Internet Engineering Task Force (IETF)
Request for Comments: 5981
Category: Experimental
ISSN: 2070-1721

J. Manner
Aalto University
M. Stiemerling
NEC
H. Tschofenig
Nokia Siemens Networks
R. Bless, Ed.
KIT

February 2011

Authorization for NSIS Signaling Layer Protocols

Abstract

Signaling layer protocols specified within the Next Steps in Signaling (NSIS) framework may rely on the General Internet Signaling Transport (GIST) protocol to handle authorization. Still, the signaling layer protocol above GIST itself may require separate authorization to be performed when a node receives a request for a certain kind of service or resources. This document presents a generic model and object formats for session authorization within the NSIS signaling layer protocols. The goal of session authorization is to allow the exchange of information between network elements in order to authorize the use of resources for a service and to coordinate actions between the signaling and transport planes.

Status of This Memo

This document is not an Internet Standards Track specification; it is published for examination, experimental implementation, and evaluation.

This document defines an Experimental Protocol for the Internet community. This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Not all documents approved by the IESG are a candidate for any level of Internet Standard; see Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc5981.

Table of Contents

1. Introduction

The Next Steps in Signaling (NSIS) framework [RFC4080] defines a suite of protocols for the next generation in Internet signaling. The design is based on a generalized transport protocol for signaling applications, the General Internet Signaling Transport (GIST) [RFC5971], and various kinds of signaling applications. Two signaling applications and their NSIS Signaling Layer Protocol (NSLP) have been designed, a Quality of Service application (QoS NSLP) [RFC5974] and a NAT/firewall application (NATFW NSLP) [RFC5973].

The basic security architecture for NSIS is based on a chain-of-trust model, where each GIST hop may choose the appropriate security protocol, taking into account the signaling application requirements. For instance, communication between two directly adjacent GIST peers may be secured via TCP/TLS. On the one hand, this model is appropriate for a number of different use cases and allows the signaling applications to leave the handling of security to GIST. On the other hand, several sessions of different signaling applications are then multiplexed onto the same GIST TLS connection.

Yet, in order to allow for finer-grain per-session or per-user admission control, it is necessary to provide a mechanism for ensuring that the use of resources by a host has been properly authorized before allowing the signaling application to commit the resource request, e.g., a QoS reservation or mappings for NAT traversal. In order to meet this requirement, there must be information in the NSLP message that may be used to verify the validity of the request. This can be done by providing the host with a Session Authorization Object that is inserted into the message and verified by the respective network elements.

This document describes a generic NSLP-layer Session Authorization Object (SESSION_AUTH) used to convey authorization information for the request. "Generic" in this context means that it is usable by all NSLPs. The scheme is based on third-party tokens. A trusted third party provides authentication tokens to clients and allows verification of the information by the network elements. The requesting host inserts the authorization information (e.g., a policy object) acquired from the trusted third party into the NSLP message to allow verification of the network resource request. Network elements verify the request and then process it based on admission policy (e.g., they perform a resource reservation or change bindings or firewall filter). This work is based on RFC 3520 [RFC3520] and RFC 3521 [RFC3521].

The default operation when using NSLP-layer session authorization is to add one authorization policy object. Yet, in order to support end-to-end signaling and request authorization from different networks, a host initiating an NSLP signaling session may add more than one SESSION_AUTH object in the message. The identifier of the authorizing entity can be used by the network elements to use the third party they trust to verify the request.

2. Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, RFC 2119 [RFC2119].

The term "NSLP node" (NN) is used to refer to an NSIS node running an NSLP protocol that can make use of the authorization object discussed in this document. Currently, this node would run either the QoS NSLP [RFC5974] or the NAT/Firewall NSLP [RFC5973] service.

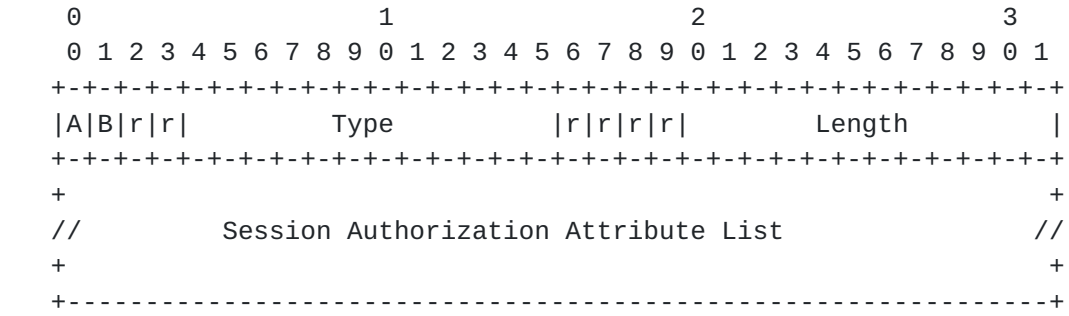
3. Session Authorization Object

This section presents a new NSLP-layer object called session authorization (SESSION_AUTH). The SESSION_AUTH object can be used in the currently specified and future NSLP protocols.

The authorization attributes follow the format and specification given in RFC3520 [RFC3520].

3.1. Session Authorization Object format

The SESSION_AUTH object contains a list of fields that describe the session, along with other attributes. The object header follows the generic NSLP object header; therefore, it can be used together with any NSLP.



The value for the Type field comes from shared NSLP object type space. The Length field is given in units of 32-bit words and measures the length of the Value component of the TLV object (i.e., it does not include the standard header).

The bits marked 'A' and 'B' are extensibility flags and are used to signal the desired treatment for objects whose treatment has not been defined in the protocol specification (i.e., whose Type field is unknown at the receiver). The following four categories of object have been identified, and are described here for informational purposes only (for normative behavior, refer to the particular NSLP documents, e.g., [RFC5974] [RFC5973]).

AB=00 ("Mandatory"): If the object is not understood, the entire message containing it MUST be rejected, and an error message sent back (usually of class/code "Protocol Error/Unknown object present").

AB=01 ("Ignore"): If the object is not understood, it MUST be deleted, and the rest of the message processed as usual.

AB=10 ("Forward"): If the object is not understood, it MUST be retained unchanged in any message forwarded as a result of message processing, but not stored locally.

AB=11 ("Refresh"): If the object is not understood, it should be incorporated into the locally stored signaling application state for this flow/session, forwarded in any resulting message, and also used in any refresh or repair message which is generated locally. This flag combination is not used by all NSLPs, e.g., it is not used in the NATFW NSLP.

The remaining bits marked 'r' are reserved. The extensibility flags follow the definition in the GIST specification. The SESSION_AUTH object defined in this specification MUST have the AB bits set to "10". An NSLP Node (NN) may use the authorization information if it is configured to do so, but may also just skip the object.

Type: SESSION_AUTH_OBJECT (0x016)

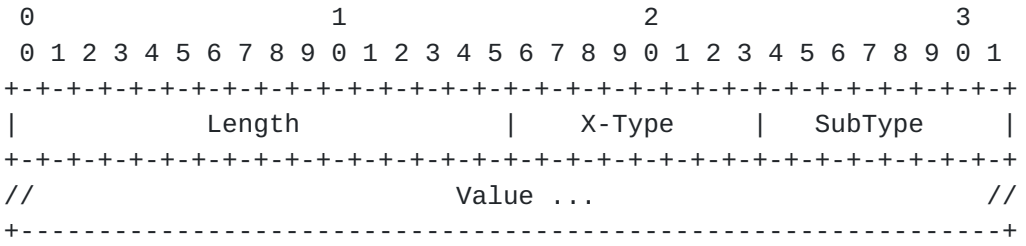
Length: Variable, contains length of session authorization object list in units of 32-bit words.

Session Authorization Attribute List: variable length

The session authorization attribute list is a collection of objects that describes the session and provides other information necessary to verify resource request (e.g., a resource reservation, binding, or firewall filter change request). An initial set of valid objects is described in Section 3.2.

3.2. Session Authorization Attributes

A session authorization attribute may contain a variety of information and has both an attribute type and sub-type. The attribute itself MUST be a multiple of 4 octets in length, and any attributes that are not a multiple of 4 octets long MUST be padded to a 4-octet boundary. All padding bytes MUST have a value of zero.



Length: 16 bits

The Length field is two octets and indicates the actual length of the attribute (including Length, X-Type, and SubType fields) in number of octets. The length does NOT include any padding of the value field to make the attribute's length a multiple of 4 octets.

