Extensible Authentication Protocol (EAP)

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC 2026.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (C) The Internet Society (2002). All Rights Reserved.

Abstract

This document defines the Extensible Authentication Protocol (EAP), an authentication protocol which supports multiple authentication mechanisms. EAP typically runs directly over the link layer without requiring IP and therefore includes its own support for in-order delivery and retransmission. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this. While EAP was originally developed for use with PPP, it is also now in use with IEEE 802.

This document obsoletes RFC 2284.
Table of Contents

1. Introduction ............................................. 3
   1.1 Specification of Requirements .......................... 3
   1.2 Terminology ........................................ 3

2. Extensible Authentication Protocol (EAP) ..................... 4
   2.1 EAP State Machine ..................................... 5

3. Media-specific issues ..................................... 6
   3.1 Lower layer assumptions ................................ 6
   3.2 EAP usage within PPP .................................... 7
   3.3 EAP usage within IEEE 802 ................................ 8

4. EAP Packet Format .......................................... 9
   4.1 Request and Response .................................... 10
   4.2 Success and Failure ..................................... 12

5. Initial EAP Request/Response Types .......................... 13
   5.1 Identity .............................................. 13
   5.2 Notification ........................................... 14
   5.3 Nak .................................................... 14
   5.4 MD5-Challenge .......................................... 15
   5.5 Vendor-specific ......................................... 16

6. IANA Considerations ....................................... 18
   6.1 Definition of Terms ..................................... 18
   6.2 Recommended Registration Policies ..................... 18

7. Normative references ....................................... 19

8. Informative references ..................................... 19

9. Security considerations .................................... 21
   9.1 Packet modification attacks ......................... 21
   9.2 Down negotiation attacks ............................. 21
   9.3 Implementation dependence ............................ 22
   9.4 Mutual authentication ................................ 22
   9.5 Key derivation ........................................ 22
   9.6 Separation of EAP server and Authenticator .......... 22
   9.7 Weak or nonexistent ciphersuites .................... 23
   9.8 Dictionary attacks ..................................... 24

ACKNOWLEDGMENTS ............................................ 24
AUTHORS’ ADDRESSES ......................................... 24
Full Copyright Statement .................................... 25
1. Introduction

This document defines the Extensible Authentication Protocol (EAP), an authentication protocol which supports multiple authentication mechanisms. EAP typically runs directly over the link layer without requiring IP and therefore includes its own support for in-order delivery and retransmission. Fragmentation is not supported within EAP itself; however, individual EAP methods may support this. While EAP was originally developed for use with PPP, it is also now in use with IEEE 802.

1.1. Specification of Requirements

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. 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

This document frequently uses the following terms:

Authenticator
The end of the link requiring the authentication.

Peer
The other end of the point-to-point link (PPP), point-to-point LAN segment (IEEE 802.1x) or 802.11 wireless link, which being authenticated by the Authenticator. In IEEE 802.1X, this end is known as the Supplicant.

Authentication Server
An Authentication Server is an entity that provides an Authentication Service to an Authenticator. This service verifies from the credentials provided by the peer, the claim of identity made by the peer.

Port Access Entity (PAE)
The protocol entity associated with a physical or virtual (802.11) Port. A given PAE may support the protocol functionality associated with the Authenticator, Peer or both.

Silently Discard
This means the implementation discards the packet without further processing. The implementation SHOULD provide the capability of logging the error, including the contents of the silently discarded packet, and SHOULD record the event in a statistics counter.
Displayable Message

This is interpreted to be a human readable string of characters, and MUST NOT affect operation of the protocol. The message encoding MUST follow the UTF-8 transformation format [RFC2044].

2. Extensible Authentication Protocol (EAP)

The Extensible Authentication Protocol (EAP) is a general protocol for authentication which supports multiple authentication mechanisms. EAP may be used on dedicated links as well as switched circuits, and wired as well as wireless links.

To date, EAP has been implemented with hosts and routers that connect via switched circuits or dial-up lines using PPP [RFC1661]. It has also been implemented with switches implementing [IEEE802], as well as Access Points implementing [IEEE80211]. EAP encapsulation on IEEE 802 media is described in [IEEE8021X].

