摘 要

简单网络管理协议（SNMP）是目前 TCP/IP 协议簇中的网络管理协议，它提供了一种监控和管理计算机网络中各种设备资源的方法，且该方法简单易行，获得了众多厂商的支持，得到了广泛推广。SNMP 发展至今已经历了 SNMPv1，SNMPv2 及 SNMPv3 等三个版本。但是，SNMPv1 与 SNMPv2 在安全性方面存在着明显不足，为此，SNMPv3 在其基础之上大大加强了安全性和用户管理性，提出了基于用户的安全模式（USM）和基于视图的访问控制模式（VACM）。USM 用于防止信息篡改、伪装、以及滞延和重复发送。VACM 则负责控制用户访问被管对象的权限。

本课题承担 SNMPv3 系统 Agent 端新增安全特性实现的研究任务，采用统一的网络管理体制，结合 SNMPv3 提出的安全功能，设计出了一个与 SNMPv1、SNMPv2 兼容的 SNMPv3 新增安全特性的实现方案。通过编码、调试实现了 USM 和 VACM 的安全功能，设计测试方案并通过全面测试。

本文主要工作包括以下内容：

1. 深入分析 SNMPv3 的相关标准文档，并与 SNMPv1、SNMPv2 进行对比，准确把握 USM 和 VACM 的功能特点。

2. 分析 SNMPv3 系统的安全特性需求，根据 SNMPv3 系统的逻辑结构，给出了 SNMPv3 系统的 USM 和 VACM 两项功能的实现方案。

3. 详细设计 SNMPv3 系统的 USM 和 VACM 两项功能的模块化实现流程，并通过编写相应的代码以及进行调试，实现 SNMPv3 系统的安全新特性。

4. 设计 USM 和 VACM 功能的测试方案，在集成环境上对 USM 和 VACM 功能进行测试，并通过分析测试结果来验证编写的代码成功实现了 SNMPv3 系统的安全新特性。

本课题的研究成果为实现功能复杂的 SNMPv3 的完整系统奠定了良好的基础。

关键字：简单网络管理，基于用户的安全模式，基于视图的访问控制模式

ABSTRACT

Simple Network Management Protocol (SNMP) is the present network management protocol in TCP/IP stack. SNMP provides a method to monitor and supervise the various devices and resources in the computer networks. Because of its simplicity, SNMP is wide spread and supported by many manufacturers. There are three versions of SNMP which are SNMPv1, SNMPv2 and SNMPv3. To overcome the lack of security in SNMPv1 and SNMPv2, SNMPv3 is greatly improved in its security and user management, and it makes use of the User-based Security Mode (USM) and the View-based Access Control Mode (VACM). USM could be used to avoid the modification of information, masquerade, disclosure, and delay or replay of messages. VACM could be used to check whether a specific type of access (read, write, notify) to a particular object is allowed.

The project detailed described in this paper is mainly about the implementation of the security features of the Agent side in SNMPv3 system. The design adopts an uniform network management architecture, gives an implementation plan of the SNMPv3 security features which is compatible with SNMPv1 and SNMPv2. The USM and VACM security function was implemented by coding and debugging. The testing plan is designed in this paper and the system test has been passed.

The main work in this paper is listed below:

1. Analyzes the SNMPv3 standards, compares it with SNMPv1 and SNMPv2, and understands the function features of USM and VACM.

2. Analyzes the security features in SNMPv3 system, gives out the USM and VACM implementation plan according to the SNMPv3 system logical architecture.

3. Designs the USM and VACM's functional blocks and implementation flows, implements the SNMPv3 system security features through the code compiling and debugging.

4. Designs the USM and VACM testing plan, tests USM and VACM functions in the integrated environment, successfully implements the SNMPv3 system's new

security features through analyzing the testing results.

Key words: SNMPv3, USM, VACM

图目录

表目录

缩略语

CBC	Cipher Block Chaining	密码块链
CMIP	Common Management Information Protocol	通用管理信息协议
DES	Data Encryption Standard	数据加密标准
HMAC	Keyed-Hashing for Message Authentication	消息认证的键控散列
MD5	Message-Digest Algorithm 5	信息摘要算法
MIB	Management Information Base	管理信息库
NMS	Network Management Station	网络管理站
OID	Object Identifier	对象标识符
SHA	Secure Hash Algorithm	安全散列算法
SNMP	Simple Network Management Protocol	简单网络管理协议
SNMPv1	Version 1 of Simple Network Management Protocol	简单网络管理协议第一版
SNMPv2	Version 2 of Simple Network Management Protocol	简单网络管理协议第二版
SNMPv3	Version 3 of Simple Network Management Protocol	简单网络管理协议第三版
USM	User-based Security Model	基于用户的安全模式
VACM	View-based Access Control Model	基于视图的访问控制模式

独 创 性 声 明

本人声明所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得电子科技大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

签名：黄晓燕　日期：2007 年 5 月 23 日

关于论文使用授权的说明

本学位论文作者完全了解电子科技大学有关保留、使用学位论文的规定，有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅。本人授权电子科技大学可以将学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存、汇编学位论文。

（保密的学位论文在解密后应遵守此规定）

签名：黄晓燕　导师签名：

日期：　　年　　月　　日

第一章 绪论

1.1 研究意义

随着计算机网络的迅速发展，网络规模变得越来越庞大，结构越来越复杂，提供的服务也越来越多，因此网络管理也就变得越来越重要。对应于两大主要的网络模型 ISO/OSI 体系和 TCP/IP 体系，网络管理协议也存在着与之对应的通用管理信息协议（CMIP）和简单网络管理协议（SNMP）。其中，CMIP 为运行在 OSI 协议集上的开放系统提供了一个网络管理框架。它是一个完全面向对象的设计,应用了面向对象的所有概念,精心设计,功能强大,但复杂的实现及由此带来的过高的成本是其最大的缺点,因此在实际应用中并没有得到广泛使用。而 SNMP 以其简单、灵活的特点得到了广泛应用。

SNMP 提供了一种对多个供应商可协同操作的网络管理方法，可对网络上支持 SNMP 协议的工作站、交换机、路由器等设备进行监控和管理。由于它的简单性和灵活性，SNMP 迅速流行起来，成为第一个被广泛使用的网络管理协议。自 SNMP 出现以来，至今已经历了三个版本，即 SNMPv1，SNMPv2 和 SNMPv3。但 SNMPv1 和 SNMPv2 在安全和访问控制方面存在明显不足：没有提供验证和加密功能，使得 SNMP 消息被篡改、被重复发送和被窃取的可能性相当高；访问控制仅仅依赖于消息中的团体名（community）字段，通常设备制造商默认读操作的团体名为 public，写操作的团体名为 private，运营商在使用设备时通常保持默认的设置，使得恶意攻击很容易发生。即使团体名经过重新设置，未作加密处理的消息也可能被人侦听，使得没有权限的管理者非法获取读写权限。

因此，最新的 SNMPv3 在前两个版本的基础上大大加强了安全性和用户可控制性，提出了基于用户的安全模式（USM）和基于视图的访问控制模式（VACM）。USM 可以防止消息被篡改，用户被伪装，以及消息被滞延和重复发送，以此提供消息级别的安全。而 VACM 则可提供 PDU 级别的安全，它负责控制用户可以访问的被管对象，以及可以进行的访问操作。

但是，新增的安全特性大大增加了 SNMPv3 实现的复杂度；同时由于不支持 SNMPv3 的陈旧设备的广泛存在，出现了 SNMP 三个版本共存的情景。因此，认

真研究 SNMPv3 的安全新特性的实现，以及 SNMPv3 支持 SNMPv1 和 SNMPv2 的兼容性是极具实际应用价值的。

1.2 研究现状

为解决网络管理中存在的信息篡改、伪装、窃听、滞延与重复发送等安全威胁， IETF 的 SNMPv3 工作组提出了基于用户的安全模式和基于视图的访问控制模式来提高 SNMPv3 的安全性能，并先后发布了关于 USM 和 VACM 的标准文档 RFC3414[1]和 RFC3415[2] ,以此作为各设备商实现 SNMPv3 安全新特性的指导性文档。

为满足人们对网络管理日益增长的安全需求并抢夺新市场，各设备提供商，例如 IBM、HP 等都致力于开发高安全性的，并具有 SNMPv1 和 SNMPv2 兼容性支持的 SNMPv3 产品。目前已成功推出了产品 SNMP Security Pack。该产品对仅支持 SNMPv1 和/或 SNMPv2 的 SNMP 管理站设备进行扩展以支持 SNMPv3 的安全特性，并提供基于 MD5 和 SHA 的用户认证机制，以及基于 DES-CBC 的加密机制，同时也可实现支持 AES 或 3DES。

此外，在 Internet 上也出现关于 SNMPv3 的开源代码，例如著名的 Net-SNMP。Net-SNMP 是一系列用于在 IPv4 或 IPv6 网络中实现 SNMPv1、SNMPv2 和 SNMPv3 的应用软件。本课题便是基于 Net-SNMP 提供的 Agent 端 SNMPv3 开源代码，并结合实际应用环境进行修改移植的。

1.3 课题预期目标

本课题初期拟定的目标是：根据 SNMPv3 提出的新的体系结构，遵循统一的网络管理机制，在原有 SNMPv1/v2 代码基础上设计 SNMPv3 的 USM 和 VACM 功能模块。但根据对原有代码和 Net-SNMP 开源代码的认真分析，如果直接在原有 SNMPv1/v2 代码基础上添加 SNMPv3 相关代码，难度会比较大，同时也不利于以后新增功能的扩展。因此，决定将原有的 SNMPv1/v2 代码全部废弃，根据 SNMPv3 提出的新的体系结构来重新实现 SNMP Agent，在此基础之上来设计并实现 SNMPv3 系统的 USM 和 VACM 功能模块。拟通过需求分析、概要设计、详细设计、编码、调试并测试的开发步骤，将 USM 和 VACM 的安全与管理特性应用于 SNMPv3 系统，使其具有防止信息篡改、伪装、窃听、滞延与重复发送等安全功

能，以及控制用户访问权限的管理功能。

具体来讲，拟通过研究 SNMPv3 的 USM 和 VACM 的功能需求，结合设备自身特点和统一的网络管理机制，概要设计出实现 USM 和 VACM 功能模块的解决方案；依据此方案详细设计功能子模块和代码实现流程，并设计主要的数据结构；编码、调试来实现 USM 和 VACM 模块的功能；设计测试方案进行测试，并交付使用。

1.4 论文安排

本文内容安排根据课题预期目标，共分为七章，其中第二、三、四、五、六章是本文的主要工作。全文章节安排如下：

第一章论述本课题的研究意义，研究现状，以及预期目标；

第二章研究本课题的主要理论依据，包括 SNMP 协议概述，SNMPv3 体系架构，SNMPv3 消息结构以及 SNMPv3 PDU；

第三章深入分析 SNMPv3 的新增安全特性，包括其安全需求，安全框架以及关键技术。

第四章设计 SNMPv3 系统安全新特性的解决方案，详细设计 USM 和 VACM 各项功能；

第五章为 SNMPv3 系统安全新特性的实现，包括 USM 和 VACM 两大模块的各项具体功能的设计和代码实现流程，以及主要数据结构、命令接口和功能函数的设计与实现；

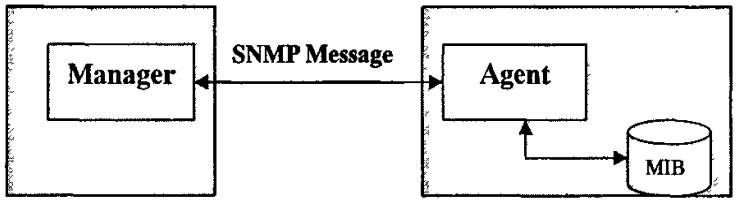
第六章为 USM 和 VACM 两大模块的功能测试；

第七章为全文工作总结。

第二章 SNMPv3 协议概述

2.1 SNMP 协议概述

SNMP 是 TCP/IP 协议簇中的应用层协议，使用 UDP 端口 161/162 进行数据传送，它提供了一种从网络上的设备中收集网络管理信息的方法。SNMP 网络管理模型采用"Manager/Agent"结构[3]。



NMS: Network Management Station，网络管理站

MD: Managed Device，被管设备

Manager: 管理器

Agent: 代理

MIB: Management Information Base，管理信息库

SNMP Message: SNMP 消息

图 2-1 "Manager/Agent"结构的参考模型

一个支持 SNMP 网络管理的系统包含以下四个基本要素：

- 两个 SNMP 实体：Manager（管理者）与 Agent（代理）。
- 网络管理协议：SNMP 协议。
- 管理信息：管理信息库（MIB）。

在 SNMP 管理体系中，被管资源的各个方面的数据变量被抽象为不同的被管对象，该被管资源所维护的全部被管对象的集合被组织为 MIB。Agent 是可以安装在任何被管理设备（PC 机，工作站，服务器，网桥，路由器等）中的软件模块，负责维护本地 MIB，以及响应 Manager 发来的命令，此外还可以主动向 Manager 通报重要事件的发生。Manager 作为管理者，可以读取和设置 Agent 所维护的 MIB 中的被管对象的值。管理协议则用于在 SNMP 实体之间，如 Manager 与 Agent 之间，

Manager 与 Manager 之间，传送协议消息[4] [5] [6]。

SNMP 协议发展至今，共有以下六种操作[7] [8] [9]：

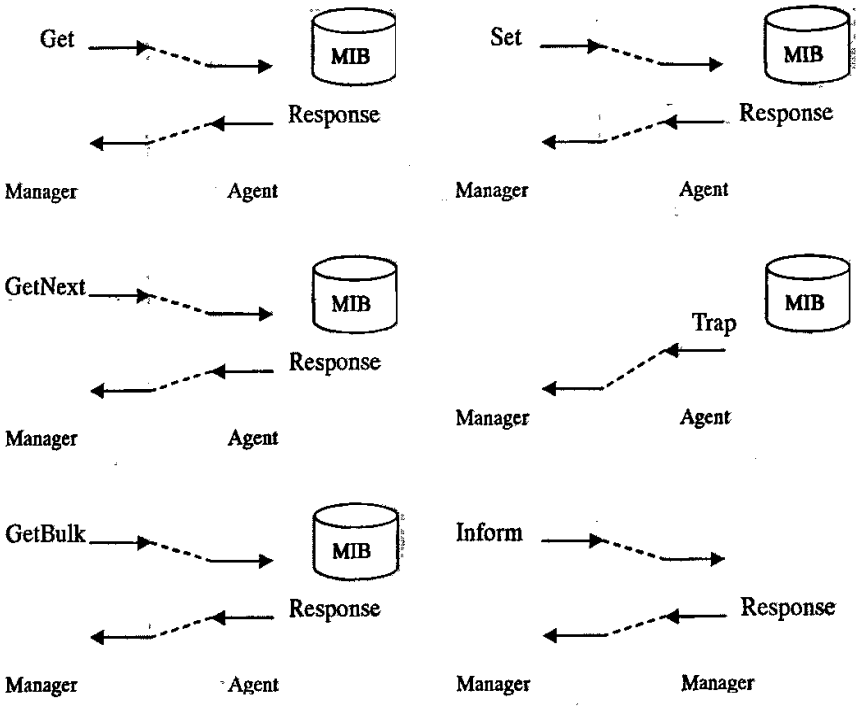


图 2-2 SNMP 协议操作

- Get：Manager 读取 Agent 处所维护的某一指定被管对象的值。
- GetNext：Manager 读取 Agent 处所维护的某一指定被管对象的后继被管对象的值。
- GetBulk：Manager 读取 Agent 处所维护的某一指定被管对象的多个后继对象的值。
- Set：Manager 设置 Agent 处所维护的被管对象的值。
- Trap：Agent 向 Manager 通报重要事件。
- Inform：用于 Manager 之间交互管理信息。

SNMPv1、SNMPv2 以及 SNMPv3 的体系结构是一致的，都采用“Manager/Agent”结构的网络管理模型，其体系架构中都包含了相同的四个基本要素。它们的主要区别则是不断完善、日益丰富而强大的功能。

2.2 SNMPv3 体系结构

SNMPv3 是建立在 SNMPv1 和 SNMPv2 的基础上的最新成果。它为 SNMP 的文档定义了更为完备的组织结构，表明 SNMP 系列协议正在变得更为成熟；它定义了统一的 SNMP 管理体系结构，采用模块化的设计思想，便于简单地实现功能的增加与修改；它总结了网络界对 SNMP 安全特性要求的发展，强调安全与管理的内在结合，因此可以认为 SNMPv3 是安全与管理特性增强了的 SNMPv2；此外，它具有很强的适应性，既可以为最简单的网络实现基本的管理功能，又能满足大型复杂网络的管理需求。

SNMPv3提出了一个新的SNMP体系架构，这个体系架构为各种基于SNMP的管理系统提供了一个通用的实现模型。SNMPv3将网络看成由许多分散的相互作用的SNMP实体构成，这些实体或者是Agent，或是Manager，或者是两者的结合。通过SNMP实体之间的相互作用来实现对网络及其资源的检测和控制。其中，每个SNMP实体由一个SNMP引擎和若干个SNMP应用构成，如图2-3所示。在一个管理域中，snmpEngineID唯一标识一个SNMP引擎。在不同管理域中的两SNMP引擎允许有相同的snmpEngineID。由于SNMP实体和SNMP引擎是一一对应的，所以snmpEngineID也用来唯一标识一个SNMP实体[4]。

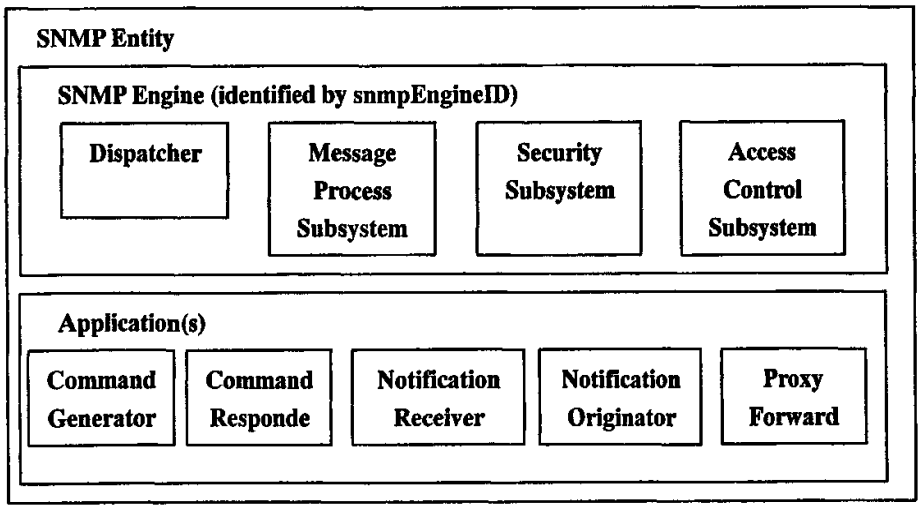


图 2-3 SNMP 实体逻辑结构图

2.2.1 SNMP 引擎

SNMP 引擎是 SNMP 实体的核心部分，主要负责消息的接收和发送，认证和

加密，以及对被管对象的访问控制。SNMP 引擎与包含它的 SNMP 实体之间是一一对应的关系。SNMP 引擎包括以下四个部分[4]：

- 调度器：是 SNMP 引擎的关键部件，每个 SNMP 引擎只有一个调度器，它允许 SNMP 引擎同时支持多种版本的 SNMP 消息。其功能包括向网络发送或从网络接受 SNMP 消息；确定 SNMP 消息的版本，与相应的消息处理模块互通；为 SNMP 应用提供抽象服务接口，向其传递 PDU 或是接收其欲发送给其它 SNMP 实体的 PDU。
- 消息处理子系统：负责准备要发送出去的消息，以及从接收到的消息里提取相关数据。消息处理子系统可包含一个或多个消息处理模块，如图 2-4 所示。每个消息处理模块定义一个特定版本的 SNMP 消息格式，功能包括形成报文格式和从报文中提取数据。

