Network Working Group
Request for Comments: 1910
Category: Experimental

G. Waters, Editor
Bell-Northern Research Ltd.

February 1996

User-based Security Model for SNMPv2

Status of this Memo

Table of Contents

1. Introduction

A management system contains: several (potentially many) nodes, each with a processing entity, termed an agent, which has access to management instrumentation; at least one management station; and, a management protocol, used to convey management information between the agents and management stations. Operations of the protocol are carried out under an administrative framework which defines authentication, authorization, access control, and privacy policies.

Management stations execute management applications which monitor and control managed elements. Managed elements are devices such as hosts, routers, terminal servers, etc., which are monitored and controlled via access to their management information.

The Administrative Infrastructure for SNMPv2 document [1] defines an administrative framework which realizes effective management in a variety of configurations and environments.

In this administrative framework, a security model defines the mechanisms used to achieve an administratively-defined level of security for protocol interactions. Although many such security models might be defined, it is the purpose of this document, User-based Security Model for SNMPv2, to define the first, and, as of this writing, only, security model for this administrative framework.

This administrative framework includes the provision of an access control model. The enforcement of access rights requires the means to identify the entity on whose behalf a request is generated. This SNMPv2 security model identifies an entity on whose behalf an SNMPv2 message is generated as a "user".

1.1. Threats

Several of the classical threats to network protocols are applicable to the network management problem and therefore would be applicable to any SNMPv2 security model. Other threats are not applicable to the network management problem. This section discusses principal threats, secondary threats, and threats which are of lesser importance.

The principal threats against which this SNMPv2 security model should provide protection are:

Modification of Information

The modification threat is the danger that some unauthorized entity may alter in-transit SNMPv2 messages generated on behalf of an authorized user in such a way as to effect unauthorized management operations, including falsifying the value of an object.

Masquerade

The masquerade threat is the danger that management operations not authorized for some user may be attempted by assuming the identity of another user that has the appropriate authorizations.

Two secondary threats are also identified. The security protocols defined in this memo do provide protection against:

Message Stream Modification

The SNMPv2 protocol is typically based upon a connectionless transport service which may operate over any subnetwork service. The re-ordering, delay or replay of messages can and does occur through the natural operation of many such subnetwork services. The message stream modification threat is the danger that messages may be maliciously re-ordered, delayed or replayed to an extent which is greater than can occur through the natural operation of a subnetwork service, in order to effect unauthorized management operations.

Disclosure

The disclosure threat is the danger of eavesdropping on the exchanges between managed agents and a management station. Protecting against this threat may be required as a matter of local policy.

There are at least two threats that an SNMPv2 security protocol need not protect against. The security protocols defined in this memo do not provide protection against:

Denial of Service

An SNMPv2 security protocol need not attempt to address the broad range of attacks by which service on behalf of authorized users is denied. Indeed, such denial-of-service attacks are in many cases indistinguishable from the type of network failures with which any viable network management protocol must cope as a matter of course.

Traffic Analysis

In addition, an SNMPv2 security protocol need not attempt to address traffic analysis attacks. Indeed, many traffic patterns are predictable - agents may be managed on a regular basis by a relatively small number of management stations - and therefore there is no significant advantage afforded by protecting against traffic analysis.

1.2. Goals and Constraints

Based on the foregoing account of threats in the SNMP network management environment, the goals of this SNMPv2 security model are as follows.

(1) The protocol should provide for verification that each received SNMPv2 message has not been modified during its transmission through the network in such a way that an unauthorized management operation might result.

(2) The protocol should provide for verification of the identity of the user on whose behalf a received SNMPv2 message claims to have been generated.

(3) The protocol should provide for detection of received SNMPv2 messages, which request or contain management information, whose time of generation was not recent.

(4) The protocol should provide, when necessary, that the contents of each received SNMPv2 message are protected from disclosure.

In addition to the principal goal of supporting secure network management, the design of this SNMPv2 security model is also influenced by the following constraints:

(1) When the requirements of effective management in times of network stress are inconsistent with those of security, the design should prefer the former.

(2) Neither the security protocol nor its underlying security mechanisms should depend upon the ready availability of other network services (e.g., Network Time Protocol (NTP) or key

management protocols).

(3) A security mechanism should entail no changes to the basic SNMP network management philosophy.

1.3. Security Services

The security services necessary to support the goals of an SNMPv2 security model are as follows.

Data Integrity
is the provision of the property that data has not been altered or destroyed in an unauthorized manner, nor have data sequences been altered to an extent greater than can occur non-maliciously.

Data Origin Authentication
is the provision of the property that the claimed identity of the user on whose behalf received data was originated is corroborated.

Data Confidentiality
is the provision of the property that information is not made available or disclosed to unauthorized individuals, entities, or processes.

For the protocols specified in this memo, it is not possible to assure the specific originator of a received SNMPv2 message; rather, it is the user on whose behalf the message was originated that is authenticated.

For these protocols, it not possible to obtain data integrity without data origin authentication, nor is it possible to obtain data origin authentication without data integrity. Further, there is no provision for data confidentiality without both data integrity and data origin authentication.

The security protocols used in this memo are considered acceptably secure at the time of writing. However, the procedures allow for new authentication and privacy methods to be specified at a future time if the need arises.

1.4. Mechanisms

The security protocols defined in this memo employ several types of mechanisms in order to realize the goals and security services described above:

- In support of data integrity, a message digest algorithm is required. A digest is calculated over an appropriate portion of an SNMPv2 message and included as part of the message sent to the recipient.

- In support of data origin authentication and data integrity, a secret value is both inserted into, and appended to, the SNMPv2 message prior to computing the digest; the inserted value overwritten prior to transmission, and the appended value is not transmitted. The secret value is shared by all SNMPv2 entities authorized to originate messages on behalf of the appropriate user.

- To protect against the threat of message delay or replay (to an extent greater than can occur through normal operation), a set of time (at the agent) indicators and a request-id are included in each message generated. An SNMPv2 agent evaluates the time indicators to determine if a received message is recent. An SNMPv2 manager evaluates the time indicators to ensure that a received message is at least as recent as the last message it received from the same source. An SNMPv2 manager uses received authentic messages to advance its notion of time (at the agent). An SNMPv2 manager also evaluates the request-id in received Response messages and discards messages which do not correspond to outstanding requests.

  These mechanisms provide for the detection of messages whose time of generation was not recent in all but one circumstance; this circumstance is the delay or replay of a Report message (sent to a manager) when the manager has has not recently communicated with the source of the Report message. In this circumstance, the detection guarantees only that the Report message is more recent than the last communication between source and destination of the Report message. However, Report messages do not request or contain management information, and thus, goal #3 in Section 1.2 above is met; further, Report messages can at most cause the manager to advance its notion of time (at the agent) by less than the proper amount.

  This protection against the threat of message delay or replay does not imply nor provide any protection against unauthorized deletion or suppression of messages. Other mechanisms defined independently of the security protocol can also be used to detect the re-ordering, replay, deletion, or suppression of messages containing set operations (e.g., the MIB variable snmpSetSerialNo [15]).

- In support of data confidentiality, an encryption algorithm is required. An appropriate portion of the message is encrypted prior to being transmitted.

1.4.1. Digest Authentication Protocol

The Digest Authentication Protocol defined in this memo provides for:

- verifying the integrity of a received message (i.e., the message received is the message sent).

  The integrity of the message is protected by computing a digest over an appropriate portion of a message. The digest is computed by the originator of the message, transmitted with the message, and verified by the recipient of the message.

- verifying the user on whose behalf the message was generated.

  A secret value known only to SNMPv2 entities authorized to generate messages on behalf of this user is both inserted into, and appended to, the message prior to the digest computation. Thus, the verification of the user is implicit with the verification of the digest. (Note that the use of two copies of the secret, one near the start and one at the end, is recommended by [14].)

