SNMPv3 Deployment
Best Current Practices

SNMP Research International, Inc.
Knoxville, Tennessee

Date: 12Feb2018

SNMP is a powerful tool for management, but if used improperly, it can open a doorway into your network through which malicious intruders can gain a foothold. Networks, devices, and applications that are managed by SNMPv1, SNMPv2c, or a poorly-configured SNMPv3 agent are vulnerable to attack.

The purpose of this white paper is to provide information about deploying SNMPv3 to people who are involved in this task at a variety of levels. Network planners, network administrators, data center operators, and equipment installers can all benefit from understanding the information contained in this document.

SNMPv3 is the latest version of the Internet Standard Simple Network Management Protocol used to monitor and control networks, devices, and applications. Monitoring is performed by sending "read" commands (**Get**, **GetNext**, **GetBulk**) from an SNMP manager and receiving a response from an SNMP agent. Control is performed by sending "write" commands (**Set**) containing new settings that are applied to the network, device, or application by the SNMP agent. Unsolicited notifications (**Trap**, **Inform**) are sent by SNMP agents to SNMP managers to convey information about an alarm condition in the network, device, or application. Earlier versions of SNMP (SNMPv1, SNMPv2c) could perform many of the same functions but with only trivial considerations for security.

Secure network management is the primary motivation for fully deploying SNMPv3 in an enterprise. Many devices including UNIX servers and network infrastructure equipment include support for SNMPv3 from the factory. Other devices such as Microsoft Windows servers and workstations have no SNMP support at all or come with an SNMPv1/SNMPv2c agent that may or may not be enabled. For these devices, SNMPv3 solutions are available from third parties including SNMP Research. Contact SNMP Research at `snmpv3@snmp.com` for any SNMPv3 needs that you may have.

The process of deploying SNMPv3 requires a knowledge of the "big picture" of your network. Upgrades to network equipment to add SNMPv3 should be prioritized for the edge devices first and work inward. An inventory of all network assets should be performed so you know which support SNMPv3, which do not, and which may need additional functionality such as modern encryption capabilities. Section 2 describes some things to look for when network equipment is evaluated for its suitability as part an SNMPv3 deployment. Consideration should be given to automating periodic updates to SNMPv3 keys (USM user pass phrases) that allow authorized users to access devices. At several points during the deployment

process, the network should be evaluated by a tool that scans for vulnerabilities and recommends changes to increase security.

This document uses SNMPv3 terminology such as command responder, notification originator, proxy forwarder, context name, context engineID, authentication protocol, and privacy protocol. Readers who are already familiar with these terms and understand their differences can skip ahead to Section 2.

1.1 SNMP Applications

Readers should be familiar with the terms SNMP manager and SNMP agent and understand their differences. The SNMPv3 specification describes agents and managers this way:

SNMP Agent - This is an SNMP entity typically containing a command responder application and a notification originator application.

SNMP Manager - This is an SNMP entity typically containing a command generator application and a notification receiver application.

Many conversations about SNMP include the word "agent" even if it does not involve an SNMP entity acting in the agent role. Likewise, many conversation about SNMP include the word "proxy" even if it does not involve an SNMP entity that contains a proxy forwarder application. It is important to use SNMP terms properly.

To be clear:

- A command generator sends SNMP **Get**, **GetNext**, **GetBulk**, and **Set** messages. It expects to receive an SNMP **Response** or **Report** message in reply.
- A command responder receives SNMP **Get**, **GetNext**, **GetBulk**, and **Set** messages and may send back an SNMP **Response** or **Report** message in reply.
- A notification originator sends SNMP **Trap** messages and expects to receive no reply. A notification originator may also send SNMP **Inform** messages and expects to receive an SNMP **Response** or **Report** message in reply.
- A notification receiver expects to receive SNMP **Trap** and **Inform** messages. It may send back an SNMP **Response** or **Report** message in reply to an **Inform**.
- A proxy forwarder may expect to receive all types of SNMP messages for the purpose of forwarding them to another SNMP entity that is the intended destination (or closer to the intended destination).

Some vendors use non-standard terms like "intelligent agent" that imply behavior beyond the normal definition of an SNMP agent. For example, an intelligent agent might present MIB objects representing an entire LAN with values aggregated from data it collected from individual devices in the LAN. To present the LAN-level MIB objects to an SNMP manager, the agent contains an command responder application that is normal for any SNMP agent to respond to **Get** requests from above. To collect the individual-level MIB objects from within the LAN, it also contains a command generator application to send **Get** requests to other devices, which is not typical behavior for an SNMP agent. To be precise, network administrators and all SNMP operators should consistently use the standard terms.

1.2 SNMP Multiplexers

Network managers are familiar with the concept that sending the same SNMP request to the same SNMPv1 or SNMPv2c agent but with a different community string may elicit a different result. SNMPv3 formalized some of these common multiplexing behaviors and provides vocabulary to identify them with precision.

1.2.1 Virtual Agents

Consider an SNMP **Get** request for the MIB object **sysObjectID.0**. Send it to an SNMP agent as an SNMPv1 or SNMPv2c message with one community string, and the **Response** message returns a value indicating that the device is a particular model of network router. Send it to the same SNMP agent with a different community string, and the **Response** message returns a value indicating that the device is a type of uninterruptible power supply (UPS).

This is an example of virtual agents. The behavior is how one would imagine if it were possible to have two SNMP agents listening on the same IP address and port, and the community string indicates which of the two agents should respond to the request; this is called *context switching*. The community string was not created for the purpose of context switching, but over time some SNMP agent vendors adopted this overloaded use for the community string as a convention.

SNMPv3 separates context switching information from the authentication information. The header of every SNMPv3 message contains a dedicated field for specifying the context, called “contextName”. When the **Get** request is sent as an SNMPv3 message, the user name for authentication can be one string (“joe”) and the contextName can be another string (“UPS”).

1.2.2 Proxy Forwarders

Consider an SNMP entity acting as a gateway onto a NAT-ted LAN. To route a **Get** request in an incoming SNMPv1 or SNMPv2c message to the correct device within the LAN, the community string acts as the discriminator. If the community string in the received SNMP message is “device1”, for example, the message may be forwarded to the address **192.168.1.1**. Similarly, if the community string in the received SNMP message is “device2”, the message may be forwarded to the address **192.168.1.2**.

This is an example of proxy forwarding that pre-dates the SNMPv3 specification. The behavior is how one would imagine if it were possible to have two SNMP agents listening on the same IP address and port, and the community string indicates which of the devices on the LAN should receive the **Get** request (SNMP message routing). The community string was not created for the purpose of SNMP message routing, but over time some SNMP agent vendors adopted this overloaded use for the community string as a convention.

SNMPv3 separates message routing information from the authentication information. The header of every SNMPv3 message contains a dedicated field for specifying the SNMP entity that is the intended final destination, called “contextSnmpEngineID”. When the **Get** request is sent as an SNMPv3 message, the user name for authentication can be one string (“joe”) and the contextSnmpEngineID can be another string.

1.3 SNMP Administration

1.3.1 Background

The original purpose for the community string was to allow only authorized operators to access an SNMP agent. Those people with the proper permission to access an SNMP agent were considered to be members of a community, and the community string was the "secret word" for entry into a protected area. If an agent received an SNMPv1 or SNMPv2c message containing a **Get** request, the agent should respond with the requested information *only* if the message contained the correct community string. It was assumed that only members of the community would ever know the correct community string.

Today, the proper term is "authentication". Authentication is proving with reasonable certainty that the sender of an SNMP message is who they claim to be. If the user "joe" has the proper permission to view the data served by an SNMPv3 agent, and if an agent receives an SNMPv3 message containing a **Get** request claiming to be from joe, the agent should respond with the requested information *only* if it can determine with reasonable certainty that the message actually came from joe. If the message came from someone pretending to be joe, it should be discarded.

The problem with SNMPv1 and SNMPv2c community strings is that there was no real way to determine with reasonable certainty that a message containing the correct community string was actually sent by a member of the community. It was relatively easy for an intruder to obtain the community string by capturing SNMP messages from the network, because the community string was transmitted "in the clear" in every SNMPv1 and SNMPv2c message. Thus, an intruder could pretend to be a member of the community (masquerade) and use the community string to gain access to all of the same information in an SNMP agent and have all of the same privileges as any legitimate member of the community. Neither SNMPv1 nor SNMPv2c could truly authenticate the sender of an SNMP message. Therefore, the community string approach is called "trivial authentication" today.

