Network Working Group	D. Harrington
Internet-Draft	Huawei Technologies (USA)
Intended status: Standards Track	W. Hardaker
Expires: April 10, 2009	Sparta, Inc.
	October 07, 2008

Transport Security Model for SNMP
draft-ietf-isms-transport-security-model-09

Status of This Memo

Abstract

This memo describes a Transport Security Model for the Simple Network Management Protocol.

This memo also defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP based internets. In particular it defines objects for monitoring and managing the Transport Security Model for SNMP.

Table of Contents

1. Introduction

TOC

This memo describes a Transport Security Model for the Simple Network Management Protocol, for use with secure Transport Models in the Transport Subsystem [I-D.ietf-isms-tmsm] (Harrington, D. and J. Schoenwaelder, "Transport Subsystem for the Simple Network Management Protocol (SNMP)," May 2009.).

This memo also defines a portion of the Management Information Base (MIB) for use with network management protocols in TCP/IP based internets. In particular it defines objects for monitoring and managing the Transport Security Model for SNMP.

It is important to understand the SNMP architecture and the terminology of the architecture to understand where the Transport Security Model described in this memo fits into the architecture and interacts with other subsystems and models within the architecture. It is expected that reader will have also read and understood RFC3411 [RFC3411] (Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," December 2002.), RFC3412 [RFC3412] (Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)," December 2002.), RFC3413 [RFC3413] (Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications," December 2002.), and RFC3418 [RFC3418] (Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)," December 2002.).

1.1. The Internet-Standard Management Framework

TOC

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410] (Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework," December 2002.).

Managed objects are accessed via a virtual information store, termed the Management Information Base or MIB. MIB objects are generally accessed through the Simple Network Management Protocol (SNMP). Objects in the MIB are defined using the mechanisms defined in the Structure of Management Information (SMI). This memo specifies a MIB module that is compliant to the SMIv2, which is described in STD 58, RFC 2578 [RFC2578] (McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., “Structure of Management Information Version 2 (SMIv2),” April 1999.), STD 58, RFC 2579 [RFC2579] (McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., “Textual Conventions for SMIv2,” April 1999.) and STD 58, RFC 2580 [RFC2580] (McCloghrie, K., Perkins, D., and J. Schoenwaelder, “Conformance Statements for SMIv2,” April 1999.).

1.2. Conventions

TOC

For consistency with SNMP-related specifications, this document favors terminology as defined in STD62 rather than favoring terminology that is consistent with non-SNMP specifications that use different variations of the same terminology. This is consistent with the IESG decision to not require the SNMPv3 terminology be modified to match the usage of other non-SNMP specifications when SNMPv3 was advanced to Full Standard.

Authentication in this document typically refers to the English meaning of "serving to prove the authenticity of" the message, not data source authentication or peer identity authentication.

The terms "manager" and "agent" are not used in this document, because in the RFC 3411 architecture, all SNMP entities have the capability of acting as either manager or agent or both depending on the SNMP applications included in the engine. Where distinction is required, the application names of Command Generator, Command Responder, Notification Originator, Notification Receiver, and Proxy Forwarder are used. See "SNMP Applications" [RFC3413] (Levi, D., Meyer, P., and B. Stewart, “Simple Network Management Protocol (SNMP) Applications,” December 2002.) for further information.

While security protocols frequently refer to a user, the terminology used in RFC3411 [RFC3411] (Harrington, D., Presuhn, R., and B. Wijnen, “An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,” December 2002.) and in this memo is "principal". A principal is the "who" on whose behalf services are provided or processing takes place. A principal can be, among other things, an individual acting in a particular role; a set of individuals, with each acting in a particular role; an application or a set of applications, or a combination of these within an administrative domain.

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 [RFC2119] (Bradner, S., “Key words for use in RFCs to Indicate Requirement Levels,” March 1997.).

TOC

1.3. Modularity

The reader is expected to have read and understood the description of the SNMP architecture, as defined in [RFC3411] (Harrington, D., Presuhn, R., and B. Wijnen, “An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,” December 2002.), and the architecture extension specified in "Transport Subsystem for the Simple Network Management Protocol" [I-D.ietf-isms-tmsm] (Harrington, D. and J. Schoenwaelder, “Transport Subsystem for the Simple Network Management Protocol (SNMP),” May 2009.), which enables the use of external "lower layer transport" protocols to provide message security, tied into the SNMP architecture through the Transport Subsystem. The Transport Security Model is designed to work with such lower-layer secure Transport Models.
In keeping with the RFC 3411 design decisions to use self-contained documents, this memo includes the elements of procedure plus associated MIB objects which are needed for processing the Transport Security Model for SNMP. These MIB objects SHOULD NOT be referenced in other documents. This allows the Transport Security Model to be designed and documented as independent and self-contained, having no direct impact on other modules, and allowing this module to be upgraded and supplemented as the need arises, and to move along the standards track on different time-lines from other modules.
This modularity of specification is not meant to be interpreted as imposing any specific requirements on implementation.

1.4. Motivation

TOC

This memo describes a Security Model to make use of Transport Models that use lower layer secure transports and existing and commonly deployed security infrastructures. This Security Model is designed to meet the security and operational needs of network administrators, maximize usability in operational environments to achieve high deployment success and at the same time minimize implementation and deployment costs to minimize the time until deployment is possible.

1.5. Constraints

TOC

The design of this SNMP Security Model is also influenced by the following constraints:

1. In times of network stress, the security protocol and its underlying security mechanisms SHOULD NOT depend solely upon the ready availability of other network services (e.g., Network

Time Protocol (NTP) or Authentication, Authorization, and Accounting (AAA) protocols).

2. When the network is not under stress, the Security Model and its underlying security mechanisms MAY depend upon the ready availability of other network services.

3. It may not be possible for the Security Model to determine when the network is under stress.

4. A Security Model should require no changes to the SNMP architecture.

5. A Security Model should require no changes to the underlying security protocol.

---

2. How the Transport Security Model Fits in the Architecture

TOC

The Transport Security Model is designed to fit into the RFC3411 architecture as a Security Model in the Security Subsystem, and to utilize the services of a secure Transport Model.
A cache, referenced by tmStateReference, is used to pass information between the Transport Security Model and a Transport Model, and vice versa. If the Transport Security Model is used with an insecure Transport Model, then the cache will not exist or not be populated with security parameters, which will cause the Transport Security Model to return an error (see section 5.2) If another Security Model (eg Community-based Security Model) is used with a secure Transport Model, then the cache may be populated but the other Security Model may be unaware of the cache and ignore its contents (eg deriving the securityName from the Community name in the message instead of deriving it from the tmSecurityName in the tmStateReference cache).
For incoming messages, a secure Transport Model creates a tmStateReference cache including a tmTransport, tmAddress, tmSecurityName and a tmTransportSecurityLevel, and it MAY include transport-specific information. The Transport Security Model will determine the security-model-independent securityName and securityLevel, and will verify that tmTransportSecurityLevel is at least as strong as the requested securityLevel. As with all security models, the securityName represents the principal on whose behalf a received SNMP message claims to have been generated. It is not possible to assure the specific principal that originated a received SNMP message; rather, it is the principal on whose behalf the message was originated that is authenticated.
For outgoing messages, the Transport Security Model creates a cache containing the transportDomain, transportAddress, and a tmSecurityName

and tmRequestedSecurityLevel and passes the tmStateReference cache to the specified Transport Model.