- verifying that a message sent to/from one SNMPv2 entity cannot be replayed to/as-if-from another SNMPv2 entity.

  Included in each message is an identifier unique to the SNMPv2 agent associated with the sender or intended recipient of the message. Also, each message containing a Response PDU contains a request-id which associates the message to a recently generated request.

  A Report message sent by one SNMPv2 agent to one SNMPv2 manager can potentially be replayed to another SNMPv2 manager. However, Report messages do not request or contain management information, and thus, goal #3 in Section 1.2 above is met; further, Report messages can at most cause the manager to advance its notion of time (at the agent) by less than the correct amount.

- detecting messages which were not recently generated.

  A set of time indicators are included in the message, indicating the time of generation. Messages (other than those containing Report PDUs) without recent time indicators are not considered authentic. In addition, messages containing Response PDUs have a request-id; if the request-id does not match that of a recently generated request, then the message is not considered to be authentic.

A Report message sent by an SNMPv2 agent can potentially be replayed at a later time to an SNMPv2 manager which has not recently communicated with that agent. However, Report messages do not request or contain management information, and thus, goal #3 in Section 1.2 above is met; further, Report messages can at most cause the manager to advance its notion of time (at the agent) by less than the correct amount.

This protocol uses the MD5 [3] message digest algorithm. A 128-bit digest is calculated over the designated portion of an SNMPv2 message and included as part of the message sent to the recipient. The size of both the digest carried in a message and the private authentication key is 16 octets.

This memo allows the same user to be defined on multiple SNMPv2 agents and managers. Each SNMPv2 agent maintains a value, agentID, which uniquely identifies the agent. This value is included in each message sent to/from that agent. Messages sent from a SNMPv2 dual-role entity [1] to a SNMPv2 manager include the agentID value maintained by the dual-role entity’s agent. On receipt of a message, an agent checks the value to ensure it is the intended recipient, and a manager uses the value to ensure that the message is processed using the correct state information.

Each SNMPv2 agent maintains two values, agentBoots and agentTime, which taken together provide an indication of time at that agent. Both of these values are included in an authenticated message sent to/received from that agent. Authenticated messages sent from a SNMPv2 dual-role entity to a SNMPv2 manager include the agentBoots and agentTime values maintained by the dual-role entity’s agent. On receipt, the values are checked to ensure that the indicated time is within a time window of the current time. The time window represents an administrative upper bound on acceptable delivery delay for protocol messages.

For an SNMPv2 manager to generate a message which an agent will accept as authentic, and to verify that a message received from that agent is authentic, that manager must first achieve time synchronization with that agent. Similarly, for a manger to verify that a message received from an SNMPv2 dual-role entity is authentic, that manager must first achieve time synchronization with the dual-role entity’s agent.

1.4.2. Symmetric Encryption Protocol

The Symmetric Encryption Protocol defined in this memo provides support for data confidentiality through the use of the Data Encryption Standard (DES) in the Cipher Block Chaining mode of operation. The designated portion of an SNMPv2 message is encrypted and included as part of the message sent to the recipient.

Two organizations have published specifications defining the DES: the National Institute of Standards and Technology (NIST) [5] and the American National Standards Institute [6]. There is a companion Modes of Operation specification for each definition (see [7] and [8], respectively).

The NIST has published three additional documents that implementors may find useful.

- There is a document with guidelines for implementing and using the DES, including functional specifications for the DES and its modes of operation [9].

- There is a specification of a validation test suite for the DES [10]. The suite is designed to test all aspects of the DES and is useful for pinpointing specific problems.

- There is a specification of a maintenance test for the DES [11]. The test utilizes a minimal amount of data and processing to test all components of the DES. It provides a simple yes-or-no indication of correct operation and is useful to run as part of an initialization step, e.g., when a computer reboots.

This Symmetric Encryption Protocol specifies that the size of the privacy key is 16 octets, of which the first 8 octets are a DES key and the second 8 octets are a DES Initialization Vector. The 64-bit DES key in the first 8 octets of the private key is a 56 bit quantity used directly by the algorithm plus 8 parity bits - arranged so that one parity bit is the least significant bit of each octet. The setting of the parity bits is ignored by this protocol.

The length of an octet sequence to be encrypted by the DES must be an integral multiple of 8. When encrypting, the data is padded at the end as necessary; the actual pad value is irrelevant.

If the length of the octet sequence to be decrypted is not an integral multiple of 8 octets, the processing of the octet sequence is halted and an appropriate exception noted. When decrypting, the padding is ignored.

2. Elements of the Model

This section contains definitions required to realize the security model defined by this memo.

2.1. SNMPv2 Users

Management operations using this security model make use of a defined set of user identities. For any SNMPv2 user on whose behalf management operations are authorized at a particular SNMPv2 agent, that agent must have knowledge of that user. A SNMPv2 manager that wishes to communicate with a particular agent must also have knowledge of a user known to that agent, including knowledge of the applicable attributes of that user. Similarly, a SNMPv2 manager that wishes to receive messages from a SNMPv2 dual-role entity must have knowledge of the user on whose behalf the dual-role entity sends the message.

A user and its attributes are defined as follows:

<userName>
An octet string representing the name of the user.

<authProtocol>
An indication of whether messages sent on behalf of this user can be authenticated, and if so, the type of authentication protocol which is used. One such protocol is defined in this memo: the Digest Authentication Protocol.

<authPrivateKey>
If messages sent on behalf of this user can be authenticated, the (private) authentication key for use with the authentication protocol. Note that a user’s authentication key will normally be different at different agents.

<privProtocol>
An indication of whether messages sent on behalf of this user can be protected from disclosure, and if so, the type of privacy protocol which is used. One such protocol is defined in this memo: the Symmetric Encryption Protocol.

<privPrivateKey>
If messages sent on behalf of this user can be protected from disclosure, the (private) privacy key for use with the privacy protocol. Note that a user’s privacy key will normally be different at different agents.

2.2. Contexts and Context Selectors

An SNMPv2 context is a collection of management information accessible (locally or via proxy) by an SNMPv2 agent. An item of management information may exist in more than one context. An SNMPv2 agent potentially has access to many contexts. Each SNMPv2 message contains a context selector which unambiguously identifies an SNMPv2 context accessible by the SNMPv2 agent to which the message is directed or by the SNMPv2 agent associated with the sender of the message.

For a local SNMPv2 context which is realized by an SNMPv2 entity, that SNMPv2 entity uses locally-defined mechanisms to access the management information identified by the SNMPv2 context.

For a proxy SNMPv2 context, the SNMPv2 entity acts as a proxy SNMPv2 agent to access the management information identified by the SNMPv2 context.

The term remote SNMPv2 context is used at an SNMPv2 manager to indicate a SNMPv2 context (either local or proxy) which is not realized by the local SNMPv2 entity (i.e., the local SNMPv2 entity uses neither locally-defined mechanisms, nor acts as a proxy SNMPv2 agent to access the management information identified by the SNMPv2 context).

Proxy SNMPv2 contexts are further categorized as either local-proxy contexts or remote-proxy contexts. A proxy SNMPv2 agent receives Get/GetNext/GetBulk/Set operations for a local-proxy context, and forwards them with a remote-proxy context; it receives SNMPv2-Trap and Inform operations for a remote-proxy context, and forwards them with a local-proxy context; for Response operations, a proxy SNMPv2 agent receives them with either a local-proxy or remote-proxy context, and forwards them with a remote-proxy or local-proxy context, respectively.

