PPP EAP GSS Authentication Protocol

1. Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

This document is an Internet-Draft. 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.

To view the entire list of current Internet-Drafts, please check the "1id-abstracts.txt" listing contained in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), ftp.nordu.net (Northern Europe), ftp.nis.garr.it (Southern Europe), munnari.oz.au (Pacific Rim), ftp.ietf.org (US East Coast), or ftp.isi.edu (US West Coast).

The distribution of this memo is unlimited.

2. Copyright Notice

Copyright (C) The Internet Society (1999). All Rights Reserved.

3. Abstract

The Point-to-Point Protocol (PPP) provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP also defines an extensible Link Control Protocol (LCP), which can be used to negotiate authentication methods, as well as an Encryption
Control Protocol (ECP), used to negotiate data encryption over PPP links, and a Compression Control Protocol (CCP), used to negotiate compression methods. The Extensible Authentication Protocol (EAP) provides a standard mechanism for support of additional authentication methods within PPP. Through the use of EAP, support for a number of authentication schemes may be added, including smart cards, Kerberos, Public Key, One Time Passwords, and others.

It is desirable to support GSS_API authentication methods within EAP, since this permits developers creating GSS_API compliant authentication methods to leverage their development efforts. This document describes how EAP-GSS, which includes support for fragmentation and reassembly, supports the use of GSS_API mechanisms within EAP. GSS_API provides for the negotiation of authentication methods through use of the SPNEGO mechanism. As a result, any GSS_API mechanism supported by SPNEGO can be used with EAP-GSS.

4. Introduction

The Point-to-Point Protocol (PPP), described in [1], provides a standard method for transporting multi-protocol datagrams over point-to-point links. PPP also defines an extensible Link Control Protocol (LCP) [3], which can be used to negotiate authentication methods, as well as an Encryption Control Protocol (ECP) [6], used to negotiate data encryption over PPP links, and a Compression Control Protocol (CCP) [13], used to negotiate compression methods. The Extensible Authentication Protocol (EAP) [5], provides a standard mechanism for support of additional authentication methods within PPP. Through the use of EAP, support for a number of authentication schemes may be added, including public key [12], smart cards, Kerberos, One Time Passwords, and others.

It is desirable to support GSS_API authentication methods within EAP, since this permits developers creating GSS_API compliant authentication methods to leverage their development efforts. This document describes how EAP-GSS, which includes support for fragmentation and reassembly, supports the use of GSS_API mechanisms within EAP. GSS_API, described in [15], provides for the negotiation of authentication methods through use of the SPNEGO mechanism, described in [20]. As a result, any GSS_API mechanism supported by SPNEGO can be used with EAP-GSS.

4.1. Requirements language

In this document, the key words "MAY", "MUST", "MUST NOT", "optional", "recommended", "SHOULD", and "SHOULD NOT", are to be interpreted as described in [11].
5. Protocol overview

5.1. Overview of the EAP-GSS conversation

As described in [5], the EAP-GSS conversation will typically begin with the authenticator and the peer negotiating EAP. The authenticator will then typically send an EAP-Request/Identity packet to the peer, and the peer will respond with an EAP-Response/Identity packet to the authenticator, containing the peer’s userId.

> From this point forward, while nominally the EAP conversation occurs between the PPP authenticator and the peer, the authenticator MAY act as a passthrough device, with the EAP packets received from the peer being encapsulated for transmission to a RADIUS server or backend security server. In the discussion that follows, we will use the term "EAP server" to denote the ultimate endpoint conversing with the peer.

Once having received the peer’s Identity, the EAP server MUST respond with an EAP-GSS/Start packet, which is an EAP-Request packet with EAP-Type=EAP-GSS, the Start (S) bit set, and no data. The EAP-GSS conversation will then begin, with the peer sending an EAP-Response packet with EAP-Type=EAP-GSS. The data field of that packet will encapsulate a GSS_API token.

The EAP server will then respond with an EAP-Request packet with EAP-Type=EAP-GSS. The data field of this packet will encapsulate a GSS_API token.

