Network Working Group
Internet-Draft
Expires: August 29, 2006

E. Rescorla
Network Resonance
February 25, 2006

Transport Layer Security (TLS) Partial Encryption Mode
draft-rescorla-tls-partial-00.txt

Status of this Memo

Copyright Notice

Abstract

This document describes an extension to TLS to allow partial encryption of record bodies. This allows the beginning of the record body to be in the clear, thus facilitating debugging and header compression.

Table of Contents

1. Introduction

Encryption in Transport Layer Security (TLS) [2] is currently an all-or-nothing proposition. The choices are a cipher suite that has encryption or one of the NULL cipher suites which offer no encryption. This has disadvantages in settings where the application layer itself has some data (such as a header) that it wishes to have in the clear (e.g., for debugging purposes) and some data (such as a payload) that it wishes to have encrypted. This document describes an extension to TLS that allows for the initial portion of the record to remain uncompressed and unencrypted.

2. Conventions Used In This Document

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 [1].

3. Negotiating the Partial Encryption Extension

The client requests support for the partial encryption feature by sending the "partial_encryption" extension in its ClientHello. The "extension_data" field contains a PartialEncryption field:

```
struct {
  uint16 InitialClearBytes;
} PartialEncryption
```

The InitialClearBytes value contains the number of bytes which will be in the clear for each application_data record. This value will obtain for the entire life of this association.

The server indicates support for the partial encryption feature sending a "partial_encryption" extension with an empty "extension_data" field. This indicates its acceptance of the extension and of the the number of bytes to be sent in the clear. If the server does not support the extension or does not accept the InitialClearBytes value, it MUST ignore the extension. The first application_data record in the new association (after the change_cipher_spec message) MUST use the new encryption mode as described below.

4. Record Processing

The partial encryption extension only matters for records of type

"application_data". All other records should be processed via the usual TLS/DTLS rules.

4.1. Record Transmission

When the partial encryption extension is in effect, the TLSCiphertext.fragment struct becomes:

```
    select (CipherSpec.cipher_type) {
            opaque plaintext_bytes[InitialClearBytes];  // New field

            case stream: GenericStreamCipher;
            case block:  GenericBlockCipher;
    } fragment;
```

The first InitialClearBytes bytes of the TLSPlaintext.fragment are inserted in the TLSCiphertext.plaintext_bytes value. The rest are passed through compression and encryption to form the GenericStreamCipher or GenericBlockCipher values. If the TLSPlaintext.fragment is less than InitialClearBytes then the entire plaintext is left un-encrypted. The same processing applies to DTLS [4].

The TLS MAC remains unchanged and is applied to both the plaintext_bytes and the TLSCompressed.fragment. Where length is computed as InitialClearBytes + TLSCompressed.length.

4.2. Record Reception

Record reception is relatively simple. The receiver knows whether the partial_plaintext extension is in effect and simply treats the first InitialClearBytes of what would otherwise be the ciphertext as plaintext. After those bytes are removed, the rest of the record can be processed as usual.

5. Security Considerations

There are two security concerns introduced by these extensions. The first involves the security of the negotiation and the second the security of the transport protocol. Because the negotiation is protected by the TLS/DTLS handshake, attackers can neither force the use of these extensions nor block them while allowing the negotiation to succeed.

The second concern is the security if the data. Obviously, no confidentiality is provided for any data in the initial plaintext. However, because the length of the initial plaintext is fixed in the

negotiation and the MAC covers the total length, an active attacker cannot convince the receiver to accept values which are encrypted as if they were plaintext or vice versa.

One concern that applies solely to DTLS is that an active attacker might manipulate MTU values to attempt to force the sender to split data across multiple records and thus have some application layer data which would otherwise be encrypted sent in the clear. DTLS itself does not do any fragmentation and applications which use this extension MUST NOT fragment the data that they send to DTLS in such a way that sensitive data could be transmitted unencrypted.

6. IANA Considerations

This document defines an extension to TLS, in accordance with [3]:

```
enum { partial_encryption (??) } ExtensionType;
```

[[NOTE: These values need to be assigned by IANA]]

7. Normative References

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

[2] Dierks, T. and E. Rescorla, "The TLS Protocol Version 1.1", draft-ietf-tls-rfc2246-bis-13 (work in progress), June 2005.

[3] Blake-Wilson, S., "Transport Layer Security (TLS) Extensions", draft-ietf-tls-rfc3546bis-02 (work in progress), October 2005.

[4] Rescorla, E. and N. Modadugu, "Datagram Transport Layer Security", draft-rescorla-dtls-05 (work in progress), June 2005.

Author's Address

Eric Rescorla
Network Resonance
2483 E. Bayshore Rd., #212
Palo Alto, CA 94303
USA

Email: ekr@networkresonance.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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