IKEv2 Mobility and Multihoming (mobike)
Internet-Draft
Expires: August 24, 2004

T. Kivinen
Safenet, Inc.

February 24, 2004

Design of the MOBIKE protocol
draft-kivinen-mobike-design-00.txt

Status of this Memo

Copyright Notice

Abstract

This document discusses the potential design decisions in the base MOBIKE (IKEv2 Mobility and Multihoming) protocol.

Table of Contents

1. Introduction

The current IKEv2 and IPsec documents explictly say that the IPsec and IKE SAs created implicitly between the IP-addresses used in the IKEv2 SA. This means that there is only one IP-address pair attached for the IKEv2 SA, and the only one IP-address pair used as a gateway endpoint address for tunnel mode IPsec SAs.

There are scenarios which requires that the IP address might change rapidly. In some cases the problem could be solved by rekeying all the IPsec and IKE SAs after the IP-address has changed. In some scenarios this might be problematic, as the device might be too slow to rekey the SAs that often, and other scenarios the rekeying and required IKEv2 authentication might require user interaction (SecurID cards etc). Because of those reason the way to update the IP-addresses tied to the IPsec and IKEv2 SAs is required.

MOBIKE protocol provides solution to the problem of the updating the IP-addresses. The MOBIKE protocol takes care following:

- Notifying the other end of IP-address list changes
- Update the IKE SA endpoint addresses based on the notifications
- Automatically switching to use new IP-address if old one does not work anymore
- Updating the tunnel mode IPsec SA tunnel endpoint addresses
- Return routability checks of new addresses if needed

The MOBIKE protocol can be used in different scenarios. Two such scenarios are discussed below.

1.1 Roaming Laptop Scenario

In the roaming laptop scenario the device that moves around is laptop, which might have several ways to connect to internet. It might for example have fixed ethernet, WLAN and GPRS access to net, and some of those can be used in different times. It tries to use the most efficent connection it has all the time, but that connection might change when user for example disconnects himself from the fixed ethernet and uses the office WLAN, and then later leaves the office and starts using GPRS during the trip to home. In home he might again use WLAN (but with different IP-addresses) etc.

The device does not use Mobile IP or anything similar, it simply wants to keep the VPN connection to the corporate security gateway

(SGW) up and running all the time. Even if the interface or the IP-addresses change the internal addresses used inside the IPsec tunnel remains same (allocated from the SGW), i.e. the applications will not detect the changes at all.

1.2 Multihoming SGW Scenario

Another possible scenario which might use MOBIKE is the SGW of the other end of the roaming laptop scenario. I.e. the SGW might have multiple interfaces to different ISPs, and wants to provide connection even when some of those connections are broken. The SGW will know beforehand what set of IP-addresses it will use, but it might need to dynamically update the clients to tell them which addresses to use. It might also use this to do some sort of load balancing, i.e. giving different clients different preferred address, to utilize all the connections. This kind of load balancing is completely internal to the SGW (i.e. the clients will simply see that the preferred IP-address to be used for tunnel endpoint changes, but they do not know why or how the SGW decided to do that), and the actual algorithms how to do that is outside the scope of MOBIKE protocol (i.e. MOBIKE does not disallow the SGW to give different sets of IP-addresses in different preference order to different clients).

Note, that the load-balancing inside the one IKE SA (i.e. one client) is not handled in the MOBIKE protocol. Each client uses only one of the IP-addresses given by the SGW at one time.

2. Major Issues

The base protocol needs to be doing following things:

- Ability to inform the peer about the current or changed address set of the sender
- Ability to infor the peer about the preferred address
- Ability to detect an outage situation and fall back to the use of another address
- Ability to prevent flooding attacks based on announcing someone else's address
- Ability to affect both the IKE and IPsec SAs

2.1 Adopting a new address / multihoming support

From the MOBIKE's point of view the multihoming support is the set of rules how and when to change to use new IP-address from the list. The other end provides a list of addresses which can be used as a destination address, and the local end needs to decide which of them to use. The MOBIKE does not include load-balancing, i.e. the local end only uses one IP-address at time, and it only changes to use new IP-address after some indication from the other end.

That indication might be direct, i.e. the other end sending address update notification, which have different preferred address than which was used before. The local end should try to use the preferred address specified by the other end.

The indication might also be indirect, i.e. the local end notices that suddenly the other ends start using different source address for the packets than what it used before.

Another type of indirect information might that there has been no traffic from the other end for some time (i.e. the current connection might be broken).

This indirect information should not directly cause any changes to the IP-addresses, but they should be used as indication that there might be need to do dead-peer-detection for the currently used address. I.e. when the local end detects that the other end started to use different source IP-address than which was used before, it should initiate dead-peer-detection for the preferred address from the other ends IP-address list (i.e. to the address which it is now

using). If that dead-peer-detection tells that the connection is alive, then there is no need to do anything. If local end does not receive any reply to the dead-peer-detection, then it should do dead-peer-detection for the other addresses in the list (in the preferred order). If it can find an address which works, it will switch to that.

