Network Working Group
Request for Comments: 4621
Category: Informational

T. Kivinen
Safenet, Inc.
H. Tschofenig
Siemens

August 2006

Design of the IKEv2 Mobility and Multihoming (MOBIKE) Protocol

Status of This Memo

This memo provides information for the Internet community. It does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Copyright Notice

Abstract

The IKEv2 Mobility and Multihoming (MOBIKE) protocol is an extension of the Internet Key Exchange Protocol version 2 (IKEv2). These extensions should enable an efficient management of IKE and IPsec Security Associations when a host possesses multiple IP addresses and/or where IP addresses of an IPsec host change over time (for example, due to mobility).

This document discusses the involved network entities and the relationship between IKEv2 signaling and information provided by other protocols. Design decisions for the MOBIKE protocol, background information, and discussions within the working group are recorded.

Table of Contents

1. Introduction

The purpose of IKEv2 is to mutually authenticate two hosts, to establish one or more IPsec Security Associations (SAs) between them, and subsequently to manage these SAs (for example, by rekeying or deleting). IKEv2 enables the hosts to share information that is relevant to both the usage of the cryptographic algorithms that should be employed (e.g., parameters required by cryptographic algorithms and session keys) and to the usage of local security policies, such as information about the traffic that should experience protection.

IKEv2 assumes that an IKE SA is created implicitly between the IP address pair that is used during the protocol execution when establishing the IKEv2 SA. This means that, in each host, only one IP address pair is stored for the IKEv2 SA as part of a single IKEv2 protocol session, and, for tunnel mode SAs, the host places this single pair in the outer IP headers. Existing IPsec documents make no provision to change this pair after an IKE SA is created (except for dynamic address update of Network Address Translation Traversal (NAT-T)).

There are scenarios where one or both of the IP addresses of this pair may change during an IPsec session. In principle, the IKE SA and all corresponding IPsec SAs could be re-established after the IP address has changed. However, this is a relatively expensive operation, and it can be problematic when such changes are frequent. Moreover, manual user interaction (for example, when using human-operated token cards (SecurID)) might be required as part of the IKEv2 authentication procedure. Therefore, an automatic mechanism is needed that updates the IP addresses associated with the IKE SA and the IPsec SAs. The MOBIKE protocol provides such a mechanism.

The MOBIKE protocol is assumed to work on top of IKEv2 [RFC4306]. As IKEv2 is built on the IPsec architecture [RFC4301], all protocols developed within the MOBIKE working group must be compatible with both IKEv2 and the architecture described in RFC 4301. This document does not discuss mobility and multi-homing support for IKEv1 [RFC2409] or the obsoleted IPsec architecture described in RFC 2401 [RFC2401].

This document is structured as follows: After some important terms are introduced in Section 2, a number of relevant usage scenarios are discussed in Section 3. Section 4 describes the scope of the MOBIKE protocol. Section 5 discusses design considerations affecting the MOBIKE protocol. Section 6 investigates details regarding the MOBIKE protocol. Finally, this document concludes in Section 7 with security considerations.

2. Terminology

This section introduces the terminology that is used in this document.

Peer

A peer is an IKEv2 endpoint. In addition, a peer implements the MOBIKE extensions, defined in [RFC4555].

Available address

An address is said to be available if the following conditions are met:

* The address has been assigned to an interface.

* If the address is an IPv6 address, we additionally require (a) that the address is valid as defined in RFC 2461 [RFC2461], and (b) that the address is not tentative as defined in RFC 2462 [RFC2462]. In other words, we require the address assignment to be complete.

  Note that this explicitly allows an address to be optimistic as defined in [RFC4429].

* If the address is an IPv6 address, it is a global unicast or unique site-local address, as defined in [RFC4193]. That is, it is not an IPv6 link-local address.

* The address and interface is acceptable for sending and receiving traffic according to a local policy.

This definition is taken from [WIP-Ark06] and adapted for the MOBIKE context.

Locally operational address

An address is said to be locally operational if it is available and its use is locally known to be possible and permitted. This definition is taken from [WIP-Ark06].

Operational address pair

A pair of operational addresses are said to be an operational address pair if and only if bidirectional connectivity can be shown between the two addresses. Note that sometimes it is necessary to consider connectivity on a per-flow level between two endpoints. This differentiation might be necessary to address certain Network Address Translation types or specific firewalls. This definition is taken from [WIP-Ark06] and adapted for the MOBIKE context. Although it is possible to further differentiate unidirectional and bidirectional operational address pairs, only bidirectional connectivity is relevant to this document, and unidirectional connectivity is out of scope.

Path

The sequence of routers traversed by the MOBIKE and IPsec packets exchanged between the two peers. Note that this path may be affected not only by the involved source and destination IP addresses, but also by the transport protocol. Since MOBIKE and IPsec packets have a different appearance on the wire, they might be routed along a different path, for example, due to load balancing. This definition is taken from [RFC2960] and adapted to the MOBIKE context.

Current path

The sequence of routers traversed by an IP packet that carries the default source and destination addresses is said to be the Current Path. This definition is taken from [RFC2960] and adapted to the MOBIKE context.

Preferred address

The IP address of a peer to which MOBIKE and IPsec traffic should be sent by default. A given peer has only one active preferred address at a given point in time, except for the small time period where it switches from an old to a new preferred address. This definition is taken from [WIP-Nik06] and adapted to the MOBIKE context.

Peer address set

We denote the two peers of a MOBIKE session by peer A and peer B. A peer address set is the subset of locally operational addresses of peer A that is sent to peer B. A policy available at peer A indicates which addresses are included in the peer address set. Such a policy might be created either manually or automatically through interaction with other mechanisms that indicate new available addresses.

Bidirectional address pair

The address pair, where traffic can be sent to both directions, simply by reversing the IP addresses. Note that the path of the packets going to each direction might be different.