SNMPv3 provides strong security for network management that qualifies it to be used by commercial enterprises and governments for communication, commerce, and defense. There is a strong authentication mechanism, and messages can be encrypted.

One of the arguments against the deployment of SNMPv3 has been that the security is difficult to administer. However, this is not a specific complaint about SNMP. *Any* system having *any* form of security is always more difficult to use than if the system had no security.

1.3.2 Designed From the Ground Up

As network administrators and SNMP architects grappled with the limitations of community strings, some thought was given to the possibility of increasing the protection of SNMP agents by periodically changing the community string. There were two problems with this proposal:

1. Since community strings are transmitted in the clear in the header of every SNMPv1 and SNMPv2c message, then after a community string is changed, an intruder need only capture one message to learn what is the new community string and begin using it to masquerade.
2. The mechanism for SNMP to make any change in a remote agent (including a change of community string) is a **Set** request. However, since the payload of an SNMPv1 and SNMPv2c **Set** request is not

encrypted, an intruder could actually learn about the change of community string before it even happened by reading the new string out of the **Set** request.

As SNMPv3 was being designed, these problems were taken into consideration, and the solutions were built into the SNMPv3 protocol at the very beginning:

1. Neither the authentication pass phrase nor the authentication key for an SNMPv3 user is ever transmitted in the header of an SNMPv3 message. Instead, a hash is performed over an SNMPv3 message using a one-way algorithm before it is transmitted, and the header carries the digest value.
2. When a **Set** request is used to change the pass phrase(s) of an existing SNMPv3 user or to create a new SNMPv3 user, neither the authentication pass phrase or key nor the privacy pass phrase or key are carried in the payload of the message. Instead, a `KeyChange` transformation is performed using a two-way (reversible) algorithm with a secret factor that is never transmitted on the wire.

Because of these considerations, SNMPv3 **Set** requests can be used to safely carry out periodic changes to the security configurations in remote SNMPv3 agents. An entire framework was created to administer the authentication and access control features in SNMPv3 entities (Section 1.3.3).

It is common today that the pass phrases for SNMPv3 users are not changed as often as they should be, but periodic changes are still the best current practice available to protect the integrity of a managed system. As a rule of thumb, pass phrases for SNMPv3 users should be changed every time the enterprise requires the changing of pass phrases for logins to UNIX and Microsoft Windows servers.

Brute force attacks are always possible, and computing power to crack mathematical algorithms is ever-increasing. However, the underlying mechanisms that protect SNMPv3 are designed to outpace the progress of would-be intruders. The algorithms available to secure SNMPv3 messages are designed to make it computationally infeasible to crack a message more quickly than an enterprise to exercise reasonable policies for changing pass phrases for all users.

1.3.3 The SNMPv3 Administration MIBs

The culmination of the efforts of SNMP and security architects is the SNMPv3 Administration Framework and the SNMPv3 Administration MIBs. These are defined by the standards documents RFC 3411-3416.

The SNMPv3 Administration Framework originally included a specification for the coexistence between SNMPv3 and the older SNMPv1 and SNMPv2c protocols. The specification, defined in the `SNMP-COMMUNITY-MIB`, maps community strings onto SNMPv3 contexts and connects them to the view-based access control (VACM) mechanisms created for SNMPv3. The RFC that contains this part of the specification, RFC 2576, republished as RFC 3584, was not allowed to advance forward in the standardization process because it referred to the older protocols that were being reclassified as Historic. However, many vendors rightly implement it (in whole or in part) as part of the SNMPv3 implementation.

Here is a list of the MIB tables defined within the SNMPv3 Administration MIBs, including the `SNMP-COMMUNITY-MIB`:

- **usmUserTable**
- **vacmSecurityToGroupTable**
- **vacmAccessTable**

- **vacmViewTreeFamilyTable**
- **snmpNotifyTable**
- **snmpTargetAddrTable**
- **snmpTargetParamsTable**
- **snmpNotifyFilterProfileTable**
- **snmpNotifyFilterTable**
- **snmpProxyTable**
- **snmpCommunityTable**

Most of the MIB objects in these tables have a `MAX-ACCESS` property of `read-create`, which means that they are intended to be modified by SNMP **Set** requests to create new rows, delete existing rows, and modify existing rows. The security of the SNMPv3 protocol can be effectively managed by applications (Section 7) when the device agents fully support the MIB tables for both read and write access.

2 Differences in SNMPv3 Products

Commercial products that support SNMPv3 contain program code that is independently created or derived from one of several sources. Therefore, not all SNMPv3 products contain the same set of features or implement the same parts of the SNMPv3 specification. When selecting SNMPv3 products, it is important to understand the differences that may be important to your deployment. It should be a goal to maximize compatibility between components supplied by different vendors.

2.1 What Are the Authentication Protocols Supported?

When the SNMPv3 specification was originally published, the documents identified only one authentication protocol by name. That protocol is called Message Digest Algorithm 5, or MD5. By today's standards, this 128-bit algorithm is too weak to protect valuable network assets against brute force attack. Using MD5 can result in a false sense of security.

The strongest authentication algorithm in common use with SNMPv3 is the Secure Hash Algorithm, or SHA-1. This 160-bit algorithm provides a better hashing capability than MD5. New manageable devices should support an SNMPv3 agent with SHA-1 as a minimum requirement.

SHA-2 is a much stronger authentication algorithm with 224-bit, 256-bit, 384-bit, and 512-bit modes. It represents the future of SNMPv3 security but is not commonly deployed as of the date of composition of this document.

When selecting an SNMP manager to monitor and control SNMPv3 devices, the network administrator should choose a software product that is capable of forming SNMPv3 messages using as many of these authentication algorithms and modes as possible. The more algorithms and modes are supported, the greater the degree of compatibility the SNMP manager will have in a heterogeneous network consisting of manageable devices from a variety of vendors. Also, if potential intruders become sophisticated enough to

break the form of authentication in use, having support for more algorithms and modes means that the network administrator has greater flexibility to switch to a different one that has not been broken.

2.2 What Are the Privacy Protocols Supported?

When the SNMPv3 specification was originally published, the documents identified only one privacy protocol by name. That protocol is called Data Encryption Standard, or DES. By today's standards, this 56-bit algorithm is too weak to protect valuable network assets against brute force attack. Using DES can result in a false sense of security.

A stronger privacy protocol is available at the present time and in common use with SNMPv3. It is called the Advanced Encryption Standard, or AES. All devices that can use AES for SNMPv3 support 128-bit encryption. Some devices also support AES in 192-bit and 256-bit modes.

Another strong privacy protocol for SNMPv3 is available but not as common called Triple-DES, or 3DES. This privacy protocol executes the original DES algorithm three times resulting in 168-bit encryption that is much stronger than the original.

When selecting new manageable devices and an SNMP manager to monitor and control SNMPv3 devices, the network administrator should choose products that are capable of forming SNMPv3 messages using as many of these privacy protocols and encryption modes as possible. The more algorithms and modes are supported, the greater the degree of compatibility the SNMP manager will have in a heterogeneous network consisting of manageable devices from a variety of vendors. Also, if potential intruders become sophisticated enough to break the form of encryption in use, having support for more algorithms and modes means that the network administrator has greater flexibility to switch to a different one that has not been broken.

2.3 What Are the Notification Types Supported?

SNMPv1 offered only one type of SNMP message for alarms, the **Trap**. A single **Trap** message sent by an agent may not reach the manager as a result of normal and expected packet loss in the network; e.g., from wireless links.

SNMPv2c offered a new type of SNMP message for alarms in addition to the **Trap**. This is called an **Inform**. When a notification originator sends an **Inform**, it expects to receive a **Response** in reply. If it does not, the agent may be programmed to retransmit the notification until it is acknowledged.

SNMPv3 supports both **Trap** and **Inform** messages. Through the SNMPv3 Administration Framework, it is easy to configure an SNMPv3 agent to retransmit an **Inform** message to one or more managers until the message is acknowledged or until a retransmission policy is exceeded.

