CONCENTRIC SUPERVISION OF SECURITY APPLICATIONS: AN ASSURANCE METHODOLOGY

by

Philip C. Hyland
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
in Partial Fulfillment of
the Requirements for the Degree
of
Doctor of Philosophy
Information Technology

Committee:

_____ Dr. Ravi Sandhu, Dissertation Director

_____ Dr. Paul Ammann, Member

_____ Dr. J. Mark Pullen, Member

_____ Dr. S.C. Chang, Committee Chairman

_____ Dr. Stephen Nash, Associate Dean for Graduate Studies and Research Director

_____ Dr. Lloyd Griffiths, Dean of School of Information Technology and Engineering

Date: _____ Spring Semester 1999
George Mason University
Fairfax, Virginia

Concentric Supervision of Security Applications: An Assurance Methodology

A thesis submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University.

By

Philip C. Hyland
Master of Science
George Mason University. 1987

Director: Dr. Ravi Sandhu, Professor
Department of Information and Software Systems Engineering

Spring Semester 1999
George Mason University
Fairfax, Virginia

ACKNOWLEDGEMENTS

I am proud to acknowledge the generous and enduring support of my wife, Dr. Munja Cha-Hyland, throughout the years of effort toward this degree. Not only has her faith in me been unfailing, but her feedback and gentle advice have continuously challenged me to accomplish my best.

I would like to thank my committee, especially my dissertation director, Dr. Ravi Sandhu, who spent many patient hours in advising sessions with always a thoughtful comment to guide me to achieve my goals.

In addition, I would like to thank my employer, Litton/TASC, Inc., not only for footing the bill for this degree, but also for creating a positive environment for work excellence. Lastly, I appreciate the patience and support of my co-workers who sometimes have had my "fully divided attention".

Table of Contents

List of Tables

List of Figures

List of Boxes

List of Abbreviations

AC	Access Control
ACTS	Advanced Communications Technology and Services
API	Application Program Interface
AS	Authentication Server
ATM	Asynchronous Transfer Mode
CA	Certificate Authority
CISS	Comprehensive Integrated Security System
CONOPS	Concept of Operations
CSSA	Concentric Supervision of Security Applications
DCE	Distributed Computing Environment
DES	Digital Encryption Standard
DNS	Domain Name Services
ECR	Event Correlation Request
EDR	Event Data Request
FTP	File Transfer Protocol
GDMO	Guidelines for the Definition of Managed Objects
GSSAPI	Generic System Security API
HTTP	HyperText Transfer Protocol
IDEF	Integration Definition
IDS	Intrusion Detection System
IETF	Internet Engineering Task Force
IPL	Information Protection Level
ISO	International Standards Organization
KG	Key Generator
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MD5	Message Digest, version 5
MIB	Management Information Base
MLS	Multi-Level Security
MPI	Multi-Protocol Interface
OSI	Open Systems Interconnection
PDU	Protocol Data Unit
RFC	Request For Comment
RMON	Remote Monitoring
SAMSON	Security and Management Services in Open Networks
SG	Security Guard
S-HTTP	Secure HTTP
SM	Security Management
SMCN	Security Mechanisms for Computer Networks
SMIB	Security Management Information Base
SMIP	Security Management Information Protocol
SMIv2	Structure of Management Information for SNMPv2
SMON	Security Monitoring
SNMP	Simple Network Management Protocol

SPSL	Security Policy Specification Language
SSL	Secure Socket Layer
TCB	Trusted Computing Base
TCP/IP	Transport Control Protocol/Internet Protocol
TLS	Transport Layer Security
TRUMPET	TMN's Regulators Users and Multiple Providers EnvironmenT
UDP/IP	User Datagram Protocol/Internet Protocol
USM	User Security Model
VACM	View-based Access Control Model
VC	Virus Checker
WAN	Wide Area Network
XMP	X/Open Management Protocol

ABSTRACT

CONCENTRIC SUPERVISION OF SECURITY APPLICATIONS: AN ASSURANCE METHODOLOGY

Philip C. Hyland, Ph.D.

George Mason University 1999

Dissertation Director: Dr. Ravi Sandhu

System administrators and operators today are facing difficulties with overseeing many new security capabilities at the same time that resources seem to become more scarce. The network management community faced a similar situation a decade ago when network were growing rapidly and the Simple Network Management Protocol (SNMP) and related standards first were implemented by vendors. This research applies a similar strategy of core standards and common data definitions to create critical functionality for security applications. We attempt to benefit from the lessons of the earlier decade by adapting and extending certain methods and concepts that have become widely accepted. Benefits include an integrated approach to sharing security event information across types of security applications along with using information from administration, operations, and assessment phases to continuously adapt to current conditions and threats.

This dissertation questions the status quo of Security Management (SM) tools that function in an isolated, disjointed fashion. We draw parallels with the network management community of a decade ago and suggest remedies. We review several security management research projects and related security technology to assess progress to date. We appraise the value of secure management to particular security

applications as a means to select case studies for detailed discussion. We discuss how security managers can benefit from an integrated "big picture" context to interpret network events, just as network managers do.

In analyzing the needs for security management, we use $IDEF_0$ diagrams to divide management functions into three phases: administration, operations, and assessment and trace data flows among modules. Different skills, authority, and data are needed to perform tasks in each phase, but some information must flow for efficient and effective functionality. We give suggestions on how a prototype system might function by describing several operational scenarios for an integrated SM framework that we call *Concentric Supervision of Security Applications (CSSA)*. CSSA enables a continuous flow of SM information by defining secure data structures, communication methods and management procedures that share a common, coherent foundation regardless of the particular security application. A unique feature of this approach is that all compatible applications can generate a security event, not just dedicated security applications and probes. This offers a great opportunity for innovation by general application vendors to expand the view of the security operator.

Contributions include the definition of new architectural elements that are necessary for deployment of a secure management infrastructure. We develop the Security MIB (SMIB), Security Management Information Protocol (SMIP), and the Security Monitor (SMON) designs as proposals for critical CSSA data elements. Specifically, we show how: **1)** the Remote Monitoring (RMON) concepts used in network management can be adapted and extended to create the Security Monitoring (SMON) MIB for monitoring security applications; **2)** the Security Management Information Protocol (SMIP), evolved from the Routing Information Protocol, can distribute critical monitoring and control information; and **3)** a core Security MIB (SMIB) may be implemented with provision for extensible plug-in modules for special/future security mechanisms.

As a whole, CSSA serves to enhance rather than replace current narrowly based, "engineered" solutions so that SM systems can support "the way people work" rather than how engineers imagine. We conclude that several factors are converging to make security management a pressing need, but that CSSA

provides a framework for an integrated, standards-based solution. Finally, beyond the technical challenges, a great amount of cooperation will be required to implement common standards and achieve the benefits of SM interoperability.

Chapter One—

Introduction

Business managers have begun to recognize information system security as a major corporate requirement. The rapid growth in sales of security products such as firewalls and virus checkers [42] is a significant indicator of the growing awareness that valuable data and computing resources need protection. Public and private research and development efforts have produced a variety of security technologies for encryption, authentication, access control, key management and anomaly detection. However, security is not strictly a technology problem; rather, system developers and operators need balanced solutions that permit reliable implementation of a single, consistent *security policy* throughout a system's lifecycle. A critical part of a complete security solution—along with sensible security policies and robust mechanisms—includes *assurance* methods to ensure that security designs and implementations continuously meet the objectives of the security policy.

Assurance activities may span from early lifecycle prevention of design errors to configuration errors during deployment, detection of anomalies during operations, or rational recovery from the effects of a security event after it has occurred. A complete security assurance plan would address all phases; however, we confine our research to Security Management (SM) activities after system deployment.

We define SM to include the real-time monitoring and control of active security applications that implement one or more security services. The importance of SM is growing rapidly as more security applications are deployed and as efforts to consolidate management activities continue. The need for SM is multiplying, much as the growth of LANs created a demand for better network management solutions. As

the active part of the assurance component, operational SM deserves and requires additional research to harness the full potential of many evolving security devices and systems.

1.1. Statement of the Problem

Although there are a number of SM research problems that are worthy of investigation, we indicate next the particular issues that we address in this dissertation.

1.1.1. General Security Management Issues

Many security tools exist for protecting data, providing security authentication, and enforcing system access control, yet security is still a leading concern when deploying Internet applications. This is especially true for applications that implement various aspects of E-business (i.e., electronic commerce combined with use of the network for vital business functions). The wealth of security tools makes the management challenge even greater. The diversity of management tools and proprietary approaches hinder a common, "big picture" view across multiple security applications because a common language and understanding of events is lacking. A broader view of security events arguably leads to more proactive response to security threats as suggested in the Defense Information Systems Agency's (DISA) automated Intrusion Detection System (IDS) effort [40]. We believe that many of the benefits that Simple Network Management Protocol (SNMP) and standard Management Information Base (MIB) definitions have brought to the network management community can be emulated in the SM arena. To balance advancements in security policies and mechanisms, a better understanding of and implementation framework for the assurance function is needed to help operators maintain situational awareness and leverage security-in-depth concepts. Improvements in SM should increase the level of trust in the systems and reduce the administrative burden of deployment and maintenance.

1.1.2. Specific Security Problems

Although many pressing issues surface in discussing SM, we will focus on a few important items that are not receiving adequate attention thereby constraining potential benefits. First, the lack of standard MIB definitions for SM data elements limits interaction between vendor applications and generic SM tools.

Second, even given consistent understanding of management elements, there is no standard protocol for propagation of security status between management entities. Third, a unified understanding of security performance parameters is needed for distributed security monitoring of security applications. Without progress on these problems, monitoring and control strategies cannot achieve the critical mass needed for consolidation and synergy among application vendors that benefited the network management community so greatly in the early 1990s.

1.1.3. Thesis Statement

We assert that with the proper security mechanisms and management infrastructure, remote monitoring with computer-assisted correlation and management of system events can be designed to be just as viable for SM as it has been for the network management community. In that light, this dissertation shows how:

- the Remote Monitoring (RMON) concepts used in network management can be adapted and extended to create the Security Monitoring (SMON) MIB, tailored for monitoring security applications,
- the Security Management Information Protocol (SMIP), an evolution of the Routing Information Protocol concept, can distribute and access critical monitoring and control information to support assessment of security events, and
- a core Security MIB (SMIB) may be implemented with provision for extensible plug-in modules for application-specific and future security mechanisms.

These designs, along with the CSSA infrastructure concepts, represent a bold step toward realizing the untapped potential for an integrated SM capability. While other security MIB developments have typically dealt with read-only status collection, our approach permits interactive management using active feedback. Integrated assessment tools allow secure adaptive control and re-configuration of security mechanisms.

1.2. Background

Computer security has been of interest since the first multi-user systems. Only recently, since vital data and critical business functions moved onto networked systems, have network security mechanisms proliferated. This, and the rapid growth of E-commerce, has created the notion of E-business, in which essential business processes are embedded in the corporate network. User expectations of system quality, privacy, performance, and reliability are growing. The rapid deployment of new security technology needs flexible, efficient management to help keep system operators from being overwhelmed by cost and skill requirements related to configuration and monitoring overhead.

Generally, security solutions try to establish perimeters or layers of protection to filter what data passes in or out. Multiple layers and access points make robust network security systems a natural example of distributed operations in both implementation and management aspects. The level of threat to the resources and data within a system makes active management of security capabilities an important distributed operations mission [7].

