Stream: Internet Engineering Task Force (IETF)
RFC: 8701
Category: Informational
Published: January 2020
ISSN: 2070-1721
Author: D. Benjamin
Google LLC

RFC 8701
Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility

Abstract

This document describes GREASE (Generate Random Extensions And Sustain Extensibility), a mechanism to prevent extensibility failures in the TLS ecosystem. It reserves a set of TLS protocol values that may be advertised to ensure peers correctly handle unknown values.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for informational purposes.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are candidates for any level of Internet Standard; see Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8701.

Copyright Notice

Copyright (c) 2020 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 (https://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. Introduction

1.1. Requirements Language

2. GREASE Values

3. Client-Initiated Extension Points

3.1. Client Behavior

3.2. Server Behavior

4. Server-Initiated Extension Points

4.1. Server Behavior

4.2. Client Behavior

5. Sending GREASE Values

6. IANA Considerations

7. Security Considerations

8. Normative References

Acknowledgments

Author's Address

1. Introduction

The TLS protocol [RFC8446] includes several points of extensibility, including the list of cipher suites and several lists of extensions. The values transmitted in these lists identify implementation capabilities. TLS follows a model where one side, usually the client, advertises capabilities, and the peer, usually the server, selects them. The responding side must ignore unknown values so that new capabilities may be introduced to the ecosystem while maintaining interoperability.

However, bugs may cause an implementation to reject unknown values. It will interoperate with existing peers, so the mistake may spread through the ecosystem unnoticed. Later, when new values are defined, updated peers will discover that the metaphorical joint in the protocol has rusted shut and the new values cannot be deployed without interoperability failures.

To avoid this problem, this document reserves some currently unused values for TLS implementations to advertise at random. Correctly implemented peers will ignore these values and interoperate. Peers that do not tolerate unknown values will fail to interoperate, revealing the mistake before it is widespread.

In keeping with the rusted joint metaphor, this technique is called "GREASE" (Generate Random Extensions And Sustain Extensibility).

1.1. Requirements Language

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. GREASE Values

This document reserves a number of TLS protocol values, referred to as GREASE values. These values were allocated sparsely to discourage server implementations from conditioning on them. For convenience, they were also chosen so all types share a number scheme with a consistent pattern while avoiding collisions with any existing applicable registries in TLS.

The following values are reserved as GREASE values for cipher suites and Application-Layer Protocol Negotiation (ALPN) [RFC7301] identifiers:

{0x0A,0x0A}

{0x1A,0x1A}

{0x2A,0x2A}

{0x3A,0x3A}

{0x4A,0x4A}

{0x5A,0x5A}

{0x6A,0x6A}

{0x7A,0x7A}

{0x8A,0x8A}

{0x9A,0x9A}

{0xAA,0xAA}

{0xBA,0xBA}

{0xCA,0xCA}

{0xDA,0xDA}

{0xEA,0xEA}

{0xFA,0xFA}

The following values are reserved as GREASE values for extensions, named groups, signature algorithms, and versions:

0x0A0A

0x1A1A

0x2A2A

0x3A3A

0x4A4A

0x5A5A

0x6A6A

0x7A7A

0x8A8A

0x9A9A

0xAAAA

0xBABA

0xCACA

0xDADA

0xEAEA

0xFAFA

The values allocated above are thus no longer available for use as TLS or DTLS [RFC6347] version numbers.

The following values are reserved as GREASE values for PskKeyExchangeModes:

0x0B

0x2A

0x49

0x68

0x87

0xA6

0xC5

0xE4

3. Client-Initiated Extension Points

Most extension points in TLS are offered by the client and selected by the server. This section details client and server behavior around GREASE values for these.

3.1. Client Behavior

When sending a ClientHello, a client **MAY** behave as follows:

- A client **MAY** select one or more GREASE cipher suite values and advertise them in the "cipher_suites" field.
- A client **MAY** select one or more GREASE extension values and advertise them as extensions with varying length and contents.
- A client **MAY** select one or more GREASE named group values and advertise them in the "supported_groups" extension, if sent. It **MAY** also send KeyShareEntry values for a subset of those selected in the "key_share" extension. For each of these, the "key_exchange" field **MAY** be any value.
- A client **MAY** select one or more GREASE signature algorithm values and advertise them in the "signature_algorithms" or "signature_algorithms_cert" extensions, if sent.
- A client **MAY** select one or more GREASE version values and advertise them in the "supported_versions" extension, if sent.
- A client **MAY** select one or more GREASE PskKeyExchangeMode values and advertise them in the "psk_key_exchange_modes" extension, if sent.
- A client **MAY** select one or more GREASE ALPN identifiers and advertise them in the "application_layer_protocol_negotiation" extension, if sent.

