Network Working Group
Request for Comments: 2943
Category: Standards Track

R. Housley
T. Horting
P. Yee
SPYRUS

September 2000

TELNET Authentication Using DSA

Status of this Memo

Copyright Notice

Abstract

This document defines a telnet authentication mechanism using the Digital Signature Algorithm (DSA) [FIPS186]. It relies on the Telnet Authentication Option [RFC2941].

1. Command Names and Codes

```
    AUTHENTICATION           37

      Authentication Commands:

        IS                        0
        SEND                      1
        REPLY                     2
        NAME                      3

      Authentication Types:

        DSS                      14

      Modifiers:

        AUTH_WHO_MASK             1
        AUTH_CLIENT_TO_SERVER     0
        AUTH_SERVER_TO CLIENT     1
```

AUTH_HOW_MASK	2
AUTH_HOW_ONE_WAY	0
AUTH_HOW_MUTUAL	2
ENCRYPT_MASK	20
ENCRYPT_OFF	0
ENCRYPT_USING_TELOPT	4
ENCRYPT_AFTER_EXCHANGE	16
ENCRYPT_RESERVED	20
INI_CRED_FWD_MASK	8
INI_CRED_FWD_OFF	0
INI_CRED_FWD_ON	8

Sub-option Commands:

DSS_INITIALIZE	1
DSS_TOKENBA	2
DSS_CERTA_TOKENAB	3
DSS_CERTB_TOKENBA2	4

2. TELNET Security Extensions

TELNET, as a protocol, has no concept of security. Without negotiated options, it merely passes characters back and forth between the NVTs represented by the two TELNET processes. In its most common usage as a protocol for remote terminal access (TCP port 23), TELNET connects to a server that requires user-level authentication through a user name and password in the clear; the server does not authenticate itself to the user.

The TELNET Authentication Option provides for user authentication and server authentication. User authentication replaces or augments the normal host password mechanism. Server authentication is normally done in conjunction with user authentication.

In order to support these security services, the two TELNET entities must first negotiate their willingness to support the TELNET Authentication Option. Upon agreeing to support this option, the parties are then able to perform sub-option negotiations to the authentication protocol to be used, and possibly the remote user name to be used for authorization checking.

Authentication and parameter negotiation occur within an unbounded series of exchanges. The server proposes a preference-ordered list of authentication types (mechanisms) which it supports. In addition to listing the mechanisms it supports, the server qualifies each mechanism with a modifier that specifies whether the authentication is to be one-way or mutual, and in which direction the authentication is to be performed. The client selects one mechanism from the list and responds to the server indicating its choice and the first set of authentication data needed for the selected authentication type. The server and the client then proceed through whatever number of iterations are required to arrive at the requested authentication.

3. Use of Digital Signature Algorithm (DSA)

DSA is also known as the Digital Signature Standard (DSS), and the names are used interchangeably. This paper specifies a method in which DSA may be used to achieve certain security services when used in conjunction with the TELNET Authentication Option. SHA-1 [FIPS180-1] is used with DSA [FIPS186].

DSA may provide either unilateral or mutual authentication. Due to TELNET’s character-by-character nature, it is not well-suited to the application of integrity-only services, therefore use of the DSA profile provides authentication but it does not provide session integrity. This specification follows the token and exchanges defined in NIST FIPS PUB 196 [FIPS196], Standard for Public Key Cryptographic Entity Authentication Mechanisms including Appendix A on ASN.1 encoding of messages and tokens. All data that is covered by a digital signature must be encoded using the Distinguished Encoding Rules (DER). However, other data may use either the Basic Encoding Rules (BER) or DER [X.208].

