PPPEXT Working Group
INTERNET-DRAFT
Category: Standards Track
<draft-ietf-pppext-rfc2284bis-06.txt>
13 September 2002
Obsoletes: RFC 2284

L. Blunk
Merit Networks, Inc.
J. Vollbrecht
Interlink Networks, Inc.
Bernard Aboba
Microsoft

Extensible Authentication Protocol (EAP)

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 multiplexing model 6
3. Media-specific issues 7
3.1 Lower layer assumptions 7
3.2 EAP usage within PPP 8
3.3 EAP usage within IEEE 802 9
4. EAP Packet Format 10
4.1 Request and Response 11
4.2 Success and Failure 13
5. Initial EAP Request/Response Types 14
5.1 Identity 14
5.2 Notification 15
5.3 Nak 16
5.4 MD5-Challenge 17
5.5 Vendor-specific 17
6. IANA Considerations 19
6.1 Definition of Terms 19
6.2 Recommended Registration Policies 20
7. Security considerations 20
7.1 Threat model 21
7.2 Security requirements 21
7.3 Identity protection 23
7.4 Packet modification attacks 23
7.5 Denial of service attacks 24
7.6 Dictionary attacks 24
7.7 Connection to an untrusted network 25
7.8 Negotiation attacks 25
7.9 Implementation idiosyncracies 26
7.10 Key derivation 26
7.11 Weak ciphersuites 27
8. Normative references 27
9. Informative references 28
ACKNOWLEDGMENTS 29
AUTHORS' ADDRESSES 29
Full Copyright Statement 30

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.

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 and access points using IEEE 802 [IEEE802]. 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 backend server which actually implements the various mechanisms while the Authenticator merely passes through the authentication exchange.

The authentication exchange proceeds as follows:

[1] The Authenticator sends a Request 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; however, an initial Identity Request is not required, and MAY be bypassed. For example, the identity may not be required where it is requested within the EAP method itself, which is pre-determined; where it is determined by the port to which the Peer has connected (leased lines, dedicated switch or dial-up ports); or where the identity is 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 the 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 sends an additional Request packet, and the Peer replies with a Response. The sequence of Requests and Responses continues as long as needed.

[4] If the initial authentication method completes unsuccessfully, then the Authenticator sends a Failure packet to the Peer. If it completes successfully, and the Authenticator does not require additional authentication methods, then the Authenticator sends a Success packet to the Peer.

[5] 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 an MD5-Challenge. If additional authentication methods are required, the Authenticator MAY send a Request packet for a subsequent authentication method, or it MAY send another Identity request. The Peer will then respond with a Response packet containing a type field matching the Request.

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

Whether authentication is mandatory is determined by the Authenticator

and Peer configurations. If authentication is not required by the Authenticator, or if the identity of the Peer is verified by another mechanism (e.g. Calling-Station-ID or MAC address), then the Authenticator MAY send a "canned" Success message.

Where the Authenticator acts as a pass-through, it MUST determine the outcome of the authentication solely based on the Accept/Reject indication sent by the backend authentication server; the outcome MUST NOT be determined by the contents of an EAP packet sent along with the Accept/Reject indication, or the absence of such a packet.

On the Peer, the determination of whether the authentication has succeeded or failed is based on Peer configuration, and the EAP method that has been selected, and not merely by receipt of a Success and Failure message. If the Peer is configured to require an EAP method providing mutual authentication, then the Peer MUST silently discard a Success packet sent by the Authenticator, prior to the conclusion of mutual authentication. Since the Success message is not protected, it may be spoofed, or sent by a rogue Authenticator seeking to avoid having to authenticate to the Peer. It should be understood that nothing within the EAP specification precludes the Peer from silently discarding such a premature Success packet, and it is the responsibility of individual EAP methods to define the ponit at which a Peer may accept a Success from the Authenticator.

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 act as a pass-through agent for a backend authentication server. Separation of the Authenticator from the backend authentication server simplifies credentials management and policy decision making. However, support for pass-through is not required.

