SNMP Security

- Perhaps the main objective of SNMPv3 was the addition of security for SNMP management.
  - Authentication and privacy of information; authorization and access control.
- The architecture supports any type of security but IETF SNMPv3 working group has specified a user-based security model.
  - Follows typical model of a user (user name -> security information).
Security Threats

- **Modification** of information: Contents modified by unauthorized user, does not include address change. This is possible because of UDP transport protocol.
- **Masquerade**: change of originating address by an unauthorized user.
- **Message stream modification**: fragments of message altered by an unauthorized user to modify the meaning of the message.
- **Disclosure**: tapping and eavesdropping.
- **Denial of service** and **traffic analysis** are not considered as threats in SNMP model.

Security Services

- **Authentication**
  - Data integrity:
    - HMAC-MD5-96
    - HMAC-SHA-96
  - Data origin authentication
    - Append to the message a unique Identifier associated with the authoritative SNMP engine.
- **Privacy**
  - Encryption is used
    - Cipher Block Chaining mode of the Data Encryption Standard (DES) is suggested.
- **Timeliness**
  - Used to prevent message redirection, delay, and replay.
  - Authoritative Engine ID, No. of engine boots and time in seconds (recommended window time of 150 s).
Role of SNMP Engines

- SNMP defines the concept of **Responsibility of Authoritative engine**:
  - For `get-request`, `get-next-request`, `get-bulk-request`, `set-request`, or `inform` the receiver is the authoritative engine.
  - For `trap`, `get-response` and `report` the sender is the authoritative engine.
  - Responsible for:
    - Unique SNMP engine ID
    - Time-stamp
- Non-authoritative engine should keep a table of the time-stamp and authoritative engine ID

**Figure 7.12 SNMPv3 Message Format**

<table>
<thead>
<tr>
<th>Header Data</th>
<th>scopedPDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Message ID</td>
<td>Context Engine ID</td>
</tr>
<tr>
<td>Max. Size</td>
<td>Context Name</td>
</tr>
<tr>
<td>Message Flag</td>
<td>Data</td>
</tr>
<tr>
<td>Message Security Model</td>
<td>Security Parameters</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Global/ Header Data</th>
<th>Security Parameters</th>
<th>Plaintext / Encrypted scopedPDU Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>Authority Engine ID</td>
<td>Authoritative Engine Boots</td>
</tr>
<tr>
<td></td>
<td>Authority Time</td>
<td>User Name</td>
</tr>
<tr>
<td></td>
<td>Authentication Parameters</td>
<td>Privacy Parameters</td>
</tr>
</tbody>
</table>

**Figure 7.12 SNMPv3 Message Format**
### Message Format Description

<table>
<thead>
<tr>
<th>Field</th>
<th>Object name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>msgVersion</td>
<td>SNMP version number of the message format</td>
</tr>
<tr>
<td>Message ID</td>
<td>msgID</td>
<td>Administrative ID associated with the message</td>
</tr>
<tr>
<td>Message Max. Size</td>
<td>msgMaxSize</td>
<td>Maximum size supported by the sender</td>
</tr>
<tr>
<td>Message flags</td>
<td>msgFlags</td>
<td>Bit fields identifying report, authentication, and privacy of the message</td>
</tr>
<tr>
<td>Message Security Model</td>
<td>msgSecurityModel</td>
<td>Security model used for the message; concurrent multiple models allowed</td>
</tr>
<tr>
<td>Security Parameters (See Table 7.8)</td>
<td>msgSecurityParameters</td>
<td>Security parameters used for communication between sending and receiving security modules</td>
</tr>
<tr>
<td>Plaintext/Encrypted scopedPDU Data</td>
<td>scopedPduData</td>
<td>Choice of plaintext or encrypted scopedPDU; scopedPDU uniquely identifies context and PDU</td>
</tr>
<tr>
<td>Context Engine ID</td>
<td>contextEngineID</td>
<td>Unique ID of a context (managed entity) with a context name realized by an SNMP entity</td>
</tr>
<tr>
<td>Context Name</td>
<td>contextName</td>
<td>Name of the context (managed entity)</td>
</tr>
<tr>
<td>PDU</td>
<td>data</td>
<td>Contains unencrypted PDU</td>
</tr>
</tbody>
</table>

### User-Based Security Model

- Based on traditional user name concept
- USM uses **security primitives** across the *abstract service interfaces* similar to the SNMP entity concept.
  - Authentication service primitives
    - authenticateOutgoingMsg
    - authenticateIncomingMsg
  - Privacy Services
    - encryptData
    - decryptData
- Based on shared secret keys
Secure Outgoing Message

- USM invokes privacy module with encryption key and scopedPDU
- Privacy module returns privacy parameters (for DES encryption) and encrypted scopedPDU
- USM then invokes the authentication module w/ authentication key and whole message and receives authenticated whole message

Secure Incoming Message

- Processing secure incoming message
- Authentication validation done first by the authentication module
- Decryption of the message done then by the privacy module
Security Parameters

The usmUserTable lists the users, authentication, and privacy parameters for the snmpEngine.

