Internet Engineering Task Force   Michael Boe
INTERNET-DRAFT draft-ietf-tn3270e-telnet-tls-06.txt   Jeffrey Altman
                                                             April 2002

TLS-based Telnet Security

Status of this memo

Copyright Notice

Abstract

Telnet service has long been a standard Internet protocol. However, a standard way of ensuring privacy and integrity of Telnet sessions has been lacking. This document proposes a standard method for Telnet servers and clients to use the Transport Layer Security (TLS) protocol. It describes how two Telnet participants can decide whether or not to attempt TLS negotiation, and how the two participants should process authentication credentials exchanged as a part of TLS startup.

Changes since -05 Draft

None. Republication for IETF Last Call

Changes since -04 Draft

Incorporated changes submitted by Eric Rescorla and Russ Housley. Mostly minor in nature except for an expansion of the text in section 4.1.1 on how certificate verification should be performed. This section was taken almost verbatim from RFC 2818.

Minor changes include:

. change MUST to SHOULD in section 3.2

. clarification to reference to Kerberos cipher suites in 4.1.1

. expansion of discussion on mutual authentication via Telnet AUTH in section 4.1.3.

. both client and server should shutdown connection upon error detection in section 4.1.3

. correction to name of PKIX Working Group in section 5.1

. expansion of discussion of finished message discussion in 5.2.

. TLS 3.1 replaced by TLS 1.0 in section 6.1

. removal of "dangerous to encrypt twice" comment from 7.0

. expansion of references.

Changes since -03 Draft

Major changes to sections describing authentication of clients and servers.

Changes since -02 Draft

o Clarify server actions in response to client initiating the TLS negotiation.

o Replace the parochial term "mainframe" with "application-server."

o Nuke explicit references to RFC1416, since there's a new RFC in the works for this and the reference isn't normative anyway.

o Use dNSName language similar to that used in the most recent HTTP TLS draft.

o Delete beginning paragraph describing server-authentication in TLS. Unclear and possibly wrong.

o Delete explicit references to SASL, since we don't actually describe ways of using SASL.

o Add section describing interaction between AUTH and STARTTLS option negotiations.

Changes since -01 Draft

o Drop possibility of a continuing a Telnet session if the TLS negotiation fails.

- Assert that server sending DO STARTTLS must be willing to negotiate a TLS session

- Change SHOULD to MUST with respect to a server requesting a client certificate.

- Add paragraph on commonName to section on check of X.509 certificate.

- Sharpen language concerning notification of possible server-certificate mismatch.

- drop place-holder section on Kerberos 5 security; replace with section on non-PKI-based authentication (after TLS negotiation).

- Prohibit fallback to SSL 2.0.

- Give more details about how authentication-after-TLS-negotiation can be achieved.

- Simplify post-TLS Telnet negotiation state-assumptions by resetting them to initial-state.

Changes since -00 Draft

- Add local authentication/authorization operational model.

- Change behavior of Telnet machine to reset at start of TLS negotiation.

- Insert narrative questioning the utility of allowing continuation of Telnet session after TLS has ended.

- change examples to reflect the above changes.

- Fix several typos.

Contents

1 Introduction

This document describes the START_TLS Telnet option. It allows TLS to be activated at the beginning of a Telnet connection to provide authentication and confidentiality of the Telnet session. This document also defines a set of advisory security policy response codes for use when negotiating TLS from within Telnet.

We are interested in addressing the interaction between the Telnet client and server that will support this secure requirement with the knowledge that this is only a portion of the total end-user to application path. Specifically, it is often true that the Telnet server does not reside on the target machine (it does not have access to a list of identities which are allowed to access to that application-server), and it is often true (e.g. 3270 access) that the telnet server can not even identify that portion of the emulation stream which contains user identification/password information. Additionally, it may be the case that the Telnet client is not co-resident with the end user and that it also may be unable to identify that portion of the data stream that deals with user identity. We make the assumption here that there is a trust relationship and appropriate protection to support that relationship between the Telnet Server and the ultimate application engine such that data on this path is protected and that the application will authenticate the end user via the emulation stream as well as use this to control access to information. We further make the assumption that the path between the end user and the client is protected.

To hold up the Telnet part of the overall secure path between the user and the application-server, the Telnet data stream must appear unintelligible to a third party. This is done by creating a shared secret between the client and server. This shared secret is used to encrypt the flow of data and (just as important) require the client to verify that it is talking to correct server (the one that the application-server trusts rather than an unintended man-in-the-middle) with the knowledge that the emulation stream itself will be used by the application-server to verify the identity of the end-user. Rather than create a specialized new protocol which accomplishes these goals we instead have chosen to use an existing IETF protocol, Transport Layer Security (TLS) (formerly known as Secure Sockets Layer (SSL)).

