Transport Layer Security Working Group
Internet-Draft
Expires: May 30, 2003

D. Taylor
Forge Research Pty Ltd

November 29, 2002

Using SRP for TLS Authentication
draft-ietf-tls-srp-04

Status of this Memo

Copyright Notice

Abstract

This memo presents a technique for using the SRP [2] (Secure Remote Password) protocol as an authentication method for the TLS [1](Transport Layer Security) protocol.

Table of Contents

1. Introduction

At the time of writing, TLS uses public key certificiates with RSA/ DSA digital signatures, or Kerberos, for authentication.

These authentication methods do not seem well suited to the applications now being adapted to use TLS (IMAP [4], FTP [6], or TELNET [7], for example). Given these protocols (and others like them) are designed to use the user name and password method of authentication, being able to safely use user names and passwords to authenticate the TLS connection provides a much easier route to additional security than implementing a public key infrastructure in certain situations.

SRP is an authentication method that allows the use of user names and passwords over unencrypted channels without revealing the password to an eavesdropper. SRP also supplies a shared secret at the end of the authetication sequence that can be used to generate encryption keys.

This document describes the use of the SRP authentication method for TLS.

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 RFC 2119.

2. SRP Authentication in TLS

2.1 Modifications to the TLS Handshake Sequence

The advent of SRP-6 [3] allows the SRP protocol to be implemented using the standard sequence of handshake messages defined in [1].

The parameters to various messages are given in the following diagram.

2.1.1 Message Sequence

Handshake Message Flow for SRP Authentication

```
      Client                                      Server
        |                                           |
   Client Hello (I) ------------------------>     |
        |  <----------------------------- Server Hello
        |  <----------------------------- Certificate*
        |  <----------------------------- Server Key Exchange (N, g, s, B)
        |  <----------------------------- Server Hello Done
   Client Key Exchange (A) ----------------->     |
   [Change cipher spec]                           |
   Finished -------------------------------->     |
        |                         [Change cipher spec]
        |  <----------------------------- Finished
        |                                           |
   Application Data  <---------------> Application Data
```

* Indicates optional or situation-dependent messages that are not always sent.

The identifiers given after each message name refer to the SRP variables included in that message. The variables I, N, g, s, A, and B are defined in [3].

An extended client hello message, as defined in [8], is used to send the client identifier (the user name).

Servers MAY add an SRP extension to the server hello message. For the cipher suites defined in this document no information is carried in the SRP extension in the server hello message. The option to add an SRP extension to the server hello message is given in case it is required in future.

2.1.2 Session Re-use

The short handshake mechanism for re-using sessions for new

connections, and renegotiating keys for existing connections will still work with the SRP authentication mechanism and handshake.

When a client attemps to re-use a session that uses SRP authentication, it MUST include the SRP extension carrying the user name (I) in the client hello message, in case the server cannot or will not allow re-use of the session, meaning a full handshake sequence is required.

If the server does agree to re-use an existing session the server MUST ignore the information in the SRP extension of the client hello message, except for its inclusion in the finished message hashes. This is to ensure attackers cannot replace the authenticated identity without supplying the proper authentication information.

2.2 SRP Verifier Message Digest Selection

Implementations conforming to this document MUST use the SHA-1 message digest with the SRP algorithm.

2.3 Changes to the Handshake Message Contents

This section describes the changes to the TLS handshake message contents when SRP is being used for authentication. The definitions of the new message contents and the on-the-wire changes are given in Section 2.6.

2.3.1 Client hello

The user name is appended to the standard client hello message using the hello message extension mechanism defined in [8].

2.3.2 Server certificate

The server MUST send a certificate if it agrees to an SRP cipher suite that requires the server to provide additional authentication in the form of a digital signature. See Section 2.5 for details of which ciphersuites defined in this document require a server certificate to be sent.

Because the server's certificate is only used for generating a digital signature in SRP cipher suites, the certificate sent MUST contain a public key that can be used for generating digital signatures.

2.3.3 Server key exchange

The server key exchange message contains the prime (N), the generator

(g), and the salt value (s) read from the SRP password file based on the value of (I) received in the client hello extension. The server key exchange message also contains the server's public key (B).

If the server has sent a certificate message, the server key exchange message MUST be signed.

2.3.4 Client key exchange

The client key exchange message carries the client's public key (A).

2.4 Calculating the Pre-master Secret

The shared secret resulting from the SRP calculations (S) (defined in

[2]) is used as the pre-master secret.

The finished messages perform the same function as the client and server evidence messages (M1 and M2) specified in [2]. If either the client or the server calculate an incorrect value, the finished messages will not be understood, and the connection will be dropped as specified in [1].

2.5 Cipher Suite Definitions

The following cipher suites are added by this draft. The usage of AES ciphersuites is as defined in [5].

```
CipherSuite TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA     = { 0x00,0x50 };

CipherSuite TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA = { 0x00,0x51 };

CipherSuite TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA = { 0x00,0x52 };

CipherSuite TLS_SRP_SHA_WITH_AES_128_CBC_SHA      = { 0x00,0x53 };

CipherSuite TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA  = { 0x00,0x54 };

CipherSuite TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA  = { 0x00,0x55 };

CipherSuite TLS_SRP_SHA_WITH_AES_256_CBC_SHA      = { 0x00,0x56 };

CipherSuite TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA  = { 0x00,0x57 };

CipherSuite TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA  = { 0x00,0x58 };
```

Cipher suites that do not include a digitial signature algorithm identifier assume the server is authenticated by its possesion of the SRP database.