To maintain the RFC3411 modularity, the Transport Model does not know which securityModel will be used for an incoming message; the Message Processing Model will determine the securityModel to be used, in a Message Processing Model dependent manner.

2.1. Security Capabilities of this Model

TOC

2.1.1. Threats

TOC

The Transport Security Model, when used with suitable secure Transport Models, provides protection against the threats identified by the RFC 3411 architecture [RFC3411] (Harrington, D., Presuhn, R., and B. Wijnen, “An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,” December 2002.).

Which threats are addressed depends on the Transport Model. The Transport Security Model does not address any threats itself, but delegates that responsibility to a secure Transport Model.

The Transport Security Model is called a Security Model to be compatible with the RFC3411 architecture. However, this Security Model does not provide security mechanisms such as authentication and encryption itself, so it SHOULD always be used with a Transport Model that provides appropriate security.

2.1.2. Security Levels

TOC

The RFC 3411 architecture recognizes three levels of security:

> - without authentication and without privacy (noAuthNoPriv)
>
> - with authentication but without privacy (authNoPriv)
>
> - with authentication and with privacy (authPriv)

The model-independent securityLevel parameter is used to request specific levels of security for outgoing messages, and to assert that specific levels of security were applied during the transport and processing of incoming messages.

The transport layer algorithms used to provide security SHOULD NOT be exposed to the Transport Security Model, as the Transport Security

Model has no mechanisms by which it can test whether an assertion made by a Transport Model is accurate.
The Transport Security Model trusts that the underlying secure transport connection has been properly configured to support security characteristics at least as strong as reported in tmTransportSecurityLevel.

2.2. No Sessions

TOC

The Transport Security Model will associate state regarding each message and each known remote engine with a combination of transportDomain, transportAddress, securityName, securityModel, and securityLevel.
The Transport Security Model does not recognize sessions of any kind, although they may be supported by a transport model.

2.3. Coexistence

TOC

There are two primary factors which determine whether Security Models can coexist. First, there must be a mechanism to select different Security Models at run-time. Second, the processing of one Security Model should not impact the processing of another Security Model.
In the RFC3411 architecture, a Message Processing Model determines which Security Model should be called. As of this writing, IANA has registered four Message Processing Models (SNMPv1, SNMPv2c, SNMPv2u/SNMPv2*, and SNMPv3) and three other Security Models (SNMPv1, SNMPv2c, and the User-based Security Model).
The SNMPv1 and SNMPv2c message processing described in RFC3584 (BCP 74) [RFC3584] (Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework," August 2003.) always selects the SNMPv1(1) Security Model for an SNMPv1 message, or the SNMPv2c(2) Security Model for an SNMPv2c message. Since there is no field in the message format that permits specifying a Security Model, RFC3584 message processing does not permit the selection of Security Models other than SNMPv1 or SNMPv2. Therefore, SNMPv1 or SNMPv2c messages that go through the SNMPv1 or SNMPv2 Message Processing Models **as defined in RFC3584** cannot use the Transport Security Model. (This does not mean an SNMPv1 or SNMPv2 message cannot use a secure transport model, only that the RFC3584 Message Processing Model will not invoke this security model.)
The SNMPv2u/SNMPv2* Message Processing Model is a historic artifact for which there is no existing IETF specification.

The SNMPv3 message processing defined in RFC3412 [RFC3412] (Case, J., Harrington, D., Presuhn, R., and B. Wijnen, “Message Processing and Dispatching for the Simple Network Management Protocol (SNMP),” December 2002.), extracts the securityModel from the msgSecurityModel field of an incoming SNMPv3Message. When the extracted value of msgSecurityModel is transportSecurityModel(YY), security processing is directed to the Transport Security Model. For an outgoing message to be secured using the Transport Security Model, msgSecurityModel should be set to transportSecurityModel(YY).
[-- NOTE to RFC editor: replace YY with actual IANA-assigned number, and remove this note.]
The Transport Security Model uses its own MIB module for processing to maintain independence from other Security Models. This allows the Transport Security Model to coexist with other Security Models, such as the User-based Security Model.
The Transport Security Model may work with multiple Transport Models, but the isAccessAllowed() application service interfaces (ASI) does not accept a value for the Transport Model. This security model MAY prepend a transport model identifier to securityName (if enabled by the snmpTsmConfigurationUsePrefix object), to allow different access control policies for identities authenticated by different Transport Models that use the Transport Security Model.
The MIB module defined in this memo allows an administrator to configure the Transport Security Model to disable support for specific transport models, and to prepend a transport model identifier to the securityName.

2.4. Security Parameter Passing

TOC

For outgoing messages, the Transport Security Model uses parameters provided by the SNMP application to look up or create an entry in the SNMP-TSM-MIB. From such an entry, the Transport Security Model creates a tmStateReference. The wholeMsg and the tmStateReference are passed to the appropriate Transport Model through a series of ASIs, as described in "Transport Subsystem for the Simple Network Management Protocol" [I-D.ietf-isms-tmsm] (Harrington, D. and J. Schoenwaelder, “Transport Subsystem for the Simple Network Management Protocol (SNMP),” May 2009.).
For incoming messages, a transport model accepts messages from the lower layer transport, and records the transport-related information and security-related information, including a human-readable name that represents the transport-authenticated identity, and a securityLevel that represents the security features provided during transport, in an implementation-dependent manner. From this information, the transport model creates a tmStateReference to pass to whichever security model is selected by the Message Processing Model. The wholeMsg and the

tmStateReference are passed to the appropriate Security Model through a series of ASIs, as described in "Transport Subsystem for the Simple Network Management Protocol" [I-D.ietf-isms-tmsm] (Harrington, D. and J. Schoenwaelder, “Transport Subsystem for the Simple Network Management Protocol (SNMP),” May 2009.).

2.5. Notifications and Proxy

TOC

The SNMP-TARGET-MIB module [RFC3413] (Levi, D., Meyer, P., and B. Stewart, “Simple Network Management Protocol (SNMP) Applications,” December 2002.) contains objects for defining management targets, including transportDomain, transportAddress, securityName, securityModel, and securityLevel parameters, for applications such as notifications and proxy. Transport type and address are configured in the snmpTargetAddrTable, and the securityModel, securityName, and securityLevel parameters are configured in the snmpTargetParamsTable. Note that if snmpTsmConfigurationUsePrefix is set to true then the securityName value defined in the SNMP-TARGET-MIB must contain the proper transport prefix for the configured target to function.

