Network Working Group
Request for Comments: 5422
Category: Informational

N. Cam-Winget
D. McGrew
J. Salowey
H. Zhou
Cisco Systems

March 2009

Dynamic Provisioning Using Flexible Authentication via Secure Tunneling Extensible Authentication Protocol (EAP-FAST)

Status of This Memo

Copyright Notice

IESG Note

EAP-FAST has been implemented by many vendors and it is used in the Internet. Publication of this specification is intended to promote interoperability by documenting current use of existing EAP methods within EAP-FAST.

The EAP method EAP-FAST-MSCHAPv2 reuses the EAP type code assigned to EAP-MSCHAPv2 (26) for authentication within an anonymous TLS tunnel. In order to minimize the risk associated with an anonymous tunnel, changes to the method were made that are not interoperable with EAP-MSCHAPv2. Since EAP-MSCHAPv2 does not support method-specific version negotiation, the use of EAP-FAST-MSCHAPv2 is implied by the use of an anonymous EAP-FAST tunnel. This behavior may cause problems in implementations where the use of unaltered EAP-MSCHAPv2 is needed inside an anonymous EAP-FAST tunnel. Since such support requires special case execution of a method within a tunnel, it also complicates implementations that use the same method code both within and outside of the tunnel method. If EAP-FAST were to be designed today, these difficulties could be avoided by utilization of unique EAP Type codes. Given these issues, assigned method types must not be re-used with different meaning inside tunneled methods in the future.

Abstract

The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol (EAP-FAST) method enables secure communication between a peer and a server by using Transport Layer Security (TLS) to establish a mutually authenticated tunnel. EAP-FAST also enables the provisioning credentials or other information through this protected tunnel. This document describes the use of EAP-FAST for dynamic provisioning.

Table of Contents

1. Introduction 4
   1.1. Specification Requirements 4
   1.2. Terminology 4
2. EAP-FAST Provisioning Modes 5
3. Dynamic Provisioning Using EAP-FAST Conversation 6
   3.1. Phase 1 TLS Tunnel 7
        3.1.1. Server-Authenticated Tunnel 7
        3.1.2. Server-Unauthenticated Tunnel 7
   3.2. Phase 2 - Tunneled Authentication and Provisioning 7
        3.2.1. Server-Authenticated Tunneled Authentication 8
        3.2.2. Server-Unauthenticated Tunneled Authentication 8
        3.2.3. Authenticating Using EAP-FAST-MSCHAPv2 8
        3.2.4. Use of Other Inner EAP Methods for EAP-FAST Provisioning 9
   3.3. Key Derivations Used in the EAP-FAST Provisioning Exchange 10
   3.4. Peer-Id, Server-Id, and Session-Id 11
   3.5. Network Access after EAP-FAST Provisioning 11
4. Information Provisioned in EAP-FAST 12

4.1. Protected Access Credential 12
4.1.1. Tunnel PAC 13
4.1.2. Machine Authentication PAC 13
4.1.3. User Authorization PAC 13
4.1.4. PAC Provisioning 14
4.2. PAC TLV Format 15
4.2.1. Formats for PAC Attributes 16
4.2.2. PAC-Key 16
4.2.3. PAC-Opaque 17
4.2.4. PAC-Info 18
4.2.5. PAC-Acknowledgement TLV 20
4.2.6. PAC-Type TLV 21
4.3. Trusted Server Root Certificate 21
4.3.1. Server-Trusted-Root TLV 22
4.3.2. PKCS#7 TLV 23
5. IANA Considerations 24
6. Security Considerations 25
6.1. Provisioning Modes and Man-in-the-Middle Attacks 25
6.1.1. Server-Authenticated Provisioning Mode and Man-in-the-Middle Attacks 26
6.1.2. Server-Unauthenticated Provisioning Mode and Man-in-the-Middle Attacks 26
6.2. Dictionary Attacks 27
6.3. Considerations in Selecting a Provisioning Mode 28
6.4. Diffie-Hellman Groups 28
6.5. Tunnel PAC Usage 28
6.6. Machine Authentication PAC Usage 29
6.7. User Authorization PAC Usage 29
6.8. PAC Storage Considerations 29
6.9. Security Claims 31
7. Acknowledgements 31
8. References 31
8.1. Normative References 31
8.2. Informative References 32
Appendix A. Examples 33
A.1. Example 1: Successful Tunnel PAC Provisioning 33
A.2. Example 2: Failed Provisioning 35
A.3. Example 3: Provisioning an Authentication Server’s Trusted Root Certificate 37

1. Introduction

EAP-FAST [RFC4851] is an EAP method that can be used to mutually authenticate the peer and server. Credentials such as a pre-shared key, certificate trust anchor, or a Protected Access Credential (PAC) must be provisioned to the peer before it can establish mutual authentication with the server. In many cases, the provisioning of such information presents deployment hurdles. Through the use of the protected TLS [RFC5246] tunnel, EAP-FAST can enable dynamic in-band provisioning to address such deployment obstacles.

1.1. Specification Requirements

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].

1.2. Terminology

Much of the terminology used in this document comes from [RFC3748]. The terms "peer" and "server" are used interchangeably with the terms "EAP peer" and "EAP server", respectively. Additional terms are defined below:

Man in the Middle (MITM)

An adversary that can successfully inject itself between a peer and EAP server. The MITM succeeds by impersonating a valid peer or server.

Provisioning

Providing a peer with a trust anchor, shared secret, or other appropriate information needed to establish a security association.

Protected Access Credential (PAC)

Credentials distributed to a peer for future optimized network authentication. The PAC consists of at most three components: a shared secret, an opaque element, and optional information. The shared secret part contains the secret key shared between the peer and server. The opaque part contains the shared secret encrypted by a private key only known to the server. It is provided to the peer and is presented back to the server when the peer wishes to obtain access to network resources. Finally, a PAC may optionally include other information that may be useful to the peer.

Tunnel PAC

A set of credentials stored by the peer and consumed by both the peer and the server to establish a TLS tunnel.

User Authorization PAC

A User Authorization PAC is server-encrypted data containing authorization information associated with a previously authenticated user. The User Authorization PAC does not contain a key, but rather it is generally bound to a Tunnel PAC, which is used with the User Authorization PAC.

Machine Authentication PAC

A Machine Authentication PAC contains server-encrypted data containing authorization information associated with a device. A Machine Authentication PAC may be used instead of a Tunnel PAC to establish the TLS tunnel to provide machine authentication and authorization information. The Machine Authentication PAC is useful in cases where the machine needs to be authenticated and authorized to access a network before a user has logged in.

2. EAP-FAST Provisioning Modes

EAP-FAST supports two modes for provisioning:

1. Server-Authenticated Provisioning Mode - Provisioning inside a TLS tunnel that provides server-side authentication.

2. Server-Unauthenticated Provisioning Mode - Provisioning inside an anonymous TLS tunnel.

The EAP-FAST provisioning modes use EAP-FAST phase 2 inside a secure TLS tunnel established during phase 1. [RFC4851] describes the EAP-FAST phases in greater detail.

In the Server-Authenticated Provisioning Mode, the peer has successfully authenticated the EAP server as part of EAP-FAST phase 1 (i.e., TLS tunnel establishment). Additional exchanges MAY occur inside the tunnel to allow the EAP server to authenticate the EAP peer before provisioning any information.

In the Server-Unauthenticated Provisioning Mode, an unauthenticated TLS tunnel is established in the EAP-FAST phase 1. The peer MUST negotiate a TLS anonymous Diffie-Hellman-based ciphersuite to signal

that it wishes to use Server-Unauthenticateded Provisioning Mode. This provisioning mode enables the bootstrapping of peers where the peer lacks strong credentials usable for mutual authentication with the server.

Since the server is not authenticated in the Server-Unauthenticated Provisioning Mode, it is possible that an attacker may intercept the TLS tunnel. If an anonymous tunnel is used, then the peer and server MUST negotiate and successfully complete an EAP method supporting mutual authentication and key derivation as described in Section 6. The peer then uses the Crypto-Binding TLV to validate the integrity of the TLS tunnel, thereby verifying that the exchange was not subject to a man-in-the-middle attack.