X-Type: 8 bits

Session authorization attribute type (X-Type) field is one octet. IANA acts as a registry for X-Types as described in Section 8, IANA Considerations. This specification uses the following X-Types:

1. AUTH_ENT_ID: The unique identifier of the entity that authorized the session.

2. SESSION_ID: The unique identifier for this session, usually created locally at the authorizing entity. See also RFC 3520 [RFC3520]; not to be confused with the SESSION-ID of GIST/ NSIS.

3. SOURCE_ADDR: The address specification for the signaling session initiator, i.e., the source address of the signaling message originator.

4. DEST_ADDR: The address specification for the signaling session endpoint.

5. START_TIME: The starting time for the session.

6. END_TIME: The end time for the session.

7. AUTHENTICATION_DATA: The authentication data of the Session Authorization Object.

SubType: 8 bits

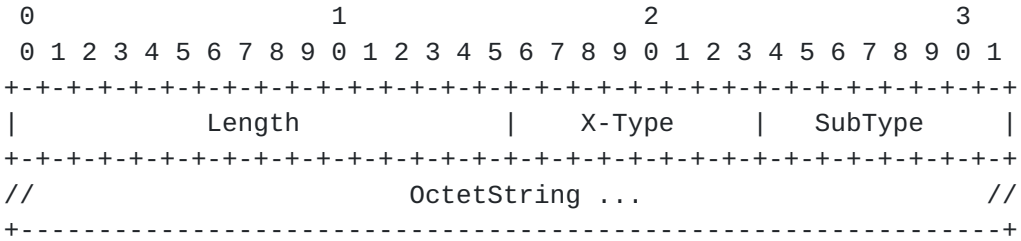
Session authorization attribute sub-type is one octet in length. The value of the SubType depends on the X-Type.

Value: variable length

The attribute-specific information.

3.2.1. Authorizing Entity Identifier

The AUTH_ENT_ID is used to identify the entity that authorized the initial service request and generated the Session Authorization Object. The AUTH_ENT_ID may be represented in various formats, and the SubType is used to define the format for the ID. The format for AUTH_ENT_ID is as follows:



Length: Length of the attribute, which MUST be > 4.

X-Type: AUTH_ENT_ID

SubType:

The following sub-types for AUTH_ENT_ID are defined. IANA acts as a registry for AUTH_ENT_ID SubTypes as described in Section 8, IANA Considerations. Initially, the registry contains the following SubTypes of AUTH_ENT_ID:

1. IPV4_ADDRESS: IPv4 address represented in 32 bits.

2. IPV6_ADDRESS: IPv6 address represented in 128 bits.

3. FQDN: Fully Qualified Domain Name as defined in [RFC1034] as an ASCII string.

4. ASCII_DN: X.500 Distinguished name as defined in [RFC4514] as an ASCII string.

5. UNICODE_DN: X.500 Distinguished name as defined in [RFC4514] as a UTF-8 string.

6. URI: Universal Resource Identifier, as defined in [RFC3986].

7. KRB_PRINCIPAL: Fully Qualified Kerberos Principal name represented by the ASCII string of a principal, followed by the @ realm name as defined in [RFC4120] (e.g., johndoe@nowhere).

8. X509_V3_CERT: The Distinguished Name of the subject of the certificate as defined in [RFC4514] as a UTF-8 string.

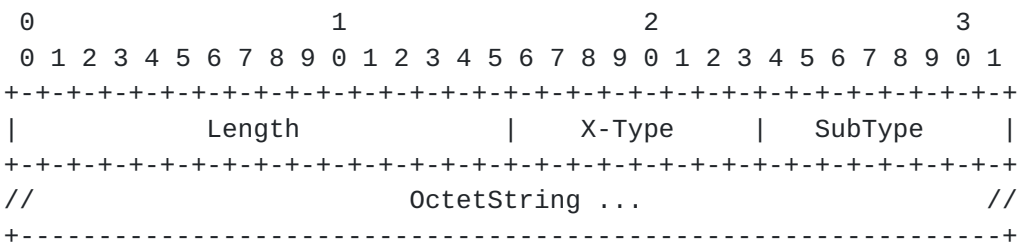
9. PGP_CERT: The OpenPGP certificate of the authorizing entity as defined as Public-Key Packet in [RFC4880].

10. HMAC_SIGNED: Indicates that the AUTHENTICATION_DATA attribute contains a self-signed HMAC signature [RFC2104] that ensures the integrity of the NSLP message. The HMAC is calculated over all NSLP objects given in the NSLP_OBJECT_LIST attribute that MUST also be present. The object specifies the hash algorithm that is used for calculation of the HMAC as Transform ID from Transform Type 3 of the IKEv2 registry [RFC5996].

OctetString: Contains the authorizing entity identifier.

3.2.2. Session Identifier

SESSION_ID is a unique identifier used by the authorizing entity to identify the request. It may be used for a number of purposes, including replay detection, or to correlate this request to a policy decision entry made by the authorizing entity. For example, the SESSION_ID can be based on simple sequence numbers or on a standard NTP timestamp.



Length: Length of the attribute, which MUST be > 4.

X-Type: SESSION_ID

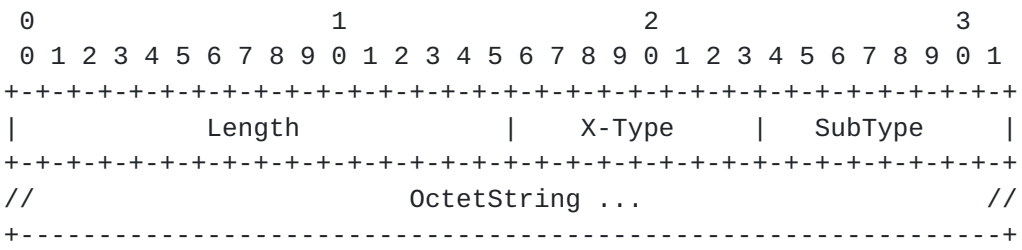
SubType:

No sub-types for SESSION_ID are currently defined; this field MUST be set to zero. The authorizing entity is the only network entity that needs to interpret the contents of the SESSION_ID; therefore, the contents and format are implementation dependent.

OctetString: The OctetString contains the session identifier.

3.2.3. Source Address

SOURCE_ADDR is used to identify the source address specification of the authorized session. This X-Type may be useful in some scenarios to make sure the resource request has been authorized for that particular source address and/or port. Usually, it corresponds to the signaling source, e.g., the IP source address of the GIST packet, or flow source or flow destination address, respectively, which are contained in the GIST MRI (Message Routing Information) object.



Length: Length of the attribute, which MUST be > 4.

X-Type: SOURCE_ADDR

SubType:

The following sub-types for SOURCE_ADDR are defined. IANA acts as a registry for SOURCE_ADDR SubTypes as described in Section 8, IANA Considerations. Initially, the registry contains the following SubTypes for SOURCE_ADDR:

1. IPV4_ADDRESS: IPv4 address represented in 32 bits.

2. IPV6_ADDRESS: IPv6 address represented in 128 bits.

3. UDP_PORT_LIST: list of UDP port specifications, represented as 16 bits per list entry.

4. TCP_PORT_LIST: list of TCP port specifications, represented as 16 bits per list entry.

5. SPI: Security Parameter Index, represented in 32 bits.

OctetString: The OctetString contains the source address information.

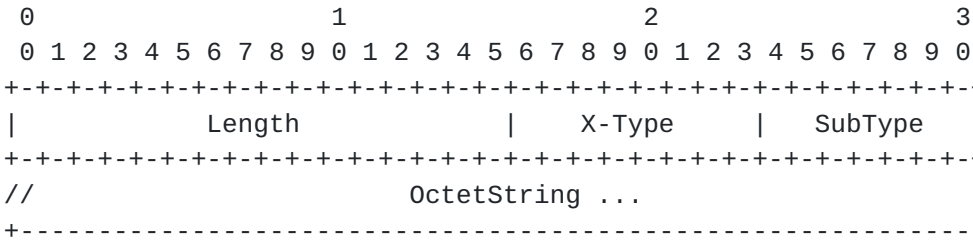
In scenarios where a source address is required (see Section 5), at least one of the sub-types 1 or 2 MUST be included in every Session Authorization Object. Multiple SOURCE_ADDR attributes MAY be included if multiple addresses have been authorized. The source address of the request (e.g., a QoS NSLP RESERVE) MUST match one of the SOURCE_ADDR attributes contained in this Session Authorization Object.

