        NEC's Simple Middlebox Configuration (SIMCO) Protocol Version 3.0

Status of This Memo

Copyright Notice

IESG Note

Abstract

   This document describes a protocol for controlling middleboxes such
   as firewalls and network address translators. It is a fully
   compliant implementation of the Middlebox Communications (MIDCOM)
   semantics described in RFC 3989. Compared to earlier experimental
   versions of the SIMCO protocol, this version (3.0) uses binary
   message encodings in order to reduce resource requirements.

Table of Contents

1. Introduction

   The Simple Middlebox Configuration (SIMCO) protocol is used to
   control firewalls and Network Address Translators (NATs). As defined
   in [RFC3234], firewalls and NATs are classified as middleboxes. A
   middlebox is a device on the datagram path between the source and
   destination that performs other functions than just IP routing. As
   outlined in [RFC3303], firewalls and NATs are potential obstacles to
   packet streams, for example, if dynamically negotiated UDP or TCP
   port numbers are used, as in many peer-to-peer communication
   applications.

   SIMCO allows applications to communicate with middleboxes on the
   datagram path in order to request a dynamic configuration at the
   middlebox that enables datagram streams to pass the middlebox.
   Applications can request pinholes at firewalls and address bindings
   at NATs.

   The semantics for the SIMCO protocol are described in [RFC3989].

1.1. Terminology

   The terminology used in this document is fully aligned with the
   terminology defined in [RFC3989]. In the remainder of the text, the
   term SIMCO refers to SIMCO version 3.0. The term "prefix-length" is
   used as described in [RFC4291] and [RFC1519]. With respect to
   wildcarding, the prefix length determines the part of an IP address
   that will be used in address match operations.

1.2. Binary Encodings

   Previous experimental versions of SIMCO used simple ASCII encodings
   with augmented BNF for syntax specification. This encoding requires
   more resources than binary encodings do for generation and parsing of
   messages. This applies to resources for coding agents and
   middleboxes as well as to resources for executing a SIMCO stack.

Low resource requirements are important properties for two main
reasons:

   - For many applications (for example, IP telephony), session setup
     times are critical. Users do accept setup times only up to some
     limit, and some signaling protocols start retransmitting
     messages if setup is not completed within a certain time.

   - Many middleboxes are rather small and relatively low-cost
     devices. For these, support of resource-intensive protocols
     might be a problem. The acceptance of a protocol on these
     devices depends, among other things, on the cost of implementing
     the protocol and of its hardware requirements.

Therefore, we decided to use a simple and efficient binary encoding
for SIMCO version 3.0, which is described in this document.

2. Compliance with MIDCOM Protocol Semantics

SIMCO version 3 is fully compliant with the MIDCOM protocol semantics
defined by [RFC3989]. SIMCO implements protocol transactions as
defined in Section 2.1.1 of [RFC3989]. All message types defined in
Section 2.1.2 of [RFC3989] are supported by SIMCO, and all mandatory
transactions are implemented. SIMCO does not implement the optional
group transactions. For all implemented transactions, SIMCO
implements all parameters concerning the information contained.

SIMCO defines a few new terms to reference functionality in the
semantics. Among these terms are Session Authentication (SA) and
Policy Enable Rule After reservation (PEA) messages. SA is used to
model the state transition given in Figure 2 of [RFC3989] from NOAUTH
to OPEN. PEA is used to model the state transition given in Figure 4
of [RFC3989] from RESERVED to ENABLED.

SIMCO implements one additional transaction, the Policy Disable Rule
(PDR) transaction, to those defined in [RFC3989]. PDR transactions
are used by security functions such as intrusion detection and attack
detection. They allow the agent to block a specified kind of
traffic. PDRs have priority above Policy Enable Rules (PERs). When
a PDR is established, all conflicting PERs (including PERs with just
a partial overlap) are terminated, and no new conflicting PER can be
established before the PDR is terminated. Support of the PDR
transaction by SIMCO is optional. For a detailed description of the
PDR transaction semantics, see Section 8.8.

3. SIMCO Sessions

   The SIMCO protocol uses a session model with two parties: an agent
   representing one or more applications and a middlebox. Both parties
   may participate in multiple sessions. An agent may simultaneously
   communicate with several middleboxes using one session per middlebox.
   A middlebox may simultaneously communicate with several agents using
   one session per agent.

```
              +-------+  SIMCO protocol  +-----------+
              | agent +------------------+ middlebox |
              +-------+                  +-----------+
```
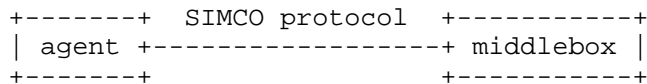
              Figure 1: Participants in a SIMCO session

   SIMCO sessions must run over a reliable transport layer protocol and
   are initiated by the agent. SIMCO implementations must support TCP,
   while other reliable transport protocols can be used as transport for
   SIMCO as well. When using TCP as transport, middleboxes must wait
   for agents to connect on port 7626. This port is assigned to SIMCO
   servers by IANA (see http://www.iana.org/assignments/port-numbers).
   The session may be secured, if required, by either IPsec or TLS
   [RFC4346] to guarantee authentication, message integrity and
   confidentiality. The use of IPsec is outlined in Section 9,
   "Security Considerations".

   The transaction semantics of sessions is explained in [RFC3989]
   Section 2.2.

4. SIMCO Message Components

   All SIMCO messages from agent to middlebox and from middlebox to
   agent are sent over the transport protocol as specified in Section 3.
   SIMCO messages are Type-Length-Value (TLV) encoded using big endian
   (network ordered) binary data representations.

   All SIMCO messages start with the SIMCO header containing message
   type, message length, and a message identifier. The rest of the
   message, the payload, contains zero, one, or more TLV message
   attributes.

4.1. Message Types

   The message type in the SIMCO header is divided into a basic type and
   a sub-type. There are four basic types of SIMCO messages:

      - request,
      - positive reply,
      - negative reply,
      - notification.

   Messages sent from the agent to the middlebox are always of basic
   type 'request message', while the basic type of messages sent from
   the middlebox to the agent is one of the three other types. Request
   messages and positive and negative reply messages belong to request
   transactions. From the agent's point of view, notification messages
   belong to notification transactions only. From the middlebox's point
   of view, a notification message may also belong to a request
   transaction. See section 2.3.4. of [RFC3989] for a detailed
   discussion of this issue.

   The message sub-type gives further information on the message type
   within the context of the basic message type. Only the combination
   of basic type and sub-type clearly identify the type of a message.

4.2. The SIMCO Header

   The SIMCO header is the first part of all SIMCO messages. It
   contains four fields: the basic message type, the message sub-type,
   the message length (excluding the SIMCO header) in octets, and the
   transaction identifier. The SIMCO header has a size of 64 bits. Its
   layout is defined in Figure 2.

```
                  Message Type
         _____ _____
        /                ^                \
        / _____ _____ \
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        | Basic Type    |   Sub-Type    |         Message Length        |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
        |              Transaction Identifier (TID)                     |
        +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
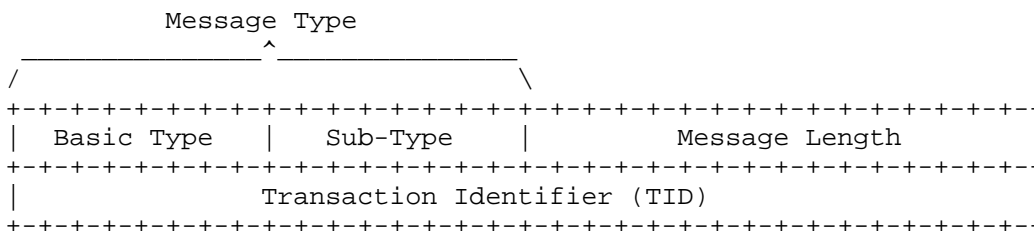
                      Figure 2: The SIMCO header

4.2.1. Basic Message Types

   For the basic type field, the following values are defined:

      0x01 : Request Message
      0x02 : Positive Reply Message
      0x03 : Negative Reply Message
      0x04 : Notification Message

4.2.2. Message Sub-types for Requests and Positive Replies

   For basic types 0x01 (request) and 0x02 (positive reply), a common
   set of values for the sub-type field is defined. Most of the sub-
   types can be used for both basic types. Restricted sub-types are
   marked accordingly.

      0x01 : (SE) session establishment
      0x02 : (SA) session authentication
      0x03 : (ST) session termination
      0x11 : (PRR) policy reserve rule
      0x12 : (PER) policy enable rule
      0x13 : (PEA) PER after reservation (request only)
      0x14 : (PDR) policy disable rule
      0x15 : (PLC) policy rule lifetime change
      0x16 : (PRD) policy rule deletion (positive reply only)
      0x21 : (PRS) policy rule status
      0x22 : (PRL) policy rule list
      0x23 : (PES) policy enable rule status (positive reply only)
      0x24 : (PDS) policy disable rule status (positive reply only)

4.2.3. Message Sub-types for Negative Replies

   For basic type 0x03 (negative reply message), the following values of
   the sub-type field are defined:

      Replies concerning general message handling
      0x10 : wrong basic request message type
      0x11 : wrong request message sub-type
      0x12 : badly formed request
      0x13 : reply message too big

      Replies concerning sessions
      0x20 : request not applicable
      0x21 : lack of resources
      0x22 : protocol version mismatch
      0x23 : authentication failed
      0x24 : no authorization
      0x25 : transport protocol problem

```
   0x26  :  security of underlying protocol layers insufficient

   Replies concerning policy rules
   0x40  :  transaction not supported
   0x41  :  agent not authorized for this transaction
   0x42  :  no resources available for this transaction
   0x43  :  specified policy rule does not exist
   0x44  :  specified policy rule group does not exist
   0x45  :  not authorized for accessing specified policy
   0x46  :  not authorized for accessing specified group
   0x47  :  requested address space not available
   0x48  :  lack of IP addresses
   0x49  :  lack of port numbers
   0x4A  :  middlebox configuration failed
   0x4B  :  inconsistent request
   0x4C  :  requested wildcarding not supported
   0x4D  :  protocol type doesn't match
   0x4E  :  NAT mode not supported
   0x4F  :  IP version mismatch
   0x50  :  conflict with existing rule
   0x51  :  not authorized to change lifetime
   0x52  :  lifetime can't be extended
   0x53  :  illegal IP Address
   0x54  :  protocol type not supported
   0x55  :  illegal port number
   0x56  :  illegal number of subsequent ports (NOSP)
   0x57  :  already enable PID
   0x58  :  parity doesn't match
```

4.2.4. Message Sub-types for Notifications

   For basic type 0x04, the following values of the sub-type field are
   defined:

```
   0x01  :  (BFM) badly formed message received
   0x02  :  (AST) asynchronous session termination
   0x03  :  (ARE) asynchronous policy rule event
```

4.2.5. Transaction Identifier

   The transaction identifier (TID) is an arbitrary number identifying
   the transaction. In a request message, the agent chooses an agent-
   unique TID, such that the same agent never uses the same TID in two
   different request messages belonging to the same session. Reply
   messages must contain the same TID as the corresponding request
   message. In a notification message, the middlebox chooses a

middlebox-unique TID, such that the same middlebox never uses the
same TID in two different notification messages belonging to the same
session.

4.3. The SIMCO Payload

A SIMCO payload consists of zero, one, or more type-length-value
(TLV) attributes. Each TLV attribute starts with a 16-bit type field
and a 16-bit length field, as shown in Figure 3.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          attribute type           |          attribute length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                              value
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


                               ...


     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                                  |
     +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
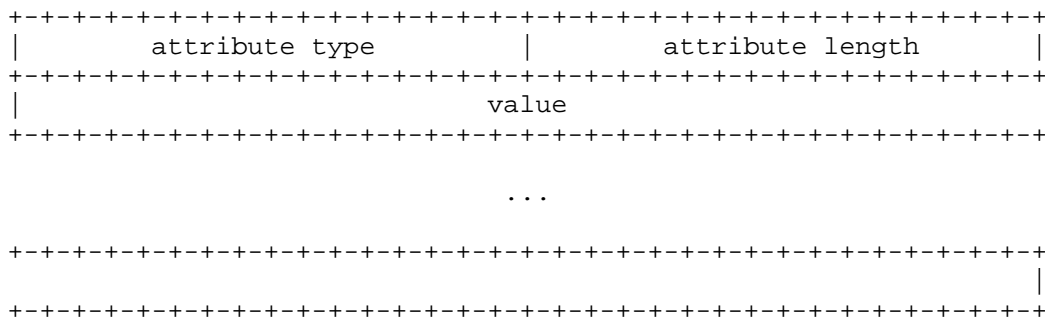
                Figure 3: Structure of TLV attribute

The attribute length field contains the length of the value field in
octets.

The following attribute types are defined:

```
    type        description                length
    -----------------------------------------------------
    0x0001  :  SIMCO protocol version      32 bits
    0x0002  :  authentication challenge    <= 4096 octets
    0x0003  :  authentication token        <= 4096 octets
    0x0004  :  middlebox capabilities      64 bits
    0x0005  :  policy rule identifier      32 bits
    0x0006  :  group identifier            32 bits
    0x0007  :  policy rule lifetime        32 bits
    0x0008  :  policy rule owner           <= 255 octets
    0x0009  :  address tuple               32, 96 or 192 bits
    0x000A  :  PRR parameter set           32 bits
    0x000B  :  PER parameter set           32 bits
```

4.3.1. SIMCO Protocol Version Attribute

   The SIMCO protocol version attribute has a length of four octets.
   The first two octets contain the version number, one the major
   version number and the other the minor version number. Two remaining
   octets are reserved.

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              0x0001            |              0x0004           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |major version #|minor version #|              reserved         |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
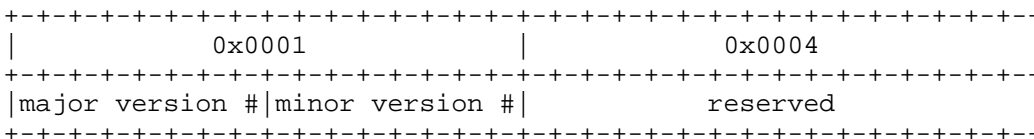
                   Figure 4: Protocol version attribute

   The SIMCO protocol specified within this document is version 3.0.
   The version numbers carried in the protocol version attribute are 3
   for major version number and 0 for minor version number.

4.3.2. Authentication Attributes

   The authentication challenge attribute and the authentication token
   attribute have the same format. Both contain a single value field
   with variable length. For both, the maximum length is limited to
   4096 octets. Please note that the length of these attributes may
   have values that are not multiples of 4 octets. In case of an
   authentication challenge attribute, the value field contains an
   authentication challenge sent from one peer to the other, requesting
   that the other peer authenticate itself.

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              0x0002           |           challenge length    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                            challenge                          |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                  ...

    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                              |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
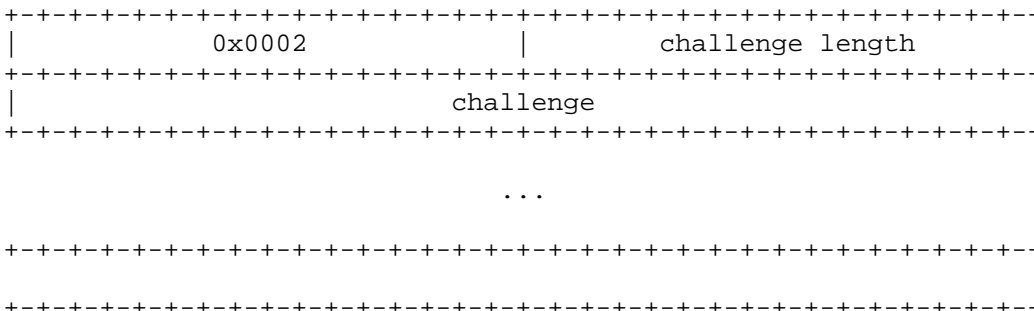
               Figure 5: Authentication challenge attribute

   The authentication token attribute is used for transmitting an
   authentication token.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0003             |      authentication length    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     authentication token
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                             ...

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
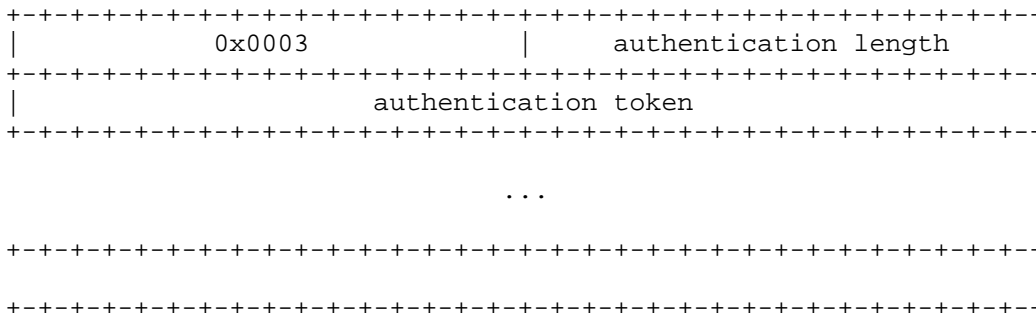
Figure 6: Authentication attribute

4.3.3. Middlebox Capabilities Attribute

   The middlebox capabilities attribute has a length of eight octets.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0004            |              0x0008            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    MB type    |I|E|P|S|IIV|EIV|           reserved            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    max policy rule lifetime                  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
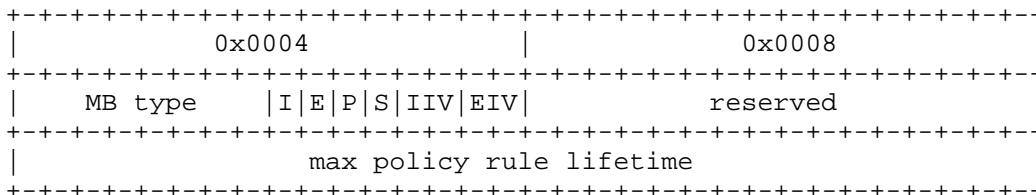
Figure 7: Capabilities attribute

   The first parameter field carries a bit field called MB type and
   provides information about the middlebox type. The following bits
   within the field are defined. The remaining ones are reserved.

       0x80 : packet filter firewall
       0x40 : network address translator
       0x10 : support of PDR transaction
       0x01 : port translation (requires 0x40 set)
       0x02 : protocol translation (requires 0x40 set)
       0x04 : twice NAT support (requires 0x40 set)

   For middleboxes that implement combinations of NAT and firewalls,
   combinations of those flags are possible. For instance, for a
   Network Address and Port Translator (NAPT) with packet filter and PDR
   transaction support, the value of the MB type parameter field is
   0xD1.

   The following four parameters fields are binary flags with a size of
   one bit:

```
          I     :   internal IP address wildcard support
          E     :   external IP address wildcard support
          P     :   port wildcard support
          S     :   persistent storage of policy rules
```

   The supported IP version for the internal and external network are
   coded into the IIV (Internal IP version) and EIV (external IP
   version) parameter fields. They both have a size of two bits.
   Allowed values are 0x1 for IP version 4 (IPv4), 0x2 for IP version 6
   (IPv6), and the combination of both (0x3) for IPv4 and IPv6 dual
   stack.

   The next parameter field with a length of 16 bits is reserved.

   The max policy rule lifetime parameter field specifies the maximum
   lifetime a policy rule may have.

4.3.4. Policy Rule Identifier Attribute

   The policy rule identifier (PID) attribute contains an identifier of
   a policy rule. The identifier has a length of four octets.

```
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             0x0005            |             0x0004            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                   policy rule identifier (PID)               |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
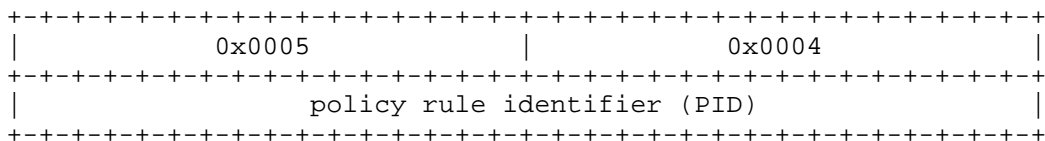
                 Figure 8: Policy rule identifier attribute

4.3.5. Group Identifier Attribute

   The group identifier (GID) attribute contains an identifier of a
   policy rule group. The identifier has a length of four octets.

```
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |             0x0006            |             0x0004            |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
       |                     group identifier (GID)                   |
       +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
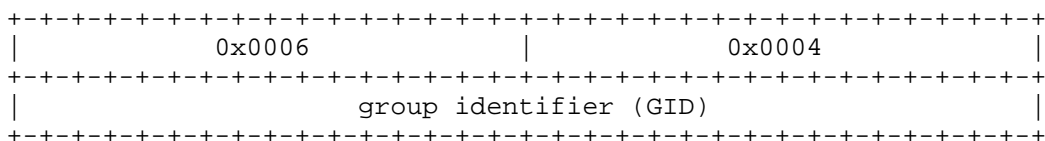
                  Figure 9: Group identifier attribute

4.3.6. Policy Rule Lifetime Attribute

   The policy rule lifetime attribute specifies the requested or actual
   remaining lifetime of a policy rule, in seconds. Its value field
   contains a 32-bit unsigned integer.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0007              |            0x0004             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                     policy rule lifetime                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
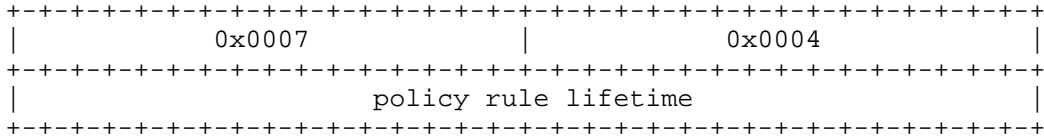
                   Figure 10: Policy rule lifetime attribute

4.3.7. Policy Rule Owner Attribute

   The policy rule owner attribute specifies the authenticated agent
   that created and owns the policy rule. Its value field does not have
   a fixed length, but its length is limited to 255 octets. Please note
   that the length of this attribute may have values that are not
   multiples of 4 octets. The owner is set by the middlebox.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x0008             |            owner length       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                            owner                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


                               ...


+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
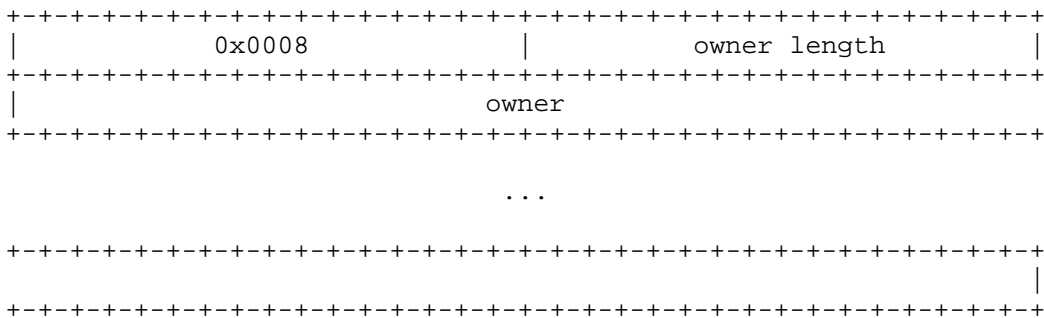
                   Figure 11: Policy rule owner attribute

4.3.8. Address Tuple Attribute

   The address tuple attribute contains a set of parameters specifying
   IP and transport addresses. The length of this attribute is 32, 96,
   or 192 bits.

   The first parameter field has a length of 4 bits. It indicates
   whether the contained parameters specify just the used protocols or
   also concrete addresses. Defined values for this field are:

      0x0 : full addresses
      0x1 : protocols only

   The second parameter field also has a length of 4 bits. It specifies
   the IP version number. Defined values for this field are:

      0x1 : IPv4
      0x2 : IPv6

The third parameter field has a length of 8 bits. It specifies a
prefix length to be used for IP address wildcarding (see Section
1.1).

The fourth parameter field has also a length of 8 bits. It specifies
the transport protocol. Defined values for this field are all values
that are allowed in the 'Protocol' field of the IPv4 header [RFC791]
or in the 'Next Header field' of the IPv6 header [RFC2460]. The set
of defined numbers for these fields is maintained by the Internet
Assigned Numbers Authority (IANA) under the label 'PROTOCOL NUMBERS'.

The fifth parameter field has also a length of 8 bits. It specifies
the location of the address. Defined values for this field are:

    0x00 : internal (A0)
    0x01 : inside   (A1)
    0x02 : outside (A2)
    0x03 : external (A3)

Port and address wildcarding can only be used in PER and PEA
transactions. The address tuple attribute carries a port number of 0
if the port should be wildcarded. For IPv4, a prefix length less
than 0x20 is IP address wildcarding. For IPv6, a prefix length less
than 0x80 is IP address wildcarding.

The port range field must be always greater than zero, but at least
1.

```
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                0x0009                 |              0x0004     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | 0x1 |IP ver.| prefix length |trnsp. protocol|   location     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                0x0009                 |              0x000C     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | 0x0 |  0x1  | prefix length |trnsp. protocol|   location     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |           port number         |             port range         |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                         IPv4 address                          |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+


         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                0x0009                 |              0x0018     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         | 0x0 |  0x2  | prefix length |trnsp. protocol|   location     |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |           port number         |             port range         |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
         |                                                               |
         +                                                               +
         |                                                               |
         +                         IPv6 address                          +
         |                                                               |
         +                                                               +
         |                                                               |
         +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
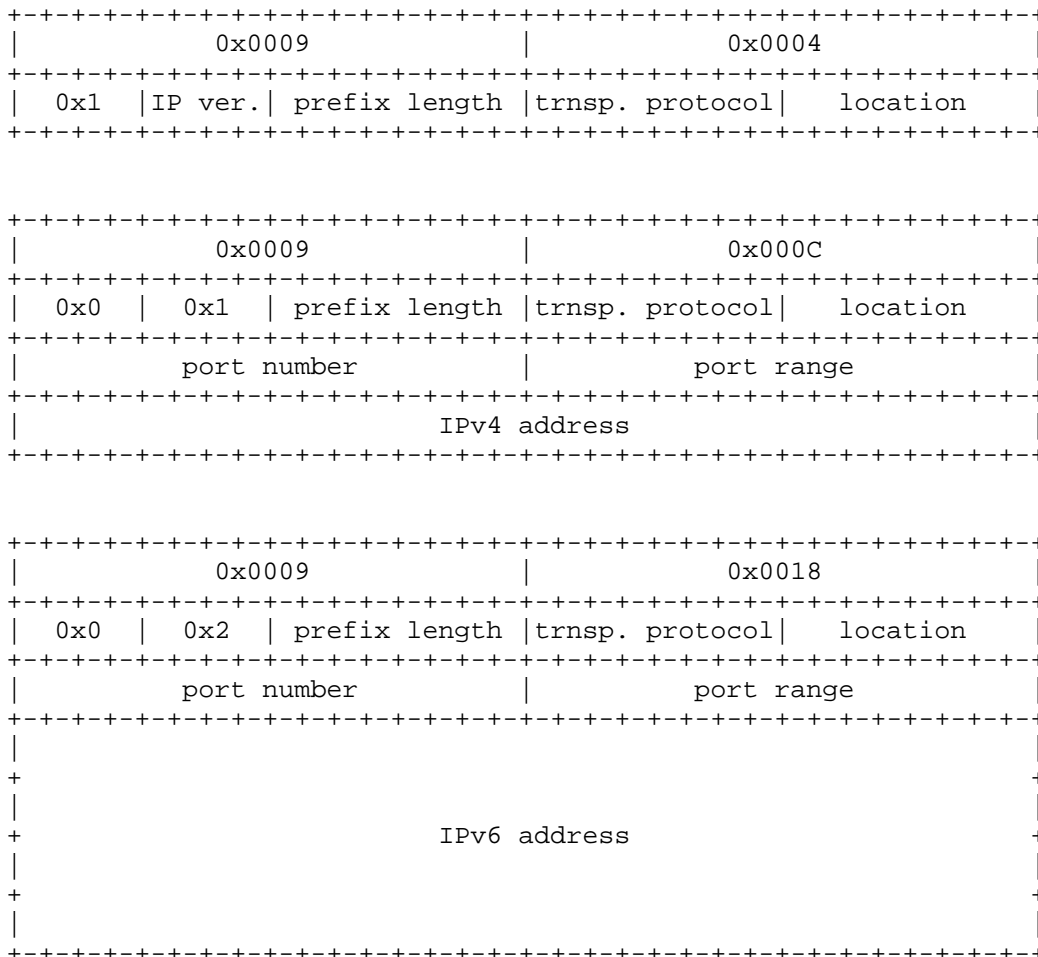
                     Figure 12: Address tuple attributes

4.3.9. PRR Parameter Set Attribute

   The policy reserve rule (PRR) parameter set attribute contains all
   parameters of the PRR request except the group identifier:

        - NAT mode
        - port parity
        - requested inside IP version
        - requested outside IP version
        - transport protocol
        - port range

The attribute value field has a total size of 32 bits. It is sub-
divided into six parameter fields.

The first parameter field, called NM, has a length of 2 bits and
specifies the requested NAT mode of the middlebox at which a
reservation is requested. Defined values for this field are:

    01 : traditional
    10 : twice

The second parameter field, called PP, has also a length of 2 bits.
It specifies the requested port parity. Defined values for this
field are:

    00 : any
    01 : odd
    10 : even

The third and the fourth parameter fields are called IPi and IPo,
respectively. Both have a length of 2 bits. They specify the
requested version of the IP protocol at the inside (IPi) or outside
(IPo) of the middlebox, respectively. Defined values for these
fields are:

    00 : any
    01 : IPv4
    10 : IPv6

The fifth parameter field has a length of 8 bits. It specifies the
transport protocol for which the reservation should be made. A value
of zero indicates that the reservation is requested for an IP address
without specific selection of a protocol and a port number. Allowed
non-zero values are the defined values for the 'protocol' field in
the IPv4 header and IPv6 extension headers. The set of defined
numbers for these fields is maintained by the Internet Assigned
Numbers Authority (IANA) under the label 'PROTOCOL NUMBERS'.

