TLS
Internet-Draft
Intended status: Standards Track
Expires: 28 April 2022

Y. Nir
Dell Technologies

25 October 2021

A Flags Extension for TLS 1.3
draft-ietf-tls-tlsflags-07

Abstract

A number of extensions are proposed in the TLS working group that carry no interesting information except the 1-bit indication that a certain optional feature is supported. Such extensions take 4 octets each. This document defines a flags extension that can provide such indications at an average marginal cost of 1 bit each. More precisely, it provides as many flag extensions as needed at 4 + the order of the last set bit divided by 8.

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 https://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 28 April 2022.

Copyright Notice

Table of Contents

1. Introduction

Since the publication of TLS 1.3 ([RFC8446]) there have been several proposals for extensions to this protocol, where the presence of the content-free extension in both the ClientHello and either the ServerHello or EncryptedExtensions indicates nothing except either support for the optional feature or an intent to use the optional feature. Examples:

* An extension that allows the server to tell the client that cross-SNI resumption is allowed: [I-D.sy-tls-resumption-group].

* An extension that is used to negotiate support for authentication using both certificates and external PSKs: [I-D.ietf-tls-tls13-cert-with-extern-psk].

* The post_handshake_auth extension from the TLS 1.3 base document indicates that the client is willing to perform post-handshake

authentication.

This document proposes a single extension called tls_flags that can enumerate such flag extensions and allowing both client and server to indicate support for optional features in a concise way.

None of the current proposed extensions allow for indication of support in ServerHello (SH), EncryptedExtensions (EE), Certificate (CT), or HelloRetryRequest (HRR) without first being indicated in ClientHello (CH). Similarly, none of the current proposed extensions allow for an indication of support in the client-side Certificate (CT) message without first being indicated in the server's CertificateRequest (CR) message. This restriction is enforced by the rules in Section 3.

1.1. Requirements and Other Notation

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

The term "flag extension" is used to denote an extension where the extension_data field is always zero-length in a particular context, and the presence of the extension denotes either support for some feature or the intent to use that feature.

The term "flag-type feature" denotes an options TLS 1.3 feature the support for which is negotiated using a flag extension, whether that flag extension is its own extension or a value in the extension defined in this document.

2. The tls_flags Extension

This document defines the following extension code point:

```
enum {
    ...
    tls_flags(TBD),
    (65535)
} ExtensionType;
```

This document also defines the data for this extension as a variable-length bit string, allowing for the encoding of up to 2040 features.

```
struct {
   opaque flags<1..255>;
} FlagExtensions;
```

The FlagExtensions field contains 8 flags in each octet. The length of the extension is the minimal length that allows it to encode all of the present flags. Within each octet, the bits are packed such that the first bit is the least significant bit and the eighth bit is

the most significant. Using zero-based indexing, the first octet holds flags 0-7, the second octet holds bits 8-15 and so on. For example, if we want to encode only flag number zero, the FlagExtension field will be 1 octet long, that is encoded as follows:

```
00000001
```

If we want to encode flags 1 and 5, the field will still be 1 octet long:

```
00100010
```

If we want to encode flags 3, 5, and 23, the field will have to be 3 octets long:

```
00101000 00000000 10000000
```

An implementation that receives an all-zero value for this extension or a value that contains trailing zero bytes MUST generate a fatal illegal_parameter alert.

Note that this document does not define any particular bits for this string. That is left to the protocol documents such as the ones in the examples from the previous section. Such documents will have to define which bit to set to show support, and the order of the bits within the bit string shall be enumerated in network order: bit zero is the high-order bit of the first octet as the flags field is transmitted.

3. Rules for The Flags Extension

A client that supports this extension and at least one flag extension SHALL send this extension with the flags field having bits set only for those extensions that it intends to set. It MUST NOT send this extension with a length of zero.

A server that supports this extension and also supports at least one of the flag-type features that use this extension and that were declared by the ClientHello extension SHALL send this extension with the intersection of the flags it supports with the flags declared by the client. The intersection operation may be implemented as a bitwise AND. The server may need to send multiple tls_flags extensions, up to one in each of the four response messages that may carry extensions: ServerHello (SH), EncryptedExtensions (EE), Certificate (CT), and HelloRetryRequest (HRR). It is up to the document for the specific feature to determine which of these should be used to acknowledge support.

A server MUST NOT indicate support for any flag-type feature not previously indicated by the client in the ClientHello in any response message. It MUST NOT include this extension in any of the four messages if it has no appropriate flag-type to indicate. This extension MUST NOT be included empty.

A server MAY indicate support for flag-type features in a flags extension of CertificateRequest (CR). Flag-type extensions that have been indicated in CR (and only such extensions) MAY be also be present in a flags extension of the Certificate (CT) message sent by the client. However, a client MUST NOT indicate support for any flag-type feature in a Certificate message that was not previously indicated by the server in its CertificateRequest message.

A server MAY indicate support for flag-type features in a flags extension of NewSessionTicket (NST). This message has no client-side response.

In summary, unsolicited flags may appear only in ClientHello, CertificateRequest, and NewSessionTicket.

An implementation that receives an invalid tls_flags extension MUST

terminate the TLS handshake with a fatal illegal_parameter alert.

