields MAY be specified, for the purposes of the fallback request handling.

The server SHOULD respond with a 101 response. If it does so, it MUST include a Sec-Websocket-Result header field containing the HMAC of the metadata - without base64 - keyed by the nonce, and also a Sec-Websocket-Server-Nonce header field with a random string chosen by the server. AGain, this string MUST contain no whitespace.

After verifying the 101 response, the client then performs an HMAC of the metadata, keyed by the server’s nonce, and sends the following strings:
"CONNECT websocket.invalid:443 HTTP/1.1" CR LF
"Host: websocket.invalid" CR LF
"Connection: close" CR LF
"Sec-Websocket-Final: "
The client’s HMAC as above, hex encoded.
CR LF CR LF

Note that although this forms the same essential syntax as a CONNECT request, this is not an HTTP request - rather it is the first portion of WebSocket payload, and completes the handshake.

On the receipt of this final handshake step, WebSocket framing can start and payload data can be exchanged.

4. IANA Considerations

None - maybe header field name registrations?

5. Security Considerations

There is some discussion about whether the websocket payload can, even after a CONNECT, be used for cache poisoning. However, the experiment by Adam Barth et al clearly demonstrates that if this occurs at all, it is less than a 1:50,000 chance of finding such a proxy, and therefore my belief based on this evidence is that there is no need to mask the payload itself.
Although this design does not encrypt the metadata, it does constrain its form (by base64 encoding it) such that an attacker cannot inject headers or additional requests.

6. Acknowledgements

This is transparently just gluing together several other people’s ideas and passing them off in some way as my own. Adam Barth, Eric Rescorla, and Greg Wilkins all deserve credit, as do other members of the Hybi Working Group.

7. References

7.1. Normative References


7.2. Informative References


Author’s Address

Dave Cridland
Isode Limited
5 Castle Business Village
36, Station Road
Hampton, Middlesex TW12 2BX
GB

Email: dave.cridland@isode.com