Server-Authenticated Provisioning Mode protects against the man-in-the-middle attack; however, it requires provisioning the peer with the credentials necessary to authenticate the server. Environments willing to trade off the security risk of a man-in-the-middle attack for ease of deployment can choose to use the Server-Unauthenticated Provisioning Mode.

Assuming that an inner EAP method and Crypto-Binding TLV exchange is successful, the server will subsequently provide credential information, such as a shared key using a PAC TLV or the trusted certificate root(s) of the server using a Server-Trusted-Root TLV. Once the EAP-FAST Provisioning conversation completes, the peer is expected to use the provisioned credentials in subsequent EAP-FAST authentications.

3. Dynamic Provisioning Using EAP-FAST Conversation

The provisioning occurs in the following steps, which are detailed in the subsequent sections and in RFC 4851. First, the EAP-FAST phase 1 TLS tunnel is established. During this process, extra material is extracted from the TLS key derivation for use as challenges in the subsequent authentication exchange. Next, an inner EAP method, such as EAP-FAST-MSCHAPv2 (Microsoft Challenge Handshake Authentication Protocol version 2), is executed within the EAP-FAST phase 2 TLS tunnel to authenticate the client using the challenges derived from the phase 1 TLS exchange. Following successful authentication and Crypto-Binding TLV exchange, the server provisions the peer with PAC information including the secret PAC-Key and the PAC-Opaque. Finally, the EAP-FAST conversation completes with Result TLV exchanges defined in RFC 4851. The exported EAP Master Session Key (MSK) and Extended MSK (EMSK) are derived from a combination of the tunnel key material and key material from the inner EAP method exchange.

3.1. Phase 1 TLS Tunnel

3.1.1. Server-Authenticated Tunnel

The provisioning EAP-FAST exchange uses the same sequence as the EAP-FAST authentication phase 1 to establish a protected TLS tunnel. Implementations supporting this version of the Sever-Authenticated Provisioning Mode MUST support the following TLS ciphersuites defined in [RFC5246]:

```
TLS_RSA_WITH_RC4_128_SHA
TLS_RSA_WITH_AES_128_CBC_SHA
TLS_DHE_RSA_WITH_AES_128_CBC_SHA
```

Other TLS ciphersuites that provide server authentication and encryption MAY be supported. The server MAY authenticate the peer during the TLS handshake in Server-Authenticated Provisioning Mode. To adhere to best security practices, the peer MUST validate the server's certificate chain when performing server-side authentication to obtain the full security benefits of Server-Authenticated provisioning.

3.1.2. Server-Unauthenticated Tunnel

Implementations supporting this version of the Sever-Unauthenticated Provisioning Mode MUST support the following TLS ciphersuite defined in [RFC5246]:

```
TLS_DH_anon_WITH_AES_128_CBC_SHA
```

Anonymous ciphersuites SHOULD NOT be allowed outside of EAP-FAST Server-Unauthenticated Provisioning Mode. Any ciphersuites that are used for Server-Unauthenticated Provisioning Mode MUST provide a key agreement contributed by both parties. Therefore, ciphersuites based on RSA key transport MUST NOT be used for this mode. Ciphersuites that are used for provisioning MUST provide encryption.

3.2. Phase 2 - Tunneled Authentication and Provisioning

Once a protected tunnel is established and the server is unauthenticated, the peer and server MUST execute additional authentication and perform integrity checks of the TLS tunnel. Even if both parties are authenticated during TLS tunnel establishment, the peer and server MAY wish to perform additional authentication within the tunnel. As defined in [RFC4851], the authentication exchange will be followed by an Intermediate-Result TLV and a Crypto-Binding TLV, if the EAP method succeeded. The Crypto-Binding TLV

provides a check on the integrity of the tunnel with respect to the endpoints of the EAP method. If the preceding is successful, then a provisioning exchange MAY take place. The provisioning exchange will use a PAC TLV exchange if a PAC is being provisioned and a Server-Trusted-Root TLV if a trusted root certificate is being provisioned. The provisioning MAY be solicited by the peer or it MAY be unsolicited. The PAC TLV exchange consists of the server distributing the PAC in a corresponding PAC TLV to the peer and the peer confirming its receipt in a final PAC TLV Acknowledgement message. The peer may also use the PAC TLV to request that the server send a PAC. The provisioning TLVs MAY be piggybacked onto the Result TLV. Many implementations process TLVs in the order they are received; thus, for proper provisioning to occur, the Result TLV MUST precede the TLVs to be provisioned (e.g., Tunnel PAC, Machine Authentication PAC, and User Authorization PAC). A PAC TLV MUST NOT be accepted if it is not encapsulated in an encrypted TLS tunnel.

A fresh PAC MAY be distributed if the server detects that the PAC is expiring soon. In-band PAC refreshing is through the PAC TLV mechanism. The decision of whether or not to refresh the PAC is determined by the server. Based on the PAC-Opaque information, the server MAY determine not to refresh a peer’s PAC, even if the PAC-Key has expired.

3.2.1. Server-Authenticated Tunneled Authentication

If Server-Authenticated Provisioning Mode is in use, then any EAP method may be used within the TLS tunnel to authenticate the peer that is allowed by the peer’s policy.

3.2.2. Server-Unauthenticated Tunneled Authentication

If Server-Unauthenticated Provisioning Mode is in use, then peer authenticates the server and the server authenticates the peer within the tunnel. The only method for performing authentication defined in this version of EAP-FAST is EAP-FAST-MSCHAPv2 (in a special way as described in the following section). It is possible for other methods to be defined to perform this authentication in the future.

3.2.3. Authenticating Using EAP-FAST-MSCHAPv2

EAP-FAST-MSCHAPv2 is a specific instantiation of EAP-MSCHAPv2 [EAP-MSCHAPv2] defined for use within EAP-FAST. The 256-bit inner session key (ISK) is generated from EAP-FAST-MSCHAPv2 by combining the 128-bit master keys derived according to RFC 3079 [RFC3079], with the MasterSendKey taking the first 16 octets and MasterReceiveKey taking the last 16 octets.

Implementations of this version of the EAP-FAST Server-Unauthenticated Provisioning Mode MUST support EAP-FAST-MSCHAPv2 as the inner authentication method. While other authentication methods exist, EAP-FAST-MSCHAPv2 was chosen for several reasons:

- It provides the ability to slow an active attack by using a hash-based challenge-response protocol.
- Its use of a challenge-response protocol, such as MSCHAPv2, provides some ability to detect a man-in-the-middle attack during Server-Unauthenticated Provisioning Mode.
- It is already supported by a large deployed base.
- It allows support for password change during the EAP-FAST provisioning modes.

When using an anonymous Diffie-Hellman (DH) key agreement, the challenges MUST be generated as defined in Section 3.3. This forms a binding between the tunnel and the EAP-FAST-MSCHAPv2 exchanges by using keying material generated during the EAP-FAST tunnel establishment as the EAP-FAST-MSCHAPv2 challenges instead of using the challenges exchanged within the protocol itself. The exchanged challenges are zeroed upon transmission, ignored upon reception, and the challenges derived from the TLS key exchange are used in the calculations. When EAP-FAST-MSCHAPv2 is used within a tunnel established using a ciphersuite other than one that provides anonymous key agreement, the randomly generated EAP-FAST-MSCHAPv2 challenges MUST be exchanged and used.

The EAP-FAST-MSCHAPv2 exchange forces the server to provide a valid ServerChallengeResponse, which must be a function of the server challenge, peer challenge, and password as part of its response. This reduces the window of vulnerability of a man-in-the-middle attack spoofing the server by requiring the attacker to successfully break the password within the peer’s challenge-response time limit.

3.2.4. Use of Other Inner EAP Methods for EAP-FAST Provisioning

Once a protected tunnel is established, typically the peer authenticates itself to the server before the server can provision the peer. If the authentication mechanism does not support mutual authentication and protection from man-in-the-middle attacks, then Server-Authenticated Provisioning Mode MUST be used. Within a server side, authenticated tunnel authentication mechanisms such as EAP-FAST-GTC (Generic Token Card) [RFC5421] MAY be used. This will enable peers using other authentication mechanisms such as password database and one-time passwords to be provisioned in-band as well.

