Internet Engineering Task Force
INTERNET DRAFT

M. Badra
ENST Paris
I. Hajjeh
ESRGroups

Expires: March 2006

October 10, 2005

TLS Multiplexing

<draft-badra-hajjeh-mtls-00.txt>

Status

Copyright Notice

Abstract

TLS is the famous protocol that provides authentication and data protection for communication between two entities. However, missing from the protocol is a way to multiplex application data over the same TLS session.

This document defines a new TLS sub-protocol called MTLS running over TLS (or DTLS) Record protocol. The MTLS design provides application multiplexing over a single TLS session. Instead of associating a TLS connection with each application, MTLS allows

several applications to protect their exchanges over a single TLS session.

1 Introduction

SMTP over TLS [SMTPTLS], HTTP over TLS [HTTPTLS], POP over TLS and IMAP over TLS [POPTLS] are examples of securing, respectively, SMTP, HTTP, POP and IMAP data exchanges using the TLS protocol [TLS].

TLS ([TLS], [TLSv1.1]) is the most deployed security protocol for securing exchanges, authenticating entities and for generating and distributing cryptographic keys. However, what is missing from the protocol is the way to multiplex application data over the same TLS session.

Actually, TLS (or DTLS [DTLS]) clients and servers MUST establish a TLS (or DTLS) session for each application they want to run over TCP (or UDP). However, some applications may agree or be configured to use the same security policies or parameters (f.g. authentication method and cipher_suite) and then to share one and only one TLS session to protect their exchanges. In this way, this document extends TLS to allow an application multiplexing functionality over TLS.

The document motivations included:

- TLS is application protocol-independent. Higher-level protocol can operate on top of the TLS protocol transparently.

- TLS is a modular nature protocol. Since TLS is developed in four independent protocols, the approach defined in this document can be added by extending the TLS protocol and with a total reuse of pre-existing TLS infrastructures and implementations.

- It provides a secure VPN tunnel over a transport layer.

- Establishing a single session for a number of applications reduces resource consumption, latency and messages flow that are associated with executing simultaneous TLS sessions.

- TLS can not forbid an intruder to analyze the traffic and cannot protect data from inference. Thus, the intruder can know the type of application data transmitted through the TLS session. However, the extension defined in this document allows, by its design, data protection against inference.

1.2 Requirements language

The key words "MUST", "SHALL", "SHOULD", and "MAY", in this document

are to be interpreted as described in RFC-2119.

2 TLS multiplexing overview and considerations

This document defines a new TLS sub-protocol called Multiplexing TLS (MTLS) to handle data multiplexing, and it specifies the content type mtls(26) for this sub-protocol.

MTLS communication can take place over TCP or UDP. The default port is TBC, but other ports can be used.

2.1 Handshake

Based on the TLS Extensions [TLSExt], a client and a server can, in an ordinary TLS handshake, negotiate the future use of MTLS. If the client does attempt to initiate a TLS connection using MTLS with a server that does not support it, it will be automatically alerted. For servers aware of MTLS but not wishing to use it, it will gracefully revert to an ordinary TLS handshake or stop the negotiation.

The negotiation starts usually with the client determining whether the server is capable of and willing to use MTLS or not. In order to allow a TLS client to negotiate the application multiplexing functionality, a new extension type SHOULD be added to the Extended Client and Extended Server Hello messages.

This document defines an extension of type "application_layer_protocol". The "extension_data" field of this extension contains a "data_multiplexing", where:

```
    Struct {
           ApplicationLayerProtocol alp_list<0..2^20-1>;
        } data_multiplexing;

    struct {
           ApplicationpProtocolName apn;
           select (Version)
              case { 3, 1 }:// TLS Version 1.0
                TCPPort tcp_port;
              case { 3, 2 }:// TLS Version 1.1
                TCPPort tcp_port;
              case { 254, 255 }:// Datagram TLS Version 1.0
                UDPPort udp_port;
        } ApplicationLayerProtocol;

    opaque TCPPort[2];
    opaque UDPPort[2];
    Opaque ApplicationpProtocolName<1..16>;
```

tcp_port (respectively udp_port) is the application port number at

the server side. The client MUST use as destination ports, the TCP (respectively UDP) port numbers that are assigned by IANA.

Application layer running on unreliable links MUST be negotiated in a separate session using the DTLS Handshake [DTLS].

Note: if the server agrees, the client SHOULD establish a single TLS (respectively DTLS) session for all applications it wishes to run over TCP (respectively UDP). In this case, the client SHOULD send a data multiplexing extension containing "ALL" as ApplicationpProtocolName value and "NULL" as TCPPort (or UDPPort) value. If the server is not able to negotiate such session, it replays with a list of applications (names and ports) it can accept to run through a single TLS session, falls back on an ordinary TLS handshake or stops the negotiation.

2.1.1. Multi-connections during application session

Once the establishment is complete, the client MAY open many connections related to an arbitrary application over the secure session. In this case, the application client MUST locally reserve a port number for each connection. Next, the client application sends its request to the MTLS client which is listening on the TBC port number. This latter will check if it has an established secure session with the requested hostname (the server). If then it checks if the application protocol name has been accepted to run over MTLS, before sends the request to the MTLS server.

2.2 MTLS sub-protocol

The structure of MTLS packet is described below. The first 8 bytes of the packet represent the source and the destination ports of the connexion, and the length contains the length of the MTLS data.

```
enum {
     change_cipher_spec(20), alert(21), handshake(22),
     application_data(23), mtls(26), (255)
  } ContentType;

struct {
       uint32 SourcePort
       uint32 DestinationPort
       uint16 length;
       opaque data[MTLSPlaintext.length];
    } MTLSPlaintext;
```

The TLS Record Layer receives data from MTLS, supposes it as uninterpreted data and applies the fragmentation and the cryptographic operations on it, as defined in [TLS].

Note: multiple MTLS fragments MAY be coalesced into a single TLSPlaintext record.

Received data is decrypted, verified, decompressed, and reassembled, then delivered to MTLS sub-protocol. Next, the MTLS sends data to the appropriate application using the source and destination port numbers and the length value.

Security Considerations

Security issues are discussed throughout this document, and in [TLS], [TLSv1.1], [DTLS] and [TLSEXT] documents.

If a fatal error related to a connexion of an arbitrary application is occurred, the secure session MUST NOT be resumed.

IANA Considerations

This document requires IANA to allocate the TBC TCP and UDP port numbers.

Acknowledgment

This document defined TLS Multiplexing for applications running over IP. Beyond that definition, generic options may be added to future versions of the current document.

References

[TLS] Dierks, T., et. al., "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[TLSExt] Blake-Wilson, S., et. al., "Transport Layer Security (TLS) Extensions", RFC 3546, June 2003.

[DTLS] Rescorla, E., Modadugu, N., "Datagram Transport Layer Security", draft-rescorla-dtls-05.txt, June 2004.

[TLSv1.1] Dierks, T., Rescorla, E., "The TLS Protocol Version 1.1", draft-ietf-tls-rfc2246-bis-13.txt, June 2005

[SMTPTLS] Hoffman, P., "SMTP Service Extension for Secure SMTP over TLS", RFC 2487, January 1999.

[HTTPTLS] Rescorla, E., "HTTP Over TLS", RFC 2818, May 2000.

[POPTLS] Newman, C., "Using TLS with IMAP, POP3 and ACAP", RFC 2595, June 1999.

Author's Addresses

Mohamad Badra

ENST Paris
France Email: Mohamad.Badra@enst.fr

Ibrahim Hajjeh
ESRGroups, Security WG
France Email: Ibrahim.Hajjeh@esrgroups.org

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.