IP Security Protocol Working Group (IPSEC)
INTERNET-DRAFT
draft-ietf-ipsec-ike-ext-meth-04.txt
Expires: 22 May 2001

T. Kivinen
SSH Communications Security

22 November 2000

ISAKMP & IKE Extension Methods

Status of This memo

Abstract

This document describes multiple extension methods of the ISAKMP [RFC 2408] and IKE [RFC 2409] protocols and how the older versions should respond when they receive such extensions. This document mainly tries to describe the best practice of extensions handling in ISAKMP [RFC 2408] and IKE [RFC 2409], so that a future version can be made without breaking the old existing versions.

Table of Contents

1. Introduction 2
2. Specification of Requirements 2
3. Terminology 2
4. Sending Error Notifications 3
5. Order of Checking 3
6. ISAKMP Major and Minor Version Numbers 3
6.1. ISAKMP Major Version Number 4
6.2. ISAKMP Minor Version number 4
7. Phase 1 Transform Identifier 6
8. Exchange type 6
9. Flags Inside the Generic Packet Header 6
10. Payload Types 7
11. Vendor ID Payload 8
12. Data Attribute Types and Values 9
12.1. Data Attributes, Protocol and Transform IDs 10
13. Reserved Fields 10
14. Identity Type 10
15. Certificate Encoding 11
16. Notify Message Type 11
17. Domain of Interpretation 12
18. Security Considerations 12
19. References 13
20. Authors' Addresses 13

1. Introduction

The ISAKMP [RFC 2408] and IKE [RFC 2409] protocols can be extended in various ways. It is not clearly defined in the current document set how to use the extension mechanisms. Also, the current document set does not clearly define what a conforming implementation should do if it receives an extension that it does not understand.

This document describes how to provide backwards compatibility with the old versions. The reader is assumed to be familiar with most of the terms and concepts described in the ISAKMP [RFC 2408] and IKE [RFC 2409] documents.

2. Specification of Requirements

This document shall use the keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED, "MAY", and "OPTIONAL" to describe requirements. They are to be interpreted as described in the [RFC 2119] document.

3. Terminology

This document uses the terms "new version" and "old version" to identify the two different protocol versions. The new version is the version that supports the new extension, and the old version is a version that does not support it. The terms should always be interpreted only in the current context.

4. Sending Error Notifications

If the other end does not send error notifications, it is very hard to create usable extensions to the protocol. In that case the only way to detect whether the other end supports the extension is to see if the negotiation times out. This may cause unnecessary delays in the negotiation process. Because of that, all implementations SHOULD send notifications back when they reject any extensions or new features.

Implementations MAY limit the number of notifications sent out. A suitable limit could be something like one notification per second per host. Implementations SHOULD resend the notification if the other end resends its own ISAKMP datagram (in case the error notification was lost). This resending SHOULD also be limited to a reasonable level.

5. Order of Checking

The order of checks performed for an ISAKMP datagram SHOULD be the following:

- Major version number
- Minor version number
- Exchange type
- Flags in the generic header
- Payload types
- Reserved field in the generic payload headers
- Payload specific checks.

6. ISAKMP Major and Minor Version Numbers

All ISAKMP datagrams contain version numbers which describe the major and minor version numbers of the sending party. In the IKE defined in RFC 2409, major and minor version numbers are not authenticated. Thus, when they are later changed to be authenticated, there might be the possibility of a version rollback attack where the attacker forces negotiating parties to fall back to the RFC 2409 version of IKE.

The major version number is changed when major changes are done to the protocol, i.e. there are changes in the generic packet encoding routines. This means that the older versions cannot understand the newer packet format at all.

The minor version number is changed when new payloads are defined or
other minor changes in the protocol take place. The older versions can

still process the generic packet structure, but there might be small variations in the fields inside the payloads.

Each party MUST NOT change the version number it is sending during one negotiation, i.e. if a host started the negotiation using version number 1.1, it MUST use that during the whole negotiation. Separate negotiations MAY have different version numbers, i.e. a newer version may restart negotiation, and start using an older version number. Both ends use their own version number, so it is completely legal for one end to start with 1.1 and for the other end to respond with 1.0.

Phase 1 and phase 2 negotiations are separate negotiations. So, a phase 1 **negotiation that creates ISAKMP SA may use version X, and a phase 2** negotiation done over that ISAKMP SA may use version Y.