At most, one instance of sub-type 3 MAY be included in every Session Authorization Object. At most, one instance of sub-type 4 MAY be included in every Session Authorization Object. Inclusion of a sub-type 3 attribute does not prevent inclusion of a sub-type 4 attribute (i.e., both UDP and TCP ports may be authorized).

If no PORT attributes are specified, then all ports are considered valid; otherwise, only the specified ports are authorized for use. Every source address and port list must be included in a separate SOURCE_ADDR attribute.

3.2.4. Destination Address

DEST_ADDR is used to identify the destination address of the authorized session. This X-Type may be useful in some scenarios to make sure the resource request has been authorized for that particular destination address and/or port.



Length: Length of the attribute in number of octets, which MUST be > 4.

X-Type: DEST_ADDR

SubType:

The following sub-types for DEST_ADDR are defined. IANA acts as a registry for DEST_ADDR SubTypes as described in Section 8, IANA Considerations. Initially, the registry contains the following SubTypes for DEST_ADDR:

1. IPV4_ADDRESS: IPv4 address represented in 32 bits.

2. IPV6_ADDRESS: IPv6 address represented in 128 bits.

3. UDP_PORT_LIST: list of UDP port specifications, represented as 16 bits per list entry.

4. TCP_PORT_LIST: list of TCP port specifications, represented as 16 bits per list entry.

5. SPI: Security Parameter Index, represented in 32 bits.

OctetString: The OctetString contains the destination address specification.

In scenarios where a destination address is required (see Section 5), at least one of the sub-types 1 or 2 MUST be included in every Session Authorization Object. Multiple DEST_ADDR attributes MAY be included if multiple addresses have been authorized. The destination

address field of the resource reservation datagram (e.g., QoS NSLP Reserve) MUST match one of the DEST_ADDR attributes contained in this Session Authorization Object.

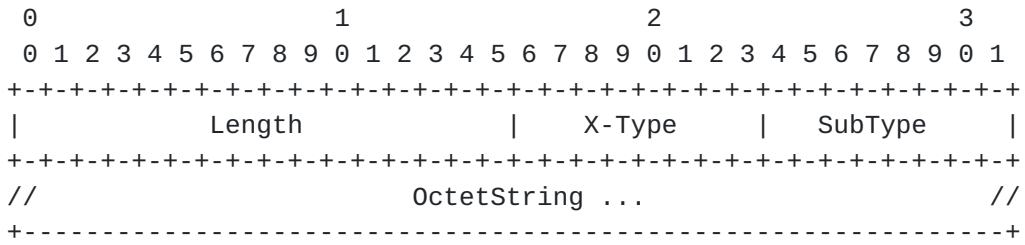
At most, one instance of sub-type 3 MAY be included in every Session Authorization Object. At most, one instance of sub-type 4 MAY be included in every Session Authorization Object. Inclusion of a sub-type 3 attribute does not prevent inclusion of a sub-type 4 attribute (i.e., both UDP and TCP ports may be authorized).

If no PORT attributes are specified, then all ports are considered valid; otherwise, only the specified ports are authorized for use.

Every destination address and port list must be included in a separate DEST_ADDR attribute.

3.2.5. Start Time

START_TIME is used to identify the start time of the authorized session and can be used to prevent replay attacks. If the SESSION_AUTH object is presented in a resource request, the network SHOULD reject the request if it is not received within a few seconds of the start time specified.



Length: Length of the attribute, which MUST be > 4.

X-Type: START_TIME

SubType:

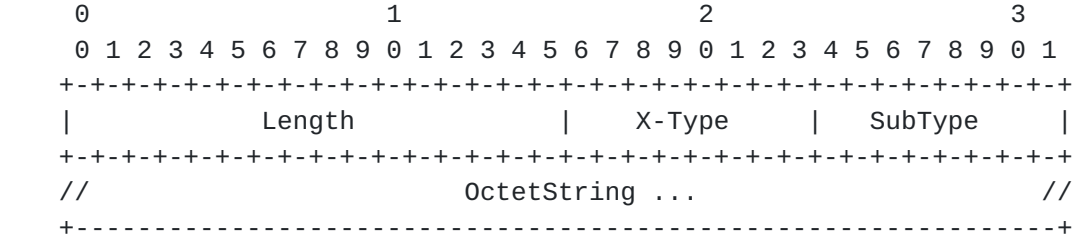
The following sub-type for START_TIME is defined. IANA acts as a registry for START_TIME SubTypes as described in Section 8, IANA Considerations. Initially, the registry contains the following SubType for START_TIME:

1 NTP_TIMESTAMP: NTP Timestamp Format as defined in RFC 5905 [RFC5905].

OctetString: The OctetString contains the start time.

3.2.6. End Time

END_TIME is used to identify the end time of the authorized session and can be used to limit the amount of time that resources are authorized for use (e.g., in prepaid session scenarios).



Length: Length of the attribute, which MUST be > 4.

X-Type: END_TIME

SubType:

The following sub-type for END_TIME is defined. IANA acts as a registry for END_TIME SubTypes as described in Section 8, IANA Considerations. Initially, the registry contains the following SubType for END_TIME:

1 NTP_TIMESTAMP: NTP Timestamp Format as defined in RFC 5905 [RFC5905].

OctetString: The OctetString contains the end time.

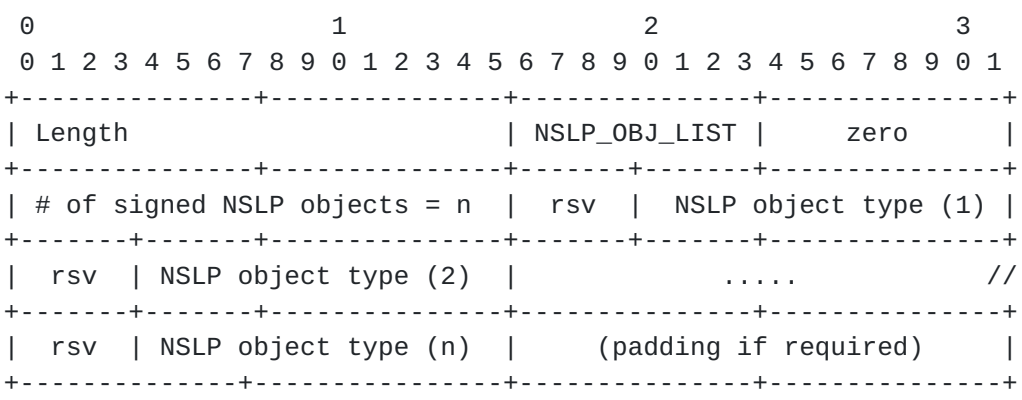
3.2.7. NSLP Object List

The NSLP_OBJECT_LIST attribute contains a list of NSLP object types that are used in the keyed-hash computation whose result is given in the AUTHENTICATION_DATA attribute. This allows for an integrity protection of NSLP PDUs. If an NSLP_OBJECT_LIST attribute has been included in the SESSION_AUTH object, an AUTHENTICATION_DATA attribute MUST also be present.

The creator of this attribute lists every NSLP object type whose NSLP PDU object was included in the computation of the hash. The hash computation has to follow the order of the NSLP object types as specified by the list. The receiver can verify the integrity of the NSLP PDU by computing a hash over all NSLP objects that are listed in this attribute (in the given order), including all the attributes of the authorization object. Since all NSLP object types are unique over all different NSLPs, this will work for any NSLP.

Basic NSIS Transport Layer Protocol (NTLP) / NSLP objects like the session ID, the NSLPID, and the MRI MUST be always included in the HMAC. Since they are not carried within the NSLP itself, but only within GIST, they have to be provided for HMAC calculation, e.g., they can be delivered via the GIST API. They MUST be normalized to their network representation from [RFC5971] again before calculating the hash. These values MUST be hashed first (in the order session ID, NSLPID, MRI), before any other NSLP object values that are included in the hash computation.

A summary of the NSLP_OBJECT_LIST attribute format is described below.



Length: Length of the attribute, which MUST be > 4.

X-Type: NSLP_OBJECT_LIST

SubType: No sub-types for NSLP_OBJECT_LIST are currently defined. This field MUST be set to 0 and ignored upon reception.

# of signed NSLP objects: The number n of NSLP object types that follow. n=0 is allowed; in that case, only a padding field is contained.

rsv: reserved bits; MUST be set to 0 and ignored upon reception.

