Network Working Group
Request for Comments: 4642
Category: Standards Track

K. Murchison
Carnegie Mellon University
J. Vinocur
Cornell University
C. Newman
Sun Microsystems

October 2006

Using Transport Layer Security (TLS) with Network News Transfer Protocol (NNTP)

Status of This Memo

This document specifies an Internet standards track protocol for the Internet community, and requests discussion and suggestions for improvements. Please refer to the current edition of the "Internet Official Protocol Standards" (STD 1) for the standardization state and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This memo defines an extension to the Network News Transfer Protocol (NNTP) that allows an NNTP client and server to use Transport Layer Security (TLS). The primary goal is to provide encryption for single-link confidentiality purposes, but data integrity, (optional) certificate-based peer entity authentication, and (optional) data compression are also possible.

Table of Contents

1. Introduction 2
   1.1. Conventions Used in This Document 3
2. The STARTTLS Extension 3
   2.1. Advertising the STARTTLS Extension 3
   2.2. STARTTLS Command 4
        2.2.1. Usage 4
        2.2.2. Description 4
        2.2.3. Examples 6
3. Augmented BNF Syntax for the STARTTLS Extension 8
   3.1. Commands 8
   3.2. Capability entries 8
4. Summary of Response Codes 8
5. Security Considerations 8
6. IANA Considerations 11
7. References 12
   7.1. Normative References 12
   7.2. Informative References 12
8. Acknowledgements 12

1. Introduction

Historically, unencrypted NNTP [NNTP] connections were satisfactory for most purposes. However, sending passwords unencrypted over the network is no longer appropriate, and sometimes integrity and/or confidentiality protection are desired for the entire connection.

The TLS protocol (formerly known as SSL) provides a way to secure an application protocol from tampering and eavesdropping. Although advanced SASL authentication mechanisms [NNTP-AUTH] can provide a lightweight version of this service, TLS is complimentary to both simple authentication-only SASL mechanisms and deployed clear-text password login commands.

In some existing implementations, TCP port 563 has been dedicated to NNTP over TLS. These implementations begin the TLS negotiation immediately upon connection and then continue with the initial steps of an NNTP session. This use of TLS on a separate port is discouraged for the reasons documented in Section 7 of "Using TLS with IMAP, POP3 and ACAP" [TLS-IMAPPOP].

This specification formalizes the STARTTLS command already in occasional use by the installed base. The STARTTLS command rectifies a number of the problems with using a separate port for a "secure" protocol variant; it is the preferred way of using TLS with NNTP.

1.1. Conventions Used in This Document

The notational conventions used in this document are the same as those in [NNTP], and any term not defined in this document has the same meaning as in that one.

The key words "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [KEYWORDS].

In the examples, commands from the client are indicated with [C], and responses from the server are indicated with [S].

2. The STARTTLS Extension

This extension provides a new STARTTLS command and has the capability label STARTTLS.

2.1. Advertising the STARTTLS Extension

A server supporting the STARTTLS command as defined in this document will advertise the "STARTTLS" capability label in response to the CAPABILITIES command ([NNTP] Section 5.2). However, this capability MUST NOT be advertised once a TLS layer is active (see Section 2.2.2) or after successful authentication [NNTP-AUTH]. This capability MAY be advertised both before and after any use of the MODE READER command ([NNTP] Section 5.3), with the same semantics.

As the STARTTLS command is related to security, cached results of CAPABILITIES from a previous session MUST NOT be relied on, as per Section 12.6 of [NNTP].