The Telnet [TELNET] application protocol can certainly benefit from the use of TLS. Since 1992 Telnet has supported over a dozen forms of end user authentication and DES encryption via the AUTH and ENCRYPT options. Since 1995, TLS (as SSL) has been used to provide privacy and integrity protection to Telnet data streams via Telnet AUTH and via dedicated IANA assigned port numbers (telnets 992). TLS offers a broad range of security levels that allow sites to proceed at an "evolutionary" pace in deploying authentication, authorization and confidentiality policies, databases and distribution methods.

This document describes how TLS can be used to provide the following:

- o creation and refresh of a shared secret;
- o negotiation and execution of data encryption and optional compressesion;
- o primary negotiation of authentication; and, if chosen
- o execution of public-key or symmetric-key based authentication.

TLS at most offers only authentication of the peers conducting the TLS dialog. In particular, it does not offer the possibility of the client providing separate credentials for authorization than were presented for authentication. After the establishment of peer to peer trust based on TLS, other forms of end user authentication including Telnet AUTH may be used to provide credentials for use in determining end user authorization.

Traditional Telnet servers have operated without such early presentation of authorization credentials for many reasons (most of which are historical). However, recent developments in Telnet server technology make it advantageous for the Telnet server to know the authorized capabilities of the remote client before choosing a communications link (be it `pty' or SNA LU) and link-characteristics to the host system (be that "upstream" link local or remote to the server). Thus, we expect to see the use of client authorization to become an important element of the Telnet evolution. Such authorization methods may require certificates presented by the client via TLS, or by the use of Telnet AUTH option, or some other as yet unstandardized method.

This document describes the START_TLS telnet option which allows TLS to be activated at the beginning of a Telnet connection using the standard "telnet" port (IANA tcp\23). It also defines a set of advisory security-policy response codes for use when negotiating TLS over Telnet.

Conventions Used in this Document
The key words "REQUIRED", "MUST", "MUST NOT", "SHOULD", "SHOULD NOT" and "MAY" in this document are to be interpreted as described in [KEYWORDS].

Formal syntax is defined using ABNF [ABNF].

In examples, "C:" and "S:" indicate lines sent by the the client and server, respectively.

2 Command Names and Codes (assigned by IANA)

START_TLS	46 (decimal)
FOLLOWS	1 (decimal)

3 Command Meanings

This document makes reference to a "server" and a "client". For the purposes of this document, the "server" is the side of the connection that did the passive TCP open (TCP LISTEN state), and the "client" is the side of the connection that did the active open.

IAC DONT START_TLS

The sender is either not a server or is not interested in negotiating a TLS connection.

IAC WONT START_TLS

The sender is either not a client or is not interested in negotiating a TLS connection.

IAC DO START_TLS

The server side of the connection sends this command to indicate a desire to negotiate a TLS connection. This command MUST NOT be sent by the client and if received by the server MUST be refused with IAC WONT START_TLS.

IAC WILL START_TLS

The client side of the connection sends this command to indicate a desire to negotiate a TLS connection. This command MUST NOT be sent by the server and if received by the client MUST be refused with IAC DONT START_TLS.

IAC SB START_TLS FOLLOWS IAC SE

The FOLLOWS sub-command when sent indicates that the next byte of data received after this command MUST be a TLS negotiation as described in [TLS]. This sub-command is sent by both the client and the server. After this sub-command has been sent, the sender MUST NOT respond to nor initiate any additional telnet commands or sub-commands. When this sub-command has been sent and received the TLS negotiation will commence. When sent by a client this sub-command will be followed by a TLS ClientHello. When sent by a server this sub-command will be followed by a TLS ServerHello.

3.1 Usage of commands and interactions with other Telnet options

The START_TLS option is an asymmetric option, with the server side allowed to send IAC DO START_TLS and the client allowed to send IAC WILL START_TLS. Sub-commands are used to synchronize the link in preparation for negotiating TLS. This synchronization takes the form of a three-way handshake:

1. As per normal Telnet option processing rules, the client MUST respond to the server's IAC DO START_TLS with either IAC WONT START_TLS or IAC WILL START_TLS (if it hasn't already done so). Once the client has sent IAC WILL START_TLS and received IAC DO START_TLS, it MUST immediately send a FOLLOWS sub-command (IAC SB START_TLS FOLLOWS IAC SE) to indicate it is ready to begin a TLS negotiation. Once the FOLLOWS sub-command has been sent, the client MUST ignore all telnet negotiations except for the FOLLOWS sub-command. When the FOLLOWS sub-command has been received the client MUST halt use of the telnet protocol, reset the telnet state machine and begin a TLS negotiation by sending a ClientHello message.

2. If the client initiates by sending IAC WILL START_TLS, the server MUST respond with either IAC DO START_TLS or IAC DONT START_TLS.