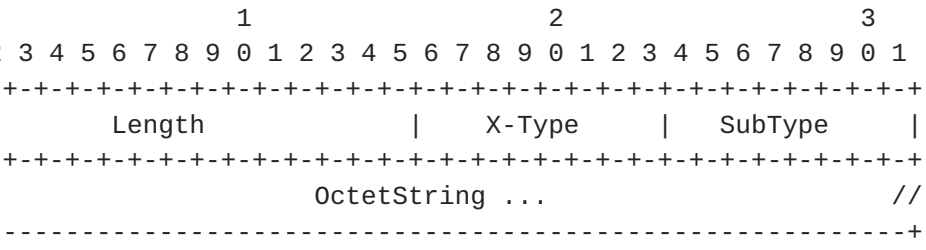
NSLP object type: the NSLP 12-bit object type identifier of the object that was included in the hash calculation. The NSLP object type values comprise only 12 bits, so four bits per type value are currently not used within the list. Depending on the number of signed objects, a corresponding padding word of 16 bits must be supplied.

padding: padding MUST be added if the number of NSLP objects is even and MUST NOT be added if the number of NSLP objects is odd. If padding has to be applied, the padding field MUST be 16 bits set to 0, and its contents MUST be ignored upon reception.

3.2.8. Authentication Data

The AUTHENTICATION_DATA attribute contains the authentication data of the SESSION_AUTH object and signs all the data in the object up to the AUTHENTICATION_DATA. If the AUTHENTICATION_DATA attribute has been included in the SESSION_AUTH object, it MUST be the last attribute in the list. The algorithm used to compute the authentication data depends on the AUTH_ENT_ID SubType field. See Section 4 entitled "Integrity of the SESSION_AUTH Object".

A summary of the AUTHENTICATION_DATA attribute format is described below.



Length: Length of the attribute, which MUST be > 4.

X-Type: AUTHENTICATION_DATA

SubType: No sub-types for AUTHENTICATION_DATA are currently defined. This field MUST be set to 0 and ignored upon reception.

OctetString: The OctetString contains the authentication data of the SESSION_AUTH.

4. Integrity of the SESSION_AUTH Object

This section describes how to ensure that the integrity of the SESSION_AUTH object is preserved.

4.1. Shared Symmetric Keys

In shared symmetric key environments, the AUTH_ENT_ID MUST be of sub-types: IPV4_ADDRESS, IPV6_ADDRESS, FQDN, ASCII_DN, UNICODE_DN, or URI. An example SESSION_AUTH object is shown below.

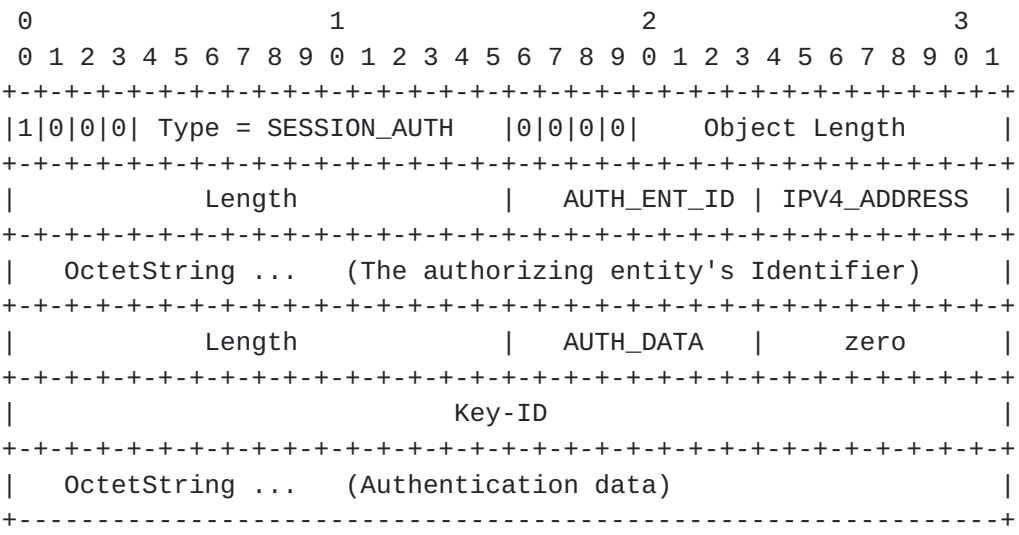


Figure 1: Example of a SESSION_AUTH Object

4.1.1. Operational Setting Using Shared Symmetric Keys

This assumes both the Authorizing Entity and the network router/PDP (Policy Decision Point) are provisioned with shared symmetric keys, policies detailing which algorithm to be used for computing the authentication data, and the expected length of the authentication data for that particular algorithm.

Key maintenance is outside the scope of this document, but SESSION_AUTH implementations MUST at least provide the ability to manually configure keys and their parameters. The key used to produce the authentication data is identified by the AUTH_ENT_ID field. Since multiple keys may be configured for a particular AUTH_ENT_ID value, the first 32 bits of the AUTHENTICATION_DATA field MUST be a Key-ID to be used to identify the appropriate key. Each key must also be configured with lifetime parameters for the time period within which it is valid as well as an associated cryptographic algorithm parameter specifying the algorithm to be used with the key. At a minimum, all SESSION_AUTH implementations MUST support the HMAC-SHA2-256 [RFC4868] [RFC2104] cryptographic algorithm for computing the authentication data.

It is good practice to regularly change keys. Keys MUST be configurable such that their lifetimes overlap, thereby allowing smooth transitions between keys. At the midpoint of the lifetime overlap between two keys, senders should transition from using the current key to the next/longer-lived key. Meanwhile, receivers simply accept any identified key received within its configured lifetime and reject those that are not.

4.2. Kerberos

Since Kerberos [RFC4120] is widely used for end-user authorization, e.g., in Windows domains, it is well suited for being used in the context of user-based authorization for NSIS sessions. For instance, a user may request a ticket for authorization to install rules in an NATFW-capable router.

In a Kerberos environment, it is assumed that the user of the NSLP requesting host requests a ticket from the Kerberos Key Distribution Center (KDC) for using the NSLP node (router) as a resource (target service). The NSLP requesting host (client) can present the ticket to the NSLP node via Kerberos by sending a KRB_CRED message to the NSLP node independently but prior to the NSLP exchange. Thus, the principal name of the service must be known at the client in advance, though the exact IP address may not be known in advance. How the name is assigned and made available to the client is implementation specific. The extracted common session key can subsequently be used to employ the HMAC_SIGNED variant of the SESSION_AUTH object.

Another option is to encapsulate the credentials in the AUTHENTICATION_DATA portion of the SESSION_AUTH object. In this case, the AUTH_ENT_ID MUST be of the sub-type KRB_PRINCIPAL. The KRB_PRINCIPAL field is defined as the Fully Qualified Kerberos Principal name of the authorizing entity. The AUTHENTICATION_DATA portion of the SESSION_AUTH object contains the KRB_CRED message that the receiving NSLP node has to extract and verify. A second SESSION_AUTH object of type HMAC_SIGNED SHOULD protect the integrity of the NSLP message, including the prior SESSION_AUTH object. The session key included in the first SESSION_AUTH object has to be used for HMAC calculation.

An example of the Kerberos AUTHENTICATION_DATA object is shown below in Figure 2.

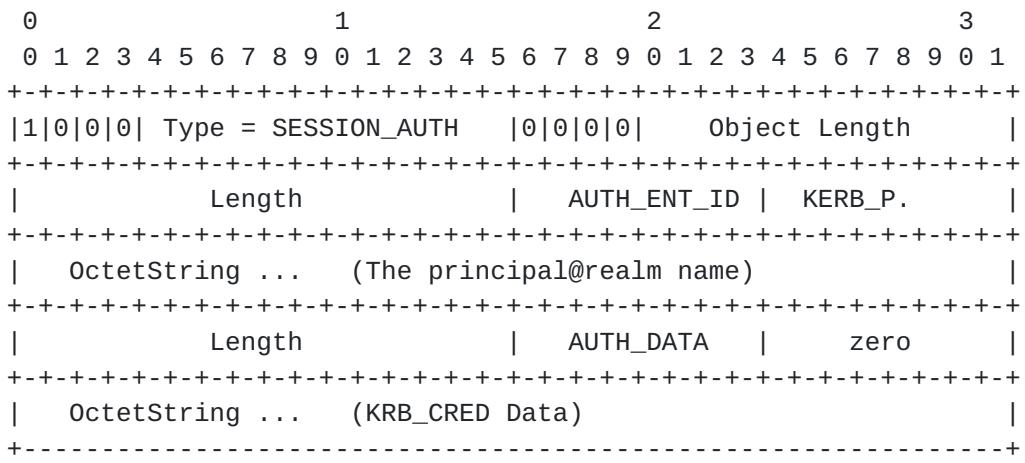


Figure 2: Example of a Kerberos AUTHENTICATION_DATA Object

4.3. Public Key

In a public key environment, the AUTH_ENT_ID MUST be of the sub-types: X509_V3_CERT or PGP_CERT. The authentication data is used for authenticating the authorizing entity. Two examples of the public key SESSION_AUTH object are shown in Figures 3 and 4.