Because the minor version number is encoded into a 4-bit field (0-15), a minor version number roll-over might occur. This means that the major version number must be incremented even if the packet structure is still actually the same. Because of this phenomenon, incrementing the minor version number SHOULD be avoided if it is not absolutely necessary.

An implementation supporting minor version X MUST support all features up to that level (it MUST at least be able to ignore the non-critical extensions, and detect critical extensions and abort the negotiation in that case).

In addition to major and minor version numbers, there is a phase 1 transform identifier inside the SA payload, which identifies the key exchange method (e.g IKE) version number.

6.1. ISAKMP Major Version Number

If an old version receives a datagram with a major version number larger than its own, it SHOULD send the INVALID-MAJOR-VERSION notification back. It MUST put its own version number inside the notification datagram. This gives the other end the opportunity to obtain the version number supported by the sender of the notification. The received ISAKMP datagram MUST then be discarded (the old version cannot parse anything else in the datagram because the generic packet structure has changed).

Note that in most cases, this notification sent by the remote host is not authenticated, so it can also lead to the version rollback attack.

A new version receiving the INVALID-MAJOR-VERSION notification MAY fall back to the older version. If falling back to the older version has security implications, then the new version SHOULD NOT fall back to a previous version but instead fail the negotiation with a clear error message.

6.2. ISAKMP Minor Version number

If an old version receives a datagram with a minor version newer than its own, it

- o SHOULD continue processing the datagram, or it

- o MAY discard the ISAKMP datagram. In that case it SHOULD send the INVALID-MINOR-VERSION notification back.

In any case the old version MUST use its own local minor version number when sending packets back, so that a new version can get the older version number and fall back to the same version if necessary.

The new version MAY always start with the latest version number and fall back to the previous version separately each time, or it MAY cache this information for some time, or the version number used may be configured manually.

The minor number MUST be updated if

- o new flags are added (see section ``Flags Inside the Generic Packet Header'')

- o new use of a RESERVED field is defined (see section ``Reserved Fields'')

The minor number MAY be updated if

- o new general purpose payload types are added (see section ``Payload Types'')

The minor number SHOULD NOT be updated if

- o new exchange types are added (see section ``Exchange type'')

- o new payload types that can only be used in a specific exchange, etc. are added (see section ``Payload Types'')

- o new IANA registered data attribute types are defined (see section ``Data Attribute Types and Values'')

- o new IANA registered data attribute values are defined (see section ``Data Attribute Types and Values'')

- o a new certificate encoding type is added (see section ``Certificate Encoding'')

- o a new identity type is defined (see section ``Identity Type'')

- o new phase 1 transform identifiers are defined (see section ``Phase 1 Transform Identifier'')

The minor number MUST NOT be updated if

o a new domain of interpretation is defined (see section ``Domain of
   Interpretation'')

7. Phase 1 Transform Identifier

Phase 1 SA payload contains a transform identifier that specifies the key exchange method used. It can be used to negotiate the key exchange method and version. This is different from the minor version number in the manner that the other end MUST send back exactly the same transform ID (i.e. it MUST select one of those values offered). It cannot modify the SA payload which contains the transform identifier.

The selected exchange type may limit the possibility to negotiate different key exchange methods, because the key exchange type might affect the format of the first packet (for example the IKE aggressive mode).

8. Exchange type

An exchange type defines a generic packet exchange between two negotiating parties. It defines the number of datagrams in the exchange, the generic meaning of the packets (i.e. in main mode the first two datagrams exchange SA payloads, the next two datagrams are used for the key exchange, and the final two datagrams are used to exchange identities and to authenticate the exchange).

If an implementor wants to add new datagrams to the existing exchanges, then the implementor MUST create a new exchange and allocate a new exchange type for it.

If an implementation receives a datagram that contains an exchange type it does not understand, it SHOULD send back the INVALID-EXCHANGE-TYPE notification. Also it MUST ignore the ISAKMP datagram.

If an implementation receives the INVALID-EXCHANGE-TYPE notification, it MAY fall back to a more standard exchange (for example, from the aggressive mode to the main mode).