Cipher suites that begin with TLS_SRP_SHA_RSA or TLS_SRP_SHA_DSS require the server to send a certificate message containing a certificate with the specified type of public key, and to sign the server key exchange message using a matching private key.

Implementations conforming to this specification MUST implement the TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA ciphersuite, SHOULD implement the

TLS_SRP_SHA_WITH_AES_128_CBC_SHA and TLS_SRP_SHA_WITH_AES_256_CBC_SHA ciphersuites, and MAY implement the remaining ciphersuites.

2.6 New Message Structures

This section shows the structure of the messages passed during a handshake that uses SRP for authentication. The representation language used is the same as that used in [1].

2.6.1 ExtensionType

A new value, "srp(6)", has been added to the enumerated ExtensionType, defined in [8]. This value MUST be used as the extension number for the SRP extension.

2.6.2 Client Hello

The user name (I) is encoded in an SRPExtension structure, and sent in an extended client hello message, using an extension of type "srp".

```
enum { client, server } ClientOrServerExtension;

struct {
   select(ClientOrServerExtension) {
      case client:
         opaque srp_I<1..2^8-1>;
      case server:
         /* empty struct */
   }
} SRPExtension;
```

2.6.3 Server Key Exchange

When the value of KeyExchangeAlgorithm is set to "srp", the server's SRP parameters are sent in the server key exchange message, encoded in a ServerSRPParams structure.

If a certificate is sent to the client the server key exchange message must be signed. The following table gives the SignatureAlgorithm value to be used for each ciphersuite.

Ciphersuite	SignatureAlgorithm
TLS_SRP_SHA_WITH_3DES_EDE_CBC_SHA	anonymous
TLS_SRP_SHA_RSA_WITH_3DES_EDE_CBC_SHA	rsa
TLS_SRP_SHA_DSS_WITH_3DES_EDE_CBC_SHA	dsa
TLS_SRP_SHA_WITH_AES_128_CBC_SHA	anonymous
TLS_SRP_SHA_RSA_WITH_AES_128_CBC_SHA	rsa
TLS_SRP_SHA_DSS_WITH_AES_128_CBC_SHA	dsa
TLS_SRP_SHA_WITH_AES_256_CBC_SHA	anonymous
TLS_SRP_SHA_RSA_WITH_AES_256_CBC_SHA	rsa
TLS_SRP_SHA_DSS_WITH_AES_256_CBC_SHA	dsa

```
struct {
   select (KeyExchangeAlgorithm) {
      case diffie_hellman:
         ServerDHParams params;
         Signature signed_params;
      case rsa:
         ServerRSAParams params;
         Signature signed_params;
      case srp:   /* new entry */
         ServerSRPParams params;
         Signature signed_params;
   };
} ServerKeyExchange;

struct {
   opaque srp_N<1..2^16-1>;
   opaque srp_g<1..2^16-1>;
   opaque srp_s<1..2^8-1>
   opaque srp_B<1..2^16-1>;
} ServerSRPParams;     /* SRP parameters */
```

2.6.4 Client Key Exchange

When the value of KeyExchangeAlgorithm is set to "srp", the client's ephemeral public key (A) is sent in the client key exchange message, encoded in an ClientSRPPublic structure.

An extra value, srp, has been added to the enumerated KeyExchangeAlgorithm, originally defined in TLS [1].

```
struct {
   select (KeyExchangeAlgorithm) {
      case rsa: EncryptedPreMasterSecret;
      case diffie_hellman: ClientDiffieHellmanPublic;
      case srp: ClientSRPPublic;   /* new entry */
   } exchange_keys;
} ClientKeyExchange;

enum { rsa, diffie_hellman, srp } KeyExchangeAlgorithm;

struct {
   opaque srp_A<1..2^16-1>;
} ClientSRPPublic;
```

3. Security Considerations

If an attacker is able to steal the SRP verifier file, the attacker can masquerade as the real host. Filesystem based X.509 certificate installations are vulnerable to a similar attack unless the server's certificate is issued from a PKI that maintains revocation lists, and the client TLS code can both contact the PKI and make use of the revocation list.

References

[1] Dierks, T. and C. Allen, "The TLS Protocol", RFC 2246, January 1999.

[2] Wu, T., "The SRP Authentication and Key Exchange System", RFC 2945, September 2000.

[3] Wu, T., "SRP-6: Improvements and Refinements to the Secure Remote Password Protocol", October 2002.

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

[5] Chown, P., "Advanced Encryption Standard (AES) Ciphersuites for Transport Layer Security (TLS)", RFC 3268, June 2002.

[6] Ford-Hutchinson, P., Carpenter, M., Hudson, T., Murray, E. and V. Wiegand, "Securing FTP with TLS", draft-murray-auth-ftp-ssl-09 (work in progress), April 2002.

[7] Boe, M. and J. Altman, "TLS-based Telnet Security", draft-ietf-tn3270e-telnet-tls-06 (work in progress), April 2002.

[8] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and T. Wright, "TLS Extensions", draft-ietf-tls-extensions-05 (work in progress), July 2002.

Author's Address

David Taylor
Forge Research Pty Ltd

EMail: DavidTaylor@forge.com.au
URI: http://www.forge.com.au/

Appendix A. Acknowledgements

Thanks to all on the IETF tls mailing list for ideas and analysis.

Thanks to Tom Wu for adapting the SRP protocol so it fits the standard TLS handshake message sequence.

Full Copyright Statement

Acknowledgement

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