Disadvantages

For use in PPP, EAP does require the addition of a new authentication

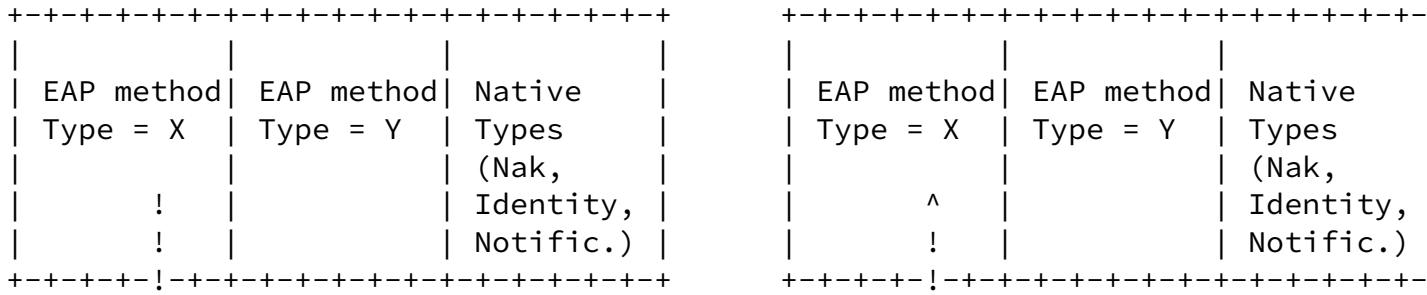
type to PPP LCP and thus PPP implementations will need to be modified to use it. It also strays from the previous PPP authentication model of negotiating a specific authentication mechanism during LCP. Similarly, switch or access point implementations need to support [IEEE8021X] in order to use EAP.

Where the Authenticator is separate from the backend authenticator server, this complicates the security analysis and, if needed, key distribution.

2.1. EAP multiplexing model

Within EAP, the Type field functions much like a port number in UDP or TCP. It is assumed that EAP implementations multiplex incoming EAP packets according to their Type, and deliver them only to the EAP method corresponding to that Type code. Peers MUST respond to an EAP Request for an unsupported Type with a Nak Response. Authenticators receiving an EAP Request with an unsupported Type MUST silently discard the Response.

EAP packets of Types Identity, Notification and Nak are assumed to be handled by methods specific to those Types. EAP packets with codes of Success or Failure do not include a Type, and therefore are not delivered to an EAP method. As a result, these messages MUST NOT used to carry data destined for delivery to other EAP methods. The architecture is illustrated in in figure 1 on the next page.



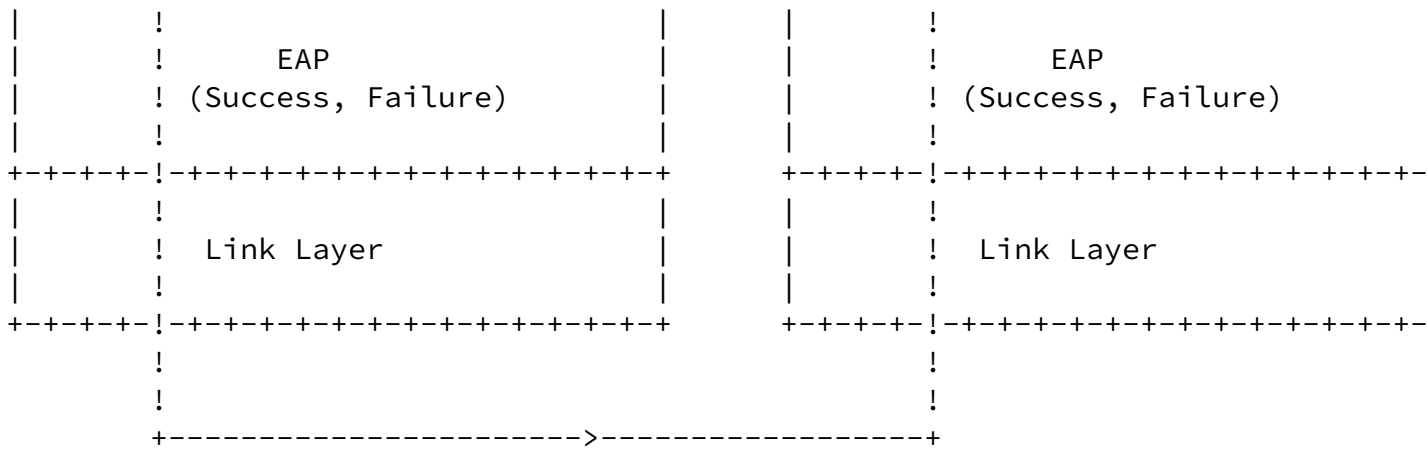


Figure 1. EAP Multiplexing Model

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 [IEEE802.11]; UDP (L2TP [RFC2661], PIC [PIC]) and TCP [PIC]. This section discusses EAP dependencies 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 packets are acknowledged, so that EAP does not assume that lower layers are reliable. However, if lower layers exhibit a high loss rate, then frequent timeouts are likely to result. Since EAP has its own retransmission behavior, when run over a reliable lower layer, it is possible for retransmission to occur both at the lower layer and the EAP layer.