One of the advantages of the EAP architecture is its flexibility. EAP is used to select a specific authentication mechanism, typically after the Authenticator requests more information in order to determine the specific authentication mechanism(s) to be used. Rather than requiring the Authenticator to be updated to support each new authentication method, EAP permits the use of a "back-end" server which actually implements the various mechanisms while the Authenticator merely passes through the authentication exchange.

The authentication exchange proceeds as follows:

1. After the link has been established, the Authenticator sends one or more Requests to authenticate the peer. The Request has a type field to indicate what is being requested. Examples of Request types include Identity, MD5-challenge, etc. The MD5-challenge type corresponds closely to the CHAP authentication protocol [RFC1994]. Typically, the Authenticator will send an initial Identity Request followed by one or more Requests for authentication information. However, an initial Identity Request is not required, and MAY be bypassed in cases where the identity is presumed (leased lines, dedicated switch or dial-up ports, etc.) or obtained in another fashion (via calling station identity or MAC address, in the Name field of the MD5-Challenge Response, etc.).

2. The Peer sends a Response packet in reply to each Request. As with the Request packet, the Response packet contains a type field which corresponds to the type field of the Request.
3. The Authenticator ends the authentication phase with a Success or Failure packet.

An Authenticator MAY authenticate the Peer using a sequence of methods. A common example of this is an Identity request followed by an EAP authentication method such as MD5-Challenge.

To accomplish this, the Authenticator and Peer first complete an EAP exchange involving the initial method, with a matching EAP type field included in both Request and Response packets.

If the initial authentication method completes unsuccessfully, then the Authenticator sends a Failure packet to the peer. If it completes successfully, and additional authentication methods are required, the Authenticator will send a Request packet for a subsequent authentication method. The Peer will then respond with a Response packet containing a type field matching the Request.

The sequence of authentication methods proceeds until either an authentication method fails (in which case the Authenticator sends a Failure packet to the peer) or the final authentication method completes successfully, in which case the Authenticator sends a Success packet to the peer.

Advantages

The EAP protocol can support multiple authentication mechanisms without having to pre-negotiate a particular one.

Certain devices (e.g. a NAS, switch or access point) do not necessarily have to understand each request type and may be able to simply act as a pass-through agent for a "back-end" server on a host. The device only need look for the success/failure code to terminate the authentication phase.

Disadvantages

For use in PPP, EAP does require the addition of a new authentication type to PPP LCP and thus PPP implementations will need to be modified to use it. It also strays from the previous PPP authentication model of negotiating a specific authentication mechanism during LCP. Similarly, switch or access point implementations need to support [IEEE8021X] in order to use EAP.

2.1. EAP State Machine

A formal description of the EAP state machine will be covered here. Some questions this section will answer:
[a] What happens if an Authenticator receives a EAP Response after sending a Failure or Success message?

[b] What happens if a Peer receives a Success or Failure message outside of an authentication conversation (e.g. before receiving an EAP Request?)

[c] Can a Peer send a Success or Failure message to an Authenticator? What does an Authenticator do if it receives one of these messages?

[d] What does an implementation do if it receives a message code that it doesn’t understand?

3. Media-specific issues

EAP has been run over a variety of lower layers including PPP, wired IEEE 802 LANs, and IEEE 802.11 wireless LANs; UDP (L2TP [RFC2661], PIC [PIC]) and TCP [PIC]. This section discusses the assumptions that EAP makes on lower layers.

3.1. Lower layer assumptions

EAP makes the following assumptions about lower layers:

[1] Unreliable transmission. With the exception of Success and Failure, EAP frames are acknowledged, so that EAP does not assume that lower layers are reliable. However, if lower layers exhibit a high loss rate, particularly where alternate indications of success or failure are not provided, then frequent timeouts are likely to result.

[2] Known MTU. EAP itself does not support fragmentation and reassembly. However, well written EAP methods SHOULD NOT make assumptions about the minimum supported MTU size, and SHOULD be capable of handling fragmentation and reassembly. As a result, EAP is capable of functioning across a range of MTU sizes, as long as the MTU is known.