The default approach is for an administrator to statically configure this information to identify the targets authorized to receive notifications or perform proxy.

These parameters are passed to the security model using the appropriate ASIs. The Transport Security Model will use the parameters to determine how to create the appropriate tmStateReference for the selected transport model.

---

3. Cached Information and References

TOC

When performing SNMP processing, there are two levels of state information that may need to be retained: the immediate state linking a request-response pair, and potentially longer-term state relating to transport and security.

The RFC3411 architecture uses caches to maintain the short-term message state, and uses references in the ASIs to pass this information between subsystems.

This document defines the requirements for a cache to handle the longer-term transport state information, using a tmStateReference parameter to pass this information between subsystems.

To simplify the elements of procedure, the release of state information is not always explicitly specified. As a general rule, if state information is available when a message being processed gets discarded, the state related to that message SHOULD also be discarded. If state information is available when a relationship between engines is

severed, such as the closing of a transport session, the state information for that relationship SHOULD also be discarded.
Since the contents of a cache are meaningful only within an implementation, and not on-the-wire, the format of the cache and the LCD are implementation-specific.

3.1. securityStateReference

TOC

The securityStateReference parameter is defined in RFC3411. Its primary purpose is to provide a mapping between a request and the corresponding response. This cache is not accessible to Transport Models, and an entry is typically only retained for the lifetime of a request-response pair of messages.

3.2. tmStateReference

TOC

For each transport session, information about the transport security is stored in a cache. The tmStateReference parameter is used to pass model-specific and mechanism-specific parameters between the Transport subsystem and transport-aware Security Models.
The tmStateReference cache will typically remain valid for the duration of the transport session, and hence may be used for several messages.
Since this cache is only used within an implementation, and not on-the-wire, the precise contents and format are implementation- dependent.
However, for interoperability between Transport Models and transport-aware Security Models, entries in this cache must include at least the following fields:

> transportDomain
>
> transportAddress
>
> tmSecurityName
>
> tmRequestedSecurityLevel
>
> tmTransportSecurityLevel
>
> tmSameSecurity
>
> tmSessionID

3.2.1. Transport information

TOC

Information about the source of an incoming SNMP message is passed up from the Transport subsystem as far as the Message Processing subsystem. However these parameters are not included in the processIncomingMsg ASI defined in RFC3411, and hence this information is not directly available to the Security Model.
A transport-aware Security Model might wish to take account of the transport protocol and originating address when authenticating the request, and setting up the authorization parameters. It is therefore necessary for the Transport Model to include this information in the tmStateReference cache, so that it is accessible to the Security Model.

- transportDomain: the transport protocol (and hence the Transport Model) used to receive the incoming message

- transportAddress: the source of the incoming message.

Note that the ASIs used for processing an outgoing message all include explicit transportDomain and transportAddress parameters. These fields within the tmStateReference cache will typically not be used for outgoing messages.

3.2.2. securityName

TOC

There are actually three distinct "identities" that can be identified during the processing of an SNMP request over a secure transport:

- transport principal: the transport-authenticated identity, on whose behalf the secure transport connection was (or should be) established. This value is transport-, mechanism- and implementation- specific, and is only used within a given Transport Model.

- tmSecurityName: a human-readable name (in snmpAdminString format) representing this transport identity. This value is transport- and implementation-specific, and is only used (directly) by the Transport and Security Models.

- securityName: a human-readable name (in snmpAdminString format) representing the SNMP principal in a model-independent manner. Depending on configuration this value may be prefixed by a transport domain specific prefix followed by a ':' (ASCII 0x3a) to help distinguish between securityNames from different secure transports.

- Note that the transport principal may or may not be the same as the tmSecurityName. Similarly, the tmSecurityName may or may not be the same as the securityName as seen by the Application and Access Control subsystems. In particular, a non-transport-aware Security Model will ignore tmSecurityName completely when determining the SNMP securityName.
- However it is important that the mapping between the transport principal and the SNMP securityName (for transport-aware Security Models) is consistent and predictable, to allow configuration of suitable access control and the establishment of transport connections.

3.2.3. securityLevel

TOC

There are two distinct issues relating to security level as applied to secure transports. For clarity, these are handled by separate fields in the tmStateReference cache:

- tmTransportSecurityLevel: an indication from the Transport Model of the level of security offered by this session. The Security Model can use this to ensure that incoming messages were suitably protected before acting on them.
- tmRequestedSecurityLevel: an indication from the Security Model of the level of security required to be provided by the transport protocol. The Transport Model can use this to ensure that outgoing messages will not be sent over an insufficiently secure session.

3.2.4. Session Information

TOC

For security reasons, if a secure transport session is closed between the time a request message is received and the corresponding response message is sent, then the response message SHOULD be discarded, even if a new session has been established. The SNMPv3 WG decided that this should be a SHOULD architecturally, and it is a security-model-specific decision whether to REQUIRE this.
When processing an outgoing message, if tmSameSecurity is true, then the tmSessionID MUST match the current transport session, otherwise the message MUST be discarded, and the dispatcher notified that sending the message failed.

*tmSameSecurity: this flag is used by a transport-aware Security Model to indicate whether the Transport Model MUST enforce this restriction.

*tmSessionID: in order to verify whether the session has changed, the Transport Model must be able to compare the session used to receive the original request with the one to be used to send the response. This typically requires some form of session identifier. This value is only ever used by the Transport Model, so the format and interpretation of this field are model-specific and implementation-dependent.

3.3. Transport Security Model Cached Information

TOC

The Transport Security Model has specific responsibilities regarding the cached information.

3.3.1. tmStateReference

TOC

For each transport model, model- and mechanism-specific parameters for the transport security need to be stored in a local configuration datastore.
To enable a security model to correlate the identity used by specific transport-model and the model-independent identity referenced by applications, a mapping is provided in the MIB module defined in this memo.
The Transport Security Model REQUIRES that the security parameters used for a response are the same as those used for the corresponding request. It is transport-model-dependent and implementation-dependent how this is ensured at the transport layer.

3.3.2. securityStateReference

TOC

The securityStateReference parameter is defined in RFC3411. Its primary purpose is to provide a mapping between a request and the corresponding response. The Transport Security Model will conceptually add the tmStateReference to the securityStateReference cache, so the security model can map transport-specific security parameters for a request to its corresponding response. How the tmStateReference is added to the securityStateReference is implementation-specific.

