Network Working Group
Request for Comments: 4739
Category: Experimental

P. Eronen
Nokia
J. Korhonen
TeliaSonera

November 2006

Multiple Authentication Exchanges in the Internet Key Exchange (IKEv2) Protocol

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Copyright Notice

Copyright (C) The IETF Trust (2006).

Abstract

The Internet Key Exchange (IKEv2) protocol supports several mechanisms for authenticating the parties, including signatures with public-key certificates, shared secrets, and Extensible Authentication Protocol (EAP) methods. Currently, each endpoint uses only one of these mechanisms to authenticate itself. This document specifies an extension to IKEv2 that allows the use of multiple authentication exchanges, using either different mechanisms or the same mechanism. This extension allows, for instance, performing certificate-based authentication of the client host followed by an EAP authentication of the user. When backend authentication servers are used, they can belong to different administrative domains, such as the network access provider and the service provider.

Table of Contents

1. Introduction

IKEv2 [IKEv2] supports several mechanisms for parties involved in the IKE_SA (IKE security association). These include signatures with public-key certificates, shared secrets, and Extensible Authentication Protocol (EAP) methods.

Currently, each endpoint uses only one of these mechanisms to authenticate itself. However, there are scenarios where making the authorization decision in IKEv2 (whether to allow access or not) requires using several of these methods.

For instance, it may be necessary to authenticate both the host (machine) requesting access, and the user currently using the host. These two authentications would use two separate sets of credentials (such as certificates and associated private keys) and might even use different authentication mechanisms.

To take another example, when an operator is hosting a Virtual Private Network (VPN) gateway service for a third party, it may be necessary to authenticate the client to both the operator (for billing purposes) and the third party’s Authentication, Authorization, and Accounting (AAA) server (for authorizing access to the third party’s internal network).

This document specifies an extension to IKEv2 that allows the use of multiple authentication exchanges, using either different mechanisms or the same mechanism. This extension allows, for instance, performing certificate-based authentication of the client host followed by an EAP authentication of the user.

Each authentication exchange requiring communication with backend AAA servers may be directed to different backend AAA servers, located even in different administrative domains. However, details of the communication between the IKEv2 gateway and the backend authentication servers are beyond the scope of this document. In particular, this document does not specify any changes to existing AAA protocols, and it does not require the use of any particular AAA protocol.

In case of several EAP authentications, it is important to notice that they are not a "sequence" (as described in Section 2.1 of [EAP]), but separate independent EAP conversations, which are usually also terminated in different EAP servers. Multiple authentication methods within a single EAP conversation are still prohibited as described in Section 2.1 of [EAP]. Using multiple independent EAP conversations is similar to the separate Network Access Provider (NAP) and Internet Service Provider (ISP) authentication exchanges planned for [PANA]. The discovery of the appropriate EAP server for each EAP authentication conversation is based on AAA routing.

1.1. Usage Scenarios

Figure 1 shows an example architecture of an operator-hosted VPN scenario that could benefit from a two-phase authentication within the IKEv2 exchange. First, the client authenticates towards the Network Access Provider (NAP) and gets access to the NAP-hosted VPN gateway. The first-phase authentication involves the backend AAA server of the NAP. After the first authentication, the client initiates the second authentication round that also involves the Third Party's backend AAA server. If both authentications succeed, the required IPsec tunnels are set up and the client can access protected networks behind the Third Party.

```
       Client                         *Network Access Provider*
     +---------+                    +---------+              +-----+
     |         |                    |  NAP's  |              | NAP |
     |Protected|     IPsec SAs      | Tunnel  | AAA Protocol | AAA |
     |Endpoint |<------------------>|Endpoint |<------------>|Serv/|
     |         |                    |         |              |Proxy|
     +---------+                    +---------+              +-----+
                                       ^                        ^
                            IPsec or  /                  AAA    |
                        Leased Line  /                 Protocol |
                                    /                           |
                                   v                            |
                           +---------+    *Third Party*         v
                           |3rd Party|                       +-----+
            Protected      | Tunnel  |                       | 3rd |
               Subnet <----|Endpoint |                       |Party|
                           |         |                       | AAA |
                           +---------+                       +-----+
```

Figure 1: Two-phase authentication used to gain access to the Third Party network via Network Access Provider. AAA traffic goes through NAP's AAA server.

The NAP's AAA server can be used to proxy the AAA traffic to the Third Party's backend AAA server. Alternatively, the AAA traffic from the NAP's tunnel endpoint could go directly to the Third Party's backend AAA servers. However, this is more or less an AAA routing issue.

1.2. Terminology

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 [KEYWORDS].

The terms and abbreviations "authenticator", "backend authentication server", "EAP server", and "peer" in this document are to be interpreted as described in [EAP].

When messages containing IKEv2 payloads are described, optional payloads are shown in brackets (for instance, "[FOO]"), and a plus sign indicates that a payload can be repeated one or more times (for instance, "FOO+").