Example:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] IHAVE
[S] STARTTLS
[S] LIST ACTIVE NEWSGROUPS
[S] .
```

2.2. STARTTLS Command

2.2.1. Usage

This command MUST NOT be pipelined.

Syntax
STARTTLS

Responses

382 Continue with TLS negotiation
502 Command unavailable [1]
580 Can not initiate TLS negotiation

[1] If a TLS layer is already active, or if authentication has occurred, STARTTLS is not a valid command (see Section 2.2.2).

NOTE: Notwithstanding Section 3.2.1 of [NNTP], the server MUST NOT return either 480 or 483 in response to STARTTLS.

2.2.2. Description

A client issues the STARTTLS command to request negotiation of TLS. The STARTTLS command is usually used to initiate session security, although it can also be used for client and/or server certificate authentication and/or data compression.

An NNTP server returns the 483 response to indicate that a secure or encrypted connection is required for the command sent by the client. Use of the STARTTLS command as described below is one way to establish a connection with these properties. The client MAY therefore use the STARTTLS command after receiving a 483 response.

If a server advertises the STARTTLS capability, a client MAY attempt to use the STARTTLS command at any time during a session to negotiate TLS without having received a 483 response. Servers SHOULD accept such unsolicited TLS negotiation requests.

If the server is unable to initiate the TLS negotiation for any reason (e.g., a server configuration or resource problem), the server MUST reject the STARTTLS command with a 580 response. Then, it SHOULD either reject subsequent restricted NNTP commands from the client with a 483 response code (possibly with a text string such as "Command refused due to lack of security") or reject a subsequent restricted command with a 400 response code (possibly with a text string such as "Connection closing due to lack of security") and close the connection. Otherwise, the server issues a 382 response,

and TLS negotiation begins. A server MUST NOT under any circumstances reply to a STARTTLS command with either a 480 or 483 response.

If the client receives a failure response to STARTTLS, the client must decide whether or not to continue the NNTP session. Such a decision is based on local policy. For instance, if TLS was being used for client authentication, the client might try to continue the session in case the server allows it to do so even with no authentication. However, if TLS was being negotiated for encryption, a client that gets a failure response needs to decide whether to continue without TLS encryption, to wait and try again later, or to give up and notify the user of the error.

Upon receiving a 382 response to a STARTTLS command, the client MUST start the TLS negotiation before giving any other NNTP commands. The TLS negotiation begins for both the client and server with the first octet following the CRLF of the 382 response. If, after having issued the STARTTLS command, the client finds out that some failure prevents it from actually starting a TLS handshake, then it SHOULD immediately close the connection.

Servers MUST be able to understand backwards-compatible TLS Client Hello messages (provided that client_version is TLS 1.0 or later), and clients MAY use backwards-compatible Client Hello messages. Neither clients nor servers are required to actually support Client Hello messages for anything other than TLS 1.0. However, the TLS extension for Server Name Indication ("server_name") [TLS-EXT] SHOULD be implemented by all clients; it also SHOULD be implemented by any server implementing STARTTLS that is known by multiple names. (Otherwise, it is not possible for a server with several hostnames to present the correct certificate to the client.)

If the TLS negotiation fails, both client and server SHOULD immediately close the connection. Note that while continuing the NNTP session is theoretically possible, in practice a TLS negotiation failure often leaves the session in an indeterminate state; therefore, interoperability can not be guaranteed.

Upon successful completion of the TLS handshake, the NNTP protocol is reset to the state immediately after the initial greeting response (see 5.1 of [NNTP]) has been sent, with the exception that if a MODE READER command has been issued, its effects (if any) are not reversed. At this point, as no greeting is sent, the next step is for the client to send a command. The server MUST discard any knowledge obtained from the client, such as the current newsgroup and article number, that was not obtained from the TLS negotiation itself. Likewise, the client SHOULD discard and MUST NOT rely on any

knowledge obtained from the server, such as the capability list, which was not obtained from the TLS negotiation itself.

The server remains in the non-authenticated state, even if client credentials are supplied during the TLS negotiation. The AUTHINFO SASL command [NNTP-AUTH] with the EXTERNAL mechanism [SASL] MAY be used to authenticate once TLS client credentials are successfully exchanged, but servers supporting the STARTTLS command are not required to support AUTHINFO in general or the EXTERNAL mechanism in particular. The server MAY use information from the client certificate for identification of connections or posted articles (either in its logs or directly in posted articles).

Both the client and the server MUST know if there is a TLS session active. A client MUST NOT attempt to start a TLS session if a TLS session is already active. A server MUST NOT return the STARTTLS capability label in response to a CAPABILITIES command received after a TLS handshake has completed, and a server MUST respond with a 502 response code if a STARTTLS command is received while a TLS session is already active. Additionally, the client MUST NOT issue a MODE READER command while a TLS session is active, and a server MUST NOT advertise the MODE-READER capability.

The capability list returned in response to a CAPABILITIES command received after a successful TLS handshake MAY be different from the list returned before the TLS handshake. For example, an NNTP server supporting SASL [NNTP-AUTH] might not want to advertise support for a particular mechanism unless a client has sent an appropriate client certificate during a TLS handshake.

2.2.3. Examples

Example of a client being prompted to use encryption and negotiating it successfully (showing the removal of STARTTLS from the capability list once a TLS layer is active), followed by a successful selection of the group and an (inappropriate) attempt by the client to initiate another TLS negotiation:

```
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] STARTTLS
[S] LIST ACTIVE NEWSGROUPS OVERVIEW.FMT
[S] OVER
[S] .
[C] GROUP local.confidential
[S] 483 Encryption or stronger authentication required
```