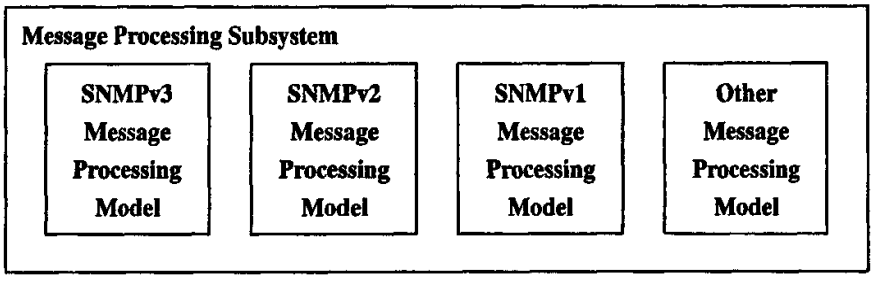


图 2-4 消息处理子系统逻辑结构图

- 安全子系统：提供消息的认证和加密等安全服务。安全子系统可包含一个或多个安全模块，如图 2-5 所示。每个安全模块定义了所要防范的网络威胁，以及所使用的安全协议，如认证和加密。SNMPv3 目前只定义了基于用户的安全模式，以后根据具体需要，可定义其它的安全模式。

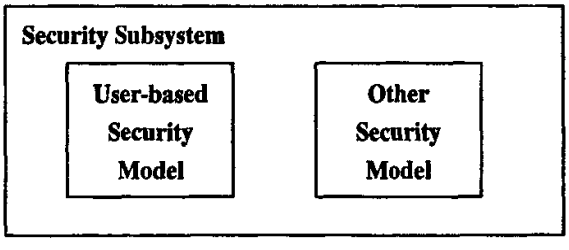


图 2-5 安全子系统逻辑结构图

- 访问控制子系统：通过一个或多个访问控制模块来提供基于 SNMP PDU 的内容授权服务，如图 2-6 所示。每个访问控制模块定义特定的访问控制策略来判

断对 MIB 的访问是否被允许。SNMPv3 目前只定义了基于视图的访问控制模式，以后根据具体需要，可定义其它的访问控制模式。

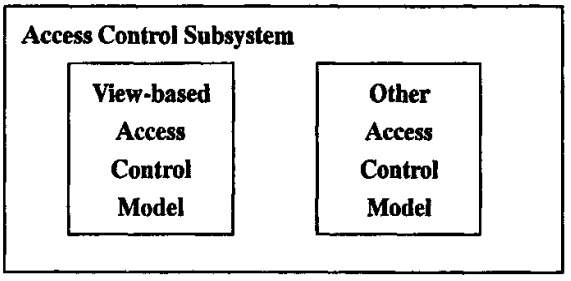


图 2-6 访问控制子系统逻辑结构图

2.2.2 SNMP 应用

SNMP 应用负责处理 PDU，完成 PDU 指定的网管操作，存取 MIB。SNMP 应用包括以下五种应用[4][10]：

- 命令发生器：用于监测和控制被管对象。即产生 Get、GetNext、GetBulk、Set、Inform 请求 PDU，并发送给调度器。。
- 命令响应器：收到命令发生器发来的请求 PDU 后，结合访问控制模块判断是否允许执行其请求的操作，如果允许，则查询或设置 MIB 中被管对象的值，并产生相应的响应 PDU。
- 通告产生器：用于触发异步消息，即用于 Manager 之间传递管理信息，或是 Agent 主动向 Manager 报告重要事件。
- 通告接收器：用于处理异步消息。
- 代理转发器：在 SNMP 实体之间转发消息。

2.2.3 SNMP Agent 结构

一个 SNMP Agent 通常包含一个或多个命令响应器，以及可选的通告接收器和通告产生器。与 Manager 的引擎不同的是，Agent 的引擎包括了访问控制子系统。本课题主要依据 SNMP Agent 逻辑结构图来实现 SNMPv3 的新增安全特性，即实现 SNMP Agent 逻辑结构图中带阴影的 User-based Security Model 和 View-based Access Control Model 两个模块[4]。

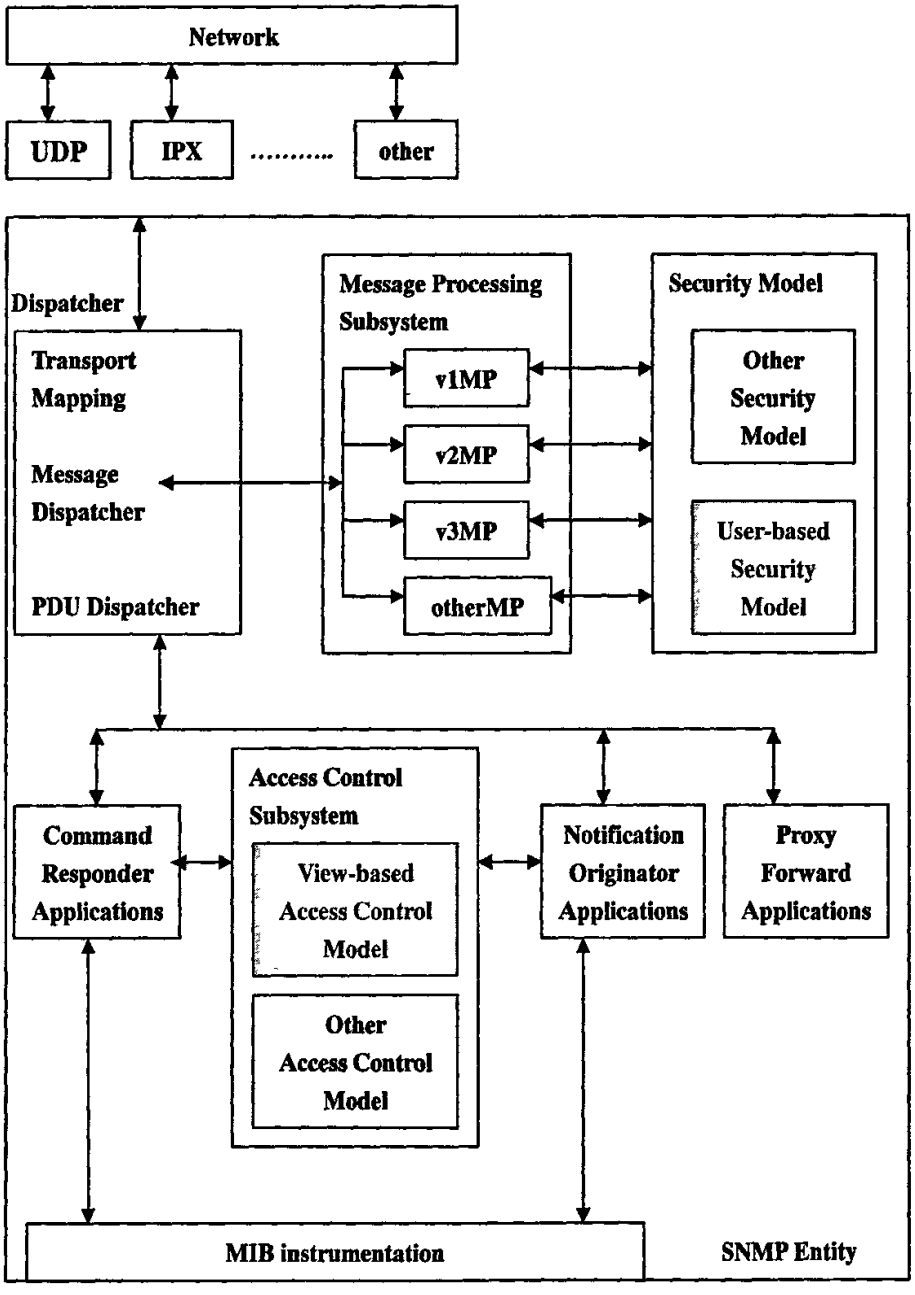


图 2-7 SNMP Agent 逻辑结构图

图 2-8 显示了 Agent 内部的各相关子系统如何通过相互作用来处理接收到的消息，以及产生并发送响应消息。具体包括了三个部分：

1. 命令响应器注册其能处理的 PDU 类型。

2. 调度器、消息处理模块、安全模块配合处理接收到的请求消息，并将请求 PDU 传递给命令响应器进行如下处理：
   (a) 检查请求 PDU 的内容。该 PDU 的操作类型必须符合之前命令相应器所注册的类型。
   (b) 调用访问控制子系统的服务原语 isAccessAllowed 来判断是否允许进行 PDU 里所请求的操作。
   (c) 如果访问被允许，命令响应器就执行请求 PDU 里要求的操作，并准备响应 PDU；反之，命令响应器则准备表示操作失败的响应 PDU。
3. 命令响应器将响应 PDU 发送给调度器，再由调度器、消息处理模块、安全模块配合产生并发送响应消息。

图中，箭头上标注的服务原语表示对该服务原语所提供的服务的一次调用；没有标注的箭头则表示调用的结果返回，并且阴影表示了调用与其返回的对应关系。

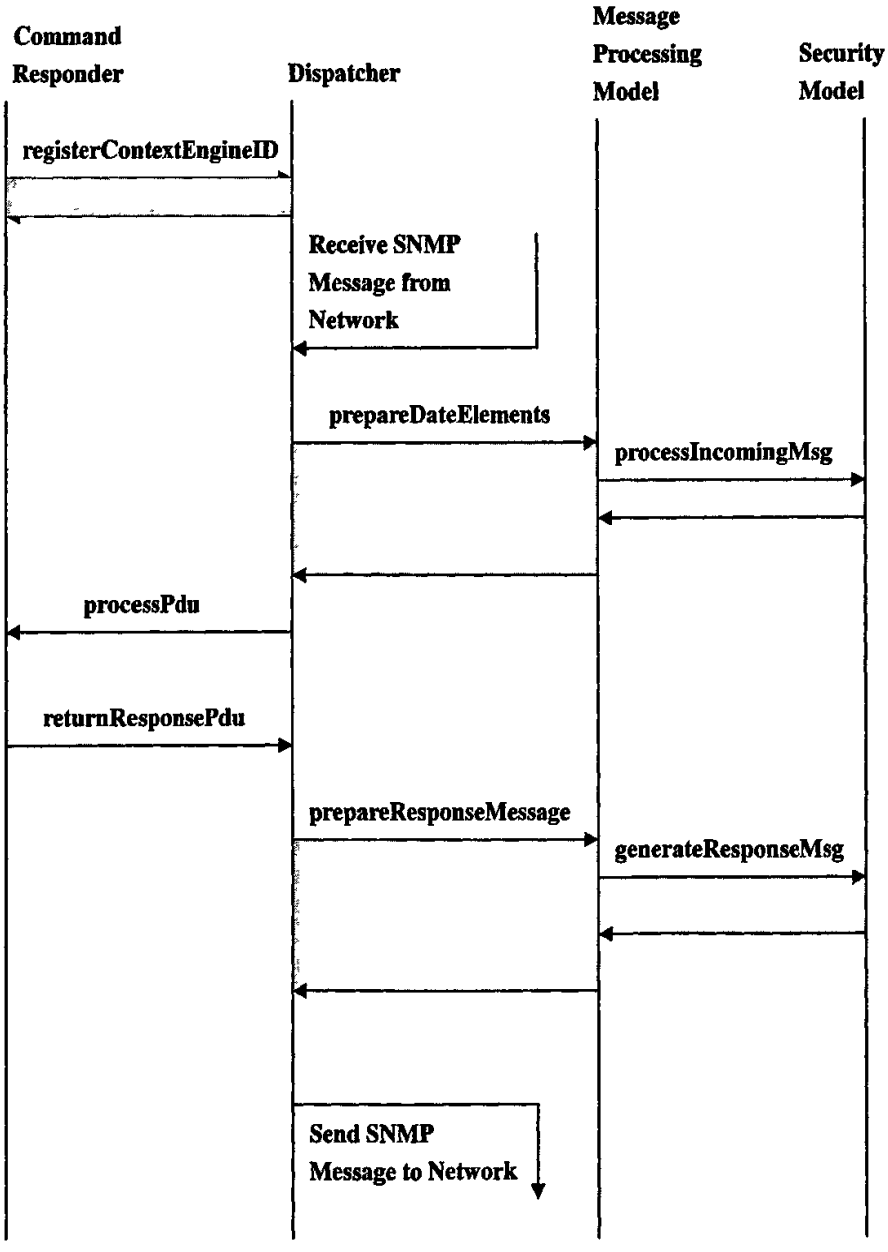


图 2-8 Agent 内部各子系统之间的交互图

2.3 SNMPv3 消息

2.3.1 SNMPv3 消息格式

按照 RFC 3412 对 SNMPv3 消息格式的描述，SNMPv3 消息由版本信息、头部数据、安全参数，以及 scopedPdu 四部分组成。

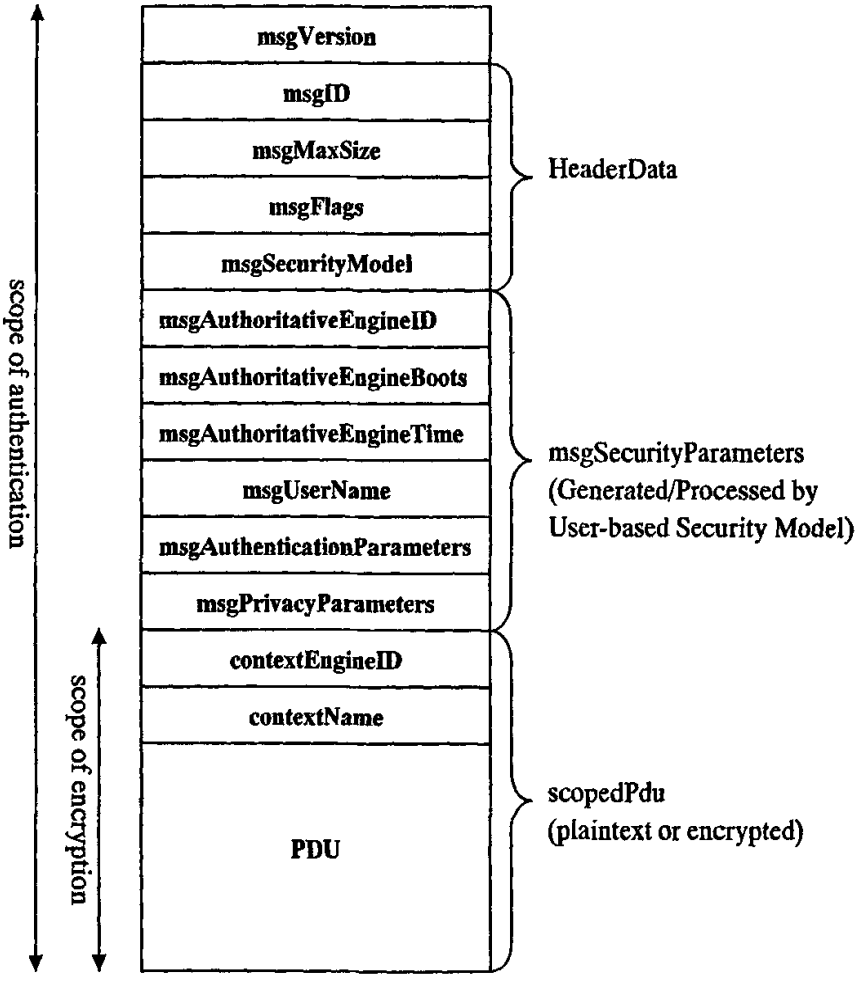


图 2-9 SNMPv3 消息结构图

- msgVersion：设置为 snmpv3(3)，表示该消息是一个 SNMPv3 消息。
- msgID：用于通信的两个 SNMP 实体唯一标识一对请求与响应消息，该请求消息与响应消息拥有相同的 msgID 值。
- msgMaxSize：消息发送者能支持的消息最大字节数，也是该实体能接收的最

大字节数。其取值范围为 484 字节～(2^31-1)字节。

- msgFlags：一个八位组，目前包含了三个标志位：reportableFlag、authFlag和privFlag，用于控制消息的处理行为。 authFlag和privFlag合起来就定义了安全级别（securityLevel），目前有三种安全级别：不认证不加密（noAuthNoPriv）、认证不加密（authNoPriv）、既认证又加密（authPriv），其安全级别依次提高。
- msgSecurityModel：标识发送端使用的安全模式，且接收端只有使用同样的安全模式才能处理该消息。SNMP 的不同版本有不同的安全模式，目前主要有 SNMPv1(1)，SNMPv2c(2)，USM(3)。对于 SNMPv3 消息，其安全模式为 USM。
- msgAuthoritativeEngineID：表示参与消息交互的权威引擎的 snmpEngineID 值。权威引擎是指不需要响应的消息（如 Trap，Response）的发送引擎，或是需要响应的消息（如 Get，GetNext，GetBulk，Set，Inform）的接收引擎。
- msgAuthoritativeEngineBoots：表示参与消息交互的权威引擎的 snmpEngineBoots 值，即权威引擎自配置了 snmpEngineID 以来重启的次数。
- msgAuthoritativeEngineTime：表示参与消息交互的权威引擎的 snmpEngineTime 值，即权威引擎最近一次重启到现在所经过的时间秒数。当其达到最大值(2^31-1)时，snmpEngineBoots 的值加 1，snmpEngineTime 的值被置为 0 并重新开始计数。
- msgUserName：用户名，用于报文的身份认证。
- msgAuthenticationParameters：进行消息认证的消息认证码，长度为 12 个八位组。如果不需要认证，则其值为空。
- msgPrivacyParameters：加密参数，用于生成进行 CBS-DES 加解密的初始化矢量(IV：Initialization Vector)。如果不需要加密，则其值为空。
- contextEngineID：用于在一个管理域中标识 SNMP context 所处的 SNMP 实体。SNMP context，即 SNMP 上下文，是能被一个 SNMP 实体访问的管理信息的集合，该集合可以跨越多个 SNMP 实体，由多个 SNMP 实体上的可访问管理信息组成。
- contextName：用于在一个 SNMP 实体内部命名一个上下文，在同一 SNMP 实体内 contextName 必须是唯一的。
- PDU：即 SNMPv2 PDU。

2.3.2 SNMPv3 PDU

SNMPv3 没有定义新的 PDU 类型，仍使用 SNMPv2 PDU，共有如下七种：GetRequest-PDU，GetNextRequest-PDU，GetBulkRequest-PDU，Response-PDU，SetRequest-PDU，InformRequest-PDU，SNMPv2-Trap-PDU[11][12]。

1. GetRequest-PDU，GetNextRequest-PDU，SetRequest-PDU，InformRequest-PDU 的格式如图 2-10 所示：

PDU Type	Request-id	0	0	Variable-bindings

图 2-10 GetRequest-PDU,GetNextRequest-PDU,SetRequest-PDU,InformRequest-PDU 格式

2. GetBulkRequest-PDU 的格式如图 2-11 所示：

PDU Type	Request-id	Non-repeaters	Max-repetitions	Variable-bindings

图 2-11 GetBulkRequest-PDU 格式

3. Response-PDU 的格式如图 2-12 所示：

PDU Type	Request-id	Error-status	Error-index	Variable-bindings

图 2-12 Response-PDU 格式

4. SNMPv2-Trap-PDU 的格式如图 2-13 所示：

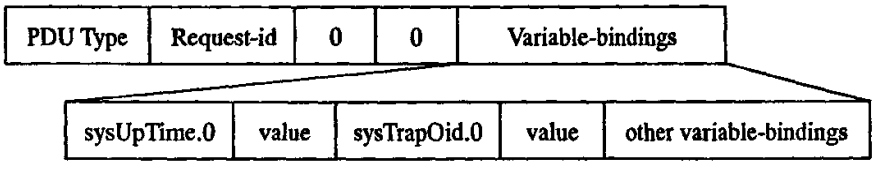


图 2-13 SNMPv2-Trap-PDU 格式

其中，variable-bindings 的格式如图 2-14 所示：

Name 1	Value 1	.....	Name n	Value n

图 2-14 variable-bindings 的格式

PDU 中各字段的含义如下：

- PUD Type：标识 PDU 的类型。
- Request-id：唯一标识一个请求。
- Error-status：表示错误状态，noError(0)，tooBig(1)，noSuchName(2)，badValue(3)，readOnly(4)，genErr(5)等错误状态。
- Error-index：当 error-status 非 0 时，指示出错变量在变量绑定列表里的位置。
- Variable-bindings：变量名与其对应值的绑定列表。
- Non-repeaters：指出只返回一个后继变量的变量数。
- Max-repetition：指出变量绑定列表里的其它变量应返回的最大后继变量数。
- sysUpTime.0：为 MIB 中 system 组 sysUpTime 节点的取值。
- sysTrapOid.0：表示当前 trap 的对象标识符(OID：Object Identifier)