Note that a client with valid GSS_API credentials may choose to reuse those credentials as part of the GSS_API authentication. This allows for improved efficiency in the case where a client repeatedly attempts to authenticate to an EAP server within a short period of time. This may have application in cases such as multilink authentication.

As a result, it is left up to the peer whether to attempt to reuse credentials, thus shortening the EAP-GSS conversation. Typically the peer’s decision will be made based on the time elapsed since the previous authentication attempt to that EAP server. Based on the expiration time of the credentials, the EAP server will decide whether to allow the reuse.

In the case where the EAP server and authenticator reside on the same device, then client will only be able to reuse credentials when connecting to the same NAS or tunnel server. Should these devices be set up in a rotary or round-robin then it may not be possible for the peer to know in advance the authenticator it will be connecting to, and therefore which credentials to attempt to reuse. As a result, it is likely that the reuse attempt will fail. In the case where the EAP
authentication is remoted then reuse is much more likely to be successful, since multiple NAS devices and tunnel servers will remote their EAP authentications to the same RADIUS server.

If the EAP server authenticates successfully, the peer MUST send an EAP-Response packet of EAP-Type=EAP-GSS, and no data. The EAP-Server then MUST respond with an EAP-Success message.

5.2. Retry behavior

As with other EAP protocols, the EAP server is responsible for retry behavior. This means that if the EAP server does not receive a reply from the peer, it MUST resend the EAP-Request for which it has not yet received an EAP-Response. However, the peer MUST NOT resend EAP-Response packets without first being prompted by the EAP server.

For example, if the initial EAP-GSS start packet sent by the EAP server were to be lost, then the peer would not receive this packet, and would not respond to it. As a result, the EAP-GSS start packet would be resent by the EAP server. Once the peer received the EAP-GSS start packet, it would send an EAP-Response encapsulating the client_hello message. If the EAP-Response were to be lost, then the EAP server would resend the initial EAP-GSS start, and the peer would resend the EAP-Response.

As a result, it is possible that a peer will receive duplicate EAP-Request messages, and may send duplicate EAP-Responses. Both the peer and the EAP-Server should be engineered to handle this possibility.

5.3. Fragmentation

It is possible that EAP-GSS messages may exceed the PPP MTU size, the maximum RADIUS packet size of 4096 octets, or even the Multilink Maximum Received Reconstructed Unit (MRRU). As described in [2], the multilink MRRU is negotiated via the Multilink MRRU LCP option, which includes an MRRU length field of two octets, and thus can support MRRUs as large as 64 KB.

However, note that in order to protect against reassembly lockup and denial of service attacks, it may be desirable for an implementation to set a maximum size for a GSS_API token. Since a typical certificate chain is rarely longer than a few thousand octets, and no other field is likely to be anywhere near as long, a reasonable choice of maximum acceptable message length might be 64 KB.

If this value is chosen, then fragmentation can be handled via the multilink PPP fragmentation mechanisms described in [2]. While this is desirable, there may be cases in which multilink or the MRRU LCP option cannot be negotiated. As a result, an EAP-GSS implementation MUST
provide its own support for fragmentation and reassembly.

Since EAP is a simple ACK-NAK protocol, fragmentation support can be added in a simple manner. In EAP, fragments that are lost or damaged in transit will be retransmitted, and since sequencing information is provided by the Identifier field in EAP, there is no need for a fragment offset field as is provided in IPv4.

EAP-GSS fragmentation support is provided through addition of a flags octet within the EAP-Response and EAP-Request packets, as well as a GSS Message Length field of four octets. Flags include the Length included (L), More fragments (M), and EAP-GSS Start (S) bits. The L flag is set to indicate the presence of the four octet GSS Message Length field, and MUST be set for the first fragment of a fragmented GSS message or set of messages. The M flag is set on all but the last fragment. The S flag is set only within the EAP-GSS start message sent from the EAP server to the peer. The GSS Message Length field is four octets, and provides the total length of the GSS_API token or set of messages that is being fragmented; this simplifies buffer allocation.