The sixth parameter field has a length of 16 bits. It contains an
unsigned integer specifying the length of the port range that should
be supported. A value of 0xFFFF indicates that the reservation
should be made for all port numbers of the specified transport
protocol. A port range field with the value of 0x0001 specifies that
only a single port number should be reserved. Values greater than
one indicate the number of consecutive port numbers to be reserved.
A value of zero is not valid for this field.

Please note that the wildcarding value 0xFFFF can only be used in the
port range field in the PRR parameter set attribute. In the address
tuple attribute, wildcarding of port numbers is specified by the port
number field having a value of zero (see Section 4.3.8).

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x000A              |              0x0004            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|NM |PP |IPi|IPo|trnsp. protocol|          port range           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
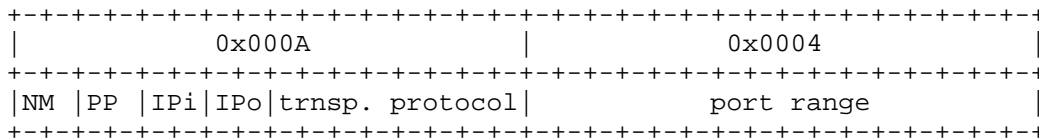
                Figure 13: PRR parameter set attribute

4.3.10. PER Parameter Set Attribute

The policy enable rule (PER) parameter set attribute contains two
parameters: the requested port parity, and the direction of the
enabled data stream. The attribute value field has a total size of
32 bits, and it is sub-divided into 3 parameter fields.

The first parameter field has a length of 8 bits. It specifies the
requested port parity. Defined values for this field are:

    0x00 : any
    0x03 : same

The second parameter field has a length of 8 bits. It specifies the
direction of the enabled data stream. Defined values for this field
are:

    0x01 : inbound
    0x02 : outbound
    0x03 : bi-directional

The third parameter field has a length of 16 bits and is reserved.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            0x000B              |              0x0004            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  port parity  |   direction   |            reserved           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```
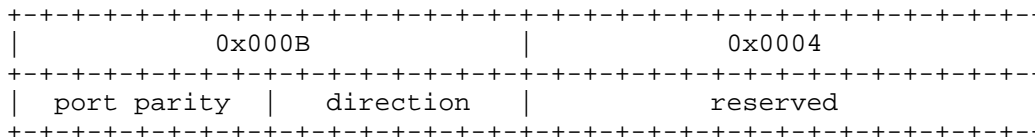
                Figure 14: PER parameter set attribute

5. SIMCO Message Formats

    In the following, the formats of the different SIMCO message types
    are defined. The definitions are grouped into protocol error
    messages, session control messages, and policy rule control messages.

5.1. Protocol Error Replies and Notifications

    When processing a received message, the middlebox might run into
    message processing problems before it can identify whether the
    message concerns session control or policy rule control. Also, it
    might not be possible to determine the message type, or it might
    detect a wrong message format. In these cases, the Badly Formed
    Message (BFM) notification or one of the following negative replies
    is sent:

        0x0401 : BFM notification
        0x0310 : wrong basic request message type
        0x0311 : wrong request message sub-type
        0x0312 : badly formed request

5.1.1. BFM Notification

    The Badly Formed Message (BFM) notification message is sent from the
    middlebox to the agent after a message was received that does not
    comply to the SIMCO message format definition. The BFM notification
    has no attributes and contains the SIMCO header only.

```
                  +--------------------------+
                  | SIMCO header             |
                  +--------------------------+
```

                  Figure 15: BFM notification structure

5.1.2. Protocol Error Negative Replies

    Protocol error negative replies are sent from the middlebox to the
    agent if a message cannot be clearly interpreted, as it does not
    comply with any defined message format. Protocol error negative
    replies include 'wrong basic request message type' (0x0310), 'wrong
    request message sub-type' (0x0311), and 'badly formed request'
    (0x0312). These replies have no attributes. They consist of the
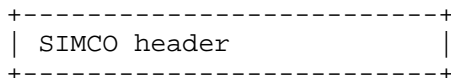   SIMCO header only.

```
                 +--------------------------+
                 | SIMCO header             |
                 +--------------------------+
```

Figure 16: Protocol error negative reply structure

5.2. Session Control Messages

Session control messages include the following list of message types
(composed of basic type and sub-type):

```
   0x0101  :  SE request
   0x0102  :  SA request
   0x0103  :  ST request
   0x0201  :  SE positive reply
   0x0202  :  SA positive reply
   0x0203  :  ST positive reply
   0x0310  :  negative reply: wrong basic request message type
   0x0311  :  negative reply: wrong request message sub-type
   0x0312  :  negative reply: badly formed request
   0x0320  :  negative reply: request not applicable
   0x0321  :  negative reply: lack of resources
   0x0322  :  negative reply: protocol version mismatch
   0x0323  :  negative reply: authentication failed
   0x0324  :  negative reply: no authorization
   0x0325  :  negative reply: transport protocol problem
   0x0326  :  negative reply: security of underlying protocol layers
                             insufficient
   0x0401  :  BFM notification
   0x0402  :  AST notification
```

5.2.1. SE Request

The Session Establishment (SE) request message is sent from the agent
to the middlebox to request establishment of a session. The SE
request message contains one or two attributes: a mandatory SIMCO
version number attribute and an optional authentication challenge
attribute requesting that the middlebox authenticate itself.

```
                 +--------------------------+
                 | SIMCO header             |
                 +--------------------------+
                 | SIMCO protocol version   |
                 +--------------------------+
                 | authentication challenge | optional
                 +--------------------------+
```
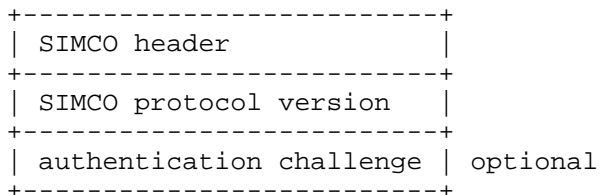
Figure 17: Structure of SE request

5.2.2. SE Positive Reply

   The Session Establishment (SE) reply message indicates completion of
   session establishment. It contains a single mandatory attribute: the
   middlebox capabilities attribute.

```
              +--------------------------+
              | SIMCO header             |
              +--------------------------+
              | middlebox capabilities   |
              +--------------------------+
```
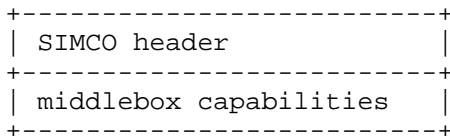
            Figure 18: Structure of SE positive reply

5.2.3. SA Positive Reply

   If the agent requested middlebox authentication, or if the middlebox
   wants the agent to authenticate itself, then the middlebox replies on
   the SE request with a Session Authentication (SA) reply message
   instead of an SE reply message. The SA reply message contains two
   optional attributes, but at least one of them needs to be present.
   The first one is an authentication challenge attribute requesting
   that the agent authenticate itself. The second one is an
   authentication token attribute authenticating the middlebox as the
   reply to an authentication request by the agent.

```
              +--------------------------+
              | SIMCO header             |
              +--------------------------+
              | authentication challenge | optional
              +--------------------------+
              | authentication token     | optional
              +--------------------------+
```
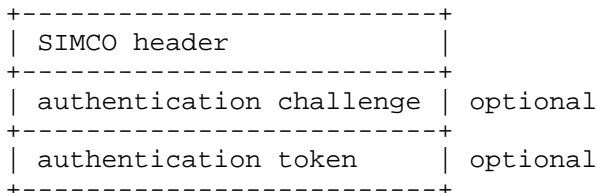
            Figure 19: Structure of SA positive reply

5.2.4. SA Request

   The Session Authentication (SA) request message is sent from the
   agent to the middlebox after an initial SE request was answered by an
   SA reply. The SE request message contains one optional attribute: an
   authentication token attribute authenticating the agent as the
   response to an authentication challenge sent by the middlebox in an
   SA reply.

```
+--------------------------+
| SIMCO header             |
+--------------------------+
| authentication token     | optional
+--------------------------+
```
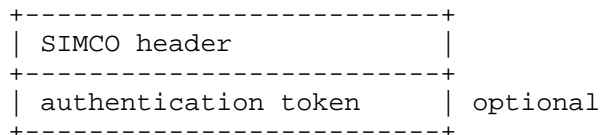
Figure 20: Structure of SA request

5.2.5. ST Request and ST Positive Reply

   The Session Termination (ST) request message is sent from the agent
   to the middlebox to request termination of a session. The ST
   positive reply is returned, acknowledging the session termination.
   Both messages have no attributes and contain the SIMCO header only.

```
+--------------------------+
| SIMCO header             |
+--------------------------+
```
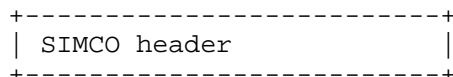
Figure 21: Structure of ST request and positive reply

5.2.6. SE Negative Replies

   There are nine different negative reply messages that can be sent
   from a middlebox to the agent if the middlebox rejects an SE request.
   Three of them are protocol error negative replies (0x031X) already
   covered in Section 4.1.2.

   The remaining six negative replies are specific to session
   establishment. One of them, the 'protocol version mismatch' negative
   reply (0x0322), contains a single attribute: the protocol version
   attribute.

```
+--------------------------+
| SIMCO header             |
+--------------------------+
| SIMCO protocol version   |
+--------------------------+
```
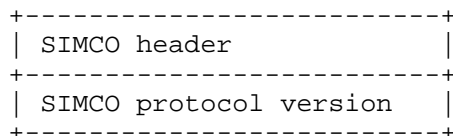
Figure 22a: Structure of SE negative replies

   The remaining three replies include 'request not applicable'
   (0x0320), 'lack of resources' (0x0321), 'authentication failed'
   (0x0323), 'no authorization' (0x0324), 'transport protocol problem'
   (0x0325), and 'security of underlying protocol layers insufficient'
   (0x0326). They consist of the SIMCO header only.

```
                +-------------------------+
                | SIMCO header            |
                +-------------------------+
```
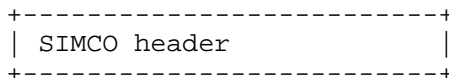
          Figure 22b: Structure of SE negative replies

5.2.7. AST Notification

   The Asynchronous Session Termination (AST) notification message is
   sent from the middlebox to the agent, if the middlebox wants to
   terminate a SIMCO session. It has no attributes and contains the
   SIMCO header only.

```
                +-------------------------+
                | SIMCO header            |
                +-------------------------+
```
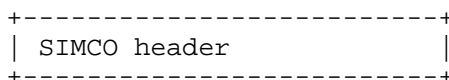
          Figure 22a: Structure of AST notifications

5.3. Policy Rule Control Messages

   Policy Rule control messages include the following list of message
   types (composed of basic type and sub-type):

   0x0111 : PRR request
   0x0112 : PER request
   0x0113 : PEA request
   0x0114 : PDR request
   0x0115 : PLC request
   0x0121 : PRS request
   0x0122 : PRL request
   0x0211 : PRR positive reply
   0x0212 : PER positive reply
   0x0214 : PDR positive reply
   0x0215 : PLC positive reply
   0x0216 : PRD positive reply
   0x0221 : PRS positive reply
   0x0223 : PES positive reply
   0x0224 : PDS positive reply
   0x0222 : PRL positive reply
   0x0310 : negative reply: wrong basic request message type
   0x0311 : negative reply: wrong request message sub-type
   0x0312 : negative reply: badly formed request
   0x0340 : negative reply: transaction not supported
   0x0341 : negative reply: agent not authorized for this transaction
   0x0342 : negative reply: no resources available for this
                              transaction
   0x0343 : negative reply: specified policy rule does not exist

```
0x0344  :  negative reply: specified policy rule group does not exist
0x0345  :  negative reply: not authorized for accessing this policy
0x0346  :  negative reply: not authorized for accessing specified
                           group
0x0347  :  negative reply: requested address space not available
0x0348  :  negative reply: lack of IP addresses
0x0349  :  negative reply: lack of port numbers
0x034A  :  negative reply: middlebox configuration failed
0x034B  :  negative reply: inconsistent request
0x034C  :  negative reply: requested wildcarding not supported
0x034D  :  negative reply: protocol type doesn't match
0x034E  :  negative reply: NAT mode not supported
0x034F  :  negative reply: IP version mismatch
0x0350  :  negative reply: conflict with existing rule
0x0351  :  negative reply: not authorized to change lifetime
0x0352  :  negative reply: lifetime can't be extended
0x0353  :  negative reply: illegal IP Address
0x0354  :  negative reply: protocol type not supported
0x0355  :  negative reply: illegal port number
0x0356  :  negative reply: illegal NOSP
0x0357  :  negative reply: already enable PID
0x0358  :  negative reply: parity doesn't match
0x0401  :  negative reply: BFM notification
0x0403  :  negative reply: ARE notification
```

5.3.1. Policy Events and Asynchronous Notifications

   SIMCO maintains an owner attribute for each policy rule at the
   middlebox. Depending on the configuration of the middlebox, several
   agents may access the same policy rule; see also [RFC3989], Sections
   2.1.5 and 2.3.4.

   To keep all agents synchronized about the state of their policy
   rules, SIMCO generates Asynchronous Rule Event (ARE) notifications.
   When an agent is reserving or enabling a policy rule, the middlebox
   sends an ARE to all agents that are authorized to access this policy
   rule. The middlebox sends an ARE to all agents authorized to access
   this policy rule when the rule lifetime is modified or if the rule is
   deleted.