3. The server SHOULD NOT send additional Telnet data or commands after sending IAC DO START_TLS except in response to client Telnet options received until after it receives either a negative response from the client (IAC WONT START_TLS) or a successful negotiation of TLS has occurred. If the client's START_TLS option response is negative, the server is free to send additional Telnet data or commands. If the client's response is affirmative (IAC WILL START_TLS), then the server MUST send the FOLLOWS sub-command (IAC SB START_TLS FOLLOWS IAC SE) and await the FOLLOWS sub-command from the client. When the FOLLOWS sub-command has been sent and received the server MUST halt use of the telnet protocol, reset the telnet state machine, and begin a TLS negotiation by sending a TLS ServerHello message.

4. If both START_TLS and AUTH [AUTH] are offered, START_TLS SHOULD be sent first and MUST take precedence if both are agreed to. AUTH MAY be renegotiated after successful establishment of the TLS session if end-user authentication via a supported method is desired.

5. If a TLS session has been established, the ENCRYPT [ENCRYPT] option MUST NOT be negotiated in either direction.

6. When the FOLLOWS sub-command has been sent and received the Telnet state machine is reset. This means that the state of all telnet options is reset to the WONT/DONT state and any data received via subcommands is forgotten. After a sucessful TLS negotiation the Telnet negotiations will be restarted as if a new connection had just been established with one exception. Since TLS is already in use, the START_TLS option MUST NOT be negotiated.

3.2 TLS Negotiation Failure

The behavior regarding TLS negotiation failure is covered in [TLS], and does not indicate that the TCP connection be broken; the semantics are that TLS is finished and all state variables cleaned up. The TCP connection may be retained.

However, it's not clear that either side can detect when the last of the TLS data has arrived. So if TLS negotiation fails, the TCP connection SHOULD be reset and the client MAY reconnect. To avoid infinite loops of TLS negotiation failures, the client MUST remember to not negotiate START_TLS if reconnecting due to a TLS negotiation failure.

4 Telnet Authentication and Authorization

Telnet servers and clients can be implemented in a variety of ways that impact how clients and servers authenticate and authorize each other. However, most (if not all) the implementations can be abstracted via the following four communicating processes:

SES	Server End System. This is an application or machine to which client desires a connection. Though not illustrated here, a single Telnet connection between client and server could have multiple SES terminations.
Server	The Telnet server.
Client	The Telnet client, which may or may not be co-located with the CES. The Telnet client in fact be a gateway or proxy for downstream clients; it's immaterial.
CES	Client End System. The system communicating with the Telnet Client. There may be more than one actual CES communicating to a single Telnet Client instance; this is also immaterial to how Client and Server can sucessfully exchange authentication and authorization details. However, see Section 5.4 for a discussion on trust implications.

What is of interest here is how the Client and Server can exchange authentication and authorization details such that these components can direct Telnet session traffic to authorized End Systems in a reliable, trustworthy fashion.

What is beyond the scope of this specification are several related topics, including:

- How the Server and SES are connected, and how they exchange data or information regarding authorization or authentication (if any).

- How the Client and CES are connected, and how they exchange data or information regarding authorization or authentication (if any).

System-to-system communications using the Telnet protocol have traditionally used no authentication techniques at the Telnet level. More recent techniques have used Telnet to transport authentication exchanges (RFC 2941). In none of these systems, however, is a remote system allowed to assume more than one identity once the Telnet preamble negotiation is over and the remote is connected to the application-endpoint. The reason for this is that the local party must in some way inform the end-system of the remote party's identity (and perhaps authorization). This process must take place before the remote party starts communicating with the end-system. At that point it's too late to change what access a client may have to an server end-system: that end-system has been selected, resources have been allocated and capability restrictions set.

This process of authentication, authorization and resource allocation can be modeled by the following simple set of states and transitions:

`unauthenticated'   The local party has not received any credentials offered by the remote. A new Telnet connection starts in this state.

The `authenticating' state will be entered from this state if the local party initiates the authentication process of the peer. The Telnet START_TLS negotiation is considered an initiation of the authentication process.

The `authorizing' state will be entered from this state either if the local party decides to begin authorization and resource allocation procedures unilaterally...or if the local party has received data from the remote party destined for local end-system.

`authenticating'   The local party has received at least some of the credentials needed to authenticate its peer, but has not finished the process.

The `authenticated' state will be entered from this state if the local party is satisfied with the credentials proferred by the client.

The `unauthenticated' state will be entered from this state if the local party cannot verify the credentials proffered by the client or if the client has not proffered any credentials. Alternately, the local party may terminate the Telnet connection instead of returning it to the `unauthenticated' state.

`authenticated'   The local party has authenticated its peer, but has not yet authorized the client to connect to any end-system resources.

The `authenticating' state will be entered from this state if the local party decides that further authentication of the client is warranted.

The `authorizing' state will be entered from this state if the local party either initiates authorization dialog with the client (or engages in some process to authorize and allocate resources on behalf of the client), or has received data from the remote party destined for a local end-system.

`authorizing'   The local party is in the process of authorizing its peer to use end-system resources, or may be in the process of allocating or reserving those resources.