For the non-proxy situation:

```
             context-A
Manager <----------------> Agent
```

the type of context is:

	context-A
Manager	remote
Agent	local
agentID	of Agent
contextSelector	locally unique

For proxy:

```
             context-B                context-C
Manager <----------------> Proxy <----------------> Agent
                           Agent
```

the type and identity of the contexts are:

	context-B	context-C
Manager	remote	--
Proxy-Agent	local-proxy	remote-proxy
Agent	--	local
agentID	of Proxy agent	of Agent
contextSelector	locally unique	locally unique

The combination of an agentID value and a context selector provides a globally-unique identification of a context. When a context is accessible by multiple agents (e.g., including by proxy SNMPv2 agents), it has multiple such globally-unique identifications, one associated with each agent which can access it. In the example above, "context-B" and "context-C" are different names for the same context.

2.3. Quality of Service (qoS)

Messages are generated with a particular Quality of Service (qoS), either:

- without authentication and privacy,
- with authentication but not privacy,
- with authentication and privacy.

All users are capable of having messages without authentication and privacy generated on their behalf. Users having an authentication protocol and an authentication key can have messages with authentication but not privacy generated on their behalf. Users having an authentication protocol, an authentication key, a privacy protocol and a privacy key can have messages with authentication and privacy generated on their behalf.

In addition to its indications of authentication and privacy, the qoS may also indicate that the message contains an operation that may result in a report PDU being generated (see Section 2.6 below).

2.4. Access Policy

An administration’s access policy determines the access rights of users. For a particular SNMPv2 context to which a user has access using a particular qoS, that user’s access rights are given by a list of authorized operations, and for a local context, a read-view and a write-view. The read-view is the set of object instances authorized for the user when reading objects. Reading objects occurs when processing a retrieval (get, get-next, get-bulk) operation and when sending a notification. The write-view is the set of object instances authorized for the user when writing objects. Writing objects occurs when processing a set operation. A user’s access rights may be different at different agents.

2.5. Replay Protection

Each SNMPv2 agent (or dual-role entity) maintains three objects:

- agentID, which is an identifier unique among all agents in (at least) an administrative domain;
- agentBoots, which is a count of the number of times the agent has rebooted/re-initialized since agentID was last configured; and,

- agentTime, which is the number of seconds since agentBoots was last incremented.

An SNMPv2 agent is always authoritative with respect to these variables. It is the responsibility of an SNMPv2 manager to synchronize with the agent, as appropriate. In the case of an SNMPv2 dual-role entity sending an Inform-Request, it is that entity acting in an agent role which is authoritative with respect to these variables for the Inform-Request.

An agent is required to maintain the values of agentID and agentBoots in non-volatile storage.

2.5.1. agentID

The agentID value contained in an authenticated message is used to defeat attacks in which messages from a manager are replayed to a different agent and/or messages from one agent (or dual-role entity) are replayed as if from a different agent (or dual-role entity).

When an agent (or dual-role entity) is first installed, it sets its local value of agentID according to a enterprise-specific algorithm (see the definition of agentID in Section 4.1).

2.5.2. agentBoots and agentTime

The agentBoots and agentTime values contained in an authenticated message are used to defeat attacks in which messages are replayed when they are no longer valid. Through use of agentBoots and agentTime, there is no requirement for an SNMPv2 agent to have a non-volatile clock which ticks (i.e., increases with the passage of time) even when the agent is powered off. Rather, each time an SNMPv2 agent reboots, it retrieves, increments, and then stores agentBoots in non-volatile storage, and resets agentTime to zero.

When an agent (or dual-role entity) is first installed, it sets its local values of agentBoots and agentTime to zero. If agentTime ever reaches its maximum value (2147483647), then agentBoots is incremented as if the agent has rebooted and agentTime is reset to zero and starts incrementing again.

Each time an agent (or dual-role entity) reboots, any SNMPv2 managers holding that agent’s values of agentBoots and agentTime need to re-synchronize prior to sending correctly authenticated messages to that agent (see Section 2.7 for re-synchronization procedures). Note, however, that the procedures do provide for a notification to be accepted as authentic by a manager, when sent by an agent which has rebooted since the manager last re-synchronized.

If an agent (or dual-role entity) is ever unable to determine its latest agentBoots value, then it must set its agentBoots value to 0xffffffff.

Whenever the local value of agentBoots has the value 0xffffffff, it latches at that value and an authenticated message always causes an usecStatsNotInWindows authentication failure.

In order to reset an agent whose agentBoots value has reached the value 0xffffffff, manual intervention is required. The agent must be physically visited and re-configured, either with a new agentID value, or with new secret values for the authentication and privacy keys of all users known to that agent.

2.5.3. Time Window

The Time Window is a value that specifies the window of time in which a message generated on behalf of any user is valid. This memo specifies that the same value of the Time Window, 150 seconds, is used for all users.

2.6. Error Reporting

While processing a received communication, an SNMPv2 entity may determine that the message is unacceptable (see Section 3.2). In this case, the appropriate counter from the snmpGroup [15] or usecStatsGroup object groups is incremented and the received message is discarded without further processing.

If an SNMPv2 entity acting in the agent role makes such a determination and the qoS indicates that a report may be generated, then after incrementing the appropriate counter, it is required to generate a message containing a report PDU, with the same user and context as the received message, and to send it to the transport address which originated the received message. For all report PDUs, except those generated due to incrementing the usecStatsNotInWindows counter, the report PDU is unauthenticated. For those generated due to incrementing usecStatsNotInWindows, the report PDU is authenticated only if the received message was authenticated.

The report flag in the qoS may only be set if the message contains a Get, GetNext, GetBulk, Set operation. The report flag should never be set for a message that contains a Response, Inform, SNMPv2-Trap or Report operation. Furthermore, a report PDU is never sent by an SNMPv2 entity acting in a manager role.

2.7. Time Synchronization

Time synchronization, required by a management entity in order to proceed with authentic communications, has occurred when the management entity has obtained local values of agentBoots and agentTime from the agent that are within the agent’s time window. To remain synchronized, the local values must remain within the agent’s time window and thus must be kept loosely synchronized with the values stored at the agent. In addition to keeping a local version of agentBoots and agentTime, a manager must also keep one other local variable, latestReceivedAgentTime. This value records the highest value of agentTime that was received by the manager from the agent and is used to eliminate the possibility of replaying messages that would prevent the manager’s notion of the agentTime from advancing.

Time synchronization occurs as part of the procedures of receiving a message (Section 3.2, step 9d). As such, no explicit time synchronization procedure is required by a management entity. Note, that whenever the local value of agentID is changed (e.g., through discovery) or when a new secret is configured, the local values of agentBoots and latestReceivedAgentTime should be set to zero. This will cause the time synchronization to occur when the next authentic message is received.

2.8. Proxy Error Propagation

When a proxy SNMPv2 agent receives a report PDU from a proxied agent and it is determined that a proxy-forwarded request cannot be delivered to the proxied agent, then the snmpProxyDrops counter [15] is incremented and a report PDU is generated and transmitted to the transport address from which the original request was received. (Note that the receipt of a report PDU containing snmpProxyDrops as a VarBind, is included among the reasons why a proxy-forwarded request cannot be delivered.)

2.9. SNMPv2 Messages Using this Model

The syntax of an SNMPv2 message using this security model differs from that of an SNMPv1 [2] message as follows:

- The version component is changed to 2.

- The data component contains either a PDU or an OCTET STRING containing an encrypted PDU.

The SNMPv1 community string is now termed the "parameters" component and contains a set of administrative information for the message.

Only the PDU is protected from disclosure by the privacy protocol. This exposes the administrative information to eavesdroppers. However, malicious use of this information is considered to be a Traffic Analysis attack against which protection is not provided.

For an authenticated SNMPv2 message, the message digest is applied to the entire message given to the transport service. As such, message generation first privatizes the PDU, then adds the message wrapper, and then authenticates the message.