5.3.2. PRR Request

   The Policy Reserve Rule (PRR) request message is sent from the agent
   to the middlebox to request reservation of an IP address (and
   potentially also a range of port numbers) at the middlebox. Besides
   the SIMCO header, the request message contains two or three
   attributes. The first one is the PRR parameter set attribute
   specifying all parameters of the request except the requested policy

rule lifetime and the group identifier. The missing parameters are
covered by the following two attributes. The last attribute, the
group identifier, is optional.

```
            +-------------------------+
            | SIMCO header            |
            +-------------------------+
            | PRR parameter set       |
            +-------------------------+
            | policy rule lifetime    |
            +-------------------------+
            | group identifier        | optional
            +-------------------------+
```
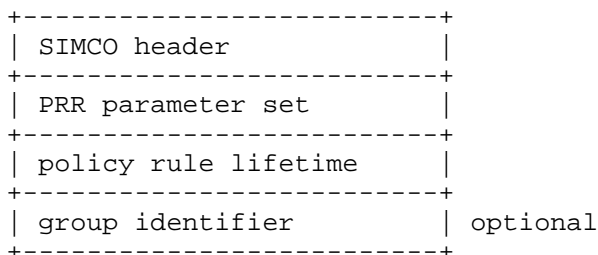
Figure 23: Structure of PRR request

5.3.3. PER Request

The Policy Enable Rule (PER) request message is sent from the agent
to the middlebox to request enabling of data communication between an
internal and an external address. Besides the SIMCO header, the
request message contains four or five attributes. The first one is
the PER parameter set attribute specifying all parameters of the
request except the internal address, the external address, the
requested policy rule lifetime, and the group identifier. The
missing parameters are covered by the following four attributes. Two
address tuple parameters specify internal and external address
tuples. Much like the PRR request, the last two attributes specify
the requested lifetime and group identifier. The group identifier
attribute is optional.

```
            +-------------------------+
            | SIMCO header            |
            +-------------------------+
            | PER parameter set       |
            +-------------------------+
            | address tuple (internal)|
            +-------------------------+
            | address tuple (external)|
            +-------------------------+
            | policy rule lifetime    |
            +-------------------------+
            | group identifier        | optional
            +-------------------------+
```
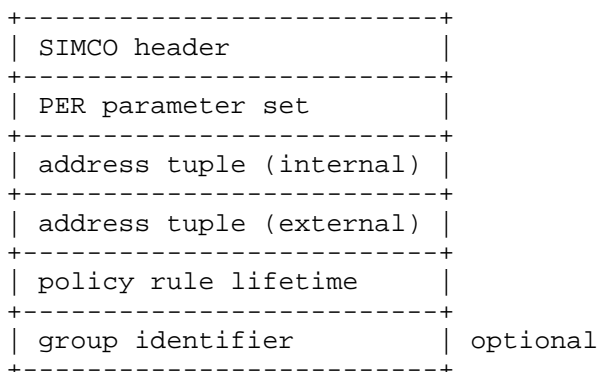
Figure 24: Structure of PER request

5.3.4. PEA Request

   The Policy Enable rule After reservation (PEA) request message is
   sent from the agent to the middlebox to request enabling of data
   communication between an internal and an external address. It is
   similar to the PER request. There is just one difference. The
   optional group identifier attribute of the PER request is replaced by
   a mandatory policy rule identifier attribute referencing an already
   established policy reserve rule established by a PRR transaction.

```
              +--------------------------+
              | SIMCO header             |
              +--------------------------+
              | PER parameter set        |
              +--------------------------+
              | address tuple (internal) |
              +--------------------------+
              | address tuple (external) |
              +--------------------------+
              | policy rule lifetime     |
              +--------------------------+
              | policy rule identifier   |
              +--------------------------+
```
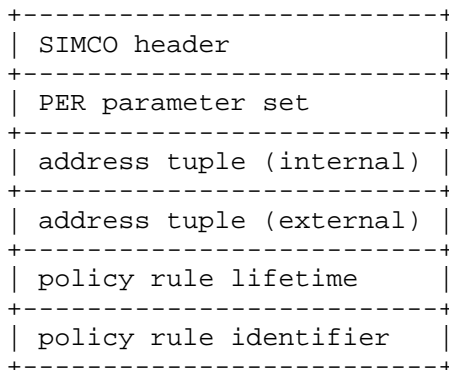
               Figure 25: Structure of PEA request

   The group identifier attribute is not included in the PEA request,
   since the group membership of the policy enable rule is inherited of
   the policy reserve rule.

5.3.5. PLC Request

   The Policy Rule Lifetime Change (PLC) request message is sent from
   the agent to the middlebox to request a change of the remaining
   policy lifetime. Besides the SIMCO header, the request message
   contains two attributes specifying the policy rule to which the
   change should be applied and specifying the requested remaining
   lifetime.

```
              +--------------------------+
              | SIMCO header             |
              +--------------------------+
              | policy rule identifier   |
              +--------------------------+
              | policy rule lifetime     |
              +--------------------------+
```
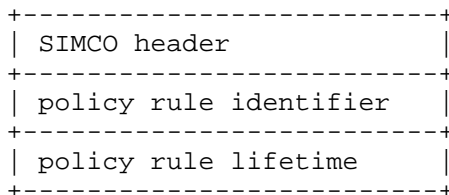
               Figure 26: Structure of PLC request

5.3.6. PRS Request

   The Policy Rule Status (PRS) request message is sent from the agent
   to the middlebox to request a report on the status of a specified
   policy rule. Besides the SIMCO header, the request message contains
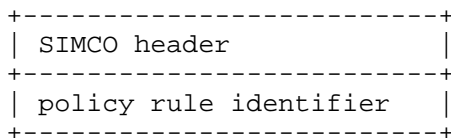   just one attribute specifying the policy rule for which the report is
   requested.

```
                    +--------------------------+
                    | SIMCO header             |
                    +--------------------------+
                    | policy rule identifier   |
                    +--------------------------+
```

               Figure 27: Structure of PRS request

5.3.7. PRL Request

   The Policy Rule List (PRL) request message is sent from the agent to
   the middlebox to request a list of all policy rules accessible to the
   agent. The message consists of the SIMCO header only.

```
                    +--------------------------+
                    | SIMCO header             |
                    +--------------------------+
```
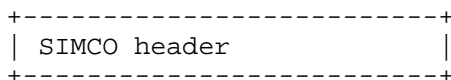
               Figure 28: Structure of PRL request

5.3.8. PDR Request

   The Policy Disable Rule (PDR) request message is sent from the agent
   to the middlebox to request a disable rule. The message consists of
   the SIMCO header, an internal address tuple, an external address
   tuple, and a lifetime attribute.

```
                    +--------------------------+
                    | SIMCO header             |
                    +--------------------------+
                    | address tuple (internal) |
                    +--------------------------+
                    | address tuple (external) |
                    +--------------------------+
                    | policy rule lifetime     |
                    +--------------------------+
```
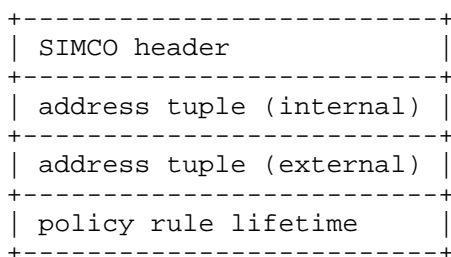
               Figure 29: Structure of PDR request

5.3.9. PRR Positive Reply

   The Policy Reserve Rule (PRR) positive reply is sent after successful
   reservation of an address at the inside or outside of the middlebox.
   The message contains four mandatory attributes and an optional
   attribute: the policy rule identifier of the new policy reserve rule,
   the corresponding group identifier, the remaining lifetime of the
   policy rule, the reserved outside address tuple, and the optional
   reserved inside address tuple. The reserved inside address tuple is
   only returned when the middlebox is of type twice-NAT.

```
            +--------------------------+
            | SIMCO header             |
            +--------------------------+
            | policy rule identifier   |
            +--------------------------+
            | group identifier         |
            +--------------------------+
            | policy rule lifetime     |
            +--------------------------+
            | address tuple (outside)  |
            +--------------------------+
            | address tuple (inside)   | optional
            +--------------------------+
```
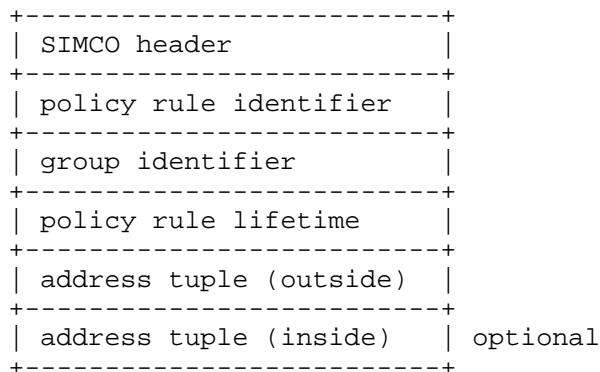
              Figure 30: Structure of PRR positive reply

5.3.10. PER Positive Reply

   The Policy Enable Rule (PER) positive reply is sent after the
   middlebox successfully enables data transfer between an internal and
   an external address (by using a PER or PEA request message). The
   message contains five attributes: the policy rule identifier of the
   new policy enable rule, the corresponding group identifier, the
   remaining lifetime of the policy rule, the address tuple at the
   outside of the middlebox, and the address tuple at the inside of the
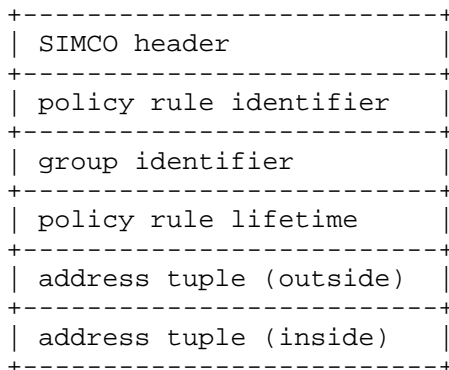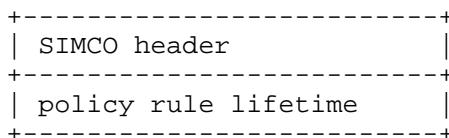   middlebox.

```
            +-------------------------+
            | SIMCO header            |
            +-------------------------+
            | policy rule identifier  |
            +-------------------------+
            | group identifier        |
            +-------------------------+
            | policy rule lifetime    |
            +-------------------------+
            | address tuple (outside) |
            +-------------------------+
            | address tuple (inside)  |
            +-------------------------+
```

         Figure 31: Structure of PER positive reply

5.3.11. PLC Positive Reply

   The Policy Lifetime Change (PLC) positive reply is sent after the
   middlebox changes the lifetime of a policy rule to a positive (non-
   zero) value. The message contains just a single attribute: the
   remaining lifetime of the policy rule.

```
            +-------------------------+
            | SIMCO header            |
            +-------------------------+
            | policy rule lifetime    |
            +-------------------------+
```

         Figure 32: Structure of PLC positive reply

5.3.12. PRD Positive Reply

   The Policy Rule Deleted (PRD) positive reply is sent after the
   middlebox changes the remaining lifetime of a policy rule to zero,
   which means that it terminates the policy rule. The message consists
   of the SIMCO header only.

```
            +-------------------------+
            | SIMCO header            |
            +-------------------------+
```
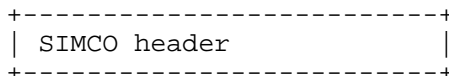
         Figure 33: Structure of PRD positive reply

5.3.13. PRS Positive Reply

   The Policy Reserve Rule Status (PRS) positive reply is used for
   reporting the status of a policy reserve rule. The message format is
   identical with the format of the PRR positive reply except that it
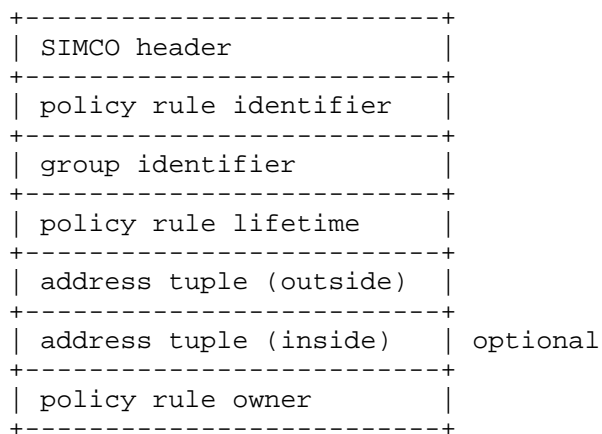   contains, in addition, a policy rule owner attribute.

```
                  +--------------------------+
                  | SIMCO header             |
                  +--------------------------+
                  | policy rule identifier   |
                  +--------------------------+
                  | group identifier         |
                  +--------------------------+
                  | policy rule lifetime     |
                  +--------------------------+
                  | address tuple (outside)  |
                  +--------------------------+
                  | address tuple (inside)   | optional
                  +--------------------------+
                  | policy rule owner        |
                  +--------------------------+
```

             Figure 34: Structure of PRS positive reply

5.3.14. PES Positive Reply

   The Policy Enable Rule Status (PES) positive reply is used for
   reporting the status of a policy enable rule.

```
                    +-------------------------+
                    | SIMCO header            |
                    +-------------------------+
                    | policy rule identifier  |
                    +-------------------------+
                    | group identifier        |
                    +-------------------------+
                    | PER parameter set       |
                    +-------------------------+
                    | address tuple (internal)|
                    +-------------------------+
                    | address tuple (inside)  |
                    +-------------------------+
                    | address tuple (outside) |
                    +-------------------------+
                    | address tuple (external)|
                    +-------------------------+
                    | policy rule lifetime    |
                    +-------------------------+
                    | policy rule owner       |
                    +-------------------------+
```
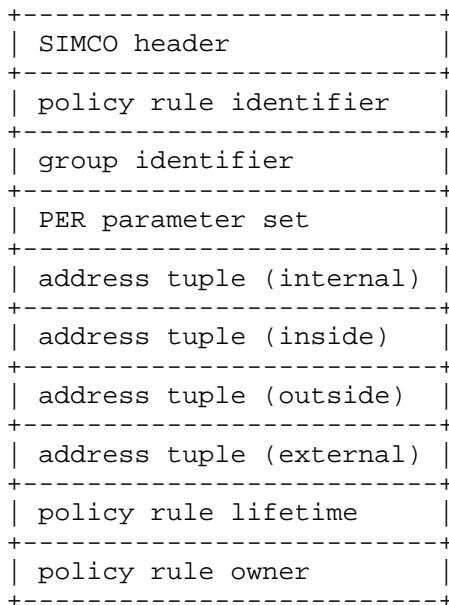
                Figure 35: Structure of PES positive reply

5.3.15. PDS Positive Reply

   The Policy Disable Rule Status (PDS) positive reply is used for
   reporting the status of a policy disable rule. The message contains
   five attributes: the policy rule identifier, the internal and
   external address tuples, the policy disable rule lifetime, and the
   policy rule owner.

```
              +-------------------------+
              | SIMCO header            |
              +-------------------------+
              | policy rule identifier  |
              +-------------------------+
              | address tuple (internal)|
              +-------------------------+
              | address tuple (external)|
              +-------------------------+
              | policy rule lifetime    |
              +-------------------------+
              | policy rule owner       |
              +-------------------------+
```
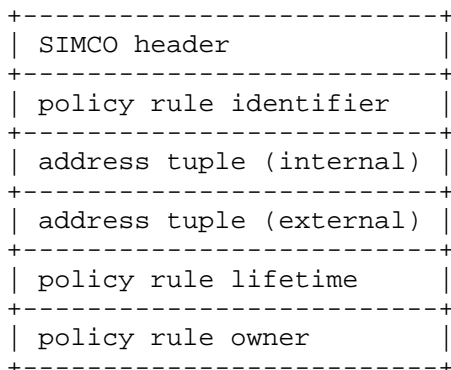
             Figure 36: Structure of PDS positive reply

3.5.16. PRL Positive Reply

   The Policy Rule List (PRL) positive reply is used for reporting the
   list of all established policy rules. The number of attributes of
   this message is variable. The message contains one policy rule
   identifier attribute per established policy rule.

```
              +-------------------------+
              | SIMCO header            |
              +-------------------------+
              | policy rule identifier  |
              +-------------------------+
              | policy rule identifier  |
              +-------------------------+
              |                         |
                         . . .
              |                         |
              +-------------------------+
              | policy rule identifier  |
              +-------------------------+
```
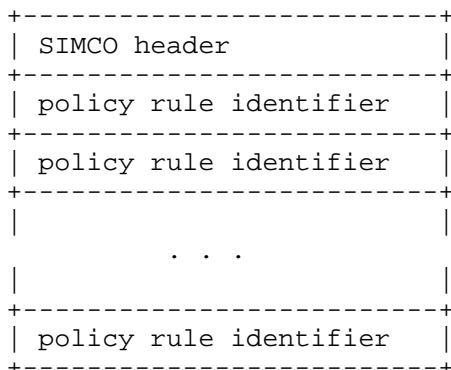
             Figure 37: Structure of PRL positive reply

