Internet Engineering Task Force
INTERNET-DRAFT
draft-schulzrinne-nsis-casp-01.ps
H. Schulzrinne, H. Tschofenig, X. Fu, A. McDonald
Columbia U./Siemens/U. Goettingen/Siemens-Roke Manor
March 3, 2003
Expires: August 2003

CASP - Cross-Application Signaling Protocol

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026.

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

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

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

To view the list Internet-Draft Shadow Directories, see http://www.ietf.org/shadow.html.

Copyright Notice

Copyright (c) The Internet Society (2003). All Rights Reserved.

Abstract

CASP is a modular potocol for establishing network control state along a data path between two nodes communicating on the Internet.

The signalling problem addressed by CASP is the same as the overall problem being addressed by the NSIS activities.

The CASP framework is defined as a modular protocol, which includes a general purpose messaging layer (M-layer), which supports a number of client layers for particular signalling applications (e.g. QoS, MIDCOM). In addition there is distinct, special purpose client component for next-peer discovery.

Contents

1 Introduction

CASP is a modular potocol for establishing network control state along a data path between two nodes communicating on the Internet.

The signalling problem addressed by CASP is the same as the overall problem being addressed by the NSIS activities, for which a set of requirements are given in [1] and an outline framework in [2].

The CASP framework is defined as a modular protocol, which includes a general purpose messaging layer (M-layer), which supports a number of client layers for particular signalling applications (e.g. QoS, MIDCOM). In addition there is distinct, special purpose client component for next-peer discovery.

The CASP messaging layer is layered over standard transport protocols such as TCP, SCTP and UDP depending on the specific messaging requirements.

The components of CASP can be related to the NSIS framework [2] in the following way: the client layer for a particular signalling application corresponds to an NSLP, such as a QoS NSLP or MIDCOM NSLP. The functionality of the NTLP is provided by the combination of the CASP messaging layer, the discovery client and the underlying transport protocols. It should be noted that the design decision has been made to reuse transport functionality from existing tranport protocols rather than reimplementing it in the CASP M-layer itself. The protocol specification in this document is phrased in terms primarily of CASP client and messaging layers, however this can be easily related to the NSIS components, as shown in Figure 1.

```
             +--------------+
            +--------------+|
NSLP       +--------------+||
           | CASP Clients ||+
           |  (e.g. QoS)  |+
           +--------------+
                   ^
- - - - - - - - -  | - - - - - - - - - - - - - - - - - - - - -
                   v
           +---------------+     +------------------------+
           |     CASP      |<--->|CASP Next-peer Discovery|
           |Messaging Layer|     | Clients (e.g. Scout)   |
NTLP       +---------------+     +------------------------+
             |           |
         +-------------------+
         |Transport Protocols|
         | (e.g. TCP, UDP)   |
         +-------------------+
             ^           ^
- - - - - -  | - - - - - | - - - - - - - - - - - - - - - - - -
             v           v
             +-----------+
             |    IP     |
             +-----------+
```

Figure 1: CASP Framework

2 Terminology

In this document, the key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in RFC 2119 [3] and indicate requirement levels for compliant CASP implementations.

3 Message Delivery

CASP Messaging Layer is not a request-response protocol, but rather used to deliver messages along the path. A CASP client can, however, send back responses that allow the client which sent a message to confirm that the initial message was delivered and to determine whether the operation was successful or encountered an error. This offers end-to-end reliability.

CASP separates signaling message delivery from discovery. Several mechanisms might be used for next-peer discovery including the scout client described in Section 5.2.

Typically, an M-layer session state consists of a session identifier (which is chosen by the initiator to uniquely identify a 'CASP M-session'), a flow identifier (see Section 4), the previous and the next CASP hop (PHOP and NHOP), refresh interval and branch identifiers. Multiple next and previous hops may be maintained for a single CASP M-layer state, differentiated using branch identifiers (see Section 6 for more details).

Not all CASP nodes need to support all client layers. Figure 2 shows an example where not all nodes support the required client (in this case the 'Foo' client). This may occur, for example, where an end-system knows that its first router is CASP aware, but this router does not support all possible clients.

```
     N1                 N2                  N3                  N4
+-----------+                          +-----------+       +-----------+
|Client Foo |                          |Client Foo |       |Client Foo |
+-----------+       +----------+       +-----------+       +-----------+
| M-layer   |<--->| M-layer  |<--->| M-layer   |<--->| M-layer   |
+-----------+       +----------+       +-----------+       +-----------+
```

Figure 2: An Example of CASP Message Delivery

Each CASP node is responsible for passing the signalling message on to its next peer. If it does not already know the address of the next peer, then it must perform a discovery operation by sending a scout request, or using some other mechanism.

CASP signaling messages are addressed peer-to-peer. Usually, a single TCP connection or SCTP association is created between each pair of peers. If a connection to the next peer is already available (whether it is from the same signaling session or another) then it is reused. If a connection is not available, then one is opened.