An SNMPv2 message is an ASN.1 value with the following syntax:

```
Message ::=
    SEQUENCE {
        version
            INTEGER { v2 (2) },

        parameters
            OCTET STRING,
        -- <model=1>
        --      <qoS><agentID><agentBoots><agentTime><maxSize>
        --      <userLen><userName><authLen><authDigest>
        --      <contextSelector>

        data
            CHOICE {
                plaintext
                    PDUs,
                encrypted
                    OCTET STRING
            }
    }
```

where:

parameters
a concatenation of the following values in network-byte order. If the first octet (<model>) is one, then

<qoS> = 8-bits of quality-of-service

bitnumber 7654 3210	meaning
.... ..00	no authentication nor privacy
.... ..01	authentication, no privacy
.... ..1.	authentication and privacy
.... .1..	generation of report PDU allowed

where bit 7 is the most significant bit.

<agentID> = 12 octets
a unique identifier for the agent (or dual-role entity).

<agentBoots> = 32-bits
an unsigned quantity (0..4294967295) in network-byte order.

<agentTime> = 32-bits
an unsigned quantity (0..2147483647) in network-byte order.

<maxSize> = 16-bits
an unsigned quantity (484..65507) in network-byte order, which identifies the maximum message size which the sender of this message can receive using the same transport domain as used for this message.

<userLen> = 1 octet
the length of following <userName> field.

<userName> = 1..16 arbitrary octets
the user on whose behalf this message is sent.

<authLen> = 1 octet
the length of following <authDigest> field.

<authDigest> = 0..255 octets
for authenticated messages, the authentication digest. Otherwise, the value has zero-length on transmission and is ignored on receipt.

<contextSelector> = 0..40 arbitrary octets
the context selector which in combination with agentID identifies the SNMPv2 context containing the management information referenced by the SNMPv2 message.

plaintext
an SNMPv2 PDU as defined in [12].

encrypted
the encrypted form of an SNMPv2 PDU.

2.10. Local Configuration Datastore (LCD)

Each SNMPv2 entity maintains a local conceptually database, called the Local Configuration Datastore (LCD), which holds its known set of information about SNMPv2 users and other associated (e.g., access control) information. An LCD may potentially be required to hold information about multiple SNMPv2 agent entities. As such, the <agentID> should be used to identify a particular agent entity in the LCD.

It is a local implementation issue as to whether information in the LCD is stored information or whether it is obtained dynamically (e.g., as a part of an SNMPv2 manager's API) on an as-needed basis.

3. Elements of Procedure

This section describes the procedures followed by an SNMPv2 entity in processing SNMPv2 messages.

3.1. Generating a Request or Notification

This section describes the procedure followed by an SNMPv2 entity whenever it generates a message containing a management operation (either a request or a notification) on behalf of a user, for a particular context and with a particular qoS value.

(1) Information concerning the user is extracted from the LCD. The transport domain and transport address to which the operation is to be sent is determined. The context is resolved into an agentID value and a contextSelector value.

(2) If the qoS specifies that the message is to be protected from disclosure, but the user does not support both an authentication and a privacy protocol, or does not have configured authentication and privacy keys, then the operation cannot be sent.

(3) If the qoS specifies that the message is to be authenticated, but the user does not support an authentication protocol, or does not have a configured authentication key, then the operation cannot be sent.

(4) The operation is serialized (i.e., encoded) according to the conventions of [13] and [12] into a PDUs value.

(5) If the operation is a Get, GetNext, GetBulk, or Set then the report flag in the qoS is set to the value 1.

(6) An SNMPv2 message is constructed using the ASN.1 Message syntax:

  - the version component is set to the value 2.

  - if the qoS specifies that the message is to be protected from disclosure, then the octet sequence representing the serialized PDUs value is encrypted according to the user's privacy protocol and privacy key, and the encrypted data is encoded as an octet string and is used as the data component of the message.

- if the qoS specifies that the message is not to be protected from disclosure, then the serialized PDUs value is used directly as the value of the data component.
- the parameters component is constructed using:
  - the requested qoS, userName, agentID and context selector,
  - if the qoS specifies that the message is to be authenticated or the management operation is a notification, then the current values of agentBoots, and agentTime corresponding to agentID from the LCD are used. Otherwise, the <agentBoots> and <agentTime> fields are set to zero-filled octets.
  - the <maxSize> field is set to the maximum message size which the local SNMPv2 entity can receive using the transport domain which will be used to send this message.
  - if the qoS specifies that the message is to be authenticated, then the <authDigest> field is temporarily set to the user's authentication key. Otherwise, the <authDigest> field is set to the zero-length string.

(7) The constructed Message value is serialized (i.e., encoded) according to the conventions of [13] and [12].

(8) If the qoS specifies that the message is to be authenticated, then an MD5 digest value is computed over the octet sequence representing the concatenation of the serialized Message value and the user's authentication key. The <authDigest> field is then set to the computed digest value.

(9) The serialized Message value is transmitted to the determined transport address.

3.2. Processing a Received Communication

This section describes the procedure followed by an SNMPv2 entity whenever it receives an SNMPv2 message. This procedure is independent of the transport service address at which the message was received. For clarity, some of the details of this procedure are left out and are described in Section 3.2.1 and its sub-sections.

(1) The snmpInPkts counter [15] is incremented. If the received message is not the serialization (according to the conventions of

[13]) of a Message value, then the snmpInASNParseErrs counter [15] is incremented, and the message is discarded without further processing.

(2) If the value of the version component has a value other than 2, then the message is either processed according to some other version of this protocol, or the snmpInBadVersions counter [15] is incremented, and the message is discarded without further processing.

(3) The value of the <model> field is extracted from the parameters component of the Message value. If the value of the <model> field is not 1, then either the message is processed according to some other security model, or the usecStatsBadParameters counter is incremented, and the message is discarded without further processing.

(4) The values of the rest of the fields are extracted from the parameters component of the Message value.

(5) If the <agentID> field contained in the parameters is unknown then:

- a manager that performs discovery may optionally create a new LCD entry and continue processing; or

- the usecStatsUnknownContexts counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

(6) The LCD is consulted for information about the SNMPv2 context identified by the combination of the <agentID> and <contextSelector> fields. If information about this SNMPv2 context is absent from the LCD, then the usecStatsUnknownContexts counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

(7) Information about the value of the <userName> field is extracted from the LCD. If no information is available, then the usecStatsUnknownUserNames counter is incremented, a report PDU [1] is generated, and the received message is discarded without further processing.

(8) If the information about the user indicates that it does not support the quality of service indicated by the <qoS> field, then the usecStatsUnsupportedQoS counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

(9) If the <qoS> field indicates an authenticated message and the user’s authentication protocol is the Digest Authentication Protocol described in this memo, then:

a) the local values of agentBoots and agentTime corresponding to the value of the <agentID> field are extracted from the LCD.

b) the value of <authDigest> field is temporarily saved. A new serialized Message is constructed which differs from that received in exactly one respect: that the <authDigest> field within it has the value of the user’s authentication key. An MD5 digest value is computed over the octet sequence representing the concatenation of the new serialized Message and the user’s authentication key.

c) if the LCD information indicates the SNMPv2 context is of type local (i.e., an agent), then:

- if the computed digest differs from the saved authDigest value, then the usecStatsWrongDigestValues counter is incremented, a report PDU is generated, and the received message is discarded without further processing. However, if the snmpEnableAuthenTraps object [15] is enabled, then the SNMPv2 entity sends authenticationFailure traps [15] according to its configuration.

- if any of the following conditions is true, then the message is considered to be outside of the Time Window:

  - the local value of agentBoots is 0xffffffff;

  - the <agentBoots> field differs from the local value of agentBoots; or,

  - the value of the <agentTime> field differs from the local notion of agentTime by more than +/- 150 seconds.

