Network Working Group D. Harrington
Internet-Draft Huawei Technologies (USA)
Intended status: Standards Track September 18, 2007
Expires: March 21, 2008

Transport Security Model for SNMP
draft-ietf-isms-transport-security-model-06

Status of This Memo

Copyright Notice

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

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].

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], RFC3412 [RFC3412], RFC3413 [RFC3413], and RFC3418 [RFC3418].

1.1. The Internet-Standard Management Framework

For a detailed overview of the documents that describe the current Internet-Standard Management Framework, please refer to section 7 of RFC 3410 [RFC3410].

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], STD 58, RFC 2579 [RFC2579] and STD 58, RFC 2580 [RFC2580].

1.2. Conventions

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] for further information.

While security protocols frequently refer to a user, the terminology used in RFC3411 [RFC3411] 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].

1.3. Modularity

The reader is expected to have read and understood the description of the SNMP architecture, as defined in [RFC3411], and the architecture extension specified in "Transport Subsystem for the Simple Network Management Protocol" [I-D.ietf-isms-tmsm], 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

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

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

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 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).

When a secure Transport Model creates a tmStateReference cache for an

incoming message, it will include a tmTransport, tmAddress, tmSecurityName and a tmTransportSecurityLevel, and it MAY include transport-specific information. When the Transport Security Model sends a message, it will create a cache containing the specified transportDomain, transportAddress, securityName, and securityLevel parameters. The Transport Security Model will pass the tmStateReference to enable the transport model to extract corresponding transport-specific information from the tmStateReference cache, in an implementation-dependent manner.

When the Transport Security Model determines that a cache does not yet exist corresponding to the specified transportDomain, transportAddress, secuirtyName, and securityLevel parameters, it creates one that contains a tmSecurityName and tmRequestedSecurityLevel and passes the tmStateReference cache to the specified Transport Model.

The Transport Security Model will determine the security-model-independent securityName and securityLevel, and will verify for incoming messages that tmTransportSecurityLevel is at least as strong as the requested securityLevel.

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

2.1.1. Threats

The Transport Security Model, when used with suitable secure Transport Models, provides protection against the threats identified by the RFC 3411 architecture [RFC3411].

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

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

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

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] 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], 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.

Note that the Transport Security Model may work with multiple Transport Models, but the isAccessAllowed() application service interfaces (ASI) only accepts a value for the Security Model, not for Transport Models. As a result, it is not possible to have different access control rules for different Transport Models that use the Transport Security Model.

2.4. Security Parameter Passing

For outgoing messages, Transport Security Model uses parameters provided by the SNMP application to determine if a corresponding tmStateReference cache exists, or to create a suitable tmStateReference cache. 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].

For incoming messages, the Transport Model accepts messages from the lower layer transport, and records the transport-related information

and security-related information, including a tmSecurityName that represents the transport-authenticated identity, and a tmTransportSecurityLevel that represents the security features provided during transport, in a cache referenced by tmStateReference. 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].

2.5. Notifications and Proxy

The SNMP-TARGET-MIB module [RFC3413] 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.

For the Transport Security Model, these will be translated as needed into tmSecurityName and tmRequestedSecurityLevel.

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

3. Cached Information and References

The RFC3411 architecture uses caches to store dynamic model-specific information, and uses references in the ASIs to indicate in a model-independent manner which cached information must flow between subsystems.

There are two levels of state that may need to be maintained: the security state in a request-response pair, and potentially long-term state relating to transport and security. This document describes caches, and differentiates the tmStateReference from the securityStateReference, but how this is represented internally is an implementation decision.

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, and if state information is available when a relationship between engines is severed, such as the closing of a transport connection, the state information for that relationship might also be discarded.

3.1. tmStateReference

For each transport model, information about the transport security may be stored in a cache to pass model- and mechanism-specific parameters.

For security reasons, the Transport Security Model REQUIRES that the security parameters used for a response are the same as those used for the corresponding request, and passes a tmSameSecurity parameter in the tmStateReference cache for outgoing messages to indicate that the same security MUST be used for the outgoing response as was used for the corresponding incoming request. It is transport-model-dependent and implementation-dependent how this is ensured at the transport layer.

Since the contents of a cache are meaningful only within an implementation, and not on-the-wire, the format of the cache is implementation-specific.

3.2. securityStateReference

The securityStateReference parameter is defined in RFC3411. Its primary purpose is to provide a mapping between a request and the corresponding response. A sample model-specific cache can be found in RFC3414 [RFC3414].