When an EAP-GSS peer receives an EAP-Request packet with the M bit set, it MUST respond with an EAP-Response with EAP-Type=EAP-GSS and no data. This serves as a fragment ACK. The EAP server MUST wait until it receives the EAP-Response before sending another fragment. In order to prevent errors in processing of fragments, the EAP server MUST increment the Identifier field for each fragment contained within an EAP-Request, and the peer MUST include this Identifier value in the fragment ACK contained within the EAP-Reponse. Retransmitted fragments will contain the same Identifier value.

Similarly, when the EAP server receives an EAP-Response with the M bit set, it MUST respond with an EAP-Request with EAP-Type=EAP-GSS and no data. This serves as a fragment ACK. The EAP peer MUST wait until it receives the EAP-Request before sending another fragment. In order to prevent errors in the processing of fragments, the EAP server MUST use increment the Identifier value for each fragment ACK contained within an EAP-Request, and the peer MUST include this Identifier value in the subsequent fragment contained within an EAP-Reponse.

5.4. Identity verification

As part of GSS_API, it is possible that the server may present a certificate to the peer, or that the peer may present a certificate to the server. If the peer has made a claim of identity in the EAP-Response/Identity (MyID) packet, the EAP server SHOULD verify that the claimed identity corresponds to the certificate presented by the peer. Typically this will be accomplished either by placing the userId within the peer certificate, or by providing a mapping between the peer
certificate and the userId using a directory service.

Similarly, the peer MUST verify the validity of the EAP server certificate, and SHOULD also examine the EAP server name presented in the certificate, in order to determine whether the EAP server can be trusted. Please note that in the case where the EAP authentication is remoted that the EAP server will not reside on the same machine as the authenticator, and therefore the name in the EAP server’s certificate cannot be expected to match that of the intended destination. In this case, a more appropriate test might be whether the EAP server’s certificate is signed by a CA controlling the intended destination and whether the EAP server exists within a target sub-domain.

5.5. Key derivation

As part of the GSS_API exchange, it is conceivable that a session key may be derived.

[Should we just use this session key or derive another session key from it??]

5.6. ECP negotiation

Since SPNEGO supports ciphersuite negotiation, peers completing the GSS_API SPNEGO negotiation will also have selected a ciphersuite, which includes key strength, encryption and hashing methods. As a result, a subsequent Encryption Control Protocol (ECP) conversation, if it occurs, has a predetermined result.

In order to ensure agreement between the EAP-GSS SPNEGO and the subsequent ECP negotiation (described in [6]), during ECP negotiation the PPP peer MUST offer only the ciphersuite negotiated in EAP-GSS. This ensures that the PPP authenticator MUST accept the EAP-GSS negotiated ciphersuite in order for the onversation to proceed. Should the authenticator not accept the EAP-GSS negotiated ciphersuite, then the peer MUST send an LCP terminate and disconnect.

Please note that it cannot be assumed that the PPP authenticator and EAP server are located on the same machine or that the authenticator understands the EAP-GSS conversation that has passed through it. Thus if the peer offers a ciphersuite other than the one negotiated in EAP-GSS there is no way for the authenticator to know how to respond correctly.
5.7. Examples

In the case where the EAP-GSS authentication is successful, the conversation will appear as follows:

Authentication Peer          Authenticator
-------------------------------          --------------
PPP LCP Request-EAP auth
PPP LCP ACK-EAP auth ->
PPP EAP-Request/Identity      <- PPP EAP-Response/Identity (MyID) ->
PPP EAP-Response/EAP-Type=EAP-GSS      <- PPP EAP-Request/EAP-Type=EAP-GSS (GSS Start, S bit set)
PPP EAP-Response/EAP-Type=EAP-GSS      <- PPP EAP-Request/EAP-Type=EAP-GSS
PPP EAP-Response/EAP-Type=EAP-GSS      <- PPP EAP-Request/EAP-Type=EAP-GSS
PPP Authentication Phase complete, NCP Phase starts
ECP negotiation
CCP negotiation