This version of the EAP-FAST provisioning mode implementation MUST support both EAP-FAST-GTC and EAP-FAST-MSCHAPv2 within the tunnel in Server-Authenticated Provisioning Mode.

It should be noted that Server-Authenticated Provisioning Mode provides significant security advantages over Server-Unauthenticated Provisioning Mode even when EAP-FAST-MSCHAPv2 is being used as the inner method. It protects the EAP-FAST-MSCHAPv2 exchanges from potential active MITM attacks by verifying the server's authenticity before executing EAP-FAST-MSCHAPv2. Server-Authenticated Provisioning Mode is the recommended provisioning mode. The EAP-FAST peer MUST use the Server- Authenticated Provisioning Mode whenever it is configured with a valid trust root for a particular server.

3.3. Key Derivations Used in the EAP-FAST Provisioning Exchange

The TLS tunnel key is calculated according to the TLS version with an extra 72 octets of key material derived from the end of the key_block. Portions of the extra 72 octets are used for the EAP-FAST provisioning exchange session key seed and as the random challenges in the EAP-FAST-MSCHAPv2 exchange.

To generate the key material, compute:

```
key_block = PRF(master_secret,
                "key expansion",
                server_random +
                client_random);
```

until enough output has been generated.

For example, the key_block for TLS 1.0 [RFC2246] is partitioned as follows:

```
client_write_MAC_secret[hash_size]
server_write_MAC_secret[hash_size]
client_write_key[Key_material_length]
server_write_key[key_material_length]
client_write_IV[IV_size]
server_write_IV[IV_size]
session_key_seed[40]
ServerChallenge[16]
ClientChallenge[16]
```

and the key_block for subsequent versions is partitioned as follows:

```
client_write_MAC_secret[hash_size]
server_write_MAC_secret[hash_size]
client_write_key[Key_material_length]
server_write_key[key_material_length]
session_key_seed[40]
ServerChallenge[16]
ClientChallenge[16]
```

In the extra key material, session_key_seed is used for the EAP-FAST Crypto-Binding TLV exchange while the ServerChallenge and ClientChallenge correspond to the authentication server's EAP-FAST-MSCHAPv2 challenge and the peer's EAP-FAST-MSCHAPv2 challenge, respectively. The ServerChallenge and ClientChallenge are only used for the EAP-FAST-MSCHAPv2 exchange when Diffie-Hellman anonymous key agreement is used in the EAP-FAST tunnel establishment.

3.4. Peer-Id, Server-Id, and Session-Id

The provisioning modes of EAP-FAST do not change the general EAP-FAST protocol and thus how the Peer-Id, Server-Id, and Session-Id are determined is based on the [RFC4851] techniques.

Section 3.4 of [RFC4851] describes how the Peer-Id and Server-Id are determined; Section 3.5 describes how the Session-Id is generated.

3.5. Network Access after EAP-FAST Provisioning

After successful provisioning, network access MAY be granted or denied depending upon the server policy. For example, in the Server-Authenticated Provisioning Mode, access can be granted after the EAP server has authenticated the peer and provisioned it with a Tunnel PAC (i.e., a PAC used to mutually authenticate and establish the EAP-FAST tunnel). Additionally, peer policy MAY instruct the peer to disconnect the current provisioning connection and initiate a new EAP-FAST exchange for authentication utilizing the newly provisioned information. At the end of the Server-Unauthenticated Provisioning Mode, network access SHOULD NOT be granted as this conversation is intended for provisioning only and thus no network access is authorized. The server MAY grant access at the end of a successful Server-Authenticated provisioning exchange.

If after successful provisioning access to the network is denied, the EAP Server SHOULD conclude with an EAP Failure. The EAP server SHALL NOT grant network access or distribute any session keys to the Network Access Server (NAS) if this exchange is not intended to provide network access. Even though the provisioning mode completes

with a successful inner termination (e.g., a successful Result TLV), the server policy defines whether or not the peer gains network access. Thus, it is feasible that the server, while providing a successful Result TLV, may conclude that its authentication policy was not satisfied and terminate the conversation with an EAP Failure.

Denying network access after EAP-FAST Provisioning may cause disruption in scenarios such as wireless devices (e.g., in IEEE 802.11 devices, an EAP Failure may trigger a full 802.11 disassociation). While a full EAP restart can be performed, a smooth transition to the subsequent EAP-FAST authentications to enable network access can be achieved by the peer or server initiating TLS renegotiation, where the newly provisioned credentials can be used to establish a server-authenticated or mutually authenticated TLS tunnel for authentication. Either the peer or server may reject the request for TLS renegotiation. Upon completion of the TLS negotiation and subsequent authentication, normal network access policy on EAP-FAST authentication can be applied.

4. Information Provisioned in EAP-FAST

Multiple types of credentials MAY be provisioned within EAP-FAST. The most common credential is the Tunnel PAC that is used to establish the EAP-FAST phase 1 tunnel. In addition to the Tunnel PAC, other types of credentials and information can also be provisioned through EAP-FAST. They may include trusted root certificates, PACs for specific purposes, and user identities, to name a few. Typically, provisioning is invoked after both the peer and server authenticate each other and after a successful Crypto-Binding TLV exchange. However, depending on the information being provisioned, mutual authentication MAY not be needed.

At a minimum, either the peer or server must prove authenticity before credentials are provisioned to ensure that information is not freely provisioned to or by adversaries. For example, the EAP server may not need to authenticate the peer to provision it with trusted root certificates. However, the peer SHOULD authenticate the server before it can accept a trusted server root certificate.

4.1. Protected Access Credential

A Protected Access Credential (PAC) is a security credential generated by the server that holds information specific to a peer. The server distributes all PAC information through the use of a PAC TLV. Different types of PAC information are identified through the PAC Type and other PAC attributes defined in this section. This document defines three types of PACs: a Tunnel PAC, a Machine Authentication PAC, and a User Authorization PAC.

4.1.1. Tunnel PAC

The server distributes the Tunnel PAC to the peer, which uses it in subsequent attempts to establish a secure EAP-FAST TLS tunnel with the server. The Tunnel PAC includes a secret key (PAC-Key), data that is opaque to the peer (PAC-Opaque), and other information (PAC-Info) that the peer can interpret. The opaque data is generated by the server and cryptographically protected so it cannot be modified or interpreted by the peer. The Tunnel PAC conveys the server policy of what must and can occur in the protected phase 2 tunnel. It is up to the server policy to include what is necessary in a PAC-Opaque to enforce the policy in subsequent TLS handshakes. For example, user identity, I-ID, can be included as the part of the server policy. This I-ID information limits the inner EAP methods to be carried only on the specified user identity. Other types of information can also be included, such as which EAP method(s) and which TLS ciphersuites are allowed. If the server policy is not included in a PAC-Opaque, then there is no limitation imposed by the PAC on the usage of the inner EAP methods or user identities inside the tunnel established by the use of that PAC.

4.1.2. Machine Authentication PAC

The Machine Authentication PAC contains information in the PAC-Opaque that identifies the machine. It is meant to be used by a machine when network access is required and no user is logged in. Typically, a server will only grant the minimal amount of access required for a machine without a user present based on the Machine Authentication PAC. The Machine Authentication PAC MAY be provisioned during the authentication of a user. It SHOULD be stored by the peer in a location that is only accessible to the machine. This type of PAC typically persists across sessions.

The peer can use the Machine Authentication PAC as the Tunnel PAC to establish the TLS tunnel. The EAP server MAY have a policy to bypass additional inner EAP method and grant limited network access based on information in the Machine Authentication PAC. The server MAY request additional exchanges to validate machine’s other authorization criteria, such as posture information etc., before granting network access.

4.1.3. User Authorization PAC

The User Authorization PAC contains information in the PAC-Opaque that identifies a user and provides authorization information. This type of PAC does not contain a PAC-Key. The PAC-Opaque portion of the User Authorization PAC is presented within the protected EAP-FAST TLS tunnel to provide user information during stateless session

resume so user authentication MAY be skipped. The User Authorization PAC MAY be provisioned after user authentication. It is meant to be short lived and not persisted across logon sessions. The User Authorization PAC SHOULD only be available to the user for which it is provisioned. The User Authorization PAC SHOULD be deleted from the peer when the local authorization state of a user's session changes, such as upon the user logs out.