The IKEv2 dead-peer-detection is done by sending empty informational exchange packet to the other end, in which case the other end will acknowledge that. If no acknowledge is received after certain timeout (and after couple of retransmissions), the local end should try other IP-addresses. The packets to other IP-addresses must use the same message-id as the original dead-peer-detection (i.e. they are simply retransmissions of the dead-peer-detection packet using different destination IP-address).

If the local end does not receive acknowledge message back from any of the IP-addresses, it should mark the IKE SA dead, and delete it (as mandated by the IKEv2 specification).

The dead-peer-detection for the other IP-addresses can also be done simultaneously, meaning that after the initial timeout of the preferred address expires, we send packets simultaneously to all other IP-addresses. The problem here is that we need to distinguish from the acknowledge packets which IP-address actually works now (i.e. we will check the acknowledge packets source IP-address, as it should match the destination IP we sent out).

Also the other end is most likely going to reply only to the first packet it receives, and that first packet might not be the most preferred IP-address. The reason the other end is only responding to the first packet it receives is that implementatins should not send retransmissions if they have just sent out identical retranmissions. This is to protect the packet multiplication problem, which can happen if some node in the network queues up packets and then send them to the destination. If destination will reply to all of them then the other end will again see multiple packets, and will reply to all of them etc.

2.2 Message representation

One of the basic design choices that is needed for the MOBIKE is the format of the messages. The IKEv2 offeres some formatting alternatives for the protocol. The basic IP-address change notifications can be sent either via informational exchange already specified in the IKEv2, or we could also have our own MOBIKE specific exchange. Using informational exchange has the main advantage that it is already specified in the IKEv2 and the implementations should

already have code for those.

One advantage of creation of the new exchange would be that we could incorporate the return routability checks to the exchange in this state (i.e. create 3-4 packet exchange). The problem here is that we might need to do the return routability checks for each IP-address separately, thus we might not be able to do it in this phase.

Another choice which needs to be done, is the basic format of the address update notifications. The address update notifications include multiple addresses, which some can be IPv4 and some IPv6 addresses. The number of addresses is most likely going to be quite small (less than 10). The format needs to give out senders preference of the use of the addresses, i.e. the sender will tell this is the preferred address to be used. The format could either contain the preference number, giving out the relative order of the addresses, or it could simply be ordered list of IP-addresses in the order of the most preferred first. In the authors opinion, the last option appears to be the best one. This is because then we do not need to define what happens if the preference numbers are identical, and we do not need to reserve space for the numbers. We do not need any priority values, we simply need ordered list.

Even when the load-balancing inside the one connection is outside the scope of the MOBIKE, there might be future work to include that. The format selected needs to be flexible enough to allow addition of some kind of extra information for the load-balancing features in the future. This might be something like one reserved field, which can later be used to store that information.

There are two basic formats for putting IP-address list in to the exchange, we can include each IP-address as separate payload (where the payload order indicates the preference value, or the payload itself might include the preference value), or we can put the IP-address list as one payload to the exchange, and that one payload will then have internal format which includes the list of IP-addresses.

Having multiple payloads each having one IP-address makes the protocol propably easier to parse, as we can already use the normal IKEv2 payload parsing procedures to get the list out. It also offers easy way for the extensions, as the payload propably contains only the type of the IP-address (or the type is encoded to the payload type), and the IP-address itself, and as each payload already has length associated to it, we can detect if there is any extra data after the IP-address. Some implementations might have problems parsing too long list of IKEv2 payloads, but if the sender sends them in the most preferred first, the other end can simply only take n

first addresses and use them. It might loose connection in some cases if all the n first address are not in use anymore, and the other end hasn't sent new list, but in most cases everything will still work.

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

The next choice is which type of payloads to use. IKEv2 already specifies a notify payload, which could be used for that. It includes some extra fields (SPI size, SPI, protocol etc), which gives 4 bytes of the extra overhead, but then there is the notify data field, which could include the MOBIKE specific data.

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

The basic protocol is most likely going to be something where we send list of all IP-addresses every time the list changes (either addresses are added, removed, or the preferred order changes). Another option is that we send some kind of incremental updates to the IP-address list. Sending incremental updates provides more compact packets (meaning we can support more IP-addresses), but on the other hand have more problems in the syncronization and packet reordering cases i.e. the 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 message id which is incremented for each packet, thus we know the sending order easily).

The address update notification protocol is not restricted to only one way, i.e. both ends might have multiple IP-addresses and both ends might send address updates. Example of that is when the roaming laptop connects to the multihoming SGW.

2.3 Scope of SA changes

When the IKE SA address set changes, do we automatically change all the IPsec SAs negotiated with the IKE SA, or do separately request a change in each IPsec SA separately.