<table>
<thead>
<tr>
<th>Security Parameters</th>
<th>USM User Group Objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>msgAuthoritativeEngineID</td>
<td>snmpEngineID (under snmpEngine Group)</td>
</tr>
<tr>
<td>msgAuthoritativeEngineBoots</td>
<td>snmpEngineBoots (under snmpEngine Group)</td>
</tr>
<tr>
<td>msgAuthoritativeEngineTime</td>
<td>snmpEngineTime (under snmpEngine Group)</td>
</tr>
<tr>
<td>msgUserName</td>
<td>usmUserName (in usmUserTable)</td>
</tr>
<tr>
<td>msgAuthenticationParameters</td>
<td>usmUserAuthProtocol (in usmUserTable)</td>
</tr>
<tr>
<td>msgPrivacyParameters</td>
<td>usmUserPrivProtocol (in usmUserTable)</td>
</tr>
</tbody>
</table>

Table 7.8 Security Parameters and Corresponding MIB Objects

Authentication Key

- **Secret key** for authentication
- Derived from the user’s (NMS) password
- MD5 or SHA-1 algorithm used to generate privKey
  - used also for privacy CBC-DES algorithm
- The Authentication key is the second digest digest2

**Procedure:**

1. Derive digest0:
   - Password repeated until it forms $2^{20}$ octets.
2. Derive digest1: (privKey)
   - Hash digest0 using MD5 or SHA-1.
3. Derive digest2:
   - Concatenate privKey, SNMP engine ID, and privKey and hash with the same algorithm
Authentication Parameter

- Authentication parameter is the Hashed Message Access Code (HMAC)
- HMAC is 96-bit long (12 octets)
- Derived from authorization key (authKey)

Procedure:
1. Derive \textit{extendedAuthKey}:
   Supplement authKey with 0s to get 64-byte string
2. Define \textit{ipad}, \textit{opad}, K1, and K2:
   - \textit{ipad} = 0x36 (00110110) repeated 64 times
   - \textit{opad} = 0x5c (01011100) repeated 64 times
   - K1 = \textit{extendedAuthKey} XOR \textit{ipad}
   - K2 = \textit{extendedAuthKey} XOR \textit{opad}
3. Derive HMAC by hashing algorithm used
   HMAC = H (K2, H (K1, wholeMsg))

Key Management

- An NMS has \textbf{only one password} hence one secret key \textit{digest1}, but communicates to several authoritative SNMP engines (i.e. agents).
- It should share a secret with each authoritative SNMP engine; this results in \textit{n} passwords.
- In order to use one password a localized key is introduced using a hash function that includes the \textit{authoritativeSnmpEngineID} and a secret key (\textit{digest1}).
  - This localized key is different for each agent.
Privacy Module

- For data integrity, message authentication, data confidentiality, and timeliness of data.
- Encryption and decryption of scoped PDU
  - context engine ID, context name, and PDU
- CBC - DES (Cipher Block Chaining – Data Encryption Standard)
- Encryption key (and initialization vector) made up of secret key and timeliness value. Secret is associated with NMS (user).
- Privacy parameters is the salt value (unique for each packet) in CBC-DES to ensure that 2 identical instances of ciphertext are not generated by an SNMP Engine.
  - salt = concatenate (snmpEngineBoots & Random Integer)

Encryption Protocol

- Cipher Block Chaining mode of Data Encryption Standard (CBC-DES) protocol
- 16-octet privKey is secret key
- First 8-octets of privKey used for the 56-bit DES key
  - Only 7 high-order bits of each octet used
- Last 8-octets of privKey used as pre-initialization vector

Figure 13.33 Basic Cryptographic Communication
Access Control

- **View-based Access Control Model**
  - Defines a set of services that an application in an agent can use to validate command requests and notification receivers.
  - It assumes that users have already been authenticated.
  - Defines five elements that comprise the VACM:
    - **Groups**: Group comprising security model and security name
      - In SNMPv1, is community name
    - **Security Level**: mapping of security to access rights.
    - **Contexts**: mapping of names of the context to access rights
    - **MIB Views** and **View Families** (As in SNMPv1 & v2)
      - MIB view is a combination of view subtrees
    - **Access Policy (viewType)**
      - read-view
      - write-view
      - notify-view
      - not-accessible

VCAM Process

- Tries to answers 6 questions:
  1. **Who** are you (group)?
  2. **Where** do you want to go (context)?
  3. **How secured** are you to access the information (security model and security level)?
  4. **Why** do you want to access the information (read, write, or send notification)?
  5. **What object type** do you want to access?
  6. **Which object instance** do you want to access?
**VACM MIB**

- **Four tables** used to achieve access control
  - **Group** defined by security-to-group table
  - **Context** defined by context table
  - **Access** determines access allowed and the view name
  - **View** tree family table determines the MIB view, which is very flexible

**Figure 7.17 VACM MIB**

**MIB Views**

- Allows for the selection of particular views of an object based on a mask.
- **Simple view:**
  - system 1.3.6.1.2.1.1
- **Complex view:**
  - All information relevant to a particular interface, system and interfaces groups
- **Family view subtrees**
  - View with all columnar objects in a row appear as separate subtree.

**Example:**

<table>
<thead>
<tr>
<th>Family view name</th>
<th>Family subtree</th>
<th>Family mask</th>
<th>Family type</th>
</tr>
</thead>
<tbody>
<tr>
<td>system</td>
<td>1.3.6.1.2.1.1</td>
<td>&quot;&quot;</td>
<td>1</td>
</tr>
</tbody>
</table>

**Example:**

<table>
<thead>
<tr>
<th>FAMILY IDENTIFIER (family name)</th>
<th>paired with bit-string value (family mask) to select or suppress columnar objects</th>
</tr>
</thead>
<tbody>
<tr>
<td>Family view name = “system”</td>
<td>Family subtree = 1.3.6.1.2.1.1 Family mask = &quot;&quot; (implies all 1s by convention)</td>
</tr>
<tr>
<td>Family type = 1 (implies value to be included)</td>
<td></td>
</tr>
</tbody>
</table>
contextName has entry?

Any group assigned to this <securityModel, securityName>?

Any entry for groupName, contextName, securityModel, securityLevel?

If so, does it refer to a MIB view of viewType?

Use viewName from previous step as index and see if a MIB view is found.

Is the variablename included in the view?