SM has two main functions - monitoring and control. The first involves data collection that provides insight to system stakeholders[1] on whether security operations are achieving the security policies intended by the system design. Status presentation may be in the form of real-time graphical displays, log records, or periodic printed reports of operational data trends or exceptions. The frequency, type, and granularity of data gathering are necessarily tradeoffs with the network traffic volume and processing load of monitoring components since more details require more messages, analysis, and storage. The second function of SM is to implement controls as a means to adjust the level of security monitoring and operational safeguards if the current level does not match security policy, the desired level of risk, or performance needs. This may result in some configuration or readjustment of capabilities and privileges,

[1] Stakeholders is a term meant to imply all responsible persons involved with a security system, not just the system operators and end users. It may include data or business application owners or equivalent security accreditors in government organizations.

depending on the application. Such adaptive response to conditions is a cutting-edge idea [59], but it is an inherent part of our CSSA infrastructure concept introduced later.

Traditionally, SM has been viewed as a special case of network management [45]. However, security and management are interdependent by their nature, therefore each needs the services of the other. Thus, *management of security* and *security of management* are different facets of the same issue. Security of management is a prerequisite for any true high reliability or secure application, particularly so for management of security which should be both highly reliable and secure. The requirement for security of management *before* management of security is essential since an insecure control system may be used to corrupt associated delivery of services or disrupt access to the resources of a system.

To date, much more work has been done to define and analyze security mechanisms than to extend management capabilities for operational security applications. Of the several types of security identified in **Figure 1**, few have been treated in an integrated manner for the purpose of maintaining a holistic view of security status. We believe this to be a critical shortfall since a unified view of security conditions and

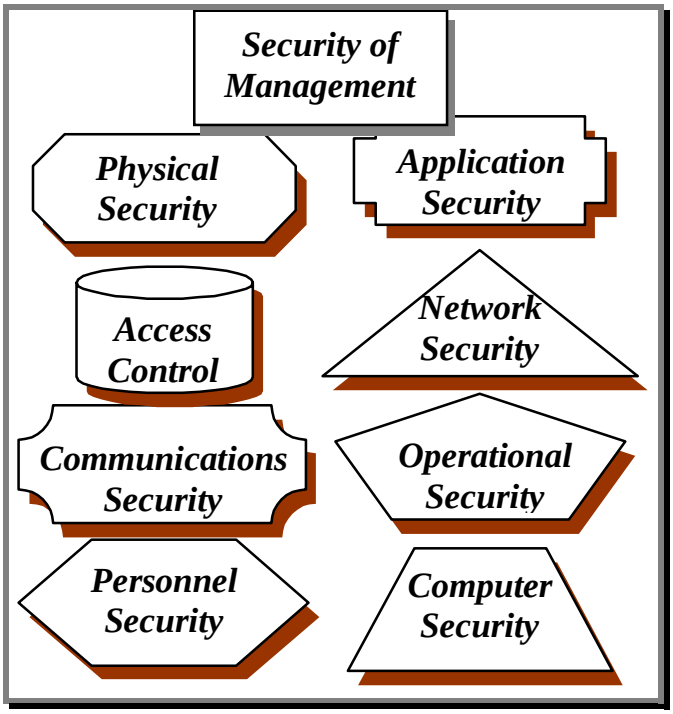

Figure 1. Types of Security

efficient administration of security are vital to effective deployment and operation of many types of security devices. We will focus on network security applications in this thesis because of the close linkages between network management and SM functions, both operationally and administratively.

1.2.1. Network Security

Network security and how it overlaps with other types of security is not precisely defined. Network security is the converse of host security, covering external security subjects such as cross-platform log-ons, inter-system process communications, traffic filtering, transport privacy and integrity, non-repudiation and trust between systems. We look at network security from the level of packet-handling security devices (i.e., firewalls and the like), but also acknowledge the importance of system-level security tools such as Certificate Authorities and IDS. The complexity and interdependent nature of network security demands an up-to-date system view and the capability to gather and correlate underlying event details. Network security management is by nature a distributed function, therefore, it requires remote management tools to efficiently monitor and administer a large number of security devices. Other applications related to network security that may utilize SM include virus checkers on email servers, secure teleconferencing, electronic commerce servers, intrusion detection applications, etc.

SM faces the same security threats as other distributed applications; however, since it may be considered the nervous system of the security system, it must be protected most diligently. SM requires largely the same security services, namely confidentiality, integrity, access control and authentication. Non-repudiation is less of an issue, although some more paranoid scenarios in E-business contracting application might suggest the need to include it within the SM purview.

Coordinated management of security is not feasible without a secure management infrastructure that protects in transit messages from modification, spoofing, and replay. Although end system and physical security are beyond the scope of this discussion, it is clear that key management, authentication, access control, and reliable implementation of management software are crucial to a secure management capability. Trusted operating systems should prevent penetration of management systems through a weak link in the management platform.

In its crudest form, SM could require human presence at every security device and manual evaluation of all significant events. On the other hand, we believe that remote monitoring with computer assisted correlation and management of system events can be designed to be just as viable for SM as it has been for network management. In fact, it may be argued that detection of sophisticated, low level attacks needs the help of computer-assisted correlation tools even more than network management systems. Whereas network degradation and faults are typically the result of random failures or mis-configuration, security incidents may be the result of a coordinated, distributed attack that is not apparent when a few events are viewed in isolation. Some network management systems use remote trend analysis and pattern recognition of management data to initiate automated responses or recommend operator action. Research on similar possibilities for security has focused on intrusion detection based on analysis of logs. Development of more general, cross-application correlation tools are more a matter of market demand and design than technology limitation. We believe the lack of standard data definitions for managed security objects and procedural standards have limited widespread deployment and interoperable implementations of SM tools.

Even a small network with modest security needs will soon face significant administrative overhead to configure and monitor firewalls, authentication servers, secure email servers, etc. Organizations are now coming to expect both privacy mechanisms and firewall protection, but competitive pressures are driving administrators to reduce labor costs of network and system management through automation and consolidation of management activities. The rapid deployment of security services in corporate and public networks reinforces the need for SM systems that are both efficient and effective.

Like other distributed applications, SM modules must speak a common language to exchange information. However, as discussed above, secure communications is a prerequisite to SM. Between 1993 and 1996, SNMPv2 (RFC 1445-1452) [14] [16] [18] [19] [22] [35] [36] [56] and its derivatives (RFC 1901-1908) [11] [12] [13] [15] [17] [20] [21] [23] proposed several security enhancements over the ubiquitous, but unsecured SNMPv1 [9]. Unfortunately, the IETF standards working group failed to reach consensus and the SNMPv2 effort collapsed under its own weight. In 1998, SNMPv3 (RFC 2271-2275)

[6] [10] [41] [52] [74] emerged to combine the best aspects of SNMPv2, SNMPv2c, and SNMPv2*. Since SNMP is more pervasive than the ISO's Common Management Information Protocol (CMIP) standard [47], SNMPv3 is an important protocol for secure transport of SM data. Some research projects such as Security and Management Services in Open Networks (SAMSON) [64] and Security Mechanisms for Computer Networks (SMCN) [58] have looked at certain integration aspects of the SM problem. However, more sustained research is needed to establish a real vision and strategic plan so as to bring order and synergy to the topic. With that thought in mind, we seek to offer some commonsense SM rules, views and tools, and a roadmap for additional research needs to enhance security assurance.

1.2.2. Security Assessment Methods

Pre-operational analysis of security may involve extensive testing and the use of rigorous logical analysis referred to as *formal methods.* This approach is widely applied in critical aviation, nuclear power and medical systems, as well as security kernels, to enhance reliability [62]. The need for highly reliable security systems cannot be satisfied merely through design and testing, especially since protection from malicious parties is a fundamental need[2]. Although security mechanisms must properly implement security (like security kernels) and must be invoked properly (not bypassed or spoofed), the assurance function typically occurs in a separate application rather than internal to the security mechanism. This gives functional independence, but creates the need for an integrated remote management framework. Developers for critical systems have found that reliable systems must address:

- Fault prevention during design and development,
- Fault detection during operations and
- Fault recovery during abnormal or error states.

Network SM applications, per our earlier definition, should concentrate on the latter two areas as they relate to networks. Thus, SM tools function as active assurance methods to monitor operational

[2] Critical systems, on the other hand, depend somewhat on the low likelihood of random conditions that cause error states, whereas computer hackers purposely search for the weak points that exist in any complex system.

security services, allowing observation and reaction to important fault, configuration, and performance status and permitting remote configuration and performance control.

With respect to the responsibility for proper security operations, security mechanisms (and security kernels) may be comparable to automobile drivers who are ultimately responsible for safe operations. SM is more like a traffic cop who reinforces the rules, adjusts controls for optimal performance, and assists in trouble spots should incidents occur. This somewhat detached stance is the reason for using the term *supervision* as opposed to management or control in the title.

1.2.3. Assessing Administrative and Operational SM

In 1987, the OSI Security Architecture (ISO 7498-2) [45] identified three categories of SM — system security management, security service management, and security mechanism management — plus security of management itself. System SM is concerned with overall policy management, interaction with other OSI and SM functions and with general security functions such as event handling, auditing and recovery. Security service management deals with interactions with particular security services such as negotiation of protection levels and mechanisms. Security mechanism management includes management of keys, encipherment, digital signatures, access controls, integrity, authentication, traffic padding, routing control and notarization. The purpose of security of management (the main feature of SNMPv3) [41] is to define and implement methods and structures to ensure vital status and control functions are protected from disclosure, modification, masquerade, replay, etc.

Although the OSI functional breakdown is useful as a reference model, it does not recognize the typical division of duties between system administrators, security managers, and systems operators. In contrast, we postulate three major phases of SM that distinguish the desirable characteristics of a management framework. The phases are *administration*, *operations*, and *assessment* of security. Each phase of SM requires access to a common core of management data, but the duties - and therefore the privileges - in each phase are different. In our concept of a SM infrastructure, which we call *Concentric Supervision of Security Applications* (CSSA), we further consider SM from the view of *who* is involved, the *purpose* of their activities, and the *types of information* affected. Our intent is to demonstrate that

different assumptions, rules-of-thumb, and tools are applicable in each phase, thus justifying our subdivision of the discipline.

A security program depends on the correctness, completeness, and reliability of three related components – *security policy*, *implementation mechanisms*, and *assurance measures*. As a starting point for discussion, **Figure 2** shows one view of the essential SM components. First, all systems function under a security policy (even if it is effectively null). Each organization that is charged with maintaining and protecting sensitive information needs to begin with a comprehensive security policy. The security policy defines what information needs protection, what measures shall be applied, and responsibilities for important actions. A complete specification of security policy is not always an attainable goal since updates and unanticipated needs may require augmentation of automated procedures with manual review methods. The tools to specify security rules may not properly capture all aspects for complete and sufficient definitions. Security policies, preferably derived from a risk assessment based on perceived threats, vulnerabilities and value of resources, set the guidelines for operational procedures and security techniques that counter security risks with controls and protective measures.