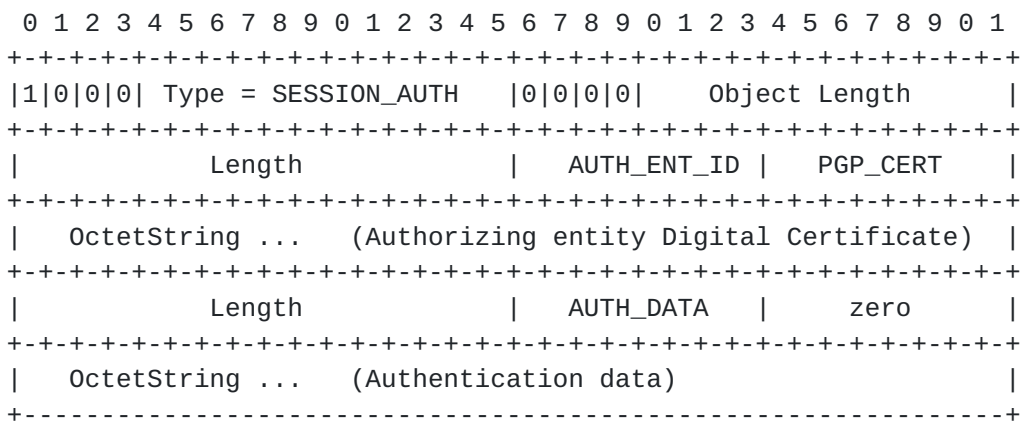


Figure 3: Example of a SESSION_AUTH_OBJECT Using a PGP Certificate

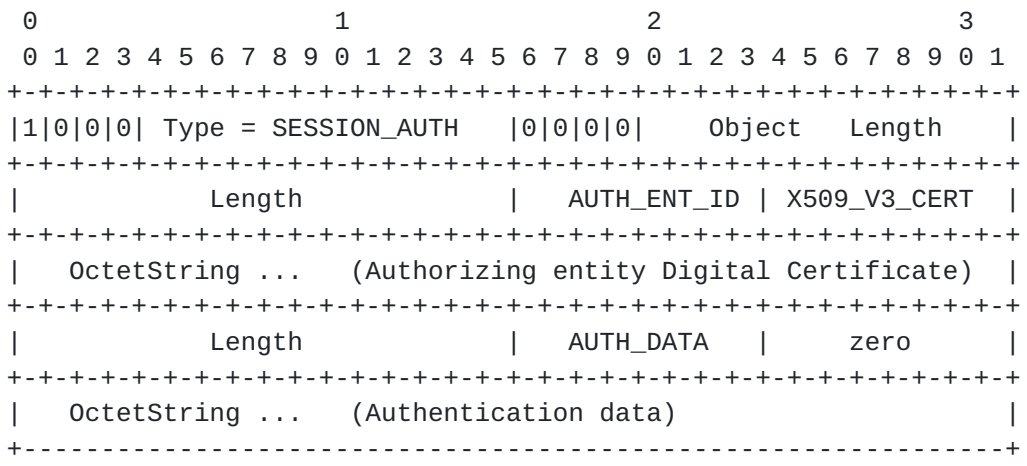


Figure 4: Example of a SESSION_AUTH_OBJECT Using an X509_V3_CERT Certificate

4.3.1. Operational Setting for Public-Key-Based Authentication

Public-key-based authentication assumes the following:

- Authorizing entities have a pair of keys (private key and public key).
- The private key is secured with the authorizing entity.
- Public keys are stored in digital certificates; a trusted party, the certificate authority (CA), issues these digital certificates.
- The verifier (PDP or router) has the ability to verify the digital certificate.

The authorizing entity uses its private key to generate AUTHENTICATION_DATA. Authenticators (router, PDP) use the authorizing entity's public key (stored in the digital certificate) to verify and authenticate the object.

4.3.1.1. X.509 V3 Digital Certificates

When the AUTH_ENT_ID is of type X509_V3_CERT, AUTHENTICATION_DATA MUST be generated by the authorizing entity following these steps:

- A signed-data is constructed as defined in RFC 5652 [RFC5652]. A digest is computed on the content (as specified in Section 6.1) with a signer-specific message-digest algorithm. The certificates field contains the chain of X.509 V3 digital certificates from each authorizing entity. The certificate revocation list is

defined in the crls field. The digest output is digitally signed following Section 8 of RFC 3447 [RFC3447], using the signer's private key.

When the AUTH_ENT_ID is of type X509_V3_CERT, verification at the verifying network element (PDP or router) MUST be done following these steps:

- Parse the X.509 V3 certificate to extract the distinguished name of the issuer of the certificate.
- Certification Path Validation is performed as defined in Section 6 of RFC 5280 [RFC5280].
- Parse through the Certificate Revocation list to verify that the received certificate is not listed.
- Once the X.509 V3 certificate is validated, the public key of the authorizing entity can be extracted from the certificate.
- Extract the digest algorithm and the length of the digested data by parsing the CMS (Cryptographic Message Syntax) signed-data.
- The recipient independently computes the message digest. This message digest and the signer's public key are used to verify the signature value.

This verification ensures integrity, non-repudiation, and data origin.

4.3.1.2. PGP Digital Certificates

When the AUTH_ENT_ID is of type PGP_CERT, AUTHENTICATION_DATA MUST be generated by the authorizing entity following these steps:

AUTHENTICATION_DATA contains a Signature Packet as defined in Section 5.2.3 of RFC 4880 [RFC4880]. In summary:

- Compute the hash of all data in the SESSION_AUTH object up to the AUTHENTICATION_DATA.
- The hash output is digitally signed following Section 8 of RFC 3447, using the signer's private key.

When the AUTH_ENT_ID is of type PGP_CERT, verification MUST be done by the verifying network element (PDP or router) following these steps:

- Validate the certificate.

- Once the PGP certificate is validated, the public key of the authorizing entity can be extracted from the certificate.

- Extract the hash algorithm and the length of the hashed data by parsing the PGP signature packet.

- The recipient independently computes the message digest. This message digest and the signer's public key are used to verify the signature value.

This verification ensures integrity, non-repudiation, and data origin.

4.4. HMAC Signed

A SESSION_AUTH object that carries an AUTH_ENT_ID of HMAC_SIGNED is used as integrity protection for NSLP messages. The SESSION_AUTH object MUST contain the following attributes:

- SOURCE_ADDR: the source address of the entity that created the HMAC

- START_TIME: the timestamp when the HMAC signature was calculated. This MUST be different for any two messages in sequence in order to prevent replay attacks. The NTP timestamp currently provides a resolution of 200 picoseconds, which should be sufficient.

- NSLP_OBJECT_LIST: this attribute lists all NSLP objects that are included in HMAC calculation.

- AUTHENTICATION_DATA: this attribute contains the Key-ID that is used for HMAC calculation as well as the HMAC data itself [RFC2104].

The key used for HMAC calculation must be exchanged securely by some other means, e.g., a Kerberos Ticket or pre-shared manual installation etc. The Key-ID in the AUTHENTICATION_DATA allows the reference to the appropriate key and also to periodically change signing keys within a session. The key length MUST be at least 64 bits, but it is ideally longer in order to defend against brute-force attacks during the key validity period. For scalability reasons it is suggested to use a per-user key for signing NSLP messages, but using a per-session key is possible, too, at the cost of a per-session key exchange. A per-user key allows for verification of the authenticity of the message and thus provides a basis for a session-based per-user authorization. It is RECOMMENDED to periodically

change the shared key in order to prevent eavesdroppers from performing brute-force off-line attacks on the shared key. The actual hash algorithm used in the HMAC computation is specified by the "Transform ID" field (given as Transform Type 3 of the IKEv2 registry [RFC5996]). The hash algorithm MUST be chosen consistently between the object creator and the NN verifying the HMAC; this can be accomplished by out-of-band mechanisms when the shared key is exchanged.

Figure 5 shows an example of an object that is used for integrity protection of NSLP messages.