- if the message is considered to be outside of the Time Window then the usecStatsNotInWindows counter is incremented, an authenticated report PDU is generated (see section 2.7), and the received message is discarded without further processing.

d) if the LCD information indicates the SNMPv2 context is not realized by the local SNMPv2 entity (i.e., a manager), then:

- if the computed digest differs from the saved authDigest value, then the usecStatsWrongDigestValues counter is incremented and the received message is discarded without further processing.

- if all of the following conditions are true:

    - if the <qoS> field indicates that privacy is not in use;

    - the SNMPv2 operation type determined from the ASN.1 tag value associated with the PDU's component is a Report;

    - the Report was generated due to a usecStatsNotInWindows error condition; and,

    - the <agentBoots> field is greater than the local value of agentBoots, or the <agentBoots> field is equal to the local value of agentBoots and the <agentTime> field is greater than the value of latestReceivedAgentTime,

  then the LCD entry corresponding to the value of the <agentID> field is updated, by setting the local value of agentBoots from the <agentBoots> field, the value latestReceivedAgentTime from the <agentTime> field, and the local value of agentTime from the <agentTime> field.

- if any of the following conditions is true, then the message is considered to be outside of the Time Window:

  - the local value of agentBoots is 0xffffffff;

  - the <agentBoots> field is less than the local value of agentBoots; or,

  - the <agentBoots> field is equal to the local value of agentBoots and the <agentTime> field is more than 150 seconds less than the local notion of agentTime.

- if the message is considered to be outside of the Time Window then the usecStatsNotInWindows counter is incremented, and the received message is discarded without further processing; however, time synchronization procedures may be invoked. Note that this procedure allows for <agentBoots> to be greater than the local value of agentBoots to allow for received messages to be accepted as authentic when received from an agent that has rebooted since the manager last re-synchronized.

- if at least one of the following conditions is true:

    - the <agentBoots> field is greater than the local value of agentBoots; or,

- the <agentBoots> field is equal to the local value of agentBoots and the <agentTime> field is greater than the value of latestReceivedAgentTime,

then the LCD entry corresponding to the value of the <agentID> field is updated, by setting the local value of agentBoots from the <agentBoots> field, the local value latestReceivedAgentTime from the <agentTime> field, and the local value of agentTime from the <agentTime> field.

(10) If the <qoS> field indicates use of a privacy protocol, then the octet sequence representing the data component is decrypted according to the user’s privacy protocol to obtain a serialized PDUs value. Otherwise the data component is assumed to directly contain the PDUs value.

(11) The SNMPv2 operation type is determined from the ASN.1 tag value associated with the PDUs component.

(12) If the SNMPv2 operation type is a Report, then the request-id in the PDU is correlated to an outstanding request, and if the correlation is successful, the appropriate action is taken (e.g., time synchronization, proxy error propagation, etc.); in particular, if the report PDU indicates a usecStatsNotInWindows condition, then the outstanding request may be retransmitted (since the procedure in Step 9d above should have resulted in time synchronization).

(13) If the SNMPv2 operation type is either a Get, GetNext, GetBulk, or Set operation, then:

a) if the LCD information indicates that the SNMPv2 context is of type remote or remote-proxy, then the usecStatsUnauthorizedOperations counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

b) the LCD is consulted for access rights authorized for communications using the indicated qoS, on behalf of the indicated user, and concerning management information in the indicated SNMPv2 context for the particular SNMPv2 operation type.

c) if the SNMPv2 operation type is not among the authorized access rights, then the usecStatsUnauthorizedOperations counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

d) The information extracted from the LCD concerning the user and the SNMPv2 context, together with the sending transport address of the received message is cached for later use in generating a response message.

e) if the LCD information indicates the SNMPv2 context is of type local, then the management operation represented by the PDUs value is performed by the receiving SNMPv2 entity with respect to the relevant MIB view within the SNMPv2 context according to the procedures set forth in [12], where the relevant MIB view is determined according to the user, the agentID, the contextSelector, the qoS values and the type of operation requested.

f) if the LCD information indicates the SNMPv2 context is of type local-proxy, then:

   i. the user, qoS, agentID, contextSelector and transport address to be used to forward the request are extracted from the LCD. If insufficient information concerning the user is currently available, then snmpProxyDrops counter [15] is incremented, a report PDU is generated, and the received message is discarded.

   ii. if an administrative flag in the LCD indicates that the message is to be forwarded using the SNMPv1 administrative framework, then the procedures described in [4] are invoked. Otherwise, a new SNMPv2 message is constructed: its PDUs component is copied from that in the received message except that the contained request-id is replaced by a unique value (this value will enable a subsequent response message to be correlated with this request); the <userName>, <qoS>, <agentID> and <contextSelector> fields are set to the values extracted from the LCD; the <maxSize> field is set to the minimum of the value in the received message and the local system’s maximum message size for the transport domain which will be used to forward the message; and finally, the message is authenticated and/or protected from disclosure according to the qoS value.

   iii. the information cached in Step 13d above is augmented with the request-id of the received message as well as the request-id, agentID and contextSelector of the constructed message.

   iv. the constructed message is forwarded to the extracted transport address.

(14) If the SNMPv2 operation type is an Inform, then:

a) if the LCD information indicates the SNMPv2 context is of type local or local-proxy then the usecStatsUnauthorizedOperations counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

b) if the LCD information indicates the SNMPv2 context is of type remote, then the Inform operation represented by the PDUs value is performed by the receiving SNMPv2 entity according to the procedures set forth in [12].

c) if the LCD information indicates the SNMPv2 context is of type remote-proxy, then:

i. a single unique request-id is selected for use by all forwarded copies of this request. This value will enable the first response message to be correlated with this request; other responses are not required and should be discarded when received, since the agent that originated the Inform only requires one response to its Inform.

ii. information is extracted from the LCD concerning all combinations of userName, qoS, agentID, contextSelector and transport address with which the received message is to be forwarded.

iii. for each such combination whose access rights permit Inform operations to be forwarded, a new SNMPv2 message is constructed, as follows: its PDUs component is copied from that in the received message except that the contained request-id is replaced by the value selected in Step i above; its <userName>, <qoS>, <agentID> and <contextSelector> fields are set to the values extracted in Step ii above; and its <maxSize> field is set to the minimum of the value in the received message and the local system’s maximum message size for the transport domain which will be used to forward this message.

iv. for each constructed SNMPv2 message, information concerning the <userName>, <qoS>, <agentID>, <contextSelector>, request-id and sending transport address of the received message, as well as the request- id, agentID and contextSelector of the constructed message, is cached for later use in generating a response message.

v. each constructed message is forwarded to the appropriate transport address extracted from the LCD in step ii above.

(15) If the SNMPv2 operation type is a Response, then:

a) if the LCD information indicates the SNMPv2 context is of type local, then the usecStatsUnauthorizedOperations counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

b) if the LCD information indicates the SNMPv2 context is of type remote, then the Response operation represented by the PDUs value is performed by the receiving SNMPv2 entity according to the procedures set forth in [12].

c) if the LCD information indicates the SNMPv2 context is of type local-proxy or remote-proxy, then:

i. the request-id is extracted from the PDUs component of the received message. The context's agentID and contextSelector values together with the extracted request-id are used to correlate this response message to the corresponding values for a previously forwarded request by inspecting the cache of information as augmented in Substep iii of Step 13f above or in Substep iv of 14c above. If no such correlated information is found, then the received message is discarded without further processing.

ii. a new SNMPv2 message is constructed: its PDUs component is copied from that in the received message except that the contained request-id is replaced by the value saved in the correlated information from the original request; its <userName>, <qoS>, <agentID> and <contextSelector> fields are set to the values saved from the received message. The <maxSize> field is set to the minimum of the value in the received message and the local system's maximum message size for the transport domain which will be used to forward the message. The message is authenticated and/or protected from disclosure according to the saved qoS value.

iii. the constructed message is forwarded to the transport address saved in the correlated information as the sending transport address of the original request.

iv. the correlated information is deleted from the cache of information.

