Network Working Group
Internet-Draft
Expires: December 28, 2001

D. Taylor
Forge Research Pty Ltd

June 29, 2001

Using SRP for TLS Authentication
draft-ietf-tls-srp-01

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 28, 2001.

Copyright Notice

Abstract

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

Table of Contents

1. Introduction

At the time of writing, TLS [1] 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 [3], FTP [4], or TELNET [5], for example). Given these protocols (and others like them) are designed to use the user name and password method of authentication, being able to use user names and passwords to authenticate the TLS connection seems to be a useful feature.

SRP [2] 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.

2. SRP Authentication in TLS

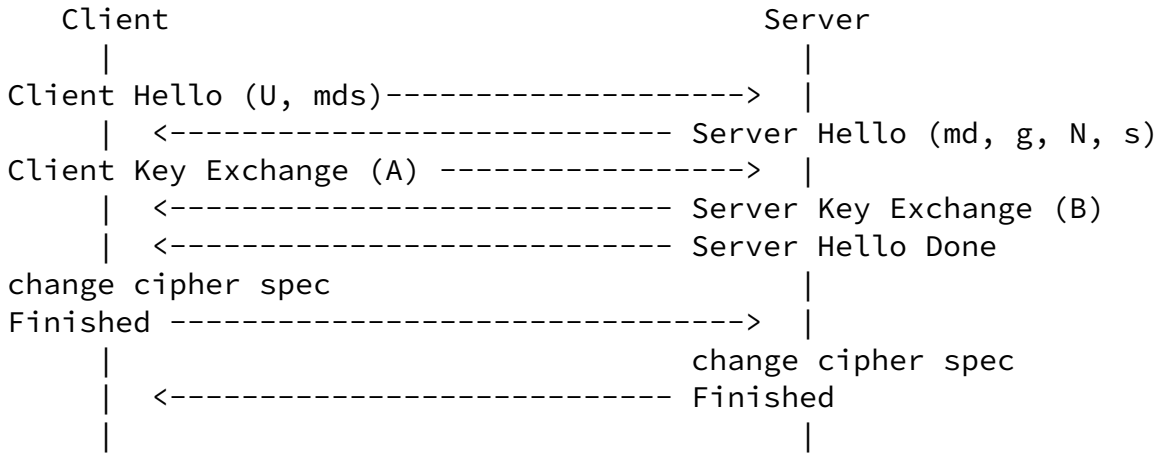
2.1 Modifications to the TLS Handshake Sequence

The SRP protocol can not be implemented using the sequence of handshake messages defined in [1] due to the sequence in which the SRP messages must be sent.

This document proposes a new sequence of handshake messages for handshakes using the SRP authentication method.

2.1.1 Message Sequence

Handshake Message Flow for SRP Authentication



The identifiers given after each message name refer to the SRP variables included in that message. The variables are defined in [2], except for (mds) and (md) which are defined in this document.

Extended client and server hello messages, as defined in [6], are used to to send the initial client and server values.

The client key exchange message is sent during the sequence of server messages. This modification is required because the client must send its public key (A) before it receives the servers public key (B), as stated in Section 3 of [2].

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 still include the SRP extension carrying the

user name (U) 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 a client requests an existing session and the server agrees to use it (meaning the short handshake will be used), the server MAY omit the SRP extension from the server hello message, as the information it contains is not used in the short handshake.

2.2 SRP Verifier Message Digest Selection

SRP uses a message digest algorithm when creating password verifiers, and when performing calculations during authentication. At the time of writing, SHA-1 is the only algorithm that has been defined for use with SRP. However, there is no reason other message digest algorithms cannot be used, and the handshake messages and extensions defined by this draft include a message digest algorithm selection mechanism.

The passwordMessageDigest enumerated, the srp_mds vector, and srp_md value are used to determine which message digest alorithm is to be

used by the client when it is performing the SRP calculation. The server determines which message digest algorithm to use based on the list of message digest algorithms requested by the client, and the list of available SRP verifiers known by the server.

The client sends a list of message digest algorithms it can use for the SRP calculation using the srp_mds vector. The server MUST select a message digest algorithm that is in the list supplied by the client, and the server MUST have access to an SRP verifier calculated with the selected message digest algorithm.

If the server has access to multiple SRP verifiers for the given user (each calculated using a different message disgest algorithm), the server may select whichever matching message digest algorithm it chooses, so long as the selected message digest algorithm appears in the list sent by the client.

If the server does not have an SRP verifier calculated with any of the message digest algorithms suggested by the client, the server must send a handshake failure alert.

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 definitons of the new message contents and the on-the-wire changes are given in Section 2.6.

2.3.1 The Client Hello Message

The user name is appended to the standard client hello message using the client hello extension mechanism defined in [6].