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

[3] Possible duplication. While it is desirable that lower layers

provide for non-duplication, this is not a requirement. The Identifier field provides both the Peer and Authenticator with the ability to detect duplicates.

Where the lower layer is reliable, it will provide the EAP layer with a non-duplicated stream of packets.

In EAP, the Authenticator is responsible for retransmission. Although EAP Peers do not retransmit Responses, they do respond to duplicate Requests. Since it is possible for the Authenticator to retransmit before receiving a Response from the Peer, Authenticators can receive duplicate Responses.

[4] Possible reordering. EAP provides its own mechanisms to detect reordering and so does not assume that the lower layer provides ordering guarantees. EAP is a "lockstep" protocol, so that the Authenticator MUST NOT send a new Request until a Response is received to an outstanding Request. Since the Peer should not expect a new Request until it has sent a Response, if it receives a new Request with a different Identifier before sending a Response to the outstanding Request, the new Request MUST be silently discarded. Similarly, if the Authenticator receives a Response with an Identifier that does not match the Identifier in the outstanding Request, the Response MUST be silently discarded.

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.

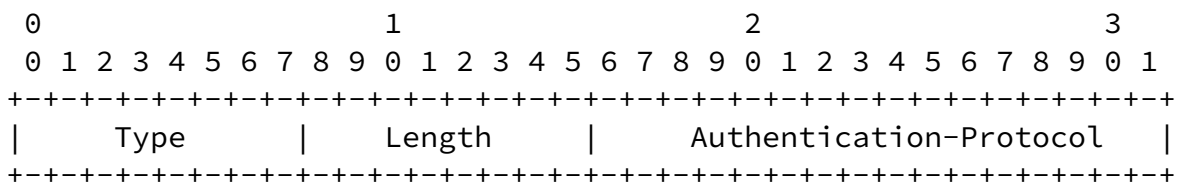
When implemented within PPP, EAP does not select a specific authentication mechanism at PPP Link Control Phase, but rather postpones this until the Authentication Phase. This allows the Authenticator to request more information before determining the specific authentication mechanism. This also permits 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. 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).



Type

3

Length

4

Authentication-Protocol

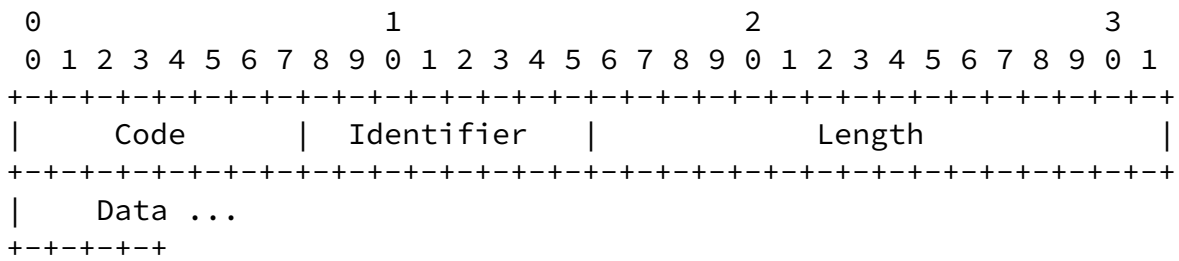
C227 (Hex) for PPP Extensible Authentication Protocol (EAP)

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].

4. EAP Packet format

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



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

Since EAP only defines Codes 1-4, EAP packets with other codes MUST be silently discarded by both Authenticators and Peers.

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.

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. 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.

Retransmitted Requests MUST be sent with the same Identifier value in order to distinguish them from new Requests. In order to avoid confusion between new Requests and retransmissions, the Identifier value chosen for each new Request MUST be unique within the EAP conversation. One way to achieve this is to start the Identifier at an initial value and increment it for each new Request. If the Authenticator receives a Response with an Identifier different from the Identifier within the last Request, then the message MUST be silently discarded. Since it is possible that the Authenticator may

retransmit prior to receiving a Response from the Peer, the Authenticator can receive duplicate responses. 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.