The `transfer-ready' state will be entered when the local party is ready to allow data to be passed between the local end-system and remote peer.

The `authenticated' state will be entered if the local party determines that the current authorization does not allow any access to a local end-system. If the remote peer is not currently authenticated, then the `unauthenticated' state will be entered instead.

`transfer-ready'   The party may pass data between the local end-system to its peer.

The `authorizing' state will be entered if the local party (perhaps due to a request by the remote peer) deallocates the communications resources to the local-end system. Alternately, the local party may enter the `authenticated' or the `unauthenticated' state.

In addition to the "orderly" state transitions noted above, some extraordinary transitions may also occur:

1. The absence of a guarantee on the integrity of the data stream between the two Telnet parties also removes the guarantee that the remote peer is who the authentication credentials say the peer is. Thus, upon being notified that the Telnet session is no longer using an integrity layer, the local party must at least deallocate all resources associated with a Telnet connection which would not have been allocable had the remote party never authenticated itself.

   In practice, this deallocation-of-resources restriction is hard to interpret consistently by both Telnet endpoints. Therefore, both parties MUST return to the initial Telnet state after negotiation of TLS. That is, it is as if the Telnet session had just started.

   This means that the states may transition from whatever the current state is to `unauthenticated'. Alternately, the local party may break the Telnet connection instead.

2. If the local party is notified at any point during the Telnet connection that the remote party's authorizations have been reduced or revoked, then the local party must treat the remote party as being unauthenticated. The local party must deallocate all resources associated with a Telnet connection which would not have been allocable had the remote party never authenticated itself.

This too may mean that the states may transition from whatever the current state is to `unauthenticated'. Alternately, the local party may break the Telnet connection instead.

The above model explains how each party should handle the authentication and authorization information exchanged during the lifetime of a Telnet connection. It is deliberately fuzzy as to what constitutes internal processes (such as "authorizing") and what is meant by "resources" or "end-system" (such as whether an end-system is strictly a single entity and communications path to the local party, or multiples of each, etc).

Here's a state transition diagram, as per [RFC2360]:

	0	1	2	3	4
Events	unauth	auth'ing	auth'ed	authorizing	trans-ready
auth-needed	sap/1	sap/1	sap/1	sap/1	der,sap/1
auth-accept	-	ain/2	-	-	-
auth-bad	-	0	wa/0	wa,der/0	der,sap/1
authz-start	szp/3	-	szp/3	-	-
data-rcvd	szp/3	qd/1	szp/3	qd/3	pd/4
authz-ok	-	-	-	4	-
authz-bad	-	-	-	der/2	wa,der,szp/3

Action	Description
sap	start authentication process
der	deallocate end-system resources
ain	authorize if needed
szp	start authorization process
qd	queue incoming data
pd	process data
wa	wipe authorization info

Event	Description
auth-needed	authentication deemed needed by local party
auth-accept	remote party's authentication creds accepted
auth-bad	remote party's authentication creds rejected or expired
authz-start	local or remote party starts authorization proceedings
data-rcvd	data destined for end-system received from remote party
authz-ok	authorization and resource allocation succeeded
authz-bad	authorization or resource allocation failed or expired

4.1 Authentication of the Server by the Client

A secure connection requires that the client be able to authenticate the identity of the server. How the authentication is performed depends upon the TLS cipher agreed upon during the negotiation. As of this writing there are three categories of cipher suites supported by TLS: ciphers supporting X.509 certificates (PKI), non-PKI ciphers, and anonymous ciphers. The following sections detail how Server authentication should be performed by the client for each cipher category.

4.1.1 PKI-based Authentication via TLS handshake

When a PKI based cipher is negotiated during the TLS negotiation, the server will deliver an X.509 certificate to the client. Before the certificate MAY be used to determine the identity of the server, the certifiicate MUST be validated as per RFC 2459.

Once validated the identity of the server is confirmed by matching the DNS name used to access the host with the name stored in the certificate. If the certificate includes the `subjectAltName' extension and it contains a `dNSName' object, then the client MUST use this name as the identity of the server. Otherwise, the (most specific) commonName field in the Subject field if the certificate MUST be used. Note that although the commonName field technique is currently in wide use, it is deprecated and Certification Authorities are encourage to use the dnsName instead.

Matching is performed using the matching rules specified by [RFC 2459]. If more than one identity of a given type is present in the certificate (e.g., more than one dnsName name, a match in any one of the set is considered acceptable.) Names may contain the wildcard character '*' which is considered to match any single domain name component or component fragment. E.g., "*.a.com" matches "foo.a.com" but not "bar.foo.a.com. "f*.com" matches "foo.com" but not "bar.com".

In some cases, an IP address is used to access the host instead of a DNS name. In these cases, a 'subjectAltName' object of type 'iPAddress' MUST be present in the certificate and MUST exactly match the IP address provided by the end user.