The goal of the security manager is to apply and enforce consistent security policies across system boundaries and throughout the organization (or security domain if the two are not synonymous). Security

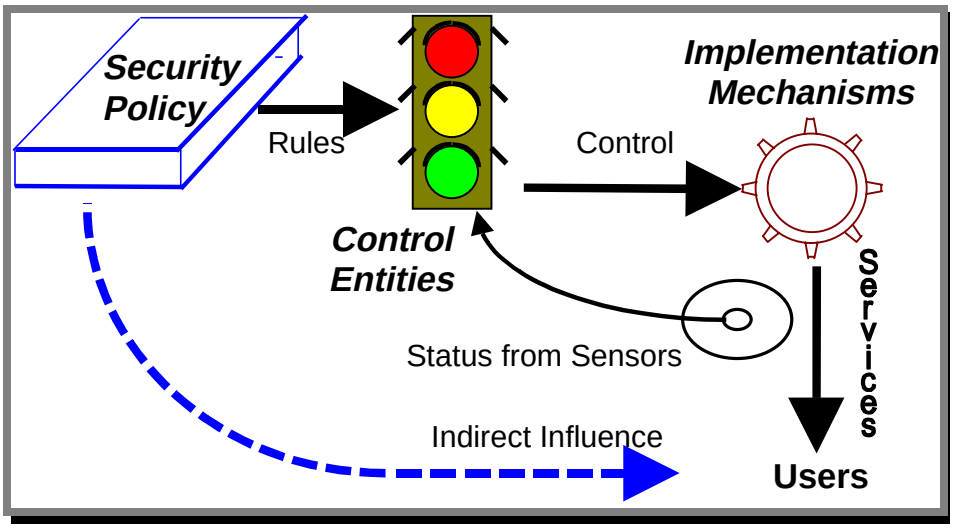

Figure 2. Security Management Components

policy has a direct impact on the rules and policing actions that ensure proper operation of the implementation mechanisms. As seen in **Figure 2**, policy has an indirect influence on users of security services. They interface with security applications; policies are typically visible only by how they control access to or denial of selected security services. The security policies of the organization determine the balance between users' ease of use and level of responsibility versus the amount and burden of controls and countermeasures.

From the security policy, a security administration process defines configuration and operational *rules* that specify security subsystem behavior. The administration process supports operational SM (represented by the traffic light labeled "Control Entities") which controls one or more security mechanisms. These mechanisms provide on-line security services that actually affect security clients. The feedback path from status sensors to the control entity represents important operational health and anomaly detection information via logs, alerts, and status polls. This feedback process is an essential feature of the CSSA concept that enables adaptive, value-added control and assessment of security configurations.

The assessment phase of SM is not visible in **Figure 2** because until recently, assessment has largely been an off-line, human process due to the substantial time and processing demands. The goal of assessment is to analyze dynamic feedback from a variety of sensors to identify possible security dangers and modify the security posture before more severe events occur and the perceived security threats become reality.

In fact, there are two major shortfalls with this initial view of SM. First, nothing in **Figure 2** integrates the management of separate security mechanisms together into a unified, coordinated capability. In other words, each mechanism might have its own parallel rules, configuration process and control structure that does not interact, or benefit from the capabilities or status information of the others. Second, **Figure 2** suggests no lifecycle view of the SM process that ensures configuration data, operational data and historical audit/performance data can all be accessed and assessed to support continuous improvement. We take the view that the initial security policy should be the starting point of a spiral process. This process should dynamically update rules to adjust and improve the security posture based on current operating

conditions, the time of day or week, and the perceived vulnerabilities, etc. We believe a common SM framework will help resolve these data management issues and thereby bring more effective and *mutually supporting* management to security devices and applications. A unique feature of our CSSA approach is that any compatible applications can generate a security event, not just dedicated security applications, and probes. This offers a great opportunity for innovation by general security application vendors (not just those who build IDS and firewalls) to detect and report security-related information and thereby extend the security horizon for the security operator.

The objective of assurance is effectiveness. SM tries to ensure that the security services delivered are adequate and compliant with the organization's security policy while minimizing the administrative overhead. As **Figure 3** depicts, the CSSA concept defines a three-phase process showing common access to Security MIB data in the middle. The CSSA process operates as an endless cycle of observation, assessment, and reconfiguration, always seeking the best management state through adaptive feedback. Each SM phase, as defined in the CSSA concept, is discussed further below.

Although the administrative SM phase could include pre-operational system development and testing, we focus here on the *user-driven* configuration activities of system installation, setup, and maintenance. The term user-driven refers to the fact that most changes and configuration issues are a result

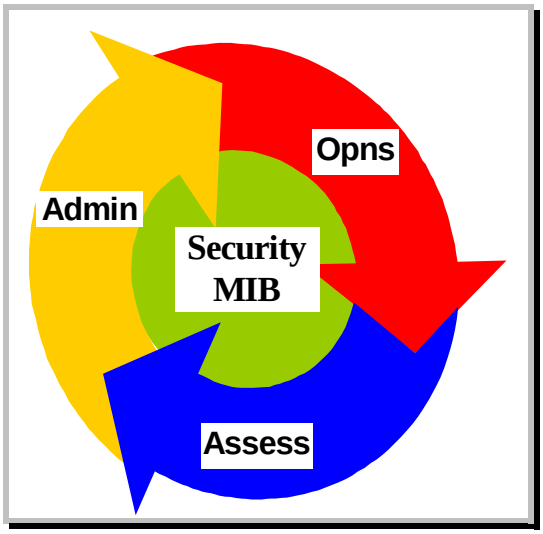

Figure 3. Concentric Supervision Cycle

of new users, moves, privileges changes, and the like. Routine administrative activities are often done in a batch mode since immediate response time is not expected. Often, administrative security actions may be stopped or interrupted for a period without degrading security. In other words, traditional administrative SM is relatively routine and part of the normal day-to-day functioning of the enterprise. Our CSSA concept describes a more dynamic process in which a status assessment may initiate on-the-fly configuration changes. Active administration minimizes system vulnerabilities by opening system access enough to meet users' needs while deliberately constraining certain activities enough to satisfy the applicable security policy.

Differences between administrative SM and operational SM are evident, especially in their purposes. Operational SM is a more active, *event-driven* component of assurance that is concerned with detection and reaction to current conditions applicable to security mechanisms. Operational SM consists of real-time interactions between the security service *providers* (mechanisms), *status sensors,* and *control entities.* Operational SM exists to detect non-compliance with security policy; thus, operational management tools must be both sensitive and continuously available to reliably track and respond to unpredictable security events.

Operational tools should be designed to monitor and maintain security posture at levels defined by the established security policy. As the desired protection level increases, the corresponding management functions must become more stringent. As an example of graduated response, **Table 1** shows several Information Protection Levels (IPL). At IPL level 1, access controls defined by administration level Ad1 would not be as restrictive as for IPL level 2 and above. Likewise, as the IPL increases, operational thresholds (Op1/OP2/Op3) may get tighter and assessment responses (As1/As2/As3) may become more conservative. The details of our concept of using IPLs to regulate security responses are discussed further in Chapter Four.

Table 1. Information Protection Levels (IPL)

IPL	Admin	Opns	Assessment
1	Ad1	Op1	As1
2	Ad2	Op2	As2
3	Ad3	Op3	As3

The assessment phase of SM is *performance-driven*; that is, it is concerned with measuring whether objectives are met and how potential changes may affect the security system. Evaluation of audit trails and pattern-matching are assessment activities that try to detect and correlate suspicious events. Such assessments may have a short-term or long-term scope. Short-term, quick response assessments generally support reaction to imminent threats that are detected during the operations phase. Based on other events, correlated in time, space, or modus operandi, an appropriate response may be initiated automatically or recommended to an operator for action. Long-term assessments may support security policy planning, trend analysis of threats, or quality of protection issues. Use of audit data or incident logs with a simulation tool has the potential to test a proposed administration state against a range of operational conditions to assess its desirability in the real world. This is a good topic for further research.

1.2.4. Security Management Research

The literature and available products related to management of security applications is quite sparse, especially compared to work in other security topics. Much of the security research has focused on protective techniques and data analysis. Intrusion detection [54], key management for multicast conferencing [2], [37], and HTTP security [67] have received some attention, but no standard security MIBs exist and no integrated SM functions to monitor and control generic security applications are in wide use. Although one group of vendors started work on an IETF draft for a firewall MIB [38] and another research group [49] considered use of a Firewall MIB for firewall control, MIB definitions have been limited to read-only status parameters. In one way, the efforts reinforce our assertion that common management parameters will benefit all players, but the group's initial work is quite narrow in scope. It does not assume use of secure SNMP; therefore, no Set actions are permitted, as would be needed to implement the CSSA concept of active, continuous observation, assessment, and re-configuration. Another report [4] describes a fresh approach to enhanced firewall configuration through knowledge modeling; however, the authors do not emphasize operations and assessment functions or the security of management.

Despite vendor hype to the contrary, management tools for secure applications are limited in capabilities and generality. Although a few firewall products such as CyberGuard™ can use SNMP to

identify security alarms or to poll for configuration and status data from a network management station, open standards that permit both monitoring and control are not generally available. One of the few security devices that permits setting configuration information via SNMP is GTE's TACLANE Information Security Manager (TISM). TISM is a unique NSA-sponsored system that uses private enterprise device MIBs over a hardware-encrypted channel to manage (rekey, zeroize, and configure virtual circuits) TACLANE™ and FASTLANE™ ATM encryptors.

SM has long been considered a sub-function of network management. It is one of the five functional areas defined in the OSI management framework [46]. International standards for security functions like audit trails, security alarms and notifications, key management, authentication, and access control have generally progressed much farther that similar work in the IETF community. Between 1992 and 1994, a European security management prototype called Project SAMSON [64] identified an integration architecture that included both CMIP and SNMP interfaces for management of security mechanisms. The prototype system implemented management interfaces for directory services, inter-domain services, key management, and security services (audit, authentication, and access control). SAMSON did not address event information sharing or statistical monitoring as provided by SMIP and SMON. Another project called WILMA [75] produced some SNMP development tools in 1995 for SM. Muftic wrote in [58] about the Security Mechanisms for Computer Networks project (1985-1990) which included SM in its Comprehensive Integrated Security System (CISS) design. A more recent project under the Advanced Communications Technology and Services (ACTS) program is called TRUMPET [70]. The project goal is to investigate the required integrity of inter-domain access to TMN-based management systems. The project will propose solutions for access to others management systems whilst respecting system integrity, confidentiality and means of auditing contractual negotiation. While TRUMPET appears to have solid backing, it is tuned to a specific security application with particular needs. Our CSSA concept ties together status and event information from many different security applications to build an aggregated system security view.

The basic concepts of common SNMP and the CMIP network management protocols are covered by authors such as Rose [61] and Stallings [68]. Distributed Management Environment (DME), once considered an important alternative, began as a part of the Open Software Foundation's (OSF) broader Distributed Computing Environment (DCE) initiative. DME aimed to address management of large, heterogeneous networks by defining a common high-level interface for network devices and applications. It used a single API called XMP (X/Open Management Protocol) [69] to access common functions of the SNMP and CMIP protocols (see **Figure 4**). As a first step, the OSF announced the Network Management Option (NMO) 1.0 specification in May 1994. The independent SAMSON project mentioned above studied and developed a working prototype using XMP with SNMP and CMIP stacks. DME has since suffered from lack of vendor interest and is now defunct.

Figure 4 also shows another management interface alternative using Secure HTTP (S-HTTP). Although use of S-HTTP is not proposed by existing working groups, the use of HTTP for GUI front-ends for non-secure management (i.e. Web-based Enterprise Management [WBEM]) has been gaining momentum. Some Internet drafts on the HyperMedia Management Protocol (HMMP) have been circulated which could use S-HTTP, or the Secure Socket Layer (SSL) directly, for privacy. To keep the options open for alternative protocols to transport management information, we use the more general term Multi-Protocol Interface (MPI) in our CSSA infrastructure defined later in Chapter Four (see **Figure 21**). MPI resides between the main management system and the external agent modules.