Implementation Note: Because the authentication process will often involve user input, some care must be taken when deciding upon retransmission strategies and authentication timeouts. By default where EAP is run over an unreliable lower layer, the EAP retransmission timer (EAP_RTO) SHOULD be computed as described in [RFC2988]. A maximum of 3-5 retransmissions is suggested.

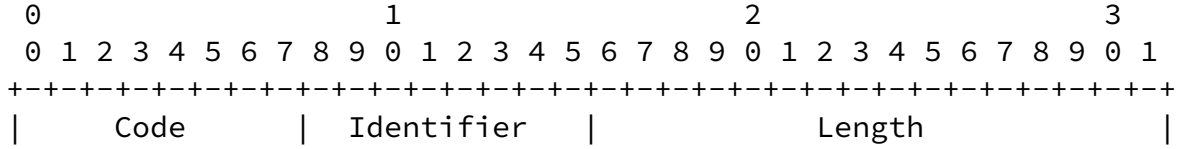
When run over a reliable lower layer (e.g. EAP over ISAKMP/TCP, as within [PIC]), the EAP retransmission timer SHOULD be set to an infinite value, so that retransmissions do not occur at the EAP

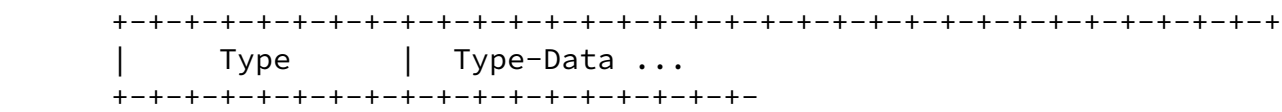
layer.

However, where the authentication process requires user input, the measured round trip times will have more to do with user responsiveness than network characteristics, so that dynamic estimation of EAP_RTO is not helpful. Instead, the retransmission timers SHOULD be set so as to provide sufficient time for the user to respond, with longer timeouts required in certain cases, such as where Token Cards are involved.

This can be accomplished by allowing the timeout value to be set by the EAP method and communicated to the Authenticator by the backend authentication server (such as via the RADIUS Session-Timeout attribute). Additionally, the Peer MUST be prepared to silently discard received retransmissions while waiting for user input, so as to mitigate the ill effects of a too small retransmission timer.

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





Code

1 for Request
2 for Response

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, 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. 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.

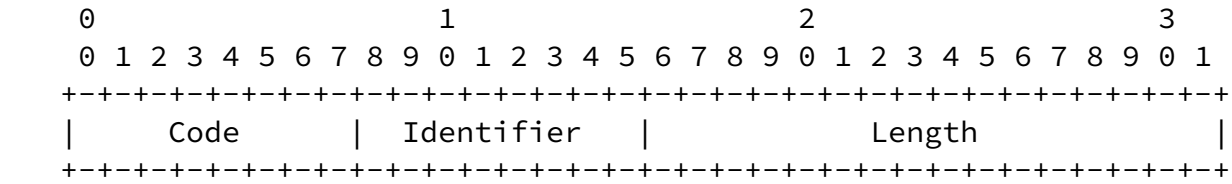
4.2. Success and Failure

The Success packet is sent by the Authenticator to the Peer to acknowledge successful completion of the authentication conversation. 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, the Authenticator cannot know whether they have been received. As a result, these packets are not retransmitted by the Authenticator, and if they are lost, the Peer will timeout. If acknowledged success and failure indications are desired, these MAY be implemented within individual EAP methods.

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



Code

3 for Success
4 for Failure

Identifier

The Identifier field is one octet, and aids in matching Responses with Requests. In order to avoid confusion between Success or Failure packets and retransmissions, the Identifier value chosen for a Success or Failure packet MUST be unique within the EAP conversation.

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, and SHOULD support Type 255. 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. The Authenticator will typically issue this as the initial Request. An optional displayable message MAY be included to prompt the Peer in the case where there is an 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, containing UTF-8 encoded 10646 characters [RFC2279]. 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, containing UTF-8 encoded 10646 characters [RFC2279]. 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 zero or more authentication Types desired by the peer. A Nak with no authentication Type(s) indicates that the Peer does not wish to authenticate using the proposed method but is not proposing an alternative.

Since the Nak Type is only valid in Responses and has very limited functionality, it MUST NOT be used as a general purpose error indication, such as 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 Nak Response MUST match the Identifier field of the Request packet that it is sent in response to.