4. Processing an Outgoing Message

TOC

An error indication may return an OID and value for an incremented counter and a value for securityLevel, and values for contextEngineID and contextName for the counter, and the securityStateReference if the information is available at the point where the error is detected.

4.1. Security Processing for an Outgoing Message

TOC

This section describes the procedure followed by the Transport Security Model.
The parameters needed for generating a message are supplied to the Security Model by the Message Processing Model via the generateRequestMsg() or the generateResponseMsg() ASI. The Transport Subsystem architectural extension has added the transportDomain, transportAddress, and tmStateReference parameters to the original RFC3411 ASIs.

```
statusInformation =                -- success or errorIndication
      generateRequestMsg(
      IN   messageProcessingModel  -- typically, SNMP version
      IN   globalData              -- message header, admin data
      IN   maxMessageSize          -- of the sending SNMP entity
      IN   transportDomain         -- (NEW) specified by application
      IN   transportAddress        -- (NEW) specified by application
      IN   securityModel           -- for the outgoing message
      IN   securityEngineID        -- authoritative SNMP entity
      IN   securityName            -- on behalf of this principal
      IN   securityLevel           -- Level of Security requested
      IN   scopedPDU               -- message (plaintext) payload
      OUT  securityParameters      -- filled in by Security Module
      OUT  wholeMsg                -- complete generated message
      OUT  wholeMsgLength          -- length of generated message
      OUT  tmStateReference        -- (NEW)  transport info
           )
```

```
   statusInformation = -- success or errorIndication
           generateResponseMsg(
           IN   messageProcessingModel  -- typically, SNMP version
           IN   globalData              -- message header, admin data
           IN   maxMessageSize          -- of the sending SNMP entity
           IN   transportDomain         -- (NEW) specified by application
           IN   transportAddress        -- (NEW) specified by application
           IN   securityModel           -- for the outgoing message
           IN   securityEngineID        -- authoritative SNMP entity
           IN   securityName            -- on behalf of this principal
           IN   securityLevel           -- Level of Security requested
           IN   scopedPDU               -- message (plaintext) payload
           IN   securityStateReference  -- reference to security state
                                        -- information from original
                                        -- request
           OUT  securityParameters      -- filled in by Security Module
           OUT  wholeMsg                -- complete generated message
           OUT  wholeMsgLength          -- length of generated message
           OUT  tmStateReference        -- (NEW) transport info
                )
```

4.2. Elements of Procedure for Outgoing Messages

TOC

1) If there is a securityStateReference, then this is a response message. Extract transportDomain, transportAddress, securityName, securityLevel, securityModel, and tmStateReference from the securityStateReference cache. Set the tmRequestedSecurityLevel to the value of the extracted securityLevel. The cachedSecurityData for this message can now be discarded. Set the tmSameSecurity parameter in the tmStateReference cache to true. Note that the securityName extracted from the cache will not contain any transport domain prefix.
2) If there is no securityStateReference create a tmStateReference cache with tmRequestedSecurityLevel set to the value of securityLevel, tmSameSecurity set to false, and tmSecurityName and tmTransportIdentity to:

> **a.** If the snmpTsmConfigurationUsePrefix object is set to false, then set the tmSecurityName set to the value of securityName.
>
> **b.** If the snmpTsmConfigurationUsePrefix object is set to true then use the transportDomain to look up its transport prefix in an implementation dependent way. (The transport prefix will likely be described in the transport domain object's DESCRIPTION for a given transportDomain.) Strip this prefix and the ':' (ASCII 0x3a) off of the securityName and place the results in the

> tmSecurityName value. If the securityName does not begin with the necessary prefix followed by a ':' (ASCII 0x3a) then snmpTsmInvalidPrefix counter is incremented and an error indication is returned to the calling module. If the prefix look-up using the transportDomain fails for any reason then the snmpTsmInvalidPrefixDomain counter is incremented and an error indication is returned to the calling module.

3) Fill in the securityParameters with a zero-length OCTET STRING ('0400').
4) Combine the message parts into a wholeMsg and calculate wholeMsgLength.
5) The wholeMsg, wholeMsgLength, securityParameters and tmStateReference are returned to the calling Message Processing Model with the statusInformation set to success.

---

5. Processing an Incoming SNMP Message

TOC

An error indication may return an OID and value for an incremented counter and a value for securityLevel, and values for contextEngineID and contextName for the counter, and the securityStateReference if the information is available at the point where the error is detected.

5.1. Security Processing for an Incoming Message

TOC

This section describes the procedure followed by the Transport Security Model whenever it receives an incoming message from a Message Processing Model. The ASI from a Message Processing Model to the Security Subsystem for a received message is:

```
statusInformation =  -- errorIndication or success
                         -- error counter OID/value if error
processIncomingMsg(
IN   messageProcessingModel    -- typically, SNMP version
IN   maxMessageSize            -- from the received message
IN   securityParameters        -- from the received message
IN   securityModel             -- from the received message
IN   securityLevel             -- from the received message
IN   wholeMsg                  -- as received on the wire
IN   wholeMsgLength            -- length as received on the wire
IN   tmStateReference          -- (NEW) from the Transport Model
OUT  securityEngineID          -- authoritative SNMP entity
OUT  securityName              -- identification of the principal
OUT  scopedPDU,                -- message (plaintext) payload
OUT  maxSizeResponseScopedPDU  -- maximum size sender can handle
OUT  securityStateReference    -- reference to security state
 )                         -- information, needed for response
```

5.2. Elements of Procedure for Incoming Messages

TOC

1) Set the securityEngineID to the local snmpEngineID.
2) If tmStateReference does not refer to a cache containing values for transportDomain, tmSecurityName and tmTransportSecurityLevel, then the snmpTsmInvalidCaches counter is incremented, an error indication is returned to the calling module, and Security Model processing stops for this message.
3) Set the securityName:

> **a.** If the snmpTsmConfigurationUsePrefix object is set false, copy the tmSecurityName to securityName.
>
> **b.** If the snmpTsmConfigurationUsePrefix object is set to true then use the transportDomain to look up the needed transport prefix in an implementation dependent way. (The transport prefix will likely be described in the transport domain object's DESCRIPTION clause for a given transportDomain.) Set the securityName to the concatenation of the transport's prefix, the colon character (':', ASCII 0x3a) and the tmSecurityName. If the prefix look-up using the transportDomain fails for any reason then the snmpTsmInvalidPrefixDomain counter is incremented and an error indication is returned to the calling module.

4) Compare the value of tmTransportSecurityLevel in the tmStateReference cache to the value of the securityLevel parameter passed in the processIncomingMsg ASI. If securityLevel specifies

privacy (Priv), and tmTransportSecurityLevel specifies no privacy (noPriv), or securityLevel specifies authentication (auth) and tmTransportSecurityLevel specifies no authentication (noAuth) was provided by the Transport Model, then the snmpTsmInadequateSecurityLevels counter is incremented, and an error indication (unsupportedSecurityLevel) together with the OID and value of the incremented counter is returned to the calling module. Transport Security Model processing stops for this message.