5.3.17. PDR Positive Replies

   The Policy Disable Rule (PDR) positive reply is sent after the
   middlebox successfully enables the policy disable rule and removal of
   conflicting policy rules. The message contains two attributes: the
   policy rule identifier of the new policy disable rule, and the
   remaining lifetime of the policy rule.

```
                  +--------------------------+
                  | SIMCO header             |
                  +--------------------------+
                  | policy rule identifier   |
                  +--------------------------+
                  | policy rule lifetime     |
                  +--------------------------+
```

              Figure 38: Structure of PDR positive reply

5.3.18. Policy Rule Control Negative Replies

   Session establishment negative replies are sent from the middlebox to
   the agent if a middlebox rejects a policy rule control request.
   Beyond protocol error replies, a number of policy rule control-
   specific negative reply messages that can be sent. They are listed
   at the beginning of Section 5.3. They all have no attributes. They
   consist of the SIMCO header only.

```
                  +--------------------------+
                  | SIMCO header             |
                  +--------------------------+
```

        Figure 39: Structure of Policy rule control negative replies

5.3.19. ARE Notification

   The Asynchronous Policy Rule Event (ARE) notification message is sent
   from the middlebox to the agent. All agents participating in an open
   SIMCO session that are authorized to access this policy rule and are
   not explicitly requesting an action (i.e., reserving, enabling, and
   changing lifetime) receive such an ARE notification, when:

     - a policy rule is deleted (lifetime attribute = 0)

     - a policy rule is reserved (lifetime attribute = lifetime)

     - a policy rule is enabled (lifetime attribute = lifetime)

     - a policy rule's lifetime changed (lifetime attribute = lifetime)

Besides the SIMCO header, the request message contains two attributes specifying the policy rule that is concerned and the current lifetime.

```
          +--------------------------+
          | SIMCO header             |
          +--------------------------+
          | policy rule identifier   |
          +--------------------------+
          | policy rule lifetime     |
          +--------------------------+
```
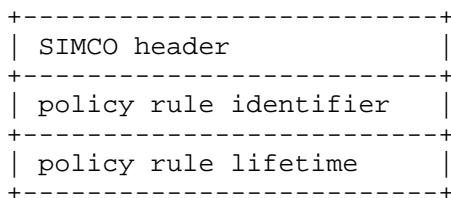
Figure 40: Structure of ARE notification

6. Message Format Checking

   This section describes common processing of all messages that are received by a middlebox.

   1) When a message arrives at a middlebox, the header is checked for consistency before the payload is processed.

      o If the header checks fail, the middlebox sends a BFM notification.

      o If a session is already established, then the middlebox also sends an AST notification and closes the connection.

   2) The middlebox waits until it has received as many octets from the agent as specified by the message length plus 8 octets (the length of the SIMCO header).

      o If the middlebox is still waiting and does not receive any more octets from the agent for 60 seconds, it sends a BFM notification.

      o If a session is already established, then the middlebox also sends an AST notification and closes the connection after sending the BFM notification; otherwise, it closes the connection without sending another message.

   3) After receiving a sufficient number of octets, the middlebox reads the transaction identifier and the basic message type.

      o If the value of the basic message type fields does not equal 0x01 (request message), then the middlebox stops processing the message and sends a negative reply of type 'wrong basic request message type' (0x0310) to the agent.

o If no session is established, then the middlebox closes the
   connection after sending the 0x0310 reply.

4) Then the middlebox checks the message sub-type.

   o If no session is established, then only sub-type 'session
      establishment' (0x01) is accepted. For all other sub-types,
      the middlebox sends a reply of type 'wrong request message
      sub-type' (0x0311) to the agent and closes the connection after
      sending the reply.

   o If a session is already established, then the middlebox checks
      if the message sub-type is one of the sub-types defined in
      Section 4.2.2. (excluding 'session establishment' (0x01),
      'session termination' (0x03), and 'policy rule
      deletion'(0x15)).

      o If not, then the middlebox stops processing the message and
         sends a reply of type 'wrong request message sub-type'
         (0x0311) to the agent.

5) Then the middlebox checks the TLV-structured attributes in the
   message.

   o If their type or number does not comply with the defined format
      for this message type, the middlebox stops processing the
      message and sends a reply of type 'badly formed request'
      (0x0312) to the agent.

   o If no session is established, then the middlebox closes the
      connection after sending the 0x0312 reply.

6) After all message format checks are passed, the middlebox
   processes the content of the attributes as described in the
   following sections.

7. Session Control Message Processing

   For session control, the agent can send SE, SA, and ST request
   messages. The middlebox then sends per request a single reply back
   to the agent. Additionally, the middlebox may send unsolicited AST
   notifications.

7.1. Session State Machine

   For each session, there is a session state machine illustrated by the
   figure below.

```
            SE/BFM
            SE/0x031X
            SE/0x032X
            +-------+
            |       v
         +----------+
         |  CLOSED  |---------------+
         +----------+               |
           |   ^  ^                 |
           |   |  |  SA/BFM         |  SE/SA
           |   |  |  SA/0x031X      |
           |   |  |  SA/0x032X      |
    SE/SE  |   |  |  ST/ST          v
           |   |  |  AST         +----------+
           |   |  +-------------|  NOAUTH  |
           |   |                +----------+
           |   | AST              |
        v  |   | ST/ST            | SA/SE
         +----------+             |
         |   OPEN   |<------------+
         +----------+
```

                 Figure 41: Session state machine

   The figure illustrates all possible state transitions of a session.
   Request transactions (SE, SA, ST) are denoted by a descriptor of the
   request message, a '/' symbol, and a descriptor of the reply message.
   Notification transactions are denoted just by the a notification
   descriptor. For example, a successful SE transaction is denoted by
   'SE/SE', and an AST notification is denoted by 'AST'.

   Initially, all sessions are in state CLOSED. From there, a
   successful SE transaction can change its state either to NOAUTH or to
   OPEN. From state NOAUTH, a successful SA transaction changes session
   state to OPEN. A failed SA transaction changes session state from

   NOAUTH back to CLOSED. Successful ST transactions and AST
   notifications change sessions from state NOAUTH or from state OPEN to
   state CLOSED.

   A SIMCO session is established in state OPEN, which is the only state
   in which the middlebox accepts requests other than SE, SA, and ST.

7.2. Processing SE Requests

   The SE request is only applicable if the session is in state CLOSED.
   If a session is in state NOAUTH or OPEN, then the middlebox sends a
   negative reply message of type 'request not applicable' (0x0320) to
   the agent, leaving the state of the session unchanged.

   Before processing the content of the SE request message, the
   middlebox may check its resources and decide that available resources
   are not sufficient to serve the agent. In such a case, the middlebox
   returns a negative reply of type 'lack of resources' (0x0321) and
   closes the connection. Furthermore, the middlebox may decide to
   reject the SE request if the selected network connection and its
   protocol specific parameters are not acceptable for the middlebox.
   In such a case, the middlebox returns a negative reply of type
   'transport protocol problem' (0x0325) and closes the connection. The
   middlebox returns a negative reply of type 'security of underlying
   protocol layers insufficient' (0x0326) and closes the connection, if
   the security properties of the network connection do not match the
   middlebox's requirements.

   Processing of an SE request message starts with checking the major
   and minor protocol version number in the protocol version attribute.
   If the middlebox does not support the specified version number, then
   the middlebox returns a negative reply message of type 'protocol
   version mismatch' (0x0322) with the protocol version attribute
   indicating a version number that is supported by the middlebox.
   After sending this reply, the middlebox closes the connection.

   If the agent is already sufficiently authenticated by means of the
   underlying network connection (for instance, IPsec or TLS), then the
   middlebox checks whether the agent is authorized to configure the
   middlebox. If it is not, the middlebox returns a negative reply of
   type 'no authorization' (0x0324) and closes the connection.

   A positive reply on the SE request may be of sub-type SE or SA. An
   SE request is sent after both parties sufficiently authenticate and
   authorize each other. An SA reply message is sent if explicit
   authentication is requested by any party. The agent requests
   explicit authentication by adding an authentication challenge
   attribute to the SE request message. The middlebox requests explicit

authentication by returning an SA reply message with an
authentication challenge attribute to the agent. If both parties
request explicit authentication, then the SA reply message contains
both an authentication challenge attribute for the agent and an
authentication token attribute authenticating the middlebox.

If the SE request message contains an authentication challenge
attribute, then the middlebox checks if it can authenticate itself.
If yes, it adds a corresponding authentication token attribute to the
SA reply. If it cannot authenticate based on the authentication
challenge attribute, it adds an authentication token attribute to the
SA reply message with a value field of length zero.

If the middlebox wants the agent to explicitly authenticate itself,
then the middlebox creates an authentication challenge attribute for
the agent and adds it to the SA reply message.

If the middlebox replies to the SE request message with an SA reply
message, then the session state changes from CLOSED to NO_AUTH.

If the SE request message did not contain an authentication challenge
attribute and if the middlebox does not request the agent to
explicitly authenticate itself, then the middlebox sends an SE reply
message in response to the SE request message. This implies that the
session state changes from CLOSED to OPEN.

The SE reply message contains a capabilities attribute describing the
middlebox capabilities.

7.3. Processing SA Requests

The SA request is only applicable if the session is in state NOAUTH.
If a session is in state CLOSED or OPEN, then the middlebox sends a
negative reply message of type 'request not applicable' (0x0320) to
the agent. The state of the session remains unchanged.

After receiving an SA request message in state NOAUTH, the middlebox
checks if the agent is sufficiently authenticated. Authentication
may be based on an authentication token attribute that is optionally
contained in the SA request message. If the agent is not
sufficiently authenticated, then the middlebox returns a negative
reply of type 'authentication failed' (0x0323) and closes the
connection.

If authentication of the agent is successful, the middlebox checks if
the agent is authorized to configure the middlebox. If not, the
middlebox returns a negative reply of type 'no authorization'
(0x0324) and closes the connection.

   If authorization is successful, then the session state changes from
   NOAUTH to OPEN, and the agent returns an SE reply message that
   concludes session setup. The middlebox states its capabilities in
   the capability attribute contained in the SE reply message.

7.4. Processing ST Requests

   The ST request is only applicable if the session is in state NOAUTH
   or OPEN. If a session is in state CLOSED, then the middlebox sends a
   negative reply message of type 'request not applicable' (0x0320) to
   the agent. The state of the session remains unchanged.

   The middlebox always replies to a correct ST request with a positive
   ST reply. The state of the session changes from OPEN or from NOAUTH
   to CLOSED. After sending the ST reply, the middlebox closes the
   connection. Requests received after receiving the ST request and
   before closing the connection are ignored by the middlebox.

7.5. Generating AST Notifications

   At any time, the middlebox may terminate an established session and
   change the session state from OPEN or from NOAUTH to CLOSED. Session
   termination is indicated to the agent by sending an AST notification.

   Before sending the notification, the middlebox ensures that for all
   requests that have been processed, according replies are returned to
   the agent, such that the agent exactly knows the state of the
   middlebox at the time of session termination. After sending the AST
   notification, the middlebox sends no more messages to the agent, and
   it closes the connection.

7.6. Session Termination by Interruption of Connection

   Section 2.2.4 of [RFC3989] describes the session behavior when the
   network connection is interrupted. The behavior is defined for the
   middlebox (i.e., the SIMCO server) only and does not consider the
   behavior of the SIMCO agent in such an event.

   If the SIMCO agent detects an interruption of the underlying network
   connection, it can terminate the session. The detection of the
   interrupted network connection can be done by several means, for
   instance, feedback of the operating system or a connection timeout.
   The definition of this detection mechanism is out of the scope of
   this memo.

8. Policy Rule Control Message Processing

    For policy rule control and monitoring, the agent can send the PRR,
    PER, PEA, PLC, PRS, and PRL requests. The middlebox then sends a
    single reply message per request message back to the agent.
    Additionally, the middlebox may send unsolicited ARE notifications at
    any time.

    The transaction semantics of policy rule control messages is
    explained in detail in [RFC3989], Section 2.3.

    For examples about protocol operation, see Section 4 of [RFC3989].

8.1. Policy Rule State Machine

    Policy rules are established by successful PRR, PEA, or PER
    transactions. Each time a policy rule is created, an unused policy
    rule identifier (PID) is assigned to the new policy rule. For each
    policy rule identifier, a state machine exists at the middlebox. The
    state machine is illustrated by the figure below.

```
                      PRR/PRR          +---------------+
           +----+     +----------------+  PID UNUSED   |<-+
           |    |     |                +---------------+  |
           |    v     v      PLC(lt=0)/ ^   |              |
           | +-------------+     PRD    |   | PER/PER    | ARE(lt=0)
           | |  RESERVED   +-----------+    |            | PLC(lt=0)/
           | +-+----+------+  ARE(lt=0)     v            |    PRD
           | |    | |               +---------------+    |
           +----+     +--------------->|   ENABLED    +--+
         PLC(lt>0)/    PEA/PER        +-+-------------+
            PLC                        |        ^
                                      +-----------+
            lt = lifetime          PLC(lt>0)/PLC
```
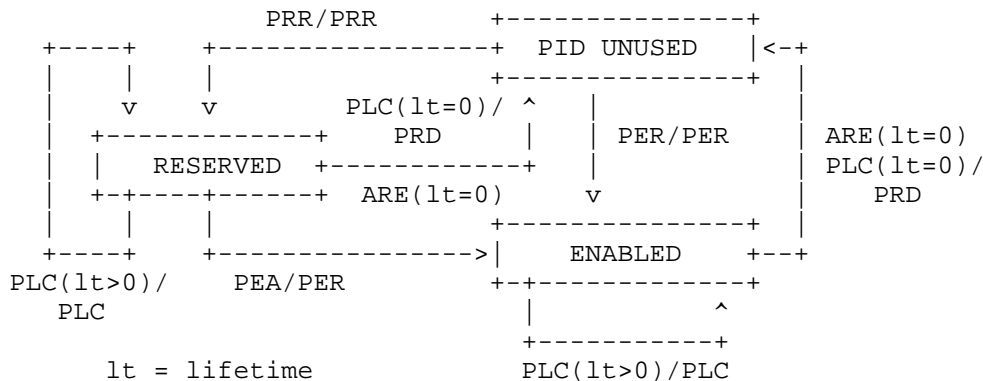
                  Figure 42: Policy rule state machine

    The figure illustrates all possible state transitions of a PID and
    its associated policy. Successful configuration request transactions
    (PER, PRR, PEA, PLC) are denoted by a descriptor of the request
    message, a '/' symbol, and a descriptor of the reply message. Failed
    configuration request transactions are not displayed, because they do
    not change the PID state. Notification transactions are denoted just
    by the a notification descriptor. For example, a successful PRR
    request transaction is denoted by 'PRR/PRR', and an ARE notification

is denoted by 'ARE'. For PLC request transactions, the descriptor
for the request message is extended by an indication of the value of
the lifetime parameter contained in the message.