(16) If the SNMPv2 operation type is a SNMPv2-Trap, then:

a) if the LCD information indicates the SNMPv2 context is of type local or local-proxy, then the usecStatsUnauthorizedOperations counter is incremented, a report PDU is generated, and the received message is discarded without further processing.

b) if the LCD information indicates the SNMPv2 context is of type remote, then the SNMPv2-Trap operation represented by the PDUs value is performed by the receiving SNMPv2 entity according to the procedures set forth in [12].

c) if the LCD information indicates the SNMPv2 context is of type remote-proxy, then:

   i. a unique request-id is selected for use in forwarding the message.

   ii. information is extracted from the LCD concerning all combinations of userName, qoS, agentID, contextSelector and transport address with which the received message is to be forwarded.

   iii. for each such combination whose access rights permit SNMPv2-Trap operations to be forwarded, a new SNMPv2 message is constructed, as follows: its PDUs component is copied from that in the received message except that the contained request-id is replaced by the value selected in Step i above; its <userName>, <qoS>, <agentID> and <contextSelector> fields are set to the values extracted in Step ii above.

   iv. each constructed message is forwarded to the appropriate transport address extracted from the LCD in step ii above.

3.2.1. Additional Details

For the sake of clarity and to prevent the above procedure from being even longer, the following details were omitted from the above procedure.

3.2.1.1. ASN.1 Parsing Errors

For ASN.1 parsing errors, the snmpInASNParseErrs counter [15] is incremented and a report PDU is generated whenever such an ASN.1 parsing error is discovered. However, if the parsing error causes the information able to be extracted from the message to be insufficient for generating a report PDU, then the report PDU is not sent.

3.2.1.2. Incorrectly Encoded Parameters

For an incorrectly encoded parameters component of the Message value (e.g., incorrect or inconsistent value of the <userLen> or <authLen> fields), the usecStatsBadParameters counter is incremented. Since the encoded parameters are in error, the report flag in the qoS cannot be reliably determined. Thus, no report PDU is generated for the incorrectly encoded parameters error condition.

3.2.1.3. Generation of a Report PDU

Some steps specify that the received message is discarded without further processing whenever a report PDU is generated. However:

- An SNMPv2 manager never generates a report PDU.

- If the operation type can reliably be determined and it is determined to be a Report, SNMPv2-Trap, Inform, or a Response then a report PDU is not generated.

- A report PDU is only generated when the report flag in the qoS is set to the value 1.

A generated report PDU must always use the current values of agentID, agentBoots, and agentTime from the LCD. In addition, a generated report PDU must whenever possible contain the same request-id value as in the PDU contained in the received message. Meeting this constraint normally requires the message to be further processed just enough so as to extract its request-id. There are two situations in which the SNMPv2 request-id cannot be determined. The first situation occurs when the userName is unknown and the qoS indicates that the message is encrypted. The other situation is when there is an ASN.1 parsing error. In cases where the the request-id cannot be determined, the default request-id value 2147483647 is used.

3.2.1.4. Cache Timeout

Some steps specify that information is cached so that a Response operation may be correlated to the appropriate Request operation. However, a number of situations could cause the cache to grow without bound. One such situation is when the Response operation does not arrive or arrives "late" at the entity. In order to ensure that the cache does not grow without bound, it is recommended that cache entries be deleted when they are determined to be no longer valid. It is an implementation dependent decision as to how long cache entries remain valid, however, caching entries more than 150 seconds is not useful since any use of the cache entry after that time would generate a usecStatsNotInWindows error condition.

3.3. Generating a Response

The procedure for generating a response to an SNMPv2 management request is identical to the procedure for transmitting a request (see Section 3.1), with these exceptions:

- The response is sent on behalf of the same user and with the same value of the agentID and contextSelector as the request.

- The PDUs value of the responding Message value is the response which results from performing the operation specified in the original PDUs value.

- The authentication protocol and other relevant information for the user is obtained, not from the LCD, but rather from information cached (in Step 13d) when processing the original message.

- The serialized Message value is transmitted using any transport address belonging to the agent for the transport domain from which the corresponding request originated - even if that is different from any transport information obtained from the LCD.

- If the qoS specifies that the message is to be authenticated or the response is being generated by a SNMPv2 entity acting in an agent role, then the current values of agentBoots and agentTime from the LCD are used. Otherwise, the <agentBoots> and <agentTime> fields are set to zero-filled octets.

- The report flag in the qoS is set to the value 0.

4. Discovery

This security model requires that a discovery process obtain sufficient information about an SNMPv2 entity’s agent in order to communicate with it. Discovery requires the SNMPv2 manager to learn the agent’s agentID value before communication may proceed. This may be accomplished by formulating a get-request communication with the qoS set to noAuth/noPriv, the userName set to "public", the agentID set to all zeros (binary), the contextSelector set to "", and the VarBindList left empty. The response to this message will be an reportPDU that contains the agentID within the <parameters> field (and containing the usecStatsUnknownContexts counter in the VarBindList). If authenticated communication is required then the discovery process may invoke the procedure described in Section 2.7 to synchronize the clocks.

