CRAM-MD5 Challenge-Response Authentication Mechanism

Protocol Purpose

CRAM-MD5 is intended to provide an authentication extension to IMAP4 that neither transfers passwords in cleartext nor requires significant security infrastructure in order to function. To this end, the protocol assumes a shared password (which we model, without loss of generality, as a shared cryptographic key) between the IMAP4 server (called S in our model) and each client A. Only a hash value of the shared password is ever sent over the network, thus precluding plaintext transmission.

Definition Reference

RFC 2195 [KCK97]

Model Authors

Paul Hankes Drielsma, ETH Zürich, July 2004

Alice&Bob style

```
 Alice-Bob Notation:
  1. A -> S: A
  2. S -> A: Ns.T.S
  3. A -> S: F(SK.T)
  where
      Ns is a nonce generated by the server;
      T is a timestamp (currently abstracted with a nonce)
      SK is the shared key between A and S
      F is a cryptographic hash function (MD5 in practice, but this is
       unimportant for our purposes).  The use of F
       is intended to ensure that only a digest of the shared
       key is transmitted, with T assuring freshness of the
       generated hash value.
```

Model Limitations

Issues abstracted from:

- We abstract away from the timestamp `T` using a standard nonce.

Problems considered: 2

Attacks Found

None

Further Notes

RFC 2195 [KCK97] states that the first message from the server S begins with a ”presumptively arbitrary string of random digits”; that is, a nonce. Unspecified, however, is what the client should do with this nonce. It does not appear in subsequent protocol message. We therefore presume it is intended to ensure replay protection, but our HLPSL specification at present does not explicitly model that the client should maintain a list of nonces previously received from the server.

HLPSL Specification

```
role client(A, S: agent,
            SK: message,
            F: function,
            SND, RCV: channel (dy))
played_by A
def=

  local  State : nat,
         T, Ns : text

  const  sec_SK : protocol_id

  init   State := 0
  transition

  1. State = 0   /\ RCV(start)
     =|>
     State' := 1 /\ SND(A)

  2. State = 1   /\ RCV(Ns'.T'.S)
     =|>
     State' := 2 /\ SND(F(SK.T'))
                 /\ witness(A,S,auth,F(SK.T'))
                 /\ secret(SK,sec_SK,{S})

end role
```

```
role server(S : agent,
            K,F: function,
            SND, RCV: channel (dy))
played_by S
def=

  local State : nat,
        A     : agent,
        T, Ns : text,
        Auth  : message

  init  State := 0

  transition
   1. State = 0   /\ RCV(A')
      =|>
      State' := 1 /\ Ns' := new()
                  /\ T' := new()
                  /\ SND(Ns'.T'.S)

   2. State = 1   /\ RCV(F(K(A.S).T))
      =|>
      State' := 2 /\ Auth' := F(K(A.S).T)
                  /\ request(S,A,auth,F(K(A.S).T))
```

```
end role
```

```
role session(A, S: agent,
             K, F: function)
def=

  local SK: message,
        SNDA, SNDS, RCVA, RCVS: channel (dy)

  init SK = K(A.S)

  composition
       client(A,S,SK,F,SNDA,RCVA)
    /\ server(S,K,F,SNDS,RCVS)

end role
```

```
role environment()
def=

 const a, s : agent,
       k, f : function,
       auth : protocol_id

 intruder_knowledge = {a,s,i,f}

 composition
      session(a,s,k,f)
   /\ session(i,s,k,f)
   /\ session(a,s,k,f)

end role
```

```
goal
```

```
  %secrecy_of SK
  secrecy_of sec_SK

  %Server authenticates Client on auth
  authentication_on auth

end goal
```

```
environment()
```

References

[KCK97] J. Klensin, R. Catoe, and P. Krumviede. RFC 2195: IMAP/POP AUTHorize Extension for Simple Challenge/Response, September 1997. Status: Proposed Standard.