第三章 SNMPv3 安全新特性分析

3.1 SNMPv1/v2 安全概述

1990 年 5 月，RFC1157 定义了 SNMPv1，随着 SNMP 应用的逐步推广，SNMPv1 暴露出了明显不足，主要表现为管理功能不完善，效率不高；以及缺乏有效的安全机制。为解决这些问题，1993 年发布了 SNMPv2，它在 SNMPv1 的基础上进行了功能性扩展，通过扩展数据类型、增加协议操作类型等方法增强了管理功能，同时在安全方面也提出了解决方案，但由于安全解决方案过于复杂，所以人们只接受了其修改版本 SNMPv2c，版本 SNMPv2c 也称为“基于团体（community）的 SNMPv2”，它保留了管理功能的改进，而安全方面则仍然使用 SNMPv1 的基于 community 的认证方式，因此，SNMPv2 并没有有效解决 SNMP 面临的安全性问题[13] [14] [15] [16] [17]。

总的来说，SNMPv1 和 SNMPv2 在安全和访问控制方面所存在的问题是：(1) 没有提供验证和加密功能，使得 SNMP 消息被篡改、被重复发送和被窃取的可能性相当高；(2) 访问控制仅仅依赖于消息中的团体名（community）字段，通常设备制造商默认读操作的团体名为 public，写操作的团体名为 private，运营商在使用设备时通常保持默认的设置，使得恶意攻击很容易发生。即使团体名经过重新设置，未作加密处理的消息也可能被人侦听，使得没有权限的管理者非法获取读写权限。

3.2 SNMPv3 安全需求

在网络环境中，管理实体之间传递的管理消息受到多种威胁，最主要的威胁有如下四种[4] [18]：

- 篡改：非授权实体篡改正在传递中的由授权实体产生的 SNMP 消息，以此执行未被授权的管理操作。
- 伪装：非授权用户假冒授权用户来进行未被授权的管理操作。
- 滞延和重传：管理消息在传输过程中被故意延迟或重复发送。

- 窃听：消息在传输过程中非授权用户对其进行拷贝，通过消息副本来获取消息中的信息。

为提高 SNMP 的安全性，SNMPv3 需考虑提供以下几种安全服务：

1. 数据完整性：保证数据在传输过程中未被修改。
2. 数据保密性：对数据提供保护措施使其内容不被泄漏。
3. 数据来源认证：实现对接收数据的发送者的身份认证。
4. 消息及时性保护：保证只处理在有效时间窗内接收到的消息。
5. 访问控制服务：通过对合法用户的授权，实现控制访问者对不同被管资源的访问权限。

3.3 SNMPv3 安全关键技术

3.3.1 概述

针对网络环境中存在的信息的篡改、伪装、窃听、滞延与重复发送等安全威胁，SNMPv3 提出了四个安全目标：对每个接收到的 SNMP 消息进行认证检查，确认其在传输过程中没有被修改，即进行数据完整性认证；提供对发送消息的用户的身份认证，防止伪装用户；提供对接收到的消息的时效检测，防止消息滞延或重传；必要时提供消息的保护，以防止消息内容的泄漏。

为完成以上安全目标，SNMPv3 提出了基于用户的安全模式（USM）。USM 提供了身份认证、消息加密和时间窗校验等安全服务。与此同时，SNMPv3 将安全与管理相结合，提出了基于视图的访问控制模式（VACM）来提高 SNMPv3 系统的用户管理性。VACM 可以在 PDU 级提供安全服务，负责控制用户可以访问的被管对象，以及可以进行的访问操作。

3.3.2 USM

在 SNMPv3 的安全通信中，USM 提供了身份认证、消息加密和时间窗校验等安全服务。USM 使用消息认证码 MAC 对交互的消息进行来源认证和完整性认证，以防止用户假冒和消息篡改，建议使用的认证协议有：HMAC-MD5-96 或 HMAC-SHA1-96[19] [20] [21] [22]。同时，USM 使用加密方法来保证消息在传输中不被泄漏，建议使用 CBC-DES 加密协议[23] [24]。此外，USM 使用时限检查机制来防止

消息滞延和重复发送。

USM 为 SNMP 实体维护了一个用户信息表 usmUserTable，包含了用户的相关属性信息，如 UserEngineID（与用户通信的权威引擎的 SnmpEngineID）、userName（用户名）、securityName（安全名）、authProtocol 和 authKey（用户使用的认证协议和密钥）、privProtocol 和 privKey（用户使用的加/解密协议和密钥）等等。通信双方实体通过所维护的用户信息，对收发的消息进行安全处理[1]。

此外，在使用认证与加密服务时，通信的 Manager 和 Agent 须共享用户的认证密钥和加密密钥。然而，一个用户可能拥有多个 Agent 的访问和控制权限，如果该用户在每个 Agent 上的密钥都相同，必将导致安全隐患；但是让用户记忆大量的密钥，又会增加用户的负担。为此，USM 采用了密钥本地化机制，用户只需记忆自己的认证密码和加密密码，由 USM 负责将密码转换成认证协议和加密协议所需的本地化的密钥。采取这种方式，可以缓解字典式密钥攻击的速度；同时，由于不同用户的密钥各不相同，并且同一个用户在不同 Agent 上的密钥也不同，因此一个用户在某一个 Agent 上的密钥收到损害不会影响到其它的 Agent。

USM 模块的用户管理，密钥本地化，身份认证，时间窗校验，以及消息加解密等功能的具体操作过程分别详见 4.2.2 小节，4.2.3 小节，4.2.5 小节，4.2.6 小节，4.2.7 小节。

3.3.2 VACM

VACM 在 SNMPv3 安全通信中提供 PDU 级别的安全服务，负责控制用户可以访问的被管对象，以及可以进行的访问操作。为提供访问控制服务，VACM 需定义组（Groups）、安全级别（securityLevel）、上下文（Contexts）、MIB 视图（MIB Views），以及访问策略（Access Policy）等五个基本要素[2]。

- 组：包含一个或多个由<安全模型，安全名>二元组标识的用户，同组的所有用户具有相同的访问权限，但每个用户只能加入一个组。每个组由组名（groupName）标识。
- 安全级别：表示在进行访问权限控制时所需使用的安全级别，共有三种：不认

证不加密（noAuthNoPriv）、认证但不加密（authNoPriv），以及既认证有加密（authPriv）。同组的用户可以定义不同的安全级别。

- 上下文：是能被一个 SNMP 实体访问的管理信息的集合，该集合可以跨越多个 SNMP 实体，由多个 SNMP 实体上的可访问管理信息组成；同时，同一管理信息可以属于多个上下文。每个上下文由上下文名（contextName）标识。
- MIB 视图：是一个管理对象的集合，由一个或多个视图子树组成。而一个视图子树是一系列具有相同 OID 前缀的 MIB 对象实例。
- 访问策略：用于定义一个组的访问权限。在一个某组通过使用某特定的安全模型和安全级别可以访问的上下文中，通过使用一个读视图（read-view），写视图（write-view），和通告视图（notify-view）来定义该组在这个上下文中的访问权限。其中，读视图表示了该组能进行读操作的 MIB 视图；写视图表示了该组能进行写操作的 MIB 视图；通告视图表示了该组能进行通告的 MIB 视图。

此外，为实现访问请求控制功能，VACM 还需使用四张表：上下文表（vacmContextTable）、组映射表（vacmSecurityToGroupTable）、访问控制表（vacmAccessTable）、视图子树表（vacmViewTreeFamilyTable）来实现访问控制机制。上下文表用于存储本地所有可被访问的上下文的名字。组映射表用于存储<安全模型，安全名>二元组与组名的对应关系。访问控制表用于存储各组的访问权限，指定了某个组在特定的上下文中，使用特定的安全模型与安全级别，能访问哪些读视图，写视图以及通告视图。视图子树表用于存储 MIB 视图内包含了哪些视图子树，以及不包含哪些视图子树的相关信息。访问控制的具体过程详见 4.3.5 小节。

第四章 SNMPv3 安全新特性设计方案

4.1 概述

本课题主要负责研究与实现 USM 和 VACM 两个模块，是整个 SNMPv3 系统开发项目的一个重要环节。SNMPv3 系统开发项目最初拟定为在原有 SNMPv1/v2 代码之上，基于 Net-SNMP 5.1 提供的 SNMPv3 系统 Agent 端的开源代码进行修改移植来实现 SNMPv3。

但是，经分析与比较，SNMPv3与SNMPv1/v2在体系结构、消息处理流程，以及数据结构上都有较大的差别。如果直接在已有的SNMPv1/v2代码基础上添加SNMPv3相关代码，难度会比较大，而且这样合并之后的SNMP协议的一致性可能会受到影响，存在着一定的风险，也不利于以后新增功能的扩展。此外，根据标准v3实体可以与v1/v2实体通信，同时在SNMPv3系统设计中也特别考虑了v3对v1/v2的兼容性支持，因此，决定将现有的SNMPv1/v2代码全部废弃，按照SNMPv3提出的新的体系结构来重新实现SNMP Agent，同时支持SNMPv1、SNMPv2和SNMPv3。这样，运行新一代SNMPv3软件的网络设备便可与网络中已存在的只支持SNMPv1和/或SNMPv2的设备通信，实现了SNMP三个版本在网络中的共存。

4.2 USM 的功能设计

根据标准并结合实际应用情况，将 USM 的功能细化为以下七个子功能：权威引擎 ID 的管理；用户管理；密钥本地化；消息重发的防止；伪装与篡改的防止；消息滞延和重复发送的防止；以及消息泄漏的防止。

4.2.1 权威引擎 ID 的管理

引擎ID用于在一个管理域里唯一标识一个SNMP引擎，它由SNMPv3引入到协议中来的。由于SNMP实体和SNMP引擎是一一对应的关系，所以引擎ID也用来唯一标识一个SNMP实体。在Manager与Agent的通信中，Agent通常作为权威引擎，所以其本地引擎ID即为权威引擎ID。

引擎 ID 包括三个部分：<enterpriseid>.< localEngineIDType>.<填充段>

- enterpriseid：占 4 个八位组，最高比特位置为 1。
- localEngineIDType：占 1 个八位组，确定后面的填充段的格式，它有 5 种取值：

  ENGINEID_TYPE_UCD_RND

  ENGINEID_TYPE_TEXT

  ENGINEID_TYPE_IPV6

  ENGINEID_TYPE_MACADDR

  ENGINEID_TYPE_IPV4

- 填充段：其长度根据 localEngineIDType 的不同而不同，填充规则如下：

ENGINEID_TYPE_UCD_RND	oldEngineID 或一个随机数+当前时间的秒数
ENGINEID_TYPE_TEXT	管理性文字信息
ENGINEID_TYPE_IPV6	IPv6 地址
ENGINEID_TYPE_MACADDR	MAC 地址
ENGINEID_TYPE_IPV4	IPv4 地址

Agent 的引擎 ID 可通过两种方法生成：一是由设备自动生成；一是由用户通过命令行配置。根据设备自身特点，若采用设备自动生成的方式，则规定使用填充 MAC 地址方式。

4.2.2 用户管理

USM 是基于用户的安全模型，每个 Agent 允许一定数量的用户对其进行访问和控制，用户管理是整个 USM 的基础。

4.2.2.1 添加用户

功能描述：为 Agent 添加一个用户，以便进行鉴权、加密以及访问控制。

输入　：用户名

输出　：如果成功给出成功信息，否则给出失败的原因

4.2.2.2 删除用户

功能描述：为 Agent 删除一个用户，以取消该用户的访问权限。

输入　：用户名

输出　：如果成功给出成功信息，否则给出失败的原因

4.2.2.3 为用户配置/修改密码

功能描述：为用户配置鉴权密码或加密密码，二者是生成鉴权密钥和加密密钥的基础。

输入　　：用户名，密码

输出　　：如果成功给出成功信息，否则给出失败的原因

4.2.3 密钥本地化

密钥本地化处理的具体操作如下：

1. 将用户的密码进行循环处理以生成长为(2^20)八位组的 digest0；
2. 如果需要使用16个八位组长的密钥，则将digest0进行MD5变换以生成digest1；如果需要使用 20 个八位组长的密钥，则将 digest0 进行 SHA-1 变换以生成 digest1；
3. 将 digest1、权威引擎的 snmpEngineID 和 digest1 连接起来生成 digest2；
4. 如果需要使用 16 个八位组长的密钥，则将 digest2 进行 MD5 变换；如果需要使用 20 个八位组长的密钥，则将 digest2 进行 SHA-1 变换，其输出结果即为用户的本地化密钥。

输入　　：用户密码，权威引擎 ID

输出　　：认证密钥或加密密钥

4.2.4 消息重发的防止

消息在传输过程中可能被截获并发给非目的地的 SNMP 实体，USM 对此问题的解决方案是在消息中添加权威引擎的 snmpEngineId。权威引擎在收到消息后检查消息中的 snmpEngineId 是否与自己的 snmpEngineId 相等，如果不相等，就立即丢弃该消息。

4.2.5 伪装与篡改的防止

为防止伪装和篡改，USM 需采用身份认证机制。身份认证是指通过数字签名技术使得 SNMP 实体在接收到消息后确认消息是否来自授权实体，并且消息在传输过程中未被改变的过程。身份认证可以防止假冒身份，保证信息的完整性。身份认证包括数据完整性与数据起源认证。USM 使用 HMAC-MD5-96 或

HMAC-SHA-96 认证协议进行身份认证。

身份认证的实现方法是：Manager 将用户的本地化认证密钥和待传输的消息作为 HMAC-MD5-96 或 HMAC-SHA-96 认证协议定义的算法的输入，通过该算法计算出消息认证码，然后将其填入消息的认证参数 msgAuthenticationParameters 字段中传输；Agent 收到该消息后使用同一认证密钥计算该消息的消息认证码，并与接收到的消息认证码进行比较，若二者相同，则认为该消息可靠，反之则不可靠，拒绝接受并返回错误信息。

USM 的认证服务提供了如下两个内部服务原语：

1. 用于认证发出消息的服务原语。

```
statusInformation =
authenticateOutgoingMsg(
  IN    authKey            /*用于认证的认证密钥*/
  IN    wholeMsg           /*待认证的消息*/
  OUT   authenticatedWholeMsg     /*认证后的消息*/
)
```

2. 用于认证接收到的消息的服务原语。

```
statusInformation =
  authenticateIncomingMsg(
  IN    authKey            /*用于认证的认证密钥*/
  IN    authParameters     /*接收到的认证参数*/
  IN    wholeMsg           /*待认证的消息*/
  OUT   authenticatedWholeMsg    /*认证后的消息*/
)
```

4.2.5.1 发送方认证

功能描述：为待传输的消息生成用于认证的消息认证码

输入　　：待发送的消息（认证参数字段为零），认证密钥

输出　　：添加了消息认证码的待发送消息

4.2.5.2 接收方认证

功能描述：验证接收到的消息是否合法，防止消息被伪装或篡改

输入　　：接收到的消息，认证密钥

输出　：如果接收方计算的消息认证码与发送方产生的相同，则返回通过认证的消息；否则返回认证失败原因

4.2.6 滞延与重复发送的防止

在 SNMPv3 的安全通信中，USM 采用了松散的时钟同步机制来判断接收到的消息是否被恶意滞延或重复发送。每个权威引擎，即 Agent，在本地维护了一个"时钟"值，它由 snmpEngineBoots 和 snmEngineTime 两部分组成，用以表示权威引擎的一个当地时间。当权威引擎第一次被安装时，snmpEngineBoots 和 snmEngineTime 均被初始化为 0；随后 snmEngineTime 的值每秒加 1，当其达到最大值（2^31－1）时，snmpEngineBoots 的值加 1，snmEngineTime 的值被置为 0 并重新开始计时。此外，若权威引擎发生重启，则将 snmpEngineBoots 的值加 1，并将 snmEngineTime 的值置为 0。当权威引擎发送消息时，就把 snmpEngineBoots 和 snmEngineTime 值分别填入 msgAuthoritativeEngineBoots 和 msgAuthoritativeEngineTime 字段，以便非授权引擎保持与权威引擎的同步。

与此同时，每个与该权威引擎通信的非权威引擎，即 Manager，通过在本地维护三个时间指示值来"跟踪"权威引擎的"时钟"值：snmpEngineBoots（权威引擎的 snmpEngineBoots 的最新值）、snmpEngineTime（非权威引擎所记录的权威引擎的 snmpEngineTime 值）和 latestReceivedEngineTime（非权威引擎从权威引擎接收到的消息里的 msgAuthoritativeEngineTime 的最大值），并在适当的条件下对其进行更新以保持与权威引擎的松散时钟同步。

首先，时间窗校验机制包含了一个时钟初始化的过程：非权威引擎 Manager 向权威引擎 Agent 发送一个需要认证的请求消息，该消息的 msgAuthoritativeEngineID 字段的值为 Agent 的 snmpEngineID，msgAuthoritativeEngineBoots 和 msgAuthoritativeEngineTime 字段的值为零，msgUserName 字段为一合法用户名。Agent 接收到该消息后，响应一个通告消息，消息的 msgAuthoritativeEngineBoots 和 msgAuthoritativeEngineTime 字段为该 Agent 的 snmpEngineBoots 和 snmpEngineTime 的值。这样，Manager 便"学到"了 Agent 的时钟信息，对本地维护的时钟进行初始化。在后续的 SNMP 消息交互中，将根据具体情况动态进行时间同步。

USM 为所有的消息规定了一个 150 秒的时间窗，只有在该时间窗里接收到的消息才是有效的。SNMP 实体每接收到一个消息，在通过身份认证判定其可靠后，

便会进行时限检查来判断该消息是否在时间窗内。

当权威引擎作为接收者时，若以下三个条件满足其中任何一个，则接收到的消息在时间窗之外，被视为无效消息：

1. 权威引擎在本地维护的 snmpEngineBoots 值为 2147483647；
2. 接收消息的 msgAuthoritativeEngineBoots 字段的值与本地的 snmpEngineBoots 值不相同；
3. 接收消息的 msgAuthoritativeEngineTime 字段的值与本地的 snmpEngineTime 值的差值超过 150 秒。

当非权威引擎作为接收者时，若以下条件至少满足一个，非权威引擎将更新本地维护的 snmpEngineBoots、snmpEngineTime 和 latestReceivedEngineTime 的值。进行更新的条件有：

1. 接收消息的 msgAuthoritativeEngineBoots 字段的值大于本地的 snmpEngineBoots 值；
2. 接收消息的 msgAuthoritativeEngineBoots 字段的值等于本地的 snmpEngineBoots 值，且 msgAuthoritativeEngineTime 字段的值大于本地的 latestReceivedEngineTime 值。

具体的更新操作如下：

1. 将本地的 snmpEngineBoots 的值设置为接收消息的 msgAuthoritativeEngineBoots 字段的值；
2. 将本地的 snmpEngineTime 的值设置为接收消息的 msgAuthoritativeEngineTime 字段的值；
3. 将本地的 latestReceivedEngineTime 的值设置为接收消息的 msgAuthoritativeEngineTime 字段的值。

若以下三个条件满足其中任何一个，则接收到的消息在时间窗之外，被视为无效消息：

1. 非权威引擎在本地维护的 snmpEngineBoots 值为 2147483647；
2. 接收消息的 msgAuthoritativeEngineBoots 字段的值小于本地的 snmpEngineBoots 的 值；
3. 接收消息的 msgAuthoritativeEngineBoots 字段的值等于本地的 snmpEngineBoots 的 值，但 msgAuthoritativeEngineTime 字段的值比本地的 snmpEngineTime 的值小，且差值大于 150 秒。