When selecting devices to deploy in a network, the network administrator should choose those in which the SNMP agent is capable of forming both **Trap** and **Inform** messages. Also, the device should support the SNMPv3 Administration Framework so that the retransmission policy can be configured according to the needs of the enterprise.

When selecting an SNMP manager to monitor and control SNMPv3 devices, the network administrator

should choose a software product that is capable of receiving both **Trap** and **Inform** messages and sending back a proper **Response** message in reply to an **Inform**. Also, the manager should offer controls to the operator for rejecting unsecure (SNMPv1 and SNMPv2c) notifications.

2.4 Are Authenticated and Encrypted Notifications Supported?

When a device experiences an alarm condition, the SNMPv3 notification message that carries this information has the option to be authenticated and encrypted.

Alarms *should* be authenticated (Section 5.1) so an intruder is not able to send false information to the network management system as part of an attack. Misdirecting investigators away from the targeted part of the network gives the intruder valuable time to complete an action and cover his tracks. By ensuring that alarms are received in a timely manner (not replayed) and that they were sent by a recognized (and authorized) sender helps to prevent misdirections.

Attackers can also capture notification messages "in flight." Alarms *should* also be encrypted (Section 5.2) so an intruder is not able to use the information carried in the message. If messages are not encrypted, the attacker may be able to see state changes occurring in the network faster than the network operators. Notifications intended for managers could inform an attacker when a device is most vulnerable or when a window of opportunity to gain a foothold is open. By encrypting the content of SNMPv3 **Trap**s and **Inform**s, this drastically reduces the chance that an attacker could make sure of the information before network operators are able to correct the alarm condition.

When selecting an SNMP manager to monitor and control SNMPv3 devices, the network administrator should choose a software product that is capable of authenticating and decrypting SNMPv3 **Trap** and **Inform** messages. Also, the SNMP manager should offer controls to the operator for rejecting unsecure (noAuthNoPriv SNMPv3) notifications.

2.5 How Are the Administration Framework and MIBs Supported?

When selecting SNMPv3 devices to deploy in a network, the network administrator should require full support for the SNMPv3 Administration Framework and the SNMPv3 Administration MIBs (Section 1.3.3). It is important to note that vendors can claim to support the RFCs that define these standards and yet not offer an implementation that is capable of effective administration. Here are some important details to investigate about any manageable device containing an SNMPv3 agent:

- Does the implementation support **Set** requests to allow the configuration of SNMPv3 users, groups, access controls, and notifications? The agent should allow changes.
- Does the implementation save information in the SNMPv3 Administration Framework to non-volatile storage? SNMPv1 and SNMPv2c configurations were typically stored in read-only memory (ROM). An SNMPv3 implementation is not compliant to the specification if changes can not be saved across reboots.
- Does the value of **snmpEngineBoots** advance correctly? If the device is restarted, the value of **snmpEngineBoots** must increase, and under no circumstances can it retreat to an earlier (smaller) value.

- Does the value of **snmpEngineTime** advance correctly? It should continuously increase with the passage of time. If it returns to zero for any reason (wrap, device restart, etc.), then **snmpEngineBoots** must increment at the same time.
- How is the value of **snmpEngineID** determined? A device must allow the administrator to assign a new value at any time, but the initial value is also important:
  - The best implementation will generate a unique value automatically from the factory. This value will not change if a hardware component (such as a network card) is replaced or if a different IP address is assigned to the device by the administrator.
  - The value should be unique when more than one instance of the SNMP agent is running. One strategy for this is to include the UDP port number for incoming SNMP messages somewhere in the bytes of the **snmpEngineID** value.
  - If the **snmpEngineID** value is not generated automatically, an acceptable alternative is to prompt for the value when the device is powered on for the first time. If a value is not specified at the prompt, the SNMPv3 agent should not be allowed to start.
  - A flawed product will allow the SNMPv3 agent to start with a preconfigured, non-unique value. This introduces a vulnerability into the configuration that can normally be remedied only by wiping the contents of the **usmUserTable** in the Administration MIBs and reentering the information from scratch.

Scanning applications are available to test for some implementation errors (Section 7). Devices that contain errors should be rejected from consideration.

2.6 How Are Multiple SNMP Versions Supported?

As of the composition date of this document, agents that support SNMPv3 nearly always support SNMPv1 and SNMPv2c simultaneously. As time passes, devices that support SNMPv3 only and no longer support SNMPv1 or SNMPv2c should become common.

In any security-conscious environment, the use of insecure protocols should be highly restricted:

1. The command responder applications in SNMP agents should reject **Set** requests if the message type is SNMPv1, SNMPv2c, or SNMPv3 noAuthNoPriv.
2. If the message type is SNMPv1, SNMPv2c, or SNMPv3 noAuthNoPriv, the command responder applications in SNMP agents should return data only from a narrow view of the device, such as the **system** and **snmpEngine** branches of the MIB; other data should not be available.
3. The notification receiver applications in SNMP managers should discard **Trap** and **Inform** messages if the message type is SNMPv1, SNMPv2c, or SNMPv3 noAuthNoPriv.

Most agents should be able to handle item 1 above. This feature has been common in agents for many years.

To implement item 2, the agent will need to support RFC 2576 or RFC 3584 in addition to RFC 3411-3416. This defines the specification for the coexistence between SNMPv1, SNMPv2c, and SNMPv3 protocols and configuration. This allows SNMPv3-style access controls to be applied to community strings.

As an alternative to item 2, the agent could be configurable to reject all attempts to access MIB data if the message type is SNMPv1, SNMPv2c, or SNMPv3 noAuthNoPriv.

One other important test for an agent is this. Can it function correctly if the SNMPv1 and SNMPv2c protocols are completely disabled? Today, this is usually accomplished by configuring an empty list of community strings.

An SNMP manager that blindly accepts **Trap** and **Inform** messages without authenticating the sender is vulnerable to tactics in which the intruder sends false information to a notification receiver to misdirect investigation tools and personnel away from the focus of his attack. When selecting an SNMP manager to monitor and control devices, the network administrator should choose a software product that is capable of filtering the receipt of notifications (item 3), at least as an option.

2.7 Does the SNMP Code Contain Known Vulnerabilities?

Over the years, there have been a few significant CERT advisories directing network administrators to upgrade devices for flaws that affected the SNMP entities derived from a particular code source or more broadly that affected all code sources. In addition, there are reports of devices produced by various manufacturers that contain flaws in the SNMP agent that make the agent less secure.

Every manageable device that is under consideration to be deployed in a network should be analyzed by a software tool that searches for vulnerabilities in the SNMP agent. Section 8 discusses this in more detail.

2.8 Does the SNMP Manager Perform Engine Discovery Securely?

When an SNMPv3 manager communicates with an SNMPv3 agent for the first time, it performs a handshake called *SNMP Engine Discovery*. This allows the manager to obtain from the agent its values for **snmpEngineID**, **snmpEngineBoots**, and **snmpEngineTime** that are needed for secure communications. When selecting an SNMP manager, the software should be checked to determine if it performs this handshake correctly as a two-stage process. Some SNMP manager products perform only a single-stage discovery process, which is not secure.

The first discovery message sent by the SNMPv3 manager is sent with security level noAuth (no authentication and no privacy). The first **Report** PDU returned by the SNMPv3 agent is sent with security level noAuth. There is no guarantee that the manager is communicating with a legitimate SNMPv3 agent. It could be an unauthorized SNMPv3 agent put in place by a hacker (who does not know the USM User's authentication key). If you always intend to communicate with the device using no security, then you don't care about this possibility. If your manager software accepts the first **Report** PDU as valid with no authentication, then it isn't programmed to defend against this possibility. For either of these cases, the software could accept the values of **snmpEngineBoots** and **snmpEngineTime** from the first stage of discovery as valid even though they have not been authenticated.

The purpose of the second discovery message is to ensure that the manager is talking with an authorized SNMPv3 agent configured by someone who knows the USM User's authentication key. The second discovery message sent by the SNMPv3 manager is sent with security level authNoPriv or authPriv. It contains the USM User's authentication key that has been localized using the **snmpEngineID** value from

the first stage of discovery. The **Report** PDU returned by the SNMPv3 agent is sent with security level authNoPriv. If the manager correctly authenticates the **Report** PDU, then it can trust that it is communicating with the correct SNMPv3 agent and can safely accept the **snmpEngineBoots** and **snmpEngineTime** values and any other data returned from the device.