5)The security data is cached as cachedSecurityData, so that a possible response to this message will use the same security parameters. Then securityStateReference is set for subsequent reference to this cached data. For Transport Security Model, the securityStateReference includes a reference to the tmStateReference cache.

6) The scopedPDU component is extracted from the wholeMsg.

7) The maxSizeResponseScopedPDU is calculated. This is the maximum size allowed for a scopedPDU for a possible Response message.

8) The statusInformation is set to success and a return is made to the calling module passing back the OUT parameters as specified in the processIncomingMsg ASI.

---

6. MIB Module Overview

TOC

This MIB module provides management of the Transport Security Model. It defines some needed textual conventions, some statistics, and a configuration scalar for use by the Transport Security Model.

6.1. Structure of the MIB Module

TOC

Objects in this MIB module are arranged into subtrees. Each subtree is organized as a set of related objects. The overall structure and assignment of objects to their subtrees, and the intended purpose of each subtree, is shown below.

6.2. The snmpTsmStats Subtree

TOC

This subtree contains counters specific to the Transport Security Model, that provide information for identifying fault conditions.

TOC

6.3. The snmpTsmConfiguration Subtree

This subtree contains configuration objects that enable administrators to specify if they want securityNames derived from transports to begin with a transport specific prefix string.

6.4. Relationship to Other MIB Modules

TOC

Some management objects defined in other MIB modules are applicable to an entity implementing the Transport Security Model. In particular, it is assumed that an entity implementing the Transport Security Model will implement the SNMPv2-MIB [RFC3418] (Presuhn, R., “Management Information Base (MIB) for the Simple Network Management Protocol (SNMP),” December 2002.) and the SNMP-FRAMEWORK-MIB [RFC3411] (Harrington, D., Presuhn, R., and B. Wijnen, “An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,” December 2002.).

6.4.1. Relationship to the SNMPv2-MIB

TOC

The 'system' group in the SNMPv2-MIB [RFC3418] (Presuhn, R., “Management Information Base (MIB) for the Simple Network Management Protocol (SNMP),” December 2002.) is defined as being mandatory for all systems, and the objects apply to the entity as a whole. The 'system' group provides identification of the management entity and certain other system-wide data. The snmpInASNParseErrs counter is incremented during the elements of procedure. The SNMP-TSM-MIB does not duplicate those objects.

6.4.2. Relationship to the SNMP-FRAMEWORK-MIB

TOC

The SNMP-FRAMEWORK-MIB provides definitions for the concepts of SnmpEngineID, enumeration of Message Processing Models, Security Models and Security Levels, and object definitions for snmpEngineID These are important for implementing the Transport Security Model, but are not needed to implement the SNMP-TSM-MIB.

TOC

6.4.3. MIB Modules Required for IMPORTS

The following MIB module imports items from [RFC2578] (McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., “Structure of Management Information Version 2 (SMIv2),” April 1999.), [RFC2579] (McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., “Textual Conventions for SMIv2,” April 1999.), [RFC2580] (McCloghrie, K., Perkins, D., and J. Schoenwaelder, “Conformance Statements for SMIv2,” April 1999.), [RFC3411] (Harrington, D., Presuhn, R., and B. Wijnen, “An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks,” December 2002.), and [RFC3419] (Daniele, M. and J. Schoenwaelder, “Textual Conventions for Transport Addresses,” December 2002.).

