Abstract

This memo defines two TLS protocol extensions. The extensions defined here allow the negotiation of a uniform level of security for the TLS authentication algorithms, and make the TLS fallback protocol version negotiation explicit.

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 December 10, 2011.

Copyright Notice

Copyright (c) 2011 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
1. Introduction

In this document we define two new TLS protocol [RFC5246] extension types. The "MAC security parameter" and the "Fallback protocols". The former is an extension to impose a uniform level of security to the TLS authentication algorithms, and the latter modifies the TLS protocol version negotiation by making explicit the fallback protocol version. We elaborate on the rationale for each extension on the subsequent sections.

2. Terminology

This document uses the same notation and terminology used in the TLS Protocol specification [RFC5246].

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 [RFC2119].

In this document we use the term security parameter, to describe the security level of an authentication algorithm. The security parameter is in bits and for a parameter X it would imply that on average 2^X operations would be required for a successful attack. This notion is identical with the security level of [ECRYPT] and the "bits of security" term defined in [SP800-57].

3. MAC security parameter

3.1. Rationale

Each TLS ciphersuite specifies among others the MAC algorithm to be used for record message authentication, as well as the PRF that is used for handshake finished message authentication. Those algorithms serve a similar purpose (message authentication), but do not necessarily match. For example TLS_RSA_WITH_AES_256_CBC_SHA256 from the main TLS protocol specification [RFC5246] uses the 256-bit HMAC-SHA256 for record message authentication and only the 96-bits of the HMAC-SHA256 PRF for finished message authentication.

Moreover the TLS PRF output size for the handshake finished message is fixed for all known ciphersuites to 96-bits, but the MAC algorithm for the record messages can be negotiated and typically is the full output of SHA1 or SHA256. This discrepancy causes confusion on profiles that explicitly specify a security parameter of 128-bits or 192-bits as in [RFC5430], as it can be argued that the actual security parameter cannot be more than 96-bits. That is because in
these cases the weakest link in message authentication is the MAC for
the handshake finished message, which has a security parameter of 96-
bits.

Having varying security parameters in an authentication protocol
makes the weakest links best candidates for attacks, thus it is
desirable to offer a consistent security parameter for the whole
protocol duration. This will benefit profiles and applications that
require a fixed security parameter. Also in case the security
parameter is lower than the output of the hash function this
extension will reduce the extra transferred bytes used for the record
layer MAC. The latter is useful for DTLS [RFC4347] datagrams that
are often transferred using a small maximum transfer unit.

3.2. The extension

To allow negotiation of a consistent security parameter for a
session’s MAC algorithms we add a new extension
"mac_security_parameter", with value TBD-BY-IANA, to the enumerated
ExtensionType defined in [RFC5246]. Clients MAY include the
extension of type "mac_security_parameter" in the (extended) client
hello. The "extension_data" field of this extension MUST contain a
"MACSecurityParameterList":

```c
enum{
      96(1), 112(2), 128(3), 192(4), 256(5) (255)
    } MACSecurityParameter;

struct {
    MACSecurityParameter mac_security_parameter_list<1..2^8-1>
} MACSecurityParameterList;
```

The "MACSecurityParameter" contains the security parameters to be
negotiated in bits.

Servers that receive an extended client hello containing a
"mac_security_parameter" extension, MAY accept the requested maximum
fragment length by including an extension of type
"mac_security_parameter" in the (extended) server hello. The
"extension_data" field of this extension SHALL contain
"MACSecurityParameter" whose value is one of the requested
parameters.

If the server replies with a "MACSecurityParameter" then the MAC of
the record messages MUST be restricted to the minimum between the
allowed size of the negotiated MAC and the selected
MACSecurityParameter size, by both parties. The "verify_data_length"
of the Finished message SHALL be set equal to the negotiated "MACSecurityParameter" size.

Note: TLS 1.2 allows ciphersuites to specify individually the "verify_data_length". Implementations conforming to this document MUST ignore the ciphersuite "verify_data_length" if a "MACSecurityParameter" has been selected by the server.

Note: A client MUST NOT send both this extension and the truncated HMAC extension [RFC6066]. Servers that received both extensions MAY terminate the handshake with an "illegal_parameter" alert. If the server chooses not to terminate the handshake it MUST ignore the truncated HMAC extension.

4. Fallback protocols

4.1. Rationale

The TLS protocol negotiation works by having the client send the highest supported version in the client hello message. If the server does not support this protocol version has to reply with his highest version. For example if the client’s highest version is TLS 1.2, and the server’s is TLS 1.1, the following negotiation should occur:

Client                                               Server

ClientHello (TLS 1.2)  -------->  ServerHello (TLS 1.1)

This however requires the client to implement TLS 1.1, as well as TLS 1.0 and SSL 3.0, because it is not possible for the client to predict the highest supported version by the server. That requirement increases client complexity as there is no way for a client to support only a single protocol, e.g. TLS 1.0 or SSL 3.0, to be used as a fallback from TLS 1.2.

4.2. The extension

This extension allows clients to indicate precisely the supported TLS protocol versions for consideration by the server as a fallback. To indicate the supported protocol versions clients MAY include an extension of type "fallback_protocols" in the (extended) client hello. The "extension_data" field of this extension MUST contain a "ProtocolVersionList":

struct {
    uint8 major;
    uint8 minor;
} ProtocolVersion;

struct {
    ProtocolVersion supported_protocol_list<1..2^8-1>
} ProtocolVersionList;

The advertised in the client hello TLS protocol version SHOULD NOT be included in the "ProtocolVersionList".

A server that receives a client hello containing the "fallback_protocols" extension, MAY use the information contained in the extension to select the highest commonly supported protocol version. Servers MUST NOT send this extension.

5. Security considerations

5.1. MAC security parameter

This extension poses a common security parameter on both the algorithms used for the record message authentication, and the algorithm uses for handshake finished message authentication. These are typically different algorithms but adhere to the MAC definition of [ECRYPT]. Under that definition the upper bounds on the security of the MAC are the size of the key used and the length of the MAC itself.

In the finished messages the TLS "master_secret" is used as the key of the MAC, that consists of 384 bits of keying material. The MAC if this extension is used will be truncated to the selected "MACSecurityParameter" size (which is always less than 384), providing a maximum security parameter of the truncated size.

The defined ciphersuites in TLS use either HMAC or the GCM cipher mode to authenticate the record messages. When HMAC is used the keying material used for HMAC is of the same size as the HMAC output, thus if truncated to the "MACSecurityParameter" will provide a maximum security parameter of the truncated size. If no truncation occurs the security parameter is that of the HMAC output size. The GCM ciphers according to [NIST-GCM] have issues when the security parameter is low (32 or 64 bits) that require special constrains, but no issues are known for the security parameters defined in this document. The maximum security parameter supported by GCM is 128.
5.2. Fallback protocols

The addition of the fallback cipher turns the implicit TLS version fallback mechanism to an explicit one, similar to the ciphersuite negotiation. For this reason it is believed that no security complications not already present in TLS are introduced.

6. IANA Considerations

This document defines new TLS extensions "mac_security_parameter" (value TBD-BY-IANA), "fallback_protocols" (value TBD-BY-IANA) whose value should be assigned from the TLS ExtensionType Registry defined in [RFC5246].

7. References

7.1. Normative References


7.2. Informative References


Author’s Address

Nikos Mavrogiannopoulos
ESAT/COSIC Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, bus 2446
Leuven-Heverlee, B-3001
Belgium

EMail: nikos.mavrogiannopoulos@esat.kuleuven.be