[3] Infrequent duplication. While it is desirable that lower layers provide for non-duplication, this is not a requirement. This implies that reliable lower layers SHOULD provide EAP with a reliable, non-duplicated stream of packets, rather than passing duplicates up to EAP.

In EAP, the Authenticator is responsible for retransmission, and a Peer needs to be prepared to handle duplicates. On long-delay links it is possible for an Authenticator to retransmit before a Response is received. Therefore, even though EAP peers do not retransmit
Responses, Authenticators need to be prepared to handle duplicates. For example, Authenticators MUST eliminate duplicates before passing Responses on to the backend authentication server. The Identifier field provides both the Peer and Authenticator with the ability to detect duplicates.

Possible reordering. EAP is an ACK/NAK protocol, so that multiple frames will not ordinarily be in flight within a single conversation, and the next frame will not be sent until the previous one is acknowledged. However, on long-delay links it is possible for an Authenticator to retransmit before a Response is received, and where the lower layer does not provide ordering guarantees, for the retransmitted frame to arrive prior to the original one. However, retransmission reordering is relatively harmless, since the Identifier field provides the means for duplicate detection, in whatever order they arrive. As a result, EAP does not assume that the lower layer provides ordering guarantees.

3.2. EAP usage within PPP

In order to establish communications over a point-to-point link, each end of the PPP link must first send LCP packets to configure the data link during Link Establishment phase. After the link has been established, PPP provides for an optional Authentication phase before proceeding to the Network-Layer Protocol phase.

By default, authentication is not mandatory. If authentication of the link is desired, an implementation MUST specify the Authentication-Protocol Configuration Option during Link Establishment phase.

The server can use the identification of the connecting host or router in the selection of options for network layer negotiations.

When implemented within PPP, EAP does not select a specific authentication mechanism at PPP Link Control Phase, but rather postpones this until the Authentication Phase. This allows the Authenticator to request more information before determining the specific authentication mechanism. This also permits the use of a "back-end" server which actually implements the various mechanisms while the PPP Authenticator merely passes through the authentication exchange. The PPP Link Establishment and Authentication phases, and the Authentication-Protocol Configuration Option, are defined in The Point-to-Point Protocol (PPP) [RFC1661].
3.2.1. Alternate indications in PPP

As noted below, an EAP Peer SHOULD be able to make use of alternative indications of success and failure, so as to be able to survive loss of an EAP Failure or Success packet. A PPP Peer can use a Network Control Protocol (NCP) packet as an alternative indication of Success. Likewise, the receipt of a PPP LCP Terminate-Request can be taken as an alternate indication of Failure.

3.2.2. PPP Configuration Option Format

A summary of the PPP Authentication-Protocol Configuration Option format to negotiate the EAP Authentication Protocol is shown below. The fields are transmitted from left to right.

Exactly one EAP packet is encapsulated in the Information field of a PPP Data Link Layer frame where the protocol field indicates type hex C227 (PPP EAP).

<table>
<thead>
<tr>
<th>Type</th>
<th>Length</th>
<th>Authentication-Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>4</td>
<td>C227 (Hex) for PPP Extensible Authentication Protocol (EAP)</td>
</tr>
</tbody>
</table>

3.3. EAP usage within IEEE 802

The encapsulation of EAP over IEEE 802 link layers is defined in [IEEE8021X]. The IEEE 802 encapsulation of EAP does not involve PPP, and IEEE 802.1X does not include support for link or network layer negotiations. As a result, within IEEE 802.1X it is not possible to negotiate non-EAP authentication mechanisms, such as PAP or CHAP [RFC1994].

Whether authentication is mandatory is determined by the switch or access point configuration. If authentication is not required, or if the
identity of the Peer is verified solely based on the MAC address, then
the Authenticator may respond to a request for EAP authentication with a
"canned" Success message.

### 3.3.1. Alternate indications in 802

As noted below, an EAP Peer SHOULD be able to make use of alternative
indications of success and failure, so as to be able to survive loss of
an EAP Failure or Success packet. An 802 Peer can use a "port down"
event (e.g. loss of carrier) as an alternative indication of Failure.
With IEEE 802 local area networks, there is no alternative indication of
Success, so that loss of a Success packet results in a Peer timeout, and
restarting of the authentication process.