---

7. MIB module definition

TOC

```
SNMP-TSM-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE,
    mib-2, Counter32
      FROM SNMPv2-SMI
    MODULE-COMPLIANCE, OBJECT-GROUP
      FROM SNMPv2-CONF
    TruthValue
       FROM SNMPv2-TC
    ;

snmpTsmMIB MODULE-IDENTITY
    LAST-UPDATED "200807100000Z"
    ORGANIZATION "ISMS Working Group"
    CONTACT-INFO "WG-EMail:   isms@lists.ietf.org
                  Subscribe:  isms-request@lists.ietf.org

                  Chairs:
                    Juergen Quittek
                    NEC Europe Ltd.
                    Network Laboratories
                    Kurfuersten-Anlage 36
                    69115 Heidelberg
                    Germany
                    +49 6221 90511-15
                    quittek@netlab.nec.de

                    Juergen Schoenwaelder
                    Jacobs University Bremen
                    Campus Ring 1
                    28725 Bremen
                    Germany
                    +49 421 200-3587
                    j.schoenwaelder@iu-bremen.de

                  Editor:
                    David Harrington
                    Huawei Technologies USA
                    1700 Alma Dr.
                    Plano TX 75075
                    USA
                    +1 603-436-8634
                    ietfdbh@comcast.net

                    Wes Hardaker
                    Sparta, Inc.
                    P.O. Box 382
```

```
                      Davis, CA  95617
                      USA
                      +1 530 792 1913
                      ietf@hardakers.net
                   "
    DESCRIPTION "The Transport Security Model MIB

                   In keeping with the RFC 3411 design decisions
                   to use self-contained documents, the RFC which
                   contains the definition of this MIB module also
                   includes the elements of procedure which are
                   needed for processing the Transport Security
                   Model for SNMP. These MIB objects
                   SHOULD NOT be modified via other subsystems
                   or models defined in other document..
                   This allows the Transport Security Model
                   for SNMP to be designed and documented as
                   independent and self- contained, having no
                   direct impact on other modules, and this
                   allows this module to be upgraded and
                   supplemented as the need arises, and to
                   move along the standards track on different
                   time-lines from other modules.

                   Copyright (C) The IETF Trust (2008). This
                   version of this MIB module is part of RFC XXXX;
                   see the RFC itself for full legal notices.
-- NOTE to RFC editor: replace XXXX with actual RFC number
--                     for this document and remove this note
                  "

    REVISION    "200807100000Z"
    DESCRIPTION "The initial version, published in RFC XXXX.
-- NOTE to RFC editor: replace XXXX with actual RFC number
--                     for this document and remove this note
                  "

    ::= { mib-2 xxxx }
-- RFC Ed.: replace xxxx with IANA-assigned number and
--          remove this note

-- ---------------------------------------------------------- --
-- subtrees in the SNMP-TSM-MIB
-- ---------------------------------------------------------- --

snmpTsmNotifications OBJECT IDENTIFIER ::= { snmpTsmMIB 0 }
snmpTsmMIBObjects    OBJECT IDENTIFIER ::= { snmpTsmMIB 1 }
snmpTsmConformance   OBJECT IDENTIFIER ::= { snmpTsmMIB 2 }
```

```
-- ------------------------------------------------------------
-- Objects
-- ------------------------------------------------------------

-- Statistics for the Transport Security Model


snmpTsmStats         OBJECT IDENTIFIER ::= { snmpTsmMIBObjects 1 }

snmpTsmInvalidCaches OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The number of messages dropped because the
                 tmStateReference referred to an invalid cache.
                "
    ::= { snmpTsmStats 1 }

snmpTsmInadequateSecurityLevels OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The number of incoming messages dropped because
                 the securityLevel asserted by the transport model was
                 less than the securityLevel requested by the
                 application.
                "
    ::= { snmpTsmStats 2 }

snmpTsmInvalidDomains OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The number of messages dropped because the
                 specified transport domain is not supported or is
                 disabled.
                "
    ::= { snmpTsmStats 3 }

snmpTsmInvalidPrefixDomain OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The number of messages dropped because
                 snmpTsmConfigurationUsePrefix was set to true and
                 there is no known prefix for the specified transport
                 domain.
                "
    ::= { snmpTsmStats 4 }
```

```
snmpTsmInvalidPrefix OBJECT-TYPE
    SYNTAX       Counter32
    MAX-ACCESS   read-only
    STATUS       current
    DESCRIPTION "The number of messages dropped because
                 the securityName associated with an outgoing message
                 did not contain a valid transport domain prefix.
                "
    ::= { snmpTsmStats 5 }


-- ------------------------------------------------------------
-- Configuration
-- ------------------------------------------------------------

-- Configuration for the Transport Security Model


snmpTsmConfiguration   OBJECT IDENTIFIER ::= { snmpTsmMIBObjects 2 }

snmpTsmConfigurationUsePrefix OBJECT-TYPE
    SYNTAX      TruthValue
    MAX-ACCESS  read-write
    STATUS      current
    DESCRIPTION "If this object is set to true then securityNames
                 passing to and from the application will contain a
                 transport domain specific prefix. If set to true then
                 the prefix will be added by the TSM to the
                 securityName for incoming messages and removed from
                 the securityName for outgoing messages. The resulting
                 securityName, when prefixed, will begin with the
                 transport specific prefix, followed by a ':'
                 character (ASCII 0x3a). The transport specific prefix
                 should be defined within the transport domain's
                 object definition.
                "
    DEFVAL { false }
    ::= { snmpTsmConfiguration 1 }

-- ------------------------------------------------------------
-- snmpTsmMIB - Conformance Information
-- ------------------------------------------------------------

snmpTsmCompliances OBJECT IDENTIFIER ::= { snmpTsmConformance 1 }

snmpTsmGroups      OBJECT IDENTIFIER ::= { snmpTsmConformance 2 }

-- ------------------------------------------------------------
-- Compliance statements
-- ------------------------------------------------------------
```

```
snmpTsmCompliance MODULE-COMPLIANCE
    STATUS      current
    DESCRIPTION "The compliance statement for SNMP engines that support
                 the SNMP-TSM-MIB
                "
    MODULE
        MANDATORY-GROUPS { snmpTsmGroup }
    ::= { snmpTsmCompliances 1 }

-- ----------------------------------------------------------------
-- Units of conformance
-- ----------------------------------------------------------------
snmpTsmGroup OBJECT-GROUP
    OBJECTS {
        snmpTsmInvalidCaches,
        snmpTsmInadequateSecurityLevels,
        snmpTsmInvalidDomains,
        snmpTsmInvalidPrefixDomain,
        snmpTsmInvalidPrefix,
        snmpTsmConfigurationUsePrefix
    }
    STATUS      current
    DESCRIPTION "A collection of objects for maintaining
                 information of an SNMP engine which implements
                 the SNMP Transport Security Model.
                "

    ::= { snmpTsmGroups 2 }


END
```

8. Security Considerations

TOC

This document describes a Security Model that permits SNMP to utilize security services provided through an SNMP Transport Model. The Transport Security Model relies on Transport Models for mutual authentication, binding of keys, confidentiality and integrity. The security threats and how those threats are mitigated should be covered in detail in the specification of the Transport Model and the underlying secure transport.

Transport Security Model relies on a Transport Model to provide an authenticated principal for mapping to securityName, and an assertion of tmTransportSecurityLevel. New transport models SHOULD provide a

transport domain prefix to allow operators to distinguish between identities whose authentication is coordinated by different transport models. This should preferably be specified in the transport domain object's DESCRIPTION clause. The prefix MUST be added and removed appropriately by the Transport Security Model if the administrator has set snmpTsmConfigurationUsePrefix to true and MUST NOT be added and removed if snmpTsmConfigurationUsePrefix is set to false.
The Transport Security Model is called a Security Model to be compatible with the RFC3411 architecture. However, this Security Model provides no security itself. It SHOULD always be used with a Transport Model that provides security, but this is a run-time decision of the operator or management application, or a configuration decision of an operator.

8.1. MIB module security

TOC

There are a number of management objects defined in this MIB module with a MAX-ACCESS clause of read-write and/or read-create. Such objects may be considered sensitive or vulnerable in some network environments. The support for SET operations in a non-secure environment without proper protection can have a negative effect on network operations. These are the tables and objects and their sensitivity/vulnerability:

- The snmpTsmConfigurationUsePrefix object could be modified creating a denial of service or authorizing SNMP messages that would not have previously been authorized by an Access Control Model (e.g. the VACM).

Some of the readable objects in this MIB module (i.e., objects with a MAX-ACCESS other than not-accessible) may be considered sensitive or vulnerable in some network environments. It is thus important to control even GET and/or NOTIFY access to these objects and possibly to even encrypt the values of these objects when sending them over the network via SNMP. These are the tables and objects and their sensitivity/vulnerability:

- snmpTsmInvalidCaches and snmpTsmInadequateSecurityLevels and snmpTsmInvalidDomains may make it easier for an attacker to detect vulnerabilities.

SNMP versions prior to SNMPv3 did not include adequate security. Even if the network itself is secure (for example by using IPsec), even then, there is no control as to who on the secure network is allowed to access and GET/SET (read/change/create/delete) the objects in this MIB module.
It is RECOMMENDED that implementers consider the security features as provided by the SNMPv3 framework (see [RFC3410] (Case, J., Mundy, R.,

Partain, D., and B. Stewart, “Introduction and Applicability Statements for Internet-Standard Management Framework,” December 2002.) section 8), including full support for the USM and Transport Security Model cryptographic mechanisms (for authentication and privacy).
Further, deployment of SNMP versions prior to SNMPv3 is NOT RECOMMENDED. Instead, it is RECOMMENDED to deploy SNMPv3 and to enable cryptographic security. It is then a customer/operator responsibility to ensure that the SNMP entity giving access to an instance of this MIB module is properly configured to give access to the objects only to those principals (users) that have legitimate rights to indeed GET or SET (change/create/delete) them.