2.9 Can the SNMP Manager Form All Request Types?

When selecting an SNMP manager to monitor and control SNMPv3 devices, the network administrator should choose a software product that is capable of forming requests that are compatible with SNMP multiplexers (Section 1.2).

The default value for the contextName field in the header of an SNMPv3 message is the empty string. To communicate with virtual agents, the manager must be able to form a request with an arbitrary string in the field.

There are two fields in the header of an SNMPv3 message that contain an **snmpEngineID** value. The "authSnmpEngineID" field is used for authentication. The "contextSnmpEngineID" field is used for routing. In a normal SNMPv3 message, the values in these two fields are the same. To forward an SNMPv3 message through an SNMP entity with a proxy forwarder application, the manager must be able to form a request in which the authSnmpEngineID and contextSnmpEngineID fields contain different values.

A network planner or administrator may not include virtual agents or proxy forwarders in the initial design of a network. However, if these features are needed later, it is an unfortunate time to discover that the SNMP manager software is incapable of forming the necessary packets. These features should be tested before making the final selection of an SNMP manager product.

2.10 Can the SNMP Manager Correctly Determine the Source IP Address of Notifications?

When selecting an SNMP manager to monitor and control devices, the network administrator should choose a software product that is capable of correctly identifying the source IP address of a received notification.

In the simplest case, an agent sends a **Trap** for **Inform** message directly to the manager. However, since the earliest days of SNMPv1, protocol messages could be rerouted through proxy forwarders (Section 1.2.2).

To handle the case of a proxy forwarder, the architects of the SNMPv1 protocol added a field to the header of an SNMPv1 **Trap** message called "agent-addr". This field carries the IPv4 address of the notification originator. The notification receiver knows the correct IP address from the message regardless of how many proxy forwarders that the message passed through to reach the manager.

The SNMPv2c protocol eliminated the agent-addr field. Therefore, a manager could only ascertain the IP address of the notification originator by obtaining the IP address from the network stack. Unfortunately, this information is only correct in the simple case of a direct transmission. If one or more proxy forwarders were involved in routing the **Trap** or **Inform** message to the notification receiver, the IP address reported by the network stack is the source address of the "last hop".

The specification for the coexistence between SNMPv1, SNMPv2c, and SNMPv3 protocols (RFC 2576 or RFC 3584) defines a MIB object called **snmpTrapAddress**. This MIB object carries the IPv4 address of the notification originator if a **Trap** passes through a proxy forwarder. This is the de facto method for an SNMP manager to determine the correct address of the notification originator. An SNMP manager should be programmed to look for this MIB object in the payload of a received notification and prioritize its value above all other information sources.

3 Procedures That Should Not Carry Forward

This section is written with the operator in mind who has previous experience with SNMPv1 and SNMPv2c and is familiar with the conventions common to those historic protocols.

3.1 Secrets Known To Groups of People

To access a device with SNMPv1 or SNMPv2c, the human operator would specify a community string in the SNMP message. The same community string was typically known by all of the people with similar responsibility needing to access the same device. Theoretically, the community string was a secret known only to the team of people tasked with managing the device.

Compare this paradigm to logging in to a UNIX or Microsoft Windows server. Each human operator has a unique user name. Each user name is verified with a pass phrase that is known only by the corresponding human operator. Logins by individuals are logged, and the user who is performing certain activities like file creation is recorded by the system. Using these records, an attack from within can be quickly associated to a particular human operator, whether that person actually perpetrated the act or was the victim of a stolen secret or hacked account.

SNMPv3 was designed for a user-based security model (USM), although other security models are possible. The intent was for each human operator to have his or her own USM user name and unique set of pass phrases known only to that person. One of the advantages of this approach is acceptance by human operators who are comfortable with authenticating themselves with pass phrases. Just as in the case of a server login, the benefits also include being able to associate an attack with a particular human operator. When new IT staff members are hired and existing staff members depart, USM users can be added and deleted individually, without affecting system operation or coworkers' access and maintaining the security integrity.

When SNMPv3 is deployed in a network, secrets should be held by individuals and not shared by groups of people. Polling should be performed in the name of the user responsible for that task, perhaps changing for whoever is on duty at that particular time. Notifications (**Trap** and **Inform** messages) should be sent in the name of the user who is ultimately responsible for dealing with alarm conditions.

Servers sometimes have pseudo users that do not represent a specific, named individual; e.g., "Administrator" or "root", for example. While not ideal, pseudo users also make their way into SNMPv3 configurations. For example, if printer management is a shared responsibility, printers might be configured to send notifications in the name of a pseudo user named "printer". If possible, pseudo users should be limited to unsolicited notifications and not be configured for actively accessing an agent's data, especially

for write access.

3.2 Copying a Running Configuration

Testing a configuration in a running agent and then copying it to other devices seems like a good idea whether the installer is a technical thinker or just using common sense. Operators who have done this with SNMPv1 and SNMPv2c configurations may tend to want to follow the same procedure for SNMPv3, but they would be making a mistake.

The configuration of community strings on any two devices can easily be identical, because the authentication is trivial. The configuration may be as simple as storing plaintext community strings in a read-only file.

The configuration for SNMPv3 should *never* be the same on any two devices! This does not mean that unique USM users and pass phrases must be created for each device. On the contrary, the same users and pass phrases can be used to access all of the devices. But the information stored on each device is unique, because the content of the configuration should be transformed prior to being used by the software.

The transformation process that changes pass phrases common to many devices into keys that are specific to a single device is called *localization*. This process is implemented differently by different vendors, but the inputs are always the same:

- Pass phrases or non-localized keys
- The SNMP entity's SNMP engine ID value

In SNMP Research software, the program that will eventually use the localized keys (the SNMP agent or manager) is the same program that performs localization. The inputs for localization are stored in the same place (e.g., the same disk file) where the localized keys are expected after the localization process is complete.

In software created by other SNMP vendors, the program that performs localization may be entirely separate from the program that will eventually use the localized keys. Also, the inputs for localization may come from a variety of sources, such as a disk file, command-line interface, a web interface, or an external source such as a NETCONF server or Diffie-Hellman key ignition engine.

Regardless of the implementation, it is important to never copy a configuration after the keys have been localized. Doing so introduces a security vulnerability, because if the SNMP agent in one device were to be compromised, then then SNMP agents in other devices having the same configuration would also be compromised.

If the implementation stores the inputs for localization in a form that can be copied–such as a disk file–then it is acceptable to copy the inputs. The same input file localized for more than one device will produce a unique configuration for each device.

3.3 Managers Blindly Accepting Traps

Section 2.4 offers a recommendation to network administrators who are selecting equipment and management applications to purchase for use in the network. The recommendation is that all SNMP entities should be able to handle notifications that are authenticated and encrypted. Further, all management applications that receive notifications should be able to **reject** unsecure notifications. These include:

- SNMPv1 **Trap**s
- SNMPv2c **Trap**s
- SNMPv2c **Inform**s
- SNMPv3 **Trap**s (noAuthNoPriv mode)
- SNMPv3 **Inform**s (noAuthNoPriv mode)

Typically, network management operators work furiously to try to get devices and management applications to talk to each other. When communication is established, this is usually followed by celebration and an abrupt end to concentrated effort. However, this should only be considered half of the job. Once the SNMP entities are talking to each other, the next step should be to get them to *stop* talking when the security is violated.

Section 2.4 also describes scenarios where intruders can look for ways of attacking the network by capturing **Trap**s and **Inform**s and understanding the alarm conditions and data carried in the payload. Also, an intruder can use **Trap**s and **Inform**s as a weapon to misdirect operators away from malicious activity.

Since the earliest days of SNMP, managers have blindly accepted **Trap**s, incorporated the information without verification, and acted upon them without due consideration. In modern times where security is a great concern, it is important for management applications to *refuse* to receive **Trap**s that are not delivered under the correct conditions.

4 Procedures That Should Be Adopted

This section describes some of the best practices for SNMPv3 that extend beyond the historic conventions of SNMPv1 and SNMPv2c.

4.1 Users and Access Controls

Each human operator that needs to access network devices with SNMP should receive a unique USM user name. This user name can be the same string of characters used for UNIX and/or Microsoft Windows server logins or be different at the discretion of the network administrator.