Unidirectional address pair

The address pair, where traffic can only be sent in one direction, and reversing the IP addresses and sending reply back does not work.

For mobility-related terminology (e.g., Make-before-break or Break-before-make), see [RFC3753].

3. Scenarios

In this section, we discuss three typical usage scenarios for the MOBIKE protocol.

3.1. Mobility Scenario

Figure 1 shows a break-before-make mobility scenario where a mobile node (MN) changes its point of network attachment. Prior to the change, the mobile node had established an IPsec connection with a security gateway that offered, for example, access to a corporate network. The IKEv2 exchange that facilitated the setup of the IPsec SA(s) took place over the path labeled as ’old path’. The involved packets carried the MN’s "old" IP address and were forwarded by the "old" access router (OAR) to the security gateway (GW).

When the MN changes its point of network attachment, it obtains a new IP address using stateful or stateless address configuration. The goal of MOBIKE, in this scenario, is to enable the MN and the GW to continue using the existing SAs and to avoid setting up a new IKE SA. A protocol exchange, denoted by ’MOBIKE Address Update’, enables the peers to update their state as necessary.

Note that in a break-before-make scenario the MN obtains the new IP address after it can no longer be reached at the old IP address. In a make-before-break scenario, the MN is, for a given period of time, reachable at both the old and the new IP address. MOBIKE should work in both of the above scenarios.

```
                          (Initial IKEv2 Exchange)
                    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>v
       Old IP   +--+        +----+                   v
       address  |MN|------> |OAR| -------------V     v
                +--+        +----+ Old path    V     v
                 .                          +----+   v>>>>> +--+
                 .move                      | R  | -------> |GW|
                 .                          |    |    >>>>> |  |
                 v                          +----+   ^      +--+
                +--+        +----+ New path    ^     ^
       New IP   |MN|------> |NAR|--------------^     ^
       address  +--+        +----+                   ^
                    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>^
                          (MOBIKE Address Update)

              ---> = Path taken by data packets
              >>>> = Signaling traffic (IKEv2 and MOBIKE)
              ...> = End host movement
```

Figure 1: Mobility Scenario

3.2. Multihoming Scenario

Another MOBIKE usage scenario is depicted in Figure 2. In this scenario, the MOBIKE peers are equipped with multiple interfaces (and multiple IP addresses). Peer A has two interface cards with two IP addresses, IP_A1 and IP_A2, and peer B has two IP addresses, IP_B1 and IP_B2. Each peer selects one of its IP addresses as the preferred address, which is used for subsequent communication. Various reasons (e.g., hardware or network link failures) may require a peer to switch from one interface to another.

```
     +------------+                                  +------------+
     | Peer A     |           *~~~~~~~~~*            | Peer B     |
     |            |>>>>>>>>>> * Network   *>>>>>>>>>>|            |
     |      IP_A1 +-------->+             +--------->+ IP_B1      |
     |            |         |             |          |            |
     |      IP_A2 +********>+             +*********>+ IP_B2      |
     |            |          *           *           |            |
     +------------+           *~~~~~~~~~*            +------------+

              ---> = Path taken by data packets
              >>>> = Signaling traffic (IKEv2 and MOBIKE)
              ***> = Potential future path through the network
                     (if Peer A and Peer B change their preferred
                      address)
```

Figure 2: Multihoming Scenario

Note that MOBIKE does not aim to support load balancing between multiple IP addresses. That is, each peer uses only one of the available address pairs at a given point in time.

3.3. Multihomed Laptop Scenario

The third scenario we consider is about a laptop that has multiple interface cards and therefore several ways to connect to the network. It may, for example, have a fixed Ethernet card, a WLAN interface, a General Packet Radio Service (GPRS) adaptor, a Bluetooth interface, or USB hardware. Not all interfaces are used for communication all the time for a number of reasons (e.g., cost, network availability, user convenience). The policies that determine which interfaces are connected to the network at any given point in time is outside the scope of the MOBIKE protocol and, as such, this document. However, as the laptop changes its point of attachment to the network, the set of IP addresses under which the laptop is reachable changes too.

In all of these scenarios, even if IP addresses change due to interface switching or mobility, the IP address obtained via the configuration payloads within IKEv2 remain unaffected. The IP address obtained via the IKEv2 configuration payloads allow the configuration of the inner IP address of the IPsec tunnel. As such, applications might not detect any change at all.

4. Scope of MOBIKE

Getting mobility and multihoming actually working requires many different components to work together, including coordinating decisions between different layers, different mobility mechanisms, and IPsec/IKEv2. Most of those aspects are beyond the scope of MOBIKE: MOBIKE focuses only on what two peers need in order to agree at the IKEv2 level (like new message formats and some aspects of their processing) required for interoperability.

The MOBIKE protocol is not trying to be a full mobility protocol; there is no support for simultaneous movement or rendezvous mechanism, and there is no support for route optimization, etc. The design document focuses on tunnel mode; everything going inside the tunnel is unaffected by the changes in the tunnel header IP address, and this is the mobility feature provided by the MOBIKE. That is, applications running inside the MOBIKE-controlled IPsec tunnel might not detect the movement since their IP addresses remain constant.

The MOBIKE protocol should be able to perform the following operations (not all of which are done explicitly by the current protocol):

- Inform the other peer about the peer address set

- Inform the other peer about the preferred address

- Test connectivity along a path and thereby detect an outage situation

- Change the preferred address

- Change the peer address set

- Ability to deal with Network Address Translation devices

Figure 3 shows an example protocol interaction between a pair of MOBIKE peers. MOBIKE interacts with the packet processing module of the IPsec implementation using an internal API (such as those based on PF_KEY [RFC2367]). Using this API, the MOBIKE module can create entries in the Security Association (SAD) and Security Policy Databases (SPD). The packet processing module of the IPsec implementation may also interact with IKEv2 and MOBIKE module using this API. The content of the Security Policy and Security Association Databases determines what traffic is protected with IPsec in which fashion. MOBIKE, on the other hand, receives information from a number of sources that may run both in kernel-mode and in user-mode. These sources form the basis on which MOBIKE makes decisions regarding the set of available addresses, the peer address set, and the preferred address. Policies may also affect the selection process.