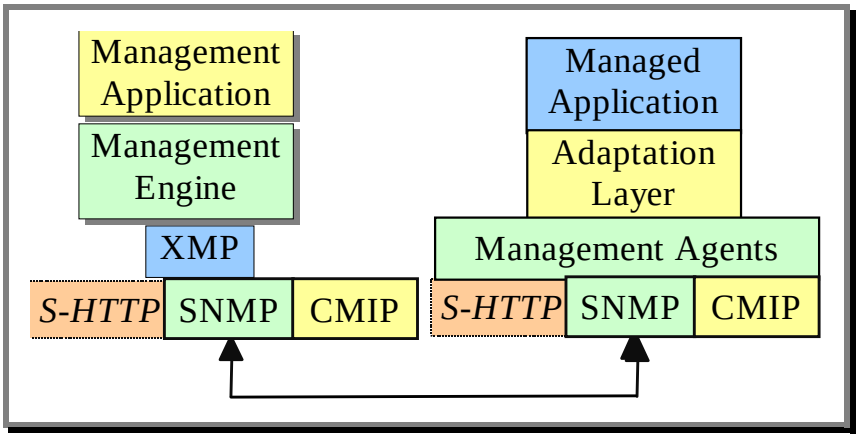

Figure 4. XMP Management Infrastructure

The Network Management Forum has attempted to reconcile the SNMP and CMIP environments. Its OMNIPoint 1 document was intended as a roadmap for compatible specifications. As part of this effort, the ISO/CCITT Internet Management Coexistence (IIMC) working group defined MIB translation rules and proxy definitions toward this end. A review of management security issues compiled by the IIMC working group is in [49].

In addition to work referenced above, much of the SM work published to date relates to IDS applications. Early IDS work involved off-line analysis methods for detecting anomalies or attack patterns in audit data from standalone systems. As analysis techniques and distributed processing capabilities have improved, more recent IDS work has become more real-time and cooperative, similar to other event-driven network management functions. Crosbie [25] proposed using "autonomous agents" for redundancy and simplicity. White, Fisch and Pooch have worked on "cooperative, peer-based" IDS [73]. Both efforts focus on the functionality of the IDS application, but intercommunication and management functions are clearly needed and identified. This suggests that the future of IDS may consist of independent, but communicating detection tools. Coordination and management of distributed IDS agents may be considered as distributed management functions that are specialized for IDS. If the IDS protocol and management capabilities were to be aligned with SNMP, then standard-based linkages to generic management applications may be the next logical progression. Additionally, linkage of IDS agents with firewall, virus checker, and other security applications using secure management tools may enable better protection than any one method could provide alone. This is the motivation behind the CSSA concept.

Multicast security management has been an active research topic due to the growing interest in Internet conferencing. As business use of multicast becomes more commonplace, the demand for multicast privacy solutions will grow. Authentication, encryption key management, and access control are big concerns as the number of participants in a conference increases. Late joins and early departures from secure sessions complicate key management due to the need to dynamically refresh the keys of active participants. Gong [37] has raised many of these issues specific to group-oriented multicast security, but many are also general SM issues. Some issues from multicast security also relate to network management

scenarios in which a central site needs to communicate with many distributed devices. Key management may be handled outside of network management standards, but using secure management to monitor and control a key management application is very conceivable. We identify key management is one of the several security applications upon which CSSA concepts may be applied.

Control of access to network transmission resources associated with multicasting has been another research concern because single multicast session can consume a large amount of bandwidth when broadcasting to a large population of receivers. Ballardie in [2] has identified problems of limiting access to multicast trees and suggested a method for controlling abuses by users who may inadvertently or maliciously consume major chunks of network resources. In previous work [44], we applied similar concepts to suggest a Packet-Filter Information Protocol (PFIP) to propagate restrictive firewall packet filter rules to reduce abusive network traffic flows.

A rudimentary SM application is discussed by Banning [3]. Banning built a distributed audit system to collect data from heterogeneous systems using network management protocols. Although the project used only a simple collection prototype and the dubious security of SNMPv1, it demonstrated many of the steps that would be needed to integrate other security applications using similar MIB definition, agent development, and value-added processing of collected data. Not every SM application should have to reapply this process independently. In fact, we believe a core set of attributes and procedures would greatly promote extension of management functions to other security applications and support greater synergy by sharing information between those applications.

The aforementioned research work along with configuration-checking tools such as Computer Oracle and Password System (COPS), Tiger, Tripwire, and Security Administrators Tool for Auditing Networking (SATAN) have improved the security position of many sites. In addition, several major commercial products such as CA Unicenter TNG™, Sun Security Manager™, and Tivoli TME10™ do support certain SM functions within their management architectures. Interestingly, none view management as a continuous cycle of activities for the purpose of adaptive SM as we have suggested in the CSSA concept. The OSI Security Architecture has little sense of temporal issues or sharing data between

administration, operations, and assessment functions. Other more generic management tools such as SNMPv3, the DCE security management framework, the Open Group's Common Data Security Architecture (CDSA) [8] and two European research projects (SAMSON [64] and SMCN [58]) focus mostly on protocols and operational tools for a few security mechanisms. Notably, the "Security Mechanisms for Computer Networks" project (1985-1990) included SM in its Comprehensive Integrated Security System (CISS) design. CISS emphasizes use of existing mechanisms and APIs for common security functions. Of the ten functional agents identified in CISS, two are devoted to security administration functions, two provide general support to all components, and six support operational SM. In addition, CISS does not address an assessment function to provide feedback for controlled re-configuration in response to current security conditions or perceived vulnerability.

1.2.5. Terminology

As technology becomes more complex, we must refine terminology and re-partition problems to understand the issues better and focus attention on solvable pieces. Below we define some important terms used throughout this dissertation.

1.2.5.1. Assurance

Assurance is the conventional term for methods to assess and ensure a security system enforces and complies with intended security policies. One may use assurance tools before, during, or after security mechanism operations. Post-processing of security events typically includes audit trail analysis and related off-line intrusion detection and trend analysis methods. Many Intrusion Detection System (IDS) applications began as post-processing functions due to limited processing and software capabilities, but most are migrating toward interactive, real-time operations [73].

1.2.5.2. Concentric Supervision of Security Applications (CSSA)

CSSA is a SM framework, an assemblage of formats, processes, and methodologies with the objective of improving security assurance by sharing security information throughout the security lifecycle and among many types of security applications.

1.2.5.3. Infrastructure

An infrastructure is the underlying foundation or basic framework of a system. A framework helps to organize and rationalize one's operational concept in a hostile environment that may include many active components and complex interactions.

1.2.5.4. Least Privilege

Least privilege is the principle that permits each subject (user or program) to be granted only the privileges necessary to perform authorized tasks. It is a means to limit unauthorized or accidental damage.

1.2.5.5. Security-in-depth

A concept, also called layered security, in which multiple, overlapping methods provide redundant protection against unwanted use of or access to sensitive resources.

1.2.5.6. Security Management (SM)

We define SM as the *real-time monitoring and control of active security applications that implement one or more security services*. The purpose of SM is to ensure that the security measures are operational, in balance with current conditions, and compliant with the security policy. Not only must the services function correctly and in a timely fashion, they must counteract existing threats to generate *justifiable* confidence in the system trustworthiness. One of the largest security pitfalls is to focus on certain security products or technologies without defining a balanced security policy and thereby gaining a false sense of security. Protection is only as strong as the weakest link.

1.2.5.7. Security Mechanisms

Security mechanisms supply security services such as encipherment, digital signatures, access controls, data integrity, authentication, routing control, and notarization.

1.2.5.8. SNMP Traps/Notifications

SNMP Traps are unsolicited messages from an SNMP agent to one or more SNMP manager nodes to announce an asynchronous event. Notifications, often associated with the OSI CMIP management standard, are used synonymously in this document.

1.3. Dissertation Overview

Chapter Two presents the major new concepts that are fundamental to CSSA, including the idea of a three phased management process, the notion of data sharing between the phases, and the definition of concentric supervision of security applications. Chapter Two also introduces our new proposals for a core Security MIB, a Security Monitoring MIB, and the Security Management Information Protocol. These concepts support security assurance by enabling security information to be shared throughout the SM lifecycle and among many types of security applications. In the system-engineering context, Chapter Two provides the Concept of Operations (CONOPS) for CSSA.

Chapter Three presents several application scenarios to demonstration the needs and viability of the SM concepts introduced in Chapter Two.

Chapter Four builds up the infrastructure components that stand upon the basic CSSA foundation. It develops a high-level design and clarifies detailed process interactions through use of IDEF diagrams and MIB stubs.

Chapters Five develops the detailed designs of the SMIB, SMON MIB, and two application-specific MIBs, the Firewall MIB and the Security Guard MIB. Extensive discussion of the design rationale and appropriate usage is provided for the more significant management parameters. The full definitions of the proposed MIBs are in the appendices.

Chapter Six expands details about the procedures and formats for the SMIP component of the CSSA infrastructure, which works with IPLs and confederated host lists to enable flexible distribution and query of security events.

Chapter Seven offers conclusions about transition issues, the state of development of SM, and the value of the research contributions herein. Future research needs are identified to move closer toward fully integrated security management.

Chapter Two—

Elements of the Concentric Supervision of Security Applications (CSSA) Concept

Correctness, completeness and reliability of management information depends upon a solid understanding of how all the pieces fit together and proper processing by both managers and managed entities. While some components of secure systems (e.g., authentication tools) are relatively well developed and understood, the management capabilities of existing security mechanisms are much less developed. Management capabilities of most security applications are not integrated with any other tools, let alone with the full spectrum of security tools. The management capabilities of existing security products are *consistently weak* at verifying proper operations through use of common status definitions. Therefore, it is fair to expect that security status is *weakly consistent* at best. At worst, the derived status could provide a false sense of assurance and perpetuate confusion, rather than establishing trust and clarity.

This state of affairs does not need to persist. The history of standards development and network management product development in the SNMP and CMIP communities shows that a modest degree of consensus on the structure of management information and protocols for exchanging the data can coalesce an entire industry and bring many new capabilities to customers. In this chapter, we propose ideas for enhancement of capabilities in the SM context based on experience with network management applications and various security tools. Our goal is to promote a better understanding of the issues and identify new approaches that can lead to integrated, consistent SM across the full range of security applications.

2.1. Purpose

The intent in this section is to introduce our new ideas as a whole, prior to more detailed discussion later. We refer to these concepts collectively as Concentric Supervision of Security Applications (CSSA). The goal of CSSA is to instill rational confidence in the proper and efficient operations of one or more security applications. We hope this may represent a step toward our ultimate objective, to stimulate security system developers to adopt useful standards and implementation conventions for integrated SM of common security applications.