Length

>=5

Type

3

Type-Data

This field MUST contain zero or more octets indicating the desired authentication Types, in order of preference, with the most preferred method first. In order to avoid an interminable negotiation, the Peer MUST only include Types that it is willing to accept.

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 zero or more of the peer's desired authentication Types. All EAP implementations MUST support the MD5-Challenge mechanism.

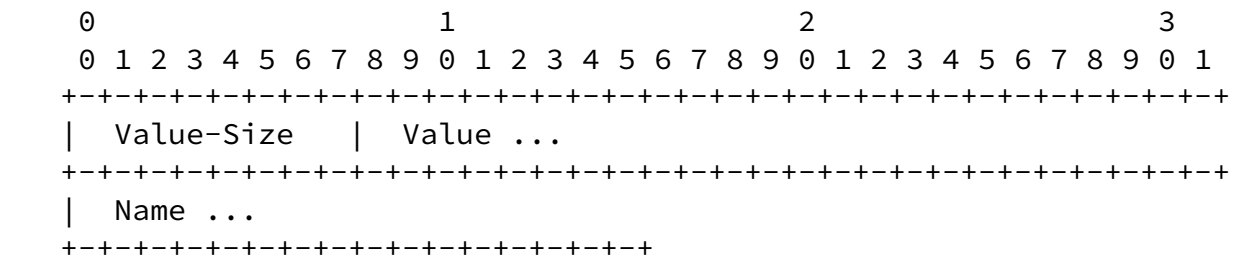
Note that the use of the Identifier field in the MD5-Challenge Type is different from that described in [RFC1994]. EAP allows for retransmission of MD5-Challenge Request packets while RFC 1994 states that both the Identifier and Challenge fields MUST change each time a Challenge (the CHAP equivalent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

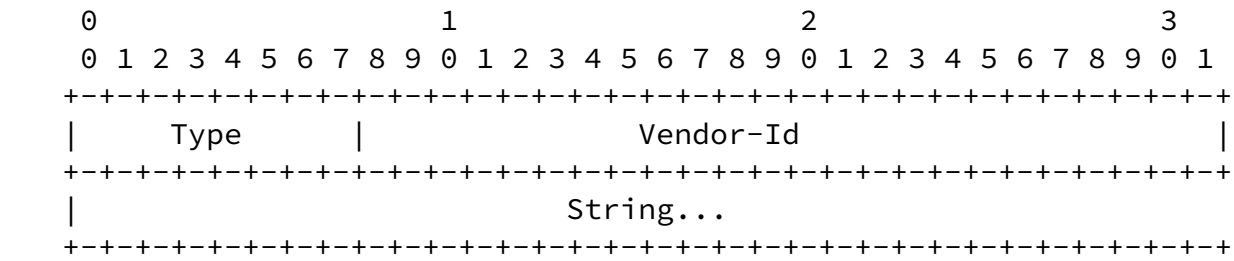
Description

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.

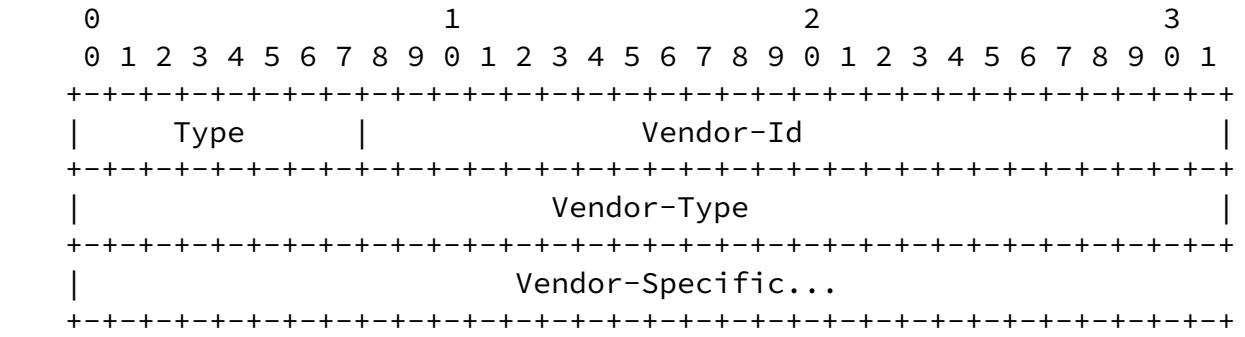
String

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



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 1-36 have been allocated, with 20 available for re-use. Method Types 37-191 may be allocated following Expert Review, with Specification Required. Release of blocks of Method Types (more than one 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. Security Considerations

EAP was designed for use with dialup PPP [RFC1661] and wired [IEEE802] networks such as Ethernet [IEEE8023]. On these networks, an attacker would need to gain physical access to the telephone or switch infrastructure in order to mount an attack. While such attacks have been documented, such as in [DECEPTION], they are assumed to be rare.