### 3.3.2. Alternate indications in 802.11

As noted below, an EAP Peer SHOULD be able to make use of alternative
indications of success and failure, so as to be able to survive loss of
an EAP Failure or Success packet. An 802.11 Peer can use a Disassociate
message as an alternative indication of Failure. With 802.11, there is
no alternative indication of Success, so that loss of a Success packet
results in a Peer timeout, and restarting of the authentication process.

### 4. EAP Packet format

A summary of the EAP packet format is shown below. The fields are
transmitted from left to right.

```
0                   1                   2                   3
                             +-------------------------------+
                             |     Code      |  Identifier   |            Length             |
                             +-------------------------------+
                             |    Data ...                          |
                             +-------------------------------+
```

**Code**

The Code field is one octet and identifies the type of EAP packet.
EAP Codes are assigned as follows:

- 1 Request
- 2 Response
- 3 Success
- 4 Failure
Identifier

The Identifier field is one octet and aids in matching Responses with Requests.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length and Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception.

Data

The Data field is zero or more octets. The format of the Data field is determined by the Code field.

4.1. Request and Response

Description

The Request packet is sent by the Authenticator to the peer. Each Request has a type field which serves to indicate what is being requested. The Authenticator MUST transmit an EAP packet with the Code field set to 1 (Request). Additional Request packets MUST be sent until a valid Response packet is received, or an optional retry counter expires. For IEEE 802.1X, the retry counter is effectively set to zero, so that retransmission never occurs, and instead the Peer times out and authentication is restarted.

Retransmitted Requests MUST be sent with the same Identifier value in order to distinguish them from new Requests. Note that the Identifier field need only be unique on a per-port basis, so that Authenticators are not restricted to only 255 simultaneous authentication conversations.

The contents of the data field is dependent on the Request type. The Peer MUST send a Response packet in reply to a Request packet. Responses MUST only be sent in reply to a received Request and never retransmitted on a timer. The Identifier field of the Response MUST match that of the Request.

Implementation Note: Because the authentication process will often involve user input, some care must be taken when deciding upon retransmission strategies and authentication timeouts. For use in PPP, it is suggested a retransmission timer of 6 seconds with a maximum of 10 retransmissions be used as default. One may wish to make these timeouts longer in certain cases (e.g. where Token...
Cards are involved). Additionally, the Peer must be prepared to silently discard received retransmissions while waiting for user input.

A summary of the Request and Response packet format is shown below. The fields are transmitted from left to right.

```
  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Code      |  Identifier   |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     Type      |  Type-Data ...|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-
```

**Code**

1 for Request;

2 for Response.

**Identifier**

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field MUST be the same if a Request packet is retransmitted due to a timeout while waiting for a Response. Any new (non-retransmission) Requests MUST modify the Identifier field. If a Peer receives a duplicate Request for which it has already sent a Response, it MUST resend its Response. If a Peer receives a duplicate Request before it has sent a Response to the initial Request (i.e. it’s waiting for user input), it MUST silently discard the duplicate Request.

**Length**

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and Type-Data fields. Octets outside the range of the Length field should be treated as Data Link Layer padding and should be ignored on reception.

**Type**

The Type field is one octet. This field indicates the Type of Request or Response. A single Type MUST be specified for each EAP Request or Response. Normally, the Type field of the Response will be the same as the Type of the Request. However, there is also a Nak Response Type for indicating that a Request type is unacceptable to
the peer. When sending a Nak in response to a Request, the Peer MUST indicate an alternative desired authentication Type which it supports. An initial specification of Types follows in a later section of this document.

Type-Data

The Type-Data field varies with the Type of Request and the associated Response.

4.2. Success and Failure

The Success packet is sent by the Authenticator to the Peer to acknowledge successful completion of an authentication method. The Authenticator MUST transmit an EAP packet with the Code field set to 3 (Success). If the Authenticator cannot authenticate the Peer (unacceptable Responses to one or more Requests) then the implementation MUST transmit an EAP packet with the Code field set to 4 (Failure). An Authenticator MAY wish to issue multiple Requests before sending a Failure response in order to allow for human typing mistakes. Success and Failure packets MUST NOT contain additional data.

