Negotiated Address Reuse (NAR)
draft-montenegro-aatn-nar-00.txt

Status of This Memo

This document is an individual submission to the Internet Engineering Task Force (IETF). Comments should be submitted to the author, or to the Alternative to Address Translation in Networks (AATN) mailing list at aatn@alpha.zk3.dec.com.

Distribution of this memo is unlimited.

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

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

Abstract

Network address (and port) translation is a useful technology. However, several shortcomings have been identified (Mobile IP, IPSEC and QOS flows). In these cases, NAT may be complemented by the use of NAR (Negotiated Address Reuse). The negotiation phase in NAR is based on SOCKS.
Table of Contents

1. Introduction ................................................... 3
   1.1. Terminology ............................................ 3
   1.2. Objectives and Assumptions ............................ 3
   1.3. Applicability and Benefits ............................ 4
   1.4. Assumptions ........................................... 4
2. Overview of the Solution ..................................... 5
   2.1 Model .................................................. 5
   2.2 Negotiation ............................................. 6
   2.3 End-to-end NAR Mappings ................................ 7
   2.4 Translated NAR Mappings ................................ 7
   2.5 Packet Delivery between NAR Client and Server ........ 8
   2.6 Protocol Handling and Demultiplexing at the NAR server .. 8
     2.6.1 TCP, UDP and ICMP Handling and Demultiplexing ...... 9
     2.6.2 IPSEC Handling and Demultiplexing .................. 10
     2.6.3 Mobile IP and TEP Handling and Demultiplexing ...... 11
3. SOCKS-based Negotiation ..................................... 14
   3.1 NAR-MODE Command ..................................... 15
   3.2 ICMP Negotiation ....................................... 16
   3.3 IPSEC Negotiation ..................................... 17
   3.4 Tunneling Negotiation ................................... 18
4. Security Considerations ..................................... 19
5. Acknowledgements ........................................... 19
References .................................................. 19
Author address ............................................... 21
1. Introduction

Network Address (and port) translation [NAPT] (henceforth called "NAT" in this document) is a very useful technology. Nevertheless, it is raising concerns [IABNAT] not only because of its architectural impurity, but also due to its limitations. This document proposes Negotiated Address Reuse (NAR) as a complement to NAT. The negotiation phase is based on the SOCKS protocol [SOCKS]. However, once this is over and data transfer begins NAR server and client behave considerably different from a SOCKS server and client.

The idea is to use NAT by default due to its transparency, and resort to NAR when necessary (or dictated by policy). At the AATN BOF held at the 41st IETF meeting (Los Angeles, 4/1/98), the applications that require a NAT alternative were initially identified as end-to-end:

- IPSEC
- Mobile IP
- QOS flows

This document specifies how NAR enables IPSEC and Mobile IP. QOS will be addressed in a future revision.

1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [9].

1.2. Objectives and Assumptions

Given that NAT does solve a large number of problems, it makes sense to continue using it. Instead of replacing the NAT box, its capabilities are augmented by NAR. This is, by far, the most practical alternative. Nevertheless, NAR does not require NAT.

A primary objective of NAR is to enable SOHO (small office home office) applications, in which the NAT/NAR box only has one public IP address (perhaps assigned dynamically by its ISP). This single public IP address hides all other systems within the private SOHO net. This is perhaps the single most common application scenario for NAT.

Finally, legacy systems outside the "local" or "private" network
MUST be supported. In the context of the model introduced in Section 2.1, this means that no changes are visible outside address space A.

1.3. Applicability and Benefits

Since TCP, UDP and ICMP are satisfactorily handled by NAT, there does not seem to be a compelling reason to use NAR for them. Nevertheless, even in this case there still are some benefits:

- NAR obviates the need to explicitly handle protocols like FTP or ICMP that embed address and port information in its payload [NAPT].
- NAR derives the benefits of using SOCKS: authenticating the user before granting service, and fine-grained authorization.

1.4. Assumptions

NAR can operate under the following assumption:

Mutual Non-Routability Assumption:

Both address spaces are mutually non-routable (see section 2.1, "Model"), that is, the routing fabric in one address space does not recognize or handle the address ranges used in the other.