In the case where the EAP-GSS authentication is successful, and fragmentation is required, the conversation will appear as follows:

Authentication Peer          Authenticator
-------------------------------          --------------
PPP LCP Request-EAP auth
PPP LCP ACK-EAP auth ->
PPP EAP-Request/Identity      <- PPP EAP-Response/Identity
PPP EAP-Response/Identity (MyID) ->

<- PPP EAP-Request/
EAP-Type=EAP-GSS
(GSS Start, S bit set)

PPP EAP-Response/
EAP-Type=EAP-GSS ->

<- PPP EAP-Request/
EAP-Type=EAP-GSS
(Fragment 1: L, M bits set)

PPP EAP-Response/
EAP-Type=EAP-GSS ->

<- PPP EAP-Request/
EAP-Type=EAP-GSS
(Fragment 2: M bit set)

PPP EAP-Response/
EAP-Type=EAP-GSS ->

<- PPP EAP-Request/
EAP-Type=EAP-GSS
(Fragment 3)

PPP EAP-Response/
EAP-Type=EAP-GSS
(Fragment 1: L, M bits set) ->

<- PPP EAP-Request/
EAP-Type=EAP-GSS

PPP EAP-Response/
EAP-Type=EAP-GSS
(Fragment 2) ->

<- PPP EAP-Request/
EAP-Type=EAP-GSS

PPP EAP-Response/
EAP-Type=EAP-GSS ->

<- PPP EAP-Success

PPP Authentication Phase complete,
NCP Phase starts

ECP negotiation

CCP negotiation

6. Detailed description of the EAP-GSS protocol

6.1. PPP EAP GSS Packet Format

A summary of the PPP EAP GSS Request/Response packet format is shown below. The fields are transmitted from left to right.
Code

1 - Request
2 - 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 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

14 - EAP GSS

Data

The format of the Data field is determined by the Code field.

6.2. PPP EAP GSS Request Packet

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

1

Identifier

The Identifier field is one octet and aids in matching responses with requests. The Identifier field MUST be changed on each Request packet.

Length

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and GSS Response fields.

Type

14 - EAP GSS

Flags

0 1 2 3 4 5 6 7 8
+-+-+-+-+-+-+-+-+
|L M S R R R R R|
+-+-+-+-+-+-+-+-+

L = Length included
M = More fragments
S = EAP-GSS start
R = Reserved

The L bit (length included) is set to indicate the presence of the four octet GSS Message Length field, and MUST be set for the first fragment of a fragmented GSS message or set of messages. The M bit (more fragments) is set on all but the last fragment. The S bit (EAP-GSS start) is set in an EAP-GSS Start message. This differentiates the EAP-GSS Start message from a fragment acknowledgement.

GSS Message Length

The GSS Message Length field is four octets, and is present only if the L bit is set. This field provides the total length of the GSS message or set of messages that is being fragmented.

GSS data

The GSS data consists of the encapsulated GSS packet.
6.3. PPP EAP GSS Response Packet

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

```
|     Code      |   Identifier  |            Length             |
+-----------------+-----------------+-----------------------------|
|     Type      |     Flags     |      GSS Message Length     |
+-----------------+-----------------+-----------------------------|
|     GSS Message Length        |       GSS Data... |
+-----------------+-----------------+-----------------------------|
```

**Code**

2

**Identifier**

The Identifier field is one octet and MUST match the Identifier field from the corresponding request.

**Length**

The Length field is two octets and indicates the length of the EAP packet including the Code, Identifier, Length, Type, and GSS data fields.

**Type**

14 - EAP GSS

**Flags**

```
|L M S R R R R R|
+----------------|
```

L = Length included
M = More fragments
S = EAP-GSS start
R = Reserved

The L bit (length included) is set to indicate the presence of the four octet GSS Message Length field, and MUST be set for the first
fragment of a fragmented GSS message or set of messages. The M bit (more fragments) is set on all but the last fragment. The S bit (EAP-GSS start) is set in an EAP-GSS Start message. This differentiates the EAP-GSS Start message from a fragment acknowledgement.