2.2. Description (What is CSSA?)

CSSA is a SM framework, an assemblage of formats, processes, and methodologies with security assurance as its watchword. It is appropriate to dissect the term CSSA, because it conveys specific meanings about the concept. First, the term *concentric* refers to several ideas useful in SM. These include the notion of applying layers of security to attain security in depth, the use of a common set of management data for SM phases and across applications, and the levels of protection assigned to various system resources (see **Figure 5**). Second, the term s*upervision* is used purposefully to connote a loose and somewhat indirect degree of control over security operations. Specifically, the supervisory activity is not

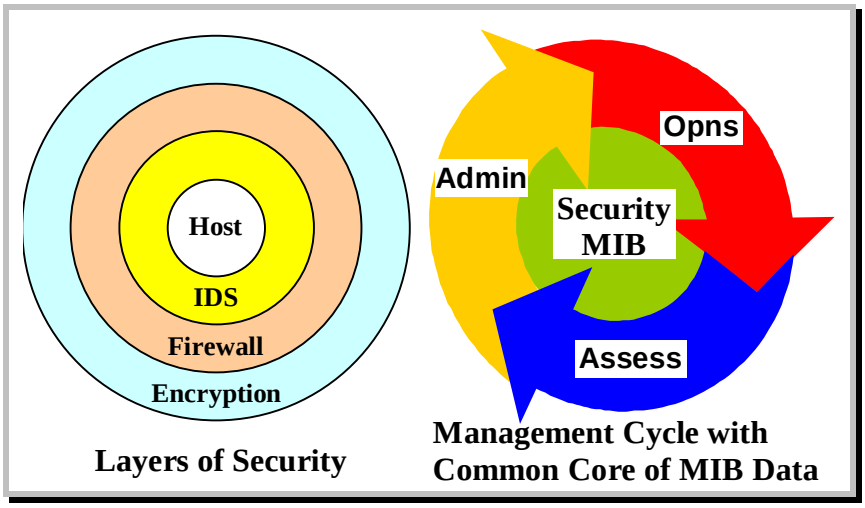

Figure 5. Concentric Concepts

directly responsible for security enforcement, rather the managed applications are. This may ease requirements for management applications to be part of the Trusted Computing Base (TCB) since the security application enforces security rules and must not accept insecure directives from management applications.

Last, the *security applications* with which we are concerned may include any generally established security mechanisms that support the security infrastructure or provide security services to users. The most notable security applications are firewalls, audit trails, key management systems and key generators, certificate authorities, virus checkers, authentication servers, security guards, and access control mechanisms. Some of these applications may be combined in a single platform, but typically analysis and accreditation of single function products is simpler.

The CSSA concept identifies several requirements needed to adequately manage security applications and translates those needs into a set of functional capabilities as suggested by **Figure 6**. The fundamental infrastructure elements are the building blocks that, along with an operational concept, provide an open, standards-based foundation for SM. Each of the CSSA components - security mechanisms, data transport, a Security Management Information Base (SMIB), a Security Monitor, and a Security Management Information Protocol - will be discussed in detail in later sections.

The CSSA concept suggests a set of functional components, as do other architectures. However,

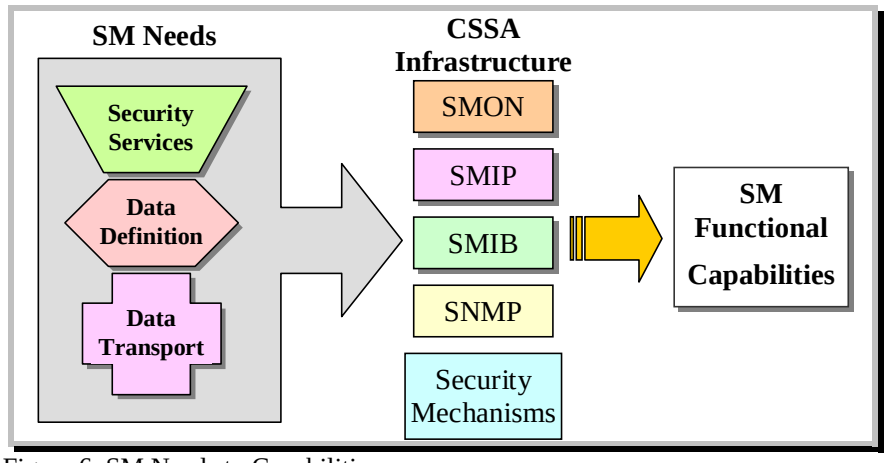

Figure 6. SM Needs to Capabilities

we also present a unique view of the SM lifecycle needs and show how the core management data may be used through a system's lifecycle and across applications. CSSA is also consistent with the concept of security domains that function with redundant layers of defense, intrusion detection tools, and reactive methods to protect resources in the case of attack. A central idea for CSSA is that the use of shared data among the three management phases, administration, operations, and assurance, enables each to perform better. Although the processes in each phase may function independently, they are able to share status about security events with one another in a mutually supporting way.

2.3. CSSA Management Phases

We postulate that all management activities fit into three distinct phases in the management lifecycle. We organize management functions into three phases because similar data, processing, and skills are used in each phase. In this section, we present details of how CSSA phases mutually support each other and use a common base of information to provide their value.

2.3.1. Administration Phase

The administration phase includes both initial configuration of security services and routine updates to add, delete, or modify user and resource information. The authority for such actions must be carefully controlled and audited. Before updates are applied, cross-checks for compliance with security policy should occur. This implies a security policy that has been tuned to a set of guiding principles. Although the development of security policies is beyond the scope of this research, it is an important and logical result of a risk management/analysis process. Although the development of comprehensive security policies may be as much an art as it is a science, a great amount of interest by functional experts has lead to number of methodologies. In fact, work is underway on a Security Policy Specification Language (SPSL) [24], which could help to clearly and concisely define policies. If the security policy is defined as a consistent set of management rules, the validation of administrative updates may be automated. Otherwise, a manual checklist procedure can verify the congruity of each change. The difficulty of verifying changes can be affected by the scope of the parameter. Some configuration parameters apply widely (e.g., current

information protection level) while others may be relevant only to specific security mechanisms (e.g., key expiration date).

Suitable control over the integrity and correctness of security configuration information is vital to compliance with a security policy. Delegation of authority to make changes needs to be flexible and very fine-grained. The senior security administrator may need full authority to update configuration, operational and assessment roles and privileges, but may delegate local tasks, such as user account updates and modification of access lists for local resources to subordinate security managers. *Least privilege* facilitates decentralization without giving full authority to each individual.

Administrative management also may be viewed as a workflow problem. A sequence of steps and cross-checks can be designed to ensure proper authority and compliance with policy rules. This process is reduced to a set of Integration Definition (IDEF) diagrams in Chapter Four. Relationships may become more complex if the security policy requires special measures such as two-person authorization (i.e., for database updates) or Chinese wall access controls. Separation of duties permits independence while enforcing integrity checks for roles in which one person could have undue authority.

2.3.2. Operations Phase

Operational SM is characterized by real-time interaction with security components that provide security services to end-users. Management tools for security operations need several capabilities such as real-time interactive control, detection and alarm mechanisms, event logging and correlation, and graphical display systems. In addition, limited access to configuration and historical data makes trouble-shooting and problem prevention easier. Different roles with tailored privilege levels may allow the "night shift" personnel or non-operations personnel only essential functions. On the other, an experienced system operator should have read access to contact information for any user within their domain and should have write access to certain configuration data within their responsibilities. Consequently, the operator would be able to notify the user in case of an alert from a certification authority that their certificate is near expiration, and if authorized, update their certification.

The driving force in design of operational SM is dealing with active or suspected security threats. Like the performance and fault management functions within the field of network management, operational SM has a strong temporal factor. Monitoring the real-time status of performance and component health, along with tracking of security events is necessary to permit active responses to changing conditions. For example, in a relatively benign security environment, an indication from a network component sensor of a performance bottleneck might result in the control element setting a more permissive filtering stance (e.g. reduced logging). This may improve performance until the backlog is reduced. On the other hand, exceeding a security monitoring threshold (such as number of repetitive log-on failures) that signals a possible attack may justify increased event logging, alarms to system operators, or automated responses to deter damage to system integrity. This adaptive response to real-time conditions is a step beyond the manual, bi-modal operation recently announced by one firewall vendor [59].

2.3.3. Assessment Phase

In many ways, the assessment phase is the most critical in creating an effective management system. Assessment is closely related to intrusion detection, which also encompasses event reporting. Assessment provides the chief feedback loop to modify system attributes based on current conditions. Events that are detected and recorded during operations from all detectors and security applications need to be matched against known patterns, related targets, or even similar times of occurrence. Depending on the danger and uncertainty of an event, immediate or long-term configuration changes may be effected. Context is crucial. Certain threats may trigger pre-determined responses such as session termination, increased monitoring, or deception mechanisms. Unfamiliar or uncertain events should result in more cautious and conservative actions.

There is debate over whether centralized assessment is preferable to distributed approaches. A major benefit of a centralized management approach is the capability to evaluate individual events against a broader system view and more sophisticated software on higher performance systems than possible with many dispersed nodes. For example, artificial intelligence (AI) methods may be helpful in correlating events across similar situations, but AI software typically requires more powerful processing capabilities

than available for agent processes. On the other hand, a consolidated assessment approach with a distributed correlation database may provide needed redundancy and scalability. The CSSA design favors consolidated assessment with hierarchical access to management data. Thus, senior management nodes can access details from peer or lower levels through manager to manager transactions.

Assessment is not necessarily a static or reactive process. As was indicated in **Table 1**, the information protection level may drive a more aggressive stance as the perceived vulnerability of an organization grows (i.e. before a new product release or during sensitive negotiations). In this way, the SM system becomes more proactive, in contrast to the reactive style of a non-integrated SM approach.

2.3.4. Management Data Storage

In many cases, important SM data for user authentication, key distribution, access control, security filters and directory services will be stored in one or more databases. The CISS model [58] calls for the aggregation of all SM data into a central Security Management Information Base (SMIB) with access controlled by a single trusted module. The access control model for SNMPv3 [74] defines a more distributed approach. **Figure 7** shows the predominant type of SMIB access during each management phase. As shown, the administration phase involves mostly writing of data, since configuration data, access control data, filtering rules, etc. are being saved or updated. On the other hand, during the operations phase access to SMIB data is more equally read-write with status updates being based on poll responses and MIB reads for display of the security status. The assessment phase will mostly read SMIB data, as requests are made for local and remote data related to an event. Additionally, some assessment results may be passed to the administration function for action, or saved to the SMIB for future reference use.

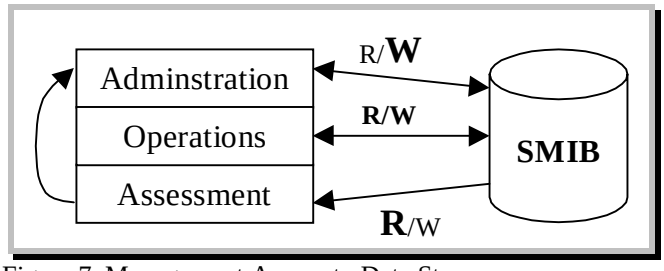

Figure 7. Management Access to Data Store

A common means to access and remotely update configuration data is important. Security mechanisms that do not implement a structured data store such as the SMIB and a secure means of network access would have to use a local configuration file for vital management data. For a large number of dispersed applications this could be very labor intensive and could lead to inconsistency among applications. It would also require a proxy agent to interface with a generic control entity using standard protocols.

2.4. CSSA Components

This section identifies CSSA needs and functionality on a component basis. The following paragraphs introduce the essential parts of the CSSA framework needed to establish a SM capability. More detailed design and development of important new contributions is provided in following chapters. Although some functions are presently available, or at least defined, others require development or standardization effort before they can contribute to the SM infrastructure. **Figure 8** shows a notional SM system with components centered on a local management entity. Security services overarch all functions, particularly the remote operations with SM agents and peer management entities.