If the hostname does not match the identity in the certificate, user oriented clients MUST either notify the user (clients MAY give the user the opportunity to continue with the connection in any case) or terminate the connection with a bad certificate error. Automated clients MUST log the error to an appropriate audit log (if available) and SHOULD terminate the connection (with a bad certificate error.) Automated clients MAY provide a configuration setting that disables this check, but MUST provide a setting which enables it.

4.1.2 Non-PKI based authentication via TLS handshake

As of this writing TLS only supports one class of non-PKI cipher suites which are based on Kerberos 5. Regardless, any non-PKI cipher suite incorporated into TLS will provide for mutual authentication. Authentication of the server is therefore implied by a successful TLS credential exchange.

4.1.3 Authentication by Telnet AUTH option (RFC 2941)

If the TLS exchange used an anonymous cipher such as Anonymous-Diffie-Hellman (ADH) or if the X.509 certificate could not be validated, then the session MUST be protected from a man in the middle attack. This can be accomplished by using a Telnet AUTH [AUTH] method that provides for mutual authentication(*) of the client and server; and which allows the TLS Finished messages sent by the client and the server to be verified. A failure to successfully perform a mutual authentication with Finished message verification via Telnet AUTH MUST result in termination of the connection by both the client and the server.

(*) The Telnet AUTH option supports both unilateral and mutual authentication methods. The distinction being that mutual authentication methods confirm the identity of both parties at the end of the negotiation. A unilateral authentication method cannot be used to verify the contents of the TLS client and server finished messages. It is worth noting that TLS usually authenticates the server to the client; whereas, Telnet AUTH usually authenticates the client to the server when unilateral methods are used.

4.2 Authentication of the Client by the Server

After TLS has been successfully negotiated the server may not have the client's identity (verified or not) since the client is not required to provide credentials during the TLS exchange. Even when the client does provide credentials during the TLS exchange, the server may have a policy that prevents their use. Therefore, the server may not have enough confidence in the client to move the connection to the authenticated state.

If further client, server or client-server authentication is going to occur after TLS has been negotiated, it MUST occur before any non-authentication-related Telnet interactions take place on the link after TLS starts. When the first non-authentication-related Telnet interaction is received by either participant, then the receiving participant MAY drop the connection due to dissatisfaction with the level of authentication.

If the server wishes to request a client certificate after TLS is initially started (presumably with no client certificate requested), it may do so. However, the server MUST make such a request immediately after the initial TLS handshake is complete.

No TLS negotiation outcome, however trustworthy, will by itself provide the server with the authorization identity if that is different from the authentication identity of the client.

The following subsections detail how the client can provide the server with authentication and authorization credentials.

4.2.1 PKI-based Authentication via TLS handshake

PKI-based authentication is used by the client transmitting an X.509 certificate to the host during the TLS handshake. There is no standard mechanism defined for how a client certificate should be mapped to a authorization identity (userid). There are several methods currently in wide practice. A telnet server compliant with this document may implement zero, one or more than one of them.

The first method is to use information stored within the certificate to determine the authorization identity. If the certificate contains an Common Name object then portions of it can be used as the authorization identity. If the Common Name contains an UID member, then it can be used directly. If the Common Name contains an Email member, then it can be used if the specified domain matches the domain of the telnet server.

The second method is to use the entire Subject Name as a entry to lookup in a directory. The directory provides a mapping between the subject name and the authorization identity.

The third method is to use the entire certificate as a entry to lookup in a directory with the directory providing a mapping between the certificate and the authorization identity.

The first method is only practical if the certificates are being issued by certificate authority managed by the same organization as the server performing the authorization.

The second and third methods can be used with certificates issued by public certificate authorities provided the certificates are delivered to the organization performing the authorization in advance via an authenticated method. The second and third methods have the added benefit that the certificates, if issued by the authorizing organization, do not require that any personal information about the subject be included in the certificate. The Subject line could be filled only with gibberish to establish its uniqueness in the directory.

4.2.2   Non-PKI Authentication via TLS handshake

TLS supports non-PKI authentication methods which can be used for securely establishing authentication identities. As of this writing, TLS supports only one non-PKI authentication method, Kerberos 5. However, it is not unlikely that other authentication methods might be incorporated into TLS ciphers in the future.

4.2.3   Telnet AUTH option

The Telnet AUTH option implements a variety of authentication methods which can be used to establish authentication and authorization identities. Some methods (e.g., KERBEROS_IV and KERBEROS_V) allow separate authentication and authorization identities to be provided. Details on the use of the Telnet AUTH option and its authentication methods can be found in RFC1416 (about to be obsoleted) and its related documents. For a current list of Telnet AUTH methods see IANA.

4.2.4 Traditional Username and Password

When all else fails the authorization identity may be provided over the secure TLS connection in the form of a username and password.

The Username MAY be transmitted to the host via the Telnet New Environment option's USER variable.

5 Security

Security is discussed throughout this document. Most of this document concerns itself with wire protocols and security frameworks. But in this section, client and server implementation security issues are in focus.