2. Solution

2.1. Solution Overview

The peers announce support for this IKEv2 extension by including a MULTIPLE_AUTH_SUPPORTED notification in the IKE_SA_INIT response (responder) and the first IKE_AUTH request (initiator).

If both peers support this extension, either of them can announce that it wishes to have a second authentication by including an ANOTHER_AUTH_FOLLOWS notification in any IKE_AUTH message that contains an AUTH payload. This indicates that the peer sending the ANOTHER_AUTH_FOLLOWS wishes to authenticate another set of credentials to the other peer. The next IKE_AUTH message sent by this peer will contain a second identity payload (IDi or IDr) and starts another authentication exchange. The IKE_AUTH phase is considered successful only if all the individual authentication exchanges complete successfully.

It is assumed that both peers know what credentials they want to present; there is no negotiation about, for instance, what type of authentication is to be done. As in IKEv2, EAP-based authentication is always requested by the initiator (by omitting the AUTH payload).

The AUTH payloads are calculated as specified in [IKEv2] Sections 2.15 and 2.16, where IDi’ refers to the latest IDi payload sent by the initiator, and IDr’ refers to the latest IDr payload sent by the responder. If EAP methods that do not generate shared keys are used, it is possible that several AUTH payloads with identical contents are sent. When such EAP methods are used, the purpose of the AUTH payload is simply to delimit the authentication exchanges, and ensure that the IKE_SA_INIT request/response messages were not modified.

2.2. Example 1: Multiple EAP Authentications

This example shows certificate-based authentication of the responder followed by an EAP authentication exchange (messages 1-10). When the first EAP exchange is ending (the initiator is sending its AUTH payload), the initiator announces that it wishes to have a second authentication exchange by including an ANOTHER_AUTH_FOLLOWS notification (message 9).

After this, a second authentication exchange begins. The initiator sends a new IDi payload but no AUTH payload (message 11), indicating that EAP will be used. After that, another EAP authentication exchange follows (messages 12-18).

```
      Initiator                   Responder
     -----------                 -----------
      1. HDR, SA, KE, Ni -->
                             <--  2. HDR, SA, KE, Nr, [CERTREQ],
                                          N(MULTIPLE_AUTH_SUPPORTED)
      3. HDR, SK { IDi, [CERTREQ+], [IDr],
                   SA, TSi, TSr, N(MULTIPLE_AUTH_SUPPORTED) }  -->
                             <--  4. HDR, SK { IDr, [CERT+], AUTH,
                                               EAP(Request) }
      5. HDR, SK { EAP(Response) }  -->
                             <--  6. HDR, SK { EAP(Request) }
      7. HDR, SK { EAP(Response) }  -->
                             <--  8. HDR, SK { EAP(Success) }
      9. HDR, SK { AUTH,
                   N(ANOTHER_AUTH_FOLLOWS) }  -->
                             <--  10. HDR, SK { AUTH }
      11. HDR, SK { IDi }  -->
                             <--  12. HDR, SK { EAP(Request) }
      13. HDR, SK { EAP(Response) }  -->
                             <--  14. HDR, SK { EAP(Request) }
      15. HDR, SK { EAP(Response) }  -->
                             <--  16. HDR, SK { EAP(Success) }
      17. HDR, SK { AUTH }  -->
                             <--  18. HDR, SK { AUTH, SA, TSi, TSr }
```

Example 1: Certificate-based authentication of the responder, followed by two EAP authentication exchanges.

2.3. Example 2: Mixed EAP and Certificate Authentications

Another example is shown below: here both the initiator and the responder are first authenticated using certificates (or shared secrets); this is followed by an EAP authentication exchange.

```
      Initiator                   Responder
     -----------                 -----------
      1. HDR, SA, KE, Ni -->
                             <--  2. HDR, SA, KE, Nr, [CERTREQ],
                                          N(MULTIPLE_AUTH_SUPPORTED)
      3. HDR, SK { IDi, [CERT+], [CERTREQ+], [IDr], AUTH,
                   SA, TSi, TSr, N(MULTIPLE_AUTH_SUPPORTED),
                   N(ANOTHER_AUTH_FOLLOWS) }  -->
                             <--  4. HDR, SK { IDr, [CERT+], AUTH }
      5. HDR, SK { IDi }  -->
                             <--  6. HDR, SK { EAP(Request) }
      7. HDR, SK { EAP(Response) }  -->
                             <--  8. HDR, SK { EAP(Request) }
      9. HDR, SK { EAP(Response) }  -->
                             <--  10. HDR, SK { EAP(Success) }
      11. HDR, SK { AUTH }  -->
                             <--  12. HDR, SK { AUTH, SA, TSi, TSr }
```

Example 2: Certificate-based (or shared-secret-based)
authentication of the initiator and the responder,
followed by an EAP authentication exchange.