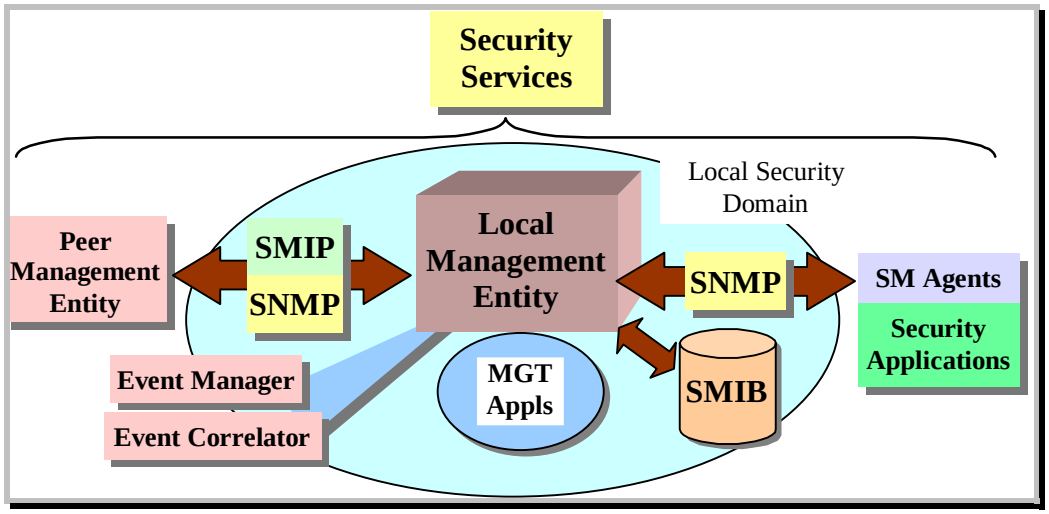

Figure 8. Framework Components

2.4.1. Standard Security Services and Security API

Security of management activities is a basic prerequisite for management of security applications. The operating environment (physical security, operating system, personnel screening, access controls, etc.) for management activities must be as well protected and secure as the security applications and mechanisms that deliver services. Otherwise, mis-configuration or other malicious tampering could circumvent the essential security mechanisms on which an application depends.

While layered security is favored for creating defense-in-depth, in reality any single vulnerability can become the single weak link that breaks the security chain. This notion is analogous to the commonsense advice to "lock the back door and the windows, not just the front door" since true security comes down to the least common denominator. The security services listed below are presumed to be properly configured and available to management entities as needed through standard interfaces such as the Generic System Security API (GSSAPI) [53] or SNMPv3:

- Access Control,
- Confidentiality,
- Authentication, and
- Data Integrity.

Chapter Four discusses the specific protocols and algorithms used in each case. Selections are made from an approved set based on the protection needs of the application.

2.4.2. Common Data Transport Mechanism

A common method of delivering management information between management entities is required just as a common language is necessary for human conversation. For SM purposes, we will use SNMP version 3, defined in RFC 2271-2275. SNMPv3 supports a distributed client-server model that permits polling for operational data in addition to random alerts from remote agent nodes. SNMPv3 supports the security services above (i.e., Access Control, Confidentiality, Authentication, and Data Integrity), but does not deal with denial of service or non-repudiation problems. The user-based security model [6] addresses authentication, integrity, and privacy needs while the view-based access control model

[74] defines how management data is handled amongst multiple entities. SNMPv3 supports a number of manager to agent transactions as well as manager to manager interactions for writing (setting), reading (getting), and modifying (e.g., incrementing) management parameters.

2.4.3. Standardized Data Elements

One of the first steps in building a unified management capability is to identify and define a set of useful parameters that can assist in situational awareness, troubleshooting, and control as applied to the particular management domain. Although universal consensus on a single set of management parameters is not feasible or advisable, we contend that a reasonable core set of management parameters can be defined. This core set of management parameters forms the basic Security MIB that supports the administration, operations, and assessment phases of the security management cycle. Once a common baseline of management data is defined, generic security tools built upon standard SNMPv3 network management station can be used to monitor and control compatible applications. The use of standard management interfaces and monitoring platforms enabled network management systems to rapidly grow in both capabilities and breadth of use during the LAN explosion of the 1990s. Similar results can be anticipated in SM.

The Security MIB (SMIB) is central to a SM concept. Despite general references by other research work (i.e., OSI, CISS) and the OSI Guidelines for the Definition of Managed Objects (GDMO) version created by the SAMSON project [65], a broad-based SMIB has not yet been fully defined or standardized. As with other MIBs (i.e., MIB-II, RFC 1213), our SMIB forms a tree structure and defines only a core set of values as mandatory elements. Additional elements can be added as branches to the tree when needed to support special new functions or special vendor features.

For this research effort, the series of MIBs will be defined in accordance with the Structure of Management Information for SNMPv2 (SMIv2) [15] to represent performance data, relevant policy, and configuration parameters for various security mechanisms. The selection and definition of managed objects to be monitored and controlled should be designed as a whole to permit effective assurance support to security operations. That is the intent of the discussion of design details in Chapters Four, Five and Six.

As depicted in **Figure 9**, the SMIB that we propose consists of the core MIB, a Security Monitor (SMON) MIB, a Security Policy MIB, an Administrative Policy MIB and optional plug-in MIBs for one or more security mechanisms. The core MIB is the only mandatory element, however extensions provide enhanced capabilities and identify existence of available services. The optional SMON MIB provides a set of common performance-oriented definitions for summarizing security status for a group of nodes. The Security Policy MIB is an encoding of the high level rules that must be implemented by security mechanisms. The Security Policy MIB holds common configuration and management information that defines organizational policies related to applications, users, and management administration. This information is used to validate initial and operational changes for security services. The Administrative Policy group captures the access and change control rules related to implementing the rules of the security policy itself. It may use access control lists and/or roles to define privileges. Each of these sub-MIBs is discussed fully in Chapter Four and Five.

As part of the design of the SMIB, an incident notification plan should be developed. For events that occur unexpectedly, notifications (SNMP traps) allow agents at distributed applications to immediately

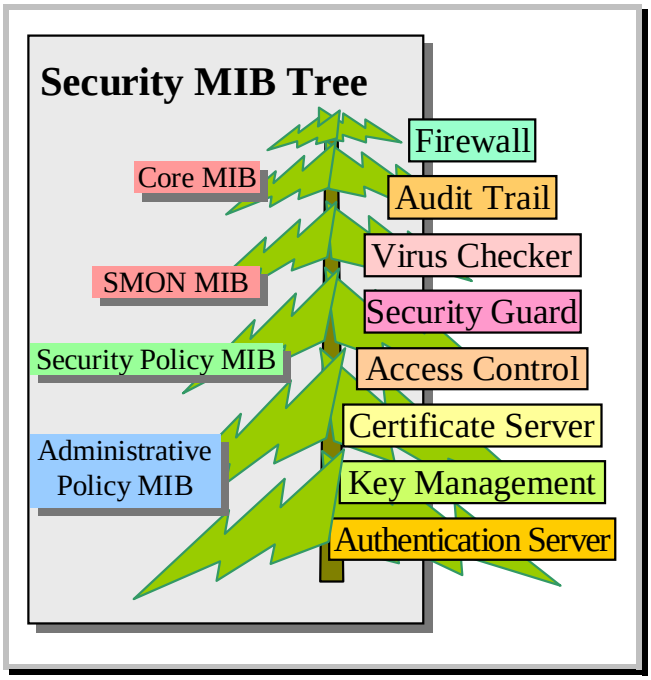

Figure 9. Core Security MIB with Plug-ins

raise an alert at the management station instead of waiting for the next polling cycle. The notification scheme needs to determine which alerts are to be defined and how they are triggered. Triggers that are too sensitive or too dull can cause an abundance or dearth of information. Either condition is sub-optimal. In addition, the amount of distinctness and diagnostic information included in each alert is an important consideration. More unique messages require more agent coding and sufficient information might be lacking to recognize a specific anomaly.

2.4.4. Security Monitor (SMON) MIB

The SMON MIB is inspired by the Remote Network Monitoring (RMON) MIB concept, originally defined in RFC 1271, then updated in RFC 1757 [71] and extended by RFC 2021 [72] and 2074 [4]. RMON is an increasingly popular standard that permits remote collection of network performance data and reporting of statistical results to central managers. RMON management goals include:

- Off-line Operation,
- Proactive Monitoring,
- Problem Detection and Reporting,
- Value Added Data, and
- Multiple Managers.

All of these remain as goals, but SMON adds secure reporting of transactions as part of its implementation for SM.

SMON maintains a real-time, active view of security event statistics, which can be polled by the management station or dumped at predefined intervals. Historical performance and event summaries may be compiled using audit trail and/or security log functions. They are oriented toward providing situational awareness and trouble-shooting support via activity-based statistics.

2.4.5. Security Management Information Protocol

An important aspect of SM beyond deciding what data to monitor and store at the local management station is propagation of event information beyond a single manager or security domain. This potential benefit must be weighed against the possibility of false messages or a perpetrator gaining access

to the information and thereby knowing the effect of his actions. The goal of event propagation is to make local events visible to approved parties to permit correlation and trend analysis between seemingly unrelated incidents. While the exact means of such correlation and analysis is outside the scope of this work, it is an important, complementary capability that may utilize CSSA management data.

Figure 10 illustrates the notion of information propagation based on the IPL in effect. In security domain A, an IPL3 is in effect, so only serious (emergency) events are forwarded to the manager and from the manager to neighboring stations. In security domain B, a serious event has resulted in declaration of IPL5, the highest level. In such a case, security applications may report a greater range of events (for a broader base of evaluation) and the management station may propagate the events more widely so that other managers are aware of possible dangers.

SMIP defines an external CSSA interface that uses SNMPv3 security services for manager to manager data updates between nodes. Chapter Six gives detailed SMIP procedures and transaction formats to convey information that may be relevant to the "big picture" amongst peer and/or hierarchical management stations. Both dissemination and query functions are proposed for SMIP. Broadcast or multicast methods would permit efficient distribution of selected warnings and announcements, while queries permit cooperative searches among correlation tools for related alarm data.

Too much or too little data sharing are undesirable extremes. Information starvation presents a stale view of conditions and too much data can affect network and system performance. The IPL state at

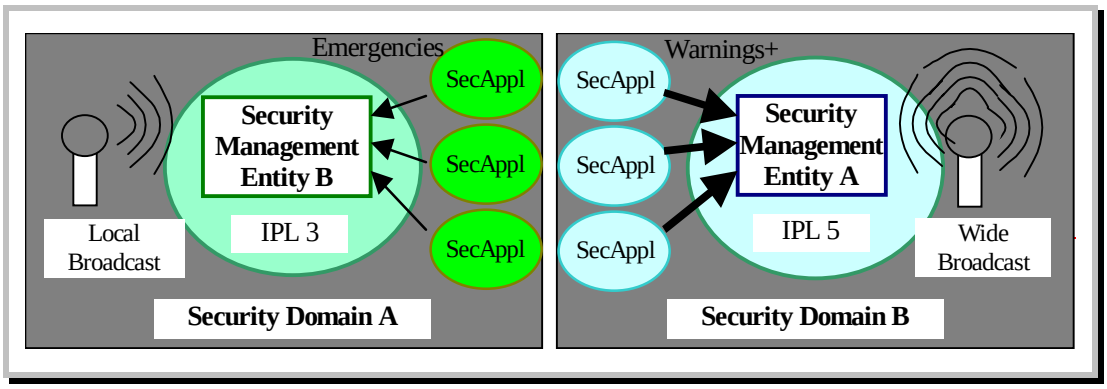

Figure 10. SMIP Propagation based on IPL

each management entity determines the type and frequency of information it sends out (see **Table 1** on page 13). A non-critical IPL of 1 may result in transmission of events only to the next senior management node. A more serious IPL may cause multicast of information to the senior node and to local neighbors. The IPL may also regulate the types of events that are announced to designated notification stations. At low IPL values, only the most serious incidents would need to be shared broadly, although everything should be logged in case of a future correlation effort. As the protective stance of sites or domains is increased, more routine events may be reported, since they could contain clues to a dispersed, low level attack.

