Network Working Group
Request for Comments: 2808
Category: Informational

M. Nystrom
RSA Laboratories

April 2000

The SecurID(r) SASL Mechanism

Status of this Memo

Copyright Notice

Abstract

SecurID is a hardware token card product (or software emulation thereof) produced by RSA Security Inc., which is used for end-user authentication. This document defines a SASL [RFC2222] authentication mechanism using these tokens, thereby providing a means for such tokens to be used in SASL environments. This mechanism is only for authentication, and has no effect on the protocol encoding and is not designed to provide integrity or confidentiality services.

This memo assumes the reader has basic familiarity with the SecurID token, its associated authentication protocol and SASL.

How to read this document

The key words "MUST", "MUST NOT", "SHALL", "SHOULD" and "MAY" in this document are to be interpreted as defined in [RFC2119].

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

1. Introduction

The SECURID SASL mechanism is a good choice for usage scenarios where a client, acting on behalf of a user, is untrusted, as a one-time passcode will only give the client a single opportunity to act maliciously. This mechanism provides authentication only.

The SECURID SASL mechanism provides a formal way to integrate the existing SecurID authentication method into SASL-enabled protocols including IMAP [RFC2060], ACAP [RFC2244], POP3 [RFC1734] and LDAPv3 [RFC2251].

2. Authentication Model

The SECURID SASL mechanism provides two-factor based user authentication as defined below.

There are basically three entities in the authentication mechanism described here: A user, possessing a SecurID token, an application server, to which the user wants to connect, and an authentication server, capable of authenticating the user. Even though the application server in practice may function as a client with respect to the authentication server, relaying authentication credentials etc. as needed, both servers are, unless explicitly mentioned, collectively termed "the server" here. The protocol used between the application server and the authentication server is outside the scope of this memo. The application client, acting on behalf of the user, is termed "the client".

The mechanism is based on the use of a shared secret key, or "seed", and a personal identification number (PIN), which is known both by the user and the authentication server. The secret seed is stored on a token that the user possesses, as well as on the authentication server. Hence the term "two-factor authentication", a user needs not only physical access to the token but also knowledge about the PIN in order to perform an authentication. Given the seed, current time of day, and the PIN, a "PASSCODE(r)" is generated by the user’s token and sent to the server.

The SECURID SASL mechanism provides one service:

- User authentication where the user provides information to the server, so that the server can authenticate the user.

This mechanism is identified with the SASL key "SECURID".

3. Authentication Procedure

a) The client generates the credentials using local information (seed, current time and user PIN/password).

b) If the underlying protocol permits, the client sends credentials to the server in an initial response message. Otherwise, the client sends a request to the server to initiate the authentication mechanism, and sends credentials after the server's response (see [RFC2222] section 5.1 for more information regarding the initial response option).

Unless the server requests a new PIN (see below), the contents of the client's initial response SHALL be as follows:

(1) An authorization identity. When this field is empty, it defaults to the authentication identity. This field MAY be used by system administrators or proxy servers to login with a different user identity. This field MUST NOT be longer than 255 octets, SHALL be terminated by a NUL (0) octet, and MUST consist of UTF-8-encoded [RFC2279] printable characters only (US-ASCII [X3.4] is a subset of UTF-8).

(2) An authentication identity. The identity whose passcode will be used. If this field is empty, it is assumed to have been transferred by other means (e.g. if the underlying protocol has support for this, like [RFC2251]). This field MUST NOT be longer than 255 octets, SHALL be terminated by a NUL (0) octet, and MUST consist of UTF-8-encoded printable characters only.

(3) A passcode. The one-time password that will be used to grant access. This field MUST NOT be shorter than 4 octets, MUST NOT be longer than 32 octets, SHALL be terminated by a NUL (0) octet, and MUST consist of UTF-8-encoded printable characters only. Passcodes usually consist of 4-8 digits.

The ABNF [RFC2234] form of this message is as follows:

```
credential-pdu = authorization-id authentication-id passcode [pin]

authorization-id = 0*255VUTF8 %x00

authentication-id = 0*255VUTF8 %x00

passcode = 4*32VUTF8 %x00

pin ::= 4*32VUTF8 %x00

VUTF8 = <Visible (printable) UTF8-encoded characters>
```

Regarding the <pin> rule, see d) below.

c) The server verifies these credentials using its own information. If the verification succeeds, the server sends back a response indicating success to the client. After receiving this response, the client is authenticated. Otherwise, the verification either failed or the server needs an additional set of credentials from the client in order to authenticate the user.

d) If the server needs an additional set of credentials, it requests them now. This request has the following format, described in ABNF notation:

```
server-request = passcode | pin

passcode       = "passcode" %x00

pin            = "pin" %x00 [suggested-pin]

suggested-pin = 4*32VUTF8 %x00 ; Between 4 and 32 UTF-8 characters
```

The 'passcode' choice will be sent when the server requests another passcode. The 'pin' choice will be sent when the server requests a new user PIN. The server will either send an empty string or suggest a new user PIN in this message.

e) The client generates a new set of credentials using local information and depending on the server's request and sends them to the server. Authentication now continues as in c) above.