A successful PRR transaction (PRR/PRR) picks a PID in state UNUSED
and changes the state to RESERVED. A successful PER transitions
picks a PID in state UNUSED and changes the state to ENABLED. A PID
in state RESERVED is changed to ENABLED by a successful PEA
transaction. In state RESERVED or UNUSED, a successful PLC
transaction with a lifetime parameter greater than zero does not
change the PID's state. A successful PLC transaction with a lifetime
parameter equal to zero changes the state of a PID from RESERVED to
UNUSED or from ENABLED to UNUSED.

A failed request transaction does not change state at the middlebox.

An ARE notification transaction with the lifetime attribute set to
zero has the same effect as a successful PLC transaction with a
lifetime parameter equal to zero.

8.2. Processing PRR Requests

Processing PRR requests is much simpler on pure firewalls than on
middleboxes with NAT functions. Therefore, this section has three
sub-sections: The first one describes initial checks that are
performed in any case. The second sub-section describes processing
of PRR requests on pure firewalls, and the third one describes
processing on all devices with NAT functions.

8.2.1. Initial Checks

When a middlebox receives a PRR request message, it first checks if
the authenticated agent is authorized for requesting reservations.
If not, it returns a negative reply message of type 'agent not
authorized for this transaction' (0x0341).

If the request contains the optional group identifier, then the
middlebox checks if the group already exists. If not, the middlebox
returns a negative reply message of type 'specified policy rule group
does not exist' (0x0344).

If the request contains the optional group identifier, then the
middlebox checks if the authenticated agent is authorized for adding
members to this group. If not, the middlebox returns a negative
reply message of type 'not authorized for accessing specified group'
(0x0346).

The middlebox may then check the PRR parameter set. A negative reply
of type 'IP version mismatch' (0x034F) is returned if the IPi field
does not match the inside IP version of the address at the middlebox.
A negative reply of type 'IP version mismatch' (0x034F) is returned
if the IPo field does not match the outside IP version of the address
at the middlebox. The requested transport protocol type is checked,
and a negative reply of type 'protocol type not supported' (0x0354)
is returned if it is not supported. The middlebox may return a
negative reply of type 'requested address space not available'
(0x0347) if the requested address space is completely blocked or not
supported by the middlebox in any way; for example, if a UDP port
number is requested and all UDP packets are blocked by a middlebox
acting as firewall.

The latter check at the middlebox is optional. If the check would
fail and is not performed at this transaction, then two superfluous
transactions will follow. First, the agent will send a request
message for a corresponding PER transaction and will receive a
negative reply on this. Second, either the agent will send a
corresponding PLC request message with lifetime set to zero in order
to delete the reservation, or the reservation will time out and the
middlebox will send an ARE notification message with the lifetime
attribute set to zero. Both transactions can be avoided if the
middlebox initially performs this check.

A reason for avoiding this check might be its complexity. If the
check is passed, the same check will have to be performed again for a
subsequent corresponding PEA request. If processing two more
transactions is considered to consume less resources than performing
the check twice, it might be desirable not to perform it during the
PRR transaction.

After checking the PRR parameter set, the middlebox chooses a
lifetime value for the new policy rule to be created, which is
greater than or equal to zero and less than or equal to the minimum
of the requested value and the maximum lifetime specified by the
middlebox capabilities attribute at session setup. Formally, the
lifetime is chosen such that

        0 <= lt_granted <= MINIMUM(lt_requested, lt_maximum)

holds, where 'lt_granted' is the actual lifetime chosen by the
middlebox, 'lt_requested' is the lifetime requested by the agent, and
'lt_maximum' is the maximum lifetime specified during capability
exchange at session setup.

   If there are further sessions in state OPEN with authenticated agents
   authorized to access the policy rule, then to each of these agents a
   corresponding ARE notification with lifetime set to lt_granted is
   sent.

   If the chosen lifetime is zero, the middlebox sends a negative reply
   of type 'middlebox configuration failed' (0x034A) to the agent.

8.2.2. Processing on Pure Firewalls

   If the middlebox is configured as a pure firewall, then it accepts
   the request after the initial checks. It establishes a new policy
   reserve rule and assigns to it a policy rule identifier in state
   RESERVED. It generates a positive PRR reply and sets the attributes
   as specified below. No configuration of the firewall function is
   required.

   The identifier chosen for the new policy rule is reported in the
   policy rule identifier attribute of the PRR reply.

   If a group identifier attribute is contained in the PRR request, then
   the middlebox adds the new policy rule to the members of this group.
   If the PRR request does not contain a group identifier attribute,
   then the middlebox creates a new group with the new policy rule as
   the only member. In any case, the middlebox reports the group of
   which the new policy rule is a member in the group identifier
   attribute of the PRR reply.

   The chosen lifetime is reported in the lifetime attribute of the PRR
   reply.

   In the address tuple (outside) attribute of the PRR reply, the first
   parameter field is set to 'protocols only' (0x1). Consequently, the
   attribute has a length of 32 bits. The IP version parameter field is
   set according to the IPo parameter field in the PRR parameter set
   attribute of the PRR request message. The prefix length parameter
   field is set to 0x00, and the transport protocol parameter field in
   the address tuple (outside) attribute of the PRR reply is set
   identically to the transport protocol attribute in the PRR parameter
   set attribute of the PRR request message. The location parameter
   field is set to 'outside' (0x02).

8.2.3. Processing on Network Address Translators

   If the middlebox is configured as a Network Address Translator (NAT),
   then it tries to reserve a NAT binding.

   The middlebox first checks the PRR parameter set further if the NM
   (NAT mode) parameter matches its configuration. A negative reply of
   type 'NAT mode not supported' (0x034E) is returned by the middlebox
   if the configuration is not matched.

   The following actions are performed, depending on the middlebox NAT
   type:

     - traditional NAT
       A NAT binding at the outside (A2) with the requested transport
       protocol, external IP version, port range, and port parity is
       reserved.

     - twice NAT
       A NAT binding at the outside (A2) with the requested transport
       protocol, external IP version, port range, and port parity is
       reserved. Furthermore, the middlebox reserves an inside (A1) NAT
       binding with the requested transport protocol, internal IP
       version, port range, and port parity.

   The identifier chosen for the new policy rule is reported in the
   policy rule identifier attribute of the PRR reply.

   After the checks are successfully performed, the middlebox
   establishes a new policy reserve rule, with the requested PRR
   parameter set, and assigns to it a policy rule identifier in state
   RESERVED. It generates a positive PRR reply and sets the attributes
   as specified below.

   If a group identifier attribute is contained in the PRR request, then
   the middlebox adds the new policy rule to the members of this group.
   If the PRR request does not contain a group identifier attribute,
   then the middlebox creates a new group with the new policy rule as
   the only member. In any case, the middlebox reports the group of
   which the new policy rule is a member in the group identifier
   attribute of the PRR reply.

   The chosen lifetime is reported in the lifetime attribute of the PRR
   reply.

   In the address tuple (outside) attribute of the PRR reply, the first
   parameter field is set to 'full addresses' (0x0). The location
   parameter field is set to 'outside' (0x02). The IP version parameter

field is set according to the IPo parameter field in the PRR
parameter set attribute of the PRR request message. For IPv4
addresses, the prefix length field is set to 0x20 to indicate a full
address, and the reserved outside IPv4 address is set in the address
field. For IPv6 addresses, the prefix length field is set to 0x80 to
indicate a full address, and the reserved outside IPv6 address is set
in the address field. The transport protocol parameter field in the
address tuple (outside) attribute of the PRR reply is set identically
to the transport protocol attribute in the PRR parameter set
attribute of the PRR request message. The reserved outside base port
number (i.e., the lowest port number of the allocated range) is
stored in the port number parameter field, and the allocated port
range is stored in the port range parameter field.

If the NM (NAT mode) parameter in the PRR parameter set attribute of
the PRR request message has the value 'traditional', then the PRR
reply message does not contain an address tuple (inside) attribute.
If otherwise (it has the value 'twice'), then the PRR reply message
contains an address tuple (inside) attribute. In the address tuple
(inside) attribute of the PRR reply, the first parameter field is set
to 'full addresses' (0x0). The location parameter field is set to
'inside' (0x01). The IP version parameter field is set according to
the IPi parameter field in the PRR parameter set attribute of the PRR
request message. For IPv4 addresses, the prefix length field is set
to 0x20 to indicate a full address, and the reserved inside IPv4
address is set in the address field. For IPv6 addresses, the prefix
length field is set to 0x80 to indicate a full address, and the
reserved inside IPv6 address is set in the address field. The
transport protocol parameter field in the address tuple (inside)
attribute of the PRR reply is set identically to the transport
protocol attribute in the PRR parameter set attribute of the PRR
request message. The reserved inside base port number (i.e., the
lowest port number of the allocated range) is stored in the port
number parameter field, and the allocated port range is stored in the
port range parameter field.

8.3. Processing PER Requests

Processing PER requests is much simpler on pure firewalls than on
middleboxes with NAT functions. Therefore, this section has three
sub-sections: The first one describes initial checks that are
performed in any case. The second sub-section describes processing
of PER requests on pure firewalls, and the third one describes
processing on all devices with NAT functions.

8.3.1. Initial Checks

   When a middlebox receives a PER request message, it first checks if
   the authenticated agent is authorized for requesting middlebox
   configurations for enabling communication. If not, it returns a
   negative reply message of type 'agent not authorized for this
   transaction' (0x0341).

   If the request contains the optional group identifier, then the
   middlebox checks if the group already exists. If not, the middlebox
   returns a negative reply message of type 'specified policy rule group
   does not exist' (0x0344).

   If the request contains the optional group identifier, then the
   middlebox checks if the authenticated agent is authorized for adding
   members to this group. If not, the middlebox returns a negative
   reply message of type 'not authorized for accessing specified group'
   (0x0346).

   Then the middlebox checks the contained address tuple attributes.

   If the first one does not have the location parameter field set to
   'internal' (0x00), or if the second one does not have the location
   parameter field set to 'external' (0x03), then the middlebox returns
   a negative reply message of type 'inconsistent request' (0x034B).

   If the transport protocol parameter field does not have the same
   value in both address tuple attributes, then the middlebox returns a
   negative reply message of type 'inconsistent request' (0x034B).

   If both address tuple attributes contain a port range parameter
   field, if both port range parameter fields have values not equal to
   0xFFFF, and if the values of both port range parameter fields are
   different, then the middlebox returns a negative reply message of
   type 'inconsistent request' (0x034B).

   Then the agent checks if wildcarding is requested and if the
   requested wildcarding is supported by the middlebox. Wildcarding
   support may be different for internal address tuples and external
   address tuples. The following parameter fields of the address tuple
   attribute can indicate wildcarding:

     - the first parameter field
       If it is set to 'protocols only' (0x1), then IP addresses and
       port numbers are completely wildcarded.

      - the transport protocol field
        If it is set to 0x00, then the transport protocol is completely
        wildcarded. Please note that a completely wildcarded transport
        protocol might still support only a limited set of transport
        protocols according to the capabilities of the middlebox. For
        example, a typical NAT implementation may apply transport
        wildcarding to UDP and TCP transport only. Wildcarding the
        transport protocol implies wildcarding of port numbers. If this
        field is set to 0x00, then the values of the port number field
        and the port range field are irrelevant.

      - the prefix length field
        If the IP version number field indicates IPv4 and the value of
        this field is less than 0x20, then IP addresses are wildcarding
        according to this prefix length. If the IP version number field
        indicates IPv6 and the value of this field is less than 0x80,
        then IP addresses are wildcarding according to this prefix
        length. If the first parameter field is set to 'protocols only'
        (0x1), then the value of the prefix length field is irrelevant.

      - the port number field
        If it is set to zero, then port numbers are completely
        wildcarded. In this case, the value of the port range field is
        irrelevant.

   If any of these kinds of wildcarding is used, and if this is in
   conflict with wildcarding support for internal or external addresses
   of the middlebox, then the middlebox returns a negative reply message
   of type 'requested wildcarding not supported' (0x034C).

   Please note that the port range field cannot be used for wildcarding.
   If it is set to a value greater than one, then middlebox
   configuration is requested for all port numbers in the interval
   starting with the specified port number and containing as many
   consecutive port numbers as specified by the parameter.

   If the direction parameter field in the PER parameter set attribute
   has the value 'bi-directional', then only transport protocol
   wildcarding is allowed. If any other kind of wildcarding is
   specified in one or both of the IP address tuple attributes, then the
   middlebox returns a negative reply message of type 'inconsistent
   request' (0x034B).

   If the PER request conflicts with any policy disable rule (see
   Section 8.8.1), then the middlebox returns a negative reply message
   of type 'conflict with existing rule' (0x0350).

After checking the address tuple attributes, the middlebox chooses a
lifetime value for the new policy rule to be created, which is
greater than or equal to zero and less than or equal to the minimum
of the requested value and the maximum lifetime specified by the
middlebox capabilities attribute at session setup. Formally, the
lifetime is chosen such that

$$0 <= lt\_granted <= MINIMUM(lt\_requested, lt\_maximum)$$

holds, where 'lt_granted' is the actual lifetime chosen by the
middlebox, 'lt_requested' is the lifetime requested by the agent, and
'lt_maximum' is the maximum lifetime specified during capability
exchange at session setup.

If there are further sessions in state OPEN with authenticated agents
authorized to access the policy rule, then to each of these agents a
corresponding ARE notification with lifetime set to lt_granted is
sent.

If the chosen lifetime is zero, the middlebox sends a negative reply
of type 'middlebox configuration failed' (0x034A) to the agent.

8.3.2. Processing on Pure Firewalls

If the middlebox is acting as a pure firewall, then it tries to
configure the requested pinhole. The firewall configuration ignores
the port parity parameter field in the PER parameter set attribute,
but it considers the direction parameter field in this attribute.
The pinhole is configured such that communication between the
specified internal and external address tuples is enabled in the
specified direction and covering the specified wildcarding. If the
configuration fails (for example, because the pinhole would conflict
with high-level firewall policies), then the middlebox returns a
negative reply message of type 'middlebox configuration failed'
(0x034A).

If the configuration was successful, the middlebox establishes a new
policy enable rule and assigns to it a policy rule identifier in
state ENABLED. It generates a positive PER reply and sets the
attributes as specified below.

The identifier chosen for the new policy rule is reported in the
policy rule identifier attribute of the PER reply.

If a group identifier attribute is contained in the PER request, then
the middlebox adds the new policy rule to the members of this group.
If the PRR request does not contain a group identifier attribute,
then the middlebox creates a new group with the new policy rule as

the only member. In any case, the middlebox reports the group of
which the new policy rule is a member in the group identifier
attribute of the PER reply.

The chosen lifetime is reported in the lifetime attribute of the PER
reply.