The list of message digests the client can use is also included. This list represents all the message digests the client can use for the SRP calculations.

2.3.2 The Server Hello Message

The message digest selected by the server (md), the generator (g), the prime (N), and the salt value (s) read from the SRP password file are appended to the server hello message using the client hello

extension mechanism defined in [6].

2.3.3 The Client Key Exchange Message

The client key exchange message carries the client's public key (A), which is calculated using both information known locally, and information received in the server hello message. This message MUST be sent before the server key exchange message.

2.3.4 The Server Key Exchange Message

The server key exchange message contains the servers public key (B). The server key exchange message MUST be sent after the client key exchange message.

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 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 numbers have been selected based on other RFCs and Internet Drafts that were current at the time of writing, so may need to be changed in future.

```
CipherSuite   TLS_SRP_WITH_3DES_EDE_CBC_SHA     = { 0x00,0x5B };
```



```
CipherSuite   TLS_SRP_WITH_RC4_128_SHA          = { 0x00,0x5C };

CipherSuite   TLS_SRP_WITH_IDEA_CBC_SHA         = { 0x00,0x5D };

CipherSuite   TLS_SRP_WITH_3DES_EDE_CBC_MD5     = { 0x00,0x5E };

CipherSuite   TLS_SRP_WITH_RC4_128_MD5          = { 0x00,0x5F };
```

```
      CipherSuite   TLS_SRP_WITH_IDEA_CBC_MD5         = { 0x00,0x60 };
```

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].

When encoding the numbers g, N, A, and B as opaque types, if the most significant bit is set, an extra byte of value 0x00 (all bits cleared) MUST be added as the most significant byte. This is done as a safeguard against implementations that do not assume these numbers are positive.

2.6.1 ExtensionType

A new value, "srp(6)", has been added to the enumerated ExtensionType, defined in [6]. This value is used as the extension number for the extensions in both the client hello message and the server hello message. This value was chosen based on the version of defined in [6] that was current at the time of writing, so may be changed in future.

2.6.2 Client Hello

The user name (U) and a list of message digests (srp_mds) are encoded in an SRPExtension structure, and sent in an extended client hello message, using an extension of type "srp".

The list of message digests represents the list of message digests the client can use for the SRP calculations.

```
enum { client, server } ClientOrServerExtension;

enum  { sha-1(0), (255) } PasswordMessageDigest;

struct {
   select(ClientOrServerExtension) {
      case client:
         opaque srp_U<1..2^8-1>;
         PasswordMessageDigest srp_mds<1..2^8-1>;
      case server:
         PasswordMessageDigest srp_md;
         opaque srp_s<1..2^8-1>
         opaque srp_N<1..2^16-1>;
         opaque srp_g<1..2^16-1>;
   }
} SRPExtension;
```

2.6.3 Server Hello

The message digest selected by the server (md), the generator (g), the prime (N), and the salt value (s) are encoded in an SRPExtension structure, which is sent in an extended server hello message, using an extension of type "srp".

The SRPParams structure is defined above.

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;
```

2.6.5 Server Key Exchange

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

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

   struct {
      opaque srp_B<1..2^16-1>;
   } ServerSRPPublic;     /* SRP parameters */
```

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 servers 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.

Not all clients and servers will be able to interoperate once the number of message digest algorithms used for creating password verifiers is increased. For example, a client may only support SHA-1, whereas the verifiers on the server were created with a different message digest algoritm.

Because the initial handshake messages are unprotected, an attacker can modify the list of message digests in the client hello message. For example, an attacker could rewrite the message to remove all but the weakest message digest. There is no way to know this has happened until the finished messages are compared.

An attacker can also modify the server hello message to use a different message digest than that selected by the server. If this happens, the handshake will fail after the change cipher spec messages are sent, as the client and server will have calculated different pre-master secret vales.

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

[4] Ford-Hutchinson, P., Carpenter, M., Hudson, T., Murray, E. and V. Wiegand, "Securing FTP with TLS", draft-murray-auth-ftp-ssl-06 (work in progress), September 2000.

[5] Boe, M. and J. Altman, "TLS-based Telnet Security", draft-ietf-tn3270e-telnet-tls-05 (work in progress), October 2000.

[6] Blake-Wilson, S., Nystrom, M., Hopwood, D., Mikkelsen, J. and T. Wright, "TLS Extensions", draft-ietf-tls-extensions-00 (work in progress), June 2001.

Author's Address

David Taylor
Forge Research Pty Ltd

EMail: DavidTaylor@forge.com.au

URI: http://www.forge.com.au/

Appendix A. Acknowledgements

The following people have contributed ideas and time to this draft: Raif Naffah, Tom Wu, Nikos Mavroyanopoulos

Full Copyright Statement

Acknowledgement

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