However, subsequently EAP has been proposed for use on wireless networks, and over the Internet, where physical security cannot be assumed. On such networks, the security vulnerabilities are greater, as are the requirements for EAP security.

This section documents the threats that exist on physically insecure networks carrying EAP, as well as laying out the security requirements for EAP use on those networks. We then discuss mechanisms by which the threats may be mitigated.

7.1. Threat model

On physically insecure networks, it is possible for an attacker to gain access to the physical medium. This enables a range of attacks, including the following:

[1] An adversary may try to discover user identities by snooping data packets.

[2] An adversary may try to modify or spoof EAP packets.

[3] An adversary may launch denial of service attacks by terminating

EAP conversations.

[4] An adversary might attempt to recover the passphrase by mounting an offline dictionary attack.

[5] An adversary may attempt to convince the Peer to connect to an untrusted network.

[6] An adversary may attempt to disrupt the EAP negotiation in order to weaken the authentication, gain access to user passwords or remove confidentiality protection.

[7] An adversary may attempt to mount a denial of service attack by modify

[8] An attacker may attempt to take advantage of weak key derivation techniques used within EAP methods.

[9] An attacker may attempt to take advantage of weak ciphersuites subsequently used after the EAP conversation is complete.

Where EAP is used over wireless networks, an attacker needs to be within the coverage area of the wireless medium in order to carry out these attacks. However, where EAP is used over the Internet, no such restrictions apply.

7.2. Security requirements

In order to address the threats that exist where EAP is used on a physically insecure medium, the following requirements are imposed:

[1] Mutual authentication. Mutual authentication of the communication endpoints MUST be provided in order to protect against rogue Authenticators.

[2] Protected conversation. On a physically insecure network, EAP messages SHOULD be integrity and replay protected, authenticated and confidential so as to protect against downgrade attacks, snooping of identities, and spoofing of packets. This includes protection of packets of types Identity, Nak and Notification, as

well as packets sent within the EAP method itself, and success and failure indications. Where EAP is used for ciphersuite or capabilities negotiation, these messages SHOULD be integrity and replay protected, authenticated and confidential.

[3] Key derivation. EAP methods used on physically insecure networks MAY derive keys in order to enable per-packet authentication, integrity and replay protection as well as confidentiality. Where EAP methods derive keys, the distributed keys SHOULD be master session keys, used only for further key derivation, independent of the ciphersuite. This eliminates the need for an EAP method to understand how to derive keys for every ciphersuite. Rather than inventing new key derivation techniques, well analyzed algorithms SHOULD be used.

[4] Dictionary attack resistance. Where EAP is used on physically insecure networks resistance against dictionary attack SHOULD be provided. Where password authentication is used, users are notoriously prone to selection of poor passwords. Without dictionary attack protection, it is easy for an attacker snooping authentication traffic to gather a large number of authentication exchanges, and successfully obtain a substantial fraction of the passwords used in those exchanges via a dictionary attack. Given the steadily declining prices of computing power, successful dictionary attacks can now be mounted at minimal expense.

[5] Support for fast reconnect. On physically insecure media such as wireless, it is often desirable to improve scalability and minimize connectivity interruptions due to authentication. Where this is desired, EAP methods MAY support "fast reconnect". After an initial authentication conversation, this enables subsequent authentication conversations to take place in shortened form.

[6] Acknowledged success and failure indications. Where EAP is used over an unreliable medium, it is possible for packets to be lost. This can result in the Peer and Authenticator having a different interpretation of the state of the authentication conversation. As a result, where EAP is used over an unreliable medium, EAP methods SHOULD support acknowledged success and failure indications.

Since proposed EAP methods may be used on physically insecure methods, it is necessary to be able to evaluate methods against the above requirements in order to determine their suitability. In order to be

suitable for publication as an RFC, EAP method specifications MUST include the following:

[a] Indication of intended use. This includes a statement of whether the method is intended for use over a physically secure or insecure network, as well as a statement of the applicable media.