Implementation Note: Because the Success and Failure packets are not acknowledged, they may be potentially lost. A peer SHOULD allow for this circumstance, by taking alternate indications of Success and Failure into account. The alternate indications are media-specific, and are described in section 3.

A summary of the Failure packet format is shown below. The fields are transmitted from left to right.

<table>
<thead>
<tr>
<th>Code</th>
<th>Identifier</th>
<th>Length</th>
</tr>
</thead>
</table>

Code

3 for Success

4 for Failure.

Identifier

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field of a Response MUST match the
Identifier field of the Request packet that it is sent in response to.

Length

4

5. Initial EAP Request/Response Types

This section defines the initial set of EAP Types used in Request/Response exchanges. More Types may be defined in follow-on documents. The Type field is one octet and identifies the structure of an EAP Request or Response packet. The first 3 Types are considered special case Types. The remaining Types define authentication exchanges. The Nak Type is valid only for Response packets, it MUST NOT be sent in a Request. The Nak Type MUST only be sent in response to a Request which uses an authentication Type code (i.e., Type > 3).

All EAP implementations MUST support Types 1-4, which are defined in this document. Follow-on RFCs MAY define additional EAP Types.

1       Identity
2       Notification
3       Nak (Response only)
4       MD5-Challenge
255       Vendor-specific

5.1. Identity

Description

The Identity Type is used to query the identity of the peer. Generally, the Authenticator will issue this as the initial Request. An optional displayable message MAY be included to prompt the Peer in the case where there expectation of interaction with a user. A Response MUST be sent to this Request with a Type of 1 (Identity).

Implementation Note: The Peer MAY obtain the Identity via user input. It is suggested that the Authenticator retry the Identity Request in the case of an invalid Identity or authentication failure to allow for potential typos on the part of the user. It is suggested that the Identity Request be retried a minimum of 3 times before terminating the authentication phase with a Failure reply. The Notification Request MAY be used to indicate an invalid authentication attempt prior to transmitting a new Identity Request (optionally, the failure MAY be indicated within the message of the new Identity Request itself).
Type

1

Type-Data

This field MAY contain a displayable message in the Request. The Response uses this field to return the Identity. If the Identity is unknown, this field should be zero bytes in length. The field MUST NOT be null terminated. The length of this field is derived from the Length field of the Request/Response packet and hence a null is not required.

5.2. Notification

Description

The Notification Type is optionally used to convey a displayable message from the Authenticator to the peer. The Peer SHOULD display this message to the user or log it if it cannot be displayed. It is intended to provide an acknowledged notification of some imperative nature. Examples include a password with an expiration time that is about to expire, an OTP sequence integer which is nearing 0, an authentication failure warning, etc. In most circumstances, notification should not be required.

Type

2

Type-Data

The Type-Data field in the Request contains a displayable message greater than zero octets in length. The length of the message is determined by Length field of the Request packet. The message MUST not be null terminated. A Response MUST be sent in reply to the Request with a Type field of 2 (Notification). The Type-Data field of the Response is zero octets in length. The Response should be sent immediately (independent of how the message is displayed or logged).

5.3. Nak

Description

The Nak Type is valid only in Response messages. It is sent in reply to a Request where the desired authentication Type is unacceptable. Authentication Types are numbered 4 and above. The Response contains
the authentication Type desired by the peer. Since the Nak Type is only valid in Responses and is used to indicate an alternate authentication method, it is not appropriate for use as a general purpose error indication. Therefore the Nak Type MUST NOT be used for communication of error messages, or negotiation of parameters specific to a particular EAP method.

Code

2 for Response.

Identifier

The Identifier field is one octet and aids in matching Responses with Requests. The Identifier field of a Response MUST match the Identifier field of the Request packet that it is sent in response to.

Length

5 or 6

Type

3

Type-Data

This field MUST contain a single octet indicating the desired authentication type.

