Network Working Group
Internet Draft
Category: Informational
Expires: August 2008

D. Mitton
RSA Security Divison of EMC
N. Cam-Winget
Cisco Systems
February 25, 2008

Using the Protected One-Time Password Protocol for EAP-FAST Provisioning
draft-cam-winget-eap-fast-potp-provisioning-02.txt

Status of this Memo

Copyright Notice

Abstract

EAP-FAST is an extensible EAP method that enables the provisioning of credentials or other information by using the Transport Layer Security (TLS) to establish a mutually authenticated tunnel. As the tunnel may be unauthenticated, EAP-FAST further enables the use of inner EAP methods to establish mutual authentication prior to provisioning. This document describes how EAP-POTP may be used as the EAP-FAST inner method for credential provisioning.

Conventions used in this document

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

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 [RFC2119].

Table of Contents

1. Introduction

EAP-FAST [EAP-FAST] is an extensible EAP method [RFC3748] that can be used to mutually authenticate peer and server as well as provisioning information such as user credentials. [FAST-PROVISION] defines how EAP-FAST is used to enable dynamic or in-band provisioning and demonstrates how other EAP authentication methods may be used inside the protected tunnel to ensure mutual authentication prior to provisioning.

As EAP-FAST enables any inner EAP method to be used, this document describes how EAP Protected-OTP [EAP-POTP] may be employed within EAP-FAST Provisioning to ensure mutual authentication during in-band provisioning.

2. Authenticating with EAP-POTP in EAP-FAST for provisioning

Once a protected tunnel is established as defined in [FAST-PROVISION], the peer must authenticate itself to the server before the server can provision the peer. Use of EAP-POTP is negotiated between the server and the peer. After the peer responds with a EAP Payload TLV containing the EAP Identity Response, the server MAY request the use of EAP-POTP as the inner EAP authentication method.

EAP-POTP allows a protected authentication based on a pre-shared secret provisioned into a one-time password generating token. Possession of the token and an optional PIN value, provides a portable strong authenticator. The EAP-POTP method is an end-to-end authentication method that requires both parties to know the one-time password generated by the token based on that shared secret. This information allows a method of secure provisioning that does not require a user-memorized or static password. Details of the EAP-POTP method can be found in [EAP-POTP].

The server MAY use EAP-POTP as the inner EAP authentication in either Server-Authenticated or Server-Unauthenticated provisioning modes.

3. Cryptographic Calculations

The Key derivations for establishing the tunnel are as defined in [EAP-FAST] Section 5. The Intermediate Compound Key Derivation following a successful EAP-POTP authentication within EAP-FAST for provisioning is defined in [FAST-PROVISION] Section 5.2 using the resulting MSK as described in [EAP-POTP] Section 4.5.

4. Security Considerations

Though EAP-POTP, like EAP-MSCHAPv2 is a username and password based authentication mechanism, it provides several features that strengthen its security:

* The current one-time password is not exchanged, but instead, authentication is based on values derived from the password, nonces from each side and inputs including the session instance information.

* The authentication processes can be configured for various sizes of hash and iteration inputs, to slow active attacks.

* The method is resistant to man-in-the-middle attacks because of cryptographic bindings to the network messages.

* The method requires mutual authentication of the derived values.

EAP-POTP derives its session keys using a multi-state hashing function (PBKDF2) [PKCS5] whose input is based on the token code, PIN input, a random or pre-shared secret, an iteration count and information about the server, and derives an authentication value for transmittal.

It also keeps a hash of the running EAP request and response messages, using an SHA256 function. The hash values are combined with the generated keys, to cryptographically bind the authentication to the current message stream and mutually authenticate.

When using EAP-POTP as the inner method, the server can only validate this value by knowing all of the same inputs. Any man-in-the-middle change would affect the derived value and cause a failure.

When using EAP-POTP during dynamic EAP-FAST provisioning, session resumption credentials MUST NOT be used for authentication.

Due to the mutual authentication and key establishment provided by EAP-POTP, Server-Unauthenticated Provisioning Mode MAY be used when EAP-POTP is used for PAC provisioning

5. IANA Considerations

This specification requires no new IANA values to be assigned. [RFC 2434]

6. Acknowledgments

Thanks to Hao Zhou, Magnus Nystrom, and Dmitri Pal for their valuable input.

This document was prepared using 2-Word-v2.0.template.dot.

7. References

7.1. Normative References

[EAP-FAST] Cam-Winget, N., et al., "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol (EAP-FAST)", RFC 4851, May 2007.