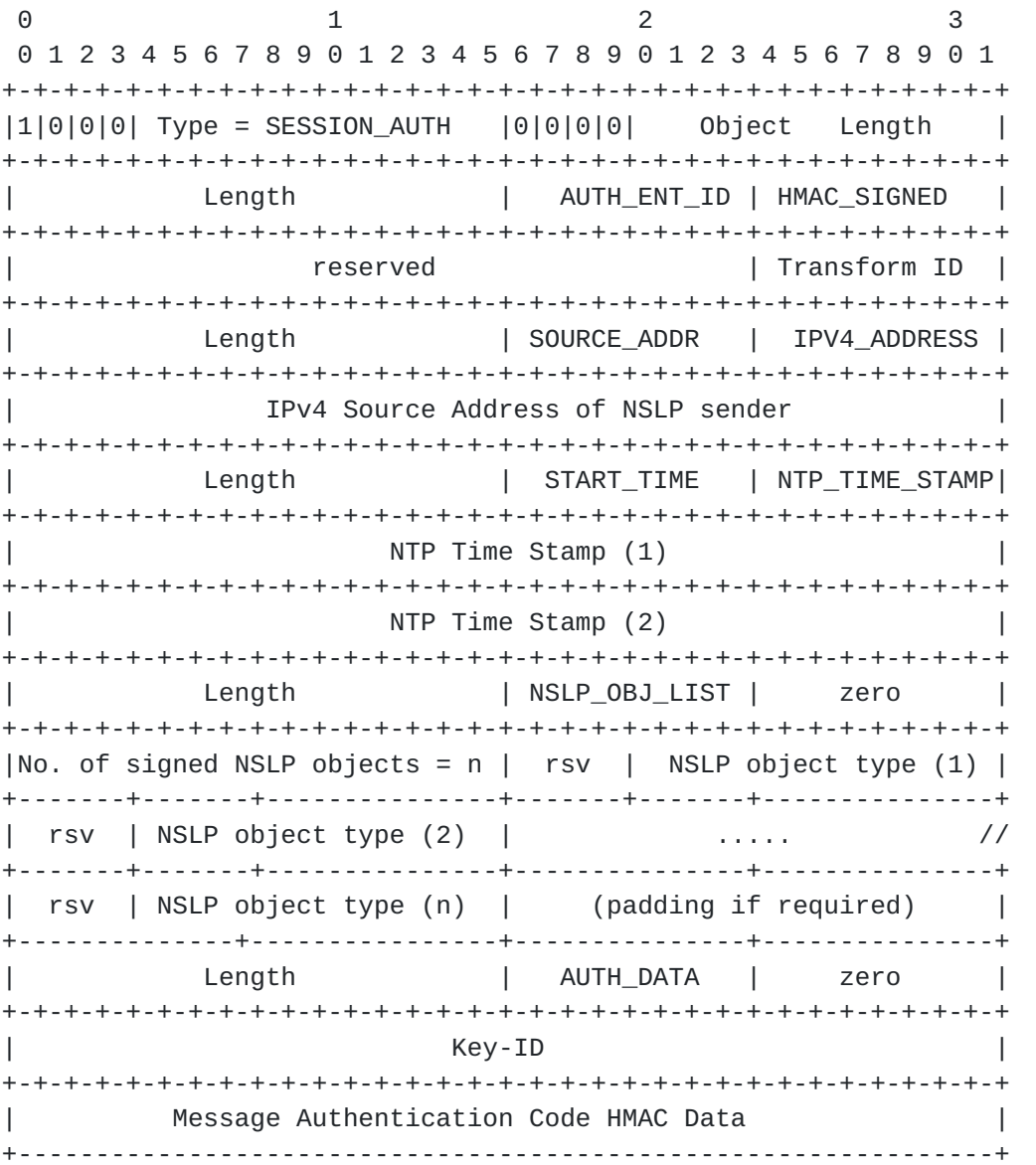


Figure 5: Example of a SESSION_AUTH_OBJECT That Provides Integrity Protection for NSLP Messages

5. Framework

RFC 3521 [RFC3521] describes a framework in which the SESSION_AUTH object may be utilized to transport information required for authorizing resource reservation for data flows (e.g., media flows). RFC 3521 introduces four different models:

1. The coupled model

2. The associated model with one policy server

3. The associated model with two policy servers

4. The non-associated model

The fields that are required in a SESSION_AUTH object depend on which of the models is used.

5.1. The Coupled Model

In the coupled model, the only information that MUST be included in the SESSION_AUTH object is the SESSION_ID; it is used by the Authorizing Entity to correlate the resource reservation request with the media authorized during session setup. Since the End Host is assumed to be untrusted, the Policy Server SHOULD take measures to ensure that the integrity of the SESSION_ID is preserved in transit; the exact mechanisms to be used and the format of the SESSION_ID are implementation dependent.

5.2. The Associated Model with One Policy Server

In this model, the contents of the SESSION_AUTH object MUST include:

- A session identifier - SESSION_ID. This is information that the authorizing entity can use to correlate the resource request with the data flows authorized during session setup.

- The identity of the authorizing entity - AUTH_ENT_ID. This information is used by an NN to determine which authorizing entity (Policy Server) should be used to solicit resource policy decisions.

In some environments, an NN may have no means for determining if the identity refers to a legitimate Policy Server within its domain. In order to protect against redirection of authorization requests to a bogus authorizing entity, the SESSION_AUTH MUST also include:

AUTHENTICATION_DATA. This authentication data is calculated over all other fields of the SESSION_AUTH object.

5.3. The Associated Model with Two Policy Servers

The content of the SESSION_AUTH object is identical to the associated model with one policy server.

5.4. The Non-Associated Model

In this model, the SESSION_AUTH object MUST contain sufficient information to allow the Policy Server to make resource policy decisions autonomously from the authorizing entity. The object is created using information about the session by the authorizing entity. The information in the SESSION_AUTH object MUST include:

- Initiating party's IP address or Identity (e.g., FQDN) - SOURCE_ADDR X-Type
- Responding party's IP address or Identity (e.g., FQDN) - DEST_ADDR X-Type
- The authorization lifetime - START_TIME X-Type
- The identity of the authorizing entity to allow for validation of the token in shared symmetric key and Kerberos schemes - AUTH_ENT_ID X-Type
- The credentials of the authorizing entity in a public-key scheme - AUTH_ENT_ID X-Type
- Authentication data used to prevent tampering with the SESSION_AUTH object - AUTHENTICATION_DATA X-Type

Furthermore, the SESSION_AUTH object MAY contain:

- The lifetime of (each of) the media stream(s) - END_TIME X-Type
- Initiating party's port number - SOURCE_ADDR X-Type
- Responding party's port number - DEST_ADDR X-Type

All SESSION_AUTH fields MUST match with the resource request. If a field does not match, the request SHOULD be denied.

6. Message Processing Rules

This section discusses the message processing related to the SESSION_AUTH object. Details of the processing of the SESSION_AUTH object within QoS NSLP and NATFW NSLP are described. New NSLP protocols should use the same logic in making use of the SESSION_AUTH object.

6.1. Generation of the SESSION_AUTH by an Authorizing Entity

1. Generate the SESSION_AUTH object with the appropriate contents as specified in Section 3.

2. If authentication is needed, the entire SESSION_AUTH object is constructed, excluding the length, type, and SubType fields of the SESSION_AUTH field. Note that the message MUST include a START_TIME to prevent replay attacks. The output of the authentication algorithm, plus appropriate header information, is appended as the AUTHENTICATION_DATA attribute to the SESSION_AUTH object.

6.2. Processing within the QoS NSLP

The SESSION_AUTH object may be used with QoS NSLP QUERY and RESERVE messages to authorize the query operation for network resources, and a resource reservation request, respectively.

Moreover, the SESSION_AUTH object may also be used with RESPONSE messages in order to indicate that the authorizing entity changed the original request. For example, the session start or end times may have been modified, or the client may have requested authorization for all ports, but the authorizing entity only allowed the use of certain ports.

If the QoS NSIS Initiator (QNI) receives a RESPONSE message with a SESSION_AUTH object, the QNI MUST inspect the SESSION_AUTH object to see which authentication attribute was changed by an authorizing entity. The QNI SHOULD also silently accept SESSION_AUTH objects in the RESPONSE message that do not indicate any change to the original authorization request.

6.2.1. Message Generation

A QoS NSLP message is created as specified in [RFC5974].

1. The policy element received from the authorizing entity MUST be copied without modification into the SESSION_AUTH object.