Each human operator should also have unique pass phrases for authentication and privacy. These pass phrases should be different from those used for server logins. Also, these pass phrases should be different for each authentication and privacy protocol. Having different pass phrases in these situations prevents an attacker from ascertaining the original pass phrase by cracking a key generated using a low bit-strength

algorithm. Also, if a server login pass phrase becomes compromised, the SNMP network is not compromised and vice-versa.

For additional information related to unique USM user names and pass phrases, refer to Section 3.1.

Human operators in the same workplace typically have different responsibilities and therefore a different set of privileges needed to carry out their duties. In a security-conscious environment, special care should be taken to limit the privileges for each user to only what is needed. SNMPv3 provides a rich set of features to describe the information in an agent that is accessible for read and write actions by a particular user and for notifications sent in the name of a particular user.

4.2 Changing Pass Phrases

An enterprise should apply to SNMPv3 and the user-based security model (USM) the same policies that govern logins for UNIX and Microsoft Windows servers. When changes are required for server pass phrases, changes should also be required for USM user pass phrases. If there are rules governing the selection of pass phrases for server, the same rules can also govern the selection of pass phrases for SNMPv3.

The reasons for changing the pass phrases for SNMPv3 are essentially the same that necessitate changes to server pass phrases. For more information, refer to Section 1.3.2.

5 When To Apply Security

5.1 Authentication

Authenticating SNMP requests is vital to protect network assets from intrusion and theft. There is a vast amount of information in network devices that could help an intruder to launch an attack, and the goal of authentication is to prevent a hacker from obtaining that information easily or from modifying device settings.

Without authentication, an intruder masquerading in the manager role can walk a device's MIB and capture all of the available information in a single sweep. If the device refuses the intruder's **Get**, **GetNext**, and **GetBulk** requests, then he or she must resort to capturing individual packets off of the network, which is a bit more difficult to accomplish. Capturing packets requires placing a network interface into promiscuous mode, and on UNIX servers this usually requires root privileges.

SNMPv1 and SNMPv2c offer only trivial authentication. Capturing a single SNMPv1 or SNMPv2c packet compromises the community string, and then there is effectively no authentication. So community-based agents are particularly vulnerable and should either be removed from service or be limited by SNMPv3-style access controls. Agents that support RFC 2576 or RFC 3584 can apply a small MIB view (or no MIB view) to managers that communicate with SNMPv1 or SNMPv2c. SNMP Research recommends configuring no MIB view at all. If SNMPv1 or SNMPv2c are required for network operations, then the MIB view should be made as small as possible (refer to Section 2.6).

Access controls can be configured so an agent responds to SNMPv3 requests received with no

authentication. This should not be performed, or at least, the MIB view should be as restrictive as SNMPv1 or SNMPv2c access.

Without authentication, an intruder masquerading in the agent role can send false information to the network management system as part of an attack. If the SNMP manager receives **Trap** and **Inform** messages from anonymous (SNMPv1/SNMPv2c) or unverified (SNMPv3 noAuthNoPriv) sources, the intruder can send investigators on a "wild goose chase" that leads away from the part of the network that is under attack.

SNMPv3 provides several protocols for authentication. The original protocol, MD5, is also the weakest and should not be used if other, strong authentication protocols are available. For details about authentication protocols, refer to Section 2.1.

The short answer to the question of "when should SNMPv3 messages be authenticated?" is, "whenever possible." There are some who argue that authentication of SNMPv3 messages adds a lot of overhead compared to SNMPv1 and SNMPv2c messages. Since SNMPv1 and SNMPv2c have essentially no authentication, there is almost no overhead, and by comparison SNMPv3 with authentication will require more time and computer resources.

If an SNMP agent contains large amounts of data or data that is polled frequently, is there a justifiable case for increasing the speed of retrieval by performing requests with no authentication? The question that should be asked is, "Is there any concern at all if anyone in the world were to be able to view the information contained by a group of MIB objects?" If the answer to the question is "No," then those MIB objects could be contained in a MIB view that is accessible by SNMPv1, SNMPv2c, and SNMPv3 with no authentication. Polling for this information could be done with no authentication, and all other information in the agent could be protected with authentication.

5.2 Privacy

As stated in the previous section, if network devices refuse an intruder's **Get**, **GetNext**, and **GetBulk** requests, then the intruder must resort to capturing individual packets off of the network to obtain the information in an SNMP agent. The privacy features of SNMPv3 protect the device information from disclosure when it is carried in an SNMPv3 message that is captured "in flight."

SNMPv1 and SNMPv2c have no privacy features at all and should not be used to convey any sensitive information to or from an agent.

SNMPv3 access controls can be configured so an agent responds to requests received with no privacy. This should never be done for **Set** requests. For **Get**, **GetNext**, and **GetBulk** requests, and also for SNMP notifications (**Trap** and **Inform** messages), the same question should be asked here that was posed above for authentication, "Is there any concern at all if anyone in the world were to be able to view the information contained by a group of MIB objects?" If the answer to the question is "No," then those MIB objects could be contained in a MIB view that is accessible without privacy protocols. The rest of the information should be protected by message encryption.

SNMPv3 provides several protocols for privacy. The original protocol, DES, is also the weakest and should not be used if other, stronger privacy protocols are available. For details about privacy protocols, refer to Section 2.2.

The short answer to the question of "when should SNMPv3 messages be encrypted for privacy?" is, "whenever necessary." For sensitive data that would be dangerous in the hands of a hacker, encryption is a necessary and justified expenditure of resources with great benefits.

Polling and notifications should be implemented with a mixture of encrypted and unencrypted messages. Privacy should be used when the data being conveyed is sensitive; and unsensitive data can be conveyed without exercising a privacy protocol. For **Set** requests, always use privacy protocols. For **Trap** and **Inform** messages, consider both the alarm type and the payload when deciding if the message should be encrypted.

6 What Users Should be Configured

USM Users are to SNMPv3 what community strings are to SNMPv1 and SNMPv2c. Early manageable devices came pre-configured with well-known community strings, and many devices stored the community strings in read-only memory (ROM) so they could not be changed. These antiquated paradigms are completely improper for SNMPv3. The security configuration of all SNMPv3 entities (both agents and managers) should be fully writable and entirely replaceable.

The SNMPv3 user-based security model (USM) uses private key cryptography. This means that the secret keys must be known on both ends of communication. Interactive management applications can prompt an operator for the USM user name and pass phrases. However, a non-interactive poller and a notification receiver application must be configured with the same keys that are stored in the agent.

SNMPv3 entities should be equipped with an initial configuration that enables the network administrator to apply changes that bring the unit into conformity with an enterprise-wide policy that includes the procedures described in Section 4. Device manufacturers should design their products so they do not rely on old procedures rooted in the past (Section 3).

6.1 Initial Configuration

When a device (containing an SNMPv3 agent) is installed in a network for the first time, it needs two kinds of USM users for SNMPv3:

1. **An initial administrative user**.
   This USM user, usually shortened to *initial user*, has full read-write access to the entire MIB when SNMPv3 requests are authenticated and encrypted. Most importantly, the initial user has the privilege to create, delete, and modify rows in all of the tables of the SNMPv3 Administration MIBs (Section 1.3.3).

   The name of the initial user could be "initial" or "root" or any other arbitrary name. The length of the name must be between 1 and 32, inclusive. The characters in the name should be alphanumeric and begin with a letter.

2. **Clone-from users**.
   These USM users are templates from which other USM users can be created. There should be at least one clone-from user in the initial configuration for every combination of authentication and encryption algorithm recognized by the SNMPv3 agent. For example, if the agent understands two

authentication protocols (MD5 and SHA-1) and four privacy protocols (DES, 128-bit AES, 256-bit AES, and Triple-DES), then there should be eight clone-from users:

User	authProtocol	privProtocol
user1	MD5	DES
user2	MD5	AES-128
user3	MD5	AES-256
user4	MD5	3DES
user5	SHA-1	DES
user6	SHA-1	AES-128
user7	SHA-1	AES-256
user8	SHA-1	3DES

The name of each clone-from user is arbitrary but must be unique. The length of each name must be between 1 and 32, inclusive. The characters in each name should be alphanumeric and begin with a letter. One strategy is to choose a name that is descriptive for the protocols it supports; e.g., "md5aes128user".