5.4. MD5-Challenge

Description

The MD5-Challenge Type is analogous to the PPP CHAP protocol [RFC1994] (with MD5 as the specified algorithm). The Request contains a "challenge" message to the peer. A Response MUST be sent in reply to the Request. The Response MAY be either of Type 4 (MD5-Challenge) or Type 3 (Nak). The Nak reply indicates the peer’s desired authentication mechanism Type. All EAP implementations MUST support the MD5-Challenge mechanism.

Note that the use of the Identifier field in the MD5-Challenge Type is different from that described in [RFC1994]. EAP allows for retransmission of MD5-Challenge Request packets while RFC 1994 states that both the Identifier and Challenge fields MUST change each time a Challenge (the CHAP equivalent of the MD5-Challenge Request packet)
is sent.

Type

4

Type-Data

The contents of the Type-Data field is summarized below. For reference on the use of this fields see the PPP Challenge Handshake Authentication Protocol [RFC1994].

5.5. Vendor-specific

Description

Due to EAP’s popularity, the original Method Type space, which only provides for 255 values, is being allocated at a pace, which if continued, would result in exhaustion within a few years. Since many of the existing uses of EAP are vendor-specific, the Vendor-Specific Method Type is available to allow vendors to support their own extended Types not suitable for general usage. The Vendor-specific Type may also be used to expand the global Method Type space beyond the original 255 values.

Peers not equipped to interpret the Vendor-specific Type MUST send a NAK, and negotiate a more suitable authentication method.

A summary of the Vendor-specific Type format is shown below. The fields are transmitted from left to right.
Type

255 for Vendor-specific

Vendor-Id

The Vendor-Id is 3 octets and represents the SMI Network Management Private Enterprise Code of the Vendor in network byte order, as allocated by IANA. A Vendor-Id of zero is reserved for use by the IETF in providing an expanded global EAP Type space.

String

The String field is one or more octets. The actual format of the information is site or application specific, and a robust implementation SHOULD support the field as undistinguished octets.

The codification of the range of allowed usage of this field is outside the scope of this specification.

It SHOULD be encoded as follows. The Vendor-Specific field is dependent on the vendor’s definition of that attribute. An example encoding of the Vendor-Specific attribute using this method follows.

Example Implementation

```
+-----------------+-----------------+-----------------+------------------+
| Type            | Vendor-Id       |
| Vendor-Type     | Vendor-Specific |
| Vendor-Specific |
```

Vendor-Type

The Vendor-Type field is four octets and represents the vendor-specific Method Type. Where a Vendor-Id of zero is present, the Vendor-Type field provides an expanded global EAP Type space, beginning with EAP Type values of 256.

Vendor-Specific

The Vendor-Specific field is dependent on the vendor’s definition of that attribute. Where a Vendor-Id of zero is present, the Vendor-
Specific field will be used for transporting the contents of EAP Methods of Types 256 or greater.

6. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to the EAP protocol, in accordance with BCP 26, [RFC2434].

There are two name spaces in EAP that require registration: Packet Codes and Method Types.

EAP is not intended as a general-purpose protocol, and allocations should not be made for purposes unrelated to authentication.

6.1. Definition of Terms

The following terms are used here with the meanings defined in BCP 26: "name space", "assigned value", "registration".

The following policies are used here with the meanings defined in BCP 26: "Private Use", "First Come First Served", "Expert Review", "Specification Required", "IETF Consensus", "Standards Action".

6.2. Recommended Registration Policies

For registration requests where a Designated Expert should be consulted, the responsible IESG Area Director should appoint the Designated Expert.

For registration requests requiring Expert Review, the eap mailing list should be consulted.

Packet Codes have a range from 1 to 255, of which 1-4 have been allocated. Because a new Packet Code has considerable impact on interoperability, a new Packet Code requires Standards Action, and should be allocated starting at 5.

The original EAP Method Type space has a range from 1 to 255, and is the scarcest resource in EAP, and thus must be allocated with care. Method Types 1-31 have been allocated, with 20 available for re-use. Method Types 32-191 may be allocated following Expert Review, with Specification Required. Release of blocks of Method Types (more than 1 at a time for a given purpose) should require IETF Consensus. EAP Type Values 192-254 are reserved and allocation requires Standards Action.