2.5. Significance of the CSSA concept

The overview of the CSSA framework above presents a foundation on which to build integrated SM systems. Although other management frameworks exist, none have addressed the entire SM lifecycle from the administration phase through operations and assessment. The notion to summarize operational security statistics in the SMON MIB is a revolutionary idea that offers a fresh approach to tracking security events. In addition, the inclusion of the SMIP in the CSSA infrastructure along with use of IPLs and confederated host lists gives a unique means to share events among management nodes and to query other nodes for relevant information.

The ability to propagate security information and access remote event histories extends the vision of security operators. Without SMIP or an equivalent secure tip-off capability, security assessments must be based on limited local data, typically from one or two single purpose applications. Although dedicated management capabilities may be best for special purpose or isolated situations, the cost-benefit ratio for general use can be excessive. The CSSA concept offers a standard, efficient approach to dynamically monitor and control the operations of security applications.

In the coming chapters we will indicate plausible implementation scenarios and detail how certain aspects may function. While the CSSA framework, SMIP, SMON and SMIB are presented here as architectural proposals, submission to the IETF as Internet draft documents is a future goal. In this chapter, several innovations have been presented. The Security MIB, the Security Monitoring MIB, and the

Security Management Information Protocol are specific proposals for making standards-based security management possible.

Chapter Three—

CSSA Scenarios

The introduction to the essential elements of the CSSA framework in Chapter Two was presented in abstract terms. We now apply the CSSA concepts to real-world situations to establish a concrete foundation for implement. This chapter presents several scenarios in which aspects of CSSA can be seen to support enhanced awareness and control of particular security applications in realistic settings. We consider some simple, practical applications that help to justify our CSSA concepts. In choosing types of security applications for the scenarios, we subjectively assessed the most worthwhile for further scrutiny. The evaluation of the preferred scenario applications used a non-rigorous analysis of three factors - value of real-time management to the application, personal research interest, and expected proliferation (i.e. volume of potential implementations).

Table 2 lists the evaluation factors and tallies the ratings. Values ranged from 1 (lowest) to 5 (highest) for each factor. The top three scores, marked with an asterisk, indicate the applications with the greatest value for continued research. We present the virus checker, firewall, and security guard scenarios below. In the following examples, we endeavor to highlight the benefits of remote management of security applications and the difficulties of integrating SM into an operational environment. We also examine which parameters may be useful in tracking and controlling both normal and abnormal operations. Finally, we suggest where sharing application data between the configuration, operational, and assessment phases is currently possible and where it may be problematic.

Table 2. Security Application Assessment

Application	Real-Time Management (A)	Research Interest (B)	Proliferation (C)	Total (A+B+C)
Certificate Authority (CA)	3	2	1	6
Host System Security (AC)	2	4	5	11
Intrusion Detection System	4	3	3	10
Authentication Server (AS)	3	2	4	9
Key Generator (KG)	2	1	2	5
S- HTTP Server	3	4	2	9
Secure Audit Trail (AT)	1	3	1	5
Secure DNS server	3	3	1	7
Secure Email server	3	4	2	9
Secure Multicast	4	3	1	8
Security Firewalls* (FW)	5	5	5	15*
Security Guard* (SG)	5	4	3	12*
Virus Checker* (VC)	4	4	5	13*

3.1. Virus Checker Scenario

Our first scenario begins with a security incident detected by a security mechanism during normal operations. At 3:40 PM, an automated virus checker identifies and reports a virus imbedded in an email message sent to Alice of ACME Corp. from the XMU.edu domain. As an active security mechanism, the virus checker functions primarily in the operations phase of CSSA. Upon generating a notification to the control entity (as shown in **Figure 11**), additional functions may be triggered.

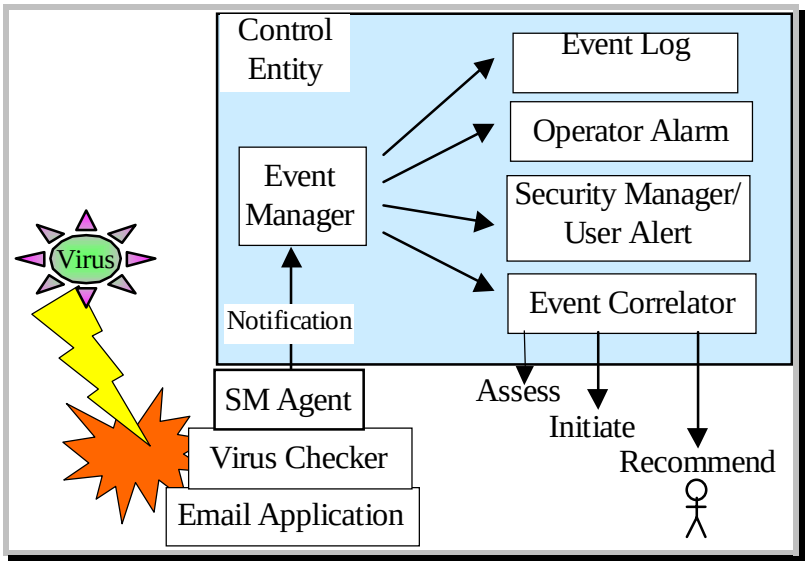

Figure 11. Response to Virus Incident

3.1.1. Event Manager

In the first stage of reaction to a notification, the event manager (dispatcher) module logs pertinent information (see **Table 3**) and may start an audible alarm at the network operator's console. The network operator, if on duty, reacts according to the severity of the alert and the importance of other on-going activities. In addition, based on the IPL and time of day, the event manager may alert the security officer via email (or pager) to review whether current actions are appropriate and whether executive management needs to be informed.

In a cooperative environment, other monitoring sites may be alerted at this point using SMIP to pass incident information that could represent a low-key, but broad-scale attack. Optionally, application users (e.g., email sender and recipient) may be notified that an incident has been detected. In some cases, it may be undesirable to release this notification information (at least to the perpetrator), because it may reveal more than desired about the detection and response capabilities.

Lastly, but most importantly, the event manager may signal the event correlator to evaluate incidents that are ambiguous or potentially attack indicators. Events that are redundant or known to be innocuous may be filtered out with only a log record retained.

3.1.2. Event Correlator

The signal to the correlator represents transition into the next phase of CSSA processing cycle

Table 3. Event Notification/Logging Format

Name	Value
Date/Time Stamp	Time of Incident
Reporting Address	IP Address of reporting entity
Type of Alarm	Security Event
Severity Level	(Critical, Major, Immediate, Warning, Minor, Informational)
SubEvent Type	Virus detection
Target Host	IP Address or DNS name
Target Resource ID	USERID or resource name
Suspected Attacker Host	IP Address or DNS name
Suspected Attacker ID	USERID
Comment Field	Additional event-specific details such as the type of virus

(i.e., security assessment). The event manager forwards data from the initial notification. The event correlator uses this data and other past information extracted from the event log or Security MIB, including alerts from peer control entities, to evaluate and recommend a course of action congruent with the desired security level. The event correlator returns an assessment to the event manager which logs it and possibly initiates an automated response or operator action recommendation.

3.1.3. Automated Response

Depending on the nature of the event, the correlator may initiate some protective measures automatically. Pending more thorough analysis, this could happen by interim configuration updates such as blocking connectivity from a dangerous site by updating packet-filtering rules. Automated responses, given the limited intelligence of most evaluation tools, should leave recommendations for weightier policy or configuration changes for human action. On the other hand, it is intuitive that some automated attacks may require automated responses to react properly and quickly enough to be effective. Automated updates may include damage mitigation that requires immediate action to prevent further damage. Examples of conservative responses might include disabling email from the suspected source address or isolating an infected disk volume. Care is needed to ensure protective measures do not cause loss of data or unintentional denial of service that is more harmful than the original incident.

When the correlator recommends a particular operator action, the operator may not respond promptly, or at all. Based on operator history, the security level, and the event severity, the response may be automated after some override period. For example, suppose ACME Corporation recorded another virus three months earlier that originated from the same XMU.edu domain. In this case, changes to the security policy or baseline configuration may be appropriate. Since XMU.edu is not a vital business partner, the event correlator chooses to block of all traffic from that domain. At this point, the relevant commands and information pass from the assessment phase into the administration phase of the CSSA cycle.

3.1.4. Validation and Reconfiguration

In the administration phase, a security administrator typically sets and updates parameters that control the operations of security mechanisms to match the security policies of the organization. Validation checks prevent inconsistent rules that might weaken security. In the CSSA concept, all security events can potentially result in configuration changes. Non-urgent changes are reviewed by the security manager and are implemented by the security administrator. An automated configuration change initiated in the assessment phase may bypass the manual process when necessary. All changes are logged as an internal security event. Since internal security actions are not assessed by the event correlator, this completes the scenario. **Figure 12** shows the sequence of processing through all phases of our model.

3.2. Firewall Scenario

The firewall scenario relates to activities predominantly in the administration phase of the CSSA cycle. It shows how CSSA notions of common data definitions for configuration parameters can improve interoperability, reduce training/retraining, etc.

The term *firewall* without further qualification is vague and is understood by everyone differently. Firewalls can take many forms and consist of various capabilities that an organization may desire to

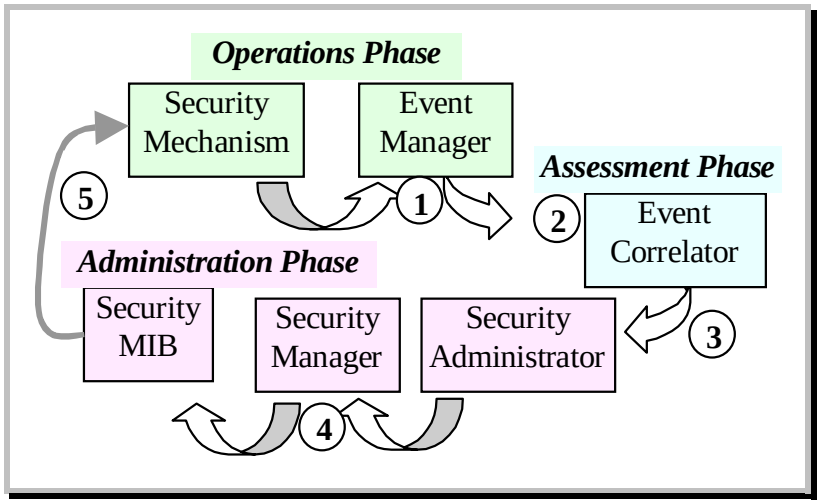

Figure 12. Event Processing Sequence

activate or not. Of the two basic types of firewalls, packet-filtering and application-level proxy servers, the trend is toward use of the more rigorous capabilities of the proxy server. Performance issues have lead vendors to announce dual mode firewalls that can act in either mode, based on performance needs [59]. The management functions, which are available to oversee one or more security applications, should be adaptable enough to recognize all firewall capabilities and to take advantage of all parameters to adapt the configuration match the conditions.

In scenario two, Bob is configuring the firewall component for a new remote office LAN as outlined in **Figure 13**. Proxy services need to be implemented based on the corporate security policy specified in **Box 1**. Typical Internet and Intranet services will include a local Email server, internal FTP to the corporate FTP server, outgoing HTTP and Telnet, and Domain Name Services (DNS). Because of the small size of the branch office and a desire to minimize costs, the firewall setup will use router-based packet filtering and the TIS Firewall Tool Kit (FWTK).