The peer address set and the preferred address needs to be made available to the other peer. In order to address certain failure cases, MOBIKE should perform connectivity tests between the peers (potentially over a number of different paths). Although a number of address pairs may be available for such tests, the most important is the pair (source address, destination address) of the current path. This is because this pair is selected for sending and receiving MOBIKE signaling and IPsec traffic. If a problem along this current path is detected (e.g., due to a router failure), it is necessary to switch to a new current path. In order to be able to do so quickly, it may be helpful to perform connectivity tests of other paths periodically. Such a technique would also help identify previously disconnected paths that become operational again.

```
     +---------------------+            +----------------+
     |    User-space       |            |                |
     |   Protocols and     |            |   MOBIKE and   |
     | Functions Relevant  |<---------->|  IKEv2 Module  |
     | MOBIKE (e.g., DHCP, |            |                |
     |     policies)       |            +----------------+
     +---------------------+                    ^
                ^                               |
                |                               |        User space
     ++++++++++API++++++++++++++++++++++++++++PF_KEY+++++++++++++++
                |                               |      Kernel space
                |                               v
                |                       +----------------+
                v                       |                |
     +---------------------+            |  IPsec engine  |
     |   Kernel-space      |<---------->| (and databases)|
     |     Protocols       |            |                |
     |    Relevant for     |            +----------------+
     |  MOBIKE (e.g., ND,  |                    ^
     |     DNA, L2)        |<---------------+   |
     +---------------------+                v   v
            ||                          +----------------+
            \/                          |                |
          Inter-  =====================>| IP forwarding, |
          faces   <=====================|input and output|
                                        |                |
                                        +----------------+

         ===> = IP packets arriving/leaving a MOBIKE node
         <->  = control and configuration operations
```

Figure 3: Framework

Please note that Figure 3 illustrates an example of how a MOBIKE implementation could work. It serves illustrative purposes only.

5. Design Considerations

This section discusses aspects affecting the design of the MOBIKE protocol.

5.1. Choosing Addresses

One of the core aspects of the MOBIKE protocol is the selection of the address for the IPsec packets we send. Choosing addresses for the IKEv2 request is a somewhat separate problem. In many cases, they will be the same (and in some design choice they will always be the same and could be forced to be the same by design).

5.1.1. Inputs and Triggers

How address changes are triggered is largely beyond the scope of MOBIKE. The triggers can include changes in the set of addresses, various link-layer indications, failing dead peer detection, and changes in preferences and policies. Furthermore, there may be less reliable sources of information (such as lack of IPsec packets and incoming ICMP packets) that do not trigger any changes directly, but rather cause Dead Peer Detection (DPD) to be scheduled earlier and, if it fails, it might cause a change of the preferred address.

These triggers are largely the same as for other mobility protocols such as Mobile IP, and they are beyond the scope of MOBIKE.

5.1.2. Connectivity

There can be two kinds of connectivity "failures": local failures and path failures. Local failures are problems locally at a MOBIKE peer (e.g., an interface error). Path failures are a property of an address pair and failures of nodes and links along this path. MOBIKE does not support unidirectional address pairs. Supporting them would require abandoning the principle of sending an IKEv2 reply to the address from which the request came. MOBIKE decided to deal only with bidirectional address pairs. It does consider unidirectional address pairs as broken and does not use them, but the connection between peers will not break even if unidirectional address pairs are present, provided there is at least one bidirectional address pair MOBIKE can use.

Note that MOBIKE is not concerned about the actual path used; it cannot even detect if some path is unidirectionally operational if the same address pair has some other unidirectional path back. Ingress filters might still cause such path pairs to be unusable, and in that case MOBIKE will detect that there is no operational address pair.

In a sense having both an IPv4 and an IPv6 address is basically a case of partial connectivity (putting both an IPv4 and an IPv6 address in the same IP header does not work). The main difference is that it is known beforehand; there is no need to discover that an IPv4/IPv6 combination does not work.

5.1.3. Discovering Connectivity

To detect connectivity, the MOBIKE protocol needs to have a mechanism to test connectivity. If a MOBIKE peer receives a reply, it can be sure about the existence of a working (bidirectional) address pair. If a MOBIKE peer does not see a reply after multiple retransmissions, it may assume that the tested address pair is broken.

The connectivity tests require congestion problems to be taken into account because the connection failure might be caused by congestion. The MOBIKE protocol should not make the congestion problem worse by sending many DPD packets.

5.1.4. Decision Making

One of the main questions in designing the MOBIKE protocol was who makes the decisions how to fix a situation when failure is detected, e.g., symmetry vs. asymmetry in decision making. Symmetric decision making (i.e., both peers can make decisions) may cause the different peers to make different decisions, thus causing asymmetric upstream/ downstream traffic. In the mobility case, it is desirable that the mobile peer can move both upstream and downstream traffic to some particular interface, and this requires asymmetric decision making (i.e. only one peer makes decisions).

Working with stateful packet filters and NATs is easier if the same address pair is used in both upstream and downstream directions. Also, in common cases, only the peer behind NAT can actually perform actions to recover from the connectivity problems, as the other peer might not be able to initiate any connections to the peer behind NAT.

5.1.5. Suggested Approach

The working group decided to select a method whereby the initiator will decide which addresses are used. As a consequence, the outcome is always the same for both parties. It also works best with NATs, as the initiator is most likely the node that is located behind a NAT.