4.2.7 消息泄漏的防止

消息在传输过程中可能被截获，PDU 的内容可能被解析出来。为了解决这个问题，USM 需采用消息加密保护机制来防止消息泄漏。USM 目前采用 CBC-DES 对称加密算法，其加密密钥和解密密钥相同。当发送一个消息时，如果需要加密，则对消息的 scopedPdu 部分进行加密并设置消息的加密参数 msgPrivacyParameters 字段；否则将该字段置空。

用 16 个八位组长的本地化加密密钥的前 8 个八位组生成 DES 密钥，其后 8 个八位组连接一个 8 个八位组长的"salt"生成初始矢量 IV。其中，"salt"是 SNMP 引擎通过连接 4 个八位组长的 snmpEngineBoots 和一个 4 个八位组长的本地生成的整数而得到的。该"salt"值被填入消息的加密参数 msgPrivacyParameters 字段中传输，以便接收实体根据该"salt"值计算出正确的初始矢量来解密消息。

加密过程是：将 scopedPdu 明文分成 64 比特长的块，若最后一块不够 64 比特，则在其后填充任意数据以补足。将每块明文与前一块的密文相异或，所得结果再进行 DES 加密，以生成当前明文块的密文；对于第一块明文，则将其与初始矢量 IV 进行异或，所得结果再进行 DES 加密，以生成第一块明文的密文。

接收实体使用接收消息里的 msgPrivacyParameters 字段的值，以及本地化加密密钥来生成初始矢量 IV 以进行解密。解密过程是：将第一块密文先进行 DES 解密，所得结果再与初始矢量 IV 相异或，以获得第一块明文；随后的每块密文都先进行 DES 解密，所得结果再与前一块密文相异或，以获得当前密文块的明文。

USM 的加密服务提供了如下两个内部服务原语：

1. 用于加密数据的服务原语。

```
statusInformation =
encryptData(
   IN     encryptKey          /*用于加密的加密密钥*/
   IN     dataToEncrypt       /*待加密的数据，即 scopedPDU*/
   OUT    encryptedData       /*加密后的数据*/
   OUT    privParameters      /*将填入消息的加密参数*/
)
```

2. 用于解密数据的服务原语。

```
statusInformation =
decryptData(
   IN     decryptKey          /*用于加密的解密密钥*/
   IN     privParameters      /*接收到的加密参数*/
   IN     encryptedData       /*接收到的加密数据*/
```

```
    OUT    decryptedData      /*解密后的数据，即 scopedPDU*/
)
```

4.2.7.1 发送方加密

功能描述：为待传输的消息的 scopedPDU 进行加密

输入　　：待加密的 scopedPDU，加密密钥

输出　　：加密后的 scopedPDU，以及加密参数

4.2.7.2 接收方解密

功能描述：为接收到的消息的 scopedPDU 进行解密

输入　　：待解密的 scopedPDU，解密密钥

输出　　：如果解密成功则返回解密后的 scopedPDU；否则返回失败原因

4.3 VACM 的功能设计

根据标准并结合实际应用情况，将 VACM 的功能细化为以下五个子功能：组的定义；MIB 视图的定义；访问控制权限的定义；团体名到安全名的映射；以及访问请求控制。

4.3.1 组的定义

功能描述：建立组名与{安全模型、安全名}的映射关系。

输入　　：安全模型，安全名，组名。目前有 V1/V2C/USM 三种安全模型，安全名和组名均为不超过 32 个字节的标识符

输出　　：配置的组只在访问控制时使用，一般不对外输出。但是，可以通过 MIB 和命令行查看到现有已经配置好的组。

4.3.2 MIB 视图的定义

每个 MIB 视图由一或多个四元组{MIB 视图名（ViewName）、子树 OID（Subtree）、掩码（Mask）、包含类型（Type）}组成。其中，MIB 视图名是一个长度不超过 32 个字节的标识符，作为每个 MIB 视图的唯一标识。子树 OID 是一个长度不超过 128 的 OID，指明以此为根的管理对象。掩码是一个最大长度不超过 16 个字节的二进制串，它表明对应位的子树 OID 是否有效。如果某位的二进制为

"1"，则子树对应位的 OID 有效，参与校验；否则，匹配时该对应位的 OID 不作匹配。包含类型是指该 MIB 视图是包含以上指定的管理对象，还是不包含以上指定的管理对象。

输入：MIB 视图名、子树 OID、掩码、包含类型

输出：配置的 MIB 视图只在访问控制时使用，一般不对外输出。但是，可以通过 MIB 和命令行查看到现有已经配置好的 MIB 视图。

4.3.3 访问控制权限的定义

访问控制权限类似于一个六元组关系，即{组、上下文、安全模型、安全级别、请求类型、MIB 视图}。其中，组由一个组名来标识。上下文由一个上下文名来标识。安全模型目前有 v1/v2c/USM 三种，SNMPv1 和 SNMPv2c 报文的安全模型分别为 v1 和 v2c，而 SNMPv3 报文的安全模型要求必须是 USM，否则请求被拒绝。合法的安全级别有不认证不加密（noAuthNoPriv）、既认证又加密（authPriv）和认证但不加密（authNoPriv）三种。请求类型有读（read）、写（write）、通告（notify）三种。MIB 视图由一个 MIB 视图名来标识。

输入：组名，上下文名，安全模型，安全级别，请求类型，MIB 视图

输出：配置的访问控制关系只在访问控制时使用，一般不对外输出。但是，可以通过 MIB 和命令行查看到现有已经配置好的访问控制关系。

4.3.4 团体名到安全名的映射

团体名到安全名的映射（Com2Sec）是 SNMPv3 的一种兼容 SNMPv1/v2 的安全机制。SNMPv3 是 SNMPv1/v2 的升级版本，需要对 SNMPv1/v2 向下兼容。但是，SNMPv1/v2 的安全机制比较差，如果仍然按照老的处理模式处理 SNMPv1/v2 的请求，将会降低系统的整体安全性。同时，VACM 需要使用安全名进行访问控制，但 SNMPv1/v2 消息没有该字段，而只有团体名（Community）字段。因此，SNMPv3 采用团体名到安全名的映射机制，通过将团体名和源地址映射为一个安全名，把 SNMPv1/v2 纳入到 SNMPv3 的访问控制中去，并规定 SNMPv1/v2 消息只能使用不认证不加密的安全级别，通过限制 SNMPv1/v2 的访问范围，来提高系统整体安全性。根据运行的网络是 IPv4 网络还是 IPv6 网络，分别提出了两种映射机制：Com2Sec 和 Com2Sec6。其中，团体名是 SNMPv1/v2 消息中的 Community 字段，最多 32 个字符。源地址是 Manager 的 IPv4 地址或 IPv6 地址。安全名的长

度最多为 32 个字符。

输入：团体名，源地址，安全名

输出：配置的团体名到安全名的映射只在访问控制时使用，一般不对外输出。但是，可以通过 MIB 和命令行查看到现有已经配置好的团体名到安全名的映射。

4.3.5 访问请求控制

VACM 在进行访问控制时，需要以下输入参数：安全模型（securityModel）、安全名（securityName）、安全级别（securityLevel）、视图类型（viewType）、上下文名（contextName）、被管对象 ID（由对象类型（object-type）和对象实例（object-instance）组成）。具体操作如下：

1. 在组映射表里，以{安全模型、安全名}作索引，查找与之对应的组名。若查找失败，则返回 noGroupName 出错信息。
2. 在上下文表中，查找是否存在由输入上下文名定义的上下文。若不存在，则返回 noSuchContext 出错信息。
3. 在访问控制表中，以{组名、上下文名、安全模型、安全级别}作索引，查找与之对应的 MIB 视图，包括读视图，写视图，通告视图。若查找失败，则返回 noAccessEntry 出错信息。
4. 根据视图类型，在第 3 步所查找到的 MIB 视图里选择与之对应的视图，并获得其视图名（viewName）。若查找失败，则返回 noSuchView 出错信息。
5. 在视图子树表中，以第 4 步得到的视图名作索引，获取该视图所包含或不包含的变量名的信息。
6. 结合第 5 步所获得的信息，判断由实体类型与实体实例组成的变量名（variableName）是否在该视图内。若不在，则返回 notInView 出错信息；反之则返回 accessAllowed 成功信息。

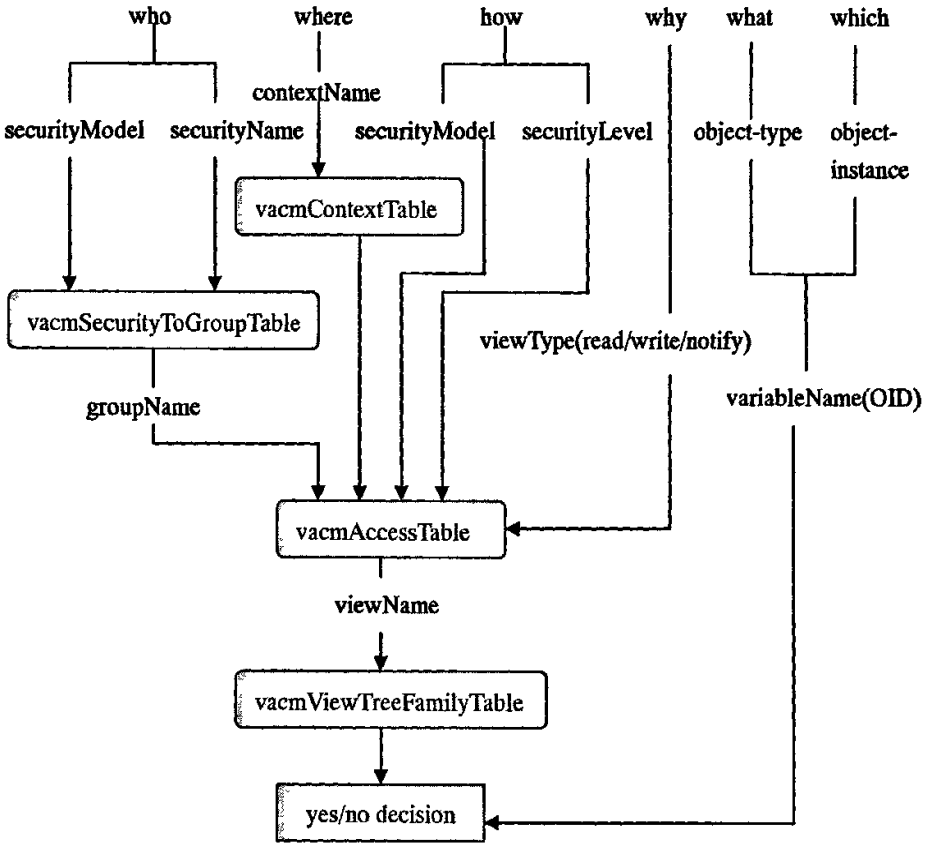


图 4-1 访问控制处理流程

4.4 外部功能设计

根据设备的实际应用特点以及 USM 和 VACM 模块在整个 SNMPv3 系统中与其它模块都有信息交互的特点，USM 和 VACM 模块的外部功能设计主要包括两方面内容：用户使用的命令行界面；在 SNMPv3 系统中，USM 和 VACM 模块与其它模块之间的软件接口方式。

4.4.1 用户界面

用户界面主要设计面向用户的相关命令行。基于 USM 和 VACM 的功能需求分析，设计了以下五类命令行：

- 引擎 ID 的配置

1. 配置 SNMP 的引擎标识 EngineID

   snmp-server engine-id <string>

2. 重置 SNMP 的引擎标识为系统自动生成

   no snmp-server engine-id

3. 显示 SNMP 的引擎标识

   show snmp-server engine-id

- USM 的配置

1. 配置不认证不加密或者只认证不加密的用户

   snmp-server user <username> {[md5|sha] <password1>}*1

2. 配置既认证又加密的用户

   snmp-server user <username> [md5|sha] <password1> [des] <password2>

3. 修改用户的认证密码

   snmp-server user change <username> {[md5|sha] <password1>}*1

4. 修改用户的认证密码和加密密码

   snmp-server user change <username> [md5|sha]

   <password1> [des] <password2>

5. 删除指定名字的用户

   no snmp-server user <username>

6. 显示所有用户或者显示指定名字的用户

   show snmp-server user {<username>}*1

- VACM 的配置

1. 将指定的用户加入 VACM 的指定组里

   snmp-server group <groupname> [v1|v2|usm] <username>

2. 删除指定组，或将某个安全模型和用户名对从组中删除

   no snmp-server group <groupname> {[v1|v2|usm] <securityname>}*1"

3. 显示所有的组或者指定名字的组

   show snmp-server group {<groupname>}*1

4. 添加 MIB 视图

   snmp-server view <viewname> <subtree> <mask> [included|excluded]

5. 删除 MIB 视图，或删除 MIB 视图下的某特定子树

no snmp-server view <viewname> {<subtree>}*1

6. 显示所有的 MIB 视图，或显示指定名字的视图

show snmp-server view {<viewname>}*1

7. 配置 VACM 访问控制权限

snmp-server access <groupname> [any|v1|v2|usm]
[noauthnopriv|authnopriv|authpriv] [readview|writeview] <viewname>

8. 删除 VACM 访问控制权限

no snmp-server access <groupname>
{[any|v1|v2|usm] [noauthnopriv|authnopriv|authpriv]}*1

9. 显示所有的 VACM 访问控制权限，或显示指定组的访问控制权限

show snmp-server access {<groupname>}*1

- Com2Sec 的配置

1. 配置 com2sec，将{团体名，源 IPv4 地址}二元组映射为安全名

snmp-server com2sec <username>
[<A.B.C.D>|<A.B.C.D/M>] <community>

2. 删除指定用户名的 com2sec，或删除指定用户名、IPv4 地址和 community 的 com2sec

no snmp-server com2sec <username>
{[<A.B.C.D>|<A.B.C.D/M>] <community>}*1

3. 显示所有的 com2sec，或显示指定用户名的 com2sec

show snmp-server com2sec {<username>}*

- Com2Sec6 的配置

1. 配置 com2sec6，将{团体名，源 IPv6 地址}二元组映射为安全名

snmp-server com2sec6 <username>
[<X:X:X:X::X>|<X:X:X: X::X/M>] <community>

2. 删除指定用户名的 com2sec6，或删除指定用户名、IPv6 地址和 community 的 com2sec6

no snmp-server com2sec6 <username>
{[<X:X:X:X::X>|<X:X:X:X::X/M>] <community>}*1

3. 显示所有的 com2sec6，或显示指定用户名的 com2sec6

show snmp-server com2sec6 {<username>}*1

4.4.2 软件接口

在SNMPv3系统中，USM和VACM模块与其它模块之间的通信采用直接函数调用来进行。为此，USM和VACM模块需提供必要的函数接口。

第五章 SNMPv3 安全新特性的实现

5.1 概述

遵循统一的网络管理机制并结合设备自身特点，将 SNMPv3 系统划分为初始化、USM、VACM、请求处理以及 Trap 五大主要模块。其中，初始化模块负责读取配置文件，注册 MIB 以及初始化重要的数据结构；USM 模块负责对消息进行认证，以及对 PDU 进行加密和解密；VACM 模块负责进行访问控制；请求处理模块集成了调度器和消息处理器的功能，负责消息的收发，消息的组织，并协调各个子系统之间的工作；Trap 模块负责产生并发送 Trap 消息。

USM 和 VACM 模块与其它三个模块都有信息交互，是整个 SNMPv3 系统安全通信的关键部分。本章叙述了 SNMPv3 系统的 USM 和 VACM 两项功能的详细设计和模块化实现流程，以及主要的数据结构、命令接口和函数接口的设计与实现。

5.2 USM 模块的软件实现

5.2.1 模块设计

5.2.1.1 模块结构

如下图所示，将 USM 模块划分为五个功能子模块：输入/输出子模块，权威引擎 ID 生成子模块，密钥生成子模块，认证子模块（包括身份认证子模块和时间窗校验子模块），以及加密/解密子模块。

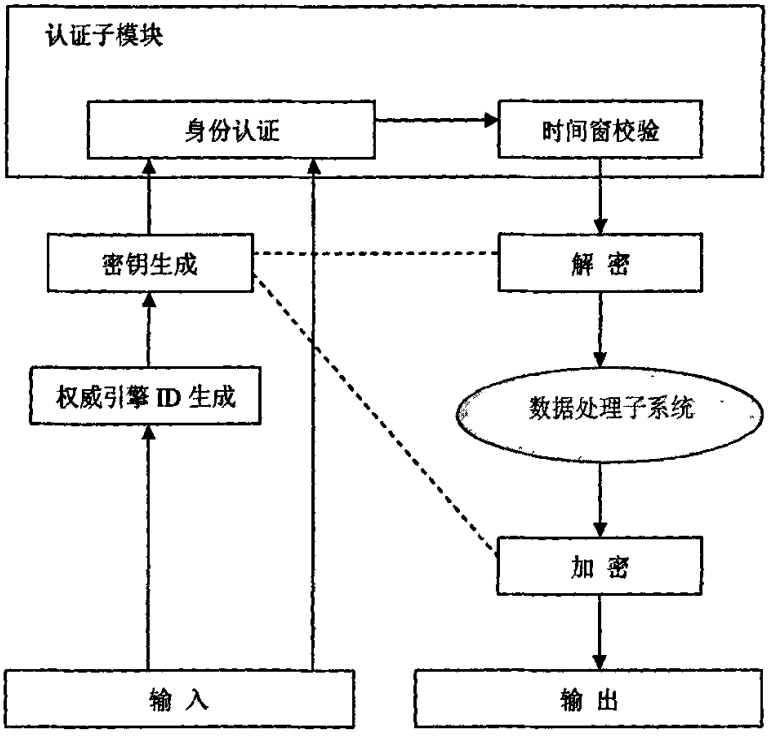


图 5-1 USM 模块逻辑结构图

5.2.1.2 模块描述

1. 输入/输出子模块

输入/输出子模块是USM模块与其它子系统交互的边界。其中，输入部分包括配置文件输入和数据包输入。由于Agent是权威引擎，其引擎ID生成子模块的输入是通过配置文件完成的；但对于非权威引擎，需要通过权威引擎发现过程来获得与之通信的权威引擎的引擎ID。

通过输入模块直接进入认证子模块的是SNMP消息。

输出有三种，通过屏幕输出，输出到文件，和发送SNMP消息到网络。前两种主要是为了辅助调试和观察其运行状态。通过网络发送SNMP消息是其对外的主要输出。

2. 权威引擎ID生成子模块

权威引擎ID生成子模块完成权威引擎ID的生成功能，其实现功能可分本地引擎ID的生成和远端权威引擎ID的发现。其输出的权威引擎ID用于生成认证密钥和加

密密钥。本课题所属的项目主要是开发Agent端的SNMPv3系统，且Agent在通信过程中一般均为权威引擎，所以其本地snmpEngineID即为权威引擎ID。

3. 密钥生成子模块

密钥生成子模块通过实现密钥本地化功能来完成认证密码和加密密码向认证密钥和加密密钥的转换过程。

4. 认证子模块

认证子模块包括了用于防止伪装和消息篡改的身份认证功能和防止消息被恶意滞延和重复发送的时间窗校验功能，以此来保证消息的可靠性与时效性。

5. 加密/解密子模块

包括加密模块和解密模块。数据处理子系统不属于USM模块，但是经USM模块解密所得的消息需通过数据处理子系统进行处理，此处可将该部分简单理解为一次函数调用。解密就是将经过认证的密文消息，在加密密钥的作用下通过解密算法将消息恢复成明文。加密是解密的逆过程。将待发送的明文消息，在加密密钥的作用下通过加密算法将其转变为密文，以实现消息的安全传递。本系统采用的加密密钥和解密密钥是相同的。

5.2.2 系统 I/O 的实现

根据 USM 模块的模块设计以及每个子模块完成的功能，确定各子模块所需要的具体输入输出数据。具体内容详见表 5-1 至表 5-14。

5.2.2.1 输入/输出子模块

表 5-1 输入模块的输入

数据名称	数据来源	说明
配置文件输入	配置文件	无
SNMP 消息	通过 socket API 获取	无

表 5-2 输出模块的输出

数据名称	数据去向	说明
回复的 SNMP 消息	通过 socket API 发送	无
屏幕输出	屏幕	可通过开关打开

日志记录	文件系统	可通过开关打开

5.2.2.2 权威引擎 ID 生成子模块

表 5-3 权威引擎 ID 生成模块的输入

数据名称	数据来源	说明
snmpEngineID 生成规则	输入模块，配置文件	MAC|IPV4|IPV6|text
EngineID 串	输入模块，配置文件	配合生成规则 text 使用，用于指定 snmpEngineID