On the other hand, NAT cannot operate under this assumption. Instead, it requires the:

Partial Routability Assumption:

One of the address spaces is routable within the other. In the examples that use the model below (section 2.1), address space B is routable within address space A, but the inverse is not true. Default routing is a straightforward way to guarantee this.

In some of the example scenarios below NAR is used to complement NAT. In these cases, the latter assumption is implicitly imposed.

NAT works by establishing mappings between tuples in both address spaces. These mappings may be established [NAPT]:

- statically (at startup time by virtue of manual configuration),
or
- dynamically (at protocol session initiation time).

Static mappings are difficult to maintain and prone to operator error. Only dynamic mappings offer the transparency benefits of NAT, but they require that the NAT box be able to establish the mapping that applies in the reverse direction (into the NAT box), by examining the packet flow in the forward direction (out of the NAT box).

Thus, NAT only works for network protocols that satisfy the:

Inferrable Mapping Assumption

A sufficient and correct mapping for inbound traffic is inferred by examining outbound traffic.

There are two reasons why this assumption may not apply:

- The protocol does not include the necessary information in outgoing packets (that is, outgoing packets never initiate sessions). This may be true because another protocol is used to establish sessions. For example, the mapping for incoming IPSEC packets (incoming SPI) is not evident from outgoing IPSEC packets (outgoing SPI). This session indicator is negotiated out-of-band, by using manual keying or IKE [IKE]. Hence, the only mechanism guaranteed to work is to use NAR to explicitly establish a negotiated mapping.

- No outgoing traffic is guaranteed. This may happen if the incoming traffic is destined for a server. Typically, this has been accommodated in NAT by static configuration. NAR provides a negotiated mechanism to programatically establish these "server" mappings.

Because NAR adds negotiated mappings, it does not require the Inferrable Mapping Assumption.

2. Overview of the Solution

2.1 Model

The model we will use is [NATMODEL]:
Hosts X and Y belong to different address spaces A and B, respectively, and N is a NAT/NAR box. N has two addresses: Na on address space A, and Nb on address space B. In [BYPASS], the assumption is that N has several addresses in address space B, different from Nb, that it can dynamically assign to or lend to X. These addresses are not shown above, but they can be denoted as Nb1, Nb2, Nb3, and so on. NAR allows that model, but also allows X (and any other hosts on address space A) to reuse Nb.

This is applicable to the very common SOHO scenario in which address space B is the internet, address space A is a private SOHO network, and N has exactly one address per address space: Na and Nb. Presumably, Nb is the single address assigned to it (perhaps dynamically) by the ISP through which N connects to the internet.

From the point of view of NAR, N is a NAR server and X is a NAR client.

2.2 Negotiation

Hosts X and Y wish to exchange traffic. Furthermore, the protocol they are exchanging does not satisfy the "Inferrable Mapping Assumption" from section 1.4.

As specified in section 3 ("SOCKS-based Negotiation"), X uses a derivative of the SOCKS [SOCKS] protocol with N to obtain an IP address on address space B. As a result of the negotiation, N lends X one of its available addresses. Frequently, this will be the same address that N uses on address space B, namely, Nb. In what follows, this is the case.

The NAR client also negotiates the values of protocol-dependent fields which will enable the NAR server, N, to demultiplex the incoming traffic (section 2.6). At the end of the negotiation phase, N establishes a negotiated mapping so incoming replies will be processed correctly.

Each mapping operates in either of two modes: (1) End-to-end mode (the default), or (2) Translated mode. These are explained in the subsequent sections.
2.3 End-to-end NAR Mappings

If we impose the "Mutual Non-Routability Assumption" from section 1.4, then the sequence continues as follows:

From that point on, X assumes address Nb as its own, and uses it as the source IP address of its outgoing packets.

X then delivers those packets to N by using a tunneling mechanism as specified in section 4 ("Delivery between NAR server and NAR client").

Packets flowing from Y to X MUST include the negotiated values so N can demultiplex them properly, and deliver them to X. As will be seen below, this does not imply any NAR support in Y. This constitutes a reuse of address Nb, because in addition to N itself, all its NAR clients are also using Nb as their address. Nb is the source IP address of packets created by N (naturally) as well as by its NAR clients (for example, X). Packets sent by Y and other correspondent hosts are sent to address Nb.