5.1 PKI-based certificate processing

A complete discussion of the proper methods for verifying X.509 certificates and their associated certificate chains is beyond the scope of this document. The reader is advised to refer to the RFCs issued by the PKIX Working Group. However, the verification of a certificate MUST include, but isn't limited to, the verification of the signature certificate chain to the point where the a signature in that chain uses a known good signing certificate in the local key chain. The verification SHOULD then continue with a check to see if the fully qualified host name which the client connected to appears anywhere in the server's certificate subject (DN).

If the certificate cannot be verified then either:

- o the end user MUST see a display of the server's certificate and be asked if he/she is willing to proceed with the session; or,

- o the end user MUST NOT see a display of server's certificate, but the certificate details are logged on whatever media is used to log other session details. This option may be preferred to the first option in environments where the end-user cannot be expected to make an informed decision about whether a mismatch is harmful. The connection MUST be closed automatically by the client UNLESS the client has been configured to explicitly allow all mismatches.

- o the connection MUST be closed on the user's behalf, and an error describing the mismatch logged to stable storage.

If the client side of the service is not interactive with a human end-user, the Telnet connection SHOULD be dropped if this host check fails.

5.2 Client and Server authentication of anonymous-TLS connections

When authentication is performed after the establishment of a TLS session which uses an anonymous cipher, it is imperative that the authentication method protect against a man in the middle attack by verifying the contents of the client's and server's TLS finished messages. Without the verification of both the client and server's TLS finished messages it is impossible to confirm that there is not a man in the middle listening and perhaps changing all the data transmitted on the connection.

Verification of the TLS finished messages can be performed as part of a Telnet AUTH option mutual authentication exchange (when using the ENCRYPT_START_TLS flag.) This can be done at the same time the verification of the authentication-type-pair is performed.

5.3   Display of security levels

The Telnet client and server MAY, during the TLS protocol negotiation phase, choose to use a weak cipher suite due to policy, law or even convenience. It is, however, important that the choice of weak cipher suite be noted as being commonly known to be vulnerable to attack. In particular, both server and client software should note the choice of weak cipher-suites in the following ways:

- If the Telnet endpoint is communicating with a human end-user, the user-interface SHOULD display the choice of weak cipher-suite and the fact that such a cipher-suite may compromise security.

- The Telnet endpoints SHOULD log the exact choice of cipher-suite as part of whatever logging/accounting mechanism normally used.

5.4   Trust Relationships and Implications

Authentication and authorization of the remote Telnet party is useful, but can present dangers to the authorizing party even if the connection between the client and server is protected by TLS using strong encryption and mutual authentication. This is because there are some trust-relationships assumed by one or both parties:

- Each side assumes that the authentication and authentication details proferred by the remote party stay constant until explicitly changed (or until the TLS session is ended).

- More stringently, each side trusts the other to send a timely notification if authentication or authorization details of the other party's end system(s) have changed.

Either of these assumptions about trust may be false if an intruder has breached communications between a client or server and its respective end system. And either may be false if a component is badly implemented or configured. Implementers should take care in program construction to avoid invalidating these trust relationships, and should document to configuring-users the proper ways to configure the software to avoid invalidation of these relationships.

5.5 Telnet negotation handling

There are two aspects to Telnet negotiation handling that affect the security of the connection. First, given the asynchronous nature of Telnet option negotiations it is possible for a telnet client or server to allow private data to be transmitted over a non-secure link. It is especially important that implementors of this telnet option ensure that no telnet option subnegotiations other than those related to authentication and establishment of security take place over an insecure connection.

The second item is related to the most common error when implementing a telnet protocol state machine. Most telnet implementations do not check to ensure that the peer responds to all outstanding requests to change states: WILL, DO, WONT, DONT. It is important that all telnet implementations ensure that requests for state changes are responded to.

6 TLS Variants and Options

TLS has different versions and different cipher suites that can be supported by client or server implementations. The following subsections detail what TLS extensions and options are mandatory. The subsections also address how TLS variations can be accommodated.

6.1 Support of previous versions of TLS

TLS has its roots in SSL 2.0 and SSL 3.0. Server and client implementations may wish to support for SSL 3.0 as a fallback in case TLS 1.0 or higher is not supported. This is permissible; however, client implementations which negotiate SSL3.0 MUST still follow the rules in Section 5.3 concerning disclosure to the end-user of transport-level security characteristics.

Negotiating the use of SSL 3.0 is done as part of the TLS negotiation; it is detailed in [TLS]. SSL 2.0 MUST NOT be negotiated.

6.2 Using Kerberos V5 with TLS

If the client and server are both amenable to using Kerberos V5, then using non-PKI authentication techniques within the confines of TLS may be acceptable (see [TLSKERB]). Note that clients and servers are under no obligation to support anything but the cipher-suite(s) mandated in [TLS]. However, if implementations do implement the KRB5 authentication as a part of TLS ciphersuite, then these implementations SHOULD support at least the TLS_KRB5_WITH_3DES_EDE_CBC_SHA ciphersuite.