表 5-4 权威引擎 ID 生成模块的输出

数据名称	数据去向	说明
snmpEngineID	密钥生成模块	输出的是参与通信的权威引擎的引擎 ID

5.2.2.3 密钥生成子模块

表 5-5 密钥生成模块的输入

数据名称	数据来源	说明
认证协议	配置文件	表明认证类型
snmpEngineID	引擎生成模块	本 ID 为权威引擎 ID
认证密码	配置文件	无
加密密码	配置文件	无

表 5-6 密钥生成模块的输出

数据名称	数据去向	说明
认证密钥	认证子模块	本地鉴权密钥
加密密钥	解密、加密子模块	本地加密/解密密钥

5.2.2.4 认证子模块

表 5-7 身份认证子模块的输入

数据名称	数据来源	说明
认证协议	配置文件	表明认证类型
认证密钥	密钥生成子模块产生的本地认证密钥	无

消息认证码	输入模块，来自从网络接收到的 SNMP 消息	用于校验消息是否可靠
wholeMsg	输入模块，从网络接收到的 SNMP 消息	在校验前需将 wholeMsg 中认证参数字段置零

表 5-8 身份认证子模块的输出

数据名称	数据去向	说明
认证结果	时间窗校验子模块	消息通过认证后，便送到时间窗校验子模块判断其时效性；否则就将其丢弃

表 5-9 时间窗校验子模块的输入

数据名称	数据来源	说明
msgAuthoritativeEngineID	输入模块，来自从网络接收到的 SNMP 消息	验证于本地 EngineID 是否相等
msgAuthoritativeEngineBoots	输入模块，来自从网络接收到的 SNMP 消息	无
msgAuthoritativeEngineTime	输入模块，来自从网络接收到的 SNMP 消息	无
snmpEngineID	引擎生成模块输出，本地 snmpEngineID	无
snmpEngineBoots	本地 snmpEngineBoots	无
snmpEngineTime	本地 snmpEngineTime	无

表 5-10 时间窗校验子模块的输出

数据名称	数据去向	说明
校验结果	解密模块	消息通过时效校验后就送到解密模块；否则就将其丢弃

5.2.2.5 加解密子模块

表 5-11 解密子模块的输入

数据名称	数据来源	说明
加密协议	配置文件	表明了加密类型
解密密钥	密钥生成模块	无
初始化向量	消息中的加密参数和本地加密密钥	无
密文	输入模块，来自从网络接收到	无

	的 SNMP 消息	

表 5-12 解密子模块的输出

数据名称	数据去向	说明
明文	数据处理子模块	无

表 5-13 加密子模块的输入

数据名称	数据来源	说明
加密协议	配置文件	表明加密类型
加密密钥	密钥生成模块	无
salt_integer64_1	随机生成	用于生成 salt
snmpEngineBoots	配置文件	用于生成 salt
明文	数据处理子模块	无

表 5-14 加密子模块的输出

数据名称	数据去向	说明
密文	输出模块输出	无
Salt	放入消息的加密参数字段，与密文一起发送	无

5.2.3 数据结构

数据结构的设计是保证软件实现的关键环节，针对 USM 模块所完成的安全功能，设计了 usmUser 和 usmStateReference 两个主要数据结构，以及相关的全局变量。

5.2.3.1 usmUser 的定义

数据结构 usmUser 存储用于身份认证与消息加解密的用户相关属性。将该结构体用链表链接起来便构成了 userList 链表，该链表对应于 USM 模块维护的用户信息表 usmUserTable。当在 userList 链表里进行查找时，以 engineID 和 secName 作为索引。

```
struct usmUser {
  u_char        *engineID;    /*该用户所对应的权威引擎的引擎 ID*/
  size_t        engineIDLen;  /*权威引擎 ID 的长度*/
```

```
  char            *name;        /*与 USM 相关的用户名*/
  char            *secName;     /*与安全模式无关的安全用户名，通常与 name 一样*/
  oid             *cloneFrom;
  size_t          cloneFromLen;
  oid             *authProtocol;  /*用户所使用的认证协议的 OID*/
  size_t          authProtocolLen;   /*认证协议的 OID 的长度*/
  u_char          *authKey;   /*用户的认证密钥*/
  size_t          authKeyLen;   /*认证密钥码的长度*/
  oid             *privProtocol;   /*用户所使用的加密协议的 OID*/
  size_t          privProtocolLen;   /*加密协议的 OID 的长度*/
  u_char          *privKey;   /*用户的加密密钥*/
  size_t          privKeyLen;   /*加密密钥的长度*/
  u_char          *userPublicString;
  int             userStatus;      /*该用户的信息在 usmUserTable 的状态*/
  int             userStorageType;    /*该用户的信息在 usmUserTable 的存储类型*/
  struct usmUser *next;
  struct usmUser *prev;
};
```

5.2.3.2 usmStateReference 的定义

数据结构 usmStateReference 用于存储用户的相关安全参数。当接收到一个 SNMP 请求消息时，就为发送该请求的用户创建一个 usmStateReference 来保存相关的安全参数。这些安全参数将在接收实体产生响应消息时使用。

```
struct usmStateReference {
  char            *usr_name;   /*用户名*/
  size_t          usr_name_length;   /*用户名的长度*/
  u_char          *usr_engine_id;   /*用户所对应的权威引擎的引擎 ID*/
  size_t          usr_engine_id_length;   /*引擎 ID 的长度*/
  oid             *usr_auth_protocol;   /*用户所使用的认证协议的 OID */
  size_t          usr_auth_protocol_length;   /*认证协议的 OID 的长度*/
  u_char          *usr_auth_key;   /*用户的认证密钥*/
  size_t          usr_auth_key_length;   /*认证密钥的长度*/
  oid             *usr_priv_protocol;   /*用户所使用的加密协议的 OID */
  size_t          usr_priv_protocol_length;  /*加密协议的 OID 的长度*/
  u_char          *usr_priv_key;  /*用户的加密密钥*/
  size_t          usr_priv_key_length;   /*加密密钥的长度*/
  u_int           usr_sec_level;   /*用户的安全级别*/
};
```

5.2.3.3 数据结构与消息的关系

SNMPv3 消息里包含了许多参数，其中有些参数与数据结构里的数据元素存在着一一对应关系。当实体接收到一个 SNMP 请求消息时，先根据消息里 msgAuthoritativeEngineID 和 msgUserName 字段的值查找链表 userList，从而获得该用户对应的 usmUser 结构体。再用得到的 usmUser 结构体里的 authProtocol 和 privProtocol 来验证输入的安全级别是否与配置的一致。如果一致，则将 usmUser 结构体里的相关信息复制到 usmStateReference 结构体里。接着，使用 usmUser 结构体里的 authProtocol 和 authKey，以及消息里的 msgAuthenticationParameters 来对整个消息进行认证。认证通过后，再使用消息里的 msgAuthoritativeEngineBoots 和 msgAuthoritativeEngineTime，以及本地的时钟记录来进行时间窗校验。通过时间窗校验后，先根据消息里的 msgPrivacyParameters，以及用户的加密密钥生成初始化矢量 IV；再按照（4.2.7）描述的解密步骤将密文解密成明文。

当产生响应 SNMP 消息时，将参考 usmStateReference 结构体里存储的安全参数来完成消息的构造。具体过程请参照（5.2.4.1）。

5.2.3.4 全局变量

为便于统一操作，USM 模块定义了两个全局变量，如表 5-15 所示。

表 5-15 USM 模块相关全局变量

变量定义	变量含义	使用说明
static unsigned char *engineID	本地引擎 ID	通常作为权威引擎 ID
static struct usmUser *userList	用户链表表头	有序链表，按 engineIDLen，engineID，name 从小到大排列

5.2.4 USM 安全处理主流程

在 SNMPv3 的安全通信中，当发送消息时，SNMPv3 消息处理模块会调用 USM 模块提供的 generateRequestMsg 或 generateResponseMsg 服务来生成发送消息；当接收消息时，SNMPv3 消息处理模块会调用 USM 模块提供的 processIncomingMsg 服务来处理接收消息。详见图 2-8。

5.2.4.1 发送消息的生成

当要发送请求或响应消息时，调用者将向 USM 模块提供参数：messageProcessingModel（SNMP 版本信息）、globalData（消息头）、maxMessageSize（消息最大长度）、securityModel（安全模型）、securityEngineID（权威引擎 ID）、securityName（用户安全名）、securityLevel（安全级别），以及 scopedPDU，若为响应消息，还需提供 securityStateReference（处理原请求时缓存的安全参数）。USM 模块将进行如下处理：

1. (a) 如果是响应消息，则在 cachedSecurityData 里查找出该用户的相关信息，并将 securityEngineID 设置为本地的 snmpEngineID。
   (b) 根据调用者提供的 securityName，在本地维护的 usmUserTable 里找到该用户的相关配置信息；如果查找失败，则返回 unknownSecurityName 出错信息。
2. 检查调用者提供的 securityLevel 是否与用户所配置的信息一致，如果不一致，则返回 unsupportedSecurityLevel 出错信息。
3. (a) 如果 securityLevel 指示该消息要进行加密保护，则调用 USM 模块提供的加密服务 **encryptData** 对 scopedPdu 进行加密。如果加密成功，则将返回的加密参数填入消息的 securityParameters 字段，并将加密后的 scopedPdu 作为消息的净负荷；如果加密失败，则返回 encryptionError 出错信息。
   (b) 如果 securityLevel 指示该消息不需要进行加密保护，则将消息的 securityParameters 字段置为空，并将明文的 scopedPdu 作为消息的净负荷。
4. 将调用者提供的 securityEngineID 填入消息的 msgAuthoritativeEngineID 字段。
5. 将 securityEngineID 所对应的 snmpEngineBoots 和 snmpEngineTime 值分别填入消息的 msgAuthoritativeEngineBoots 和 msgAuthoritativeEngineTime 字段。
6. 将 userName 填入消息的 msgUserName 字段。
7. (a) 如果 securityLevel 指示该消息需要进行认证，则调用 USM 模块提供的认证服务 **authenticateOutgoingMsg** 对整个消息进行认证。如果认证成功，则将认证所得的消息认证码填入消息的 msgAuthenticationParameters 字段；如果认证失败，则返回 authenticationFailure 出错信息。
   (b) 如果 securityLevel 指示该消息不需要进行认证，则将消息的 msgAuthenticationParameters 字段置为空。
8. 将安全处理后的消息，以及其长度返回给调用者。

5.2.4.2 接收消息的处理

当接收到消息时，调用者将向 USM 模块提供参数：messageProcessingModel（SNMP 版本信息）、maxMessageSize（消息最大长度）、securityParameters（接收到的消息的安全参数）、securityModel（接收到的消息的安全模型）、securityLevel（安全级别），wholeMsg（从网络上接收到的消息），以及 wholeMsgLength（消息的长度）。USM 模块将进行如下处理：

1. 检查 securityParameters 的格式是否正确，若不正确则返回 parseError 出错信息。
2. 将消息的 msgAuthoritativeEngineID 字段的值作为将要返回给调用者的 securityEngineID 值。同时，准备缓存安全参数的 cachedSecurityData 以及将应用这些安全参数的 securityStateReference。若 msgAuthoritativeEngineID 字段的值是接收引擎不知道的，则返回 unknownEngineID 出错信息。
3. 根据 msgUserName 和 msgAuthoritativeEngineID 在本地维护的 usmUserTable 里查找该用户的相关配置信息，若查找失败，则返回 unknownSecurityName 出错信息。
4. 校验调用者提供的 securityLevel 是否与用户所配置的信息一致。若不一致，则返回 unsupportedSecurityLevel 出错信息。
5. 如果 securityLevel 指示该消息进行了认证，则调用 USM 模块的认证服务 **authenticateIncomingMsg** 对消息进行认证。若认证失败，则返回 authenticationFailure 出错信息。
6. 对消息进行时限检查，判断其是否在时间窗内。若消息不在时间窗内，则返回 notInTimeWindow 出错信息。
7. (a) 如果 securityLevel 指示该消息进行了加密保护，则调用 USM 模块的解密服务 **decryptData** 对消息的净负荷进行解密以得到将返回给调用者的明文 scopedPdu。若解密失败，则返回 decryptionError 出错信息。
   (b) 如果消息没有进行加密保护，则消息的净负荷直接作为将返回给调用者的明文 scopedPdu。
8. 计算出 maxSizeResponseScopedPDU 的值。
9. 从 usmUserTable 里获得该用户的 securityName。
10. 将安全参数 msgUserName、usmUserAuthProtocol,、usmUserAuthKey、usmUserPrivProtocol 以及 usmUserPrivKey 存入 cachedSecurityData。
11. 将 securityEngineID、securityName、scopedPDU、maxSizeResponseScopedPDU,

以及 securityStateReference 返回给调用者。

5.2.5 USM 外部接口的实现

外部接口的实现主要包括用户命令接口和函数接口的实现。

5.2.5.1 命令接口

【命令格式】**snmp-server engine-id <string>**

【命令功能】配置 SNMP 的引擎标识 EngineID。

【默认状态】SNMP 引擎标识的默认值为系统自动生成

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
string	SNMP 的引擎标识	系统自动生成

【使用指导】管理员可以根据需要自行配置 SNMP 引擎标识。

【配置实例】在下面的例子中，SNMP 引擎标识被配置为 A3B4C5D6。

```
HammerOS(config)# snmp-server engine-id A3B4C5D6
```

【命令格式】**no snmp-server engine-id**

【命令功能】重置 SNMP 的引擎标识为系统自动生成。

【默认状态】SNMP 引擎标识的默认值为系统自动生成

【命令模式】配置模式

【参数说明】无

【使用指导】管理员可以根据将 SNMP 引擎标识重置为系统自动生成。

【配置实例】在下面的例子中，SNMP 引擎标识被重置为系统自动生成。

```
HammerOS(config) #no snmp-server engine-id
```

【命令格式】**show snmp-server engine-id**

【命令功能】显示 SNMP 的引擎标识。

【默认状态】SNMP 引擎标识的默认状态为系统自动生成

【命令模式】配置模式

【配置实例】下面的例子显示 SNMMP 的引擎标识。

```
HammerOS(config)# show snmp-server engine-id
```

【相关命令】snmp-server engine-id

【命令格式】**snmp-server user <username> {[md5|sha] <password1>}*1**

【命令功能】为系统配置不认证不加密或者只认证不加密的用户

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
md5|sha	采用的认证算法为 hmac-md5 还是 hmac-sha	无
password1	认证密码	无

【使用指导】可以通过这条命令配置不认证不加密的用户，也可以配置只认证不加密的用户。

【配置实例】下面的例子分别配置不认证不加密的用户和采用 hmac-md5 算法进行认证但不加密的用户。

```
HammerOS(config)# snmp-server user    userNOAuth
HammerOS(config)# snmp-server user    userAuth
                        md5    authPwd
```

【命令格式】**snmp-server user <username> [md5|sha] <password1> [des] <password2>**

【命令功能】为系统配置既认证又加密的用户

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
md5|sha	采用的认证算法为 hmac-md5 还是 hmac-sha	无
password1	认证密码	无
des	采用的加密算法为 des	无
password2	用于加密的密码	无

【使用指导】采用这条命令时，需要提供所有的参数。

【配置实例】下面的例子配置既认证又加密的用户。

```
HammerOS(config)# snmp-server user authPrivUser
                    md5  authPwd  des  privPwd
```

【命令格式】**snmp-server user change <username>**
{[md5|sha] <password1>}*1

【命令功能】修改用户的认证密码

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
md5|sha	指定采用的认证算法是 hmac-md5 还是 hmac-sha	无
password1	认证密码	无

【使用指导】可以通过这条命令可以修改只认证不加密用户的认证密码。

【配置实例】下面的例子将采用 hmac-md5 认证的用户的认证密码改为 authPwd2。

```
HammerOS(config)# snmp-server user change userAuth
                    md5 authPwd2
```

【命令格式】**snmp-server user change <username>**
[md5|sha] <password1> [des] <password2>

【命令功能】修改用户的认证密码和加密密码

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
md5|sha	采用的认证算法为 hmac-md5 还是 hmac-sha	无

password1	认证密码	无
des	采用的加密算法为 des	无
password2	用于加密的密码	无

【使用指导】采用这条命令时，需要提供所有的参数。

【配置实例】下面的例子将用户的认证密码修改为 authPwd2，加密密码修改为 privPwd2。

```
HammerOS(config)# snmp-server user authPrivUser
                    md5 authPwd2 des privPwd2
```

【命令格式】**no snmp-server user <username>**

【命令功能】删除指定名字的用户

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无

【使用指导】通过这条命令可以删除指定的用户。

【配置实例】下面的例子首先创建了一个不鉴权不加密的用户，然后将其删除。

```
HammerOS(config)# snmp-server user noauthUser
HammerOS(config)# no snmp-server user noauthUser
```

【命令格式】**show snmp-server user {<username>}*1**

【命令功能】显示所有用户或者显示指定名字的用户

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无

【使用指导】通过这条命令可以显示所有用户的信息，也可以显示指定名字的用户的信息。

【配置实例】下面的例子显示所有用户的信息。

HammerOS(config)# show snmp-server user

5.2.5.2 函数接口

- 处理接收消息

函数原型：**int usm_process_in_msg(**

```
int msgProcModel,      /* IN    消息的版本号(UNUSED) */
size_t maxMsgSize,     /* IN    支持的最大消息长度*/
u_char * secParams,    /* IN    指向安全参数的指针*/
int secModel,          /* IN    安全模型*/
int secLevel,          /* IN    安全级别*/
u_char * wholeMsg,     /* IN    指向接收到的 v3 消息的指针*/
size_t wholeMsgLen,    /* IN    接收到的 v3 消息的长度*/
u_char * secEngineID,  /* OUT   指向 snmpEngineID 的指针*/
size_t * secEngineIDLen,  /* IN/OUT  指向 snmpEngineID 的长度的指针*/
char *secName,         /* OUT   指向 securityName 的指针*/
size_t * secNameLen, /* IN/OUT  指向 securityName 的长度的指针*/
u_char ** scopedPdu,   /* OUT   指向明文 scopedPdu 的指针*/
size_t * scopedPduLen,    /* IN/OUT 指向明文 scopedPdu 的长度的指针*/
size_t * maxSizeResponse,     /* OUT   指向响应 PDU 的最大长度的指针*/
void **secStateRf,     /* OUT   指向安全状态引用的指针 */
netsnmp_session * sess,    /* IN    指向接收该消息的 session */
u_char msg_flags   /* IN    消息的 msgFlag 字段*/)
```

返回值说明：

SNMPERR_SUCCESS

SNMPERR_USM_GENERICERROR

SNMPERR_USM_UNKNOWNSECURITYNAME

SNMPERR_USM_UNSUPPORTEDSECURITYLEVEL

SNMPERR_USM_ENCRYPTIONERROR

SNMPERR_USM_AUTHENTICATIONFAILURE

SNMPERR_USM_PARSEERROR

SNMPERR_USM_UNKNOWNENGINEID

SNMPERR_USM_NOTINTIMEWINDOW

SNMPERR_USM_DECRYPTIONERROR

功能说明：处理接收到的 SNMPv3 消息

处理流程：

1. 调用函数 **usm_parse_security_parameters()**提取消息里的安全参数，包括 AuthoritativeEngineID，AuthoritativeEngineBoots，AuthoritativeEngineTime，UserName，PrivacyParameters，AuthenticationParameters。
2. 根据获取的安全参数设置安全状态引用 secStateRef 里的的 uer_name，user_engine_id，uer_sec_level 的值。
3. 验证 secEngineID 是否存在，并记录对应的 EngineTime 和 EngineBoots。
4. 用 secEngineID 和 secName 在 userList 里查找对应的 usmUser，通过调用函数 **usm_get_user_from_list()**来完成。
5. 调用函数 **usm_check_secLevel()**来检查传入的 secLevel 是否与用户的配置信息一致。
6. 使用用户的 authProtocol，authKey 和前面提取出的 AuthenticationParameters 对整个消息进行认证，通过调用函数 **sc_check_keyed_hash()**来完成。
7. 使用用户的 authProtocol，authKey，privProtocol，privKey 来设置安全状态引用 secStateRef 里的 uer_auth_protocol，uer_auth_key，uer_priv_protocol，uer_priv_key 的值。
8. 调用函数 **usm_check_and_update_timeline()**对消息进行时间窗校验。
9. 调用函数 asn_parse_sequence()对密文进行检查。
10. 通过 privKey 和提取出的 PrivacyParameters 生成解密用的初始化矢量 IV。
11. 调用函数 **sc_decrypt()**对接收到的密文进行解密，解密所得的明文放在输出参数*scopedPdu 中。