2. The SESSION_AUTH object (containing the policy element) is inserted in the NSLP message in the appropriate place.

6.2.2. Message Reception

The QoS NSLP message is processed as specified in [RFC5974] with the following modifications.

1. If the QoS NSIS Entity (QNE) is policy aware then it SHOULD use the Diameter QoS application or the RADIUS QoS protocol to communicate with the PDP. To construct the AAA message it is necessary to extract the SESSION_AUTH object and the QoS-related objects from the QoS NSLP message and to craft the respective RADIUS or Diameter message. The message processing and object format are described in the respective RADIUS or Diameter QoS protocol, respectively. If the QNE is policy unaware, then it ignores the policy data objects and continues processing the NSLP message.

2. If the response from the PDP is negative, the request must be rejected. A negative response in RADIUS is an Access-Reject, and in Diameter is based on the 'DIAMETER_SUCCESS' value in the Result-Code AVP of the QoS-Authz-Answer (QAA) message. The QNE must construct and send a RESPONSE message with the status of the authorization failure as specified in [RFC5974].

3. Continue processing the NSIS message.

6.2.3. Authorization (QNE or PDP)

1. Retrieve the policy element from the SESSION_AUTH object. Check the AUTH_ENT_ID type and SubType fields and return an error if the identity type is not supported.

2. Verify the message integrity.

   * Shared symmetric key authentication: The QNE or PDP uses the AUTH_ENT_ID field to consult a table keyed by that field. The table should identify the cryptographic authentication algorithm to be used along with the expected length of the authentication data and the shared symmetric key for the authorizing entity. Verify that the indicated length of the authentication data is consistent with the configured table entry and validate the authentication data.

   * Public Key: Validate the certificate chain against the trusted Certificate Authority (CA) and validate the message signature using the public key.

* HMAC signed: The QNE or PDP uses the Key-ID field of the AUTHENTICATION_DATA attribute to consult a table keyed by that field. The table should identify the cryptographic authentication algorithm to be used along with the expected length of the authentication data and the shared symmetric key for the authorizing entity. Verify that the indicated length of the authentication data is consistent with the configured table entry and validate the integrity of the parts of the NSLP message, i.e., session ID, MRI, NSLPID, and all other NSLP elements listed in the NSLP_OBJECT_LIST authentication data as well as the SESSION_AUTH object contents (cf. Section 6.4).

* Kerberos: If AUTHENTICATION_DATA contains an encapsulated KRB_CRED message (cf. Section 4.2), the integrity of the KRB_CRED message can be verified within Kerberos itself. Moreover, if the same NSLP message contains another SESSION_AUTH object using HMAC_SIGNED, the latter can be used to verify the message integrity as described above.

3. Once the identity of the authorizing entity and the validity of the service request have been established, the authorizing router/PDP MUST then consult its authorization policy in order to determine whether or not the specific request is finally authorized (e.g., based on available credits and on information in the subscriber's database). To the extent to which these access control decisions require supplementary information, routers/PDPs MUST ensure that supplementary information is obtained securely.

4. Verify that the requested resources do not exceed the authorized QoS.

6.2.4. Error Signaling

When the PDP (e.g., a RADIUS or Diameter server) fails to verify the policy element, the appropriate actions described in the respective AAA document need to be taken.

The QNE node MUST return a RESPONSE message with the INFO_SPEC error code 'Authorization failure' as defined in the QoS NSLP specification [RFC5974]. The QNE MAY include an INFO_SPEC Object Value Info to indicate which SESSION_AUTH attribute created the error.

6.3. Processing with the NATFW NSLP

This section presents processing rules for the NATFW NSLP [RFC5973].

6.3.1. Message Generation

A NATFW NSLP message is created as specified in [RFC5973].

1. The policy element received from the authorizing entity MUST be copied without modification into the SESSION_AUTH object.

2. The SESSION_AUTH object (containing the policy element) is inserted in the NATFW NSLP message in the appropriate place.

6.3.2. Message Reception

The NATFW NSLP message is processed as specified in [RFC5973] with the following modifications.

1. If the router is policy aware, then it SHOULD use the Diameter application or the RADIUS protocol to communicate with the PDP. To construct the AAA message, it is necessary to extract the SESSION_AUTH object and the objects related to NATFW policy rules from the NSLP message and to craft the respective RADIUS or Diameter message. The message processing and object format is described in the respective RADIUS or Diameter protocols. If the router is policy unaware, then it ignores the policy data objects and continues processing the NSLP message.

2. Reject the message if the response from the PDP is negative. A negative response in RADIUS is an Access-Reject, and in Diameter is based on the 'DIAMETER_SUCCESS' value in the Result-Code AVP.

3. Continue processing the NSIS message.

6.3.3. Authorization (Router/PDP)

1. Retrieve the policy element from the SESSION_AUTH object. Check the AUTH_ENT_ID type and SubType fields and return an error if the identity type is not supported.

2. Verify the message integrity.

   * Shared symmetric key authentication: The network router/PDP uses the AUTH_ENT_ID field to consult a table keyed by that field. The table should identify the cryptographic authentication algorithm to be used, along with the expected length of the authentication data and the shared symmetric key for the authorizing entity. Verify that the indicated length of the authentication data is consistent with the configured table entry and validate the authentication data.

* Public Key: Validate the certificate chain against the trusted Certificate Authority (CA) and validate the message signature using the public key.

* HMAC signed: The QNE or PDP uses the Key-ID field of the AUTHENTICATION_DATA attribute to consult a table keyed by that field. The table should identify the cryptographic authentication algorithm to be used along with the expected length of the authentication data and the shared symmetric key for the authorizing entity. Verify that the indicated length of the authentication data is consistent with the configured table entry and validate the integrity of parts of the NSLP message, i.e., session ID, MRI, NSLPID, and all other NSLP elements listed in the NSLP_OBJECT_LIST authentication data as well as the SESSION_AUTH object contents (cf. Section 6.4).

* Kerberos: If AUTHENTICATION_DATA contains an encapsulated KRB_CRED message (cf. Section 4.2), the integrity of the KRB_CRED message can be verified within Kerberos itself. Moreover, an if the same NSLP message contains another SESSION_AUTH object using HMAC_SIGNED, the latter can be used to verify the message integrity as described above.

3. Once the identity of the authorizing entity and the validity of the service request have been established, the authorizing router/PDP MUST then consult its authorization policy in order to determine whether or not the specific request is authorized. To the extent to which these access control decisions require supplementary information, routers/PDPs MUST ensure that supplementary information is obtained securely.

6.3.4. Error Signaling

When the PDP (e.g., a RADIUS or Diameter server) fails to verify the SESSION_AUTH object, the appropriate actions described in the respective AAA document need to be taken. The NATFW NSLP node MUST return an error message of class 'Permanent failure' (0x5) with error code 'Authorization failed' (0x02).

6.4. Integrity Protection of NSLP Messages

The SESSION_AUTH object can also be used to provide an integrity protection for every NSLP signaling message, thereby also authenticating requests or responses. Assume that a user has deposited a shared key at some NN. This NN can then verify the integrity of every NSLP message sent by the user to the NN. Based on this authentication, the NN can apply authorization policies to actions like resource reservations or opening of firewall pinholes.

The sender of an NSLP message creates a SESSION_AUTH object that contains the AUTH_ENT_ID attribute set to HMAC_SIGNED (cf. Section 4.4) and hashes with the shared key over all NSLP objects that need to be protected and lists them in the NSLP_OBJECT_LIST. The SESSION_AUTH object itself is also protected by the HMAC. By inclusion of the SESSION_AUTH object into the NSLP message, the receiver of this NSLP message can verify its integrity if it has the suitable shared key for the HMAC. Any response to the sender should also be protected by inclusion of a SESSION_AUTH object in order to prevent attackers from sending unauthorized responses on behalf of the real NN.

If a SESSION_AUTH object is present that has an AUTH_ENT_ID attribute set to HMAC_SIGNED, the integrity of all NSLP elements listed in the NSLP_OBJECT_LIST has to be checked, including the SESSION_AUTH object contents itself. Furthermore, session ID, MRI, and NSLPID have to be included into the HMAC calculation, too, as specified in Section 3.2.7. The key that is used to calculate the HMAC is referred to by the Key-ID included in the AUTHENTICATION_DATA attribute. If the provided timestamp in START_TIME is not recent enough or the calculated HMAC differs from the one provided in AUTHENTICATION_DATA, the message must be discarded silently and an error should be logged locally.

7. Security Considerations

This document describes a mechanism for session authorization to prevent theft of service. There are three types of security issues to consider: protection against replay attacks, integrity of the SESSION_AUTH object, and the choice of the authentication algorithms and keys.