[b] Indication of security claims. This includes a statement of the claimed security properties of the method. In particular, the specification MUST indicate whether the method claims to satisfy the requirements described above.

[c] Description of key hierarchy. EAP methods deriving keys MUST describe how keys for 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.

[d] Indication of vulnerabilities. If the method is intended for use on a physically insecure network, yet does not satisfy the above requirements, the specification MUST indicate which requirements are not satisfied, and discuss the security implications.

7.3. Identity protection

An Identity exchange is an optional within the EAP conversation. Therefore, it is possible to omit the Identity exchange entirely, or to postpone it until later in the conversation once a protected channel has been negotiated.

However, within networks where backend authentication proxies or relays are present, it may be necessary to locate the appropriate backend authentication server before the authentication conversation can proceed. The realm portion of the Network Access Identifier (NAI) [RFC2486] is typically included within the Response-Identity in order to enable the authentication exchange to be routed to the appropriate backend authentication server. Therefore while the user-name portion of the NAI may be omitted in the Identity-Response, where proxies or relays are present, the realm portion may be required.

7.4. Packet modification attacks

While individual EAP methods such as EAP TLS [RFC2716] may provide for authentication, integrity and replay protection of data placed within the Type-Data portion of EAP Request and Response packets, EAP itself does not provide built-in support for per-packet data origin authentication, replay or integrity protection.

This means that an attacker may inject or modify EAP packets, including Request and Response packets of Types Identity, Notification, Nak, MD5-Challenge as well as Success and Failure packets. An attacker may also modify the EAP headers within EAP packets where the Type-Data field is protected, but not the EAP headers.

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 IEEE 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.

To provide protection for EAP messages sent over physically insecure media, a variety of techniques may be used. This includes encapsulation of EAP within ISAKMP [RFC2408], as is done in PIC [PIC], or within TLS [RFC2246].

7.5. Denial of service attacks

EAP headers are unprotected, as are packets of Types Identity, Nak, Notification as well as Success and Failure. Since the Identifier is only a single octet, and implementations typically start it at zero (0), incrementing with each new Request, it is very easy to guess. This makes it easy for an attacker to deny service by spoofing Failure packets, as well as by inserting Nak packets so as to prolong the method negotiation. It is also possible for the attacker to attempt to replay packets from previous conversations.

Several mechanisms are recommended for addressing this vulnerability:

[a] Silent discard. Since only a single EAP Request can be in progress between an Authenticator and a Peer at a given time, if a Peer receives a new Request before sending a Response, the new Request can be silently discarded. This increases resilience against spoofed Requests. Similarly, an Authenticator can silently discard Responses with Identifiers that do not correspond to the Identifier included in the last Request, or that represent duplicate Responses.

[b] Where the EAP conversation is protected via per-packet data origin authentication, integrity and replay protection, spoofing or

data modification attacks can be detected.

7.6. Dictionary attacks

Password authentication algorithms such as EAP-MD5, MS-CHAPv1 [RFC2433] and Kerberos V [RFC1510] are known to be vulnerable to dictionary

attacks. MS-CHAPv1 vulnerabilities are documented in [PPTPv1]; Kerberos vulnerabilities are described in [KRBATTACK], [KRBLIM], and [KERB4WEAK].

In order to protect against dictionary attacks, an authentication algorithm resistant to dictionary attack may be used. If an authentication algorithm is used that is known to be vulnerable to dictionary attack, then the authentication may be encrypted to obtain additional protection.

7.7. Connection to an untrusted network

Lf a one-way authentication method is negotiated, such as EAP-MD5, then the Authenticator's identity will not be verified. While this might be acceptable over a link that is believed to be physically secure, this level of security is not acceptable where EAP runs over wireless media or IP, since this leaves the Peer vulnerable to connection to an untrusted network. As a result, where EAP is used over a physically insecure network, it is highly desirable to use a method supporting 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. However, in general, completing a single, mutual authentication is preferrable to two one-way authentications, one in each direction.

7.8. Negotiation attacks

In a negotiation attack, the attacker attempts to convince the Peer and Authenticator to negotiate a less secure EAP method.

To avoid downgrading from a mutually authenticating method to one that only authenticates the peer, where EAP is used over a physically insecure network, it is desirable for the peer to only accept

negotiation of a mutually authenticating method.