Other than the fact that X is aware of and actively uses Nb as its address when talking to Y, this is very similar to NAT [NAPT]. The difference in N's role is it simply shuttles packets between X and Y. Since there is no translation involved, packets prepared by X arrive unmodified at Y (except for fields that change during the course of normal routing operations, as specified in Section 3.3.3.1.1.1 of [Kent98b]), and viceversa.

In this mode of operation, the NAR client in effect establishes a virtual presence at the NAR server. Datagrams are exchanged between the peer Y and the X’s virtual presence within NAR server N.

The above mode of operation is "End-to-end NAR."

2.4 Translated NAR Mappings

It is also possible to mix NAR with the benefits of translation. For example, if we impose the "Partial Routability Assumption" from section 1.4, the sequence might proceed as follows:

X continues using address Xa as its own, and uses it as the source IP address of its outgoing packets.

X then delivers those packets to the routing infrastructure in address space A. Eventually, N receives the packets from X to Y, and processes them according to NAT rules (this may imply new
extensions for NAT): by changing the source IP address of the packets from Xa to Nb before injecting them out into address space B on their way to Y. In Y’s replies the destination IP address is set to Nb.

In this case, NAR is only used to establish the mapping, after which NAT (perhaps extended to handle the specific protocol being exchanged) takes over.

The translation undergone by the packet is, of course, protocol dependent and if defined in [NAPT], follows those directives exactly (for example: UDP, TCP, ICMP, FTP, and so on). Translations for other protocols like IPSEC, and tunneling are defined below.

The above mode of operation is "Translated NAR."

2.5 Packet Delivery between NAR Client and Server

In Translated NAR, packets between X and N flow as in the regular NAT case, that is, direct delivery without recourse to tunneling MUST be used. This is the default mode unless explicitly overriden by End-to-end NAR during the negotiation phase.

When operating in the End-to-end mode, the NAR client and server MUST tunnel packets to each other. If tunneling, IP in IP [IPIP] MUST be used, unless GRE [GRE] is explicitly agreed upon by NAR client and server during the negotiation phase.

2.6 Protocol Handling and Demultiplexing at the NAR server

Upon receiving a packet addressed to Nb, the NAR server N must determine if it is destined for one of its own processes, or for a process on one of its NAR clients. This determination depends on the protocol, and, in general, uses a "tuple" or set of fields in the packet. By establishing a mapping between a tuple and the corresponding NAR client, the NAR server unequivocally identifies the destination of matching packets.

Given that the NAR clients may reuse the NAR server’s single IP address on address space B, namely, Nb, this is the value that will appear in the destination IP address field of incoming packets from Y. This means that the destination IP address cannot be relied on to identify the destination of a packet. Instead, the NAR server uses a field within the IP payload for demultiplexing purposes. The destination IP address is still checked to make sure that it matches what was negotiated (frequently, Nb). The SOCKS-based negotiation
phase produces the required and unique tuples. They are mapped to
the client in the SOCKS-based negotiation.

The sections below explains processing at the NAR box for the
following protocols:

- TCP or UDP
- ICMP
- IPSEC
- Layer 3 tunneling (Mobile IP, TEP)

The explanations below assume that the NAR server N is examining a
packet sent by Y, destined for X.

2.6.1 TCP, UDP and ICMP Handling and Demultiplexing

TCP, UDP and ICMP demultiplexing is done based on the same tuples as
NAT \[NAPT\].

When N receives a TCP packet, it demultiplexes it based on the
following tuple:

- source IP address
- source TCP port
- destination TCP port
- destination IP address

The NAR server looks for a match in its mappings table and tunnels
the packet to the corresponding NAR client.

UDP packets are demultiplexed based on this tuple:

- source UDP port
- destination UDP port
- destination IP address

The source IP address in an incoming UDP reply is ignored because in
a multihomed host it may not match the destination IP address of the
original request (RFC 1123).
ICMP is demultiplexed based on the following tuple:

- source IP address
- ICMP identifier
- destination IP address

If operating on an End-to-end NAR mapping, the NAR server need not modify further the encapsulated IP packet within the ICMP payload as required by NAT [NAPT].