The first issue, replay attacks, MUST be prevented. In the non-associated model, the SESSION_AUTH object MUST include a START_TIME field, and the NNs as well as Policy Servers MUST support NTP to ensure proper clock synchronization. Failure to ensure proper clock synchronization will allow replay attacks since the clocks of the different network entities may not be in sync. The start time is used to verify that the request is not being replayed at a later time. In all other models, the SESSION_ID is used by the Policy Server to ensure that the resource request successfully correlates with records of an authorized session. If a SESSION_AUTH object is replayed, it MUST be detected by the policy server (using internal algorithms), and the request MUST be rejected.

The second issue, the integrity of the SESSION_AUTH object, is preserved in untrusted environments by including the AUTHENTICATION_DATA attribute in such environments.

In environments where shared symmetric keys are possible, they should be used in order to keep the SESSION_AUTH object size to a strict minimum, e.g., when wireless links are used. A secondary option would be Public Key Infrastructure (PKI) authentication, which provides a high level of security and good scalability. However, PKI authentication requires the presence of credentials in the SESSION_AUTH object, thus impacting its size.

The SESSION_AUTH object can also serve to protect the integrity of NSLP message parts by using the HMAC_SIGNED Authentication Data as described in Section 6.4.

When shared keys are used, e.g., in AUTHENTICATION_DATA (cf. Section 4.1) or in conjunction with HMAC_SIGNED (cf. Section 4.4), it is important that the keys are kept secret, i.e., they must be exchanged, stored, and managed in a secure and confidential manner, so that no unauthorized party gets access to the key material. If the key material is disclosed to an unauthorized party, authentication and integrity protection are ineffective.

Furthermore, security considerations for public-key mechanisms using the X.509 certificate mechanisms described in [RFC5280] apply. Similarly, security considerations for PGP (Pretty Good Privacy) described in [RFC4880] apply.

Further security issues are outlined in RFC 4081 [RFC4081].

8. IANA Considerations

The SESSION_AUTH_OBJECT NSLP Message Object type is specified as 0x016.

This document specifies an 8-bit Session authorization attribute type (X-Type) field as well as 8-bit SubType fields per X-Type, for which IANA has created and will maintain corresponding sub-registries for the NSLP Session Authorization Object.

Initial values for the X-Type registry and the registration procedures according to [RFC5226] are as follows:

Registration Procedure:
   Specification Required

X-Type	Description
0	Reserved
1	AUTH_ENT_ID
2	SESSION_ID
3	SOURCE_ADDR
4	DEST_ADDR
5	START_TIME
6	END_TIME
7	NSLP_OBJECT_LIST
8	AUTHENTICATION_DATA
9-127	Unassigned
128-255	Reserved for Private or Experimental Use

In the following, registration procedures and initial values for the SubType registries are specified.

Sub-registry: AUTH_ENT_ID (X-Type 1) SubType values

Registration Procedure:
Specification Required

Registry:

SubType	Description
0	Reserved
1	IPV4_ADDRESS
2	IPV6_ADDRESS
3	FQDN
4	ASCII_DN
5	UNICODE_DN
6	URI
7	KRB_PRINCIPAL
8	X509_V3_CERT
9	PGP_CERT
10	HMAC_SIGNED
11-127	Unassigned
128-255	Reserved for Private or Experimental Use

Sub-registry: SOURCE_ADDR (X-Type 3) SubType values

Registration Procedure:
Specification Required

Registry:

SubType	Description
0	Reserved
1	IPV4_ADDRESS
2	IPV6_ADDRESS
3	UDP_PORT_LIST
4	TCP_PORT_LIST
5	SPI
6-127	Unassigned
128-255	Reserved for Private or Experimental Use

Sub-registry: DEST_ADDR (X-Type 4) SubType values

Registration Procedure:
Specification Required

Registry:

0	Reserved
1	IPV4_ADDRESS
2	IPV6_ADDRESS
3	UDP_PORT_LIST
4	TCP_PORT_LIST
5	SPI
6-127	Unassigned
128-255	Reserved for Private or Experimental Use

Sub-registry: START_TIME (X-Type 5) SubType values

Registration Procedure:
Specification Required

Registry:

SubType	Description
0	Reserved
1	NTP_TIMESTAMP
2-127	Unassigned
128-255	Reserved for Private or Experimental Use

Sub-registry: END_TIME (X-Type 6) SubType values

Registration Procedure:
Specification Required

Registry:

SubType	Description
0	Reserved
1	NTP_TIMESTAMP
2-127	Unassigned
128-255	Reserved for Private or Experimental Use

9. Acknowledgments

We would like to thank Xioaming Fu and Lars Eggert for providing reviews and comments. Helpful comments were also provided by Gen-ART reviewer Ben Campbell, as well as Sean Turner and Tim Polk from the Security Area. This document is largely based on the RFC 3520 [RFC3520] and credit therefore goes to the authors of RFC 3520 -- namely, Louis-Nicolas Hamer, Brett Kosinski, Bill Gage, and Hugh Shieh. Part of this work was funded by Deutsche Telekom Laboratories within the context of the BMBF-funded ScaleNet project.

10. References

10.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3447] Jonsson, J. and B. Kaliski, "Public-Key Cryptography Standards (PKCS) #1: RSA Cryptography Specifications Version 2.1", RFC 3447, February 2003.

[RFC5905] Mills, D., Martin, J., Burbank, J., and W. Kasch, "Network Time Protocol Version 4: Protocol and Algorithms Specification", RFC 5905, June 2010.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, October 2010.

[RFC5973] Stiemerling, M., Tschofenig, H., Aoun, C., and E. Davies, "NAT/Firewall NSIS Signaling Layer Protocol (NSLP)", RFC 5973, October 2010.

[RFC5974] Manner, J., Karagiannis, G., and A. McDonald, "NSIS Signaling Layer Protocol (NSLP) for Quality-of-Service Signaling", RFC 5974, October 2010.

[RFC5996] Kaufman, C., Hoffman, P., Nir, Y., and P. Eronen, "Internet Key Exchange Protocol Version 2 (IKEv2)", RFC 5996, September 2010.

10.2. Informative References

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[RFC2104] Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, February 1997.

[RFC3520] Hamer, L-N., Gage, B., Kosinski, B., and H. Shieh, "Session Authorization Policy Element", RFC 3520, April 2003.

[RFC3521] Hamer, L-N., Gage, B., and H. Shieh, "Framework for Session Set-up with Media Authorization", RFC 3521, April 2003.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4080] Hancock, R., Karagiannis, G., Loughney, J., and S. Van den Bosch, "Next Steps in Signaling (NSIS): Framework", RFC 4080, June 2005.

[RFC4081] Tschofenig, H. and D. Kroeselberg, "Security Threats for Next Steps in Signaling (NSIS)", RFC 4081, June 2005.

[RFC4120] Neuman, C., Yu, T., Hartman, S., and K. Raeburn, "The Kerberos Network Authentication Service (V5)", RFC 4120, July 2005.

[RFC4514] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): String Representation of Distinguished Names", RFC 4514, June 2006.

[RFC4868] Kelly, S. and S. Frankel, "Using HMAC-SHA-256, HMAC-SHA-384, and HMAC-SHA-512 with IPsec", RFC 4868, May 2007.

[RFC4880] Callas, J., Donnerhacke, L., Finney, H., Shaw, D., and R. Thayer, "OpenPGP Message Format", RFC 4880, November 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5652] Housley, R., "Cryptographic Message Syntax (CMS)", STD 70, RFC 5652, September 2009.

Authors' Addresses

Jukka Manner
Aalto University
Department of Communications and Networking (Comnet)
P.O. Box 13000
Aalto FI-00076
Finland

Phone: +358 9 470 22481
EMail: jukka.manner@tkk.fi

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

Phone: +49 (0) 6221 4342 113
EMail: martin.stiemerling@neclab.eu
URI: http://www.stiemerling.org

Hannes Tschofenig
Nokia Siemens Networks
Linnoitustie 6
Espoo 02600
Finland

Phone: +358 (50) 4871445
EMail: Hannes.Tschofenig@gmx.net
URI: http://www.tschofenig.priv.at

Roland Bless (editor)
Karlsruhe Institute of Technology
Institute of Telematics
Zirkel 2, Building 20.20
P.O. Box 6980
Karlsruhe 76049
Germany

Phone: +49 721 608 46413
EMail: roland.bless@kit.edu
URI: http://tm.kit.edu/~bless