3.1. Unilateral Authentication with DSA

Unilateral authentication must be done client-to-server. What follows are the protocol steps necessary to perform DSA authentication as specified in FIPS PUB 196 under the TELNET Authentication Option framework. Where failure modes are encountered, the return codes follow those specified in the TELNET Authentication Option. They are not enumerated here, as they are invariant among the mechanisms used. FIPS PUB 196 employs a set of exchanges that are transferred to provide authentication. Each exchange employs various fields and tokens, some of which are optional. In addition, each token has several subfields that are optional. A conformant subset of the fields and subfields have been selected. The tokens are ASN.1 encoded as defined in Appendix A of FIPS PUB 196, and each token is named to indicate the direction in which it flows (e.g., TokenBA flows from Party B to Party A). All data that is covered by a digital signature must be encoded using the

Distinguished Encoding Rules (DER). Data that is not covered by a digital signature may use either the Basic Encoding Rules (BER) or DER [X.208]. Figure 1 illustrates the exchanges for unilateral authentication.

During authentication, the client may provide the user name to the server by using the authentication name sub-option. If the name sub-option is not used, the server will generally prompt for a name and password in the clear. The name sub-option must be sent after the server sends the list of authentication types supported and before the client finishes the authentication exchange, this ensures that the server will not prompt for a user name and password. In figure 1, the name sub-option is sent immediately after the server presents the list of authentication types supported.

For one-way DSS authentication, the two-octet authentication type pair is DSS AUTH_CLIENT_TO_SERVER | AUTH_HOW_ONE_WAY | ENCRYPT_OFF | INI_CRED_FWD_OFF. This indicates that the DSS authentication mechanism will be used to authenticate the client to the server and that no encryption will be performed.

CertA is the clients certificate. Both certificates are X.509 certificates that contain DSS public keys[RFC2459]. The client must validate the server’s certificate before using the DSA public key it contains.

Within the unbounded authentication exchange, implementation is greatly simplified if each portion of the exchange carries a unique identifier. For this reason, a single octet sub-option identifier is carried immediately after the two-octet authentication type pair.

The exchanges detailed in Figure 1 below presume knowledge of FIPS PUB 196 and the TELNET Authentication Option. The client is Party A, while the server is Party B. At the end of the exchanges, the client is authenticated to the server.

```
------------------------------------------------------------------
 Client (Party A)                   Server (Party B)

                                 <-- IAC DO AUTHENTICATION

 IAC WILL AUTHENTICATION     -->

                                 <-- IAC SB AUTHENTICATION SEND
                                     <list of authentication options>
                                     IAC SE

 IAC SB AUTHENTICATION
 NAME <user name>            -->

 IAC SB AUTHENTICATION IS
 DSS
 AUTH_CLIENT_TO_SERVER |
     AUTH_HOW_ONE_WAY |
     ENCRYPT_OFF |
     INI_CRED_FWD_OFF
 DSS_INITIALIZE
 IAC SE                     -->

                                 <-- IAC SB AUTHENTICATION REPLY
                                     DSS
                                     AUTH_CLIENT_TO_SERVER |
                                         AUTH_HOW_ONE_WAY |
                                         ENCRYPT_OFF |
                                         INI_CRED_FWD_OFF
                                     DSS_TOKENBA
                                     Sequence( TokenID, TokenBA )
                                     IAC SE

 IAC SB AUTHENTICATION IS
 DSS
 AUTH_CLIENT_TO_SERVER |
     AUTH_HOW_ONE_WAY |
     ENCRYPT_OFF |
     INI_CRED_FWD_OFF
 DSS_CERTA_TOKENAB
 Sequence( TokenID, CertA, TokenAB )
 IAC SE                     -->
------------------------------------------------------------------
```
Figure 1

3.2. Mutual Authentication with DSA

Mutual authentication is slightly more complex. Figure 2 illustrates the exchanges.