Even though NAR does support TCP, UDP and ICMP, it may be simpler to allow NAT or Translated NAR to handle these cases. Translated NAR is useful, for example, to negotiate programatically a mapping for a server process. Once that is done, the actual translation (if in Translated NAR mode) is exactly as specified in [NAPT].

### 2.6.2 IPSEC Handling and Demultiplexing

IPSEC packets (protocols 51 and 50 for AH and ESP, respectively) [Kent98a, Kent98b] from Y destined for X arrive at NAR server N. They are demultiplexed based on the following tuple:

- protocol (50 or 51)
- SPI
- destination IP address

During negotiation, the NAR client X and server N arrive at an SPI value to denote the incoming security association from Y to X. Once N and X make sure that the SPI is unique within both of their SPI spaces, X communicates its value to Y as part of the IPSEC security association establishment process, namely, Quick Mode in IKE [IKE] or manual assignment.

Given that Y sends packets to address Nb using the negotiated SPI, N is able to find a matching mapping and either:

- tunnel the packet to X (at address Xa), or,
- overwrite the destination IP address on the packet with Xa instead of Nb.

N MUST deliver packets to X by tunneling if:
- operating under the "Mutually Non-Routable Assumption". Not doing so may prevent successful delivery of the packet within address space A (perhaps because of ingress filtering within the routing fabric of address space A), given that the source and destination IP addresses belong to address space B (Yb and Nb, respectively).

- the packet being forwarded includes an AH header [Kent98b] (protocol 51) immediately following the IP header. Since AH renders the address fields in the preceding IP header immutable, NAT is out of the question, and tunneling is the only alternative.

Given that NAR establishes the appropriate mapping with the expected SPI, translation (by NAT) of the destination IP address on incoming packets from Y is possible, but only if the packet includes an ESP header [Kent98a] (protocol 50) immediately following the IP header. In this case, traditional NAT has been said to "break" IPSEC, not because the IP address fields are immutable and therefore preclude NAT, but because there has been no way to establish the required mapping. In other words, the ESP protocol does not satisfy the "Inferrable Mapping Assumption."

One problem still remains: how does Y know that it is supposed to send packets to X via Nb? Y is not NAR-aware, but it is definitely ISAKMP-aware. Y sees ISAKMP packets coming from address Nb, regardless of whether X is operating in End-to-end or Translated NAR mode. To prevent Y from deriving the identity of its ISAKMP peer based on the source address of the packets (Nb), X MUST exchange client identifiers with Y during Quick Mode (Phase 2). ISAKMP traffic is simply UDP (port 500) so it is certainly handled correctly by NAT. The proper use of identifiers (IDii and IDir) during Phase 1 allow the clear separation between those identities and the source IP address of the packets.

2.6.3 Mobile IP and TEP Handling and Demultiplexing

Suppose NAR client X sets up a layer 3 tunnel with host Y in address space B, perhaps because X and Y implement Mobile IP [MIP] or TEP [TEP]. X could be a foreign agent, or a co-located mobile node, and Y is the home agent. Suppose Xb is the mobile node’s home address.

Similar to IPSEC, Mobile IP and TEP has two distinct phases:

- tunnel setup via a UDP-based protocol (registration protocol), and
data transfer via tunneled packets.

Let’s examine first the data transfer phase.

Say the incoming packet looks like this:

```
INCOMING IP IN IP TUNNELED PACKET FROM Y TO N
+-----------------+
|        +-------+|
| Yb->Nb | CN->Xb||
|        +-------+|
+--------+--------+
```

Here, the original packet is sent by a correspondent node at IP address CN to the mobile node at address Xb. The packet was intercepted by the home agent at Yb and sent towards the mobile node via address Nb.

Once the packet reaches N (via address Nb), the NAR server must identify which of its NAR clients is the ultimate destination for the internal packet. In order to do so, it needs a tuple guaranteed to be unique among all of its NAR clients.

The unique tuple sufficient for demultiplexing IP in IP packets [IPIP] (protocol 4) is:

- destination IP address of the encapsulated (internal) header

  This is mobile node X’s home address (Xb in the above example). In tunneling applications other than Mobile IP, it is simply another address for X, significant within address space B. At first glance, it seems like this is unique among all NAR clients. However, Xb could be a private address in use within address space B. Another mobile node roaming from another address space, say, address space C could also be using the same address. So Xb by itself is not unique, but the combination with the remote tunnel endpoint is (see next item).