The CASP messaging layer can also use an unreliable transport service, such as UDP or raw IP packets. Use of unreliable transport protocols is only allowed if the message size is guaranteed not to exceed 512 bytes and if measures are in place to prevent network congestion. UDP is used for the scout discovery client where end-to-end addressing and use of the router alert option is required.

4 CASP Message Formats

CASP messages and their constituent objects are defined in a similar manner to RSVP [4].

A CASP message consists of a common header, followed by a body consisting of a variable number of variable-length objects, which are identified as being of a particular type. The following subsections define the format of the common header, the standard object header, and the construction of CASP messages.

The permissible choices of object types to form a CASP message are specified using Backus-Naur Form (BNF) augmented with square brackets surrounding optional sub-sequences. The BNF implies an order for the objects in a message. However, object order normally makes no logical difference (except for the Length and Common Headers which MUST appear first, when present). Objects SHOULD be sent in the order specified, but the receiver MUST be able to correctly parse CASP messages with objects in any order.

4.1 Length Header

The length header is REQUIRED when messages are being sent on a stream-based transport, such as TCP. If a message-based transport protocol such as SCTP or UDP is being used then it MUST NOT be included in the message.

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|                        Length (bytes)                         |
+---------------+---------------+---------------+---------------+
```

The field in the length header is:

- Length: 32 bits

  The length of the CASP message in bytes. This includes the length of the common header and this length header.

4.2 Common Header

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|R T D U / / / /|      TTD      |   Hop count   |     Type      |
+---------------+---------------+---------------+---------------+
|                                                               |
+                                                               +
|                            Session                            |
+                          Identifier                           +
|                          (16 bytes)                           |
+                                                               +
|                                                               |
+---------------+---------------+---------------+---------------+
```

The fields in the common header are:

- Flags: 8 bits

  Currently four flags are defined:

  – The reverse (R) bit indicates that a node should route in the opposite direction to the data flow.
  – The tear-down (T) bit indicates that this message tears down all CASP M-layer state (and any associated client state) for the CASP M-session. If not set, the message establishes or refreshes M-layer state.
  – The discovery (D) bit requests that the node perform a new discovery operation. If not set, the old next-hop should be used if possible. (This bit cannot be set if R is set, and would usually not be set if the T bit is set).

  - The unsecure (U) (or "tainted") bit indicates that the message has traversed a hop without channel security.

- TTD: 8 bits

  The TTD (Time to Deliver) value is decremented by each CASP hop. If the TTD reaches zero then the message should be discarded.

- Hop count: 8 bits

  The hops value is incremented by each CASP hop.

- Type: 8 bits

  The CASP message type. Currently valid types are:

  - Type 1: CASP Signaling Message
  - Type 2: CASP Scout Request Message
  - Type 3: CASP Scout Response Message

- Session Identifier

  Identifies a signaling application session.

  The session identifier is a 128-bit globally unique value, and SHOULD be a cryptographically random integer.

  The session identifier could alternatively be defined to be a local identifier together with the creator IP address, however, this may lose global uniqueness due to NATs.

  A node MAY use a MAC address in the construction of this value. This has privacy issues, but may be useful in case the device cannot generate 128 bits of random data.

  A random identifier, together with channel protection, makes it easier to securely identify the session owner. It is not, however, a substitute for signed objects or purpose-built keys.

4.3 Object Formats

Each object consists of one or more 32-bit words with a one word header, with the following format:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+---------------+---------------+---------------+---------------+
|         Length (bytes)        |   Class-Num   |     C-Type    |
+---------------+---------------+---------------+---------------+
|                                                               |
//                      (Object contents)                      //
|                                                               |
+---------------+---------------+---------------+---------------+
```

An object header has the following fields:

- Length: 16 bits

  The length field contains the total object length (excluding padding) in bytes, including the object header.

- Class-Num: 8 bits

  Identifies the object class; values of this field are defined in Appendix A. Each object class has a name, which is always capitalised in this document. A CASP implementation must recognize the following classes:

  - FLOW_ID
    Contains information about the flow which should receive a particular client treatment. It is contained at the CASP M-layer to allow policy based forwarding and NAT devices to inspect this object without major effort. It typically contains the IP addresses of the data sender and data receiver, and possibly some additional demultiplexing information (such as protocol type, source and destination ports, SPI or flow label).
  - CASP_TIMEOUT
    Contains the refresh interval for the CASP M-layer state.
  - CLIENT_DATA
    Carries the client data part of the message.
  - ERROR
    Indicates that an M-layer error occurred, and supplies an error code for it.
  - SCOUT_COOKIE_I
    Contains a random nonce generated by the scout initiator.
  - SCOUT_COOKIE_R
    Contains a cookie generated by the scout responder.

- C-Type: 8 bits

  Object type, unique within Class-Num. Values are defined in Appendix A.

Each CASP object MUST be padded to align on a 32-bit (word) boundary, using the minimal number of additional bytes. Up to three zero-valued bytes are added to the end of the data object field field until a word boundary is reached. The length of the padding is not included in the Length field of the object.

4.4 CASP Signaling Messages

CASP Signaling Messages all have the same basic format. The client 'payload' being carried does not affect the format of the message, and has no affect on the behaviour at the CASP M-layer.

The peer-to-peer method for delivering CASP messages along a path is described in Section 3.

The format of a CASP signaling message is as follows:

```
<CASP Signaling Message> ::= [<Length Header>] <Common Header>
                              <FLOW_ID>
                              [<CASP_TIMEOUT>] [<SCOUT_COOKIE_R>]
                              [<CLIENT_DATA> | <ERROR>]
```

The Length Header MUST be included if using a stream-based transport such as TCP. If a message-based transport, such as SCTP, is being used then it MUST NOT be present.

For CASP Signaling Messages the Type field in the Common Header MUST be set to 1.

The SCOUT_COOKIE_R object would only usually be included in the first CASP message after performing a next-node discovery using the CASP scout client. A node receiving a message containing a SCOUT_COOKIE_R object should verify that it is valid (i.e. that it was one sent by this node) and remove it from the CASP Signaling Message before forwarding it. If the cookie is invalid then it should send back a message containing an ERROR object with the value 'COOKIE_ERROR'.

The CASP_TIMEOUT object is optional, and the M-layer state timer defaults to 30 seconds if an explicit timer is not specified.

The C-Type of the CLIENT_DATA object is used to determine which CASP client at the node to pass the message to. If that client type is unsupported at the node, then the M-layer simply passes the message on to the next peer.

At an intermediate CASP node, the basic processing procedure is:

- Look up M-layer state using Flow ID. (If Flow ID is not present, then message is malformed and must be discarded.)
- If M-layer state not found:
  - If R bit set, message is malformed, and must be discarded. Processing is then complete.
  - Otherwise, create M-layer state and record previous peer in it
- If the appropriate client for the given Client Data object is available, pass message up to it
- If R bit is set:
  - Send message on to 'previous' hop. This should use pre-existing transport connection (e.g. TCP or SCTP) if available, otherwise open a new one. Processing is then complete.
- If 'next' hop not recorded in M-layer state:
  - Perform discovery operation
  - Record 'next' hop in M-layer state
- Send message on to 'next' hop (using pre-existing TCP connection if available, otherwise open a new one). Processing is then complete.

4.5 Scout Request Messages

Scout Request Messages are sent when the CASP M-layer wants to determine the 'next-hop'

The format of a CASP Scout Request Message is as follows:

```
<Scout Request Message> ::= <Common Header> <SCOUT_COOKIE_I>
```

The use of scout messages is described in section 5.2.

Since scout messages are always sent using UDP, the Length Header MUST NOT be present.

For CASP Scout Request Messages the Type field in the Common Header MUST be set to 2.

At the present time the other fields of the common header are unused in scout messages. The TTD and Hop count fields become applicable if a capability discovery method is added (see section 10).

4.6 Scout Response Messages

A Scout Response Message is sent after receiving a Scout Request Message. It is addressed to the CASP node which sent the request.

The format of a CASP scout request message is as follows:

```
<Scout Response Message> ::= <Common Header>
                             <SCOUT_COOKIE_I> <SCOUT_COOKIE_R>
```

The use of scout messages is described in section 5.2.

Since scout messages are always sent using UDP, the Length Header MUST NOT be present.

For CASP Scout Response Messages the Type field in the Common Header MUST be set to 3.

As for the Scout Request Message, the other fields in the common header are not processed for the Scout Response Message in the current version of this protocol.

The SCOUT_COOKIE_I object should be copied from the Scout Request Message is being replied to.

On receipt of a scout response, the sender of the scout request MUST verify that the SCOUT_COOKIE_I was sent by it.

Likewise, the SCOUT_COOKIE_R should be constructed such that the creator of it can later verify when presented with such a cookie whether it was really one that it sent or not. The creation of the value inside the SCOUT_COOKIE_R payload should not cause per-session state creation at the scout responder.

5 Peer Discovery

5.1 Introduction

CASP separates the peer discovery procedure from signaling message delivery. This provides the possibility to cover path-coupled as well as path-decoupled signaling with the same protocol. Only the discovery procedure is different.

This document includes a description of the path-coupled discovery procedure called Scout which is designed similar to the RSVP PATH message [4].

Scout discovery messages are only required if the next NSIS node is more than one network-layer hop away and if there is no other suitable means of discovering the next NSIS node. Manual configuration or routing table lookup is also a viable option of discovery of the next NSIS peer which avoids the use of dynamic discovery procedures.

To discover the next CASP aware node, N, the node wishing to send a CASP message performs the following steps:

1. If M-session state has already been setup with N then the next CASP node is already known. The client message payload is sent with the CASP message to N. Done.

2. If the M-session state is not available at the current node the IP address N of the next CASP node for a given destination IP address has to be discovered using some discovery procedure (e.g. routing table inspection or a scout message).

3. Once the M-session state is created and next-peer information is stored, the CASP message payload is transmitted to N.

Note that the discovery procedure has to be done for each new signaling session, since a node cannot generally determine the next CASP node by inspecting the destination address.

5.2 Scout Protocol

The Scout protocol is used to discover the next suitable CASP node and the required soft-state refresh interval. (Other mechanisms that incur lower overhead and delay are preferred if available.) Each CASP node that needs to discover the next node ”triggers” a scout message that generates a response indicating the next node. A node replying a Scout message indicates the identity (IP address) of the next CASP aware node. Providing capability discovery with a Scout message is discussed in the Section 10.2.

Scout messages are UDP packets containing some CASP objects and have the IP router alert option [5, 6] set. There are scout requests and responses that follow the usual UDP request-response pattern of reversing source and destination address and ports. Scout requests have an IP destination address set to the destination address of the triggering CASP request. The IP source address is set to the IP address of the Scout message transmitting node. The scout message is forwarded like a normal UDP/IP packet. The destination node always turns around the scout message.

Scout messages have their own reliability mechanism. They are retransmitted periodically, with exponentially increasing retransmission interval, starting at 500 ms.

6 Route Change and Mobility

The CASP M-layer is designed to support route changes, and also allow mobility where the IP address of an endpoint of the data flow may change. Although the CASP M-layer assists with dealing with these situations, it also requires the client to act appropriately to install new reservations and teardown old ones.

6.1 Rerouting

Routes may change in the network for a number of reasons, and an path-coupled signalling protocol must be able to act on such changes.

When a route changes, CASP determines the new next CASP node and adds it to the M-layer state, associating it with a new *next-hop branch identifier*.

When a CASP node receives a message for an existing session (same session and flow identifiers), but from a different previous node, it adds the new previous node to the M-layer state with a new *previous-hop branch identifier*.

Branch identifiers are implementation-internal and are not sent in any CASP messages, as they only have local significance. An implementation may use a counter for the branch identifier, incrementing it by 1 when a new branch occurs. A node needs to be able to determine which branch is the ’most recent’. Previous hop and next hop branch identifiers are not related to one another, and are defined within the scope of an M-layer state.

In the example in Figure 3, the data initially travels from S, through N1, N2, N3 and N6 to D. At N1 the next CASP node is identified as N2, and given a next-hop branch identifier (NH-B) of 0. At N6, N3 is stored as the previous hop, with a previous-hop branch identifier (PH-B) of 0.

A route change occurs, so that the data flow now travels through N1, N4, N5 and N6.

There are a number of ways that the CASP M-layer instance at N1 can determine that the route has changed. It may be that N2 or N3 can determine that they are no longer seeing the data flow, and so signal this fact back upstream, causing rediscovery to be initiated. If N1 has direct visibility of routing information (e.g. N2 and N3 are direct IP neighbours), then it may be able to detect the change based from that. Further examination of this issue is needed.

```
        Initial Route
                   -->
                  /     +----+    +----+
                 /  ----| N2 |----| N3 |---
                /  /    +----+    +----+   \
               /  /                         \
                 / NH-B=0           PH-B=0 \
  +-----+     +----+                       +----+     +-----+
  |  S  |-----| N1 |                       | N6 |-----|  D  |
  +-----+     +----+                       +----+     +-----+
                 \ NH-B=1           PH-B=1 /
               \  \                         /
                \  \    +----+    +----+   /
                 \  ----| N4 |----| N5 |---
                  \     +----+    +----+
                   -->
          New Route
```

Figure 3: Rerouting

However, as a consequence of N1 performing rediscovery, a new signalling path is created. At the point two signalling paths exist for the single data flow. Both paths are identified by the same session and flow identifiers, but are distinguished locally at the CASP nodes where the change occurs by different branch identifiers.

The process of removing the existing client state (and supporting M-layer state) from the old route is a topic for further investigation. If the client relevant to this session is available at N1 or N6 then it may be able to perform an appropriate action, based on the local information that a branch has occurred (as in the RSVP local repair mechanism). Otherwise, the M-layer at N1 or N6 may need to signal back to an endpoint that there is a new signalling route, for some upstream or downstream CASP client to perform the necessary reservation setup along the new path, and teardown along the old path.

6.2 Mobility with Address Changes

Each CASP message contains a *session identifier* object. This is used to reference signaling application state, as described in [2], and has end-to-end significance.

In the example shown in Figure 4, a mobile node (S) acting as data source is attached to an access network (AN1) and is sending data towards D. S then is able to make use of a second interface to attach to another access network (AN2). It performs an application-layer renegotiation, and changes to sending data using its AN2 interface (and associated IP address).

This is still the same signalling application (CASP client) session as before, but now using a different source IP address (i.e. a different flow identifier) and a partially changed route. The "session identifier" has its main significance at the client, allowing it to combine state, so that only a single reservation is held in network C. At the M-layer, there will be two flow identifiers, each with associated previous and next hops.

In this case, the reservation across AN1, and network A may persist, so that the mobile node can immediately switch back to it when it leaves AN2, and continue as before.