The initial user should be used to authenticate SNMPv3 **Set** requests that install the operational configuration (Section 6.2). The initial user is temporary: after the operational configuration is installed, the initial user should be deleted. From that point forward, the user name of the administrator responsible for network management should be used to make subsequent changes to the configuration.

If the keys belonging to a clone-from user are compromised, then any USM users created from that template are also compromised. Therefore, it is important to prevent those keys from ever being exposed to the network. An intruder should never have the opportunity to capture an SNMPv3 message containing a digest created with the authentication key of a clone-from user. Likewise, an intruder should never have the opportunity to capture an SNMPv3 message containing data that has been encrypted using the privacy key of a clone-from user. To ensure this can not happen, clone-from users should never be used to send an SNMPv3 command or notification. It follows that clone-from users should never be assigned to an access control group (no clone-from user name should ever appear in any **vacmSecurityToGroupEntry**).

The names of the initial user and clone-from users can be pre-configured from the factory or be entered by the operator who is installing the device into the network. The pass phrase for the initial user may also be pre-configured from the factory since the initial user is temporary. However, it is strongly recommended that the pass phrases for the clone-from users should never be pre-configured. This should always be entered at installation time.

The configuration entries in the **vacmSecurityToGroupTable**, **vacmAccessTable**, and **vacmViewTreeFamilyTable** that enable the initial user to create and delete rows in the SNMPv3 Administration MIBs are probably best pre-configured from the factory, and these can be re-used or replaced for the operational configuration.

6.2 Operational Configuration

The operational configuration of a device (containing an SNMPv3 agent) consists of the clone-from users from the initial configuration plus:

- USM users corresponding to each human operator authorized to remotely access the device;
- Other USM users that do not correspond to a human operator (not recommended, see Section 3.1);
- MIB views (**vacmViewTreeFamilyTable**) that assign names to sets of MIB branches;
- Access controls (**vacmAccessTable**) that assign MIB views with read, write, and notify privileges to named groups; and,
- Group assignments (**vacmSecurityToGroupTable**) that assign individual users to a named group.

The USM user name corresponding to the human administrator responsible for overall network management assumes the role of the initial user after the initial user is deleted. This user name is now used to authenticate SNMPv3 **Set** requests that apply future changes to the operational configuration of the SNMPv3 Administration MIBs.

The operational configuration of the device also consists of:

- Notifications (which type to send), which may include **Trap** or **Inform** messages, or both (**snmpNotifyTable**);
- Destinations (**snmpTargetAddrTable**), the IP addresses where the device should send **Trap** and **Inform** messages;
- Parameters (**snmpTargetParamsTable**) that indicate the USM user name(s) of the sender(s) of **Trap** messages as well as the sender(s) and retransmission policy(ies) for **Inform** messages; and,
- Filters (**snmpNotifyFilterProfileTable** and **snmpNotifyFilterTable**) that reduce the transmission of **Trap** and **Inform** messages at the source.

The parameters in the **snmpTargetParamsTable** tell the SNMPv3 agent for each destination IP address which USM user should be used to send the **Trap** or **Inform** message, and also the security level and protocols for authentication and encryption. It is recommended that this USM user should always correspond to a human operator, however pseudo-users are still commonly used for this purpose (Section 3.1).

If the device will allow limited access by SNMPv1 or SNMPv2c, then the operational configuration of the device also consists of:

- Community strings (**snmpCommunityTable**); and,
- Group assignments (**vacmSecurityToGroupTable**) that attach community strings to a named group.

Note that groups containing community strings should have no write privileges and restricted read and notify privileges.

6.3 In Management Applications

Management applications generate SNMP commands (**Get**, **GetNext**, **GetBulk**, **Set**) on behalf of a human operator. To send these commands in an SNMPv3 message, the management application must possess the USM user name and private keys that enable access to the target devices. The manager can get this information in one of two ways:

1. The management application can prompt the human operator for his or her USM user name and pass phrases at the start of a session. For example, the operator might "log in" to the management application at the start of the work shift and log out at the end of the work shift. The management application generates the private keys it needs from the credentials entered by the operator for polling and interactive commands during the length of that work shift. The management application can then localize the private keys based on the engine ID of each device to be polled.
2. The management application can be pre-configured with the USM user name and localized keys needed to access network devices for commands. Unfortunately, this means that whoever is standing in front the system console could perform polling and send interactive commands using an operator's credentials even if the human operator is not present.

Management applications may also receive alarms from devices as SNMPv3 **Trap** messages. If a **Trap** is to be authenticated and possibly decrypted, the management application must possess the localized keys for the USM user name and engine ID specified in the received message. To ensure that **Trap**s are received even when the operator is not present, the localized keys for devices that send **Trap**s should be pre-configured. Note that these are exactly the same keys in choice 2, above.

Management applications may also receive alarms from devices as SNMPv3 **Inform** messages. **Inform**s work differently from **Trap**s (see Section 6.5). If an **Inform** is to be authenticated or decrypted, the management application must possess the localized keys for the USM user name specified in the received message and the manager's engine ID. To ensure that **Inform**s are received even when the operator is not present, the localized keys should be pre-configured. Note that these are *not* the same keys in choice 2, above, so a person standing in front the system console could not perform polling and send interactive commands using an operator's credentials without the operator being present.

Unlike managed devices, management applications do not need configuration information for most of the tables in the SNMPv3 Administration MIB to carry out its primary function of sending SNMP commands and receiving SNMP notifications. User credentials that are pre-configured or memory-resident during an active session are appropriate to store in the **usmUserTable**.

Note that a management application may reside on a managed device so that its pre-configured credentials can be updated by a key management application (Section 7)[1]. In this case, the manager should implement the **usmUserTable**, and the configuration should also contain clone-from users (see the description in Section 6.1).

Management applications that allow USM user information to be pre-configured but can not be externally managed may import user names, pass phrases, engine IDs, and other necessary information in the form of a text file, sometimes called a seed file. Seed files can be created by hand, or they can be generated by an external application that discovers SNMPv3 devices in a network. For more information about generating seed files, contact SNMP Research.

6.4 In Devices That Do Not Support SNMPv3 Administration

An SNMPv3 agent that does not support SNMPv3 administration standards is probably configured with the same ad hoc methodologies that were commonly used in the past. For SNMPv1 and SNMPv2c agents, this

[1] In SNMP Research's product line, an example of this is when a BRASS Management Application runs as an EMANATE Subagent beneath an EMANATE Master Agent.

usually meant uploading a fixed, or static configuration file to a device. This is a procedure that should not carry forward (Section 3), but if it will be done as an interim solution, Section 3.2 describes how to do it without compromising SNMPv3 security.

In these devices, an initial configuration probably does not make sense. The configuration to be uploaded will be the operational configuration. If the SNMPv3 agent does not support **Set** requests, or if **Set**s are not allowed to the **usmUserTable**, then it is not necessary to include clone-from users.

Operating devices that do not support the SNMPv3 Administration MIBs is not a valid excuse to avoid regular changes of pass phrases for all USM users. Since these devices are not compatible with standards-based key management applications (Section 7), the ad hoc methodologies will need to be repeated indefinitely to upload new configurations periodically.

6.5 Traps Versus Informs

The configuration of SNMPv3 USM users for **Trap**s is the same as for polling (SNMP **Get**s) or write operations (SNMP **Set**s). So if alarms are communicated from devices to management application via SNMPv3 **Trap**s, no *additional* configuration is required in the agents. For this reason, **Trap**s are usually considered to be the easiest type of SNMPv3 notification to configure.

The configuration of SNMPv3 USM users for **Inform**s is similar to SNMP **Get**s/**Set**s/**Trap**s but in "reverse," because the roles are in reverse. With SNMP **Get**s/**Set**s, the agent receives the original message and must decide if it should send back a **Response**, so it is authoritative with respect to security. With SNMP **Inform**s, the manager receives the original message and must decide if it should send back a **Response**, so it is authoritative with respect to security.

When the agent is authoritative, the SNMPv3 USM user keys are localized with the agent's engineID. When the manager is authoritative, the SNMPv3 USM user keys are localized with the manager's engineID. In the agent, in addition to other configuration entries that are required, there must be a usmUserEntry where the first index is the engineID of the manager to which the **Inform** is to be sent.