2.4. Example 3: Multiple Initiator Certificates

This example shows yet another possibility: the initiator has two different certificates (and associated private keys), and authenticates both of them to the responder.

```
      Initiator                   Responder
     -----------                 -----------
      1. HDR, SA, KE, Ni -->
                             <--  2. HDR, SA, KE, Nr, [CERTREQ],
                                          N(MULTIPLE_AUTH_SUPPORTED)
      3. HDR, SK { IDi, [CERT+], [CERTREQ+], [IDr], AUTH,
                   SA, TSi, TSr, N(MULTIPLE_AUTH_SUPPORTED),
                   N(ANOTHER_AUTH_FOLLOWS) }  -->
                             <--  4. HDR, SK { IDr, [CERT+], AUTH }
      5. HDR, SK { IDi, [CERT+], AUTH }  -->
                             <--  6. HDR, SK { SA, TSi, TSr }
```

Example 3: Two certificate-based authentications of the initiator, and one certificate-based authentication of the responder.

2.5. Example 4: Multiple Responder Certificates

This example shows yet another possibility: the responder has two different certificates (and associated private keys), and authenticates both of them to the initiator.

```
      Initiator                   Responder
     -----------                 -----------
      1. HDR, SA, KE, Ni -->
                             <--  2. HDR, SA, KE, Nr, [CERTREQ],
                                          N(MULTIPLE_AUTH_SUPPORTED)
      3. HDR, SK { IDi, [CERT+], [CERTREQ+], [IDr], AUTH,
                   SA, TSi, TSr, N(MULTIPLE_AUTH_SUPPORTED) }  -->
                             <--  4. HDR, SK { IDr, [CERT+], AUTH,
                                               N(ANOTHER_AUTH_FOLLOWS) }
      5. HDR, SK { }  -->
                             <--  6. HDR, SK { IDr, [CERT+], AUTH,
                                               SA, TSi, TSr }
```

Example 4: Two certificate-based authentications of the responder, and one certificate-based authentication of the initiator.

3. Payload Formats

3.1. MULTIPLE_AUTH_SUPPORTED Notify Payload

The MULTIPLE_AUTH_SUPPORTED notification is included in the IKE_SA_INIT response or the first IKE_AUTH request to indicate that the peer supports this specification. The Notify Message Type is MULTIPLE_AUTH_SUPPORTED (16404). The Protocol ID and SPI Size fields MUST be set to zero, and there is no data associated with this Notify type.

3.2. ANOTHER_AUTH_FOLLOWS Notify Payload

The ANOTHER_AUTH_FOLLOWS notification payload is included in an IKE_AUTH message containing an AUTH payload to indicate that the peer wants to continue with another authentication exchange. The Notify Message Type is ANOTHER_AUTH_FOLLOWS (16405). The Protocol ID and SPI Size fields MUST be set to zero, and there is no data associated with this Notify type.

4. IANA Considerations

This document defines two new IKEv2 notifications, MULTIPLE_AUTH_SUPPORTED and ANOTHER_AUTH_FOLLOWS, whose values are allocated from the "IKEv2 Notify Message Types" namespace defined in [IKEv2].

This document does not define any new namespaces to be managed by IANA.

5. Security Considerations

Security considerations for IKEv2 are discussed in [IKEv2]. The reader is encouraged to pay special attention to considerations relating to the use of EAP methods that do not generate shared keys. However, the use of multiple authentication exchanges results in at least one new security consideration.

In normal IKEv2, the responder authenticates the initiator before revealing its identity (except when EAP is used). When multiple authentication exchanges are used to authenticate the initiator, the responder has to reveal its identity before all of the initiator authentication exchanges have been completed.

6. Acknowledgments

The authors would like to thank Bernard Aboba, Jari Arkko, Spencer Dawkins, Lakshminath Dondeti, Henry Haverinen, Russ Housley, Mika Joutsenvirta, Charlie Kaufman, Tero Kivinen, Yoav Nir, Magnus Nystrom, Mohan Parthasarathy, and Juha Savolainen for their valuable comments.

7. References

7.1. Normative References

[IKEv2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

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

7.2. Informative References

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

[PANA] Yegin, A., Ohba, Y., Penno, R., Tsirtsis, G., and C. Wang, "Protocol for Carrying Authentication for Network Access (PANA) Requirements", RFC 4058, May 2005.

Authors’ Addresses

Pasi Eronen
Nokia Research Center
P.O. Box 407
FIN-00045 Nokia Group
Finland

EMail: pasi.eronen@nokia.com

Jouni Korhonen
TeliaSonera
P.O. Box 970
FIN-00051 Sonera
Finland

EMail: jouni.korhonen@teliasonera.com

Full Copyright Statement

Copyright (C) The IETF Trust (2006).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST, AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM 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.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Acknowledgement

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