For mutual DSS authentication, the two-octet authentication type pair is DSS AUTH_CLIENT_TO_SERVER | AUTH_HOW_MUTUAL | ENCRYPT_OFF | INI_CRED_FWD_OFF. This indicates that the DSS authentication mechanism will be used to mutually authenticate the client and the server and that no encryption will be performed.

```
---------------------------------------------------------------------
 Client (Party A)                   Server (Party B)

IAC WILL AUTHENTICATION        -->

                                    <-- IAC DO AUTHENTICATION

                                    <-- IAC SB AUTHENTICATION SEND
                                        <list of authentication options>
                                        IAC SE

 IAC SB AUTHENTICATION
 NAME <user name>              -->

 IAC SB AUTHENTICATION IS
 DSS
 AUTH_CLIENT_TO_SERVER |
     AUTH_HOW_MUTUAL |
     ENCRYPT_OFF |
     INI_CRED_FWD_OFF
 DSS_INITIALIZE
 IAC SE                        -->

                                    <-- IAC SB AUTHENTICATION REPLY
                                        DSS
                                        AUTH_CLIENT_TO_SERVER |
                                            AUTH_HOW_MUTUAL |
                                            ENCRYPT_OFF |
                                            INI_CRED_FWD_OFF
                                        DSS_TOKENBA
                                        Sequence( TokenID, TokenBA )
                                        IAC SE
 Client (Party A)                   Server (Party B)

 IAC SB AUTHENTICATION IS
 DSS
 AUTH_CLIENT_TO_SERVER |
     AUTH_HOW_MUTUAL |
     ENCRYPT_OFF |
     INI_CRED_FWD_OFF
 DSS_CERTA_TOKENAB
 Sequence( TokenID, CertA, TokenAB )
 IAC SE                        -->

                                    <-- IAC SB AUTHENTICATION REPLY
                                        DSS
                                        AUTH_CLIENT_TO_SERVER |
                                            AUTH_HOW_MUTUAL |
                                            ENCRYPT_OFF |
                                            INI_CRED_FWD_OFF
                                        DSS_CERTB_TOKENBA2
                                        Sequence( TokenID, CertB,
                                                  TokenBA2 )
                                        IAC SE
---------------------------------------------------------------------
```

Figure 2

4. ASN.1 Syntax

As stated earlier, a conformant subset of the defined fields and subfields from FIPS PUB 196 have been selected. This section provides the ASN.1 syntax for that conformant subset.

Figure 1 and Figure 2 include representations of the structures defined in this section. Implementors should refer to the following table to determine the ASN.1 definitions that match the figure references:

Figure 1	Sequence(TokenID, TokenBA)	MessageBA
	Sequence(TokenID, CertA, TokenAB)	MessageAB
Figure 2	Sequence(TokenID, TokenBA)	MessageBA
	Sequence(TokenID, CertA, TokenAB)	MessageAB
	Sequence(TokenID, CertB, TokenBA2)	MessageBA2

The following ASN.1 definitions specify the conformant subset of FIPS 196. For simplicity, no optional fields or subfields are included. The ASN.1 definition for CertificationPath is imported from CCITT Recommendation X.509 [X.509], and The ASN.1 definition for Name is imported from CCITT Recommendation X.501 [X.501]. These ASN.1

definitions are not repeated here. All DSA signature values are encoded as a sequence of two integers, employing the same conventions specified in RFC 2459, section 7.2.2.

```
      MessageBA  ::=  SEQUENCE  {
        tokenId       [0] TokenId,
        tokenBA           TokenBA  }

      TokenBA  ::=  SEQUENCE  {
        ranB              RandomNumber,
        timestampB        TimeStamp  }

      MessageAB  ::=  SEQUENCE  {
        tokenId       [0] TokenId,
        certA         [1] CertData,
        tokenAB           TokenAB  }

      TokenAB  ::=  SEQUENCE  {
        ranA              RandomNumber,
        ranB              RandomNumber,
        entityB           EntityName,
        timestampB        TimeStamp,
        absigValue        OCTET STRING  }

      MessageBA2  ::=  SEQUENCE  {
        tokenId       [0] TokenId,
        certB         [1] CertData,
        tokenBA2          TokenBA2  }

      TokenBA2  ::=  SEQUENCE  {
        ranB          [0] RandomNumber,
        ranA          [1] RandomNumber,
        entityA           EntityName,
        timestampB2       TimeStamp,
        ba2sigValue       OCTET STRING  }

      CertData  ::=  SEQUENCE  {
        certPath      [0] CertificationPath  }  -- see X.509

      EntityName  ::=  SEQUENCE OF CHOICE  {    -- only allow one!
        directoryName [4] Name  }               -- see X.501

      RandomNumber  ::=  INTEGER                -- 20 octets
      TokenId  ::=  SEQUENCE  {
        tokenType         INTEGER,             -- see table below
        protoVerNo        INTEGER  }           -- always 0x0001

      TimeStamp  ::=  GeneralizedTime
```