```
                          Initial route
                     --------------------->
                ^                                       \
               /      /-----\     /---\                  \
              /    ---| AN1 |-----| A |---                \
             /    /   \-----/     \---/   \                \
Address: sa1     /                         \                v Address: da
Port: sp1       /                           \                  Port: dp
       +-----+                               /---\     +-----+
       |  S  |                               | C |-----|  D  |
       +-----+                               \---/     +-----+
Address: sa2    \                           /
Port: sp2        \                         /                 ^
             \    \   /-----\     /---\   /                 /
              \    ---| AN2 |-----| B |---                 /
               \      \-----/     \---/                   /
                v                                        /
                     -------------------->
                          Second route
```

Figure 4: Mobility with Address Changes

The session identifier remains constant, since the signaling application session remains constant. A new flow identifier is introduced, due to the new network interface. Multiple reservations for the application session exist simultaneously, along different paths and for different flow identifiers. The CASP clients in network C may be able to merge the reservations for the different flow identifiers, by making use of the common session identifier.

Issues from this example are also applicable to the situation where a change in Mobile-IP Care of Address (CoA) occurs.

This mobility scenario suggests that the 'session identifier' on its own is insufficient for identifying a previous hop at a CASP M-layer node, since two flows exist using the same session identifier.

7 CASP over Tunnels

CASP supports any type of tunnel described in [7] without additional complexity. Modifications to the flow identifier do not cause problems for CASP. Since CASP can be started and the terminated anywhere along the path it is simple to trigger a recursive CASP messaging exchange for a tunneled region. Since the discovery procedure is separated from message delivery no special considerations apply.

CASP can operate over any types of tunnels (for example IPsec, IP-in-IP, IPv4/IPv6) if both ingress node and egress node of a tunnel support CASP. In case that CASP is not supported at these nodes then the CASP messages are automatically hidden inside the tunnel region. The scout messages then do not discover CASP nodes inside the tunneled region because of the encapsulation of the IP Router Alert Option of the discovery message. Hence the egress node is the next discovered CASP peer (assuming the egress node is CASP aware).

It is therefore a local descision by a CASP aware ingress node to skip interior nodes with the help of the discovery procedure.

8 IANA Considerations

A future version of the document will include IANA considerations for Classes, C-Types and port numbers for the CASP protocol.

9 Security Considerations

This document describes two protocol components within CASP: a path-coupled discovery mechanism called Scout and the CASP Messaging Layer (M-Layer). Subsequently a brief summary of the security protection mechanisms is provided for both protocols. Additional information about security requirements is available in [1], security threats are described in [8]. A motivation for peer-to-peer security protection based on observations of authorization and charging for a QoS NSLP protocol is found at [9]. Additional security properties required by a firewall and nat traversal NSLP protocol are elaborated in [10].

9.1 Scout Security Protection

Scout messages allow the discovery of nodes participating in the CASP protocol (if no other discovery mechanism is used). Scout messages experience security protection with the help of cookies. The concept of cookies was introduced by Karn and Simpson [11]. The security protection of Scout thereby tries to accomplish the following goals:

- The separation between discovery and signaling message delivery provides a major security advantage. Discovery messages are difficult to protect and are therefore separated from regular message delivery.
- A CASP node receiving a Scout message should not install state. This prevents basic denial of service attacks. To allow such a functionality concepts used in Mobile IPv6 are applied (i.e., the cookie is cryptographically generated and does not require per-session state establishment).
- A CASP node transmitting a Scout message wants to make sure that a response received should match the transmitted Scout message. Hence a cookie is included which has to be returned by the Scout responder.
- A CASP node transmitting a Scout message wants to make sure that a request transmitted is not forged by an adversary which redirects CASP messages to another CASP node. This can only be detected by securely exchanging the cookies again after security association is available.

9.2 CASP M-layer Security Protection

CASP M-layer signaling messages can be given security protection. In case that a transport layer protocol connection is established then protection with TLS [12] is possible (TLS Record Layer). In case of TLS the key exchange protocol is built-in (TLS Handshake Protocol).

IPsec, which operates at the network layer, separates authentication and key exchange from signaling message protection in a more sophisticated way. This allows a number of protocols to be used in order to establish a IPsec SA such as IKE, IKEv2 [13] and also KINK [14]. This gives administrators a large degree of freedom to fit CASP into their existing security infrastructure.

For signaling message protection IPsec would be used in transport mode between two peers whereby the SPD traffic selector can be configured to protect signaling messages with a source and destination address set to the corresponding peer and with the port numbers set appropriately.

Providing channel security and the separation between authentication and key exchange and the efficient signaling message protection a number of performance advantages can be achieved. CASP M-session state between two peers can be efficiently secured with a single IPsec or TLS security association (if desired). As motivated below IPsec ESP can be used to protect the content of the signaling message including its payload between neighboring peers. This provides protection against nodes which do participate in the signaling exchange. Different key management protocols can and will be used depending on the environment. For intra-domain communication per-shared secret authentication between neighboring peers is an option. Once the distance (e.g. number of IP hops) between between interacting peers gets larger or to protect messages between different trust domains public key based key management might provide better scalability properties. Since some architectures and corporate networks extensively use Kerberos as their preferred key management system it is also possible to use KINK in such an environment.