Once the EAP-FAST phase 1 TLS tunnel is established, the peer MAY present a User Authorization PAC to the server in a PAC TLV. This is sent as TLS application data, but it MAY be included in the same message as the Finished Handshake message sent by the peer. The User Authorization PAC MUST only be sent within the protection of an encrypted tunnel to an authenticated entity. The server will decrypt the PAC and evaluate the contents. If the contents are valid and the server policy allows the session to be resumed based on this information, then the server will complete the session resumption and grant access to the peer without requiring an inner authentication method. This is called stateless session resume in EAP-FAST. In this case, the server sends the Result TLV indicating success without the Crypto-Binding TLV and the peer sends back a Result TLV indicating success. If the User Authorization PAC fails the server validation or the server policy, the server MAY either reject the request or continue with performing full user authentication within the tunnel.

4.1.4. PAC Provisioning

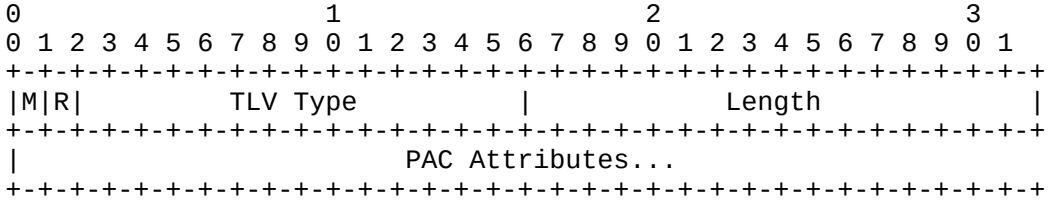
To request provisioning of a PAC, a peer sends a PAC TLV containing a PAC attribute of PAC Type set to the appropriate value. For a Tunnel PAC, the value is '1'; for a Machine Authentication PAC, the value is '2'; and for a User Authorization PAC, the value is '3'. The request MAY be issued after the peer has determined that it has successfully authenticated the EAP server and validated the Crypto-Binding TLV to ensure that the TLS tunnel's integrity is intact. Since anonymous DH ciphersuites are only allowed for provisioning a Tunnel PAC, if an anonymous ciphersuite is negotiated, the Tunnel PAC MAY be provisioned automatically by the server. The peer MUST send separate PAC TLVs for each type of PAC it wants to provision. Multiple PAC TLVs can be sent in the same packet or different packets. When requesting the Machine Authentication PAC, the peer SHOULD include an I-ID TLV containing the machine name prefixed by "host/". The EAP server will send the PACs after its internal policy has been satisfied, or it MAY ignore the request or request additional authentications if its policy dictates. If a peer receives a PAC with an unknown type, it MUST ignore it.

A PAC-TLV containing PAC-Acknowledge attribute MUST be sent by the peer to acknowledge the receipt of the Tunnel PAC. A PAC-Acknowledge TLV MUST NOT be used by the peer to acknowledge the receipt of other types of PACs.

Please see Appendix A.1 for an example of packet exchanges to provision a Tunnel PAC.

4.2. PAC TLV Format

The PAC TLV provides support for provisioning the Protected Access Credential (PAC) defined within [RFC4851]. The PAC TLV carries the PAC and related information within PAC attribute fields. Additionally, the PAC TLV MAY be used by the peer to request provisioning of a PAC of the type specified in the PAC Type PAC attribute. The PAC TLV MUST only be used in a protected tunnel providing encryption and integrity protection. A general PAC TLV format is defined as follows:



M

0 - Non-mandatory TLV
1 - Mandatory TLV

R

Reserved, set to zero (0)

TLV Type

11 - PAC TLV

Length

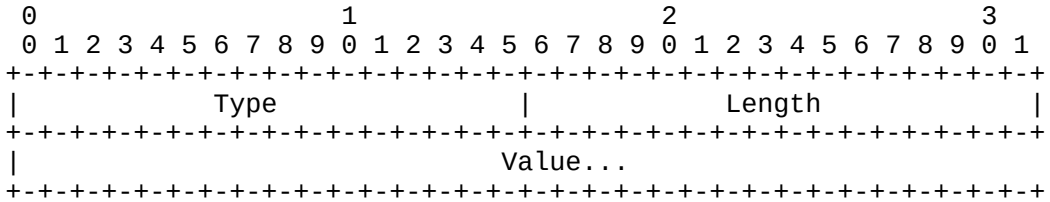
Two octets containing the length of the PAC attributes field in octets.

PAC Attributes

A list of PAC attributes in the TLV format.

4.2.1. Formats for PAC Attributes

Each PAC attribute in a PAC TLV is formatted as a TLV defined as follows:



Type

The Type field is two octets, denoting the attribute type. Allocated Types include:

1 - PAC-Key
2 - PAC-Opaque
3 - PAC-Lifetime
4 - A-ID
5 - I-ID
6 - Reserved
7 - A-ID-Info
8 - PAC-Acknowledgement
9 - PAC-Info
10 - PAC-Type

Length

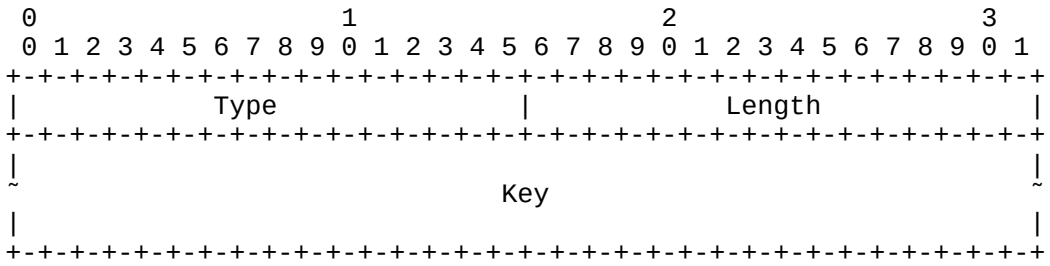
Two octets containing the length of the Value field in octets.

Value

The value of the PAC attribute.

4.2.2. PAC-Key

The PAC-Key is a secret key distributed in a PAC attribute of type PAC-Key. The PAC-Key attribute is included within the PAC TLV whenever the server wishes to issue or renew a PAC that is bound to a key such as a Tunnel PAC. The key is a randomly generated octet string, which is 32 octets in length. The generator of this key is the issuer of the credential, which is identified by the Authority Identifier (A-ID).



Type

1 - PAC-Key

Length

2-octet length indicating a 32-octet key

Key

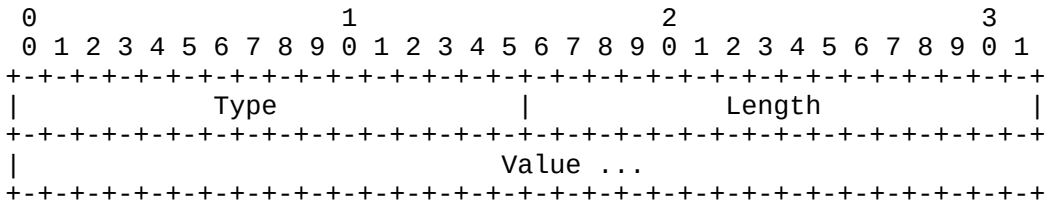
The value of the PAC-Key.

4.2.3. PAC-Opaque

The PAC-Opaque attribute is included within the PAC TLV whenever the server wishes to issue or renew a PAC or the client wishes to present a User Authorization PAC to the server.

The PAC-Opaque is opaque to the peer and thus the peer MUST NOT attempt to interpret it. A peer that has been issued a PAC-Opaque by a server stores that data and presents it back to the server according to its PAC Type. The Tunnel PAC is used in the ClientHello SessionTicket extension field defined in [RFC5077]. If a peer has opaque data issued to it by multiple servers, then it stores the data issued by each server separately according to the A-ID. This requirement allows the peer to maintain and use each opaque datum as an independent PAC pairing, with a PAC-Key mapping to a PAC-Opaque identified by the A-ID. As there is a one-to-one correspondence between the PAC-Key and PAC-Opaque, the peer determines the PAC-Key and corresponding PAC-Opaque based on the A-ID provided in the EAP-FAST/Start message and the A-ID provided in the PAC-Info when it was provisioned with a PAC-Opaque.