- 处理外发消息

函数原型：**int usm_generate_out_msg(**

```
int msgProcModel,     /* IN    消息的版本号(UNUSED) */
u_char * globalData,        /* IN   指向消息的头部数据的指针*/
size_t globalDataLen,       /* IN   消息的头部数据的长度*/
int maxMsgSize,     /* IN    支持的最大消息长度*/
int secModel,       /* IN    安全模型*/
u_char * secEngineID,       /* IN   指向 snmpEngineID 的指针*/
size_t secEngineIDLen,      /* IN   SnmpEngineID 的长度 */
char *secName,      /* IN   指向 securityName 的指针*/
size_t secNameLen,  /* IN   SecurityName 的长度 */
int secLevel,       /* IN   安全级别*/
```

```
u_char * scopedPdu,      /* IN  指向消息的 scopedPdu 的指针*/
size_t scopedPduLen,     /* IN  scopedPdu 的长度 */
void *secStateRef,   /* IN 指向安全状态引用的指针*/
u_char * secParams,      /* OUT  安全参数 */
size_t * secParamsLen,     /* IN/OUT  安全参数的长度 */
u_char ** wholeMsg,      /* OUT  指向产生的待发送消息的指针*/
size_t * wholeMsgLen   /* IN/OUT 待发消息的长度 */)
```

返回值说明：

SNMPERR_SUCCESS

SNMPERR_USM_AUTHENTICATIONFAILURE

SNMPERR_USM_ENCRYPTIONERROR

SNMPERR_USM_GENERICERROR

SNMPERR_USM_UNKNOWNSECURITYNAME

SNMPERR_USM_GENERICERROR

SNMPERR_USM_UNSUPPORTEDSECURITYLEVEL

功能说明：生成外发消息

处理流程：

1. 如果是生成响应消息，则从传入的安全状态引用 secStateRef 中获取用户的相关安全参数。如果是生成请求消息，则调用函数 **usm_get_user()**，根据 secEngineID，secName 来查找对应的 usmUser，以获取该用户的相关信息。
2. 调用函数 **usm_check_secLevel_vs_protocols()**来校验传入的安全级别是否与用户的配置信息一致。
3. 根据 EngineID 获取 EngineBoots 和 EngineTime 的值。
4. 调用函数 **usm_calc_offsets()**来设置消息里每个字段的偏移量。
5. 将指针 wholeMsg 指向生成消息的开始地址。
6. 若需要对消息进行加密保护，则调用函数 **usm_set_salt()**计算用于生成初始化矢量的 salt 的值，并将该 salt 填入消息的 msgPrivacyParameters 字段。
7. 调用函数 **sc_encrypt()**对明文的 scopedPdu 进行加密，并将密文置入消息的净负荷字段。
8. 根据第 4 步计算出的各字段偏移量，以及传入的相关参数，向消息的如下字段：

msgAuthoritativeEngineID，msgAuthoritativeEngineBoots，msgAuthoritativeEngineTime，msgUserName，填入相应的值。

9. 若需要对消息进行认证，则调用函数 sc_generate_keyed_hash()对消息进行认证，并将所得的消息认证码填入消息的 msgAuthenticationParameters 字段。

- 创建用户

函数原型：**void usm_parse_create_usmUser(**

const char *token, /*目前暂不使用，为 NULL */

char *line /*配置文件中行指针*/)

返回值说明：无

功能说明：通过命令行创建新用户

处理流程：

1. 调用函数 **usm_create_user()**创建缺省空用户。
2. 根据指针 line 所指向的配置文件信息来初始化 usmUser 的各数据元素。
   (a) 若没有配置 engineID，则调用函数 **snmpv3_generate_engineID()**来自动生成本地引擎 ID。若配置了，则根据配置文件来设置用户的 engineID。
   (b) 调用函数 **generate_Ku()**和 **generate_Kul()**来完成密钥的本地化。
3. 调用函数 **usm_add_user()**将创建的用户信息 usmUser 添加到 userList 链表里。

- 删除用户

函数原型：**int usm_remove_named_user(**

char* usrname /*被删除用户的用户名*/)

返回值说明：删除的用户数目

功能说明：用命令行删除指定用户

处理流程：

查找并删除 userList 链表中所有用户名为 usrname 的 usmUser。

5.3 VACM 模块的软件实现

5.3.1 数据结构

VACM 模块需使用三个链表：组映射链表（groupList）、访问控制权限链表（accessList）、视图链表（viewList）。其链表元素分别为结构体 vacm_groupEntry，vacm_accessEntry，vacm_viewEntry。

根据设备的应用特点，规定 Manager 关于某个 Agent 能访问的所有被管对象都在该 Agent 本地，即一个上下文包含了某个实体所有的可访问被管对象，因此就不需要实现上下文（Context），进而对消息里的 ContextEngineID 字段和 ContextName 字段也不需作任何处理。

针对 VACM 模块所完成的用户管理功能，设计了 vacm_groupEntry、vacm_accessEntry、vacm_viewEntry、com2SecEntry、com6SecEntry 等五个主要数据结构，以及相关的全局变量。

5.3.1.1 vacm_groupEntry 的定义

数据结构 vacm_groupEntry 记录了{安全模型，安全名}二元组与组名的映射关系。将该结构体用链表链接起来便构成了 groupList 链表，该链表对应于 VACM 模块维护的组映射表 vacmSecurityToGroupTable。当在 groupList 链表里进行查找时，以 securityModel 和 securityName 作为索引。

```
struct vacm_groupEntry {
  int       securityModel;          /*安全模型*/
  char      securityName[VACMSTRINGLEN];    /* 安全名(用户名)，首位为长度*/
  char      groupName[VACMSTRINGLEN];       /* 组名 */
  int       storageType;  /* 存储方式 */
  int       status;     /* 状态 */
  struct vacm_groupEntry   *reserved;
  struct vacm_groupEntry   *next;
};
```

5.3.1.2 vacm_accessEntry 的定义

数据结构 vacm_accessEntry 存储了某特定组的访问权限。将该结构体用链表链接起来便构成了 accessList 链表，该链表对应于 VACM 模块维护的访问控制表

vacmAccessTable。当在 accessList 链表里进行查找时，以 groupName，contextPrefix，securityModel 和 securityLevel 作为索引。

```
struct vacm_accessEntry {
  char          groupName[VACMSTRINGLEN];     /* 组名，首位为长度*/
  char          contextPrefix[VACMSTRINGLEN];  /*Context 前缀，首位为长度*/
  int           securityModel;          /* 安全模型 */
  int           securityLevel;          /* 安全级别 */
  int           contextMatch;           /* 匹配方式[exact/prefix]    */
  char          readView[VACMSTRINGLEN];       /* 读视图     */
  char          writeView[VACMSTRINGLEN];      /* 写视图     */
  char          notifyView[VACMSTRINGLEN];     /* Notify 视图*/
  int           storageType;      /* 存储方式 */
  int           status;           /* 状态 */
  struct vacm_accessEntry *reserved;
  struct vacm_accessEntry *next;
};
```

5.3.1.3 vacm_viewEntry 的定义

数据结构 vacm_viewEntry 定义了一个 MIB 视图的相关信息。将该结构体用链表链接起来便构成了 viewList 链表，该链表对应于 VACM 模块维护的视图子树表 vacmViewTreeFamilyTable。当在 viewList 链表里进行查找时，以 viewName 和 viewName 作为索引。

```
struct vacm_viewEntry {
  char      viewName[VACMSTRINGLEN];     /* 视图名，首位为长度 */
  oid       viewSubtree [MAX_OID_LEN];       /* 子树，首位为长度 */
  size_t    viewSubtreeLen;       /* 子树长度+ 1 */
  u_char    viewMask[VACMSTRINGLEN];       /* 子树掩码 ，二进制方式存储*/
  size_t    viewMaskLen;        /* 子树掩码长度 */
  int       viewType;           /* 视图类型[included/excluded]  */
  int       viewStorageType;    /* 存储方式 */
  int       viewStatus;         /* 状态 */
  struct vacm_viewEntry *reserved;
  struct vacm_viewEntry *next;
};
```

5.3.1.4 com2SecEntry 的定义

数据结构 com2SecEntry 定义了{团体名，源 IPv4 地址}与安全名的映射关系。将该结构体用链表链接起来便构成了 com2SecList 链表，用于存储所有已配置的

com2Sec 的信息。

```
typedef struct _com2SecEntry {
    char            community[VACMSTRINGLEN];   /*团体名*/
    unsigned long   network;  /* 子网 IPv4 地址*/
    unsigned long   mask;  /*子网掩码*/
    char            secName[VACMSTRINGLEN];  /*安全名*/
    struct _com2SecEntry *next;
} com2SecEntry;
```

5.3.1.5 com2Sec6Entry 的定义

数据结构 com2Sec6Entry 定义了{团体名，源 IPv6 地址}与安全名的映射关系。将该结构体用链表链接起来便构成了 com2Sec6List 链表，用于存储所有已配置的 com2Sec6 的信息。

```
typedef struct _com2Sec6Entry {
    char            community[VACMSTRINGLEN];  /*团体名*/
    struct sockaddr_in6 network;  /*子网 IPv6 地址*/
    struct sockaddr_in6 mask;  /*子网掩码*/
    char            secName[VACMSTRINGLEN]; /*安全名*/
    struct _com2Sec6Entry *next;
} com2Sec6Entry;
```

5.3.1.6 全局变量说明

为便于统一操作，VACM 模块定义了如表 5-16 所示的全局变量。

表 5-16 VACM 模块主要全局变量

变量定义	变量含义	使用说明
static struct vacm_viewEntry *viewList	View 链表头指针	无
static struct vacm_accessEntry *accessList	Access 链表头指针	无
static struct vacm_groupEntry *groupList	Group 链表头指针	无
static struct vacm_viewEntry *viewScanPtr	View 链表访问指针	用于查找链表
static struct vacm_accessEntry *accessScanPtr	Access 链表访问指针	用于查找链表
static struct vacm_groupEntry *groupScanPtrt	Group 链表访问指针	用于查找链表
int snmp_view_number	View 链表长度	无
int snmp_access_number	Access 链表长度	无
int snmp_group_number	Group 链表长度	无
com2SecEntry *com2SecList	Com2sec 链表头指针	无

com2Sec6Entry *com2Sec6List	Com2sec6 链表头指针	无

5.3.2 VACM 访问控制主流程

VACM 访问控制流程如图 5-2 所示：

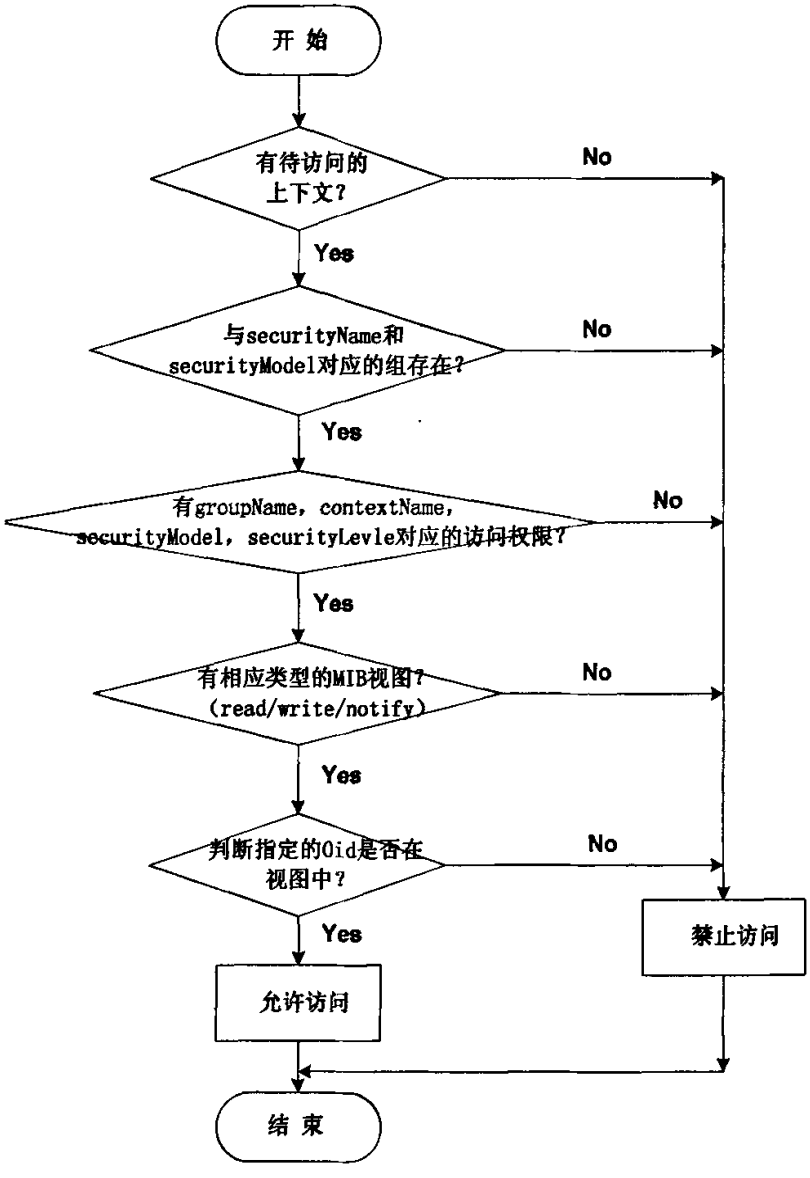


图 5-2 VACM 访问控制流程

5.3.3 系统 I/O 的实现

5.3.3.1 输入

当 VACM 对 SNMP 请求进行访问控制处理时，有两方面输入：

1. 配置的访问控制策略，通过查找 groupList，accessList 和 viewList 三个链表可获得某具体用户的访问控制权限。
2. 调用该访问控制服务的调用者需提供的输入参数：
   （1） contextName：上下文名称，默认为空字符串。
   （2） securityModel：安全模型，由管理站用户指定，明文传送。
   （3） securityName：安全名。
   （4） securityLevel：安全级别，是否进行认证和加密，明文传送。
   （5） 请求类型：由 PDU 类型确定，有三种类型：read/write/notify。
   （6） 请求对象：被管对象 ID。

5.3.3.2 输出

根据上述两方面的输入信息，执行 5.3.2 小节描述的访问控制过程，查找到一个 MIB 视图，然后查看所要访问的对象是否在这个 MIB 视图之内，如果在，则允许访问并返回成功信息；反之则拒绝访问并返回错误原因。输出的返回值有如下几种：

accessAllowed	：允许被访问
notInView	：查找到 MIB 视图，但是被访问对象不在该视图里
noSuchView	：没有查找到 MIB 视图
noGroupName	：没有查找到组名
noAccessEntry	：没由查找到访问控制权限
otherError	：其它错误

5.3.4 VACM 外部接口的实现

外部接口的实现主要包括用户命令接口和函数接口的实现。

4.3.4.1 命令接口

【命令格式】**snmp-server group <groupname> [v1|v2|usm] <username>**
【命令功能】将指定的用户加入 VACM 的指定组里
【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
groupname	组名	无
v1	采用 SNMPv1 的安全模型	无
v2	采用 SNMPv2 的安全模型	无
usm	采用 SNMPv3 usm 安全模型	无
username	用户名	无

【使用指导】通过这条命令可以将用户和安全模型对与 VACM 组对应起来。

【配置实例】下面的例子首先创建了一个用户 myname，然后加入了一个组，组名为 groupName，安全模型为 usm。

HammerOS(config)# snmp-server user myname md5 authPwd

HammerOS(config)# snmp-server group groupName usm myname

【命令格式】**no snmp-server group <groupname>**

{[v1|v2|usm] < username >}*1"

【命令功能】删除指定组，或将某个{安全模型，用户名}对从组中删除。

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
groupname	组名	无
v1	采用 SNMPv1 的安全模型	无
v2	采用 SNMPv2 的安全模型	无
usm	采用 SNMPv3 USM 安全模型	无
username	用户名	无

【使用指导】通过这条命令可以删除指定的 VACM 组，也可以将某个安全模型和用户名对从组中删除。

【配置实例】下面的例子删除名字为 groupName 的组。

HammerOS(config)# no snmp-server group groupName

下面的例子将 SNMPv3 USM 安全模型和 myname 安全名对从 groupName 中删除。

```
HammerOS(config)# no snmp-server group groupName
                    usm myname
```

【命令格式】**show snmp-server group {<groupname>}*1**

【命令功能】显示 VACM 组

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
groupname	组名	无

【使用指导】通过这条命令可以显示所有的组或者指定名字的组。

【配置实例】下面的第一条命令显示所有组，第二条命令显示名字为 group11 的组。

```
HammerOS(config)# show snmp-server group
HammerOS(config)# show snmp-server group group11
```

【命令格式】**snmp-server view <viewname> <subtree>**
<mask> [included|excluded]

【命令功能】配置 MIB 视图

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
viewname	视图名	无
subtree	子树 OID	无
mask	子树掩码	无
included	包含该子树	无
excluded	不包含该子树	无

【使用指导】通过这条命令可以配置 MIB 视图的相关信息。

【配置实例】下面的例子首先创建了一个视图名为 view11 的 MIB 视图，子树 oid 为.1.3.6.1.2.1.1，子树掩码为 ff，模式为包含。

```
HammerOS(config)# snmp-server view view11 .1.3.6.1.2.1.1 ff included
```

【命令格式】**no snmp-server view <viewname> {<subtree>}*1**

【命令功能】删除 MIB 视图或 MIB 视图下的某个子树

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
viewname	视图名	无
subtree	子树 OID	无

【使用指导】通过这条命令可以删除 MIB 视图，也可以删除 MIB 视图下的某个子树。

【配置实例】下面的例子首先创建了一个名为 view11 的视图，然后将其删除。

```
HammerOS(config)# snmp-server view view11 .1.3.6.1.2.1.1 ff included
HammerOS(config)# no snmp-server view view11
```

下面的例子删除 view12 下的 1.3.6.1 的 subtree。

```
HammerOS(config)# no snmp-server view view12 1.3.6.1
```

【命令格式】**show snmp-server view {<viewname>}*1**

【命令功能】显示 MIB 视图

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
viewname	视图名	无

【使用指导】通过这条命令可以显示所有的 MIB 视图，也可以显示指定名字的

MIB 视图。

【配置实例】下面第一条命令显示所有的视图，第二条命令显示名字为 view11 的视图。

```
HammerOS(config)# show snmp-server view
HammerOS(config)# show snmp-server view   view11
```

【命令格式】snmp-server access <groupname> [v1|v2|usm] [noauthnopriv|authnopriv|authpriv] [readview|writeview] <viewname>

【命令功能】配置 VACM 访问控制权限

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
groupname	组名	无
any	任意的安全模型	无
v1	SNMPv1 安全模型	无
v2	SNMPv2 安全模型	无
usm	SNMPv3 USM 安全模型	无
noauthnopriv	不认证不加密	无
authnopriv	认证不加密	无
authpriv	既认证又加密	无
readview	可读视图	无
writeview	可写视图	无
viewname	MIB 视图名	无

【使用指导】通过这条命令可以添加 VACM 访问控制权限

【配置实例】下面的命令配置一个访问控制权限，组名为 group11，安全模型为 v2，不认证不加密，可读视图为 readview11

```
HammerOS(config)# snmp-server access group11 v2 noauthnopriv
                  readview readview11
```