---

9. IANA Considerations

TOC

[DISCUSS: should we have default ports for request/response traffic and for notifications?]
IANA is requested to assign:

1. an SMI number under mib-2, for the MIB module in this document,

2. a value, preferably 4, to identify the Transport Security Model, in the Security Models registry at http://www.iana.org/assignments/snmp-number-spaces. This should result in the following table of values:

```
Value   Description                          References
-----   -----------                          ----------
  0     reserved for 'any'                   [RFC3411]
  1     reserved for SNMPv1                  [RFC3411]
  2     reserved for SNMPv2c                 [RFC3411]
  3     User-Based Security Model (USM)      [RFC3411]
  YY    Transport Security Model (TSM)       [RFC XXXX]

-- NOTE to RFC editor: replace XXXX with actual RFC number
--                     for this document and remove this note
-- NOTE to RFC editor: replace YY with actual IANA-assigned number,
                       throughout this document and remove this note.
```

TOC

10. Acknowledgements

The editors would like to thank Jeffrey Hutzelman for sharing his SSH insights, and Dave Shields for an outstanding job wordsmithing the existing document to improve organization and clarity.

---

11. References

TOC

11.1. Normative References

TOC

[RFC2119]	Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels," BCP 14, RFC 2119, March 1997 (TXT, HTML, XML).
[RFC2578]	McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)," STD 58, RFC 2578, April 1999 (TXT).
[RFC2579]	McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2," STD 58, RFC 2579, April 1999 (TXT).
[RFC2580]	McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2," STD 58, RFC 2580, April 1999 (TXT).
[RFC3411]	Harrington, D., Presuhn, R., and B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks," STD 62, RFC 3411, December 2002 (TXT).
[RFC3412]	Case, J., Harrington, D., Presuhn, R., and B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)," STD 62, RFC 3412, December 2002 (TXT).
[RFC3413]	Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications," STD 62, RFC 3413, December 2002 (TXT).
[RFC3418]	Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)," STD 62, RFC 3418, December 2002 (TXT).
[RFC3419]	Daniele, M. and J. Schoenwaelder, "Textual Conventions for Transport Addresses," RFC 3419, December 2002 (TXT).
[I-D.ietf-isms-tmsm]	Harrington, D. and J. Schoenwaelder, "Transport Subsystem for the Simple Network Management Protocol (SNMP)," draft-ietf-isms-tmsm-18 (work in progress), May 2009 (TXT).

11.2. Informative References

TOC

[RFC3410]	Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework," RFC 3410, December 2002 (TXT).
[RFC3414]	Blumenthal, U. and B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)," STD 62, RFC 3414, December 2002 (TXT).
[RFC3584]	Frye, R., Levi, D., Routhier, S., and B. Wijnen, "Coexistence between Version 1, Version 2, and Version 3 of the Internet-standard Network Management Framework," BCP 74, RFC 3584, August 2003 (TXT).

Appendix A. Notification Tables Configuration

TOC

The SNMP-TARGET-MIB and SNMP-NOTIFICATION-MIB [RFC3413] (Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications," December 2002.) are used to configure notification originators with the destinations to which notifications should be sent.

Most of the configuration is security-model-independent and transport-model-independent.

The values we will use in the examples for the five model-independent security and transport parameters are:

> transportDomain = snmpSSHDomain
>
> transportAddress = 192.0.2.1:162
>
> securityModel = Transport Security Model
>
> securityName = sampleUser
>
> securityLevel = authPriv

The following example will configure the Notification Originator to send informs to a Notification Receiver at host 192.0.2.1 port 162 using the securityName "sampleUser". The columns marked with a "*" are the items that are Security Model or Transport Model specific.

The configuration for the "sampleUser" settings in the SNMP-VIEW-BASED-ACM-MIB objects are not shown here for brevity. First we configure which type of notification should be sent for this taglist (toCRTag). In this example, we choose to send an Inform.

```
       snmpNotifyTable row:
            snmpNotifyName                   CRNotif
            snmpNotifyTag                    toCRTag
            snmpNotifyType                   inform
            snmpNotifyStorageType            nonVolatile
            snmpNotifyColumnStatus           createAndGo
```

Then we configure a transport address to which notifications associated with this taglist should be sent, and we specify which snmpTargetParamsEntry should be used (toCR) when sending to this transport address.

```
            snmpTargetAddrTable row:
               snmpTargetAddrName                toCRAddr
           *   snmpTargetAddrTDomain             snmpSSHDomain
               snmpTargetAddrTAddress            192.0.2.1:162
               snmpTargetAddrTimeout             1500
               snmpTargetAddrRetryCount          3
               snmpTargetAddrTagList             toCRTag
               snmpTargetAddrParams              toCR   (must match below)
               snmpTargetAddrStorageType         nonVolatile
               snmpTargetAddrColumnStatus        createAndGo
```

Then we configure which principal at the host should receive the notifications associated with this taglist. Here we choose "sampleUser", who uses the Transport Security Model.

```
           snmpTargetParamsTable row:
               snmpTargetParamsName              toCR
               snmpTargetParamsMPModel           SNMPv3
           *   snmpTargetParamsSecurityModel     TransportSecurityModel
               snmpTargetParamsSecurityName      "sampleUser"
               snmpTargetParamsSecurityLevel     authPriv
               snmpTargetParamsStorageType       nonVolatile
               snmpTargetParamsRowStatus         createAndGo
```

A.1. Transport Security Model Processing for Notifications

TOC

The Transport Security Model is called using the generateRequestMsg() ASI, with the following parameters (* are from the above tables):

```
statusInformation =                  -- success or errorIndication
      generateRequestMsg(
      IN   messageProcessingModel  -- *snmpTargetParamsMPModel
      IN   globalData              -- message header, admin data
      IN   maxMessageSize          -- of the sending SNMP entity
      IN   transportDomain         -- *snmpTargetAddrTDomain
      IN   transportAddress        -- *snmpTargetAddrTAddress
      IN   securityModel           -- *snmpTargetParamsSecurityModel
      IN   securityEngineID        -- immaterial; TSM will ignore.
      IN   securityName            -- snmpTargetParamsSecurityName
      IN   securityLevel           -- *snmpTargetParamsSecurityLevel
      IN   scopedPDU               -- message (plaintext) payload
      OUT  securityParameters      -- filled in by Security Module
      OUT  wholeMsg                -- complete generated message
      OUT  wholeMsgLength          -- length of generated message
      OUT  tmStateReference        -- reference to transport info
           )
```

The Transport Security Model will determine the Transport Model based on the snmpTargetAddrTDomain. The selected Transport Model will select the appropriate transport connection using the snmpTargetAddrTAddress, snmpTargetParamsSecurityName, and snmpTargetParamsSecurityLevel.