Note that when the agent is authoritative, the manager must be configured with the private keys used by each and every USM user (that sends **Trap**s) in each and every SNMPv3 agent. Also, the agent must be configured with the private keys for its local USM users. So if there are a thousand agents and one manager with one user sending **Trap**s,

- there is one usmUserEntry entry in each of the one-thousand (1,000) agents with keys that are localized with that agent's engineID; plus,
- there are one-thousand (1,000) usmUserEntry entries in the manager, one for each unique set of private keys localized with each agent's engineID;
- there are two-thousand (2,000) usmUserEntry entries total.

When the manager is authoritative, the manager is configured with the private keys for its local USM users. Also, the agent must be configured with the private keys used by each and every USM user (that sends **Inform**s) for each manager that will receive the **Inform**s. So if there are a thousand agents and one manager with one user sending **Inform**s,

- there is one usmUserEntry entry in each of the one-thousand (1,000) agents with a common set of

keys that are localized with the manager's engineID; plus,

- there is one usmUserEntry entry in the manager localized with its own engineID;
- there are 1,001 entries total.

For the above reasons, it is easily argued that the configuration for **Inform**s is much simpler and easier than the configuration for **Trap**s! However, if your network management system performs both SNMP commands (**Get**s, **Set**s) and receives **Inform**s, then you'll need both sets of configuration entries (3,001 entries total in the above example).

7 SNMP Key Management Applications

The purpose of a key management application is to synchronize private keys between SNMPv3 agents and SNMPv3 managers. This section describes how key management applications should be used as part of the operation of today's enterprise networks.

Because of the sensitive nature of the information contained in a key management application's database (USM users and pass phrases, access controls, etc. installed in every device in the network), this type of application should only be accessible by senior-level network administrators.

7.1 Background

Enterprise networks can consist of hundreds, thousands, or tens of thousands of manageable devices (or even more). In the past, SNMPv1 and SNMPv2c were used to monitor large networks, and administrators often relied on the fact that manageable devices would respond to well-known community strings like "public" from the factory.

In the past, management of large networks was limited to monitoring. While it would have been desirable for authorized administrators to correct problems remotely with SNMP **Set** requests, opening up a device for write operations was always risky with community-based SNMP. An intruder with the ability to capture an SNMP packet could acquire the ability to make malicious changes quickly and with relative ease.

Devices that support SNMPv3 can be enabled for safe write operations by using the strong authentication and privacy mechanisms built into the protocol. To maintain the secure state of the network, it is necessary to adhere to some conventional practices including periodic changes to pass phrases for all USM users, deleting access for users who cease to be authorized (i.e., no longer employed), and so on.

SNMPv3 is robust because every device is a stand-alone entity that does not rely on an external server for its security. This means that changes to pass phrases must be applied to all devices individually. If an operator ceases to be an employee, his or her USM user should be deleted quickly from all of the devices where the operator previously had access. To carry out these operations across every manageable device in a large network requires automation.

7.2 Automation for Updates

A key management application is a central location where the entire list of authorized users and their pass phrases is kept, along with the complete list of manageable network devices to which their SNMP access extends. Changes to USM users or pass phrases are entered into the application, and the application propagates the changes through the network by performing SNMPv3 **Set** requests to each device's **usmUserTable**.

After localized keys are updated on each device, it is also necessary to make the same changes to the SNMP command generator applications and SNMP notification receiver applications that monitor and manage the enterprise. Ideally, the management application will be installed on a managed server and it's local configuration datastore can be updated in the same manner as devices in the network. This requires the management application to support the SNMPv3 Administration MIB as well (for the **usmUserTable**).

7.3 Access Controls

In addition to synchronizing private keys, a key management application will likely also help an administrator manage the access control settings for USM users in SNMPv3 agents throughout the network.

Different operators are responsible for different subnets or classes of devices. Some operators have limited privileges, such as a Helpdesk Staff Member. Other operators have a broader scope of privileges, such as a Remote Office Administrator. Some operators may have read-only access, such as the Inventory Control Personnel responsible for checking levels of toner in all networked printers. Various staff may need to receive alarms, but rarely does any single staff member want or need to see all of the **Trap**s generated by all devices.

A well-designed key management application will not only allow changes to USM users and pass phrases in the **usmUserTable**, but it will facilitate changes of users to group assignments and access levels for groups. Changes are entered into the application, and the application propagates the changes through the network by performing SNMPv3 **Set**s to these SNMPv3 Administration MIB tables:

- **vacmSecurityToGroupTable**
- **vacmAccessTable**
- **vacmViewTreeFamilyTable**

7.4 Notifications

In addition to performing other tasks, a key management application may also manage the flow of SNMPv3 notifications. To do this, the application should also be the central location where the complete list of SNMP notification receivers (**Trap** receivers) is kept. As a property of each device in its database, the application should be able to associate the destination(s) where each device should send its **Trap**s. Those associations are entered into the application, and the application propagates the changes through the network by performing SNMPv3 **Set**s to these SNMPv3 Administration MIB tables:

- **snmpNotifyTable**

- **snmpTargetAddrTable**
- **snmpTargetParamsTable**
- **snmpNotifyFilterProfileTable**
- **snmpNotifyFilterTable**

7.5 Community Strings

If the enterprise policies permit limited access to SNMPv3-enabled devices by SNMPv1 and SNMPv2c management applications, or if **Trap**s to SNMPv1 and SNMPv2c notification receivers are allowed, these could also be maintained through a key management application if both the application and the target devices support RFC 2576 or RFC 3584.

These RFCs define the coexistence between community-based SNMP and SNMPv3. A community string can be added or deleted in the **snmpCommunityTable**. Community strings in the table can be assigned to access control groups with associated privileges and MIB views, just as a USM user can be. Since a key management application may already manage access controls (Section 7.3) for USM users, it is only a small additional step to support access controls for community strings.

In this case, the key management application should also be the central location where the complete list of SNMP community strings is kept. Changes to community strings are entered into the application, and the application propagates the changes through the network by performing SNMPv3 **Set** requests to each device's **snmpCommunityTable** and the tables listed in Section 7.3.

7.6 Examples of Key Management Applications

7.6.1 Simple PolicyPro

Simple PolicyPro® (SNMP Research) is a complete key management application with full automation for updates initiated through the graphical user interface (Section 7.2).

Simple PolicyPro configures both SNMPv3 agents and managers that support the SNMPv3 Administration MIBs. It synchronizes changes to USM users and private keys end-to-end for SNMP commands and notifications (Section 7.4), including both **Trap**s and **Inform**s (Section 6.3). It also configures community strings in SNMPv3 devices that support RFC 2576 or RFC 3584 (Section 7.5).

Simple PolicyPro provides complete control for the operator to define access controls for read, write, and notify actions for all users and community strings assigned to a group (Section 7.3). Any number of groups with all associated MIB views can be added and removed together or individually from any number of devices in the network. These sophisticated details are simplified by organizing them into containers called "policies," from which the software derives its name.

7.6.2 SNMPv3 Configuration Wizard

The SNMPv3 Configuration Wizard (SNMP Research) is a minimal application for modifying the SNMPv3 Administration MIBs on individual network devices. It does not act as a central location for storing lists of USM users, community strings, and network devices, so it does not offer any features for automating changes across multiple devices. Instead, this application is primarily a teaching tool designed to reinforce the concepts of remote configuration of security in SNMPv3 agents.

The SNMPv3 Configuration Wizard has two primary functions:

- **Configure Get and/or Set access to a device.**
  This function offers a choice to configure a USM user (**usmUserTable**) or community string (**snmpCommunityTable**). At each step, the application fetches information stored in the SNMPv3 Administration MIB tables in the device (since it has no central location for storing information) to present to the operator. The operator may select existing groups (**vacmSecurityToGroupTable**), access controls (**vacmAccessTable**), and MIB views (**vacmViewTreeFamilyTable**) or create new groups, access controls, and MIB views.
- **Configure notifications.**
  This function offers a choice to configure a **Trap** or **Inform** (**snmpNotifyTable**) and a version number for SNMP (**snmpTargetParamsTable**). At each step, the application fetches information stored in the SNMPv3 Administration MIB tables in the device (since it has no central location for storing information) to present to the operator. The operator may select existing destinations (**snmpTargetAddrTable**) and parameters (**usmUserTable** or **snmpCommunityTable**) or create new.