5.2. NAT Traversal (NAT-T)

5.2.1. Background and Constraints

Another core aspect of MOBIKE is the treatment of different NATs and Network Address Port Translations (NAPTs). In IKEv2 the tunnel header IP addresses are not sent inside the IKEv2 payloads, and thus there is no need to do unilateral self-address fixing (UNSAF

[RFC3424]). The tunnel header IP addresses are taken from the outer IP header of the IKE packets; thus, they are already processed by the NAT.

The NAT detection payloads are used to determine whether the addresses in the IP header were modified by a NAT along the path. Detecting a NAT typically requires UDP encapsulation of IPsec ESP packets to be enabled, if desired. MOBIKE is not to change how IKEv2 NAT-T works in particular, any kind of UNSAF or explicit interaction with NATs (e.g., MIDCOM [RFC3303] or NSIS NATFW NSLP [WIP-Sti06]) is beyond the scope of the MOBIKE protocol. The MOBIKE protocol will need to define how MOBIKE and NAT-T are used together.

The NAT-T support should also be optional. If the IKEv2 implementation does not implement NAT-T, as it is not required in some particular environment, implementing MOBIKE should not require adding support for NAT-T either.

The property of being behind NAT is actually a property of the address pair and thereby of the path taken by a packet. Thus, one peer can have multiple IP addresses, and some of those might be behind NAT and some might not.

5.2.2. Fundamental Restrictions

There are some cases that cannot be carried out within MOBIKE. One of those cases is when the party "outside" a symmetric NAT changes its address to something not known by the other peer (and the old address has stopped working). It cannot send a packet containing the new addresses to the peer because the NAT does not contain the necessary state. Furthermore, since the party behind the NAT does not know the new IP address, it cannot cause the NAT state to be created.

This case could be solved using some rendezvous mechanism outside IKEv2, but that is beyond the scope of MOBIKE.

5.2.3. Moving behind a NAT and Back

The MOBIKE protocol should provide a mechanism whereby a peer that is initially not behind a NAT can move behind NAT when a new preferred address is selected. The same effect might be accomplished with the change of the address pair if more than one path is available (e.g., in the case of a multi-homed host). An impact for the MOBIKE protocol is only caused when the currently selected address pair causes MOBIKE packets to traverse a NAT.

Similarly, the MOBIKE protocol provides a mechanism to detect when a NATed path is changed to a non-NATed path with the change of the address pair.

As we only use one address pair at time, effectively the MOBIKE peer is either behind NAT or not behind NAT, but each address change can change this situation. Because of this, and because the initiator always chooses the addresses, it is enough to send keepalive packets only to that one address pair.

Enabling NAT-T involves a few different things. One is to enable the UDP encapsulation of ESP packets. Another is to change the IKE SA ports from port 500 to port 4500. We do not want to do unnecessary UDP encapsulation unless there is really a NAT between peers, i.e., UDP encapsulation should only be enabled when we actually detect NAT. On the other hand, as all implementations supporting NAT-T must be able to respond to port 4500 all the time, it is simpler from the protocol point of view to change the port numbers from 500 to 4500 immediately upon detecting that the other end supports NAT-T. This way it is not necessary to change ports after we later detected NAT, which would have caused complications to the protocol.

If we changed the port only after we detected NAT, then the responder would not be able to use the IKE and IPsec SAs immediately after their address is changed to be behind NAT. Instead, it would need to wait for the next packet from the initiator to see what IP and port numbers are used after the initiator changed its port from 500 to 4500. The responder would also not be able to send anything to the initiator before the initiator sent something to the responder. If we do the port number changing immediately after the IKE_SA_INIT and before IKE_AUTH phase, then we get the rid of this problem.

5.2.4. Responder behind a NAT

MOBIKE can work in cases where the responder is behind a static NAT, but the initiator would need to know all the possible addresses to which the responder can move. That is, the responder cannot move to an address which is not known by the initiator, in case initiator also moves behind NAT.

If the responder is behind a NAPT, then it might need to communicate with the NAT to create a mapping so the initiator can connect to it. Those external firewall pinhole opening mechanisms are beyond the scope of MOBIKE.

In case the responder is behind NAPT, then finding the port numbers used by the responder is outside the scope of MOBIKE.

5.2.5. NAT Prevention

One new feature created by MOBIKE is NAT prevention. If we detect NAT between the peers, we do not allow that address pair to be used. This can be used to protect IP addresses in cases where the configuration knows that there is no NAT between the nodes (for example IPv6, or fixed site-to-site VPN). This avoids any possibility of on-path attackers modifying addresses in headers. This feature means that we authenticate the IP address and detect if they were changed. As this is done on purpose to break the connectivity if NAT is detected, and decided by the configuration, there is no need to do UNSAF processing.

5.2.6. Suggested Approach

The working group decided that MOBIKE uses NAT-T mechanisms from the IKEv2 protocol as much as possible, but decided to change the dynamic address update (see [RFC4306], Section 2.23, second to last paragraph) for IKEv2 packets to "MUST NOT" (it would break path testing using IKEv2 packets; see Section 6.2). The working group also decided only to send keepalives to the current address pair.

5.3. Scope of SA Changes

Most sections of this document discuss design considerations for updating and maintaining addresses in the database entries that relate to an IKE SA. However, changing the preferred address also affects the entries of the IPsec SA database. The outer tunnel header addresses (source and destination IP addresses) need to be modified according to the current path to allow the IPsec protected data traffic to travel along the same path as the MOBIKE packets. If the MOBIKE messages and the IPsec protected data traffic travel along a different path, then NAT handling is severely complicated.

The basic question is then how the IPsec SAs are changed to use the new address pair (the same address pair as the MOBIKE signaling traffic). One option is that when the IKE SA address is changed, all IPsec SAs associated with it are automatically moved along with it to a new address pair. Another option is to have a separate exchange to move the IPsec SAs separately.