7 Protocol Examples

The following sections provide skeletal examples of how Telnet clients
and servers can negotiate TLS.

7.1   Successful TLS negotiation

The following protocol exchange is the typical sequence that starts TLS:

```
// typical successful opening exchange
  S: IAC DO START_TLS
  C: IAC WILL START_TLS IAC SB START_TLS FOLLOWS IAC SE
  S: IAC SB START_TLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation completes with a Finished exchanged]
// either side now able to send further Telnet data or commands
```

7.2   Successful TLS negotiation, variation

The following protocol exchange is the typical sequence that starts TLS,
but with the twist that the (TN3270E) server is willing but not
aggressive about doing TLS; the client strongly desires doing TLS.

```
// typical successful opening exchange
  S: IAC DO TN3270E
  C: IAC WILL START_TLS IAC
  S: IAC DO START_TLS
  C: IAC WILL START_TLS IAC SB START_TLS FOLLOWS IAC SE
  S: IAC SB START_TLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation completes with a Finished
                exchanged]
// note that server retries negotiation of TN3270E after TLS
// is done.
  S: IAC DO TN3270E
  C: IAC WILL TN3270E
// TN3270E dialog continues....
```

7.3   Unsuccessful TLS negotiation

This example assumes that the server does not wish to allow the Telnet
session to proceed without TLS security; however, the client's version
of TLS does not interoperate with the server's.

```
//typical unsuccessful opening exchange
  S: IAC DO START_TLS
  C: IAC WILL START_TLS IAC SB START_TLS FOLLOWS IAC SE
  S: IAC SB START_TLS FOLLOWS IAC SE
```

```
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation fails with server sending
               ErrorAlert message]
  S: [TCP level disconnect]
//  server (or both) initiate TCP session disconnection
```

This example assumes that the server wants to do TLS, but is willing to allow the session to proceed without TLS security; however, the client's version of TLS does not interoperate with the server's.

```
//typical unsuccessful opening exchange
  S: IAC DO START_TLS
  C: IAC WILL START_TLS IAC SB START_TLS FOLLOWS IAC SE
  S: IAC SB START_TLS FOLLOWS IAC SE
// server now readies input stream for non-Telnet, TLS-level negotiation
  C: [starts TLS-level negotiations with a ClientHello]
  [TLS transport-level negotiation ensues]
  [TLS transport-level negotiation fails with server sending
               ErrorAlert message]
  S: [TCP level disconnect]
// session is dropped
```

7.4   Authentication via Kerberos 4 after TLS negotiation

Here's an implementation example of using Kerberos 4 to authenticate the client after encrypting the session with TLS. Note the following details:

- The client strictly enforces a security policy of proposing Telnet AUTH first, but accepting TLS. This has the effect of producing a rather verbose pre-TLS negotiation sequence; however, the end-result is correct. A more efficient pre-TLS sequence can be obtained by changing the client security policy to be the same as the server's for this connection (and implementing policy-aware negotiation code in the Telnet part of the client).

  A similar efficient result can be obtained even in the absence of a clear client security policy if the client has cached server security preferences from a previous Telnet session to the same server.