The identity of CASP aware peers in intra- and inter-domain communication is the IP address. For intra-domain communication only connections from peers known to be within the same administrative domain should be accepted. For inter-domain communication the IP address as an identity might not always be sufficient. Instead an identity should be used which allows accounting and charging procedures to be matched to the indicated identity. For communication between the end host and a network the preferred identity will be the user name which corresponds to an identity used during the network access procedures. Since a network access authentication protocol is likely to be executed when a host arrives at a new network AAA procedures are likely to create the necessary financial settlement. It is therefore helpful to use an identity which can be mapped to the identity used during the network access procedures to make authorization and charging easier. This is particularly of relevance if the client carries QoS information. For other NSLPs the authorization procedure might be different but the identity used in the authentication and key exchange procedure (e.g. IKE, IKEv2 or KINK) has to be accessible especially for an entity in the network at the NSLP layer. (Note that this is simpler in case of TLS where the authenticated identity of the user is available to the NSLP via an API.)

Establishing a security association between the end host and the network is challenging since a number of different scenarios with different requirements (e.g. wireless access networks, corporate network, ad hoc networks, etc.) need to be supported. We believe that that these requirements can best be met by allowing a flexible integration of existing authentication and key exchange protocols. Little needs to be done for signaling message protection itself if existing transport protocols are used which allow the security protocol support.

Non peer-to-peer protection is accomplished with the help of CMS. This selective object protection is, however, provided at the client and not at the M-layer.

The session ownership problem described in [8] makes an efficient security protection difficult. For this version of CASP confidentiality protection of the session identifier can be provided by both IPsec and TLS (as provided by most cipher-suites for TLS and IPSec ESP without NULL encryption) to Security protection for the session ownership first version of the protocol may rely prevent eavesdroppers to learn the 128-bit randomly generated session identifier. This type of solution prevents an adversary not participating in the protocol execution from attacking the protocol. Still a number of desireable properties of the protocol can be preserved which disappear if a more complex solution is chosen. Note that CASP-aware network elements along the CASP chain must know the session identifier in order for the protocol to operate correctly and hence some trust into these nodes is required. However, based on the protocol discussion in the NSIS working group and the agreement on desireable protocol properties some additional enhancements (possibly at the NSLP layer) might be required.

10 Open Issues

Some of the issues regarding CASP are outlined below; their suitability and implications are currently under investigation.

10.1 Advanced Discovery Mechanisms

As mentioned above, CASP nodes need to discover the next peer. In addition to the scout protocol, a variety of next CASP peer discovery mechanisms are envisioned as below. Furthermore, several next peer discovery mechanisms can be used together along one CASP chain.

Extending routing protocols: For example, OSPF [15] could indicate CASP capability via an Options bit in the common LSA header or a new LSA. For inter-domain discovery, one solution would be adding a CASP capability option to BGP advertisements.

Service discovery: Using standard service discovery mechanisms such as SLP [16], CASP nodes can find out about local CASP nodes and their capabilities.

First node: By adding an option to router advertisements [17], local nodes can discover the first CASP node in their path.

DHCP: If there is a single CASP node in a local network, DHCP [18] can advertise this node.

Diretory-based discovery: For example, by creating new DNS entries per AS for CASP NTLP and its NSLPs, CASP can also support path-decoupled (next-AS) discovery.

10.2 Capability Discovery

In order to address only nodes which support a certain capability (i.e. a specific NSLP protocol), capability discovery (e.g. a capability vector for the NSLP protocols supported by a particular node or certain security capabilities) may be necessary.

10.3 Other Issues

In addition to unicast scenarios, CASP could support a limited multicast model, source-specific multicast (SSM) [19], by a special way of scouting. Supporting SSM in scout requires additional care (e.g. scout requestor's address should be included in the scout request message, the destination address of the scout message must be set to the SSM destination address, etc.).

The scout protocol could use ICMP instead of UDP, with a new ICMP message type.

11 Summary

The CASP framework relies on existing transport protocols and consists of a messaging layer and a client layer. The messaging layer is application independent and is responsible for delivering of signaling messages and associated NTLP state. In contrast to this application independent component of CASP, the client layer is the application-dependent part. The discovery of next peers along the data path is handled by the Scout protocol, which is a specialized client protocol. CASP attempts to satisfy the NSIS requirements [1] and framework [2]. The CASP framework is designed to be network-friendly, light-weight, flexible and extensible:

- Separation of a generic messaging layer from an application-specific client layer allows easily adding other client layer protocols (NSLPs). Each NSLP only relies on common NTLP services and can be changed without affecting other NSLPs.
- Separation of a next peer discovery functionality from the signaling message delivery allows easier security protection of signaling procedures, and avoids complexity in NTLP. CASP only needs to label the scout (a discovery protocol) packets in the same manner as the data packets, but can assign labels to CASP signaling messages based on the handling needed for them. This also helps removing the restriction on the signaling protocol, such as message size to be limited to MTU or else introducting lower-layer overhead; no additions are allowed in mid-stream.
- CASP messages consist of a sequence of message objects. New objects can be added at the messaging and the client layer as needed to support new functionality.
- While most signaling messages for classical signaling applications are likely to be small and the overall data volume modest, CASP recognizes that there are potential applications that may need to deliver larger volumns of signaling messages that are significantly larger than typical network MTUs. Similarly, cryptographic signatures may cause even common signaling messages to exceed MTU size. Also, during overload situations, user applications will be tempted to retry their reservation requests frequently, so that congestion and flow control is desirable. By reusing existing transport protocol for delivering CASP messages, CASP greatly reduces the complexity of protocol implementations and avoid subtle interoperability problems. Due to the re-use of transport connections, CASP session setup latency is, on average, low.

12 Acknowledgements

We would like to thank Jochen Eisl for his contributions to the first version of CASP. A number of people provided input to the initial version of the CASP draft, including Wolfgang Buecker, Jorge Cuellar, Dirk Kroeselberg, Rainer Falk and Cornel Pampu. We would also like to thank Robert Hancock for his comments on the relationship to the NSIS framework.

A Object Definitions

Classes where the objects contain IP addresses are defined for both IPv4 and IPv6.

All unused fields should be set to zero and ignored on receipt.

A.1 FLOW_ID Class

FLOW_ID class = 1

- IPv4 with ports FLOW_ID Object: Class = 1, C-Type = 1