When new ISAKMP exchange types are added, the minor version number SHOULD NOT be updated. When new key exchange specific exchange types are added, the phase 1 transform identifier SHOULD NOT be updated. If old existing key exchange specific exchange types are modified, then the phase 1 transform identifier SHOULD be updated.

A new version MAY always start with the new exchange type and fall back to a previous, more standard exchange type separately each time, or it MAY cache this information for some time, or the exchange type used may be configured manually.

9. Flags Inside the Generic Packet Header

Flags are a part of the generic ISAKMP packet structure. Currently, three flags are defined (encryption, commit bit, authentication only bit).

When new flags are defined, the minor version number MUST be updated.

When a new flag is added, the specification MUST indicate if this flag has any security implications and whether a new version should fail the negotiation if the other end is using an old version.

If the old version receives a datagram with a newer minor version number, and some unknown flags are set, it

o SHOULD continue the exchange and ignore the new flags, or it

o MAY fail the negotiation. In that case it SHOULD send the INVALID-FLAGS notification back.

If a new version notices (from the version numbers) that the other end is using an old version, it MUST fail the negotiation if it tried to set a flag that has security implications. If the flag it set does not have security implications, it MAY continue the exchange.

10. Payload Types

Each payload inside a datagram contains a type field in the generic payload header. The payload type describes the internal structure of the payload. Unknown payloads can be ignored because the generic payload header contains the length of the payload data.

Payload types in the special private range are to be used for mutually consenting implementations only. Implementations MUST NOT send payloads of a private type unless the both parties have both sent and received a familiar vendor ID payload. After this exchange of the vendor ID payloads during the phase 1, implementations MAY immediately start sending private payloads.

Note that RFC 2409 does not protect vendor ID payloads from tampering, so implementations should not enable anything based on vendor IDs if that feature has security implications.

When new payload types are defined (other than private-use payloads), and a new version can detect from somewhere else than from version numbers that the other end understands or does not understand the new payload types, then the minor version number SHOULD NOT be updated. If there is no way to detect if the other end understands the newly defined payload types, then the either the minor version number of the ISAKMP packet or the phase 1 transform identifier SHOULD be updated. If the new payloads are inside the key exchange method specific exchange, then it is enough to only update the phase 1 transform identifier.

For example, if the newly defined payload type can only be used in a certain new exchange type (like attribute payload inside the transactional exchange), then an old version will fail the negotiation because of the new exchange type, and a new version can detect that. There is no need to update the version number in that case.

This allows for creating optional features in the ISAKMP protocol in
such a manner that the implementors do not need to implement them all.

Every time the minor version number or the phase 1 transform identifier is updated, all the implementations MUST understand all the new mandatory payloads. In the case of a new generic payload that can be used in several exchanges etc., the minor version number or the phase 1 transform identifier MAY be updated.

When a new payload type is added, the corresponding specification MUST indicate if the new payload has any security implications and whether a new version should fail the negotiation if the other end is using an old version. The specification MUST also indicate whether it is mandatory to implement the new feature or not.

If the implementation receives an unknown private payload type, it

o SHOULD ignore the payload and continue, or it

o MAY fail the negotiation. In that case it SHOULD send the INVALID-PAYLOAD-TYPE notification back.

If the implementation receives an unknown payload type from the RESERVED range and the version numbers (both ISAKMP major/minor version numbers and phase 1 transform ID) are the same, it MUST fail the negotiation, and it SHOULD send INVALID-PAYLOAD-TYPE notification back.

If the implementation receives an unknown payload type from the RESERVED range, and the minor version number (or the phase 1 transform ID) of the other end is newer, it
o SHOULD ignore the payload and continue, or it

o MAY fail the negotiation. In that case it SHOULD send the INVALID-PAYLOAD-TYPE notification back.

If the new version has sent out a payload of a type that is defined in a newer version of the protocol than the other end understands (this can be detected by checking the minor version number), and the payload has security implications then it MUST fail the negotiation.

There may be a need to add a criticality flag in the generic payload header in the next version of ISAKMP [RFC 2408]. This would allow an old version to detect immediately whether it can safely ignore the payload or whether it MUST fail the negotiation (in that case it SHOULD send an error notification). This criticality flag could be added to the reserved field of the generic payload header (there are 8 reserved bits inside the generic payload header). See section ``Reserved Fields'' for more information about how an old version should handle the criticality flag.