GSS Message Length

The GSS Message Length field is four octets, and is present only if the L bit is set. This field provides the total length of the GSS message or set of messages that is being fragmented.

GSS data

The GSS data consists of the encapsulated GSS packet.

7. References


INTERNET-DRAFT     PPP EAP GSS Authentication Protocol   1 December 1999


8. Security Considerations

8.1. Certificate revocation

Since the EAP server is on the Internet during the EAP conversation, the server is capable of following a certificate chain or verifying whether the peer’s certificate has been revoked. In contrast, the peer may or may not have Internet connectivity, and thus while it can validate the EAP server’s certificate based on a pre-configured set of CAs, it may not be able to follow a certificate chain or verify whether the EAP server’s certificate has been revoked.

In the case where the peer is initiating a voluntary Layer 2 tunnel using PPTP or L2TP, the peer will typically already have a PPP interface and Internet connectivity established at the time of tunnel initiation. As a result, during the EAP conversation it is capable of checking for certificate revocation.

However, in the case where the peer is initiating an initial PPP conversation, it will not have Internet connectivity and is therefore not capable of checking for certificate revocation until after NCP negotiation completes and the peer has access to the Internet. In this case, the peer SHOULD check for certificate revocation after connecting to the Internet.

8.2. Separation of the EAP server and PPP authenticator

As a result of the EAP-GSS conversation, the EAP endpoints will mutually authenticate, negotiate a ciphersuite, and derive a session key for subsequent use in PPP encryption. Since the peer and EAP client reside on the same machine, it is necessary for the EAP client module to pass the session key to the PPP encryption module.

The situation may be more complex on the PPP authenticator, which may or may not reside on the same machine as the EAP server. In the case where the EAP server and PPP authenticator reside on different machines, there are several implications for security. Firstly, the mutual authentication defined in EAP-GSS will occur between the peer and the EAP server, not between the peer and the authenticator. This means that as a result of the EAP-GSS conversation, it is not possible for the peer to validate the identity of the NAS or tunnel server that it is speaking to.

The second issue 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 or tunnel server that needs to use the key. The specification of this transit mechanism is outside the scope of...
8.3. Relationship of PPP encryption to other security mechanisms

It is envisaged that EAP-GSS will be used primarily with dialup PPP connections. However, there are also circumstances in which PPP encryption may be used along with Layer 2 tunneling protocols such as PPTP and L2TP.

In compulsory layer 2 tunneling, a PPP peer makes a connection to a NAS or router which tunnels the PPP packets to a tunnel server. Since with compulsory tunneling a PPP peer cannot tell whether its packets are being tunneled, let alone whether the network device is securing the tunnel, if security is required then the client must make its own arrangements. In the case where all endpoints cannot be relied upon to implement IPSEC, TLS, or another suitable security protocol, PPP encryption provides a convenient means to ensure the privacy of packets transiting between the client and the tunnel server.

9. Acknowledgments

Thanks to Terence Spies, Paul Leach, and Mike Swift of Microsoft for useful discussions of this problem space.

10. Authors’ Addresses

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: +1 (425) 936-6605
EMail: bernarda@microsoft.com

11. Full Copyright Statement

Copyright (C) The Internet Society (1999). All Rights Reserved. This document and translations of it may be copied and furnished to others, and derivative works that comment on or otherwise explain it or assist in its implementation may be prepared, copied, published and distributed, in whole or in part, without restriction of any kind, provided that the above copyright notice and this paragraph are included on all such copies and derivative works. However, this document itself may not be modified in any way, such as by removing the copyright notice or references to the Internet Society or other Internet organizations, except as needed for the purpose of developing Internet standards in which case the procedures for copyrights defined in the Internet
Standards process must be followed, or as required to translate it into languages other than English. The limited permissions granted above are perpetual and will not be revoked by the Internet Society or its successors or assigns. This document and the information contained herein is provided on an "AS IS" basis and THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE."

12. Expiration Date

This memo is filed as <draft-aboba-pppext-eapgss-00.txt>, and expires August 1, 2000.