```
+---------------+---------------+---------------+---------------+
|                     IPv4 Source Address                       |
+---------------+---------------+---------------+---------------+
|                  IPv4 Destination Address                     |
+---------------+---------------+---------------+---------------+
|          Source Port          |       Destination Port        |
+---------------+---------------+---------------+---------------+
|   Protocol    |                      //                       |
+---------------+---------------+---------------+---------------+
```

- IPv4 with IPsec FLOW_ID Object: Class = 1, C-Type = 2

```
+---------------+---------------+---------------+---------------+
|                     IPv4 Source Address                       |
+---------------+---------------+---------------+---------------+
|                  IPv4 Destination Address                     |
+---------------+---------------+---------------+---------------+
|                              SPI                              |
+---------------+---------------+---------------+---------------+
|   Protocol    |                      //                       |
+---------------+---------------+---------------+---------------+
```

- IPv6 FLOW_ID Object: Class = 1, C-Type = 3

```
+---------------+---------------+---------------+---------------+
|                                                               |
+                                                               +
|                             IPv6                              |
+                        Source Address                         +
|                          (16 bytes)                           |
+                                                               +
|                                                               |
+---------------+---------------+---------------+---------------+
|                                                               |
+                                                               +
|                             IPv6                              |
+                     Destination Address                       +
|                          (16 bytes)                           |
+                                                               +
|                                                               |
```

```
+---------------+---------------+---------------+---------------+
|          //           |         Flow Label (20 bits)          |
+---------------+---------------+---------------+---------------+
```

- Source Address

  Source address of the application data flow.

- Destination Address

  Destination address of the application data flow.

- Protocol

  The IP Protocol Identifier for the data flow. For the IPv4 with IPsec FLOW_ID this SHOULD indicate either AH or ESP.

- Source Port

  The UDP/TCP/SCTP (or similar) source port for the session.

- Destination Port

  The UDP/TCP/SCTP (or similar) destination port for the session.

- Flow Label

  The IPv6 flow label for the data flow.

Other FLOW_ID C-Types could be defined in the future to support other demultiplexing conventions in the transport-layer or application-layer.

A.2 CASP_TIMEOUT Class

CASP_TIMEOUT Class = 2

- CASP_TIMEOUT Object: Class = 2, C-Type = 1

```
+---------------+---------------+---------------+---------------+
|                       Timeout (seconds)                       |
+---------------+---------------+---------------+---------------+
```

- Timeout

  The time in seconds after which CASP M-layer soft-state should be removed if no refresh is received.

A.3 CLIENT_DATA Class

CLIENT_DATA class = 3

- All CLIENT_DATA Objects are variable length opaque data.

  The C-Type identifies the client which should be used to process this message. Currently defined values for CASP clients are:

  - QoS: Class = 3, C-Type = 1

```
+---------------+---------------+---------------+---------------+
|                                                               |
//                        (Client Data)                        //
|                                                               |
+---------------+---------------+---------------+---------------+
```

- Client Data

  The contents of the Client Data object is of variable length. It has no significance at the M-layer and the client may choose to format it in any way.

A.4 ERROR Class

ERROR Class = 4

- ERROR Object: Class = 4, C-Type = 1