5. Definitions

```
SNMPv2-USEC-MIB DEFINITIONS ::= BEGIN

IMPORTS
    MODULE-IDENTITY, OBJECT-TYPE, Counter32, Unsigned32,
    snmpModules
        FROM SNMPv2-SMI
    TEXTUAL-CONVENTION
        FROM SNMPv2-TC
    MODULE-COMPLIANCE, OBJECT-GROUP
        FROM SNMPv2-CONF;


usecMIB MODULE-IDENTITY
    LAST-UPDATED "9601120000Z"
    ORGANIZATION "IETF SNMPv2 Working Group"
    CONTACT-INFO
            "        Glenn W. Waters

             Postal: Bell-Northern Research, Ltd.
                     P.O. Box 3511, Station C
                     Ottawa, ON, K1Y 4H7
                     Canada

                Tel: +1 613 763 3933

             E-mail: gwaters@bnr.ca"
    DESCRIPTION
            "The MIB module for SNMPv2 entities implementing the user-
            based security model."
    ::= { snmpModules 6 }


usecMIBObjects OBJECT IDENTIFIER ::= { usecMIB 1 }


-- Textual Conventions

AgentID ::= TEXTUAL-CONVENTION
    STATUS       current
    DESCRIPTION
            "An agent’s administratively-unique identifier.

            The value for this object may not be all zeros or all ’ff’H.

            The initial value for this object may be configured via an
            operator console entry or via an algorithmic function.  In
            the later case, the following guidelines are recommended:

              1) The first four octets are set to the binary equivalent
                 of the agent’s SNMP network management private
                 enterprise number as assigned by the Internet Assigned
                 Numbers Authority (IANA).  For example, if Acme
                 Networks has been assigned { enterprises 696 }, the
                 first four octets would be assigned ’000002b8’H.

              2) The remaining eight octets are the cookie whose
                 contents are determined via one or more enterprise-
                 specific methods.  Such methods must be designed so as
                 to maximize the possibility that the value of this
                 object will be unique in the agent’s administrative
                 domain.  For example, the cookie may be the IP address
                 of the agent, or the MAC address of one of the
                 interfaces, with each address suitably padded with
                 random octets.  If multiple methods are defined, then
                 it is recommended that the cookie be further divided
                 into one octet that indicates the method being used and
                 seven octets which are a function of the method."
    SYNTAX     OCTET STRING (SIZE (12))


-- the USEC Basic group
--
-- a collection of objects providing basic instrumentation of
-- the SNMPv2 entity implementing the user-based security model


usecAgent       OBJECT IDENTIFIER ::= { usecMIBObjects 1 }

agentID OBJECT-TYPE
    SYNTAX     AgentID
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The agent’s administratively-unique identifier."
    ::= { usecAgent 1 }

agentBoots OBJECT-TYPE
    SYNTAX     Unsigned32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of times that the agent has re-initialized
            itself since its initial configuration."
    ::= { usecAgent 2 }
```

```
agentTime OBJECT-TYPE
    SYNTAX     Unsigned32 (0..2147483647)
    UNITS      "seconds"
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The number of seconds since the agent last incremented the
            agentBoots object."
    ::= { usecAgent 3 }

agentSize OBJECT-TYPE
    SYNTAX     INTEGER (484..65507)
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The maximum length in octets of an SNMPv2 message which
            this agent will accept using any transport mapping."
    ::= { usecAgent 4 }


-- USEC statistics
--
-- a collection of objects providing basic instrumentation of
-- the SNMPv2 entity implementing the user-based security model

usecStats      OBJECT IDENTIFIER ::= { usecMIBObjects 2 }


usecStatsUnsupportedQoS OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because they requested a quality-of-
            service that was unknown to the agent or otherwise
            unavailable."
    ::= { usecStats 1 }

usecStatsNotInWindows OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because they appeared outside of the
            agent’s window."
    ::= { usecStats 2 }
```

```
usecStatsUnknownUserNames OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because they referenced a user that was
            not known to the agent."
    ::= { usecStats 3 }

usecStatsWrongDigestValues OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because they didn’t contain the expected
            digest value."
    ::= { usecStats 4 }

usecStatsUnknownContexts OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because they referenced a context that
            was not known to the agent."
    ::= { usecStats 5 }

usecStatsBadParameters OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because the <parameters> field was
            improperly encoded or had invalid syntax."
    ::= { usecStats 6 }

usecStatsUnauthorizedOperations OBJECT-TYPE
    SYNTAX     Counter32
    MAX-ACCESS read-only
    STATUS     current
    DESCRIPTION
            "The total number of packets received by the SNMPv2 entity
            which were dropped because the PDU type referred to an
            operation that is invalid or not authorized."
    ::= { usecStats 7 }


-- conformance information

usecMIBConformance
               OBJECT IDENTIFIER ::= { usecMIB 2 }

usecMIBCompliances
               OBJECT IDENTIFIER ::= { usecMIBConformance 1 }
usecMIBGroups  OBJECT IDENTIFIER ::= { usecMIBConformance 2 }


-- compliance statements

usecMIBCompliance MODULE-COMPLIANCE
    STATUS  current
    DESCRIPTION
            "The compliance statement for SNMPv2 entities which
            implement the SNMPv2 USEC model."
    MODULE  -- this module
        MANDATORY-GROUPS { usecBasicGroup,
                           usecStatsGroup }
    ::= { usecMIBCompliances 1 }


-- units of conformance

usecBasicGroup OBJECT-GROUP
    OBJECTS { agentID,
              agentBoots,
              agentTime,
              agentSize }
    STATUS  current
    DESCRIPTION
            "A collection of objects providing identification, clocks,
            and capabilities of an SNMPv2 entity which implements the
            SNMPv2 USEC model."
    ::= { usecMIBGroups 1 }

usecStatsGroup OBJECT-GROUP
    OBJECTS { usecStatsUnsupportedQoS,
              usecStatsNotInWindows,
              usecStatsUnknownUserNames,
              usecStatsWrongDigestValues,
              usecStatsUnknownContexts,
              usecStatsBadParameters,
              usecStatsUnauthorizedOperations }
    STATUS  current
    DESCRIPTION
            "A collection of objects providing basic error statistics of
            an SNMPv2 entity which implements the SNMPv2 USEC model."
    ::= { usecMIBGroups 2 }

END
```

6. Security Considerations

6.1. Recommended Practices

This section describes practices that contribute to the secure, effective operation of the mechanisms defined in this memo.

- A management station must discard SNMPv2 responses for which neither the request-id component nor the represented management information corresponds to any currently outstanding request.

  Although it would be typical for a management station to do this as a matter of course, when using these security protocols it is significant due to the possibility of message duplication (malicious or otherwise).

- A management station must generate unpredictable request-ids in authenticated messages in order to protect against the possibility of message duplication (malicious or otherwise).

- A management station should perform time synchronization using authenticated messages in order to protect against the possibility of message duplication (malicious or otherwise).

- When sending state altering messages to a managed agent, a management station should delay sending successive messages to the managed agent until a positive acknowledgement is received for the previous message or until the previous message expires.

  No message ordering is imposed by the SNMPv2. Messages may be received in any order relative to their time of generation and each will be processed in the ordered received. Note that when an authenticated message is sent to a managed agent, it will be valid for a period of time of approximately 150 seconds under normal circumstances, and is subject to replay during this period. Indeed, a management station must cope with the loss and re-ordering of messages resulting from anomalies in the network as a matter of course.

However, a managed object, snmpSetSerialNo [15], is specifically defined for use with SNMPv2 set operations in order to provide a mechanism to ensure the processing of SNMPv2 messages occurs in a specific order.

- The frequency with which the secrets of an SNMPv2 user should be changed is indirectly related to the frequency of their use.

  Protecting the secrets from disclosure is critical to the overall security of the protocols. Frequent use of a secret provides a continued source of data that may be useful to a cryptanalyst in exploiting known or perceived weaknesses in an algorithm. Frequent changes to the secret avoid this vulnerability.

  Changing a secret after each use is generally regarded as the most secure practice, but a significant amount of overhead may be associated with that approach.

  Note, too, in a local environment the threat of disclosure may be less significant, and as such the changing of secrets may be less frequent. However, when public data networks are the communication paths, more caution is prudent.

6.2. Defining Users

The mechanisms defined in this document employ the notion of "users" having access rights. How "users" are defined is subject to the security policy of the network administration. For example, users could be individuals (e.g., "joe" or "jane"), or a particular role (e.g., "operator" or "administrator"), or a combination (e.g., "joe-operator", "jane-operator" or "joe-admin"). Furthermore, a "user" may be a logical entity, such as a manager station application or set of manager station applications, acting on behalf of a individual or role, or set of individuals, or set of roles, including combinations.

Appendix A describes an algorithm for mapping a user "password" to a 16 octet value for use as either a user’s authentication key or privacy key (or both). Passwords are often generated, remembered, and input by a human. Human-generated passwords may be less than the 16 octets required by the authentication and privacy protocols, and brute force attacks can be quite easy on a relatively short ASCII character set. Therefore, the algorithm is Appendix A performs a transformation on the password. If the Appendix A algorithm is used, agent implementations (and agent configuration applications) must ensure that passwords are at least 8 characters in length.

Because the Appendix A algorithm uses such passwords (nearly) directly, it is very important that they not be easily guessed. It is suggested that they be composed of mixed-case alphanumeric and punctuation characters that don’t form words or phrases that might be found in a dictionary. Longer passwords improve the security of the system. Users may wish to input multiword phrases to make their password string longer while ensuring that it is memorable.

Note that there is security risk in configuring the same "user" on multiple systems where the same password is used on each system, since the compromise of that user’s secrets on one system results in the compromise of that user on all other systems having the same password.

The algorithm in Appendix A avoids this problem by including the agent’s agentID value as well as the user’s password in the calculation of a user’s secrets; this results in the user’s secrets being different at different agents; however, if the password is compromised the algorithm in Appendix A is not effective.