If IPsec SAs should be updated separately, then a more efficient format than the Notify payload is needed to preserve bandwidth. A Notify payload can only store one Security Parameter Index (SPI) per payload. A separate payload could have a list of IPsec SA SPIs and the new preferred address. If there is a large number of IPsec SAs, those payloads can be quite large unless list of ranges of SPI values are supported. If we automatically move all IPsec SAs when the IKE

SA moves, then we only need to keep track of which IKE SA was used to create the IPsec SA, and fetch the IP addresses from the IKE SA, i.e., there is no need to store IP addresses per IPsec SA. Note that IKEv2 [RFC4306] already requires the implementations to keep track of which IPsec SAs are created using which IKE SA.

If we do allow the address set of each IPsec SA to be updated separately, then we can support scenarios where the machine has fast and/or cheap connections and slow and/or expensive connections and wants to allow moving some of the SAs to the slower and/or more expensive connection, and prevent the move, for example, of the news video stream from the WLAN to the GPRS link.

On the other hand, even if we tie the IKE SA update to the IPsec SA update, we can create separate IKE SAs for this scenario. For example, we create one IKE SA that has both links as endpoints, and it is used for important traffic; then we create another IKE SA which has only the fast and/or cheap connection, which is used for that kind of bulk traffic.

The working group decided to move all IPsec SAs implicitly when the IKE SA address pair changes. If more granular handling of the IPsec SA is required, then multiple IKE SAs can be created one for each set of IPsec SAs needed.

5.4. Zero Address Set Functionality

One of the features that is potentially useful is for the peer to announce that it will now disconnect for some time, i.e., it will not be reachable at all. For instance, a laptop might go to suspend mode. In this case, it could send address notification with zero new addresses, which would mean that it will not have any valid addresses anymore. The responder would then acknowledge that notification and could then temporarily disable all SAs and therefore stop sending traffic. If any of the SAs get any packets, they are simply dropped. This could also include some kind of ACK spoofing to keep the TCP/IP sessions alive (or simply setting the TCP/IP keepalives and timeouts large enough not to cause problems), or it could simply be left to the applications, e.g., allow TCP/IP sessions to notice if the link is broken.

The local policy could then indicate how long the peer should allow remote peers to remain disconnected.

From a technical point of view, this would provide following two features:

- There is no need to transmit IPsec data traffic. IPsec-protected data can be dropped, which saves bandwidth. This does not provide a functional benefit, i.e., nothing breaks if this feature is not provided.

- MOBIKE signaling messages are also ignored. The IKE SA must not be deleted, and the suspend functionality (realized with the zero address set) may require the IKE SA to be tagged with a lifetime value since the IKE SA should not be kept alive for an undefined period of time. Note that IKEv2 does not require that the IKE SA has a lifetime associated with it. In order to prevent the IKE SA from being deleted, the dead-peer detection mechanism needs to be suspended as well.

Due to its complexity and no clear requirement for it, it was decided that MOBIKE does not support this feature.

5.5. Return Routability Check

Changing the preferred address and subsequently using it for communication is associated with an authorization decision: Is a peer allowed to use this address? Does this peer own this address? Two mechanisms have been proposed in the past to allow a peer to determine the answer to these questions:

- The addresses a peer is using are part of a certificate. [RFC3554] introduced this approach. If the other peer is, for example, a security gateway with a limited set of fixed IP addresses, then the security gateway may have a certificate with all the IP addresses appearing in the certificate.

- A return routability check is performed by the remote peer before the address is updated in that peer’s Security Association Database. This is done in order to provide a certain degree of confidence to the remote peer that the local peer is reachable at the indicated address.

Without taking an authorization decision, a malicious peer can redirect traffic towards a third party or a black hole.

A MOBIKE peer should not use an IP address provided by another MOBIKE peer as a current address without computing the authorization decision. If the addresses are part of the certificate, then it is not necessary to execute the return routability check. The return routability check is a form of authorization check, although it provides weaker guarantees than the inclusion of the IP address as a part of a certificate. If multiple addresses are communicated to the remote peer, then some of these addresses may be already verified.

Finally, it would be possible not to execute return routability checks at all. In case of indirect change notifications (i.e., something we notice from the network, not from the peer directly), we only move to the new preferred address after successful dead-peer detection (i.e., a response to a DPD test) on the new address, which is already a return routability check. With a direct notification (i.e., notification from the other end directly) the authenticated peer may have provided an authenticated IP address (i.e., inside IKE encrypted and authenticated payload; see Section 5.2.5). Thus, it is would be possible simply to trust the MOBIKE peer to provide a proper IP address. In this case, a protection against an internal attacker (i.e., the authenticated peer forwarding its traffic to the new address) would not provided. On the other hand, we know the identity of the peer in that case. There might be problems when extensions are added to IKEv2 that do not require authentication of end points (e.g., opportunistic security using anonymous Diffie-Hellman).

There is also a policy issue of when to schedule a return routability check. Before moving traffic? After moving traffic?

The basic format of the return routability check could be similar to dead-peer detection, but potential attacks are possible if a return routability check does not include some kind of a nonce. In these attacks, the valid end point could send an address update notification for a third party, trying to get all the traffic to be sent there, causing a denial-of-service attack. If the return routability check does not contain any nonce or other random information not known to the other peer, then the other peer could reply to the return routability checks even when it cannot see the request. This might cause a peer to move the traffic to a location where the original recipient cannot be reached.