【命令格式】**no snmp-server access <groupname>**
{[any|v1|v2|usm] [noauthnopriv|authnopriv|authpriv]}*1

【命令功能】删除 VACM 访问控制权限

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
groupname	组名	无
any	任意的安全模型	无
v1	SNMPv1 安全模型	无
v2	SNMPv2 安全模型	无
usm	SNMPv3 usm 安全模型	无
noauthnopriv	不认证不加密	无
authnopriv	认证不加密	无
authpriv	既认证又加密	无

【使用指导】通过这条命令可以删除指定组的所有访问控制权限，或该指定组在特定安全模式与安全级别下的访问控制权限。

【配置实例】下面的例子首先创建了一个 access，然后将其删除。

```
HammerOS(config)# snmp-server access group11 v2
                                noauthnopriv readview readview11
HammerOS(config)# no snmp-server access group11
```

【命令格式】**show snmp-server access {<groupname>}*1**

【命令功能】显示 VACM 访问控制权限

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
groupname	组名	无

【使用指导】通过这条命令可以显示所有的访问控制权限，也可以显示指定组的访问控制权限。

【配置实例】下面第一条命令显示所有的访问控制映射，第二条命令显示组名为 group11 的访问控制映射。

HammerOS(config)# show snmp-server access

HammerOS(config)# show snmp-server access group11

【命令格式】**snmp-server com2sec <username> [<A.B.C.D>|<A.B.C.D/M>] <community>**

【命令功能】配置 com2sec，将{团体名，源 IPv4 地址}二元组映射为安全名

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
A.B.C.D	允许访问 Agent 的 Manager 的 IPv4 地址	无
A.B.C.D/M	允许访问 Agent 的子网 IPv4 地址，M 为子网掩码	无
community	SNMPv1/v2 的团体名	无

【使用指导】通过这条命令可以将 SNMPv1/v2 的 community 映射为用户，增强安全性。

【配置实例】下面的例子添加了一个 com2sec，用户名为 myname，允许访问 agent 的子网 IP 为 10.21.1.16,子网掩码为 24 位，community 为 public。

HammerOS(config)# snmp-server com2sec myname

10.21.1.16/24 public

【命令格式】**no snmp-server com2sec <username>**

{[<A.B.C.D>|<A.B.C.D/M>] <community>}*1

【命令功能】删除指定用户名的 com2sec，或删除指定用户名、IPv4 地址和 community 的 com2sec

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
A.B.C.D	允许访问 Agent 的 Manager 的 IPv4 地址	无
A.B.C.D/M	允许访问 Agent 的子网 IPv4 地址，M 为子网掩码	无
community	SNMPv1/v2 的团体名	无

【使用指导】通过这条命令可以删除指定用户名的 com2sec，或删除指定用户名、IP 地址和 community 的 com2sec。

【配置实例】下面的例子删除了用户名为 myname，子网 IP 为 10.21.1.16,子网掩码为 24，community 为 public 的 com2sec。

```
HammerOS(config)# no snmp-server com2sec
                    myname 10.21.1.16/24 public
```

【命令格式】**show snmp-server com2sec {<username>}*1**

【命令功能】显示所有的 com2sec，或显示指定用户名的 com2sec

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无

【使用指导】通过这条命令可以显示所有的 com2sec，或显示指定用户名的 com2sec。

【配置实例】下面的命令显示所有的 com2sec。

```
HammerOS(config)# show snmp-server com2sec
```

【命令格式】**snmp-server com2sec6 <username>**
[<X:X:X:X::X>|<X:X:X: X::X/M>] <community>

【命令功能】配置 com2sec6，将{团体名，源 IPv6 地址}二元组映射为安全名

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
X:X:X:X::X	允许访问 Agent 的 Manager 的 IPv6 地址	无
X:X:X: X::X/M	允许访问 Agent 的子网 IPv6 地址，M 为子网掩码	无
community	SNMPv1/v2 的团体名	无

【使用指导】通过这条命令可以将 SNMPv1/v2 的 community 映射为用户，增强安全性。

【配置实例】下面的例子添加了一个 com2sec6，用户名为 myname，允许访问 agent 的 IPv6 地址为 10:21:1::16，community 为 public。

```
HammerOS(config)# snmp-server com2sec6   myname
                                10:21:1::16   public
```

【命令格式】**no snmp-server com2sec6 <username>**

{[<X:X:X:X::X>|<X:X:X:X::X/M>] <community>}*1

【命令功能】删除指定用户名的 com2sec6 或者删除指定用户名、IPv6 地址和 community 的 com2sec6

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无
X:X:X:X::X	允许访问 Agent 的 Manager 的 IPv6 地址	无
X:X:X: X::X/M	允许访问 Agent 的子网 IPv6 地址，M 为子网掩码	无
community	SNMPv1/v2 的团体名	无

【使用指导】通过这条命令可以删除指定用户名的 com2sec6，或者删除指定用户名、IPv6 地址和 community 的 com2sec6。

【配置实例】下面的例子删除了用户名为 myname，IPv6 地址为 10:21:1::16，community 为 public 的 com2sec6。

HammerOS(config)# no snmp-server com2sec6 myname
10:21:1::16 public

【命令格式】**show snmp-server com2sec6 {<username>}*1**

【命令功能】显示所有的 com2sec6 或显示指定用户名的 com2sec6

【默认状态】无

【命令模式】配置模式

【参数说明】

参数	参数说明	缺省配置
username	用户名	无

【使用指导】通过这条命令可以显示所有的 com2sec6，或显示指定用户名的 com2sec6。

【配置实例】下面的命令显示所有的 com2sec6。

HammerOS(config)# show snmp-server com2sec6

5.3.4.2 函数接口

- 访问控制处理

函数原型：**int vacm_in_view(**

```
netsnmp_pdu *pdu,     /*存储消息相关信息的 PDU 结构体*/
oid * name,           /* 被访问对象的 OID */
size_t namelen,       /* OID 长度 */
int check_subtree     /* 标志是否通过子树检查来查找匹配的视图项*/)
```

返回值说明：

VACM_SUCCESS

VACM_NOSECNAME

VACM_NOGROUP

VACM_NOACCESS

VACM_NOVIEW

VACM_NOTINVIEW

VACM_SUBTREE_UNKNOWN

功能说明：判断请求的被管对象是否允许被访问

处理流程：

1. 如果是 SNMPv1 或 SNMPv2 的 PDU，则根据是 IPv4 网络还是 IPv6 网络，分别调用函数 **netsnmp_udp_getSecName()**或 **netsnmp_udp6_getSecName()**来查

找对应的安全名。若查找失败，则返回错误码 VACM_NOSECNAME。

2. 如果是 SNMPv3 的 PDU，且通过调用函数 **find_sec_mod()**说明该 PDU 的安全模型没有被注册，则返回错误码 VACM_NOSECNAME。
3. 调用函数 **vacm_getGroupEntry()**，根据 PDU 里的安全名与安全模型在 groupList 链表里查找对应的 groupEntry，若查找失败，则返回错误码 VACM_NOGROUP。
4. 调用函数 **vacm_getAccessEntry()**，根据第 3 步获得的 groupEntry 中的组名，以及 PDU 里的上下文名、安全模型和安全级别在 accessList 链表里查找对应的 accessEntry，若查找失败，则返回错误码 VACM_NOACCESS。
5. 根据 SNMP 消息类型，从第 4 步获得的 accessEntry 获得对应的 MIB 视图名：
   (a) 若为 SNMP_MSG_GET、SNMP_MSG_GETNEXT、SNMP_MSG_GETBULK，则获取读视图名；
   (b) 若为 SNMP_MSG_SET，则获取写视图名；
   (c) 若为 SNMP_MSG_TRAP、SNMP_MSG_TRAP2、SNMP_MSG_INFORM，则获取通告视图名；
6. 如果输入参数 check_subtree 标志通过子树检查来查找对应的视图项，则调用函数 **vacm_checkSubtree()**来获取 viewEntry。若没有对应的子树，则返回错误码 VACM_SUBTREE_UNKNOWN；若被访问对象没包含在视图内，则返回错误码 VACM_NOTINVIEW；若被访问对象包含在视图内，则返回成功信息 VACM_SUCCESS。
7. 调用函数 **vacm_getViewEntry()**，根据第 5 步获取的视图名，被访问对象 ID 在 viewList 链表里查找对应的 viewEntry，若查找失败，则返回错误码 VACM_NOVIEW；若被访问对象没包含在查找到的视图内，则返回错误码 VACM_NOTINVIEW；若被访问对象包含在查找到的视图内，则返回成功信息 VACM_SUCCESS。

- 创建 viewEntry

函数原型：**struct vacm_viewEntry *vacm_createViewEntry(**

```
const char *viewName,   /*视图名*/
oid * viewSubtree,   /*子树 OID*/
size_t viewSubtreeLen /*子树 OID 的长度*/)
```

返回值说明：指向所创建的 viewEntry 的指针

功能说明：根据视图名和子树 ID 创建对应的 viewEntry

处理流程：

1. 创建结构体 vacm_viewEntry，并根据传入的参数对其进行初始化；
2. 将新创建的 viewEntry 按从小到大的顺序插入 viewList 链表里，并使全局变量 snmp_view_number 的值加 1。

相关函数：

(1) **struct vacm_viewEntry *vacm_getViewEntry**(const char *viewName,
oid * viewSubtree, size_t viewSubtreeLen, int mode)

(2) **void vacm_destroyViewEntry**(const char *viewName,
oid * viewSubtree, size_t viewSubtreeLen)

(3) **int vacm_destroyWholeView**(const char *viewName)

- 创建 groupEntry

函数原型：**struct vacm_groupEntry *vacm_createGroupEntry(**
int securityModel, /*安全模型*/
const char *securityName /*安全名*/)

返回值说明：指向所创建的 groupEntry 的指针

功能说明：根据安全模型和安全名创建对应的 groupEntry

处理流程：

1. 创建结构体 vacm_groupEntry，并根据传入的参数对其进行初始化；
2. 将新创建的 groupEntry 按从小到大的顺序插入 groupList 链表里，并使全局变量 snmp_group_number 的值加 1。

相关函数：

(1) **struct vacm_groupEntry *vacm_getGroupEntry**(int securityModel,
const char *securityName)

(2) **void vacm_destroyGroupEntry**(int securityModel,
const char *securityName)

(3) **int vacm_destroyWholeGroup**(const char *groupName)

- 创建 accessEntry

函数原型：**struct vacm_accessEntry *vacm_createAccessEntry(**

```
const char *groupName,   /*组名*/
const char *contextPrefix,   /*上下文前缀*/
int securityModel,   /*安全模型*/
int securityLevel    /*安全级别*/)
```

返回值说明：指向所创建的 accessEntry 的指针

功能说明：根据组名、上下文前缀、安全模型和安全名创建对应的 accessEntry

处理流程：

1. 创建结构体 vacm_accessEntry，并根据传入的参数对其进行初始化；
2. 将新创建的 accessEntry 按从小到大的顺序插入 accessList 链表里，并使全局变量 snmp_access_number 的值加 1。

相关函数：

(1) **struct vacm_accessEntry *vacm_getAccessEntry**(const char *groupName,
const char *contextPrefix, int securityModel, int securityLevel)

(2) **void vacm_destroyAccessEntry**(const char *groupName,
const char *contextPrefix,int securityModel, int securityLevel)

(3) **int vacm_destroyWholeGroupAccess**(const char *groupName)

第六章 SNMPv3 安全新特性的功能测试

本文通过测试整个 SNMPv3 系统来测试 USM 和 VACM 两大安全功能。采用的主要测试方法是通过使用报文解析软件来捕获 Manager 与 Agent 之间交互的 SNMP 消息，并通过分析消息的内容来验证是否正确实现了 USM 和 VACM 两大功能。

本文的测试数据和结果均取自实际的集成测试环境，成功验证了 USM 和 VACM 两大模块运行稳定。目前，本课题所属的项目所开发的 SNMPv3 系统软件已嵌入商用版本投放市场进行进一步的检验。

6.1 测试目标

验证 SNMPv3 安全新特性的实现，包括验证 USM 和 VACM 两大模块的功能。

6.2 测试环境与工具

本文使用的集成测试环境如图 6-1 所示，由安装了报文解析软件 Sniffer 和网管软件 MIB-Browser 的 PC 和支持 SNMPv3 的路由器组成。PC 作为网管工作站，即 Manager；路由器则作为被管设备，它运行着本项目开发的 SNMPv3 Agent 端软件。PC 通过路由器上的 console 口来对路由器进行管理，同时 PC 作为 Manager 的相关设置通过使用网管软件 MIB-Browser 来配置，路由器上的 Agent 端的相关设置则通过命令行界面进行配置。

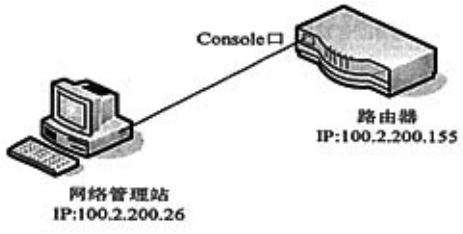


图 6-1 SNMPv3 安全新特性的集成测试环境

6.3 测试过程及测试结果

6.3.1 USM 功能测试

USM 功能测试主要测试在 noAuthNoPriv，authNoPriv 以及 authPriv 三种安全级别下进行 Get 操作的运行情况。

6.3.1.1 基于 noAuthNoPriv 的 USM 功能测试

1、配置路由器。

- 启动 SNMP。
- 配置 v3 用户 user_n。
- 将用户 user_n 添加到 group_n 组里。
- 配置 MIB 视图 test。
- 配置 group_n 组的安全级别为 noauthnopriv，读 MIB 视图为 test。

配置命令如图 6-2(a)所示。

```
HammerOS(config)#snmp-server enable
HammerOS(config)#snmp-server user user_n
HammerOS(config)#snmp-server group group_n v3 user_n
HammerOS(config)#snmp-server view test 1.3.6 e0 include
HammerOS(config)#snmp-server access group_n usm noauthnopriv readview test
```

图 6-2(a) 路由器配置命令

2、使用 MIB-Browser 软件配置网管工作站。

- 设置 SNMP 端口号为 161。
- 设置用户名为 user_u。
- 设置认证协议和加密协议为空。

配置界面如图 6-2(b)所示。

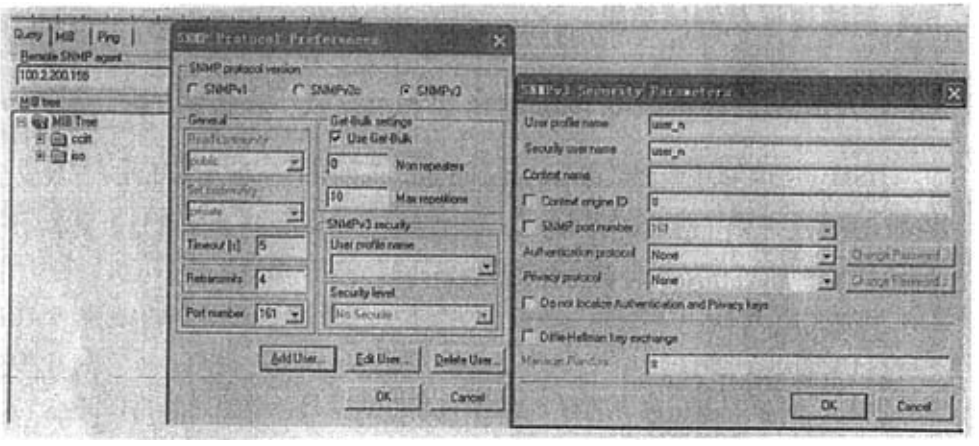


图 6-2(b) 网管工作站的属性设置

3、网管工作站对节点 1.3.6.1.2.1.1.5.0 进行 Get 操作。由 Sniffer 软件获取并解析的网管工作站发送给路由器的 SNMP 消息如图 6-2(c)所示。

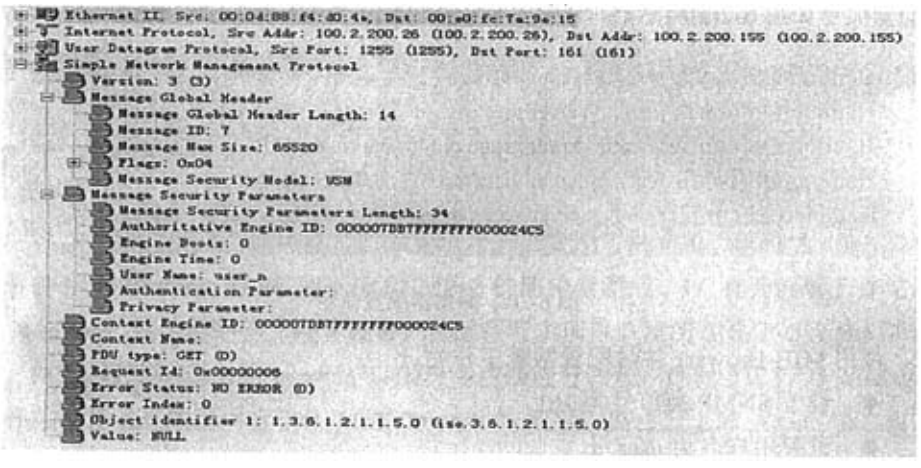


图 6-2(c) 网管工作站发送的 SNMP Get 消息

4、由 Sniffer 软件获取并解析的路由器返回给网管工作站的响应 SNMP 消息如图 6-2(d)所示。

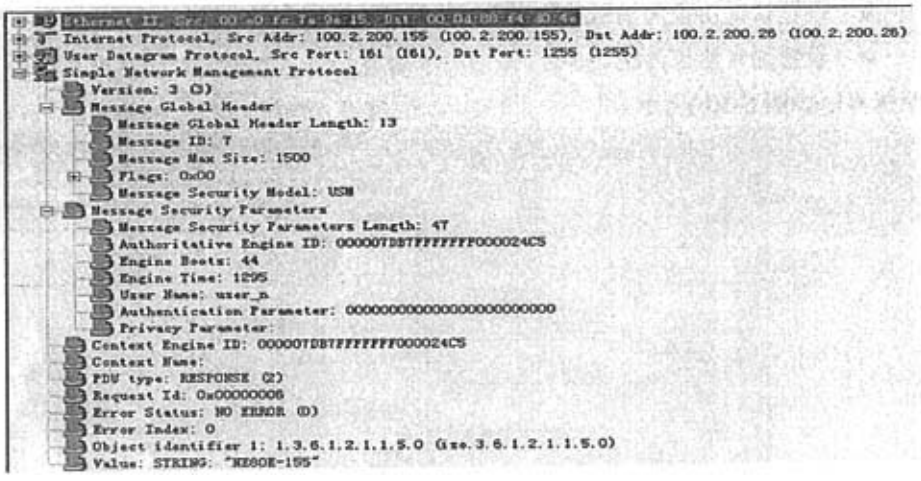


图 6-2(d) 路由器返回的 Response 消息

由上述的测试操作及测试结果可验证：在安全级别为 noAuthNoPriv 的情况下，USM 的功能运行正常。

6.3.1.2 基于 AuthNoPriv 的 USM 功能测试

1、配置路由器。

- 启动 SNMP。
- 配置 v3 用户 user_a，该用户使用 MD5 作为认证算法，认证密码为 123。
- 将用户 user_a 添加到 group_a 组里。
- 配置 MIB 视图 test。
- 配置 group_a 组的安全级别为 authnopriv，读 MIB 视图为 test。

配置命令如图 6-3(a)所示。

```
HammerOS(config)#snmp-server enable
HammerOS(config)#snmp-server user user_a md5 123
HammerOS(config)#snmp-server group group_a v3 user_a
HammerOS(config)#snmp-server view test 1.3.6 e0 include
HammerOS(config)#snmp-server access group_a usm authnopriv readview test
```

图 6-3(a) 路由器配置命令

2、使用 MIB-Browser 软件配置网管工作站。

- 设置 SNMP 端口号为 161。
- 设置用户名为 user_a。
- 设置认证协议为 HMAC-MD5，并设置认证密码为 123。
- 设置加密协议为空。