If we want to update each IPsec SA separately, we propably need more efficient format than notification payload, as it can only store one SPI per payload. I.e. we want separate payload which have list of IPsec SA SPIs and new address set for them. If we have lots of IPsec SAs, those payloads can be quite large unless we support ranges in SPIs or at least have some kind of notation of move those SAs not moved separately (i.e. rest of the SA indication). We also have some

problems that we need to keep state per IPsec SA which IP-addresses are used for that SA. If we automatically move all IPsec SAs when the IKE SA moves, then we only need to keep track which IKE SA was used to create the IPsec SA, and fetch the IP-addresses from that (Note, that IKEv2 [I-D.ietf-ipsec-ikev2] already requires implemenations to keep track which IPsec SAs are created using which IKE SA).

If we do allow each IPsec SAs address sets to be updated separately, then we can support scenarios, where the machine have fast and/or cheap connection and slow and/or expensive connection, and it wants to allow moving some of the SAs to the slower and/or more expensive connection, and forbid some SAs to move. I.e. never move 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, then we need to create separate IKE SAs for this scenario, i.e. we create one IKE SA which have both links as endpoints, and it is used for important traffic, and then we create another IKE SA which have only the fast and/or cheap connection, which is then used for that kind of bulk traffic.

3. Miscallaneous issues

There are also some smaller issues, which needs to be decided in the MOBIKE protocol. Those issues include whether we allow disconnection notifications, do we need to do return routablity checks always, and what shall we do with simultaneous movement.

3.1 Zero Address Set

One of the features which might be usefull would be for the peer to announce the other end 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 the peer could send address notification with zero new addressess, which means that it will not have any valid addresses anymore. The responder of that kind of notification would then acknoledge that, and could then temporarely disable all SAs. If any of the SAs gets any packets they are simply dropped. This could also include some kind of ACK faking to keep the TCP/IP sessions alive, or it could simply be left to the applications, i.e. allow TCP/IP sessions to notice the link is broken.

The local policy could then decide how long the peer would allow other peers to be disconnected, i.e. whether this is only allowed for few minutes, or do they allow users to disconnect Friday evening and reconnect Monday morning (consuming resources during that, but on the other hand not more than is normally used during week days).

3.2 When to do Return Routability Checks

One of the decisions that needs to be done, when to do return routability checks. The simple approach is to do it always. Another option is to do it every time new IP-address is taken in to use. The basic format of the return routability check could be similar than dead-peer-detection, but the problem is that if that fails then the IKEv2 specification requires the IKE SA to be deleted. Because of this we might need to do some kind of other exchange.

If the other end is SGW with limited set of fixed IP-addresses, then the SGW can get certificate having all the IP-addresses in the certificate. If the certificate includes all the IP-addresses, it is no point to do weaker return routability check, the data in the certificate is already properly authenticated after the IKE SA is created, so the peer might simply use that and ignore return routability checks.

Another option is to use draft-dupont-mipv6-3bombing [I.D.dupont-mipv6-3bombing] approach: do it only if you had to send

the update from some other address than indicated preferred address.

Final option would simply not to do return routability checks at all. If we use indirect change notifications then we only move to the new IP address after successfull dead-peer-detection on the new address, which is already return routability check. In the direct change notifications the authenticated peer have given out authenticated IP-address, thus we could simply trust the other end. There is no way external attacker can cause any attacks, but we are not protected by the internal attacker, i.e. the authenticatede peer forwarding its traffic to the new address. On the other hand we do know the identity of the peer in that case.

3.3 Simultaneous Movements

As we are not creating full mobility solution, but are instead concentrating on the VPN style scenarios, we do not need to solve the simultaneous movement recovery problem. We assume that the one end (SGW) will have fixed set of addresses (from which some subset might be in use), thus it cannot move to the address not known by the other end. This means that the solutions how to recover from cases where both ends move and the movement notifications do not reach other ends, is outside the scope of the MOBIKE WG.

4. Security Considerations

As all the messages are already authenticated by the IKEv2 there is no problem that any attackers would modify the actual contents of the packets. The IP addresses in 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, they should not cause any direct actions.

One type of attacks which needs to be taken care of the MOBIKE protocol is also various flooding attacks. See [I-D.nikander-mobileip-v6-ro-sec] and [Aur02] for more information about flooding attacks.

5. IANA Considerations

No IANA assignments are needed.

Normative references

[RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768, August 1980.

[I-D.ietf-ipsec-ikev2] Kaufman, C., "Internet Key Exchange (IKEv2) Protocol", draft-ietf-ipsec-ikev2-12 (work in progress), January 2004.

[Kiv04] Kivinen, T., "MOBIKE protocol", draft-kivinen-mobike-protocol-00 (work in progress), February 2004.

Non-normative references

[I-D.nikander-mobileip-v6-ro-sec]
Nikander, P., "Mobile IP version 6 Route Optimization Security Design Background", draft-nikander-mobileip-v6-ro-sec-02 (work in progress), December 2003.

[I-D.dupont-mipv6-3bombing]
Dupont, F., "A note about 3rd party bombing in Mobile IPv6", draft-dupont-mipv6-3bombing-00 (work in progress), February 2004.

[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.

Author's Address

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

EMail: kivinen@safenet-inc.com

Intellectual Property Statement

Full Copyright Statement

Acknowledgment

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