The PAC-Opaque attribute format is summarized as follows:



Type

2 - PAC-Opaque

Length

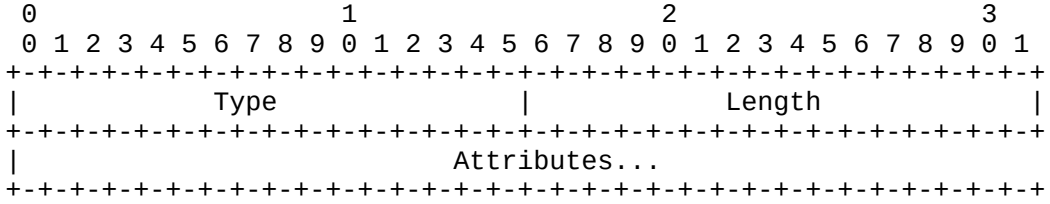
The Length filed is two octets, which contains the length of the Value field in octets.

Value

The Value field contains the actual data for the PAC-Opaque. It is specific to the server implementation.

4.2.4. PAC-Info

The PAC-Info is comprised of a set of PAC attributes as defined in Section 4.2.1. The PAC-Info attribute MUST contain the A-ID, A-ID-Info, and PAC-Type attributes. Other attributes MAY be included in the PAC-Info to provide more information to the peer. The PAC-Info attribute MUST NOT contain the PAC-Key, PAC-Acknowledgement, PAC-Info, or PAC-Opaque attributes. The PAC-Info attribute is included within the PAC TLV whenever the server wishes to issue or renew a PAC.



Type

9 - PAC-Info

Length

2-octet Length field containing the length of the attributes field in octets.

Attributes

The attributes field contains a list of PAC attributes. Each mandatory and optional field type is defined as follows:

3 - PAC-LIFETIME

This is a 4-octet quantity representing the expiration time of the credential expressed as the number of seconds, excluding leap seconds, after midnight UTC, January 1, 1970. This attribute MAY be provided to the peer as part of the PAC-Info.

4 - A-ID

The A-ID is the identity of the authority that issued the PAC. The A-ID is intended to be unique across all issuing servers to avoid namespace collisions. The A-ID is used by the peer to determine which PAC to employ. The A-ID is treated as an opaque octet string. This attribute MUST be included in the PAC-Info attribute. The A-ID MUST match the A-ID the server used to establish the tunnel. Since many existing implementations expect the A-ID to be 16 octets in length, it is RECOMMENDED that the length of an A-ID be 16 octets for maximum interoperability. One method for generating the A-ID is to use a high-quality random number generator to generate a 16-octet random number. An alternate method would be to take the hash of the public key or public key certificate belonging a server represented by the A-ID.

5 - I-ID

Initiator identifier (I-ID) is the peer identity associated with the credential. This identity is derived from the inner EAP exchange or from the client-side authentication during tunnel establishment if inner EAP method authentication is not used. The server employs the I-ID in the EAP-FAST phase 2 conversation to validate that the same peer identity used to execute EAP-FAST phase 1 is also used in at minimum one inner EAP method in EAP-FAST phase 2. If the server is enforcing the I-ID validation on the inner EAP method, then the I-ID MUST be included in the PAC-Info, to

enable the peer to also enforce a unique PAC for each unique user. If the I-ID is missing from the PAC-Info, it is assumed that the Tunnel PAC can be used for multiple users and the peer will not enforce the unique-Tunnel-PAC-per-user policy.

7 - A-ID-Info

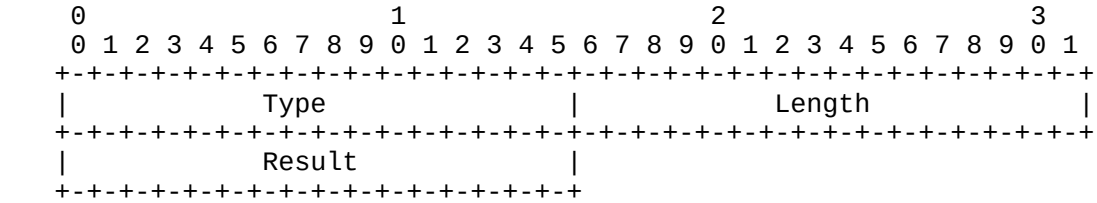
Authority Identifier Information is intended to provide a user-friendly name for the A-ID. It may contain the enterprise name and server name in a human-readable format. This TLV serves as an aid to the peer to better inform the end-user about the A-ID. The name is encoded in UTF-8 [RFC3629] format. This attribute MUST be included in the PAC-Info.

10 - PAC-type

The PAC-Type is intended to provide the type of PAC. This attribute SHOULD be included in the PAC-Info. If the PAC-Type is not present, then it defaults to a Tunnel PAC (Type 1).

4.2.5. PAC-Acknowledgement TLV

The PAC-Acknowledgement is used to acknowledge the receipt of the Tunnel PAC by the peer. The peer includes the PAC-Acknowledgement TLV in a PAC-TLV sent to the server to indicate the result of the processing and storing of a newly provisioned Tunnel PAC. This TLV is only used when Tunnel PAC is provisioned.



Type

8 - PAC-Acknowledgement

Length

The length of this field is two octets containing a value of 2.

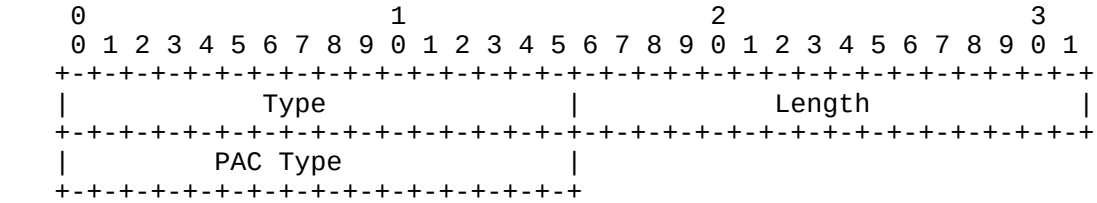
Result

The resulting value MUST be one of the following:

1 - Success
2 - Failure

4.2.6. PAC-Type TLV

The PAC-Type TLV is a TLV intended to specify the PAC type. It is included in a PAC-TLV sent by the peer to request PAC provisioning from the server. Its format is described below:



Type

10 - PAC-Type

Length

2-octet Length field with a value of 2

PAC Type

This 2-octet field defines the type of PAC being requested or provisioned. The following values are defined:

1 - Tunnel PAC
2 - Machine Authentication PAC
3 - User Authorization PAC

4.3. Trusted Server Root Certificate

Server-Trusted-Root TLV facilitates the request and delivery of a trusted server root certificate. The Server-Trusted-Root TLV can be exchanged in regular EAP-FAST authentication mode or provisioning mode. The Server-Trusted-Root TLV is always marked as optional, and

cannot be responded to with a Negative Acknowledgement (NAK) TLV. The Server-Trusted-Root TLV MUST only be sent as an inner TLV (inside the protection of the tunnel).

After the peer has determined that it has successfully authenticated the EAP server and validated the Crypto-Binding TLV, it MAY send one or more Server-Trusted-Root TLVs (marked as optional) to request the trusted server root certificates from the EAP server. The EAP server MAY send one or more root certificates with a Public Key Cryptographic System #7 (PKCS#7) TLV inside Server-Trusted-Root TLV. The EAP server MAY also choose not to honor the request. Please see Appendix A.3 for an example of a server provisioning a server trusted root certificate.