Method Type 255 is allocated for Vendor-Specific extensions and the use of that should be encouraged instead of allocation from the original global Method Type space, for functions specific only to one vendor's
implementation of EAP, where no interoperability is deemed useful.

When used with a Vendor-Id of zero, Method Type 255 can also be used to provide for an expanded Method Type space. Expanded Method Type values 256-4294967295 may be allocated after Type values 1-191 have been allocated.

7. Normative references


8. Informative references


9. Security Considerations

Security issues are the primary topic of this RFC. Known security issues with EAP include:

- Packet modification attacks
- Down negotiation attacks
- Implementation dependence
- Mutual authentication
- Key derivation
- Separation of EAP server and Authenticator
- Weak of non-existent ciphersuites
- Dictionary attacks

9.1. Packet modification attacks

While individual EAP methods such as EAP-TLS [RFC2716] may provide for authentication and integrity protection of material sent within the data portion of an EAP message, EAP itself does not provide built-in support for per-frame integrity protection. This means that an attacker may inject or modify EAP messages, including Request and Response messages of types Identity, Notification, Nak, MD5-Challenge, Success, and Failure. In the case of PPP and IEEE 802 wired links, it is assumed that such attacks are restricted to attackers who can gain access to the physical link.

However, where EAP is run over wireless media such as 802.11, or over IP, such as within protocols supporting PPP or Ethernet tunneling [RFC2661], this assumption is no longer valid and the vulnerability to attack is much greater. As a result, when EAP is used with wireless media or over IP, the EAP conversation SHOULD be integrity protected and encrypted. This may be accomplished using mechanisms such as TLS [RFC2246] or IKE [RFC2409], as is done in PIC [PIC].

9.2. Down negotiation attacks

In practice, within or associated with each EAP server, it is anticipated that a particular named user will be authenticated by a predefined method or sequence of methods, without leaving the user any choice. Enabling negotiation would make the user vulnerable to attacks which negotiate the least secure method from among a set (such as PAP rather than EAP). This vulnerability is particularly acute where EAP runs over wireless media or IP since EAP method negotiation is unprotected.

Instead, for each named user there should be an indication of exactly one method or sequence of methods used to authenticate that user name. If a user needs to make use of different authentication methods under
different circumstances, then distinct identities SHOULD be employed, each of which identifies exactly one authentication method or sequence of methods.

9.3. Implementation dependence

The interaction of authentication protocols with link layer technologies such as PPP and IEEE 802 are highly implementation dependent.

For example, upon failure of authentication, some PPP implementations do not terminate the link, instead limiting the kind of traffic in the Network-Layer Protocols to a filtered subset, which in turn allows the user opportunity to update secrets or send mail to the network administrator indicating a problem. Similarly, while in IEEE 802.1X an authentication failure will result denied access to the controlled port, limited traffic may be permitted on the uncontrolled port.

In EAP there is no provision for retries of failed authentication. However, in PPP the LCP state machine can renegotiate the authentication protocol at any time, thus allowing a new attempt. Similarly, in IEEE 802.1X the supplicant or Authenticator can re-authenticate at any time. It is recommended that any counters used for authentication failure not be reset until after successful authentication, or subsequent termination of the failed link.

9.4. Mutual authentication

In EAP there is no requirement that authentication be full duplex or that the same protocol be used in both directions. It is perfectly acceptable for different protocols to be used in each direction. This will, of course, depend on the specific protocols negotiated. If a one-way authentication method is negotiated, such as EAP-MD5, then the Authenticator’s identity will not be verified.

For use on wireless media such as 802.11 [IEEE80211], or over the Internet, where physical security can no longer be assumed, EAP methods providing mutual authentication SHOULD be used in order to guard against rogue EAP servers.