配置界面如图 6-3(b)所示。

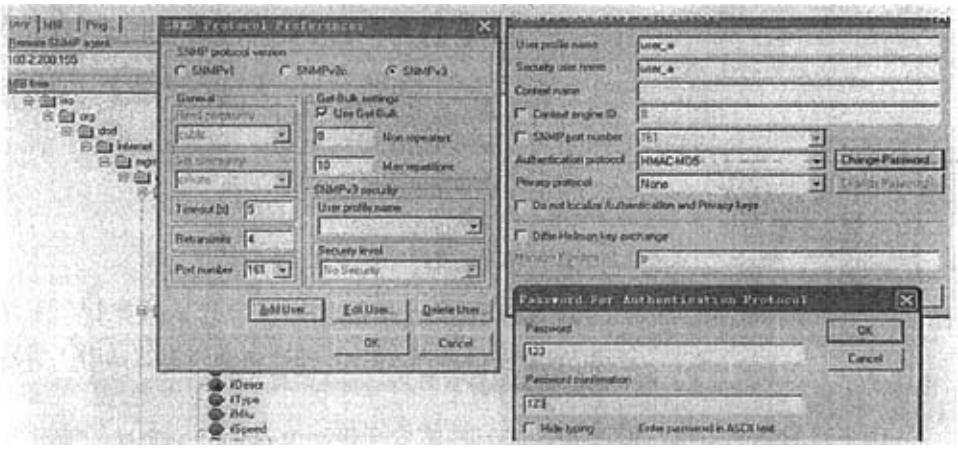


图 6-3(b) 网管工作站的属性设置

3、网管工作站对节点 1.3.6.1.2.1.1.5.0 进行 Get 操作。由 Sniffer 软件获取并解析的网管工作站发送给路由器的 SNMP 消息如图 6-3(c)所示。

图 6-3(c) 网管工作站发送的 SNMP Get 消息

4、由 Sniffer 软件获取并解析的路由器返回给网管工作站的响应 SNMP 消息如图

6-3(d)所示。

图 6-3(d) 路由器返回的 Response 消息

由上述的测试操作及测试结果可验证：在安全级别为 authNoPriv 的情况下，USM 的功能运行正常。

6.3.1.3 基于 authPriv 的 USM 功能测试

1、配置路由器。

- 启动 SNMP。
- 配置 v3 用户 user_p，该用户使用 MD5 作为认证算法，认证密码为 123；使用 DES 作为加密算法，加密密码为 123。
- 将用户 user_p 添加到 group_p 组里。
- 配置 MIB 视图 test。
- 配置 group_p 组的安全级别为 authpriv，读 MIB 视图为 test。

配置命令如图 6-4(a)所示。

```
HammerOS(config)#snmp-server enable
HammerOS(config)#snmp-server user user_p md5 123 des 123
HammerOS(config)#snmp-server group group_p v3 user_p
HammerOS(config)#snmp-server view test 1.3.6 e0 include
HammerOS(config)#snmp-server access group_p usm authpriv readview test
```

图 6-4(a) 路由器配置命令

2、使用 MIB-Browser 软件配置网管工作站。

- 设置 SNMP 端口号为 161。
- 设置用户名为 user_p。
- 设置认证协议为 HMAC-MD5，并设置认证密码为 123。
- 设置加密协议为 CBC-DES，并设置加密密码为 123。

配置界面如图 6-4(b)所示。

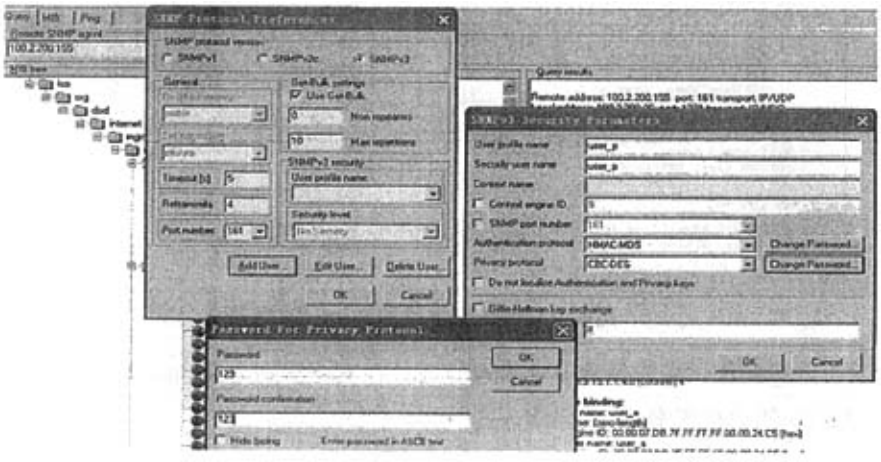


图 6-4(b) 网管工作站的属性设置

3、网管工作站对节点 1.3.6.1.2.1.1.5.0 进行 Get 操作。由 Sniffer 软件获取并解析的网管工作站发送给路由器的 SNMP 消息如图 6-4(c)所示。

图 6-4(c) 网管工作站发送的 SNMP Get 消息

4、由 Sniffer 软件获取并解析的路由器返回给网管工作站的响应 SNMP 消息如图

6-4(d)所示。

图 6-4(d) 路由器返回的 Response 消息

由上述的测试操作及测试结果可验证：在安全级别为 authPriv 的情况下，USM 的功能运行正常。

6.3.2 VACM 功能测试

为了验证 VACM 的访问控制功能，特意测试一个访问失败的情况，即当对一个不在该用户访问权限内的 MIB 节点进行 Get 操作，测试访问是否被拒绝。

1、配置路由器。

- 启动 SNMP.
- 配置 v3 用户 u1。
- 将用户 u1 添加到 g1 组里。
- 配置 MIB 视图 test。
- 配置 g1 组的安全级别为 noauthnopriv，读 MIB 视图为 test。

配置命令如图 6-5(a)所示。

```
HammerOS(config)#snmp-server enable
HammerOS(config)#snmp-server user u1
HammerOS(config)#snmp-server group g1 v3 u1
HammerOS(config)#snmp-server view test 1.3.6.1.4.1 fc include
HammerOS(config)#snmp-server access group_a usm noauthnopriv readview test
```

图 6-5(a) 路由器配置命令

2、使用 MIB-Browser 软件配置网管工作站。

- 设置 SNMP 端口号为 161。
- 设置用户名为 u1。
- 设置认证协议和加密协议为空。

配置界面如图 6-5(b)所示。

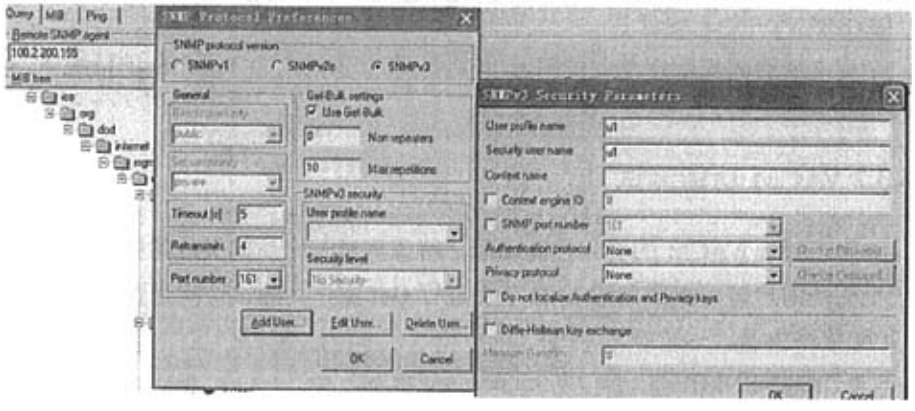


图 6-5(b) 网管工作站的属性设置

3、网管工作站对不在视图 test 里的节点 1.3.6.1.2.1.2.1.0 进行 Get 操作。由 Sniffer 软件获取并解析的网管工作站发送给路由器的 SNMP 消息如图 6-5(c)所示。

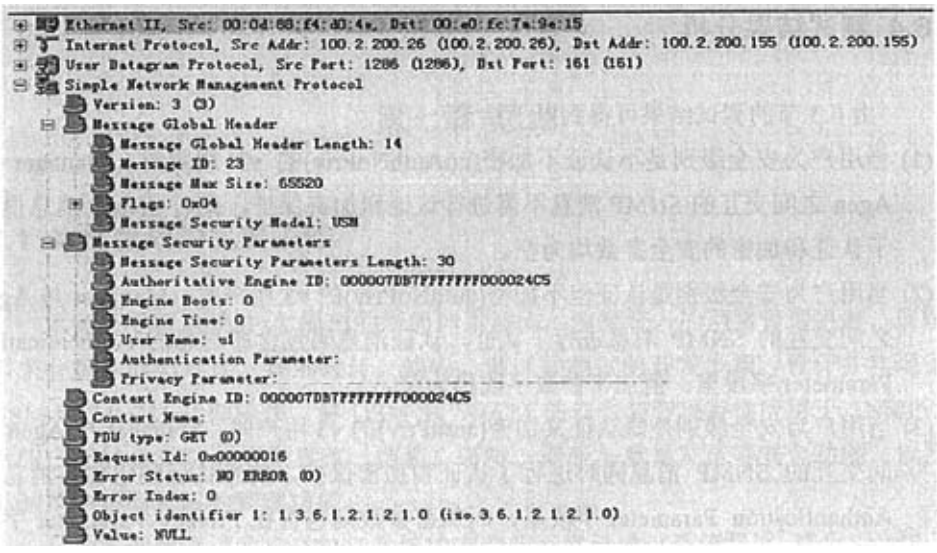


图 6-5(c) 网管工作站发送的 SNMP Get 消息

4、由 Sniffer 软件获取并解析的路由器返回给网管工作站的响应 SNMP 消息如图 6-5(d)所示。

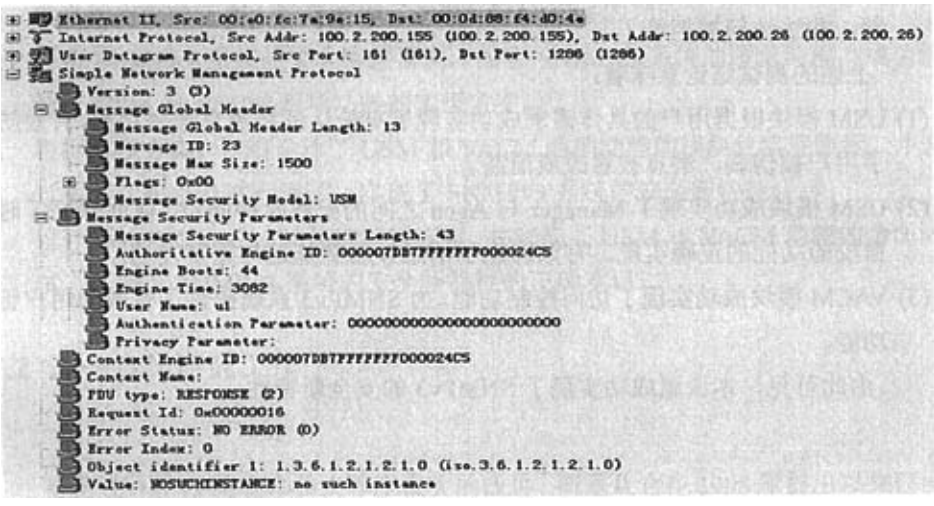


图 6-5(d) 路由器返回的 Response 消息

由上述的测试操作及测试结果可验证：VACM 成功实现了访问控制功能。

6.4 测试结果分析

由 6.3 节的测试结果可得到以下结论：

(1) 当用户为安全级别是不认证不加密(noAuthNoPriv)的 v3 用户时，Manager 与 Agen 之间交互的 SNMP 消息不需进行认证和加密保护，此时 SNMP 消息里用于认证和加密的安全参数均为空。

(2) 当用户为安全级别是认证但不加密(authNoPriv)的 v3 用户时，Manager 与 Agen 之间交互的 SNMP 消息进行了认证，认证消息码包含在消息的 Authentication Parameter 字段里；但加密参数字段仍为空。

(3) 当用户为安全级别是既认证又加密(authPriv)的 v3 用户时，Manager 与 Agen 之间交互的 SNMP 消息同时进行了认证和加密保护，消息认证码包含在消息的 Authentication Parameter 字段里，而加密参数则包含在 Privacy Parameter 字段里，且此时消息的 scopedPDU 部分是经加密后的密文。

(4) 通过观察 Manager 与 Agen 之间交互的 SNMP 消息里的 EngineBoots 和 EngineTime 字段，可验证实现了 Manager 与 Agen 之间的松散时钟同步。

(5) 当用户访问一个不在自己的访问权限内的被管对象时，通过 VACM 进行访问控制，非法访问被拒绝了。

上述的测试结论意味着：

(1) USM 模块根据用户的具体需求成功实现了身份认证和消息加密功能，有效防止了用户被伪装，消息被篡改或泄漏。

(2) USM 模块成功实现了 Manager 与 Agen 之间的松散时钟同步，以此保证了时间窗校验功能的正确实现，有效防止了消息被恶意延迟与重复发送。

(3) VACM 模块成功实现了访问控制功能，为 SNMPv3 系统提供了有效的用户管理功能。

由此可见，本课题成功实现了 SNMPv3 的安全新特性。

第七章 总结

7.1 本文完成的主要工作

本课题根据 SNMPv3 提出的新的体系结构，遵循统一的网络管理机制，通过需求分析、概要设计、详细设计、编码、调试并测试的开发步骤，设计并实现了 USM 和 VACM 功能模块，将 USM 和 VACM 的安全与管理特性应用于 SNMPv3 系统，使其具有防止信息篡改、伪装、窃听、滞延与重复发送等安全功能，以及控制用户访问权限的管理功能。

目前本课题完成的 SNMPv3 系统的安全新特性已通过系统测试，整个 SNMPv3 软件已运行于公司的相关产品上，获得用户好评。

本人作为课题的主研人员之一，独立承担并圆满完成了以下工作：

1. 深入分析 SNMPv3 的相关标准文档，对比 SNMPv3 与 SNMPv1、SNMPv2 在安全性方面的区别，准确把握了 USM 和 VACM 的功能特点。
2. 分析 SNMPv3 系统的安全特性需求，结合 SNMPv3 系统的逻辑结构，成功设计了 USM 和 VACM 两项功能的实现方案。
3. 根据实现方案，详细设计了 USM 和 VACM 两项功能的模块化实现流程，并通过编写代码以及进行调试，实现了 SNMPv3 系统的安全新特性。
4. 设计 USM 和 VACM 功能的测试方案，并完成了 USM 和 VACM 功能的全面测试，证明了 SNMPv3 系统的安全新特性的正确实现。

7.2 进一步的研究工作

虽然 SNMPv3 在安全性上作了很大的改进，但是从它在 1998 年提出以来已经过了九年，随着密码破译技术的发展，SNMPv3 也存在着如下安全隐患：

(1) DES 的安全性问题

DES 算法的弱点是保密强度不够，不能提供足够的安全性，加之其密钥长度为 56 位，使得通过应用高性能计算机便可在较短的时间内将其破译。此外，虽然 3DES 的密钥长度比 DES 提高了两倍，但由于 DES 算法本身保密强度较弱的先天

不足，使得 3DES 的保密性也不够强大。解决的方法是使用算法更好，其密钥长度更长的 AES。

(2) MD5 的安全性问题

MD5 的应用极其广泛，但是如今已经发现了很多碰撞，即不同的信息具有相同的哈希值。因此，可用 SHA-1 来代替 MD5 进行认证计算。然而，SHA-1 也不是绝对安全的，同样基于 HMAC 算法的具有 16 位哈希位的 MD5 被破译，也就意味着仅具有 20 位哈希位的 SHA-1 也存在着被破译的可能。解决的办法是采用具有更高哈希位数的算法，或者等待新的国际标准的确立。

此外，SNMPv3 完整系统的实现也需要进行更多的研究与实践。

由此可见，提供既能保持 SNMP 的简单性又同时能够提供强有力的安全与管理特性的 SNMPv3 完整系统的实现方案将是当前以及将来一段时间的主要研究课题。

致谢

在学位论文完成之际，首先要向我的导师雷维礼老师、毛玉明老师和马立香老师表示衷心的感谢，感谢三位老师对我的学术方面的培养和生活方面的帮助，感谢他们对我的鼓励与启迪以及对此论文提出的宝贵意见。他们严谨的治学态度和诲人不倦的良师风范，将时刻指引我今后的研究和个人发展方向。

同时也感谢港湾网络软件平台部的全体同事，以及本教研室的所有同学，感谢他们在学习和生活等各个方面给予的帮助和支持。

参考文献

[1] U. Blumenthal, B. Wijnen, "User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", STD 62, RFC3414, December 2002.

[2] Wijnen, B., Presuhn, R. and K. McCloghrie, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", STD 62, RFC3415, December 2002.

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

[4] D. Harrington, R. Presuhn, B. Wijnen, "An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks", STD 62, RFC3411, December 2002.

[5] J. Case, D. Harrington, R. Presuhn, B. Wijnen, "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", STD 62, RFC3412, December 2002.

[6] J. Case, M. Fedor, M. Schoffstall, J. Davin, "A Simple Network Management Protocol (SNMP)", RFC1157, May 1990.

[7] R. Presuhn, "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC3416, December 2002.

[8] M. Rose, " A Convention for Defining Traps for use with the SNMP", RFC1215, March 1991.

[9] J. Case, K. McCloghrie, M. Rose, S. Waldbusser, "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", RFC1905, January 1996.

[10] Blumenthal, U., Hien, N., and Wijnen, B. "Key Derivation for Network Management Applications." *IEEE Network*, May/June, 1997.

[11] Stallings, W. "SNMP and SNMPv2: The Infrastructure for Network Management." *IEEE Commun. Mag.*, March 1998.

[12] Stallings, W. SNMP, SNMPv2, SNMPv3, and RMON 1 and 2, Third Edition. Reading, MA: Addison-Wesley, 1998.

[13] Wijnen B, Harrington D, Presuhn R. An architecture for describing SNMP management frameworks[S]. RFC 2571, 1999.

[14] Blumenthal, U.; Hien, N.; and Wijnen, B. SNMPv3 Handbook. Reading, MA: Addison-Wesley,

1999.

[15] David T Perkins. SNMP Versions[S]. The Simple Times, 1997, 5(1) :13-14.

[16] Harrington D. The evolution of architectural concepts in the SNMPV3[J]. The Simple Times, 1997, 5(1) : 2-8.

[17] 程妍,卢泽新,冯艳玲.SNMPv1、SNMPv2c 和 SNMPv3 协议结构的分析与比较.现代计算机,2004.

[18] 张毅,王小非.SNMPv3 协议安全机制的研究.计算机与数字工程,2006.

[19] 林楚国,王勇,管军霖. DES 和 RSA 在基于 SNMPv3 的分布式网络管理系统中的应用.沿海企业与科技,2005

[20] 雷振甲.计算机网络管理及系统开发[M].北京:电子工业出版社,2002.

[21] 金鹏.采用 SNMPv3 的网管系统的性能分析.计算机应用,2004.

[22] 杨明.密码编码学与网络安全:原理与实践[M](第二版).北京:电子工业出版社,2002.

[23] 吴昊.基于 DES 算法和 RSA 算法的数据加密方案[J].郑州:焦作工学院学报,2002.

[24] 刘莊.公开密钥密码体制在 SNMPv3 安全模型中的应用研究[J].安徽:计算机与信息技术,2003.

攻硕期取得的研究成果

黄晓燕，女，汉族，生于1982年4月，籍贯四川省成都市。

教育背景

2004.7-2007.3　电子科技大学 通信与信息系统学科　　硕士学位

2000.9-2004.7　电子科技大学 通信学院 计算机通信专业　　学士学位

奖励情况

2005　研究生一等奖学金

基本技能

英语六级

计算机二级

发表论文

黄晓燕，雷维礼. 以太接入网技术. 通信与信息技术, 2004.

攻读硕士学位期间参加的科研项目

港湾网络(北京)有限公司，SNMPv3 系统的开发

华为技术有限公司成都研究所，OptiX2500+ 交叉与时钟集成单板的开发