The IKEv2 NAT-T mechanism does not perform return routability checks. It simply uses the last seen source IP address used by the other peer as the destination address to which response packets are to be sent. An adversary can change those IP addresses and can cause the response packets to be sent to a wrong IP address. The situation is self-fixing when the adversary is no longer able to modify packets and the first packet with an unmodified IP address reaches the other peer. Mobility environments make this attack more difficult for an adversary since the attack requires the adversary to be located somewhere on the individual paths ({CoA1, ..., CoAn} towards the destination IP address), to have a shared path, or, if the adversary is located near the MOBIKE client, to follow the user mobility patterns. With IKEv2 NAT-T, the genuine client can cause third-party bombing by redirecting all the traffic pointed to him to a third party. As the MOBIKE protocol tries to provide equal or better security than IKEv2 NAT-T mechanism, it should protect against these attacks.

There may be return routability information available from the other parts of the system too (as shown in Figure 3), but the checks done may have a different quality. There are multiple levels for return routability checks:

- None; no tests.

- A party willing to answer the return routability check is located along the path to the claimed address. This is the basic form of return routability check.

- There is an answer from the tested address, and that answer was authenticated and integrity- and replay-protected.

- There was an authenticated and integrity- and replay-protected answer from the peer, but it is not guaranteed to originate at the tested address or path to it (because the peer can construct a response without seeing the request).

The return routability checks do not protect against third-party bombing if the attacker is along the path, as the attacker can forward the return routability checks to the real peer (even if those packets are cryptographically authenticated).

If the address to be tested is carried inside the MOBIKE payload, then the adversary cannot forward packets. Thus, third-party bombings are prevented (see Section 5.2.5).

If the reply packet can be constructed without seeing the request packet (for example, if there is no nonce, challenge, or similar mechanism to show liveness), then the genuine peer can cause third-party bombing, by replying to those requests without seeing them at all.

Other levels might only provide a guarantee that there is a node at the IP address that replied to the request. There is no indication as to whether or not the reply is fresh or whether or not the request may have been transmitted from a different source address.

5.5.1. Employing MOBIKE Results in Other Protocols

If MOBIKE has learned about new locations or verified the validity of a remote address through a return routability check, can this information be useful for other protocols?

When the basic MOBIKE VPN scenario is considered, the answer is no. Transport and application layer protocols running inside the VPN tunnel are unaware of the outer addresses or their status.

Similarly, IP-layer tunnel termination at a gateway rather than a host endpoint limits the benefits for "other protocols" that could be informed -- all application protocols at the other side are unaware of IPsec, IKE, or MOBIKE.

However, it is conceivable that future uses or extensions of the MOBIKE protocol make such information distribution useful. For instance, if transport mode MOBIKE and SCTP were made to work together, it would potentially be useful for SCTP dynamic address reconfiguration [WIP-Ste06] to learn about the new addresses at the same time as MOBIKE. Similarly, various IP-layer mechanisms may make use of the fact that a return routability check of a specific type has been performed. However, care should be exercised in all these situations.

[WIP-Cro04] discusses the use of common locator information pools in a IPv6 multi-homing context; it assumes that both transport and IP-layer solutions are used in order to support multi-homing, and that it would be beneficial for different protocols to coordinate their results in some way, for instance, by sharing throughput information of address pairs. This may apply to MOBIKE as well, assuming it coexists with non-IPsec protocols that are faced with the same or similar multi-homing choices.

Nevertheless, all of this is outside the scope of the current MOBIKE base protocol design and may be addressed in future work.

5.5.2. Return Routability Failures

If the return routability check fails, we need to tear down the IKE SA if we are using IKEv2 INFORMATIONAL exchanges to send return routability checks. On the other hand, return routability checks can only fail permanently if there was an attack by the other end; thus, tearing down the IKE SA is a suitable action in that case.

There are some cases, where the return routability check temporarily fails, that need to be considered here. In the first case, there is no attacker, but the selected address pair stops working immediately after the address update, before the return routability check.

What happens is that the initiator performs the normal address update; it succeeds, and then the responder starts a return routability check. If the address pair has broken down before that, the responder will never get back the reply to the return routability check. The responder might still be using the old IP address pair, which could still work.

The initiator might be still seeing traffic from the responder, but using the old address pair. The initiator should detect that this traffic is not using the latest address pair, and after a while it should start dead peer detection on the current address pair. If that fails, then it should find a new working address pair and update addresses to that. The responder should notice that the address pair was updated after the return routability check was started and change the ongoing return routability check to use the new address pair. The result of that return routability check needs to be discarded as it cannot be trusted; the packets were retransmitted to a different IP address. So normally the responder starts a new return routability check afterward with the new address pair.

The second case is where there is an attacker along the path modifying the IP addresses. The peers will detect this as NAT and will enable NAT-T recovery of changes in the NAT mappings. If the attacker is along the path long enough for the return routability check to succeed, then the normal recovery of changes in the NAT mappings will take care of the problem. If the attacker disappears before return routability check is finished, but after the update, we have a case similar to the last. The only difference is that now the dead peer detection started by the initiator will succeed because the responder will reply to the addresses in the headers, not the current address pair. The initiator will then detect that the NAT mappings are changed, and it will fix the situation by doing an address update.

The important thing for both of these cases is that the initiator needs to see that the responder is both alive and synchronized with initiator address pair updates. That is, it is not enough that the responder is sending traffic to an initiator; it must also be using the correct IP addresses before the initiator can believe it is alive and synchronized. From the implementation point of view, this means that the initiator must not consider packets having wrong IP addresses as packets that prove the other end is alive, i.e., they do not reset the dead peer detection timers.

5.5.3. Suggested Approach

The working group selected to use IKEv2 INFORMATIONAL exchanges as a return routability check, but included a random cookie to prevent redirection by an authenticated attacker. Return routability checks are performed by default before moving the traffic. However, these tests are optional. Nodes may also perform these tests upon their own initiative at other times.

It is worth noting that the return routability check in MOBIKE is different from Mobile IPv6 [RFC3775], which does not perform return routability operations between the mobile node and its home agent at all.