Transport models do not have access to the securityStateReference. For the Transport Security Model, it is important to ensure that the security parameters used for a request match those used for the corresponding response. The Transport Security Model will conceptually add the tmStateReference to the securityStateReference cache, so the transport 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

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

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

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.

2) If there is no securityStateReference, create a tmStateReference cache with tmSecurityName set to the value of securityName, tmRequestedSecurityLevel set to the value of securityLevel, and tmSameSecurity set to false.

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

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

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

1) Set the securityEngineID to the local snmpEngineID.

2) If tmStateReference does not refer to a cache containing values for 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 securityName to the value of tmSecurityName from the cache referenced by tmStateReference.

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

This MIB module provides management of the Transport Security Model. It defines some needed textual conventions, and some statistics.

6.1. Structure of the MIB Module

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 tsmStats Subtree

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

6.3. Relationship to Other MIB Modules

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] and the SNMP-FRAMEWORK-MIB [RFC3411].

6.3.1. Relationship to the SNMPv2-MIB

The 'system' group in the SNMPv2-MIB [RFC3418] 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-TRANSPORT-SM-MIB does not duplicate those objects.

6.3.2. Relationship to the SNMP-FRAMEWORK-MIB

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-TRANSPORT-SM-MIB.

6.3.3. MIB Modules Required for IMPORTS

The following MIB module imports items from [RFC2578] and [RFC2580].

7. MIB module definition

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

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

snmpTsmMIB MODULE-IDENTITY
    LAST-UPDATED "200710140000Z"
    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
                       "
        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 documents.
                        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 (2007). 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     "200710140000Z"
        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-TRANSPORT-SM-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 actual securityLevel provided was less than
                    the requested securityLevel.
                   "
       ::= { snmpTsmStats 2 }

   -- -------------------------------------------------------------
   -- 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-TRANSPORT-SM-MIB"
       MODULE
           MANDATORY-GROUPS { snmpTsmGroup }
       ::= { snmpTsmCompliances 1 }

   -- ----------------------------------------------------------------
   -- Units of conformance
   -- ----------------------------------------------------------------
   snmpTsmGroup OBJECT-GROUP
       OBJECTS {
           snmpTsmInvalidCaches,
           snmpTsmInadequateSecurityLevels
       }
       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

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.

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

There are no management objects defined in this MIB module that have a MAX-ACCESS clause of read-write and/or read-create. So, if this MIB module is implemented correctly, then there is no risk that an intruder can alter or create any management objects of this MIB module via direct SNMP SET operations.

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:

- o snmpTsmInvalidCaches and snmpTsmInadequateSecurityLevels 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] 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

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)	[RFCXXXX]

-- 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.

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.

[RFC2578] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Structure of Management Information Version 2 (SMIv2)", STD 58, RFC 2578, April 1999.

[RFC2579] McCloghrie, K., Ed., Perkins, D., Ed., and J. Schoenwaelder, Ed., "Textual Conventions for SMIv2", STD 58, RFC 2579, April 1999.

[RFC2580] McCloghrie, K., Perkins, D., and J. Schoenwaelder, "Conformance Statements for SMIv2", STD 58, RFC 2580, April 1999.

[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.

[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.

[RFC3413] Levi, D., Meyer, P., and B. Stewart, "Simple Network Management Protocol (SNMP) Applications", STD 62, RFC 3413, December 2002.

[RFC3418] Presuhn, R., "Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3418, December 2002.

[I-D.ietf-isms-tmsm] Harrington, D. and J. Schoenwaelder, "Transport Subsystem for the Simple Network Management Protocol (SNMP)", draft-ietf-isms-tmsm-10 (work in progress), September 2007.

10.2. Informative References

[RFC3410] Case, J., Mundy, R., Partain, D., and B. Stewart, "Introduction and Applicability Statements for Internet-Standard Management Framework", RFC 3410, December 2002.

[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.

[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.

Appendix A. Notification Tables Configuration

The SNMP-TARGET-MIB and SNMP-NOTIFICATION-MIB [RFC3413] 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

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. Open Issues

none.

Appendix C. Change Log

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

Author's Address

David Harrington
Huawei Technologies (USA)
1700 Alma Dr. Suite 100
Plano, TX 75075
USA

Phone: +1 603 436 8634
EMail: dharrington@huawei.com

Full Copyright Statement

Intellectual Property

Acknowledgement

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).