The address tuple (internal) attribute of the PER request is reported
as address tuple (outside) attribute of the PER reply. The address
tuple (external) attribute of the PER request is reported as address
tuple (inside) attribute of the PER reply.

8.3.3. Processing on Network Address Translators

If the middlebox is configured as a NAT, then it tries to configure
the requested NAT binding. The actions taken by the NAT are quite
similar to the actions of the Policy Reserve Rule (PRR) request, but
in the PER request a NAT binding is enabled.

The following actions are performed, depending on the middlebox NAT
type:

   - traditional NAT
     A NAT binding is established between the internal and external
     address tuple with the requested transport protocol, port range,
     direction, and port parity. The outside address tuple is
     created.

   - twice NAT
     A NAT binding is established between the internal and external
     address tuple with the requested transport protocol, port range,
     and port parity. But two address tuples are created: an outside
     address tuple and an inside address tuple.

Should the configuration fail in either NAT case, a negative reply
'middlebox configuration failed' (0x034A) is returned.

If the configuration was successful, the middlebox establishes a new
policy enable rule and assigns to it a policy rule identifier in
state ENABLED. It generates a positive PER reply and sets the
attributes as specified below.

The identifier chosen for the new policy rule is reported in the
policy rule identifier attribute of the PER reply.

If a group identifier attribute is contained in the PER request, then
the middlebox adds the new policy rule to the members of this group.
If the PRR request does not contain a group identifier attribute,

then the middlebox creates a new group with the new policy rule as
the only member. In any case, the middlebox reports the group of
which the new policy rule is a member in the group identifier
attribute of the PER reply.

The chosen lifetime is reported in the lifetime attribute of the PER
reply.

In the address tuple (outside) attribute of the PER reply, the first
parameter field is set to 'full addresses' (0x0). The location
parameter field is set to 'outside' (0x02). The IP version parameter
field is set according to the IP version parameter field in the PER
parameter set attribute of the PER request message. For IPv4
addresses, the prefix length field is set to 0x20 to indicate a full
address, and the reserved outside IPv4 address is set in the address
field. For IPv6 addresses, the prefix length field is set to 0x80 to
indicate a full address, and the reserved outside IPv6 address is set
in the address field. The transport protocol parameter field in the
address tuple (outside) attribute of the PER reply is set identically
to the transport protocol attribute in the PER parameter set
attribute of the PER request message. The reserved outside base port
number (i.e., the lowest port number of the allocated range) is
stored in the port number parameter field, and the allocated port
range is stored in the port range parameter field.

The address tuple (inside) is only returned if the middlebox is a
twice NAT; otherwise, it is omitted. In the address tuple (inside)
attribute of the PER reply, the first parameter field is set to 'full
addresses' (0x0). The location parameter field is set to 'inside'
(0x01). The IP version parameter field is set according to the IP
version parameter field in the PER parameter set attribute of the PER
request message. For IPv4 addresses, the prefix length field is set
to 0x20 to indicate a full address, and the reserved inside IPv4
address is set in the address field. For IPv6 addresses, the prefix
length field is set to 0x80 to indicate a full address, and the
reserved inside IPv6 address is set in the address field. The
transport protocol parameter field in the address tuple (inside)
attribute of the PER reply is set identically to the transport
protocol attribute in the PER parameter set attribute of the PER
request message. The reserved inside base port number (i.e., the
lowest port number of the allocated range) is stored in the port
number parameter field, and the allocated port range is stored in the
port range parameter field.

8.3.4. Processing on Combined Firewalls and NATs

   Middleboxes that are combinations of firewalls and NATs are
   configured in such a way that first the NAT bindings are configured
   and afterwards the firewall pinholes. This sequence is needed since
   the firewall rules must be configured according to the outside
   address tuples and for twice NATs the inside address tuples as well.
   This aspect of middlebox operation may be irrelevant to SIMCO, since
   some NATs already do firewall configuration on their own.

8.4. Processing PEA Requests

   Processing PEA requests is much simpler on pure firewalls than on
   middleboxes with NAT functions. Therefore, this section has three
   sub-sections: The first one describes initial checks that are
   performed in any case. The second sub-section describes processing
   of PEA requests on pure firewalls, and the third one describes
   processing on all devices with NAT functions.

8.4.1. Initial Checks

   When a middlebox receives a PEA request message, it first checks if
   the authenticated agent is authorized for requesting middlebox
   configurations for enabling communication. If not, it returns a
   negative reply message of type 'agent not authorized for this
   transaction' (0x0341).

   Then the middlebox checks the policy rule identifier attribute
   contained in the PEA message. If no policy rule with this identifier
   exists, then the middlebox returns a negative reply message of type
   'specified policy rule does not exist' (0x0343). If there exists a
   policy with this identifier and if it is in a state other than
   RESERVED, then the middlebox returns a negative reply message of type
   'inconsistent request' (0x034B).

   If a policy rule with this identifier exists, but the authenticated
   agent is not authorized for terminating this policy reserve rule,
   then the middlebox returns a negative reply message of type 'agent
   not authorized for accessing this policy' (0x0345).

   Then the middlebox checks the contained address tuple attributes.

   If the first one does not have the location parameter field set to
   'internal' (0x00) or if the second one does not have the location
   parameter field set to 'external' (0x03), then the middlebox returns
   a negative reply message of type 'inconsistent request' (0x034B).

If the transport protocol parameter field does not have the same
value in both address tuple attributes, then the middlebox returns a
negative reply message of type 'inconsistent request' (0x034B).

If both address tuple attributes contain a port range parameter
field, if both port range parameter fields have values not equal to
0xFFFF, and if the values of both port range parameter fields are
different, then the middlebox returns a negative reply message of
type 'inconsistent request' (0x034B).

Then the agent checks if wildcarding is requested and if the
requested wildcarding is supported by the middlebox. Wildcarding
support may be different for internal address tuples and external
address tuples. The following parameter fields of the address tuple
attribute can indicate wildcarding:

   - the first parameter field
     If it is set to 'protocols only' (0x1), then IP addresses and
     port numbers are completely wildcarded.

   - the transport protocol field
     If it is set to 0x00, then IP the transport protocol is
     completely wildcarded. Please note that a completely wildcarded
     transport protocol might still support only a limited set of
     transport protocols according to the capabilities of the
     middlebox. For example, a typical NAT implementation may apply
     transport wildcarding to UDP and TCP transport only.

   - the prefix length field
     If the IP version number field indicates IPv4 and the value of
     this field is less than 0x20, then IP addresses are wildcarding
     according to this prefix length. If the IP version number field
     indicates IPv6 and the value of this field is less than 0x80,
     then IP addresses are wildcarding according to this prefix
     length. If the first parameter field is set to 'protocols only'
     (0x1), then the value of the prefix length field is irrelevant.

   - the port number field
     If it is set to zero, then port numbers are completely
     wildcarded.

   - the port range field
     If it is set to a value greater than one, then port numbers are
     wildcarded within an interval starting with the specified port
     number and containing as many consecutive port numbers as
     specified by the parameter.

If any of these kinds of wildcarding is used, and if this is in
conflict with wildcarding support for internal or external addresses
of the middlebox, then the middlebox returns a negative reply message
of type 'requested wildcarding not supported' (0x034C).

If the PEA request conflicts with any policy disable rule (see
Section 8.8.1), then the middlebox returns a negative reply message
of type 'conflict with existing rule' (0x0350).

After checking the address tuple attributes, the middlebox chooses a
lifetime value for the new policy enable rule to be created, which is
greater than or equal to zero and less than or equal to the minimum
of the requested value and the maximum lifetime specified by the
middlebox capabilities attribute at session setup. Formally, the
lifetime is chosen such that

        0 <= lt_granted <= MINIMUM(lt_requested, lt_maximum)

holds, where 'lt_granted' is the actual lifetime chosen by the
middlebox, 'lt_requested' is the lifetime requested by the agent, and
'lt_maximum' is the maximum lifetime specified during capability
exchange at session setup.

If there are further sessions in state OPEN with authenticated agents
authorized to access the policy rule, then to each of these agents a
corresponding ARE notification with lifetime set to lt_granted is
sent.

If the chosen lifetime is zero, the middlebox sends a negative reply
of type 'middlebox configuration failed' (0x034A) to the agent.

8.4.2. Processing on Pure Firewalls

If the middlebox is configured as a pure firewall, then it tries to
configure the requested pinhole. The firewall configuration ignores
the port parity parameter field in the PER parameter set attribute,
but it considers the direction parameter field in this attribute.
The pinhole is configured such that communication between the
specified internal and external address tuples is enabled in the
specified direction and covering the specified wildcarding. If the
configuration fails, then the middlebox returns a negative reply
message of type 'middlebox configuration failed' (0x034A).

If the configuration was successful, the middlebox replaces the
policy reserve rule referenced by the policy rule identifier
attribute in the PEA request message with a new policy enable rule.
The policy enable rule re-uses the policy rule identifier of the
replaced policy reserve rule. The state of the policy rule

   identifier changes from RESERVED to ENABLED. The policy reserve rule
   is a member of the same group as the replaced policy reserve rule
   was.

   Then the middlebox generates a positive PER reply and sets the
   attributes as specified below.

   The identifier chosen for the new policy rule is reported in the
   policy rule identifier attribute of the PER reply.

   The group identifier is reported in the group identifier attribute of
   the PER reply.

   The chosen lifetime is reported in the lifetime attribute of the PER
   reply.

   The address tuple (internal) attribute of the PER request is reported
   as the address tuple (outside) attribute of the PER reply. The
   address tuple (external) attribute of the PER request is reported as
   the address tuple (inside) attribute of the PER reply.

8.4.3. Processing on Network Address Translators

   If the middlebox is configured as a NAT, then it tries to configure
   the requested NAT binding, i.e., enabling the already reserved
   binding. The already reserved NAT binding from the PRR request is
   now enabled in the middlebox.

   If the enable configuration was successful, the middlebox replaces
   the policy reserve rule referenced by the policy rule identifier
   attribute in the PEA request message with a new policy enable rule.
   The policy enable rule re-uses the policy rule identifier of the
   replaced policy reserve rule. The state of the policy rule
   identifier changes from RESERVED to ENABLED. The policy reserve rule
   is a member of the same group as the replaced policy reserve rule
   was.

   Then the middlebox generates a positive PER reply and sets the
   attributes as specified below.

   The reserved outside address tuple is reported as the address tuple
   (outside) attribute of the PER reply. The reserved inside address
   tuple is reported as the address tuple (inside) attribute of the PER
   reply. Both reserved outside and inside address tuples are taken
   from the reserve policy rule generated during the PRR transaction.

8.5. Processing PLC Requests

   When a middlebox receives a PLC request message, it first checks if
   the authenticated agent is authorized for requesting policy rule
   lifetime changes. If not, it returns a negative reply message of
   type 'agent not authorized for this transaction' (0x0341).

   Then the middlebox checks the policy rule identifier attribute
   contained in the PLC message. If no policy rule with this identifier
   exists, then the middlebox returns a negative reply message of type
   'specified policy rule does not exist' (0x0343).

   If a policy rule with this identifier exists, but the authenticated
   agent is not authorized for changing the lifetime of this policy
   rule, then the middlebox returns a negative reply message of type
   'agent not authorized for accessing this policy' (0x0345).

   Then the middlebox chooses a lifetime value for the new policy rule,
   which is greater than zero and less than or equal to the minimum of
   the requested value and the maximum lifetime specified by the
   middlebox capabilities attribute at session setup. Formally, the
   lifetime is chosen such that

        0 <= lt_granted <= MINIMUM(lt_requested, lt_maximum)

   holds, where 'lt_granted' is the actual lifetime chosen by the
   middlebox, 'lt_requested' is the lifetime requested by the agent, and
   'lt_maximum' is the maximum lifetime specified during capability
   exchange at session setup. This procedure implies that the chosen
   lifetime is zero if the requested lifetime is zero.

   If the chosen lifetime is greater than zero, the middlebox changes
   the lifetime of the policy rule to the chosen value and generates a
   PLC reply message. The chosen lifetime is reported in the lifetime
   attribute of the message.

   If otherwise (the chosen lifetime is zero), then the middlebox
   terminates the policy rule and changes the PID state from ENABLED or
   RESERVED, respectively, to UNUSED.

   The middlebox generates a PRD reply message and sends it to the
   requesting agent. If there are further sessions in state OPEN with
   authenticated agents authorized to access the policy rule, then to
   each of these agents a corresponding ARE notification with lifetime
   set to zero is sent.

8.6. Processing PRS Requests

   When a middlebox receives a PRS request message, it first checks if
   the authenticated agent is authorized for receiving policy status
   information. If not, it returns a negative reply message of type
   'agent not authorized for this transaction' (0x0341).

   Then the middlebox checks the policy rule identifier attribute
   contained in the PRS message. If no policy rule with this identifier
   exists in state RESERVED or ENABLED, then the middlebox returns a
   negative reply message of type 'specified policy rule does not exist'
   (0x0343).

   If a policy rule with this identifier exists, but the authenticated
   agent is not authorized to receive status information for this policy
   rule, then the middlebox returns a negative reply message of type
   'agent not authorized for accessing this policy' (0x0345).

   If the checks described above are passed, the middlebox accepts the
   requests and generates a reply. If the policy rule for which status
   information is requested is in state RESERVED, then a PRS reply is
   generated and sent to the agent. If otherwise (the policy rule is in
   state ENABLED), then a PES reply is generated and sent to the agent.
   For policy disable rules, a PDS reply is generated and sent to the
   agent.

   In both message formats, the lifetime attribute reports the current
   remaining lifetime of the policy rule, and the owner attribute
   reports the owner of the policy rule for which status information is
   requested.

   The PRS reply message format is identical to the PRR reply message
   format except for an appended owner attribute. In the PRS reply, the
   attributes that are common with the PRR reply (except for the
   lifetime attribute) have exactly the same values as the corresponding
   attributes of the PRR reply that was sent as part of the PRR
   transaction that established the policy reserve rule.

   In the PES reply message, the PER parameter set attribute, the
   address tuple (internal) attribute, and the address tuple (external)
   attribute have exactly the same values as the corresponding
   attributes of the PER or PEA request that were sent as part of the
   corresponding transaction that established the policy enable rule.

In the PES reply message, the policy rule identifier attribute, the
group identifier attribute, the address tuple (inside) attribute, and
the address tuple (outside) attribute have exactly the same values as
the corresponding attributes of the PER reply that was sent as part
of the PER or PEA transaction that established the policy enable
rule.

In the PDS reply message, the policy rule identifier attribute, the
address tuple (internal) attribute, and the address tuple (external)
attribute have exactly the same values as the corresponding
attributes of the PDR request message.

This transaction does not change the state of any policy rule.

8.7. Processing PRL Requests

When a middlebox receives a PRL request message, it first checks if
the authenticated agent is authorized for receiving policy
information. If not, it returns a negative reply message of type
'agent not authorized for this transaction' (0x0341).

