Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 06, 2014

P. Hoffman
VPN Consortium

December 03, 2013

Minimal Unauthenticated Encryption (MUE) for HTTP/2
draft-hoffman-httpbis-minimal-unauth-enc-01

Abstract

HTTP/2 can be run under TLS as a transport if the origin for the request is an https: URL. There is a desire to encrypt much more web traffic than just the traffic that is already encrypted with TLS. This document proposes a method to establish unauthenticated encryption within normal HTTP/2 flows to prevent passive surveillance using as few as zero additional round trips. The method described here is definitely less robust than normal TLS and is thus not offered as an alternative to TLS for https: URLs; it is only offered as a fast way within HTTP/2 to cause unauthenticated encryption.

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 06, 2014.

Table of Contents

1. Introduction

The httpbis WG is investigating how to encrypt HTTP/2 [I-D.ietf-httpbis-http2] traffic using opportunistic methods for origins that use the http: URL. An earlier proposal ([I-D.nottingham-http2-encryption]) uses HTTP/2 headers to signal that the communication should switch to TLS [RFC5246]. This proposal uses a different mechanism: it creates a pair of encryption keys in HTTP/2 and uses that to encrypt the content while still running on port 80. A very comparison of the two methods is given in Section 6.

The protocol in this document is called minimal unauthenticated encryption (MUE). "Minimal" means that the protocol uses as few round trips as possible, and "unauthenticated" means that neither party is authenticated to the other. The lack of authentication makes this protocol faster and less complicated, but at the expense of making it suitable for preventing only passive surveillance, not attacks where the attacker is on-path.

The protocol described here starts encryption within three messages (1.5 round trips), and can start encryption successfully before the first message is sent back from the server to the client. This reduction of round trips before the beginning of encryption is possible because the protocol limits the types of key agreement and cryptographic negotiation that is possible. Other models have been proposed that can come to agreement with sometimes using fewer than two round trips, but at the expense of much more complicated error handling and sometimes needing even more than two round trips. Notably, QUIC [QUIC] appears to have achieved the most aggressive reduction under optimum circumstances, but with a pretty complicated model for connections between peers that don't know each other or when a peer changes long-term keys for any reason.

1.1. Terminology

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 RFC 2119, BCP 14 [RFC2119] and indicate requirement levels for compliant CBOR implementations.

2. Protocol Overview

MUE is negotiated in HTTP/2 headers. MUE is only initiated by the client. There are only three possible MUE messages: C1 (the first client message), S1 (the first server message), and C2 (the second client message, which is not needed if the server agreed to all the client's offers in C1). The two parties agree to a shared preliminary key using Diffie-Hellman key agreement, derive a pair of shared encryption keys using an agreed-to key derivation function (KDF), and start encrypting messages with an agreed-to encryption algorithm.

2.1. Cryptography Used in MUE

MUE uses three cryptographic primitves: key agreement, KDF, encryption.

The key agreement function used by MUE MUST be a form of anonymous Diffie-Hellman. This assures that that forward secrecy is always established as long as either the client or the server uses a newly-generated value.

The KDF MUST be an HMAC-based KDF as described in [RFC5869].

The encryption algorithm used by MUE MUST use authenticated encryption with associated data (AEAD) as defined in [RFC5116].

2.2. Client Initiation

C1 contains:

- the client's initial keying material
- the identifier for the type of key agreement used, including any parameters that are needed
- a list of all the crypto algorithms (key agreement, KDF, encryption) the client is willing to use
- the client nonce

If the client supports all the mandatory-to-implement algorithms (see Section 5), it can use the key agreement from that list and have a good guess that the server will be able to agree immediately. If the client has had a key agreement with the server in the past, the client might remember the server's algorithms and use those if they do not match the mandatory-to-implement list.

2.3. Server Response

S1 contains one of three things: agreement with the proposals in C1 and a finishing of the exchange, preliminary keying material based on one of the other key agreement algorithms and a list of the accepted algorithms, or an explicit rejection of doing MUE at all.

2.3.1. Server Can Use Key Material from C1 and Agrees With Proposed Algorithms

In the case of full agreement, S1 contains the following:

- an indicator of server success
- the server's keying material

- the list of just the agreed-to crypto algorithms (key agreement, KDF, encryption)
- the server nonce

The two sides use the KDF to convert the preliminary key to a pair of encryption keys. At that point, both the client and the server can begin encrypting HTTP/2 traffic. There is thus no C2 message.

2.3.2. Server Cannot Use Key Material from C1, But Agrees With Proposed Algorithms

If the server cannot use the client's keying material from C1 (most likely because the server does not support the key agreement type), but the client's list of crypto algorithms contains enough algorithms for the server to want to try, the server starts the key agreement over with a different key agreement algorithm. In this case, S1 contains:

- an indicator that the server is going to try to initiate the key exchange
- the server's initial keying material
- the identifier for the type of key agreement used, including any parameters that are needed
- the list of just the agreed-to crypto algorithms (key agreement, KDF, encryption)
- the server nonce

After receiving this S1, the client has two choices: reject the message (and send a C2 that would contain just an error indication), or accept it. In the latter case, C2 contains:

- an indicator of client success
- the client's keying material
- the list of just the agreed-to crypto algorithms (key agreement, KDF, encryption)
- the client nonce

The two sides use the KDF to convert the preliminary key to a pair of encryption keys. At that point, both the client and the server can begin encrypting HTTP/2 traffic.