Clients **MUST** reject GREASE values when negotiated by the server. In particular, the client **MUST** fail the connection if a GREASE value appears in any of the following:

- The "version" value in a ServerHello or HelloRetryRequest
- The "cipher_suite" value in a ServerHello
- Any ServerHello extension
- Any HelloRetryRequest, EncryptedExtensions, or Certificate extension in TLS 1.3
- The "namedcurve" value in a ServerKeyExchange for an Ephemeral Elliptic Curve Diffie-Hellman (ECDHE) cipher in TLS 1.2 [RFC5246] or earlier
- The signature algorithm in a ServerKeyExchange signature in TLS 1.2 or earlier
- The signature algorithm in a server CertificateVerify signature in TLS 1.3

Note that this can be implemented without special processing on the client. The client is already required to reject unknown server-selected values, so it may leave GREASE values as unknown and reuse the existing logic.

3.2. Server Behavior

When processing a ClientHello, servers **MUST NOT** treat GREASE values differently from any unknown value. Servers **MUST NOT** negotiate any GREASE value when offered in a ClientHello. Servers **MUST** correctly ignore unknown values in a ClientHello and attempt to negotiate with one of the remaining parameters. (There may not be any known parameters remaining, in which case parameter negotiation will fail.)

Note that these requirements are restatements or corollaries of existing server requirements in TLS.

4. Server-Initiated Extension Points

Some extension points are offered by the server and selected by the client. This section details client and server behavior around GREASE values for these.

4.1. Server Behavior

When sending a CertificateRequest in TLS 1.3, a server **MAY** behave as follows:

- A server **MAY** select one or more GREASE extension values and advertise them as extensions with varying length and contents.
- A server **MAY** select one or more GREASE signature algorithm values and advertise them in the "signature_algorithms" or "signature_algorithms_cert" extensions, if present.

When sending a NewSessionTicket message in TLS 1.3, a server **MAY** select one or more GREASE extension values and advertise them as extensions with varying length and contents.

Servers **MUST** reject GREASE values when negotiated by the client. In particular, the server **MUST** fail the connection if a GREASE value appears in any of the following:

- Any Certificate extension in TLS 1.3
- The signature algorithm in a client CertificateVerify signature

Note that this can be implemented without special processing on the server. The server is already required to reject unknown client-selected values, so it may leave GREASE values as unknown and reuse the existing logic.

4.2. Client Behavior

When processing a CertificateRequest or NewSessionTicket, clients **MUST NOT** treat GREASE values differently from any unknown value. Clients **MUST NOT** negotiate any GREASE value when offered by the server. Clients **MUST** correctly ignore unknown values offered by the server and attempt to negotiate with one of the remaining parameters. (There may not be any known parameters remaining, in which case parameter negotiation will fail.)

Note that these requirements are restatements or corollaries of existing client requirements in TLS.

5. Sending GREASE Values

Implementations advertising GREASE values **SHOULD** select them at random. This is intended to encourage implementations to ignore all unknown values rather than any individual value. Implementations **MUST** honor protocol specifications when sending GREASE values. For instance,

Section 4.2 of [RFC8446] forbids duplicate extension types within a single extension block. Implementations sending multiple GREASE extensions in a single block must therefore ensure the same value is not selected twice.

Implementations **SHOULD** balance diversity in GREASE advertisements with determinism. For example, a client that randomly varies GREASE value positions for each connection may only fail against a broken server with some probability. This risks the failure being masked by automatic retries. A client that positions GREASE values deterministically over a period of time (such as a single software release) stresses fewer cases but is more likely to detect bugs from those cases.

6. IANA Considerations

This document updates the "TLS Cipher Suites" registry, available at <https://www.iana.org/assignments/tls-parameters>:

Value	Description	DTLS-OK	Recommended	Reference
{0x0A,0x0A}	Reserved	Y	N	[RFC8701]
{0x1A,0x1A}	Reserved	Y	N	[RFC8701]
{0x2A,0x2A}	Reserved	Y	N	[RFC8701]
{0x3A,0x3A}	Reserved	Y	N	[RFC8701]
{0x4A,0x4A}	Reserved	Y	N	[RFC8701]
{0x5A,0x5A}	Reserved	Y	N	[RFC8701]
{0x6A,0x6A}	Reserved	Y	N	[RFC8701]
{0x7A,0x7A}	Reserved	Y	N	[RFC8701]
{0x8A,0x8A}	Reserved	Y	N	[RFC8701]
{0x9A,0x9A}	Reserved	Y	N	[RFC8701]
{0xAA,0xAA}	Reserved	Y	N	[RFC8701]
{0xBA,0xBA}	Reserved	Y	N	[RFC8701]
{0xCA,0xCA}	Reserved	Y	N	[RFC8701]
{0xDA,0xDA}	Reserved	Y	N	[RFC8701]
{0xEA,0xEA}	Reserved	Y	N	[RFC8701]
{0xFA,0xFA}	Reserved	Y	N	[RFC8701]

Table 1: Additions to the TLS Cipher Suites Registry

This document updates the "TLS Supported Groups" registry, available at <https://www.iana.org/assignments/tls-parameters>:

Value	Description	DTLS-OK	Recommended	Reference
2570	Reserved	Y	N	[RFC8701]
6682	Reserved	Y	N	[RFC8701]
10794	Reserved	Y	N	[RFC8701]
14906	Reserved	Y	N	[RFC8701]
19018	Reserved	Y	N	[RFC8701]

Value	Description	DTLS-OK	Recommended	Reference
23130	Reserved	Y	N	[RFC8701]
27242	Reserved	Y	N	[RFC8701]
31354	Reserved	Y	N	[RFC8701]
35466	Reserved	Y	N	[RFC8701]
39578	Reserved	Y	N	[RFC8701]
43690	Reserved	Y	N	[RFC8701]
47802	Reserved	Y	N	[RFC8701]
51914	Reserved	Y	N	[RFC8701]
56026	Reserved	Y	N	[RFC8701]
60138	Reserved	Y	N	[RFC8701]
64250	Reserved	Y	N	[RFC8701]

Table 2: Additions to the TLS Supported Groups Registry

This document updates the "TLS ExtensionType Values" registry, available at <https://www.iana.org/assignments/tls-extensiontype-values>:

Value	Extension Name	TLS 1.3	Recommended	Reference
2570	Reserved	CH, CR, NST	N	[RFC8701]
6682	Reserved	CH, CR, NST	N	[RFC8701]
10794	Reserved	CH, CR, NST	N	[RFC8701]
14906	Reserved	CH, CR, NST	N	[RFC8701]
19018	Reserved	CH, CR, NST	N	[RFC8701]
23130	Reserved	CH, CR, NST	N	[RFC8701]
27242	Reserved	CH, CR, NST	N	[RFC8701]
31354	Reserved	CH, CR, NST	N	[RFC8701]
35466	Reserved	CH, CR, NST	N	[RFC8701]
39578	Reserved	CH, CR, NST	N	[RFC8701]

Value	Extension Name	TLS 1.3	Recommended	Reference
43690	Reserved	CH, CR, NST	N	[RFC8701]
47802	Reserved	CH, CR, NST	N	[RFC8701]
51914	Reserved	CH, CR, NST	N	[RFC8701]
56026	Reserved	CH, CR, NST	N	[RFC8701]
60138	Reserved	CH, CR, NST	N	[RFC8701]
64250	Reserved	CH, CR, NST	N	[RFC8701]

Table 3: Additions to the TLS ExtensionType Values Registry

This document updates the "TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs" registry, available at <https://www.iana.org/assignments/tls-extensiontype-values>:

Protocol	Identification Sequence	Reference
Reserved	0x0A 0x0A	[RFC8701]
Reserved	0x1A 0x1A	[RFC8701]
Reserved	0x2A 0x2A	[RFC8701]
Reserved	0x3A 0x3A	[RFC8701]
Reserved	0x4A 0x4A	[RFC8701]
Reserved	0x5A 0x5A	[RFC8701]
Reserved	0x6A 0x6A	[RFC8701]
Reserved	0x7A 0x7A	[RFC8701]
Reserved	0x8A 0x8A	[RFC8701]
Reserved	0x9A 0x9A	[RFC8701]
Reserved	0xAA 0xAA	[RFC8701]
Reserved	0xBA 0xBA	[RFC8701]
Reserved	0xCA 0xCA	[RFC8701]
Reserved	0xDA 0xDA	[RFC8701]
Reserved	0xEA 0xEA	[RFC8701]

Protocol	Identification Sequence	Reference
Reserved	0xFA 0xFA	[RFC8701]

Table 4: Additions to the TLS Application-Layer Protocol Negotiation (ALPN) Protocol IDs Registry

7. Security Considerations

GREASE values cannot be negotiated, so they do not directly impact the security of TLS connections.

Historically, when interoperability problems arise in deploying new TLS features, implementations have used a fallback retry on error with the feature disabled. This allows an active attacker to silently disable the new feature. By preventing a class of such interoperability problems, GREASE reduces the need for this kind of fallback. Implementations **SHOULD NOT** retry with GREASE disabled on connection failure. While allowing an attacker to disable GREASE is unlikely to have immediate security consequences, such a fallback would prevent GREASE from defending against extensibility failures.

If an implementation does not select GREASE values at random, it is possible it will allow for fingerprinting of the implementation or perhaps even of individual users. This can result in a negative impact to a user's privacy.

8. 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>.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <https://www.rfc-editor.org/info/rfc5246>.

[RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347, January 2012, <https://www.rfc-editor.org/info/rfc6347>.

[RFC7301] 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/rfc7301>.

[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>.

Acknowledgments

The author would like to thank Adam Langley, Nick Harper, and Steven Valdez for their feedback and suggestions. In addition, the rusted joint metaphor is originally due to Adam Langley.

Author's Address

David Benjamin
Google LLC
Email: davidben@google.com