11. Vendor ID Payload

The vendor ID payload is a payload that can be included anywhere in the

phase 1 negotiation. It gives the other end a possibility to recognize
the remote implementation. These payloads are not authenticated in the
RFC 2409 version of the IKE.

The vendor ID has two uses. The first one is that by sending a vendor ID payload along the SA payload, the initiator specifies whose private-use values it is using (it SHOULD only send only one vendor ID payload, or at least all the vendor ID payloads MUST NOT have overlapping private numbers defining different things).

When initiator wants to use some private-use values in the exchange, it just adds its own vendor ID payload(s). When the responder receives the vendor ID payload(s) along with for example the SA payload, it can find out whose private-use values are inside the SA payload by checking the vendor ID payload.

The second use is to allow for vendor specific extensions, after both ends have sent and received familiar vendor IDs.

Implementations MUST NOT fail a negotiation because of the presence of the vendor ID payload(s), i.e. they MUST be able to ignore it.

If familiar vendor ID payloads have been exchanged (both sent and received) then implementations MAY do anything, including using private extensions, private payloads, new identity types, running nethack over the ISAKMP SA, etc.

12. Data Attribute Types and Values

SA payloads and some other payloads in the ISAKMP contain data attributes. Data attribute consists of an attribute type and a value. The data attribute type and value number spaces are divided into two parts: The IANA range and the private-use range.

The phase 1 data attribute types and values are defined in the IKE document and ISAKMP documents. This part should probably be separated from those documents to separate IKE DOI. The Phase 2 data attributes are defined in the DOI [RFC 2407] document.

The private-use data attribute TYPES can be used anywhere, and when they are used, the sender SHOULD send vendor ID payload(s) specifying whose private-use values the sender is using.

When adding new IANA registered data attribute TYPES, the minor version number of the protocol SHOULD NOT be updated. When adding new IANA registered data attribute TYPES, the phase 1 transform identifier MAY be updated.

The private-use data attribute VALUES can also be used anywhere, and when they are used, the sender SHOULD send vendor ID payload(s) specifying whose private-use values the sender is using.

When adding new IANA registered data attribute VALUES, the minor version number of the protocol SHOULD NOT be updated. When adding new IANA

registered data attribute VALUES, the phase 1 transform identifier MAY
be updated.

12.1. Data Attributes, Protocol and Transform IDs

The proposal or transform payload MUST NOT be selected by the responder if it contains unknown protocol IDs, transform IDs, data attribute types, or data attribute values.

This means that an initiator SHOULD always include a proposal without any private-use types or values so that if the other end does not understand them, then it may select the transform or proposal without private-use types or values.

13. Reserved Fields

Lots of payloads in the ISAKMP contain RESERVED fields that are defined to be zero and whose contents MUST be checked. This makes extension of the payloads very difficult to implement. Changing this so that their contents MUST be checked only if the version numbers are the same makes it much easier to introduce backwards compatible extensions to the protocol in the future.

When a new use of a RESERVED field is defined, the minor version number MUST be updated.

When a new use of a RESERVED field is defined, the corresponding specification MUST indicate if this new use of the RESERVED field has any security implications and whether a new version should fail the negotiation if the other end is using an old version and the new version tried to use this new usage for a RESERVED field.

If an old implementation receives a packet that contains a non-zero RESERVED field, and the minor version number of the other end is newer than the local one, then it

- SHOULD ignore the contents of the RESERVED field and continue, or it

- MAY ignore the ISAKMP datagram. In that case it SHOULD send the INVALID-RESERVED-FIELD notification back.

If the new version notices that the other end is using the old version, it MUST fail the negotiation if it tried to use the RESERVED field in such a way that has security implications. If the new defined use of the RESERVED field does not have security implications, it MAY continue the exchange.

14. Identity Type

The identity type is used to specify the interpretation of the identity payload contents. The identity type is specified in the DOI document, but the generic structure is defined in the ISAKMP document. This generic structure contains this identity type value.

When a new identity type is specified, the minor version number or the phase 1 transform identifier SHOULD NOT be incremented.

If an old version receives an unknown identity type, it MUST fail the negotiation, and it SHOULD send the INVALID-ID-INFORMATION notification back.