Note 1: Case d) above may occur e.g. when the clocks on which the server and the client relies are not synchronized.

Note 2: If the server requests a new user PIN, the client MUST respond with a new user PIN (together with a passcode), encoded as a UTF-8 string. If the server supplies the client with a suggested PIN, the client accepts this by replying with the same PIN, but MAY replace it with another one. The length of the PIN is application-dependent as are any other requirements for the PIN, e.g. allowed characters. If the server for some reason does not accept the received PIN, the client MUST be prepared to receive either a message indicating the failure of the authentication or a repeated request for a new PIN. Mechanisms for transferring knowledge about PIN requirements from the server to the client are outside the scope of this memo. However, some information MAY be provided in error messages transferred from the server to the client when applicable.

4. Examples

4.1 IMAP4

The following example shows the use of the SECURID SASL mechanism with IMAP4. The example is only designed to illustrate the protocol interaction but do provide valid encoding examples.

The base64 encoding of the last client response, as well as the "+ " preceding the response, is part of the IMAP4 profile, and not a part of this specification itself.

```
S: * OK IMAP4 server ready
C: A001 CAPABILITY
S: * CAPABILITY IMAP4 AUTH=CRAM-MD5 AUTH=SECURID
S: A001 OK done
C: A002 AUTHENTICATE SECURID
S: +
C: AG1hZ251cwAxMjM0NTY3OAA=
S: A002 OK Welcome, SECURID authenticated user: magnus
```

4.2 LDAPv3

The following examples show the use of the SECURID SASL mechanism with LDAPv3. The examples are only designed to illustrate the protocol interaction, but do provide valid encoding examples. Usernames, passcodes and PINs are of course fictitious. For readability, all messages are shown in the value-notation defined in [X680]. <credential-pdu> values are shown hex-encoded in the 'credentials' field of LDAP's 'BindRequest' and <server-request> values are shown hex-encoded in the 'serverSaslCreds' field of LDAP's 'BindResponse'.

4.2.1 LDAPv3 Example 1

Initial response message, successful authentication.

```
C: { messageID 1,
     protocolOp bindRequest :
       { version 1,
         name '434E3D4D41474E5553'H, -- "CN=MAGNUS"
         authentication sasl :
           { mechanism '53454355524944'H, -- "SECURID"
             credentials '006d61676e7573003132333435363738 00'H
           }
       }
   }
```

```
   S: { messageID 1,
        protocolOp bindResponse :
          { resultCode success,
            matchedDN  ''H,
            errorMessage ''H,
          }
      }
```

4.2.2 LDAPv3 Example 2

Initial response message, server requires second passcode.

```
   C:  {
       messageID 1,
       protocolOp bindRequest : {
           version 1,
           name '434E3D4D41474E5553'H, -- "CN=MAGNUS"
           authentication sasl : {
               mechanism '53454355524944'H, -- "SECURID"
               credentials '006d61676e75730031323334353637380 0'H
           }
       }
   }

   S:  {
       messageID 1,
       protocolOp bindResponse : {
           resultCode saslBindInProgress,
           matchedDN  ''H,
           errorMessage ''H,
           serverSaslCreds '70617373636f646500'H
       }
   }

   C:  {
       messageID 1,
       protocolOp bindRequest : {
           version 1,
           name '434E3D4D41474E5553'H, -- "CN=MAGNUS"
           authentication sasl : {
               mechanism '53454355524944'H, -- "SECURID"
               credentials '006d61676e75730038373635343332310 0'H
           }
       }
   }

   S:  {
       messageID 1,
```

```
        protocolOp bindResponse : {
            resultCode success,
            matchedDN  ''H,
            errorMessage ''H,
        }
    }
```