9.5. Key derivation

This specification does not describe how keys are to be derived. However, EAP methods deriving keys SHOULD provide keying material that can be used with any ciphersuite, since the negotiated ciphersuite may not be known beforehand. In addition, EAP methods deriving keys MUST describe how authentication/integrity, encryption and IVs are to be derived from the provided keying material, and how cryptographic separation is maintained between keying material used for different
purposes. Rather than inventing new key derivation algorithms, reuse of existing algorithms such as those specified in IKE [RFC2409], or TLS [RFC2246] is RECOMMENDED.

9.6. Separation of EAP server and Authenticator

It is possible for the EAP endpoints to mutually authenticate, negotiate a ciphersuite, and derive session key(s) for subsequent use with link layer authentication, integrity protection and encryption.

This does not present an issue on the peer, since the Peer and EAP client reside on the same machine; all that is required is for the EAP client module to pass the session key to the link layer security module.

The situation is more complex when the Authenticator does not reside on the same machine as the EAP server. For example, the EAP server may be a backend security server.

In the case where the EAP server and Authenticator reside on different machines, there are several implications for security. Firstly, mutual authentication will occur between the Peer and the EAP server, not between the Peer and the Authenticator. This means that it is not possible for the Peer to validate the identity of the Authenticator.

The second issue that arises in the case of an EAP server and Authenticator residing on different machines is that the session key negotiated between the Peer and EAP server will need to be transmitted to the Authenticator. Therefore a mechanism needs to be provided to transmit the session key from the EAP server to the Authenticator that needs to use the key. The specification of this transit mechanism is outside the scope of this document.

9.7. Weak or non-existent ciphersuites

EAP authentication may be used in situations where after authentication, data packets are sent without integrity protection or confidentiality. These scenarios are inherently insecure. Without per-packet authentication an attacker with access to the media can inject packets, "flip bits" within existing packets, or even hijack the session completely. Similarly, without encryption, it is possible to snoop data packets.

As a result, it should be understood that where attackers may easily gain access to the physical medium (such as in wireless networks, or where EAP is run over the Internet), EAP MUST be used in concert with credible ciphersuites in order to provide defensible security. Due to the many of known attacks to which it is vulnerable, Wired Equivalent Privacy (WEP) [IEEE80211] does not qualify as a credible ciphersuite.
9.8. Dictionary attacks

Password authentication algorithms such as EAP-MD5, MS-CHAPv1 [RFC2433] and Kerberos V [RFC1510] are known to be vulnerable to offline dictionary attacks. MS-CHAPv1 vulnerabilities are documented in [PPTPv1]; Kerberos vulnerabilities are described in [KRBACK], [KRBLIM], and [KERB4WEAK].

As a result, EAP authentication algorithms vulnerable to offline dictionary attack SHOULD NOT be used without encrypting and integrity protecting the EAP exchange.

Acknowledgments

This protocol derives much of its inspiration from Dave Carrel’s AHA draft as well as the PPP CHAP protocol [RFC1994]. Bill Simpson provided much of the boilerplate used throughout this document. Al Rubens (Merit) also provided valuable feedback, as did Glen Zorn (Cisco) and Ashwin Palekar (Microsoft).

Authors’ Addresses

Larry J. Blunk
Merit Network, Inc.
4251 Plymouth Rd., Suite C
Ann Arbor, MI  48105
EMail: ljb@merit.edu
Phone: 734-763-6056
FAX:  734-647-3185

John R. Vollbrecht
Interlink Networks, Inc.
775 Technology Drive, Suite 200
Ann Arbor, MI  48108
USA
Phone: +1 734 821 1205
Fax: +1 734 821 1235
EMail: jrv@interlinknetworks.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
EMail: bernarda@microsoft.com
Phone: +1 425 706 6605
Open issues

This specification has quite a few open issues. Some not otherwise noted include:

[1] What if a Nak contains an alternative authentication type that the Authenticator does not support?

[2] The specification contradicts itself with respect to the legality of a Nak message containing no Type-Data field (length = 5). Is this ok or not?

[3] How does an Authenticator determine the correct language to use in a Notification? Is there a way for a Peer to indicate that it would like Notifications to be sent in a particular language?

Expiration Date

This memo is filed as <draft-ietf-pppext-rfc2284bis-02.txt>, and expires August 24, 2002.