In practice, within or associated with each backend authentication 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 EAP-MD5 instead of a mutually authenticating method).

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.

7.9. Implementation idiosyncracies

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.

7.10. Key derivation

It is possible for the EAP endpoints to mutually authenticate, negotiate a ciphersuite, and derive session key(s) for subsequent use with per-

packet authentication, integrity protection and encryption. Since the Peer and EAP client reside on the same machine, the EAP client module passes the derived session key to the link layer security module.

In the case where the backend authentication server and Authenticator reside on different machines, there are several implications for security:

[a] Mutual authentication will occur between the Peer and the backend authentication 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.

[b] The session key negotiated between the Peer and backend authentication server will need to be transmitted to the Authenticator. Therefore a mechanism needs to be provided to transmit the session key from the backend authentication server to the Authenticator that needs to use the key. The specification of this transit mechanism is outside the scope of this document.

This specification does not provide guidance on how EAP methods are to derive keys. Key derivation is an art that is best practiced by professionals; rather than inventing new key derivation algorithms, reuse of existing algorithms such as those specified in IKE [RFC2409], or TLS [RFC2246] is recommended.

7.11. Weak ciphersuites

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

On physically insecure media, EAP SHOULD be used in concert with credible ciphersuites in order to provide defensible security. In particular, EAP SHOULD NOT be used for authentication without subsequent invocation of per-packet integrity protection and authentication, since

this would leave the session vulnerable to hijacking. Due to the many of known attacks to which it is vulnerable, Wired Equivalent Privacy (WEP) [IEEE80211] does not qualify as a credible ciphersuite.

8. Normative references

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

[RFC2044] Yergeau, F., "UTF-8, a transformation format of Unicode and ISO 10646", RFC 2044, October 1996.

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

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

[RFC2434] Alvestrand, H. and Narten, T., "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2988] Paxson, V., Allman, M., "Computing TCP's Retransmission Timer", RFC 2988, November 2000.

[IEEE802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, ANSI/IEEE Std 802, 1990.

[IEEE8021X] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std 802.1X-2001, June 2001.

9. Informative references

[DECEPTION] Slatalla, M., and Quittner, J., "Masters of Deception." HarperCollins, New York, 1995.

[RFC1510] Kohl, J., Neuman, C., "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, November 1998.

[RFC2486] Beadles, M., Aboba, B., "The Network Access Identifier", RFC 2486, January 1999.

[RFC2401] Atkinson, R., Kent, S., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2408] Maughan, D., Schertler, M., Schneider, M., Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[RFC2433] Zorn, G., Cobb, S., "Microsoft PPP CHAP Extensions", RFC 2433, October 1998.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and Palter, B., "Layer Two Tunneling Protocol L2TP", RFC 2661, August 1999.

[RFC2716] Aboba, B., Simon, D.,"PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[KRBATTACK] Wu, T., "A Real-World Analysis of Kerberos Password Security", Stanford University Computer Science Department, http://theory.stanford.edu/~tjw/krbpass.html

[KRBLIM] Bellovin, S.M., Merritt, M., "Limitations of the kerberos authentication system", Proceedings of the 1991 Winter USENIX Conference, pp. 253-267, 1991.

[KERB4WEAK] Dole, B., Lodin, S., and Spafford, E., "Misplaced trust: Kerberos 4 session keys", Proceedings of the Internet

Society Network and Distributed System Security Symposium, pp. 60-70, March 1997.

[PIC] Sheffer, Y., Krawczyk, H., Aboba, B., "PIC, A Pre-IKE Credential Provisioning Protocol", Internet draft (work in progress), draft-ietf-ipsra-pic-05.txt, February 2002.

[PPTPv1] Schneier, B, Mudge, "Cryptanalysis of Microsoft's Point-to- Point Tunneling Protocol", Proceedings of the 5th ACM Conference on Communications and Computer Security, ACM Press, November 1998.

[IEEE8023] ISO/IEC 8802-3 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, (also ANSI/IEEE Std 802.3-1996), 1996.

[IEEE80211] Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1997, 1997.

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

Full Copyright Statement

Open issues

Open issues relating to this specification are tracked on the following web site:

http://www.drizzle.com/~aboba/EAP/eapissues.html

Expiration Date

This memo is filed as <draft-ietf-pppext-rfc2284bis-06.txt>, and expires April 24, 2003.