- source IP address of the external header

  This, the remote end of the tunnel, is Yb in the above example. Its combination with the previous item is guaranteed to be unique among all of N’s NAR clients.

- destination IP address of the external header
This, the local end of the tunnel, is Nb in the above example, and, given that it is frequently used by N and all of its NAR client, is usually not unique.

Once N identifies the ultimate destination of the packet, Xa, it must deliver the internal packet there. From section 2.5, there are two means to do so: tunneling (if using End-to-end NAR) and direct delivery as in regular NAT (if using Translated NAR). As end-to-end mode adds yet another IP header, whenever possible, Translated mode is preferred. In this case, the packet that arrives at Xa is:

DELIVERING IP IN IP PACKET FROM N TO X IN TRANSLATED MODE

+-----------------+
|                  |
| Na->Xa | CN->Xb |
|                  |
+-----------------+

If using End-to-end mode, the packet that arrives at Xa is:

DELIVERING IP IN IP PACKET FROM N TO X IN END-TO-END MODE

+---------------------------+
|                            |
|                           |
|                             |     +-------+     |
| Na->Xa | Yb->Nb | CN->Xb |     +-------+     |
|                             |     +--------+     |
+---------------------------+

GRE packets [Hanks94] (protocol 47) without a Key field are only handled if their Protocol Type field has a value of 0x800 (other values are outside the scope of this document), and are demultiplexed based on the same tuple as IP in IP packets. In GRE terminology, the tuple is:

- destination IP address of the payload (internal) packet
- source IP address of the delivery (external) packet
- destination IP address of the delivery (external) packet

GRE packets with a Key field could be demultiplexed based on the tunnel identifier [TEP], but it is simpler to just use the above tuple in all GRE cases.
GRE packets with a Routing field are outside the scope of this document.

Keeping in mind that Y cannot address X directly, how does the latter guarantee that the former send tunneled packets to it via N? It must do without requiring any new functionality at node Y (that is, Y is unaware of NAR or NAT processing at N). Accordingly, this tunnel must be configured using the Registration Protocol as defined in [MIP].

Given that Mobile IP registration protocol is UDP (port 434) traffic, it, in fact, works over NAT. If mobile node X is registering with home agent Y, it must use Nb as the care-of address of home address Xb. This guarantees that Y sends packets to X via Nb, after which the negotiated mapping takes over such that N delivers packets to Xa, as outlined above.

It is possible to shield the mobile node X from any knowledge of NAR by integrating Mobile IP into the NAR server. All the NAR server needs is to be able to create the mapping between the above tuple and the mobile node. This information is available from the Registration protocol itself, so an explicit SOCKS-based negotiation phase is not required. For this to work, the mobile node must learn the care-of address it must use, namely, Nb from the agent advertisements instead of relying on the SOCKS-based negotiation to produce this value. If the mobile node has acquired a co-located address, the foreign agent issuing agent advertisements (perhaps the NAR server itself) must use the ‘R’ bit to force the mobile node’s Registration Request to go through it.

3. SOCKS-based Negotiation

In the negotiation phase, both NAR server and NAR client must agree on values used in different protocols fields. Example values are:

- IPv4 addresses used as tunnel endpoints or as externally visible addresses
- ports for TCP or UDP
- SPI values for IPSEC

The negotiation used by NAR is based on SOCKS [SOCKS] messages, and, where possible, reuses the exact same formats. However, the behavior of the entities involved, namely, the NAR client and NAR server, is very different from that between a SOCKS client and server: instead of setting up an application layer gateway, the objective is to
establish a mapping that (1) enables NAT (Translated NAR), or (2) allows the client to acquire a virtual presence at the server (End-to-end NAR).

The sequence of the SOCKS negotiation is altered slightly in order to allow for NAR mode to be set after the authentication phase (Section 3 of [SOCKS]), and before the SOCKS commands are issued (Section 4 of [SOCKS]). In addition to SOCKS commands (formatted according to Section 4 of [SOCKS], this document defines NAR commands. However, these MUST NOT be exchanged before the negotiation enters NAR mode, as specified in section 3.1, below.