- The server strictly enforces a security policy of proposing TLS first, but falling back to Telnet AUTH.

```
  C: IAC WILL AUTHENTICATION
  C: IAC WILL NAWS
  C: IAC WILL TERMINAL-TYPE
  C: IAC WILL NEW-ENVIRONMENT
  S: IAC DO START_TLS
```

```
C: IAC WILL START_TLS
C: IAC SB START_TLS FOLLOWS IAC SE
S: IAC DO AUTHENTICATION
S: IAC DO NAWS
S: IAC WILL SUPPRESS-GO-AHEAD
S: IAC DO SUPPRESS-GO-AHEAD
S: IAC WILL ECHO
S: IAC DO TERMINAL-TYPE
S: IAC DO NEW-ENVIRONMENT
S: IAC SB AUTHENTICATION SEND
  KERBEROS_V4 CLIENT_TO_SERVER|MUTUAL|ENCRYPT_REQ
  KERBEROS_V4 CLIENT_TO_SERVER|MUTUAL
  KERBEROS_V4 CLIENT_TO_SERVER|ONE_WAY
  SSL CLIENT_TO_SERVER|ONE_WAY   IAC SE
S: IAC SB TERMINAL-TYPE SEND  IAC SE
S: IAC SB NEW-ENVIRONMENT SEND  IAC SE
S: IAC SB START_TLS FOLLOWS  IAC SE
[TLS - handshake starting]
[TLS - OK]
C: IAC WILL AUTHENTICATION
C: IAC WILL NAWS
C: IAC WILL TERMINAL-TYPE
C: IAC WILL NEW-ENVIRONMENT
<wait for outstanding negotiations>
S: IAC DO AUTHENTICATION
S: IAC DO NAWS
S: IAC WILL SUPPRESS-GO-AHEAD
C: IAC DO SUPPRESS-GO-AHEAD
S: IAC DO SUPPRESS-GO-AHEAD
C: IAC WILL SUPPRESS-GO-AHEAD
S: IAC WILL ECHO
C: IAC DO ECHO
S: IAC DO TERMINAL-TYPE
S: IAC DO NEW-ENVIRONMENT
S: IAC SB AUTHENTICATION SEND
  KERBEROS_V4 CLIENT_TO_SERVER|MUTUAL
  KERBEROS_V4 CLIENT_TO_SERVER|ONE_WAY   IAC SE
C: IAC SB AUTHENTICATION NAME jaltman IAC SE
C: IAC SB AUTHENTICATION IS
  KERBEROS_V4 CLIENT_TO_SERVER|MUTUAL AUTH
  04 07 0B "CC.COLUMBIA.EDU" 00 "8(" 0D 9E 9F AB A0 "L" 15 8F A6
  ED "x" 19 F8 0C "wa" CA "z`" 1A E2 B8 "Y" B0 8E "KkK" C6 AA "<" FF
  FF 98 89 "|" 90 AC DF 13 "2" FC 8E 97 F7 BD AE "e" 07 82 "n" 19 "v"
  7F 10 C1 12 B0 C6 "|" FA BB "s1Y" FF FF 10 B5 14 B3 "(" BC 86 "`"
  D2 "z" AB "Qp" C4 "7" AB "]8" 8A 83 B7 "j" E6 "IK" DE "|YIVN"
  IAC SE
S: IAC SB TERMINAL-TYPE SEND  IAC SE
S: IAC SB NEW-ENVIRONMENT SEND  IAC SE
S: IAC SB AUTHENTICATION REPLY
 KERBEROS_V4 CLIENT_TO_SERVER|MUTUAL ACCEPT  IAC SE
C: IAC SB AUTHENTICATION IS
```

```
   KERBEROS_V4 CLIENT|MUTUAL CHALLENGE "[" BE B7 96 "j" 92 09 "~" IAC SE
  S: IAC SB AUTHENTICATION REPLY
   KERBEROS_V4 CLIENT_TO_SERVER|MUTUAL RESPONSE "df" B0 D6 "vR_/"  IAC SE
  C: IAC SB TERMINAL-TYPE IS VT320 IAC SE
  C: IAC SB NEW-ENVIRONMENT IS VAR USER VALUE jaltman VAR SYSTEMTYPE \\
          VALUE WIN32 IAC SE
  C: IAC SB NAWS 162 49 IAC SE
```

Here are several things to note about the above example:

- o After TLS is successfully negotiated, all non-TLS Telnet settings are forgotten and must be renegotiated.

- o After TLS is successfully negotiated, the server offers all authentication types that are appropriate for a session using TLS. Note that the server, post TLS-negotiation, isn't offering Telnet ENCRYPT or AUTH SSL, since (a) it's useless to encrypt twice, and (b) TLS and/or SSL can be applied only once to a Telnet session.

8 References

[ANBF]   D. Crocker, Ed., P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC2235, November 1997.

[AUTH]   T.Ts'o, Ed,. J. Altman, "Telnet Authentication Option", RFC2941, September 2000.

[ENCRYPT]   T.Ts'o, "Telnet Encryption Option", RFC2946, September 2000.

[KEYWORDS]   Bradner, S. "Key words for use in RFCs to Indicate Requirement Levels", RFC2119, March 1997.

[RFC927]   Brian A. Anderson. "TACACS User Identification Telnet Option", RFC927, December 1984

[RFC2360]   G. Scott, Editor. "Guide for Internet Standard Writers", RFC2360, June 1998.

[RFC2459]   Housley, R., Ford, W., Polk, W. and D.Solo, "Internet Public Key Infrastructure: Part I: X.509 Certificate and CRL Profile", RFC2459, January 1999.

[TELNET]   J. Postel, J. Reynolds. "Telnet Protocol Specifications", RFC854, May 1983.

[TLS]   Tim Dierks, C. Allen. "The TLS Protocol", RFC2246, January 1999.

[TLSKERB]   Ari Medvinsky, Matthew Hur. "Addition of Kerberos Cipher Suites to Transport Layer Security (TLS)", RFC2712, October 1999.

9 Authors

Michael Boe
Cisco Systems Inc.
170 West Tasman Drive
San Jose CA 95134 USA

Email: mboe@cisco.com

Jeffrey Altman
Columbia University
612 West 115th Street
New York NY 10025 USA

jaltman@columbia.edu

10 Mailing list

General Discussion:tn3270e@list.nih.gov
To Subscribe: listserv@list.nih.gov
In Body: sub tn3270e <first_name>
Archive: listserv@list.nih.gov

Associated list: telnet-wg@bsdi.com

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