4.2.3 LDAPv3 Example 3

Initial response message, server requires new PIN and passcode, and supplies client with a suggested new PIN (which the client accepts).

```
    C:  {
        messageID 1,
        protocolOp bindRequest : {
            version 1,
            name '434E3D4D41474E5553'H, -- "CN=MAGNUS"
            authentication sasl : {
                mechanism '53454355524944'H, -- "SECURID"
                credentials '006d61676e75730031323334353637380 0'H
            }
        }
    }

    S:  {
        messageID 1,
        protocolOp bindResponse : {
            resultCode saslBindInProgress,
            matchedDN  ''H,
            errorMessage ''H,
            serverSaslCreds '70696e006b616c6c6500'H
        }
    }

    C:  {
        messageID 1,
        protocolOp bindRequest : {
            version 1,
            name '434E3D4D41474E5553'H, -- "CN=MAGNUS"
            authentication sasl : {
                mechanism '53454355524944'H, -- "SECURID"
            credentials '006d61676e757300383734343436373400 6b616c6c6500'H
            }
        }
    }

    S:  {
        messageID 1,
```

```
        protocolOp bindResponse : {
            resultCode success,
            matchedDN  ''H,
            errorMessage ''H,
        }
    }
```

5. Security Considerations

This mechanism only provides protection against passive eavesdropping attacks. It does not provide session privacy, server authentication or protection from active attacks. In particular, man-in-the-middle attacks, were an attacker acts as an application server in order to acquire a valid passcode are possible.

In order to protect against such attacks, the client SHOULD make sure that the server is properly authenticated. When user PINs are transmitted, user authentication SHOULD take place on a server-authenticated and confidentiality-protected connection.

Server implementations MUST protect against replay attacks, since an attacker could otherwise gain access by replaying a previous, valid request. Clients MUST also protect against replay of PIN-change messages.

5.1 The Race Attack

It is possible for an attacker to listen to most of a passcode, guess the remainder, and then race the legitimate user to complete the authentication. As for OTP [RFC2289], conforming server implementations MUST protect against this race condition. One defense against this attack is outlined below and borrowed from [RFC2289]; implementations MAY use this approach or MAY select an alternative defense.

One possible defense is to prevent a user from starting multiple simultaneous authentication sessions. This means that once the legitimate user has initiated authentication, an attacker would be blocked until the first authentication process has completed. In this approach, a timeout is necessary to thwart a denial of service attack.

6. IANA Considerations

By registering the SecurID protocol as a SASL mechanism, implementers will have a well-defined way of adding this authentication mechanism to their product. Here is the registration template for the SECURID SASL mechanism:

SASL mechanism name:	SECURID
Security Considerations:	See corresponding section of this memo
Published specification:	This memo
Person & email address to contact for further information:	See author's address section below
Intended usage:	COMMON
Author/Change controller:	See author's address section below

7. Intellectual Property Considerations

RSA Security Inc. does not make any claims on the general constructions described in this memo, although underlying techniques may be covered. Among the underlying techniques, the SecurID technology is covered by a number of US patents (and foreign counterparts), in particular US patent no. 4,885,778, no. 5,097,505, no. 5,168,520, and 5,657,388.

SecurID is a registered trademark, and PASSCODE is a trademark, of RSA Security Inc.

8. References

[RFC1734] Myers, J., "POP3 AUTHentication command", RFC 1734, December 1994.

[RFC2026] Bradner, S., "The Internet Standards Process -- Revision 3", BCP 9, RFC 2026, October 1996.

[RFC2060] Crispin, M., "Internet Message Access Protocol - Version 4rev1", RFC 2060, December 1996.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2222] Myers, J., "Simple Authentication and Security Layer", RFC 2222, October 1997.

[RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", RFC 2234, November 1997.

[RFC2244] Newman, C. and J. Myers, "ACAP -- Application Configuration Access Protocol", RFC 2244, November 1997.

[RFC2251] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[RFC2279] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[RFC2289] Haller, N., Metz, C., Nesser, P. and M. Straw, "A One-Time Password System", RFC 2289, February 1998.

[X3.4] ANSI, "ANSI X3.4: Information Systems - Coded Character Sets - 7-Bit American National Standard Code for Information Interchange (7-Bit ASCII)," American National Standards Institute.

[X680] ITU-T, "Information Technology - Abstract Syntax Notation One (ASN.1): Specification of Basic Notation," International Telecommunication Union, 1997.

9. Acknowledgements

The author gratefully acknowledges the contributions of various reviewers of this memo, in particular the ones from John Myers. They have significantly clarified and improved the utility of this specification.

10. Author’s Address

Magnus Nystrom
RSA Laboratories
Box 10704
121 29 Stockholm
Sweden

Phone: +46 8 725 0900
EMail: magnus@rsasecurity.com

11. Full Copyright Statement

Acknowledgement

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