Finally, the objective of NAR negotiations is not to establish a relay for outgoing packets (as these simply rely on usual routing in order to be delivered to the end system), but for incoming packets from the "outside" end system to the NAR client.

Negotiations for UDP and TCP are defined in [SOCKS].

3.1 NAR-MODE Command

NAR-MODE is a new SOCKS command, with an identifier of TBD, used by the NAR client to advise the server that all subsequent exchanges are to be interpreted as a NAR, not a SOCKS negotiation. NAR-MODE is formatted according to Section 4 of [SOCKS]. Servers which do not support the NAR-MODE command should respond with the "Command not supported" error.

The client MUST set DST.PORT to a port number of 0. and DST.ADDR to the address of the target address (Yb, in the examples above).

The NAR-MODE command implies a persistent connection, because the client is requesting the server to treat all subsequent negotiations as NAR, not SOCKS negotiations. Accordingly, the server MUST hold the connection after the NAR-MODE command is completed to perform another SOCKS command. Unless otherwise specified, command syntax follows SOCKS. The semantics, however, are dictated by NAR.

The reply to the command is also formatted as a standard reply [SOCKS, sec 6.]. The address returned in BND.ADDR MUST be the IPv4 address offered by the NAR server for reuse by the NAR client (Nb, in the examples above).

Flag bits in the request and reply are defined as follows:

- TUNNELED-DELIVERY X’01’
- GRE X’02’
NAR mode uses the direct delivery method by default (Section 2.5). If the client wishes to use the tunneled delivery method (End-to-end NAR) it sets the TUNNELED-DELIVERY in the FLAG field of the request. Furthermore, the client may petition the use of GRE by setting that bit in the FLAG field of the request.

The server grants the above requests by setting the corresponding bits in the FLAG field of its reply to the client. The server MAY also impose any of the options above by setting the corresponding bit(s) in the FLAG field of its reply, regardless of what the client requested. The client MUST conform to what the server requests. The server MAY ignore non-conforming clients.

3.2 ICMP Negotiation

The objective of the ICMP negotiation is to establish a mapping between the ICMP tuple in section 2.6.1 and the IPv4 address of the NAR client. The NAR server derives the latter from the TCP connection in use for the NAR negotiation (on TCP port 1080). It knows the value of the remote IP endpoint from the value of DST.ADDR in the NAR-MODE command, and the value of the local IP endpoint from the value of BND.ADDR in the reply to the NAR-MODE command. All that is needed is a command to negotiate the value of the remaining element of the tuple: ICMP identifier.

The ICMP-ASSOCIATE request is a new NAR command used to establish an association within the NAR relay process to handle ICMP datagrams:

```
+----+-----+-------------+
|VER | CMD |   ICMP.ID   |
+----+-----+-------------+
   1   1     2     |
+-------------------+
```

The fields in the ICMP-ASSOCIATE packet are (with values in network octet order):

- **VER**: protocol version: X’01’
- **CMD**: ICMP-ASSOCIATE TBD
- **ICMP.ID**: Value to be used in the Identifier field of the ICMP packet by the client. It MAY be set by the client in the request, and MUST be confirmed or overridden by the server in the reply.

The reply to the command is also formatted as shown above. The CMD field is used as a reply field in the response, with reply values as
3.3 IPSEC Negotiation

The objective of the IPSEC negotiation is to establish a mapping between the tuple in section 2.6.2 and the IPv4 address of the NAR client. The NAR server derives the latter from the TCP connection in use for the NAR negotiation (on TCP port 1080). It knows the value of the reused IP address (same value as BND.ADDR in the reply to the NAR-MODE command). All that is needed is a command to negotiate the remaining values of the tuple: the incoming SPI and protocol.

The IPSEC-ASSOCIATE request is a new NAR command used to establish an association within the NAR relay process to handle IPSEC datagrams:

```
+----+-----+-------------+--------+
|VER | CMD | IN.PROTOCOL | IN.SPI |
+----+-----+-------------+--------+
| 1  | 1   | 1           | 4      |
```

The fields in the IPSEC-ASSOCIATE packet are (with values in network octet order):

- **VER**: protocol version: X'01'
- **CMD**: IPSEC-ASSOCIATE TBD
- **IN.PROTOCOL**: D'50' (X'32') for ESP, D'51' (X'33') for AH.
- **IN.SPI**: SPI that the remote node uses when sending datagrams to the NAR client (incoming SPI). Typically not set by the client on the request, but by the server on the reply. It must then be communicated to the remote node via IKE or manual key exchange.

