Network Working Group
Internet-Draft
Expires: August 23, 2002

S. Josefsson
RSA Security

February 22, 2002

The EAP SecurID(r) Mechanism
draft-josefsson-eap-securid

Status of this Memo

Copyright Notice

Abstract

This document describe an EAP mechanism based on SecurID. SecurID is a hardware token card product (or software emulation thereof) produced by RSA Security, which is used for end-user authentication. The SecurID EAP mechanism can be used to provide authentication in protocols utilizing EAP, such as PPP, IEEE 802.11 Wireless LAN and possibly Bluetooth in the future.

The intent is to publish this as a Informational RFC.

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 [4].

Table of Contents

1. Introduction and Background

The SecurID EAP mechanism is a good choice for authentication in 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 EAP mechanism provides a formal way to integrate the existing SecurID authentication method into EAP-enabled protocols including PPP, Wireless LAN and possibly Bluetooth in the future.

Integrity and confidentiality of the one-time passcode are outside of the scope of this document, solutions such as PEAP [2] are recommended to solve this problem.

1.1 Rationale Behind the Design

Integrating SecurID within EAP could also have been achieved by describing a profile of the existing EAP Generic Token Card (GTC) mechanism.

The advantages of using a new EAP mechanism for SecurID is that the protocol syntax becomes well-defined. This makes it easier to programmatically use the EAP mechanism in the client and server. This is unlike GTC, which uses text strings, intended to be interpreted and acted upon by humans. The advantage of using a GTC profile for SecurID would be that of reduced deployment costs, assuming that existing EAP clients implement GTC because it is required by the EAP specification. However, investigations have shown [1] that EAP implementations in general do not support GTC. Hence, the costs of introducing a new EAP mechanism for SecurID and a SecurID profile of GTC are roughly the same. Thus our decision was based on the technical argument that a new EAP mechanism for SecurID makes for a cleaner design and easier implementation.

2. Authentication Model

The SecurID EAP mechanism provides two-factor based user authentication as defined below.

There are basically three entities in the authentication mechanism described here:

- o A client, acting on behalf of a user possessing a SecurID token.
- o An application server, to which the user wants to connect.
- o 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 etcetera as needed, both servers are, unless explicitly mentioned, collectively denoted as "the server" here, or "authenticator" using EAP terminology. The protocol used between the application server and the authentication server is outside the scope of this memo. The client, acting on behalf of the user, is denoted as "peer" using EAP terminology.

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" is generated by the user's token and sent to the server.

The SecurID EAP mechanism provides only one service, namely user authentication where the user provides information to the server, so that the server can authenticate the user.

The SecurID EAP mechanism type number is TBD.

3. Authentication Procedure

a) The optional EAP Identity Request/Response is exchanged, as per RFC 2284 [3]. The Identity indicated here interacts with the "Authentication identity" below.

b) The authenticator sends a EAP Request of type SecurID with a zero length Type-Data.

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

d) The peer sends credentials to the server in a EAP Response of type SecurID. The contents of the EAP Response Type-Data field in a peer's initial response contains the "credential-pdu" 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 defaults to the EAP Identity above which in this case MUST have been sent. 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 f) below.

e) The authenticator verifies these credentials using its own information. If the verification succeeds, the authenticator sends back a EAP Success. After receiving this response, the peer is authenticated and this protocol is finished. Otherwise, the verification either failed and a EAP Failure packet is sent and this protocol is finished, or the authenticator needs an additional set of credentials from the peer in order to authenticate it.

f) If the authenticator needs an additional set of credentials, it sends a EAP Request with type SecurID now. The Type-Data field of this request contains the "server-request" as follows:

```
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.

g) The peer generates a new set of credentials using local information and depending on the authenticator's request and sends them to the authenticator using the same format as in d) above, with the <pin> field present. Authentication now continues as in e) above.

Note 1: Case f) 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 using EAP Notification or a repeated request for a new PIN as described in the protocol

above. 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. Security Considerations

This mechanism only provides protection against passive eavesdropping attacks. It does not provide session privacy, session integrity, 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. Similarily, this mechanism does not protect from session hijacking taking place after authentication. This mechanism do not protect against replay attacks, where the attacker gains access by replaying a previous, valid request. When PIN codes are transmitted, they are sent without protection and is also subject to replay attacks.

In order to protect against these attacks, the client MUST only use this mechanism over a server authenticated and (when PIN codes are exchanged) confidentiality-protected connection by using, e.g., PEAP [2].

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.

4.1 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 [5] conforming server implementations MUST protect against this race condition. One defense against this attack is outlined below and borrowed from [5]; 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.

5. IANA Considerations

The IANA is asked to assign a new EAP mechanism number to the protocol defined in this document.

6. Intellectual Property Considerations

RSA Security 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.

7. Acknowledgments

This document is influenced by The SecurID(r) SASL Mechanism [6]. This document was improved by comments from, and discussion with, H kan Andersson, Jan-Ove Larsson and Magnus Nystr÷m.

References

[1] Aboba, B., "Presentation to PPP Extensions WG at 52:th IETF meeting in Salt Lake City", December 2001.

[2] Andersson, H., Josefsson, S., Zorn, G. and B. Aboba, "Protected Extensible Authentication Protocol (PEAP)", Work in progress draft-josefsson-pppext-eap-tls-eap-01.txt, October 2001.

[3] Blunk, L. and J. Vollbrecht, "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

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

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

[6] Nystrom, M., "The SecurID(r) SASL Mechanism", RFC 2808, April 2000.

Author's Address

Simon Josefsson
RSA Security
Arenav gen 29
Stockholm 121 29
Sweden

Phone: +46 8 7250914
EMail: sjosefsson@rsasecurity.com

Full Copyright Statement

Acknowledgement

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