6.3. Conformance

To be termed a "Secure SNMPv2 implementation", an SNMPv2 implementation:

- must implement the Digest Authentication Protocol.

- must, to the maximal extent possible, prohibit access to the secret(s) of each user about which it maintains information in a LCD, under all circumstances except as required to generate and/or validate SNMPv2 messages with respect to that user.

- must implement the SNMPv2 USEC MIB.

In addition, an SNMPv2 agent must provide initial configuration in accordance with Appendix A.1.

Implementation of the Symmetric Encryption Protocol is optional.

7. Editor’s Address

Glenn W. Waters
Bell-Northern Research Ltd.
P.O. Box 3511, Station C
Ottawa, Ontario K1Y 4H7
CA

Phone: +1 613 763 3933
EMail: gwaters@bnr.ca

8. Acknowledgements

This document is the result of significant work by three major contributors:

Keith McCloghrie (Cisco Systems, kzm@cisco.com)
Marshall T. Rose (Dover Beach Consulting, mrose@dbc.mtview.ca.us)
Glenn W. Waters (Bell-Northern Research Ltd., gwaters@bnr.ca)

The authors wish to acknowledge James M. Galvin of Trusted Information Systems who contributed significantly to earlier work on which this memo is based, and the general contributions of members of the SNMPv2 Working Group, and, in particular, Aleksey Y. Romanov and Steven L. Waldbusser.

A special thanks is extended for the contributions of:

Uri Blumenthal (IBM)
Shawn Routhier (Epilogue)
Barry Sheehan (IBM)
Bert Wijnen (IBM)

9. References

[1] McCloghrie, K., Editor, "An Administrative Infrastructure for SNMPv2", RFC 1909, Cisco Systems, January 1996.

[2] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol", STD 15, RFC 1157, SNMP Research, Performance Systems International, MIT Laboratory for Computer Science, May 1990.

[3] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, MIT Laboratory for Computer Science, April 1992.

[4] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", RFC 1908, January 1996.

[5] Data Encryption Standard, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 46-1. Supersedes FIPS Publication 46, (January, 1977; reaffirmed January, 1988).

[6] Data Encryption Algorithm, American National Standards Institute. ANSI X3.92-1981, (December, 1980).

[7] DES Modes of Operation, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 81, (December, 1980).

[8] Data Encryption Algorithm - Modes of Operation, American National Standards Institute. ANSI X3.106-1983, (May 1983).

[9] Guidelines for Implementing and Using the NBS Data Encryption Standard, National Institute of Standards and Technology. Federal Information Processing Standard (FIPS) Publication 74, (April, 1981).

[10] Validating the Correctness of Hardware Implementations of the NBS Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-20.

[11] Maintenance Testing for the Data Encryption Standard, National Institute of Standards and Technology. Special Publication 500-61, (August, 1980).

[12] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S., Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, January 1996.

[13] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1906, January 1996.

[14] Krawczyk, H., "Keyed-MD5 for Message Authentication", Work in Progress, IBM, June 1995.

[15] The SNMPv2 Working Group, Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1907 January 1996.

APPENDIX A - Installation

A.1. Agent Installation Parameters

During installation, an agent is configured with several parameters. These include:

(1) a security posture

The choice of security posture determines the extent of the view configured for unauthenticated access. One of three possible choices is selected:

minimum-secure,
semi-secure, or
very-secure.

(2) one or more transport service addresses

These parameters may be specified explicitly, or they may be specified implicitly as the same set of network-layer addresses configured for other uses by the device together with the well-known transport-layer "port" information for the appropriate transport domain [13]. The agent listens on each of these transport service addresses for messages sent on behalf of any user it knows about.

(3) one or more secrets

These are the authentication/privacy secrets for the first user to be configured.

One way to accomplish this is to have the installer enter a "password" for each required secret. The password is then algorithmically converted into the required secret by:

- forming a string of length 1,048,576 octets by repeating the value of the password as often as necessary, truncating accordingly, and using the resulting string as the input to the MD5 algorithm. The resulting digest, termed "digest1", is used in the next step.

- a second string of length 44 octets is formed by concatenating digest1, the agent’s agentID value, and digest1. This string is used as input to the MD5 algorithm. The resulting digest is the required secret (see Appendix A.2).

With these configured parameters, the agent instantiates the following user, context, views and access rights. This configuration information should be readOnly (persistent).

- One user:

	privacy not supported	privacy supported
<userName>	"public"	"public"
<authProtocol>	Digest Auth. Protocol	Digest Auth. Protocol
<authPrivateKey>	authentication key	authentication key
<privProtocol>	none	Symmetric Privacy Protocol
<privPrivateKey>	--	privacy key

- One local context with its <contextSelector> as the empty-string.

- One view for authenticated access:

    - the <all> MIB view is the "internet" subtree.

- A second view for unauthenticated access. This view is configured according to the selected security posture. For the "very-secure" posture:

    - the <restricted> MIB view is the union of the "snmp" [15], "usecAgent" and "usecStats" subtrees.

  For the "semi-secure" posture:

    - the <restricted> MIB view is the union of the "snmp" [15], "usecAgent", "usecStats" and "system" subtrees.

  For the "minimum-secure" posture:

    - the <restricted> MIB view is the "internet" subtree.

- Access rights to allow:

    - read-only access for unauthenticated messages on behalf of the user "public" to the <restricted> MIB view of contextSelector "".

    - read-write access for authenticated but not private messages on behalf of the user "public" to the <all> MIB view of contextSelector "".

    - if privacy is supported, read-write access for authenticated and private messages on behalf of the user "public" to the

<all> MIB view of contextSelector "".

A.2. Password to Key Algorithm

The following code fragment demonstrates the password to key algorithm which can be used when mapping a password to an authentication or privacy key. (The calls to MD5 are as documented in RFC 1321.)

```
void password_to_key(password, passwordlen, agentID, key)
    u_char *password;       /* IN */
    u_int   passwordlen;    /* IN */
    u_char *agentID;        /* IN - pointer to 12 octet long agentID */
    u_char *key;            /* OUT - caller supplies pointer to 16
                               octet buffer */ {
    MD5_CTX     MD;
    u_char      *cp, password_buf[64];
    u_long      password_index = 0;
    u_long      count = 0, i;

    MD5Init (&MD);   /* initialize MD5 */

    /* loop until we’ve done 1 Megabyte */
    while (count < 1048576) {
        cp = password_buf;
        for(i = 0; i < 64; i++) {
            *cp++ = password[ password_index++ % passwordlen ];
            /*
             * Take the next byte of the password, wrapping to the
             * beginning of the password as necessary.
             */
        }
        MDupdate (&MD, password_buf, 64);
        count += 64;
    }
    MD5Final (key, &MD);              /* tell MD5 we’re done */

    /* localize the key with the agentID and pass through MD5
      to produce final key */
    memcpy (password_buf, key, 16);
    memcpy (password_buf+16, agentID, 12);
    memcpy (password_buf+28, key, 16);

    MD5Init (&MD);
    MDupdate (&MD, password_buf, 44);
    MD5Final (key, &MD);

    return; }
```

A.3. Password to Key Sample

The following shows a sample output of the password to key algorithm.

With a password of "maplesyrup" the output of the password to key algorithm before the key is localized with the agent’s agentID is:

```
 ’9f af 32 83 88 4e 92 83 4e bc 98 47 d8 ed d9 63’H
```

After the intermediate key (shown above) is localized with the agentID value of:

```
 ’00 00 00 00 00 00 00 00 00 00 00 02’H
```

the final output of the password to key algorithm is:

```
 ’52 6f 5e ed 9f cc e2 6f 89 64 c2 93 07 87 d8 2b’H
```