The SNMPv3 Configuration Wizard has no provision for deleting existing rows in any of the SNMPv3 Administration MIB tables.

The SNMPv3 Configuration Wizard makes no attempt to configure any management applications to synchronize private keys installed or updated in SNMPv3 agents.

8 Scanning for Security Vulnerabilities

Networks are filled with devices containing hidden vulnerabilities. The following are just a few of the possibilities.

- Very old devices that may not have been used in many years might remain powered on, which offers intruders an open door with an SNMPv1 agent that responds to **Set** requests sent with the well-known community string “public”.
- Old and new devices with read-only community-based access to the full MIB may be exposing an address book of mission critical devices through readable ARP caches.
- Devices that send SNMPv1, SNMPv2c, and noAuthNoPriv SNMPv3 **Trap**s, combined with management applications that blindly accept all **Trap**s, present opportunities for intruders. They could learn valuable information about the state of devices. They could also send false information to managers to cover their tracks and mislead investigators.

- Devices containing SNMP agents that have not been patched for vulnerabilities announced in past CERT advisories may be installed in the recesses of a network. The same vulnerabilities may be reintroduced into a network by installing new equipment containing old software supplied by disreputable manufacturers.
- Devices configured with antiquated authentication and privacy protocols that no longer offer real protection from brute force attacks.
- SNMPv3 agents that do not comply with SNMPv3 specifications–combined with SNMPv3 management applications that overlook error conditions–can create opportunities for a system to be compromised by an intruder remain undetected.

It is important to routinely scan a network for security vulnerabilities in the network management system. After problems are discovered, then corrective actions should be taken to eliminate them. For example,

- Very old devices should be taken out of service if they are no longer needed.
- All devices in service should support SNMPv3 with strong authentication protocols (Section 2.1) and privacy protocols (Section 2.2). Devices that fail to meet current standards should be patched or taken out of service.
- All devices that contain a known vulnerability or do not comply with the SNMPv3 specification (Section 2.5) should be patched or taken out of service.
- Management applications should be configured to reject SNMP notifications that are not authenticated. All devices should be configured to send **Trap** and **Inform** messages using strong authentication. Encryption should be used if the notification carries information that would be useful to an intruder to gain a foothold in the network.

8.1 Examples of Scanning Applications

8.1.1 SNMP Security Analyzer

The SNMP Security Analyzer (SNMP Research) has a discovery engine that can find all the devices with an SNMP agent on the network. Once the SNMP Security Analyzer finds an agent on the network, it conducts extensive testing on that SNMP agent. It looks for misconfiguations that may prevent SNMPv3 communication or weaken SNMP security. It also looks for known SNMPv3 security vulnerabilities. As an alternative to a network-wide discovery, SNMP Security Analyzer can work from a pre-determined set of IP addresses. Once the analysis is complete, the SNMP Security Analyzer automatically generates reports that identify and prioritize the discovered issues.

9 Firewall Boundaries

Sometimes, network operators who are responsible for SNMP-based management view firewalls as an obstacle to overcome. An example of this is a firewall that blocks SNMP messages (or all UDP traffic) and is under the control of an administrator with conflicting priorities, who reports to a different part of the organization.

At other times, network operators who are responsible for SNMP-based management view firewalls as part of a solution to some problem they need to solve. For example, if the network contains indispensable legacy devices for which an SNMPv3 agent is unavailable, it may be necessary to preserve and protect the SNMPv1 or SNMPv2c interface from attackers by hiding the devices behind a firewall.

The following sections discuss strategies for managing devices behind firewalls with SNMP.

9.1 Proxy Forwarder

The standards-based solution for routing SNMP messages from one network domain onto another network domain is a *proxy forwarder*. Section 1.1 introduced the proxy forwarder as one of the SNMP applications defined by the SNMPv3 specification. Section 1.2.2 provided an example of how the proxy forwarder could be used as a gateway to a NAT-ted LAN. The proxy forwarder may also be an acceptable solution for crossing some firewall boundaries.

The proxy forwarder application receives SNMP messages from a UDP/IP transport and transmits SNMP messages onto a UDP/IP transport. Therefore, this solution works only when the firewall does not block UDP messages entirely. At minimum, the SNMP entity with proxy forwarder application would need to run on a DMZ host to which UDP messages arriving at port 161 on the WAN interface are routed.

When the SNMP entity containing the proxy forwarder application receives an SNMPv3 message from the WAN side of the firewall, it authenticates and optionally decrypts the message. If the contextSnmpEngineID does not equal the SNMP engine ID of the SNMP entity, it determines that the message should be forwarded. Then it looks in the **snmpProxyTable** (Section 1.3.3) and cross-references the other MIB tables in the SNMPv3 Administration MIBs to determine where and how to forward the message. The protocol version of the outbound SNMP message can be SNMPv1, SNMPv2c, or SNMPv3. For SNMPv3, the USM User and keys can (and probably should) be different than the USM User and keys used to authenticate the message received from the WAN. As previously stated, the destination IP address can be non-routable NAT address.

When the SNMP entity containing the proxy forwarder application receives an SNMPv3 **Response** message or notification (**Trap** or **Inform**) from a device in the protected network, it authenticates and decrypts the message as necessary. Then it looks in the **snmpProxyTable** (and other tables) or in its forwarding cache to determine where and how to forward the message [back] to the SNMPv3 manager.

For maximum protection of legacy devices, the computer system hosting the proxy forwarder could have two physical interfaces connected to two networks that are otherwise completely disjointed. The operating system running on that computer system could have IP forwarding turned off, so no routing is possible except the SNMP message routing performed by the SNMP entity. In this case, there is router and no firewall at all, but the concept and the application is the same as if there were.

If the proxy forwarder has one limitation, it is that the configuration must include information about every device in the protected network to which SNMP messages are to be forwarded. If there are ten thousand devices in the protected network, then the configuration of the SNMPv3 Administration MIBs may include tens of thousands of entries. Issues of scalability should be considered when an SNMPv3 deployment strategy includes proxy forwarder applications.

SNMP Research has products available that support the proxy forwarder application. Contact SNMP

Research for more information.

9.2 Transport-Layer Security

Another standards-based solution that would enable management across firewalls is SNMP over TLS (Transport-Layer Security). This approach requires that both the manager and the agent support SNMPv3 and TLS. When the manager is ready to poll an agent or send **Set** requests, or if the agent needs to send a **Trap** or **Inform** message to a manager, the sender opens a TCP connection that is secured by X.509 certificates.

This approach would satisfy the requirement of firewall administrators who choose to block all UDP traffic at the router.

This approach is not suited for most legacy systems, which would not contain an SNMP agent that supports SNMPv3 and TLS.

SNMP Research has products available that support the SNMP over TLS. Contact SNMP Research for more information.

9.3 DSSP Remote Forwarder

SNMP Research products built on BRASS™ (Bilingual Request And Security Subsystem) technology are able to cross one or more layers of firewall boundaries by deploying an optional software component in the remote network called the DSSP Remote Forwarder™. When a BRASS Management Application sends an SNMP command, the BRASS Server determines if the destination agent is reachable through the local network or exists behind a firewall. If the agent is behind a firewall where a DSSP Remote Forwarder is known to exist, the BRASS Server routes the message to the DSSP Remote Forwarder and onto the remote network, crossing the firewall(s) through an encrypted tunnel. Once on the other side, the DSSP Remote Forwarder transmits the SNMP message onto the remote network as regular UDP. The protocol version of the outbound SNMP message can be SNMPv1, SNMPv2c, or SNMPv3.

This solution works with legacy systems that support only SNMPv1 or SNMPv2c. The encrypted tunnel provides protection from disclosure of the content of the SNMP messages, including the community string and PDU payload.

This solution has one advantage over the proxy forwarder. The DSSP Remote Forwarder requires no configuration, and the BRASS Management Application does not need to keep a list of contextSnmpEngineID values for the agents with which it needs to communicate. The BRASS Management Application provides the destination IP address to the BRASS Server, and the BRASS Server conveys this information to the DSSP Remote Forwarder through the encrypted tunnel along with the SNMP message.

To use the DSSP Remote Forwarder with a third-party SNMP manager, SNMP Research offers a plug-in called the Distributed SNMP Security Pack™ (DSSP). For more information about DSSP or the DSSP Remote Forwarder, contact SNMP Research.