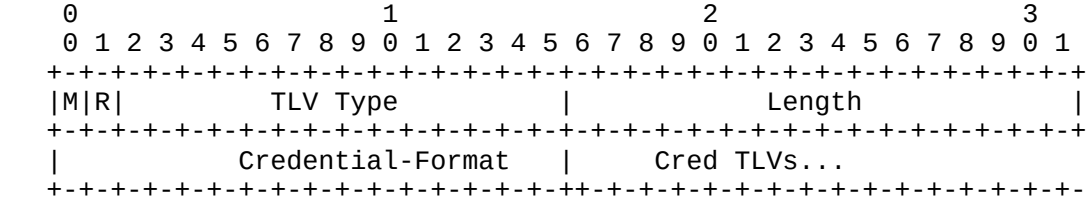
4.3.1. Server-Trusted-Root TLV

The Server-Trusted-Root TLV allows the peer to send a request to the EAP server for a list of trusted roots. The server may respond with one or more root certificates in PKCS#7 [RFC2315] format.

If the EAP server sets the credential format to PKCS#7-Server-Certificate-Root, then the Server-Trusted-Root TLV should contain the root of the certificate chain of the certificate issued to the EAP server packaged in a PKCS#7 TLV. If the Server certificate is a self-signed certificate, then the root is the self-signed certificate.

If the Server-Trusted-Root TLV credential format contains a value unknown to the peer, then the EAP peer should ignore the TLV.

The Server-Trusted-Root TLV is defined as follows:



M

0 - Non-mandatory TLV

R

Reserved, set to zero (0)

TLV Type

18 - Server-Trusted-Root TLV [RFC4851]

Length

>=2 octets

Credential-Format

The Credential-Format field is two octets. Values include:

1 - PKCS#7-Server-Certificate-Root

Cred TLVs

This field is of indefinite length. It contains TLVs associated with the credential format. The peer may leave this field empty when using this TLV to request server trust roots.

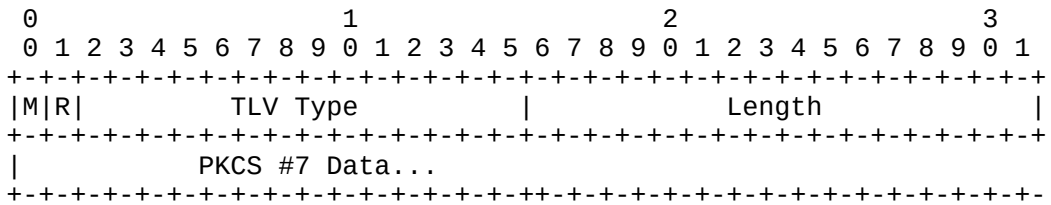
4.3.2. PKCS#7 TLV

The PKCS#7 TLV is sent by the EAP server to the peer inside the Server-Trusted-Root TLV. It contains PKCS#7-wrapped [RFC2315] X.509 certificates. The format consists of a certificate or certificate chain in a Certificates-Only PKCS#7 SignedData message as defined in [RFC2311].

The PKCS#7 TLV is always marked as optional, which cannot be responded to with a NAK TLV. EAP-FAST server implementations that claim to support the dynamic provisioning defined in this document SHOULD support this TLV. EAP-FAST peer implementations MAY support this TLV.

If the PKCS#7 TLV contains a certificate or certificate chain that is not acceptable to the peer, then the peer MUST ignore the TLV.

The PKCS#7 TLV is defined as follows:



M

0 - Optional TLV

R

Reserved, set to zero (0)

TLV Type

20 - PKCS#7 TLV [RFC4851]

Length

The length of the PKCS #7 Data field.

PKCS #7 Data

This field contains the X.509 certificate or certificate chain in a Certificates-Only PKCS#7 SignedData message.

5. IANA Considerations

This section explains the criteria to be used by the IANA for assignment of Type value in the PAC attribute, the PAC Type value in the PAC- Type TLV, and the Credential-Format value in the Server-Trusted-Root TLV. The "Specification Required" policy is used here with the meaning defined in BCP 26 [RFC5226].

A registry of values, named "EAP-FAST PAC Attribute Types", has been created for the PAC attribute types. The initial values that populate the registry are:

1 - PAC-Key
2 - PAC-Opaque
3 - PAC-Lifetime
4 - A-ID
5 - I-ID
6 - Reserved
7 - A-ID-Info
8 - PAC-Acknowledgement
9 - PAC-Info
10 - PAC-Type

Values from 11 to 63 are allocated for management by Cisco. Values 64 to 255 are assigned with a "Specification Required" policy.

A registry of values, named "EAP-FAST PAC Types", has been created for PAC-Type values used in the PAC-Type TLV. The initial values that populate the registry are:

1 - Tunnel PAC
2 - Machine Authentication PAC
3 - User Authorization PAC

Values from 4 to 63 are allocated for management by Cisco. Values 64 to 255 are assigned with a "Specification Required" policy.

A registry of values, named "EAP-FAST Server-Trusted-Root Credential Format Types", has been created for Credential-Format values used in the Server-Trusted-Root TLV. The initial values that populate the registry are:

1 - PKCS#7-Server-Certificate-Root

Values from 2 to 63 are allocated for management by Cisco. Values 64 to 255 are assigned with a "Specification Required" policy.

6. Security Considerations

The Dynamic Provisioning EAP-FAST protocol shares the same security considerations outlined in [RFC4851]. Additionally, it also has its unique security considerations described below:

6.1. Provisioning Modes and Man-in-the-Middle Attacks

EAP-FAST can be invoked in two different provisioning modes: Server-Authenticated Provisioning Mode and Server-Unauthenticated Provisioning Mode. Each mode provides different levels of resistance to man-in-the-middle attacks. The following list identifies some of the problems associated with a man-in-the-middle attack:

- Disclosure of secret information such as keys, identities, and credentials to an attacker
- Spoofing of a valid server to a peer and the distribution of false credentials
- Spoofing of a valid peer and receiving credentials generated for that peer
- Denial of service

6.1.1. Server-Authenticated Provisioning Mode and Man-in-the-Middle Attacks

In Server-Authenticated Provisioning Mode, the TLS handshake assures protected communications with the server because the peer must have been securely pre-provisioned with the trust roots and/or other authentication information necessary to authenticate the server during the handshake. This pre-provisioning step prevents an attacker from inserting themselves as a man-in-the-middle of the communications. Unfortunately, secure pre-provisioning can be difficult to achieve in many environments.

Cryptographic binding of inner authentication mechanisms to the TLS tunnel provides additional protection from man-in-the-middle attacks resulting from the tunneling of authentication mechanisms.

Server-Authenticated Provisioning Mode provides a high degree of protection from man-in-the-middle attacks.

6.1.2. Server-Unauthenticated Provisioning Mode and Man-in-the-Middle Attacks

In Server-Unauthenticated Provisioning Mode, the TLS handshake does not assure protected communications with the server because either an anonymous handshake is negotiated or the peer lacks the necessary information to complete the authentication of the server. This allows an attacker to insert itself in the middle of the TLS communications.

EAP-FAST Server-Unauthenticated Provisioning Mode mitigates the man-in-the-middle attack through the following techniques:

- Binding the phase 2 authentication method to secret values derived from the phase 1 TLS exchange:

  In the case of EAP-FAST-MSCHAPv2 used with an anonymous Diffie-Hellman ciphersuite, the challenges for the EAP-FAST-MSCHAPv2 exchange are derived from the TLS handshake and are not transmitted within the EAP-FAST-MSCHAPv2 exchange. Since the man-in-the-middle attack does not know these challenges, it cannot successfully impersonate the server without cracking the EAP-FAST-MSCHAPv2 message from the peer before the peer times out.

- Cryptographic binding of secret values derived from the phase 2 authentication exchange with secret values derived from the phase 1 TLS exchange:

This makes use of the cryptographic binding exchange defined within EAP-FAST to discover the presence of a man-in-the-middle attack by binding secret information obtained from the phase 2 EAP-FAST-MSCHAPv2 exchange with secret information from the phase 1 TLS exchange.

While it would be sufficient to only support the cryptographic binding to mitigate the MITM, the binding of the EAP-FAST-MSCHAPv2 random challenge derivations to the TLS key agreement protocol enables early detection of a man-in-the-middle attack. This guards against adversaries who may otherwise relay the inner EAP authentication messages between the true peer and server, and it enforces that the adversary successfully respond with a valid challenge response.