```
[C] STARTTLS
[S] 382 Continue with TLS negotiation
[TLS negotiation occurs here]
[Following successful negotiation, traffic is protected by TLS]
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] LIST ACTIVE NEWSGROUPS OVERVIEW.FMT
[S] OVER
[S] .
[C] GROUP local.confidential
[S] 211 1234 3000234 3002322 local.confidential
[C] STARTTLS
[S] 502 STARTTLS not allowed with active TLS layer
```

Example of a request to begin TLS negotiation declined by the server:

```
[C] STARTTLS
[S] 580 Can not initiate TLS negotiation
```

Example of a failed attempt to negotiate TLS, followed by two attempts at selecting groups only available under a security layer (in the first case, the server allows the session to continue; in the second, it closes the connection). Note that unrestricted commands such as CAPABILITIES are unaffected by the failure:

```
[C] STARTTLS
[S] 382 Continue with TLS negotiation
[TLS negotiation is attempted here]
[Following failed negotiation, traffic resumes without TLS]
[C] CAPABILITIES
[S] 101 Capability list:
[S] VERSION 2
[S] READER
[S] STARTTLS
[S] LIST ACTIVE NEWSGROUPS OVERVIEW.FMT
[S] OVER
[S] .
[C] GROUP local.confidential
[S] 483 Encryption or stronger authentication required
[C] GROUP local.private
[S] 400 Closing connection due to lack of security
```

3. Augmented BNF Syntax for the STARTTLS Extension

This section describes the formal syntax of the STARTTLS extension using ABNF [ABNF]. It extends the syntax in Section 9 of [NNTP], and non-terminals not defined in this document are defined there. The [NNTP] ABNF should be imported first before attempting to validate these rules.

3.1. Commands

This syntax extends the non-terminal "command", which represents an NNTP command.

```
command =/ starttls-command

starttls-command = "STARTTLS"
```

3.2. Capability entries

This syntax extends the non-terminal "capability-entry", which represents a capability that may be advertised by the server.