The IN.SPI field contains the SPI for incoming IPSEC datagrams. If the client is not in possession of this information at the time of the IPSEC-ASSOCIATE, the client MUST use an SPI number of all zeros.

The reply to the command is also formatted as shown above. The CMD field is used as a reply field in the response, with reply values as specified in [SOCKS, Section 6]. The SPI returned is assigned by the NAR server for the NAR client to use as its own and overrides whatever value the client may have used in the request.
3.4 Tunneling Negotiation

The objective of the TUNNEL negotiation is to establish a mapping between the tuple in section 2.6.3 and the IPv4 address of the NAR client. The NAR server derives the latter from the TCP connection in use for the NAR negotiation (on TCP port 1080). It knows the value of the remote tunnel endpoint from the value of DST.ADDR in the NAR-MODE command, and the value of the local tunnel endpoint from the value of BND.ADDR in the reply to the NAR-MODE command. All that is needed is a command to negotiate the value of the remaining elements of the tuple: tunneling protocol, and the internal tunnel address on the local side.

The TUNNEL-ASSOCIATE request is a new NAR command used to establish an association within the NAR relay process to handle tunneled datagrams:

```
+----+-----+------+-+-----------+-------------+
|VER | CMD | ATYP | LTUN.ADDR| TUN.PROTOCOL|
+----+-----+------+-+-----------+-------------+
| 1  |  1  |  1   | Variable |      1      |
+----+-----+------+-+-------------+
```

The fields in the TUNNEL-ASSOCIATE packet are (with values in network octet order):

- **VER**: protocol version: X’01’
- **CMD**: TUNNEL-ASSOCIATE TBD
- **ATYP**: address type of following address
  - IP V4 address: X’01’
  - DOMAINNAME: X’03’
  - IP V6 address: X’04’
- **LTUN.ADDR**: Internal Address of the tunnel on the local side. In Mobile IP, this corresponds to the value of the NAR client’s home address (Xb in the example above). MUST be set by the client on the request, confirmed by the server on the reply.
- **TUN.PROTOCOL**: D’4’ (X’4’) for IP in IP, D’47’ (X’2F’) for GRE.

The reply to the command is also formatted as shown above. The CMD field is used as a reply field in the response, with reply values as specified in [SOCKS, Section 6].
4. Security Considerations

This document does not add any security issues to those already posed by NAT, or normal routing operations. Routing decisions are based on a tuple with typically one element: destination IP address. This document just adds more elements to the tuple. Furthermore, by allowing an End-to-end mode of operation and by introducing a negotiation phase to address reuse, the mechanisms described here are more secure and less arbitrary than NAT.

A word of caution is in order: SPI values are meant to be random, and thus serve also as cookies to reduce off-the-path denial-of-service attacks. However, NAR support for IPSEC, renders the SPI a negotiated item; in addition to being a unique value at the receiver X, it must also be unique at the NAR box, N. Limiting the range of the SPI values available to the NAR clients reduces their entropy slightly, thus (slightly) weakening their effectiveness as an anti-clogging token.

5. Acknowledgements

Many thanks to Pat Calhoun, Michael Speer and Vipul Gupta for helpful discussion and ideas. The latter’s [NATMODEL] proved invaluable in composing this document.

Some text in section 3 was taken (with slight modifications) from [SOCKS] and draft-ietf-aft-socks-ext-00.

References


V. Gupta, message to the AATN mailing list, Message-Id: <libSDtMail.9802121343.10882.vgupta@nobel>, February 12, 1998.


M. Leech, M. Ganis, Y. Lee, R. Kuris, D. Koblas, L. Montenegro


Author address

Questions about this document may be directed at:

Gabriel E. Montenegro
Sun Microsystems, Inc.
901 San Antonio Road
Mailstop UMPK 15-214
Mountain View, California 94303

Voice: +1-415-786-6288
Fax: +1-415-786-6445

E-Mail: gabriel.montenegro@eng.sun.com