```
+---------------+---------------+---------------+---------------+
|                          Error Code                           |
+---------------+---------------+---------------+---------------+
```

- Error Code

  Provides an indication of the what error occurred. Currently defined codes are:

  - UNKNOWN: 0
  - COOKIE_ERROR: 1

A.5 SCOUT_COOKIE_I Class

SCOUT_COOKIE_I Class = 5

- SCOUT_COOKIE_I object: Class = 5, C-Type = 1

```
+---------------+---------------+---------------+---------------+
|                           Cookie(i)                           |
+                (64-bit Scout Request cookie)                  +
|                                                               |
+---------------+---------------+---------------+---------------+
```

- Cookie(i)

  The initiator cookie is a 64-bit random number, selected by the scout initiator.

A.6 SCOUT_COOKIE_R Class

SCOUT_COOKIE_R Class = 6

- SCOUT_COOKIE_R object: Class = 6, C-Type = 1

```
+---------------+---------------+---------------+---------------+
|                            Cookie(r)                          |
+                (64-bit Scout Response cookie)                 +
|                                                               |
+---------------+---------------+---------------+---------------+
```

- Cookie(r)

  The responder cookie is a 64-bit value, selected by the responder.

B Authors' Addresses

Henning Schulzrinne
Dept. of Computer Science
Columbia University
1214 Amsterdam Avenue
New York, NY 10027
USA
EMail: schulzrinne@cs.columbia.edu

Hannes Tschofenig
Siemens AG
Otto-Hahn-Ring 6
Munich 81739
Germany
EMail: Hannes.Tschofenig@mchp.siemens.de

Xiaoming Fu
Institute for Informatics
University of Goettingen
Lotzestrasse 16-18
Goettingen 37083
Germany
EMail: fu@cs.uni-goettingen.de

Andrew McDonald
Roke Manor Research
Old Salisbury Lane
Romsey, Hampshire
UK
EMail: andrew.mcdonald@roke.co.uk

References

[1] M. Brunner, "Requirements for QoS signaling protocols," Internet Draft, Internet Engineering Task Force, July 2002. Work in progress.

[2] R. Hancock, I. Freytsis, G. Karagiannis, J. Loughney, and S. V. den Bosch, "Next steps in signaling: Framework," Internet Draft, Internet Engineering Task Force, 2002. Work in progress.

[3] S. Bradner, "Key words for use in RFCs to indicate requirement levels," RFC 2119, Internet Engineering Task Force, Mar. 1997.

[4] R. Braden, Ed., L. Zhang, S. Berson, S. Herzog, and S. Jamin, "Resource ReSerVation protocol (RSVP) – version 1 functional specification," RFC 2205, Internet Engineering Task Force, Sept. 1997.

[5] D. Katz, "IP router alert option," RFC 2113, Internet Engineering Task Force, Feb. 1997.

[6] C. Partridge and A. Jackson, "IPv6 router alert option," RFC 2711, Internet Engineering Task Force, Oct. 1999.

[7] A. Terzis, J. Krawczyk, J. Wroclawski, and L. Zhang, "RSVP operation over IP tunnels," RFC 2746, Internet Engineering Task Force, Jan. 2000.

[8] H. Tschofenig and D. Kroeselberg, "Security threats for nsis," internet draft, Internet Engineering Task Force, 2003. Work in progress.

[9] H. Tschofenig, M. Buechli, S. Van den Bosch, and H. Schulzrinne, "Nsis authentication, authorization and accounting issues," Internet Draft, Internet Engineering Task Force, 2003. Work in progress.

[10] H. Tschofenig, H. Schulzrinne, and C. Aoun, "A firewall/nat traversal client for casp," internet draft, Internet Engineering Task Force, 2003. Work in progress.

[11] P. Karn and W. Simpson, "Photuris: Session-key management protocol," RFC 2522, Internet Engineering Task Force, Mar. 1999.

[12] T. Dierks and C. Allen, "The TLS protocol version 1.0," RFC 2246, Internet Engineering Task Force, Jan. 1999.

[13] D. Harkins, C. Kaufman, *et al.*, "Propsal for the IKEv2 protocol," Internet Draft, Internet Engineering Task Force, Apr. 2002. Work in progress.

[14] M. Thomas *et al.*, "Kerberized internet negotiation of keys (KINK)," Internet Draft, Internet Engineering Task Force, Nov. 2001. Work in progress.

[15] J. Moy, "OSPF version 2," RFC 2328, Internet Engineering Task Force, Apr. 1998.

[16] E. Guttman, C. Perkins, J. Veizades, and M. Day, "Service location protocol, version 2," RFC 2608, Internet Engineering Task Force, June 1999.

[17] T. Narten, E. Nordmark, and W. Simpson, "Neighbor discovery for IP version 6 (ipv6)," RFC 2461, Internet Engineering Task Force, Dec. 1998.

[18] R. Droms, "Dynamic host configuration protocol," RFC 2131, Internet Engineering Task Force, Mar. 1997.

[19] H. Holbrook and B. Cain, "Source-specific multicast for IP," Internet Draft, Internet Engineering Task Force, Feb. 2002. Work in progress.