The ciphersuite used to establish phase 1 of the Server-Unauthenticated Provisioning Mode MUST be one in which both the peer and server provide contribution to the derived TLS master key. Ciphersuites that use RSA key transport do not meet this requirement. The authenticated and anonymous ephemeral Diffie-Hellman ciphersuites provide this type of key agreement.

This document specifies EAP-FAST-MSCHAPv2 as the inner authentication exchange; however, it is possible that other inner authentication mechanisms to authenticate the tunnel may be developed in the future. Since the strength of the man-in-the-middle protection is directly dependent on the strength of the inner method, it is RECOMMENDED that any inner method used provide at least as much resistance to attack as EAP-FAST-MSCHAPv2. Cleartext passwords MUST NOT be used in Server-Unauthenticated Provisioning Mode. Note that an active man-in-the-middle attack may observe phase 2 authentication method exchange until the point that the peer determines that authentication mechanism fails or is aborted. This allows for the disclosure of sensitive information such as identity or authentication protocol exchanges to the man-in-the-middle attack.

6.2. Dictionary Attacks

It is often the case that phase 2 authentication mechanisms are based on password credentials. These exchanges may be vulnerable to both online and off-line dictionary attacks. The two provisioning modes provide various degrees of protection from these attacks.

In online dictionary attacks, the attacker attempts to discover the password by repeated attempts at authentication using a guessed password. Neither mode prevents this type of attack by itself. Implementations should provide controls that limit how often an attacker can execute authentication attempts.

In off-line dictionary attacks, the attacker captures information that can be processed off-line to recover the password. Server-Authenticated Provisioning Mode provides effecting mitigation because the peer will not engage in phase 2 authentication without first authenticating the server during phase 1. Server-Unauthenticated Provisioning Mode is vulnerable to this type of attack. If, during phase 2 authentication, a peer receives no response or an invalid response from the server, then there is a possibility there is a man-in-the-middle attack in progress. Implementations SHOULD log these events and, if possible, provide warnings to the user. Implementations are also encouraged to provide controls, which are appropriate to their environment, that limit how and where Server-Unauthenticated Provisioning Mode can be performed. For example, an implementation may limit this mode to be used only on certain interfaces or require user intervention before allowing this mode if provisioning has succeeded in the past.

Another mitigation technique that should not be overlooked is the choice of good passwords that have sufficient complexity and length and a password-changing policy that requires regular password changes.

6.3. Considerations in Selecting a Provisioning Mode

Since Server-Authenticated Provisioning Mode provides much better protection from attacks than Server-Unauthenticated Provisioning Mode, Server-Authenticated Provisioning Mode SHOULD be used whenever possible. The Server-Unauthenticated Provisioning Mode provides a viable option as there may be deployments that can physically confine devices during the provisioning or are willing to accept the risk of an active dictionary attack. Further, it is the only option that enables zero-touch provisioning and facilitates simpler deployments requiring little to no peer configuration. The peer MAY choose to use alternative secure out-of-band mechanisms for PAC provisioning that afford better security than the Server Unauthenticated Provisioning Mode.

6.4. Diffie-Hellman Groups

To encourage interoperability implementations of EAP-FAST, anonymous provisioning modes MUST support the 2048-bit group "14" in [RFC3526].

6.5. Tunnel PAC Usage

The basic usage of the Tunnel PAC is to establish the TLS tunnel. In this operation, it does not have to provide user authentication as user authentication is expected to be carried out in phase 2 of EAP-FAST. The EAP-FAST Tunnel PAC MAY contain information about the

identity of a peer to prevent a particular Tunnel PAC from being used to establish a tunnel that can perform phase 2 authentication other peers. While it is possible for the server to accept the Tunnel PAC as authentication for the peer, many current implementations do not do this. The ability to use PAC to authenticate peers and provide authorizations will be the subject of a future document. [RFC5077] gives an example PAC-Opaque format in the Recommended Ticket Construction section.

6.6. Machine Authentication PAC Usage

In general, the Machine Authorization PAC is expected to provide the minimum access required by a machine without a user. This will typically be a subset of the privilege a registered user has. The server provisioning the PAC should include information necessary to validate it at a later point in time. This would include expiration information. The Machine Authentication PAC includes a key so it can be used as a Tunnel PAC. The PAC-Key MUST be kept secret by the peer.

6.7. User Authorization PAC Usage

The User Authorization PAC provides the privilege associated with a user. The server provisioning the PAC should include the information necessary to validate it at a later point in time. This includes expiration and other information associated with the PAC. The User Authorization PAC is a bearer credential such that it does not have a key that used to authenticate its ownership. For this reason, this type of PAC MUST NOT be sent in the clear. For additional protection, the PAC MAY be bound to a Tunnel PAC used to establish the TLS tunnel. On the peer, the User Authorization PAC SHOULD only be accessible by the user for which it is provisioned.

6.8. PAC Storage Considerations

The main goal of EAP-FAST is to protect the authentication stream over the media link. However, host security is still an issue. Some care should be taken to protect the PAC on both the peer and server. The peer must securely store both the PAC-Key and PAC-Opaque, while the server must secure storage of its security association context used to consume the PAC-Opaque. Additionally, if alternate provisioning is employed, the transportation mechanism used to distribute the PAC must also be secured.

Most of the attacks described here would require some level of effort to execute: conceivably greater than their value. The main focus therefore, should be to ensure that proper protections are used on both the peer and server. There are a number of potential attacks that can be considered against secure key storage such as:

- Weak Passphrases

  On the peer side, keys are usually protected by a passphrase. In some environments, this passphrase may be associated with the user's password. In either case, if an attacker can obtain the encrypted key for a range of users, he may be able to successfully attack a weak passphrase. The tools are already in place today to enable an attacker to easily attack all users in an enterprise environment through the use of email viruses and other techniques.

- Key Finding Attacks

  Key finding attacks are usually mentioned in reference to web servers where the private Secure Socket Layer (SSL) key may be stored securely, but at some point, it must be decrypted and stored in system memory. An attacker with access to system memory can actually find the key by identifying their mathematical properties. To date, this attack appears to be purely theoretical and primarily acts to argue strongly for secure access controls on the server itself to prevent such unauthorized code from executing.

- Key duplication, Key substitution, Key modification

  Once keys are accessible to an attacker on either the peer or server, they fall under three forms of attack: key duplication, key substitution, and key modification. The first option would be the most common, allowing the attacker to masquerade as the user in question. The second option could have some use if an attacker could implement it on the server. Alternatively, an attacker could use one of the latter two attacks on either the peer or server to force a PAC re-key, and take advantage of the potential MITM/dictionary attack vulnerability of the EAP-FAST Server-Unauthenticated Provisioning Mode.

Another consideration is the use of secure mechanisms afforded by the particular device. For instance, some laptops enable secure key storage through a special chip. It would be worthwhile for implementations to explore the use of such a mechanism.

6.9. Security Claims

The [RFC3748] security claims for EAP-FAST are given in Section 7.8 of [RFC4851]. When using anonymous provisioning mode, there is a greater risk of off-line dictionary attack since it is possible for a man-in-the-middle attack to capture the beginning of the inner EAP-FAST-MSCHAPv2 conversation. However, as noted previously, it is possible to detect the man-in-the-middle attack.

7. Acknowledgements

The EAP-FAST design and protocol specification is based on the ideas and contributions from Pad Jakkahalli, Mark Krischer, Doug Smith, Ilan Frenkel, Max Pritikin, Jan Vilhuber, and Jeremy Steiglitz. The authors would also like to thank Jouni Malinen, Pasi Eronen, Jari Arkko, Chris Newman, Ran Canetti, and Vijay Gurbani for reviewing this document.

8. References

8.1. Normative References

[EAP-MSCHAPv2] Microsoft Corporation, "MS-CHAP: Extensible Authentication Protocol Method for Microsoft Challenge Handshake Authentication Protocol (CHAP) Specification", January 2009. http://msdn2.microsoft.com/en-us/library/cc224612.aspx

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

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2311] Dusse, S., Hoffman, P., Ramsdell, B., Lundblade, L., and L. Repka, "S/MIME Version 2 Message Specification", RFC 2311, March 1998.