Then the middlebox generates a PRL reply message. For each policy
rule at the middlebox in state RESERVED or ENABLED that the
authenticated agent can access, a policy rule identifier attribute is
generated and added to the PRL reply message before the message is
sent to the agent. A negative reply message of type 'reply message
too big' (0x0313) is generated if the number of policy rule
attributes to be returned exceeds the maximum transport unit size of
the underlying network connection or the maximum length of a SIMCO
message. The total size of a SIMCO message is limited to 65,536
octets in total (see Section 4.2 for the SIMCO header).

This transaction does not change the state of any policy rule.

8.8. Processing PDR requests

Processing of PDR requests is structured into five sub-sections. The
first one describes the general extension of the MIDCOM protocol
semantics by PDR. The second sub-section describes the initial
checks that are performed in any case. The third sub-section
describes the processing of PDR requests on pure firewalls. The
fourth one describes processing on devices with NATs, and the fifth
describes processing of devices with combined firewall and NAT
functions.

8.8.1. Extending the MIDCOM semantics

   The Policy Disable Rule (PDR) extends the MIDCOM protocol semantics
   [RFC3989] by another policy rule type. The PDR is intended to be
   used for dynamically blocking unwanted traffic, particularly in case
   of an attack, for example, a distributed denial of service attack.

   PDR requests follow the same ownership concept as all other
   transactions do (see [RFC3989], Section 2.1.5). However, PDR
   prioritization over PERs is independent of ownership. A PDR always
   overrules a conflicting PER, even if the respective owners are
   different. Typically, only a highly privileged agent will be allowed
   to issue PDR requests.

   A PDR rule and PER rule conflict with each other if their address
   tuples overlap such that there exists at least one IP packet that
   matches address tuple A0 of both rules in the internal network and
   that matches address tuple A3 of both rules in the external network.
   Note that the packet may be translated from the internal to external
   network, or vice versa.

   Let's assume, for instance, that a policy enable rule (PER) enables
   all traffic from any external host using any UDP port to a certain
   UDP port of a certain internal host:

        PER A3={ any external IP address,   UDP, any port   }
        PER A0={ internal IP address 10.1.8.3, UDP, port 12345 }

   Then this conflicts with a policy disable rule (PDR) blocking all UDP
   traffic from a potentially attacking host:

        PDR A3={ external IP address 192.0.2.100, UDP, any port }
        PDR A0={ any internal IP address,   UDP, any port }

   If a new PDR is established, then all conflicting PERS are terminated
   immediately. A new PER can only be established if it does not
   conflict with any already existing PDR.

8.8.2. Initial Checks

   When a middlebox receives a PDR request message, it first checks if
   the authenticated agent is authorized for requesting middlebox
   configurations for disabling communication. If not, it returns a
   negative reply message of type 'agent not authorized for this
   transaction' (0x0341).

   Then the middlebox checks the contained address tuple attributes.

If the first one does not have the location parameter field set to 'internal' (0x00), or if the second one does not have the location parameter field set to 'external' (0x03), then the middlebox returns a negative reply message of type 'inconsistent request' (0x034B).

If the transport protocol parameter field does not have the same value in both address tuple attributes, then the middlebox returns a negative reply message of type 'inconsistent request' (0x034B).

If both address tuple attributes contain a port range parameter field, if both port range parameter fields have values not equal to 0xFFFF, and if the values of both port range parameter fields are different, then the middlebox returns a negative reply message of type 'inconsistent request' (0x034B).

Then the agent checks if wildcarding is requested and if the requested wildcarding is supported by the middlebox. Wildcarding support may be different for internal address tuples and external address tuples. The following parameter fields of the address tuple attribute can indicate wildcarding:

  - the first parameter field
    If it is set to 'protocols only' (0x1), then IP addresses and
    port numbers are completely wildcarded.

  - the transport protocol field
    If it is set to 0x00, then the transport protocol is completely
    wildcarded. Please note that a completely wildcarded transport
    protocol might still support only a limited set of transport
    protocols according to the capabilities of the middlebox. For
    example, a typical NAT implementation may apply transport
    wildcarding to UDP and TCP transport only. Wildcarding the
    transport protocol implies wildcarding of port numbers. If this
    field is set to 0x00, then the values of the port number field
    and the port range field are irrelevant.

  - the prefix length field
    If the IP version number field indicates IPv4 and the value of
    this field is less than 0x20, then IP addresses are wildcarding
    according to this prefix length. If the IP version number field
    indicates IPv6 and the value of this field is less than 0x80,
    then IP addresses are wildcarding according to this prefix
    length. If the first parameter field is set to 'protocols only'
    (0x1), then the value of the prefix length field is irrelevant.

       - the port number field
         If it is set to zero, then port numbers are completely
         wildcarded. In this case, the value of the port range field is
         irrelevant.

   If any of these kinds of wildcarding is used, and if this is in
   conflict with wildcarding support for internal or external addresses
   of the middlebox, then the middlebox returns a negative reply message
   of type 'requested wildcarding not supported' (0x034C).

   Please note that the port range field cannot be used for wildcarding.
   If it is set to a value greater than one, then middlebox
   configuration is requested for all port numbers in the interval
   starting with the specified port number and containing as many
   consecutive port numbers as specified by the parameter.

   The specified policy disable rule is activated, and the middlebox
   will terminate any conflicting policy enable rule immediately.
   Conflicts are defined in Section 8.8.1. Agents with open sessions
   that have access to the policy rules to be terminated are notified
   via the ARE notification.

   The middlebox will reject all requests for new policy enable rules
   that conflict with the just established PDR as long as the PDR is not
   terminated. In such a case, a negative 'conflict with existing rule'
   (0x0350) reply will be generated.

   After checking the address tuple attributes, the middlebox chooses a
   lifetime value for the new policy rule to be created, which is
   greater than or equal to zero and less than or equal to the minimum
   of the requested value and the maximum lifetime specified by the
   middlebox capabilities attribute at session setup. Formally, the
   lifetime is chosen such that

        $0 <= lt\_granted <= MINIMUM(lt\_requested, lt\_maximum)$

   holds, where 'lt_granted' is the actual lifetime chosen by the
   middlebox, 'lt_requested' is the lifetime requested by the agent, and
   'lt_maximum' is the maximum lifetime specified during capability
   exchange at session setup.

   If there are further sessions in state OPEN with authenticated agents
   authorized to access the policy rule, then to each of these agents a
   corresponding ARE notification with lifetime set to lt_granted is
   sent.

   If the chosen lifetime is zero, the middlebox sends a negative reply
   of type 'middlebox configuration failed' (0x034A) to the agent.

8.8.3. Processing on Pure Firewalls

   If the middlebox is acting as a pure firewall, then it tries to
   configure the requested disable rule, i.e., configuring a blocking
   rule at the firewall. The disable rule is configured such that
   communication between the specified internal and external address
   tuples is blocked, covering the specified wildcarding. If the
   configuration fails (for example, because the blocking rule would
   conflict with high-level firewall policies), then the middlebox
   returns a negative reply message of type 'middlebox configuration
   failed' (0x034A).

   If the configuration was successful, the middlebox establishes a new
   policy disable rule and assigns to it a policy rule identifier in
   state ENABLED. It generates a positive PDR reply and sets the
   attributes as specified below.

   The identifier chosen for the new policy rule is reported in the
   policy rule identifier attribute of the PDR reply.

   The chosen lifetime is reported in the lifetime attribute of the PDR
   reply.

8.8.4. Processing on Network Address Translators

   If the middlebox is configured as a NAT, then it tries to block the
   specified address tuple in the NAT. The mechanisms used for this
   depend on the implementation and capabilities of the NAT.

   Should the configuration fail in either NAT case, a negative reply
   'middlebox configuration failed' (0x034A) is returned.

   If the configuration was successful, the middlebox establishes a new
   policy disable rule and assigns to it a policy rule identifier in
   state ENABLED. It generates a positive PDR reply and sets the
   attributes as specified below.

   The identifier chosen for the new policy rule is reported in the
   policy rule identifier attribute of the PDR reply.

   The chosen lifetime is reported in the lifetime attribute of the PDR
   reply.

8.8.5. Processing on Combined Firewalls and NATs

   Middleboxes that are combinations of firewall and NAT are configured
   in such a way that first the firewall is configured with the blocking
   rule and afterwards the NAT is configured to block the address tuple.
   This aspect of middlebox operation may be irrelevant to SIMCO, since
   some NATs already do firewall configuration on their own.

8.9 Generating ARE Notifications

   At any time, the middlebox may terminate a policy rule by deleting
   the configuration of the rule and by changing the corresponding PID
   state from ENABLED or from RESERVED, respectively, to UNUSED.

   For each session in state OPEN with authenticated agents authorized
   to access the policy rule, the middlebox generates a corresponding
   ARE notification with the lifetime attribute set to zero and sends it
   to the respective agent. The identifier of the terminated policy
   rule is reported in the policy rule identifier attribute of the ARE
   notification.

   After sending the notification, the middlebox will consider the
   policy rule non-existent. It will not process any further
   transaction on this policy rule.

   In the case of PRR, PER, PEA, and PLC (reserving and enabling policy
   rules and changes of the lifetime), the middlebox generates an ARE
   notification after processing the request. This ARE notification is
   generated for each session in state OPEN with authenticated agents
   (other than the requesting agent) who are authorized to access the
   policy rule. Through this ARE notification all other agents are kept
   synchronized with the latest state of the policy rules.

9. Security Considerations

9.1. Possible Threats to SIMCO

   Middleboxes, such as firewalls and NATs, are usually operated for
   improving the network security and for extending the IP address space
   (note that stand-alone NATs are not considered to improve security;
   see [RFC2663]). The configuration of middleboxes from an external
   entity looks quite counterproductive on the first glimpse, since an
   attacker using this can possibly configure the middlebox in such way
   that no filtering is applied anymore or that NAT bindings are
   configured for malicious use. So the middlebox is not performing the
   intended function anymore. Possible threats to SIMCO are:

      - Man-in-the-middle attack
        A malicious host intercepts messages exchanged between then SIMCO
        agent and middlebox and can change the content of the messages on
        the fly. This man-in-the-middle attack would result, from the
        agent's view, in a proper middlebox configuration, but the
        middlebox would not be configured accordingly. The man in the
        middlebox could open pinholes that compromise the protected
        network's security.

      - Changing content
        The message content could be changed in such a way that the
        requested policy rule configuration is not configured in the
        middlebox, but that any other unwanted configuration could be.
        That way, an attacker can open the firewall for his own traffic.

      - Replaying
        Already sent messages could be re-sent in order to configure the
        middlebox in such a way that hosts could configure policy rules
        without the permission of an application-level gateway or system
        administrator.

      - Wiretapping
        An already configured policy rule could be re-used by other hosts
        if the policy rule is configured with too broad a wildcarding
        (see below). These hosts could send unwanted traffic.

9.2. Securing SIMCO with IPsec

   The previous subsection identifies several issues on security for
   SIMCO. SIMCO can rely on IPsec mechanisms, as defined in [RFC4302]
   and [RFC4303], for ensuring proper operations.

When SIMCO relies on IPsec, it uses IPsec in transport mode with an
authentication header (AH) [RFC4302] and an encapsulating security
payload (ESP) [RFC4303], so that IP traffic between SIMCO agent and
middlebox is protected. The authentication header is used for
protecting the whole packet against content changes and replaying.
The ESP header is used to prevent wiretapping.

At either the agent or middlebox side, the following should be pre-
configured: the IP addresses of the agent or middlebox, TCP (as the
transport protocol), and the port numbers (if possible). Only
packets from the pre-configured address of the agents or middlebox
should be accepted.

The keys for authentication for both the SIMCO agent and middlebox
are pre-configured at each side. For replay protection, the use of a
key management system is recommended. For the Internet Key Exchange
(IKE) protocol, see [RFC4306].

10. IAB Considerations on UNSAF

UNilateral Self-Address Fixing (UNSAF) is described in [RFC3424] as a
process at originating endpoints that attempt to determine or fix the
address (and port) by which they are known to another endpoint.
UNSAF proposals, such as STUN [RFC3489], are considered a general
class of work-arounds for NAT traversal and solutions for scenarios
with no middlebox communication (MIDCOM).

This document describes a protocol implementation of the MIDCOM
semantics and thus implements a middlebox communication (MIDCOM)
solution. MIDCOM is not intended as a short-term work-around, but
more as a long-term solution for middlebox communication. In MIDCOM,
endpoints are not involved in allocating, maintaining, and deleting
addresses and ports at the middlebox. The full control of addresses
and ports at the middlebox is located at the SIMCO server.

Therefore, this document addresses the UNSAF considerations in
[RFC3424] by proposing a long-term alternative solution.

11. Acknowledgements

The authors would like to thank Sebastian Kiesel and Andreas Mueller
for valuable feedback from their SIMCO implementation and Mary Barnes
for a thorough document review.

12. Normative References

   [RFC3989]   Stiemerling, M., Quittek, J., and T. Taylor, "Middlebox
               Communications (MIDCOM) Protocol Semantics", RFC 3989,
               February 2005.

   [RFC4302]   Kent, S., "IP Authentication Header", RFC 4302, December
               2005.

   [RFC4303]   Kent, S., "IP Encapsulating Security Payload (ESP)", RFC
               4303, December 2005.

   [RFC4346]   Dierks, T. and E. Rescorla, "The Transport Layer Security
               (TLS) Protocol Version 1.1", RFC 4346, April 2006.

13. Informative References

   [RFC791]   Postel, J., "Internet Protocol", STD 5, RFC 791,
               September 1981.

   [RFC1519]   Fuller, V., Li, T., Yu, J., and K. Varadhan, "Classless
               Inter-Domain Routing (CIDR): an Address Assignment and
               Aggregation Strategy", RFC 1519, September 1993.

   [RFC2460]   Deering, S. and R. Hinden, "Internet Protocol, Version 6
               (IPv6) Specification", RFC 2460, December 1998.

   [RFC2663]   Srisuresh, P. and M. Holdrege, "IP Network Address
               Translator (NAT) Terminology and Considerations", RFC
               2663, August 1999.

   [RFC3234]   Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and
               Issues", RFC 3234, February 2002.

   [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.

   [RFC3489]   Rosenberg, J., Weinberger, J., Huitema, C., and R. Mahy,
               "STUN - Simple Traversal of User Datagram Protocol (UDP)
               Through Network Address Translators (NATs)", RFC 3489,
               March 2003.

   [RFC3932]   Alvestrand, H., "The IESG and RFC Editor Documents:
                Procedures", BCP 92, RFC 3932, October 2004.

   [RFC4291]   Hinden, R. and S. Deering, "IP Version 6 Addressing
                Architecture", RFC 4291, February 2006.

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

Authors' Addresses

   Martin Stiemerling
   NEC Europe Ltd.
   Network Laboratories Europe
   Kurfuersten-Anlage 36
   69115 Heidelberg
   Germany

   Phone: +49 6221 4342-113
   EMail: stiemerling@netlab.nec.de

   Juergen Quittek
   NEC Europe Ltd.
   Network Laboratories Europe
   Kurfuersten-Anlage 36
   69115 Heidelberg
   Germany

   Phone: +49 6221 4342-115
   EMail: quittek@netlab.nec.de

   Cristian Cadar
   Muelheimer Strasse 23
   40239 Duesseldorf
   Germany

   EMail: ccadar2@yahoo.com