4. IANA Considerations

IANA is requested to assign a new value from the TLS ExtensionType Values registry:

* The Extension Name should be tls_flags
* The TLS 1.3 value should be CH,SH,EE
* The Recommended value should be Y
* The Reference should be this document

IANA is also requested to create a new registry under the TLS namespace with name "TLS Flags" and the following fields:

* Value, which is a number between 0 and 2039. All potential values are available for assignment.
* Flag Name, which is a string
* Message, which like the "TLS 1.3" field in the ExtensionType registry contains the abbreviations of the messages that may contain the flag: CH, SH, EE, etc.

* Recommended, which is a Y/N value determined in the document defining the optional feature.
* Reference, which is a link to the document defining this flag.

The policy for this shall be "Specification Required" as described in [RFC8126].

The initial contents of the registry shall be one entry, as follows:

* Value shall be 8
* Flag Name shall be resumption_across_names
* Message shall be NST

* Recommended shall be set to no (N)

* The reference shall the the RFC-to-be [I-D.ietf-tls-cross-sni-resumption].

4.1. Guidance for IANA Experts

This extension allows up to 2040 flags. However, they are not all the same, because the length of the extension is determined by the highest set bit.

We would like to allocate the flags in such a way that the typical extension is as short as possible. The scenario we want to guard against is that in a few years some extension is defined that all implementations need to support and that is assigned a high number because all of the lower numbers have already been allocated. An example of such an extension is the Renegotiation Indication Extension defined in [RFC5746].

For this reason, the IANA experts should allocate the flags as follows:

* Flags 0-7 are reserved for documents coming out of the TLS working group with a specific request to assign a low number.

* Flags 8-31 are for standards-track documents that the experts believe will see wide adoption among either all users of TLS or a significant group of TLS users. For example, an extension that will be used by all web clients or all smart objects. The only entry in the initial registry is from this range.

* Flags 32-63 are for other documents, including experimental, that are likely to see significant adoption.

* Flags 64-79 are not to be allocated. They are for reserved for private use.

* Flags 80-2039 can be used for temporary allocation in experiments, for flags that are likely to see use only in very specific

environments, for national and corporate extensions, and as overflow, in case one of the previous categories has been exhausted.

5. Security Considerations

The extension described in this document provides a more concise way to express data that could otherwise be expressed in individual extensions. It does not send in the clear any information that would otherwise be sent encrypted, nor vice versa. For this reason this extension is neutral as far as security is concerned.

Extension authors should be aware that acknowledging flags in a tls_flags extension of the ServerHello and HelloRetryRequest messages expose this response to passive observers. Unless there is a special reason to place the response in the ServerHello, most flags should go in other (encrypted) messages.

6. Acknowledgements

The idea for writing this was expressed at the mic during the TLS session at IETF 104 by Eric Rescorla.

The current bitwise formatting was suggested on the mailing list by Nikos Mavrogiannopoulos.

Improvement to the encoding were suggested by Ilari Liusvaara, who also asked for a better explanation of the semantics of missing extensions.

Useful comments received from Martin Thomson, including the suggestion to eliminate the option to have the server send unsolicited flag types and the rules for where unsolicited flags can appear.

7. References

7.1. Normative References

[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/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/info/rfc8174>.

[RFC8446] 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/rfc8446>.

7.2. Informative References

[I-D.ietf-tls-cross-sni-resumption]
Vasiliev, V., "Transport Layer Security (TLS) Resumption across Server Names", Work in Progress, Internet-Draft, draft-ietf-tls-cross-sni-resumption-01, 13 May 2021, <https://www.ietf.org/archive/id/draft-ietf-tls-cross-sni-resumption-01.txt>.

[I-D.ietf-tls-tls13-cert-with-extern-psk]
Housley, R., "TLS 1.3 Extension for Certificate-Based Authentication with an External Pre-Shared Key", Work in Progress, Internet-Draft, draft-ietf-tls-tls13-cert-with-extern-psk-07, 23 December 2019, <https://www.ietf.org/archive/id/draft-ietf-tls-tls13-cert-with-extern-psk-07.txt>.

[I-D.sy-tls-resumption-group]
Sy, E., "TLS Resumption across Server Name Indications for TLS 1.3", Work in Progress, Internet-Draft, draft-sy-tls-resumption-group-00, 1 March 2019, <https://www.ietf.org/archive/id/draft-sy-tls-resumption-group-00.txt>.

[RFC5746] Rescorla, E., Ray, M., Dispensa, S., and N. Oskov, "Transport Layer Security (TLS) Renegotiation Indication Extension", RFC 5746, DOI 10.17487/RFC5746, February 2010, <https://www.rfc-editor.org/info/rfc5746>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/info/rfc8126>.

Appendix A. Change Log

RFC EDITOR: PLEASE REMOVE THIS SECTION AS IT IS ONLY MEANT TO AID THE WORKING GROUP IN TRACKING CHANGES TO THIS DOCUMENT.

draft-ietf-tls-tlsflags-02 set the maximum number of flags to 2048, and added guidance for the IANA experts.

draft-ietf-tls-tlsflags-01 allows server-only flags and allows the client to send an empty extension. Also modified the packing order of the bits.

draft-ietf-tls-tlsflags-00 had the same text as draft-nir-tls-tlsflags-02, and was re-submitted as a working group document following the adoption call.

Version -02 replaced the fixed 64-bit string with an unlimited bitstring, where only the necessary octets are encoded.

Version -01 replaced the enumeration of 8-bit values with a 64-bit bitstring.

Version -00 was a quickly-thrown-together draft with the list of supported features encoded as an array of 8-bit values.

Author's Address

Yoav Nir
Dell Technologies
9 Andrei Sakharov St
Haifa 3190500
Israel

Email: ynir.ietf@gmail.com