[FAST-PROVISION] Cam-Winget, N. et al., "Dynamic Provisioning via Secure Tunneling Extensible Authentication Protocol (EAP-FAST)", draft-cam-winget-eap-fast-provisioning-06 (work in progress), February 2008.

[EAP-POTP] Nystrom M., "The EAP Protected One-Time Password Protocol (EAP-POTP)", RFC 4793, February 2007.

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

7.2. Informative References

[RFC3748] Blunk, L., Vollbrecht, J., Aboba, B., Carlson, J., and H.Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[PKCS5] RSA Laboratories, "Password-Based Cryptography Standard", PKCS #5 v2.0, March 1999.

[RFC3979] Bradner, S., "Intellectual Property Rights in IETF Technology", RFC 3979, March 2005.

[RFC3978] Bradner, S., "IETF Rights in Contributions", RFC 3978, March 2005.

[RFC2434] Narten, T., and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 2434, October 1998.

8. Author's Addresses

Nancy Cam-Winget
Cisco Systems Inc.
3625 Cisco Way
San Jose, CA 95134

Email: ncamwing@cisco.com

David Mitton
RSA Security Division of EMC
174 Middlesex Turnpike
Bedford, MA 01730

Email: dmitton@rsasecurity.com

APPENDIX A: Example of a successful Tunnel PAC provisioning using EAP-POTP mutual authentication

The following exchanges show anonymous DH with a successful EAP-POTP exchange within Phase 2 to provision a Tunnel PAC, the conversation will appear as follows:

```
    Authenticating Peer     Authenticator
    -------------------     -------------
                            <- EAP-Request/
                            Identity
    EAP-Response/
    Identity (MyID1) ->
                            <- EAP-Request/
                            EAP-Type=EAP-FAST, V=1
                            (EAP-FAST Start, S bit set, A-ID)

    EAP-Response/
    EAP-Type=EAP-FAST, V=1
    (TLS client_hello without
    PAC-Opaque extension)->
                            <- EAP-Request/
                            EAP-Type=EAP-FAST, V=1
                            (TLS server_hello,
                             TLS Server Key Exchange
                             TLS Server Hello Done)
    EAP-Response/
    EAP-Type=EAP-FAST, V=1 ->
    (TLS Client Key Exchange
     TLS change_cipher_spec,
     TLS finished)

                            <- EAP-Request/
                            EAP-Type=EAP-FAST, V=1
                            (TLS change_cipher_spec
                             TLS finished)
    EAP-Response/
    EAP-Type=EAP-FAST, V=1 ->
    (Acknowledgement)

    TLS channel established
    (messages sent within the TLS channel)

                           <-  EAP Payload TLV,
                               EAP-Request/
                               EAP Identity Request
```

```
       EAP Payload TLV, EAP-Response/
       EAP Identity Response ->

                              <-  EAP Payload TLV,
                                  EAP-Request,
                                  OTP-X,
                                  Version TLV:
                                  Highest=0, Lowest=0
                                  Server-Info TLV: N=0
                                  Session Identifier=V1
                                  Session Identifier=V2
                                  Nonce=V3
                                  OTP TLV:
                                  P=1,C=0,N=0,T=0,E=0,R=0
                                  Pepper Length=0
                                  Iteration Count=V4

       EAP Payload TLV, ->
       EAP-Response,
       OTP-X,
       Version TLV:
       Highest=0
       OTP TLV:
       P=1,C=0,N=0,T=0,E=0,R=0
       Iteration Count=V4
       Authentication Data=V5
       User Identifier TLV:
       User Identifier=V6
       Token Key Identifier TLV:
       Token Key Identifier=V7
                                <-  EAP Payload TLV,
                                  EAP-Request,
                                  OTP-X,
                                  Confirm TLV:
                                  C=0
                                  Authentication Data=V8
                                  Pepper Identifier=V9
                                  Encrypted Pepper=V10

       EAP Payload TLV, ->
       EAP-Response,
       OTP-X,
       Confirm TLV:
       (no data)
```

```
                            <- Intermediate Result TLV (Success)
                               Crypto-Binding-TLV(Version=1,
                               EAP-FAST Version=1, SNonce,
                               CompoundMAC)

      Intermediate Result TLV (Success)
      Crypto-Binding-TLV(Version=1,
      EAP-FAST Version=1,
      CNonce, CompoundMAC),
      PAC TLV (PAC-Type=User Authorization PAC)->
                            <- Result TLV (Success)
                               PAC TLV

      Result TLV (Success)
      PAC Acknowledgment ->

      TLS channel torn down
      (messages sent in cleartext)

                            <- EAP-Success
```

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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