```
capability-entry =/ starttls-capability

starttls-capability = "STARTTLS"
```

4. Summary of Response Codes

This section contains a list of each new response code defined in this document and indicates whether it is multi-line, which commands can generate it, what arguments it has, and what its meaning is.

Response code 382
   Generated by: STARTTLS
   Meaning: continue with TLS negotiation

Response code 580
   Generated by: STARTTLS
   Meaning: can not initiate TLS negotiation

5. Security Considerations

Security issues are discussed throughout this memo.

In general, the security considerations of the TLS protocol [TLS] and any implemented extensions [TLS-EXT] are applicable here; only the most important are highlighted specifically below. Also, this extension is not intended to cure the security considerations

described in Section 12 of [NNTP]; those considerations remain relevant to any NNTP implementation.

NNTP client and server implementations MUST implement the TLS_RSA_WITH_RC4_128_MD5 [TLS] cipher suite and SHOULD implement the TLS_DHE_DSS_WITH_3DES_EDE_CBC_SHA [TLS] cipher suite. This is important, as it assures that any two compliant implementations can be configured to interoperate. All other cipher suites are OPTIONAL.

Before the TLS handshake has begun, any protocol interactions are performed in the clear and may be modified by an active attacker. For this reason, clients and servers MUST discard any sensitive knowledge obtained prior to the start of the TLS handshake upon the establishment of a security layer. Furthermore, the CAPABILITIES command SHOULD be re-issued upon the establishment of a security layer, and other protocol state SHOULD be re-negotiated as well.

Note that NNTP is not an end-to-end mechanism. Thus, if an NNTP client/server pair decide to add TLS confidentiality, they are securing the transport only for that link. Similarly, because delivery of a single Netnews article may go between more than two NNTP servers, adding TLS confidentiality to one pair of servers does not mean that the entire NNTP chain has been made private. Furthermore, just because an NNTP server can authenticate an NNTP client, it does not mean that the articles from the NNTP client were authenticated by the NNTP client when the client itself received them (prior to forwarding them to the server).

During the TLS negotiation, the client MUST check its understanding of the server hostname against the server's identity as presented in the server Certificate message, in order to prevent man-in-the-middle attacks. Matching is performed according to these rules:

- The client MUST use the server hostname it used to open the connection (or the hostname specified in TLS "server_name" extension [TLS-EXT]) as the value to compare against the server name as expressed in the server certificate. The client MUST NOT use any form of the server hostname derived from an insecure remote source (e.g., insecure DNS lookup). CNAME canonicalization is not done.

- If a subjectAltName extension of type dNSName is present in the certificate, it SHOULD be used as the source of the server's identity.

- Matching is case-insensitive.

- A "*" wildcard character MAY be used as the left-most name component in the certificate. For example, *.example.com would match a.example.com, foo.example.com, etc., but would not match example.com.

- If the certificate contains multiple names (e.g., more than one dNSName field), then a match with any one of the fields is considered acceptable.

If the match fails, the client SHOULD either ask for explicit user confirmation or terminate the connection with a QUIT command and indicate the server's identity is suspect.

Additionally, clients MUST verify the binding between the identity of the servers to which they connect and the public keys presented by those servers. Clients SHOULD implement the algorithm in Section 6 of [PKI-CERT] for general certificate validation, but MAY supplement that algorithm with other validation methods that achieve equivalent levels of verification (such as comparing the server certificate against a local store of already-verified certificates and identity bindings).

A man-in-the-middle attack can be launched by deleting the STARTTLS capability label in the CAPABILITIES response from the server. This would cause the client not to try to start a TLS session. Another man-in-the-middle attack would allow the server to announce its STARTTLS capability, but alter the client's request to start TLS and the server's response. An NNTP client can partially protect against these attacks by recording the fact that a particular NNTP server offers TLS during one session and generating an alarm if it does not appear in the CAPABILITIES response for a later session. (Of course, the STARTTLS capability would not be listed after a security layer is in place.)

If the client receives a 483 or 580 response, the client has to decide what to do next. The client has to choose among three main options: to go ahead with the rest of the NNTP session, to (re)try TLS later in the session, or to give up and postpone newsreading/transport activity. If an error occurs, the client can assume that the server may be able to negotiate TLS in the future and should try to negotiate TLS in a later session. However, if the client and server were only using TLS for authentication and no previous 480 response was received, the client may want to proceed with the NNTP session, in case some of the operations the client wanted to perform are accepted by the server even if the client is unauthenticated.

6. IANA Considerations

This section gives a formal definition of the STARTTLS extension as required by Section 3.3.3 of [NNTP] for the IANA registry.

- The STARTTLS extension provides connection-based security via the Transport Layer Security (TLS).

- The capability label for this extension is "STARTTLS".

- The capability label has no arguments.

- This extension defines one new command, STARTTLS, whose behavior, arguments, and responses are defined in Section 2.2.

- This extension does not associate any new responses with pre-existing NNTP commands.

- This extension does affect the overall behavior of both server and client, in that after successful use of the STARTTLS command, all communication is transmitted with the TLS protocol as an intermediary.

- This extension does not affect the maximum length of commands or initial response lines.

- This extension does not alter pipelining, but the STARTTLS command cannot be pipelined.

- Use of this extension does alter the capabilities list; once the STARTTLS command has been used successfully, the STARTTLS capability can no longer be advertised by CAPABILITIES.

  Additionally, the MODE-READER capability MUST NOT be advertised after a successful TLS negotiation.

- This extension does not cause any pre-existing command to produce a 401, 480, or 483 response.

- This extension is unaffected by any use of the MODE READER command, however the MODE READER command MUST NOT be used in the same session following a successful TLS negotiation.

- Published Specification: This document.

- Contact for Further Information: Authors of this document.

- Change Controller: IESG <iesg@ietf.org>.

7. References

7.1. Normative References

[ABNF] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 4234, October 2005.

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

[NNTP] Feather, C., "Network News Transfer Protocol (NNTP)", RFC 3977, October 2006.

[PKI-CERT] Housley, R., Polk, W., Ford, W., and D. Solo, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 3280, April 2002.

[TLS] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1", RFC 4346, April 2006.

[TLS-EXT] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J., and T. Wright, "Transport Layer Security (TLS) Extensions", RFC 4366, April 2006.

7.2. Informative References

[NNTP-AUTH] Vinocur, J., Murchison, K., and C. Newman, "Network News Transfer Protocol (NNTP) Extension for Authentication", RFC 4643, October 2006.

[SASL] Melninov, A., Ed. and K. Zeilenga, Ed, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

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

8. Acknowledgements

A significant amount of the text in this document was lifted from RFC 2595 by Chris Newman and RFC 3207 by Paul Hoffman.

Special acknowledgement goes also to the people who commented privately on intermediate revisions of this document, as well as the members of the IETF NNTP Working Group for continual insight in discussion.

Authors' Addresses

Kenneth Murchison
Carnegie Mellon University
5000 Forbes Avenue
Cyert Hall 285
Pittsburgh, PA 15213 USA

EMail: murch@andrew.cmu.edu

Jeffrey M. Vinocur
Department of Computer Science
Upson Hall
Cornell University
Ithaca, NY 14853

EMail: vinocur@cs.cornell.edu

Chris Newman
Sun Microsystems
3401 Centrelake Dr., Suite 410
Ontario, CA 91761

EMail: Chris.Newman@sun.com

Full Copyright Statement

Copyright (C) The Internet Society (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).