5.6. IPsec Tunnel or Transport Mode

The current MOBIKE design is focused only on the VPN type usage and tunnel mode. Transport mode behavior would also be useful and might be discussed in future documents.

6. Protocol Details

6.1. Indicating Support for MOBIKE

In order for MOBIKE to function, both peers must implement the MOBIKE extension of IKEv2. If one of the peers does not support MOBIKE, then, whenever an IP address changes, IKEv2 will have to be re-run in order to create a new IKE SA and the respective IPsec SAs. In MOBIKE, a peer needs to be confident that its address change messages are understood by the other peer. If these messages are not understood, it is possible that connectivity between the peers is lost.

One way to ensure that a peer receives feedback on whether its messages are understood by the other peer is to use IKEv2 messaging for MOBIKE and to mark some messages as "critical". According to the IKEv2 specification, either such messages have to be understood by the receiver, or an error message has to be returned to the sender.

A second way to ensure receipt of the above-mentioned feedback is by using Vendor ID payloads that are exchanged during the initial IKEv2 exchange. These payloads would then indicate whether or not a given peer supports the MOBIKE protocol.

A third approach would use the Notify payload to indicate support of MOBIKE extension. Such Notify payloads are also used for indicating NAT traversal support (via NAT_DETECTION_SOURCE_IP and NAT_DETECTION_DESTINATION_IP payloads).

Both a Vendor ID and a Notify payload may be used to indicate the support of certain extensions.

Note that a MOBIKE peer could also attempt to execute MOBIKE opportunistically with the critical bit set when an address change has occurred. The drawback of this approach is, however, that an unnecessary message exchange is introduced.

Although Vendor ID payloads and Notify payloads are technically equivalent, Notify payloads are already used in IKEv2 as a capability negotiation mechanism. Hence, Notify payloads are used in MOBIKE to indicate support of MOBIKE protocol.

Also, as the information of the support of MOBIKE is not needed during the IKE_SA_INIT exchange, the indication of the support is done inside the IKE_AUTH exchange. The reason for this is the need to keep the IKE_SA_INIT messages as small as possible so that they do not get fragmented. IKEv2 allows that the responder can do stateless processing of the first IKE_SA_INIT packet and request a cookie from the other end if it is under attack. To mandate the responder to be able to reassemble initial IKE_SA_INIT packets would not allow fully stateless processing of the initial IKE_SA_INIT packets.

6.2. Path Testing and Window size

As IKEv2 has a window of outgoing messages, and the sender is not allowed to violate that window (meaning that if the window is full, then the sender cannot send packets), it can cause some complications to path testing. Another complication created by IKEv2 is that once the message is created and sent to the other end, it cannot be modified in its future retransmissions. This makes it impossible to know what packet actually reached the other end first. We cannot use IP headers to find out which packet reached the other end first because if the responder gets retransmissions of the packet it has already processed and replied to (and those replies might have been lost due unidirectional address pair), it will retransmit the previous reply using the new address pair of the request. Because of this, it might be possible that the responder has already used the IP address information from the header of the previous packet, and the reply packet ending up at the initiator has a different address pair.

Another complication comes from NAT-T. The current IKEv2 document says that if NAT-T is enabled, the node not behind NAT SHOULD detect if the IP address changes in the incoming authenticated packets and update the remote peers’ addresses accordingly. This works fine with NAT-T, but it causes some complications in MOBIKE, as MOBIKE needs the ability to probe other address pairs without breaking the old one.

One approach to fix this would be to add a completely new protocol that is outside the IKE SA message id limitations (window code), outside identical retransmission requirements, and outside the dynamic address updating of NAT-T.

Another approach is to make the protocol so that it does not violate window restrictions and does not require changing the packet on retransmissions, and change the dynamic address updating of NAT-T to "MUST NOT" for IKE SA packets if MOBIKE is used. In order not to violate window restrictions, the addresses of the currently ongoing exchange need to be changed to test different paths. In order not to require that the packet be changed after it is first sent requires that the protocol restart from the beginning in case the packet was retransmitted to different addresses (because the sender does not know which packet the responder got first, i.e., which IP addresses it used).

The working group decided to use normal IKEv2 exchanges for path testing and decided to change the dynamic address updating of NAT-T to MUST NOT for IKE SA packets; a new protocol outside of IKEv2 was not adopted.

6.3. Message Presentation

The IP address change notifications can be sent either via an informational exchange already specified in IKEv2, or via a MOBIKE-specific message exchange. Using an informational exchange has the main advantage that it is already specified in the IKEv2 protocol and implementations can already incorporate the functionality.

Another question is the format of the address update notifications. The address update notifications can include multiple addresses, of which some may be IPv4 and some IPv6 addresses. The number of addresses is most likely going to be limited in typical environments (with less than 10 addresses). The format may need to indicate a preference value for each address. The format could either contain a preference number that determines the relative order of the addresses or could simply be an ordered list of IP addresses. If using preference numbers, then two addresses can have the same preference value; an ordered list avoids this situation.

Load balancing is currently outside the scope of MOBIKE; however, future work might include support for it. The selected format needs to be flexible enough to include additional information in future versions of the protocol (e.g., to enable load balancing). This may be realized with an reserved field, which can later be used to store additional information. As other information may arise that may have to be tied to an address in the future, a reserved field seems like a prudent design in any case.

There are two basic formats that place IP address lists into a message. One includes each IP address as separate payload (where the payload order indicates the preference order, or the payload itself might include the preference number). Alternatively, we can put the IP address list as one payload to the exchange, and that one payload will then have an internal format that includes the list of IP addresses.

Having multiple payloads, each one carrying one IP address, makes the protocol probably easier to parse, as we can already use the normal IKEv2 payload parsing procedures. It also offers an easy way for the extensions, as the payload probably contains only the type of the IP address (or the type is encoded to the payload type), and the IP address itself. As each payload already has a length field associated to it, we can detect if there is any extra data after the IP address. Some implementations might have problems parsing more than a certain number of IKEv2 payloads, but if the sender sends them in the most preferred first, the receiver can only use the first addresses it was willing to parse.