Appendix B. Processing Differences between USM and Secure Transport

TOC

USM and secure transports differ in the processing order and responsibilities within the RFC3411 architecture. While the steps are the same, they occur in a different order, and may be done by different subsystems. The following lists illustrate the difference in the flow and the responsibility for different processing steps for incoming messages when using USM and when using a secure transport. (Note that these lists are simplified for illustrative purposes, and do not represent all details of processing. Transport Models must provide the detailed elements of procedure.)

With USM, SNMPv1, and SNMPv2c Security Models, security processing starts when the Message Processing Model decodes portions of the ASN.1 message to extract header fields that are used to determine which Security Model should process the message to perform authentication, decryption, timeliness checking, integrity checking, and translation of parameters to model-independent parameters. By comparison, a secure transport performs those security functions on the message, before the ASN.1 is decoded.

Step 6 cannot occur until after decryption occurs. Step 6 and beyond are the same for USM and a secure transport.

B.1. USM and the RFC3411 Architecture

TOC

1) decode the ASN.1 header (Message Processing Model)
2) determine the SNMP Security Model and parameters (Message Processing Model)
3) verify securityLevel. [Security Model]
4) translate parameters to model-independent parameters (Security Model)
5) authenticate the principal, check message integrity and timeliness, and decrypt the message. [Security Model]
6) determine the pduType in the decrypted portions (Message Processing Model), and
7) pass on the decrypted portions with model-independent parameters.

B.2. Transport Subsystem and the RFC3411 Architecture

TOC

1) authenticate the principal, check integrity and timeliness of the message, and decrypt the message. [Transport Model]
2) translate parameters to model-independent parameters (Transport Model)
3) decode the ASN.1 header (Message Processing Model)
4) determine the SNMP Security Model and parameters (Message Processing Model)
5) verify securityLevel [Security Model]
6) determine the pduType in the decrypted portions (Message Processing Model), and
7) pass on the decrypted portions with model-independent security parameters

If a message is secured using a secure transport layer, then the Transport Model should provide the translation from the authenticated identity (e.g., an SSH user name) to a human-friendly identifier in step 2. The security model will provide a mapping from that identifier to a model-independent securityName.

Appendix C. Open Issues

TOC

> Does TSM need to have a mapping table to handle the translations from tmSecurityName to securityName?
>
> Do we need administratively definable transform selection?
>
> Do we need to let operators disable support for some transports?

Appendix D. Change Log

TOC

From -08- to -09-

> Added the transport domain specific prefix adding/removing support as agreed to within the ISMS WG. The implementation is a bit different than what was originally discussed and is now housed entirely within this document and requires only a string allocation in the TM documents. In the end this form greatly reduced the documentation and procedure complexity in most documents.
>
> Added the snmpTsmConfigurationUsePrefix scalar.
>
> Removed the snmpTsmLCDTable since it is no longer needed.
>
> Removed the snmpTsmLCDDomainTable since it is not needed with the prefix addition replaced the functionality.

From -07- to -08-

> Added tables to the MIB module to define a Transport Security Model-specific LCD, and updated the Elements of Procedure. This was because references to an abstract LCD sort of owned by both the security model and the transport model were found confusing.
>
> Realized we referred to the MIB module in text as SNMP-TRANSPORT-SM-MIB, but SNMP-TSM-MIB in the module. Changed all occurrences of SNMP-TRANSPORT-SM-MIB to SNMP-TSM-MIB, following RFC4181 guidelines for naming.
>
> Updated Security Considerations to warn about writable objects, and added the new counter to the readable objects list.

Changed snmpTsmLCDName to snmpTsmLCDTmSecurityName

From -05- to -06-

Fixed a bunch of editorial nits

Fixed the note about terminology consistent with SNMPv3.

Updated MIB assignment to by rfc4181 compatible

Replaced tmSameSession with tmSameSecurity to eliminate session-matching from the security model.

Eliminated all reference to the LCD from the Transport Security Model; the LCD is now TM-specific.

Added tmTransportSecurityLevel and tmRequestedSecurityLevel to clarify incoming versus outgoing

From -04- to -05-

Removed check for empty securityParameters for incoming messages

Added a note about terminology, for consistency with SNMPv3 rather than with RFC2828.

From -03- to -04-

Editorial changes requested by Tom Petch, to clarify behavior with SNMPv1/v2c

Added early discussion of how TSM fits into the architecture to clarify behavior when RFC3584 security models are co-resident.

Editorial changes requested by Bert Wijnen, to eliminate version-specific discussions.

Removed sections on version-specific message formats.

Removed discussion of SNMPv3 in Motivation section.

Added discussion of request/response session matching.

From -02- to -03-

Editorial changes suggested by Juergen Schoenwaelder

Capitalized Transport Models, Security Models, and Message Processing Models, to be consistent with RFC341x conventions.

> Eliminated some text that duplicated RFC3412, especially in Elements of Procedure.
>
> Changed the encoding of msgSecurityParameters
>
> Marked the (NEW) fields added to existing ASIs
>
> Modified text intro discussing relationships to other MIB modules.

From -01- to -02-

> Changed transportSecurityModel(4) to transportSecurityModel(YY), waiting for assignment
>
> cleaned up elements of procedure [todo]s
>
> use the same errorIndication as USM for unsupportedSecurityLevel
>
> fixed syntax of tsmInadequateSecurity counter
>
> changed the "can and will use" the same security parameters to "can use", to allow responses that have different security parameters than the request.
>
> removed "Relationship to the SNMP-FRAMEWORK-MIB"
>
> cleaned up "MIB Modules Required for IMPORTS"

From -00- to -01-

> made the Transport Model not know anything about the Security Model.
>
> modified the elements of procedure sections, given the implications of this change.
>
> simplified elements of procedure, removing most info specified in architecture/subsystem definitions.
>
> rethought the coexistence section
>
> noted the implications of the Transport Security Model on isAccessAllowed()
>
> modified all text related to the LCD.
>
> removed most of the MIB (now the TSM has no configuration parameters).
>
> added counters needed to support elements of procedure
>
> renamed MIB module, and registered under snmpModules

updated IANA and Security Considerations

updated references.

modified the notification configurations.

From SSHSM-04- to Transport-security-model-00

added tsmUserTable

updated Appendix - Notification Tables Configuration

remove open/closed issue appendices

changed tmSessionReference to tmStateReference

Authors' Addresses

TOC

	David Harrington
	Huawei Technologies (USA)
	1700 Alma Dr. Suite 100
	Plano, TX 75075
	USA
Phone:	+1 603 436 8634
EMail:	dharrington@huawei.com
	Wes Hardaker
	Sparta, Inc.
	P.O. Box 382
	Davis, CA 95617
	US
Phone:	+1 530 792 1913
EMail:	ietf@hardakers.net

Full Copyright Statement

TOC

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.