[RFC2315] Kaliski, B., "PKCS #7: Cryptographic Message Syntax Version 1.5", RFC 2315, March 1998.

[RFC3079] Zorn, G., "Deriving Keys for use with Microsoft Point-to-Point Encryption (MPPE)", RFC 3079, March 2001.

[RFC3526] Kivinen, T. and M. Kojo, "More Modular Exponential (MODP) Diffie-Hellman groups for Internet Key Exchange (IKE)", RFC 3526, May 2003.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

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

[RFC4851] Cam-Winget, N., McGrew, D., Salowey, J., and H. Zhou, "The Flexible Authentication via Secure Tunneling Extensible Authentication Protocol Method (EAP-FAST)", RFC 4851, May 2007.

[RFC5077] Salowey, J., Zhou, H., Eronen, P., and H. Tschofenig, "Transport Layer Security (TLS) Session Resumption without Server-Side State", RFC 5077, January 2008.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5421] Cam-Winget, N. and H. Zhou, "Basic Password Exchange within the Flexible Authentication via Secure Tunneling Extensible Authentication Protocol (EAP-FAST)", RFC 5421, March 2009.

8.2. Informative References

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

Appendix A. Examples

A.1. Example 1: Successful Tunnel PAC Provisioning

The following exchanges show anonymous DH with a successful EAP-FAST-MSCHAPv2 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-FAST,
                               (S=1, A-ID)

        EAP-Response/EAP-FAST
        (TLS Client Hello without
        PAC-Opaque in SessionTicket extension)->

                                <- EAP-Request/EAP-FAST
                                  (TLS Server Hello,
                                   TLS Server Key Exchange
                                  TLS Server Hello Done)

        EAP-Response/EAP-FAST
        (TLS Client Key Exchange
         TLS Change Cipher Spec
         TLS Finished)   ->

                                <- EAP-Request/EAP-FAST
                               ( TLS change_cipher_spec,
                                TLS finished,
                               EAP-Payload-TLV
                               (EAP-Request/Identity))

       // TLS channel established
          (Subsequent messages sent within the TLS channel,
                                   encapsulated within EAP-FAST)

       // First EAP Payload TLV is piggybacked on the TLS Finished as
          Application Data and protected by the TLS tunnel
```

```
          EAP Payload TLV
          (EAP-Response/Identity) ->

                                <-  EAP Payload TLV
                                    (EAP-Request/EAP-FAST-MSCHAPv2
                                     (Challenge))

          EAP Payload TLV
          (EAP-Response/EAP-FAST-MSCHAPv2
           (Response)) ->

                                <-  EAP Payload TLV
                                    (EAP-Request/EAP-FAST-MSCHAPv2)
                                    (Success))
          EAP Payload TLV
          (EAP-Response/EAP-FAST-MSCHAPv2
           (Success)) ->
                                 <- Intermediate Result TLV(Success)
                                    Crypto-Binding-TLV (Version=1,
                                    EAP-FAST Version=1, Nonce,
                                    CompoundMAC)

          Intermediate Result TLV (Success)
          Crypto-Binding-TLV (Version=1,
          EAP-FAST Version=1, Nonce,
          CompoundMAC)
          PAC-TLV (Type=1)
                                 <- Result TLV (Success)
                                    PAC TLV

          Result TLV (Success)
          PAC Acknowledgment ->

          TLS channel torn down
          (messages sent in cleartext)

                                 <- EAP-Failure
```

A.2. Example 2: Failed Provisioning

The following exchanges show a failed EAP-FAST-MSCHAPv2 exchange within phase 2, where the peer failed to authenticate the server. The conversation will appear as follows:

```
     Authenticating Peer     Authenticator
     -------------------     -------------
                             <- EAP-Request/Identity
     EAP-Response/
     Identity (MyID1) ->
                             <- EAP-Request/EAP-FAST
                                (s=1, A-ID)

     EAP-Response/EAP-FAST
     (TLS Client Hello without
     SessionTicket extension)->

                             <- EAP-Request/EAP-FAST
                             (TLS Server Hello
                             TLS Server Key Exchange
                             TLS Server Hello Done)
     EAP-Response/EAP-FAST
     (TLS Client Key Exchange
      TLS Change Cipher Spec,
      TLS Finished)   ->

                                          <- EAP-Request/EAP-FAST
                            ( TLS change_cipher_spec,
                             TLS finished,
                            EAP-Payload-TLV
                            (EAP-Request/Identity))

    // TLS channel established
       (Subsequent messages sent within the TLS channel,
                               encapsulated within EAP-FAST)

    // First EAP Payload TLV is piggybacked on the TLS Finished as
       Application Data and protected by the TLS tunnel


     EAP Payload TLV
     (EAP-Response/Identity)->

                            <-  EAP Payload TLV
                               (EAP-Request/EAP-FAST-MSCHAPv2
                                 (Challenge))
```

```
        EAP Payload TLV
        (EAP-Response/EAP-FAST-MSCHAPv2
         (Response)) ->

                              <-  EAP Payload TLV
                                  (EAP-Request EAP-FAST-MSCHAPv2
                                   (Success))

        // peer failed to verify server MSCHAPv2 response
        EAP Payload TLV
        (EAP-Response/EAP-FAST-MSCHAPv2
         (Failure)) ->

                              <-  Result TLV (Failure)

        Result TLV (Failure) ->
        TLS channel torn down
        (messages sent in cleartext)

                               <- EAP-Failure
```

A.3. Example 3: Provisioning an Authentication Server's Trusted Root Certificate

The following exchanges show a successful provisioning of a server trusted root certificate using anonymous DH and EAP-FAST-MSCHAPv2 exchange within phase 2. The conversation will appear as follows:

```
   Authenticating Peer     Authenticator
   -------------------     -------------
                           <- EAP-Request/
                           Identity
   EAP-Response/
   Identity (MyID1) ->
                           <- EAP-Requese/EAP-FAST
                           (s=1, A-ID)

   EAP-Response/EAP-FAST
   (TLS Client Hello without
   SessionTicket extension)->
                           <- EAP-Request/EAP-FAST
                           (TLS Server Hello,
                           (TLS Server Key Exchange
                            TLS Server Hello Done)

   EAP-Response/EAP-FAST
   (TLS Client Key Exchange
    TLS Change Cipher Spec,
    TLS Finished)  ->

                           <- EAP-Request/EAP-FAST
                           (TLS Change Cipher Spec
                            TLS Finished)
                            (EAP-Payload-TLV(
                            EAP-Request/Identity))

   // TLS channel established
      (messages sent within the TLS channel)

   // First EAP Payload TLV is piggybacked on the TLS Finished as
      Application Data and protected by the TLS tunnel

   EAP-Payload TLV
   (EAP-Response/Identity) ->

                           <- EAP Payload TLV
                              (EAP-Request/EAP-FAST-MSCHAPv2
                              (Challenge))
```

```
      EAP Payload TLV
      (EAP-Response/EAP-FAST-MSCHAPv2
       (Response)) ->

                              <-  EAP Payload TLV
                                  (EAP-Request/EAP-FAST-MSCHAPv2
                                   (success))

      EAP Payload TLV
      (EAP-Response/EAP-FAST-MSCHAPv2
       (Success) ->

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

      Intermediate Result TLV(Success)
      Crypto-Binding TLV (Version=1
      EAP-FAST Version=1, Nonce,
      CompoundMAC)
      Server-Trusted-Root TLV
      (Type = PKCS#7) ->
                               <- Result TLV (Success)
                                  Server-Trusted-Root TLV
                                  (PKCS#7 TLV)

      Result TLV (Success) ->

      // TLS channel torn down
         (messages sent in cleartext)

                               <- EAP-Failure
```

Authors' Addresses

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

EMail: ncamwing@cisco.com

David McGrew
Cisco Systems
3625 Cisco Way
San Jose, CA 95134
US

EMail: mcgrew@cisco.com

Joseph Salowey
Cisco Systems
2901 3rd Ave
Seattle, WA 98121
US

EMail: jsalowey@cisco.com

Hao Zhou
Cisco Systems
4125 Highlander Parkway
Richfield, OH 44286
US

EMail: hzhou@cisco.com