The TokenId.TokenType is used to distinguish the message type and the authentication type (either unilateral or mutual). The following table provides the values needed to implement this specification:

Message Type	Authentication Type	TokenId.TokenType
MessageBA	Unilateral	0x0001
	Mutual	0x0011
MessageAB	Unilateral	0x0002
	Mutual	0x0012
MessageBA	Mutual	0x0013

5. Security Considerations

This entire memo is about security mechanisms. For DSA to provide the authentication discussed, the implementation must protect the private key from disclosure.

Implementations must randomly generate DSS private keys, ’k’ values used in DSS signatures, and nonces. The use of inadequate pseudo-random number generators (PRNGs) to generate cryptographic values can result in little or no security. An attacker may find it much easier to reproduce the PRNG environment that produced the values, searching the resulting small set of possibilities, rather than using a brute force search. The generation of quality random numbers is difficult. RFC 1750 [RFC1750] offers important guidance in this area, and Appendix 3 of FIPS PUB 186 [FIPS186] provides one quality PRNG technique.

6. Acknowledgements

We would like to thank William Nace for support during implementation of this specification.

7. IANA Considerations

The authentication type DSS and its associated suboption values are registered with IANA. Any suboption values used to extend the protocol as described in this document must be registered with IANA before use. IANA is instructed not to issue new suboption values without submission of documentation of their use.

8. References

FIPS180-1	Secure Hash Standard. FIPS Pub 180-1. April 17, 1995. <http://csrc.nist.gov/fips/fips180-1.pdf>
FIPS186	Digital Signature Standard (DSS). FIPS Pub 186. May 19, 1994. <http://csrc.nist.gov/fips/fips186.pdf>
FIPS196	Standard for Entity Authentication Using Public Key Cryptography. FIPS Pub 196. February 18, 1997. <http://csrc.nist.gov/fips/fips196.pdf>
RFC1750	Eastlake, 3rd, D., Crocker, S. and J. Schiller, "Randomness Recommendations for Security", RFC 1750, December 1994.
RFC2459	Housley, R., Ford, W., Polk, W. and D. Solo, "Internet X.509 Public Key Infrastructure: X.509 Certificate and CRL Profile", RFC 2459, January 1999.
RFC2941	T’so, T. and J. Altman, "Telnet Authentication Option", RFC 2941, September 2000.
X.208	CCITT. Recommendation X.208: Specification of Abstract Syntax Notation One (ASN.1). 1988.
X.501	CCITT. Recommendation X.501: The Directory - Models. 1988.
X.509	CCITT. Recommendation X.509: The Directory - Authentication Framework. 1988.

9. Authors' Addresses

Russell Housley
SPYRUS
381 Elden Street, Suite 1120
Herndon, VA 20172
USA

EMail: housley@spyrus.com

Todd Horting
SPYRUS
381 Elden Street, Suite 1120
Herndon, VA 20172
USA

EMail: thorting@spyrus.com

Peter Yee
SPYRUS
5303 Betsy Ross Drive
Santa Clara, CA 95054
USA

EMail: yee@spyrus.com

10. Full Copyright Statement

Acknowledgement

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