Having all IP addresses in one big MOBIKE-specified internal format provides more compact encoding and keeps the MOBIKE implementation more concentrated to one module.

Another choice is which type of payloads to use. IKEv2 already specifies a Notify payload. It includes some extra fields (SPI size, SPI, protocol, etc.), which gives 4 bytes of the extra overhead, and there is the notification data field, which could include the MOBIKE-specific data.

Another option would be to have a custom payload type, which would then include the information needed for the MOBIKE protocol.

The working group decided to use IKEv2 Notify payloads, and put only one data item per notify. There will be one Notify payload for each item to be sent.

6.4. Updating Address Set

Because the initiator decides all address updates, the initiator needs to know all the addresses used by the responder. The responder also needs that list in case it happens to move to an address not known by the initiator, and it needs to send an address update notification to the initiator. It might need to try different addresses for the initiator.

MOBIKE could send the whole peer address list every time any of the IP addresses change (addresses are added or removed, the order changes, or the preferred address is updated) or an incremental update. Sending incremental updates provides more compact packets (meaning we can support more IP addresses), but on the other hand this approach has more problems in the synchronization and packet reordering cases. That is, incremental updates must be processed in order, but for full updates we can simply use the most recent one and ignore old ones, even if they arrive after the most recent one (IKEv2 packets have a message ID that is incremented for each packet; thus, it is easy to know the sending order).

The working group decided to use a protocol format where both ends send a full list of their addresses to the other end, and that list overwrites the previous list. To support NAT-T, the IP addresses of the received packet are considered as one address of the peer, even when they are not present in the list.

7. Security Considerations

As all the packets are already authenticated by IKEv2, there is no risk that any attackers would undetectedly modify the contents of the packets. The IP addresses in the IP header of the packets are not authenticated; thus, the protocol defined must take care that they are only used as an indication that something might be different, and that they do not cause any direct actions, except when doing NAT traversal.

An attacker can also spoof ICMP error messages in an effort to confuse the peers about which addresses are not working. At worst, this causes denial of service and/or the use of non-preferred addresses.

One type of attack that needs to be taken care of in the MOBIKE protocol is the bombing attack type. See [RFC4225] and [Aur02] for more information about flooding attacks.

See the security considerations section of [RFC4555] for more information about security considerations of the actual protocol.

8. Acknowledgements

This document is the result of discussions in the MOBIKE working group. The authors would like to thank Jari Arkko, Pasi Eronen, Francis Dupont, Mohan Parthasarathy, Paul Hoffman, Bill Sommerfeld, James Kempf, Vijay Devarapalli, Atul Sharma, Bora Akyol, Joe Touch, Udo Schilcher, Tom Henderson, Andreas Pashalidis, and Maureen Stillman for their input.

We would like to particularly thank Pasi Eronen for tracking open issues on the MOBIKE mailing list. He helped us make good progress on the document.

9. References

9.1. Normative references

[RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, December 2005.

[RFC4306] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", RFC 4306, December 2005.

9.2. Informative References

[Aur02] Aura, T., Roe, M., and J. Arkko, "Security of Internet Location Management", In Proc. 18th Annual Computer Security Applications Conference, pages 78-87, Las Vegas, NV USA, December 2002.

[RFC2367] McDonald, D., Metz, C., and B. Phan, "PF_KEY Key Management API, Version 2", RFC 2367, July 1998.

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

[RFC2409] Harkins, D. and D. Carrel, "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2461] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[RFC2462] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC2960] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and V. Paxson, "Stream Control Transmission Protocol", RFC 2960, October 2000.

[RFC3303] Srisuresh, P., Kuthan, J., Rosenberg, J., Molitor, A., and A. Rayhan, "Middlebox communication architecture and framework", RFC 3303, August 2002.

[RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[RFC3554] Bellovin, S., Ioannidis, J., Keromytis, A., and R. Stewart, "On the Use of Stream Control Transmission Protocol (SCTP) with IPsec", RFC 3554, July 2003.

[RFC3753] Manner, J. and M. Kojo, "Mobility Related Terminology", RFC 3753, June 2004.

[RFC3775] Johnson, D., Perkins, C., and J. Arkko, "Mobility Support in IPv6", RFC 3775, June 2004.

[RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[RFC4225] Nikander, P., Arkko, J., Aura, T., Montenegro, G., and E. Nordmark, "Mobile IP Version 6 Route Optimization Security Design Background", RFC 4225, December 2005.

[RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD) for IPv6", RFC 4429, April 2006.

[RFC4555] Eronen, P., "IKEv2 Mobility and Multihoming Protocol (MOBIKE)", RFC 4555, June 2006.

[WIP-Ark06] Arkko, J. and I. Beijnum, "Failure Detection and Locator Pair Exploration Protocol for IPv6 Multihoming", Work in Progress, June 2006.

[WIP-Cro04] Crocker, D., "Framework for Common Endpoint Locator Pools", Work in Progress, February 2004.

[WIP-Nik06] Nikander, P., "End-Host Mobility and Multihoming with the Host Identity Protocol", Work in Progress, June 2006.

[WIP-Ste06] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and P. Conrad, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", Work in Progress, June 2006.

[WIP-Sti06] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", Work in Progress, June 2006.

Authors’ Addresses

Tero Kivinen
Safenet, Inc.
Fredrikinkatu 47
HELSINKI FI-00100
FI

EMail: kivinen@safenet-inc.com

Hannes Tschofenig
Siemens
Otto-Hahn-Ring 6
Munich, Bavaria 81739
Germany

EMail: Hannes.Tschofenig@siemens.com
URI: http://www.tschofenig.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).