2.3.3. Explicit Rejection by the Server

The server can explicitly reject the client's attempt to start MUE for any reason it wants. For example, the server may not be able to use any of the KDFs or encryption algorithms proposed by the client. Another example is that the server might not want to encrypt this particular traffic because it knows that the traffic is internal to a security zone. In this case, S1 would contain just an error indication, and there would be no C2 message.

2.4. Key Derivation

When the two parties have a preliminary key, it needs to be converted into a pair of encryption keys that can be used for encryption by the KDF. Each KDF is defined separately; one KDM is defined in Section 2.4.1.

Using the terminology of RFC 5869:

- the IKM is the preliminary key
- L is 2 times the length of the encryption keys needed plus 8
- the OKM is split into three items in the following order:
  - the client-to-server encryption key
  - the server-to-client encryption key
  - an 8-octet value that is used as a nonce for the AEAD

2.4.1. Definiton of HKDF-SHA256-MUE

This document defines one HKDF, "HKDF-SHA256-MUE". It follows the pattern described in RFC 5869 and has the following fields defined:

- Hash: SHA-256
- Salt: C1 | S1 | C2 (if C2 doesn't exist, the salt is just C1 | S1)
- Info: the value 0x6d75652d687474702f32

3. Message Format for the New Headers

...is a bikeshed that can be painted later. Seriously, it can be any binary format such as CBOR [RFC7049], some new TLV construct, or any of the many binary formats listed in Appendix E of the CBOR specification.

4. Encrypting HTTP/2 Content

...is not specified here yet. The topic of "I have a shared key and agreed-to encryption algorithm, I want to encrypt some of this HTTP content" has been widely discussed over the decade and many full and partial protocols have been presented. This document does not yet choose any of those, but it will clearly have to do so before it is complete. The httpbis WG can decide, given the differences between HTTP/1.1 and HTTP/2, what material (headers, messages, and so on) could be encrypted once a key and an algorithm is agreed to. (It is noted that there are two encryption keys, one for messages from the client to the server, and the other for the messages from the server to the client.)

5. Mandatory-to-Implement Crypto Algorithms

The following are the mandatory-to-implement algorithms. A client using just these choices is likely to be able to achieve encryption for the server's first message. (Note that the actual values can change as this document is discussed, but the list should have at least one value for each item.)

- Key agreement: ECDH with NIST curve P-256 [RFC5114]
- KDF: HKDF-SHA256-MUE as defined in Section 2.4.1
- Encryption: AES-256 in GCM mode [RFC5084]

6. Comparison of MUE with Redirection to TLS

One reason for doing encryption in the HTTP/2 stream instead of switching ports is that the transport semantics are clearer because HTTP/2 continues to run over TCP instead of in TLS. If a web browser starts with an http: URL type in the address bar, but then switches to TLS, should the GUI change at all? Would such a change be confusing to the user? Even if there is no change, how would the browser deal with a change in origins for any relative links in the HTML that is served from the https: redirection?

Another reason for using MUE is that it takes far fewer round trips than redirection to TLS. Browser vendors have been trying to reduce

the number of round trips before a user sees web content, and adding many round trips in order to do opportunistic encryption may be considered too expensive to them. MUE can achieve unauthenticated encryption in the first server response; redirection to TLS takes the three-way TCP handshake plus two TCP-level round trips.

Another reason that the httpbis WG might be interested in MUE is that it plays well with HTTP proxies given that it is all done in HTTP.

The main reason for doing encryption by redirecting to TLS is that the HTTP world already has TLS stacks and knows how to use them. Although there might need to be some changes either to the TLS API or even TLS itself to allow for unauthenticated TLS, this still might be much less work than adding crypto to HTTP/2 itself.

Another reason for doing encryption by redirection is that there will be complexity in specifying how HTTP/2 clients and servers will know which headers and body content is encrypted.

7. IANA Considerations

There will be a registry with all the codepoint values for the crypto algorithms. It will start small.

8. Security Considerations

MUE is only used for http: URLs, not at all for https: URLs. MUE and TLS solve different security problems and should definitely not be mistaken for each other. In all circumstances where TLS is used with server authentication, TLS has stronger security properties than MUE. Having said that, there is no security problem using MUE in the HTTP that is being carried under TLS.

Because MUE uses no authentication, it is susceptible to man-in-the-middle (MITM) attacks from active attackers. MUE's only security goal is to prevent passive surveillance, not to give any assurance to either party that there is no MITM.

Given the difference between MUE's security and TLS's security, and given the fact that many users do not even understand what security is given by TLS, implementers who add a green-lock GUI element for this type of encryption are likely to confuse users.

9. References

9.1. Normative References

[I-D.ietf-httpbis-http2]
Belshe, M., Peon, R., Thomson, M., and A. Melnikov, "Hypertext Transfer Protocol version 2.0", draft-ietf-httpbis-http2-08 (work in progress), November 2013.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5084] Housley, R., "Using AES-CCM and AES-GCM Authenticated Encryption in the Cryptographic Message Syntax (CMS)", RFC 5084, November 2007.

[RFC5114] Lepinski, M. and S. Kent, "Additional Diffie-Hellman Groups for Use with IETF Standards", RFC 5114, January 2008.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010.

9.2. Informative References

[I-D.nottingham-http2-encryption]
Nottingham, M., "Opportunistic Encryption for HTTP URIs", draft-nottingham-http2-encryption-01 (work in progress), October 2013.

[QUIC] Langley, A. and W-T. Chang, "QUIC Crypto", June 2013, <https://docs.google.com/document/d/1g5nIXAIkN_Y-7XJW5K45IblHd_L2f5LTaDUDwvZ5L6g>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, October 2013.

Author's Address

Paul Hoffman
VPN Consortium

Email: paul.hoffman@vpnc.org