A new version MAY always start with the new identity type and fall back to a previous more standard identity type separately each time, or it MAY cache this information for some time, or it MAY manually configure the identity type to be used.

15. Certificate Encoding

Certificate encoding is used to specify the interpretation of the certificate payload contents.

When a new certificate encoding type is added, the minor version number or the phase 1 transform identifier SHOULD NOT be incremented.

If an old version receives an unknown certificate encoding type, it

o SHOULD just ignore the payload and continue, or it

o MAY fail the negotiation. In that case it SHOULD send the INVALID-CERT-ENCODING notification back.

16. Notify Message Type

Messages containing notify payload are sent to either notify an error situation or to give out status information. Each notify payloads contain a notify message type which describes the message type.

The notify message types are divided in the several separate ranges:

1 - 8191	ISAKMP error code range
8192 - 16383	Private use error code range
16384 - 24575	ISAKMP status code range
24576 - 32767	DOI status code range
32768 - 40959	Private use status code range

If an unknown error (1 <= code <= 16383) notification type is received, the receiver MUST treat it as a fatal error and abort the negotiation.

If an unknown status (16384 <= code <= 40959) notification type is
received, the receiver MUST ignore the notification payload.

For example, a new keep-alive protocol for the ISAKMP SA may be defined by just defining that both ends must send a new STILL-CONNECTED notification every 60 seconds. If the other end never sees those notifications, it just assumes that the other end does not support this feature, and ceases sending any further keep-alive packets. If that new STILL-CONNECTED status code is selected from the status code range, then old implementations will just ignore them.

When using notifications, implementations must take care of what to do with the notifications which are not authenticated (i.e. those received before the ISAKMP SA is ready). If there is no ISAKMP SA established with the remote host, then most of the notifications may still be trusted in order to avoid lengthy timeouts in error situations. If there is a ISAKMP SA established, then unauthenticated notifications SHOULD be ignored.

17. Domain of Interpretation

Each SA payload (and some others like notify and delete payloads) specifies the domain of interpretation for the exchange. There is no version numbers in the DOI, so if a new version of DOI is incompatible with the previous version, a new DOI number MUST be allocated. In the normal case, there is no need to have a version number in the DOI, and additions to it may be done without updating the DOI number.

If an unknown domain of interpretation is received, the responder MUST discard the ISAKMP datagram and it SHOULD send the DOI-NOT-SUPPORTED notification back. This usually also means that the negotiation is aborted.

When a new domain of interpretation is defined, the minor version number MUST NOT be incremented. If ISAKMP DOI is modified, there might be a need to update the DOI number.

18. Security Considerations

This document describes how to use the extension mechanisms defined in ISAKMP [RFC 2408] and IKE [RFC 2409]. Because some of those extensions might have security implications, it is required that when new extensions are defined, it is also explained what security implications they have and what the implementations supporting them should do if the other end does not support the extensions.

One security problem comes from the ISAKMP [RFC 2408] and IKE [RFC 2409] protocol, because the version number, exchange type, and flags fields are not authenticated in the RFC 2409 version of IKE protocol. The [REVISED-HASH] describes a way to fix this problem by updating the phase 1 **transform id number.**

If a real security problem is later found from that version of protocol,

the implementors MUST make sure that they never fall back to any
previous version because the attacker can force falling back to a
previous version by changing the version numbers inside the datagrams.

Also the vendor ID payloads, notifications etc. inside the phase 1 packets are not authenticated in the RFC 2409 version of IKE. This means that implementations SHOULD NOT enable any security critical extensions based on those unathenticated payloads.

Another security problem comes from the fact that there is no way to send authenticated notifications before the phase 1 (ISAKMP) SA is finished. This means that most of the error notifications about the Phase 1 exchange are sent without any kind of protection.

19. References

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

[RFC 2409] Harkins D., Carrel D., "The Internet Key Exchange (IKE)", November 1998

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

[RFC 2407] Piper D., "The Internet IP Security Domain of Interpretation for ISAKMP", November 1998

[REVISED-HASH] Kivinen T., "Fixing IKE Phase 1 & 2 Authentication HASHs", November 2000

20. Authors' Addresses

Tero Kivinen
SSH Communications Security Corp
Fredrikinkatu 42
FIN-00100 HELSINKI
Finland
E-mail: kivinen@ssh.fi