Bob finds that the design of packet filtering rules and implementation of appropriate proxy services is a daunting task, even for an experienced network engineer. Existing tools do not use common conventions and each stores their configuration data in different formats. Although some vendors have delivered configuration tools to simplify creation of rules, Bob is forced to deal with the arcane configuration methods built into the FWTK software. Although Bob has worked with another firewall product, the consistency in rule syntax across vendors is low. Without a reliable means of rule verification, this leads to errors, a need for retraining, and vulnerabilities.

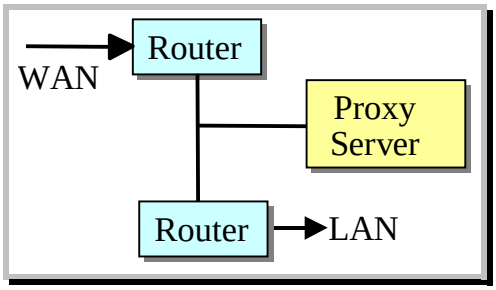

Figure 13. Remote Site Firewall

Box 1. Security Policy

All services not specifically required for operations will be disabled.
All Email and FTP traffic to/from government sites will be encrypted.
Strong authentication is required for dial-up remote access service (RAS).
Incoming TELNET and FTP is prohibited except for FTP to the FTP server on the bastion host.
Passwords and user keys must be updated at least every six months, must be at least 8 characters and be of medium randomness.
System administrators must re-validate all accounts at least every six months.
No auto forwarding of email is permitted to external accounts.
All outgoing Email will be checked for security releasability (dirty word check).
All incoming email, FTP and HTTP traffic will be virus checked
No site access is permitted from non-US domains.
More than three security events from a single domain will require explicit authorization for continued access.

Given that Bob has established an initial security configuration, the next step is to install and test it. This process opens new possibilities for errors and undiscovered faults due to limited testing rigor. While a number of vulnerability testing tools such as COPS, Tiger, and SATAN have been useful to detect site configuration problems, they are static tools and do not test configurations against local security policies. In addition, none uses local security events as input to dynamically tailor assessments. That would require standardized security policy formats, common event and configuration parameter definitions, and new, smarter verification tools.

Another approach for validating security configurations is simulation of the operational environment using modeling tools. A simulation environment for SM similar comparable to those that exist for network modeling would permit the effect of changes to be considered under a broad range of assumptions about the threats and vulnerabilities. Mechanisms could be evaluated under increasing levels of stress to determine performance and points of failure. Such a security simulation environment is not currently available, at least not publicly. It would require modules to simulate performance and functionality of each category of security mechanism. Standard parameters would be needed to define critical performance capabilities and measurable values.

Once the validation of Bob's newly developed security configuration is complete, approval and deployment can proceed. Audit records must capture who makes each change, when, why and how secondary approval, if required by policy, was accomplished. This information becomes a permanent part of the site security log for future reference. An integrated administration/configuration management tool would make this phase of SM much easier, but Bob uses the only tools he has — manual configuration.

3.3. Security Guard Scenario

In a Multi-Level Security (MLS) environment, security guards are designed to provide a trusted interface for restricted data flow between two security domains that implement different security policies. Typically, the difference in policies may involve different classification levels or merely two user communities with different need-to-know restrictions. In our final scenario, a government agency is planning for remote management of a security device that will provide a trusted interface between systems running at two different security levels. **Figure 14** shows the main components of the security guard.

Active supervision of the information security guard function will minimize risk of malfunctions, but minimal human interaction is another desired feature. Typically, a semi-automated review process may include automated checks for data integrity, template matching and filtering for unreleasable words or phrases, followed with a final human review of images or graphics. In the present case, a security guard with highly advanced releasability evaluation software will apply a set of rules (see **Box 2**). This permits only properly marked, formatted, and signed data items to pass from the higher security domain to the

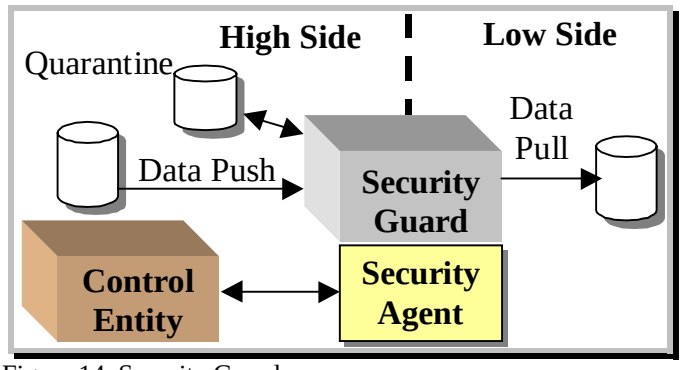

Figure 14. Security Guard

Box 2. Security Guard Processing Rules

Is item to be released in approved data format?

Are data fields in item properly marked and approved for release?

Are all values within ranges allowed in value template?

Do all fields meet custom filter rules?

Is item to be released signed by an approved releaser?

Is released item encrypted with the certified public key of approved recipient?

lower domain. As an additional precaution against inadvertent third party disclosures, the outgoing data is encrypted with the public key of the recipient. This provides a level of "need-to-know" protection for valid releases and limits the scope of distribution for any improper releases. The guard will operate independently given a proper configuration, but will generate status and alarm information to indicate its operational health and any abnormal conditions. To minimize the logic required within the security guard, the CSSA monitoring actions will be initiated by the control entity except for security event notifications.

The first task before beginning operations of the security guard is to configure required parameters. There are two types of parameters — mechanism administration data and operational use data. The first type of data determines how the security device will communicate and operate. Operational use data controls the basic guard functions to validate releasable data. **Table 4** lists some representative administration and operational use data items. Configuration data that is unique to the security guard may be stored locally for immediate access. Some configuration data may be used by several mechanisms besides the guard, so it is stored centrally in the SMIB and accessed on demand. The list of authorized releasers and their public keys helps to document an audit trail of release actions using digital signatures.

Once configured, the security guard can be tested and placed into operations. To monitor the

Table 4. Configuration Parameters

Administration Data	Operational Use Data
Notification Destination(s)	Custom Filter Data
Control Entity Public Key(s)	Data Value Template
Local Agent Private Key	Authorized Releasers
Throughput Threshold	Public Keys of Releasers
Input Buffer Threshold	Approved Destinations

guard operations for maximum benefit, two measures of quality are paramount. The first priority in security applications is effectiveness of the solution. Therefore, the number of false negatives (releases allowed that should not have been) should be zero (or at least, closer to zero than a human would allow). This prevents catastrophic release failures that negate all potential benefits of the system. The number of false positives (releases prevented that were actually okay) should be low also. This factor determines how efficient the system is. If too many releases are stopped when subsequent manual review shows no problem, it indicates inadequate intelligence in evaluation algorithms. Manual review results in lower efficiency and slower payback of the technology investment. Once the security guard is accredited as effective and is reasonably efficient, system operations can begin.

The administration and operational use data defined above is used during operations to guide how the security guard functions. If a receive buffer for data nears capacity, an alert may be generated to the control entity to slow input. If the throughput of the guard and the security threat are low, then a less intensive screening process may be set. For example, if several files are backlogged while waiting full processing, some non-essential, time-consuming steps may be skipped to speed things up.

Other events from the security guard may signal the control entity of problems like quarantined files that fail security screening and require human intervention. Reasons for flagging files and the results of manual evaluations can be logged and translated into revised configuration data that will permit better processing in the future. Even if the tools to implement the cyclic processing of SM data are not highly automated, the process of continuous feedback and rule enhancement should improve responsiveness and quality of operations over time. The critical factor for enhancing system performance is to use assessment results to update configuration parameters or relevant security policies. While the feedback process may occur manually, synergies can grow as automation permits better linkages between CSSA phases and security applications.

Chapter Four—

CSSA Infrastructure (High-Level Design)

To realize the goal of reliable secure systems, an effective security assurance infrastructure is crucial. Conventional wisdom says that any complex system inherently has undetected faults (vulnerabilities in the security context); therefore, security policies and mechanisms alone are not enough to protect vital resources from determined intruders. Although simulations, developmental testing of software components, and verification of initial configuration may be important, we focus on the active methods to detect suspected security breaches during operations.

An infrastructure is the underlying foundation or basic framework of a system. The SM framework helps to organize and rationalize the supporting elements behind the operational concept, in spite of a potentially hostile environment that may include many active components and complex interactions. As discussed in Chapter Two, security management must maintain specific criteria for security and trust while adding value and providing flexibility to accommodate the types of applications offered by vendors and services needed by different users. A modular infrastructure, as we present here, has the advantage that it can be augmented and tailored to current and future needs without as much operational disruption or replacement of systems.

In this chapter, we establish the high-level design of the necessary capabilities, functions, and actions that contribute to the CSSA goals of effective and efficient management of generic security applications. In large measure, we use the network management paradigm as a model because it has a

similar operational context and it has been generally successful. We use functional process decomposition and IDEF models to detail our understanding of security management needs (data flows and processes) before allocating them to infrastructure components. Then we discuss both existing infrastructure capabilities and those that need to be developed, in order to weave together the pieces of an integrated design. Detailed development of some specific proposals for new capabilities is reserved for the next chapter. We also identify other missing or weak capabilities with suggestions for future work. We consider transition issues of moving from present processes and systems to the objective framework in Chapter Seven.

4.1. Infrastructure Driving Requirements

A wide variety of products and vendors are competing for pieces of the security services market, many with unconfirmed protective value. Although this situation is changing with establishment of industry testing standards, there may still be concern as to whether the right security tool is properly configured and operating correctly. The diverse and competitive security marketplace makes integrated management a much greater challenge, but also a more valuable goal.

Standards-based tools offer the only real hope of enjoying the potential benefits of cross-product interoperability and mutual situational awareness between applications and security domains. The ultimate solution for security management should satisfy the following general requirements:

- Use generic security services through standard APIs,
- Seamlessly interoperate with each security product on generic platforms,
- Use consistent configuration management and policy rules,
- Display a consolidated status in the users' choice of view and level of detail, and
- Correlate multiple events in real-time despite divergent types, sources, or times of occurrence.

4.2. SM Functional Decomposition

There are many perspectives from which to view the SM function. Earlier, we presented a three-phase action cycle. The following section uses Integration Definition, type zero ($IDEF_0$) diagrams to

dissect and clarify the processes, resources, and data flows that are inherently part of a SM system. This decomposition will help to identify the types of data used across the three SM phases.

4.2.1. Lifecycle View

The lifecycle of a security system includes several stages of development, testing, deployment, operation, and maintenance. An organizational security program must support all stages for a specific implementation and each stage requires a quality control function to support security assurance. **Figure 15** shows four high level stages of a system lifecycle. With the CSSA concept we are most interested in the active security management processes that are centered on the last stage, Operate Security Systems.

Figure 16 further breaks down an operational security system into four component processes using an $IDEF_0$ diagram. The three phases of an operational SM system that we identified in Chapter Two are shown as:

- Administer security privileges and controls,
- Manage security operations, and
- Evaluate security events.

However, an additional block (A42), called "Perform Security Services", recognizes the real purpose of the overall security system and is shown for completeness. Each security mechanism implements part of this process according to its design. The SM processes interface to specific security

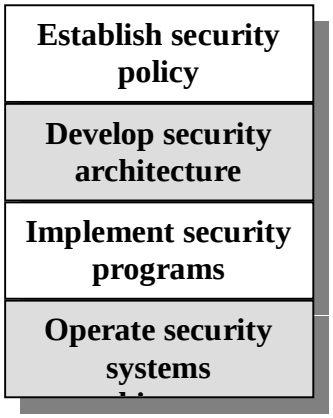

Figure 15. Stages of SM Lifecycle

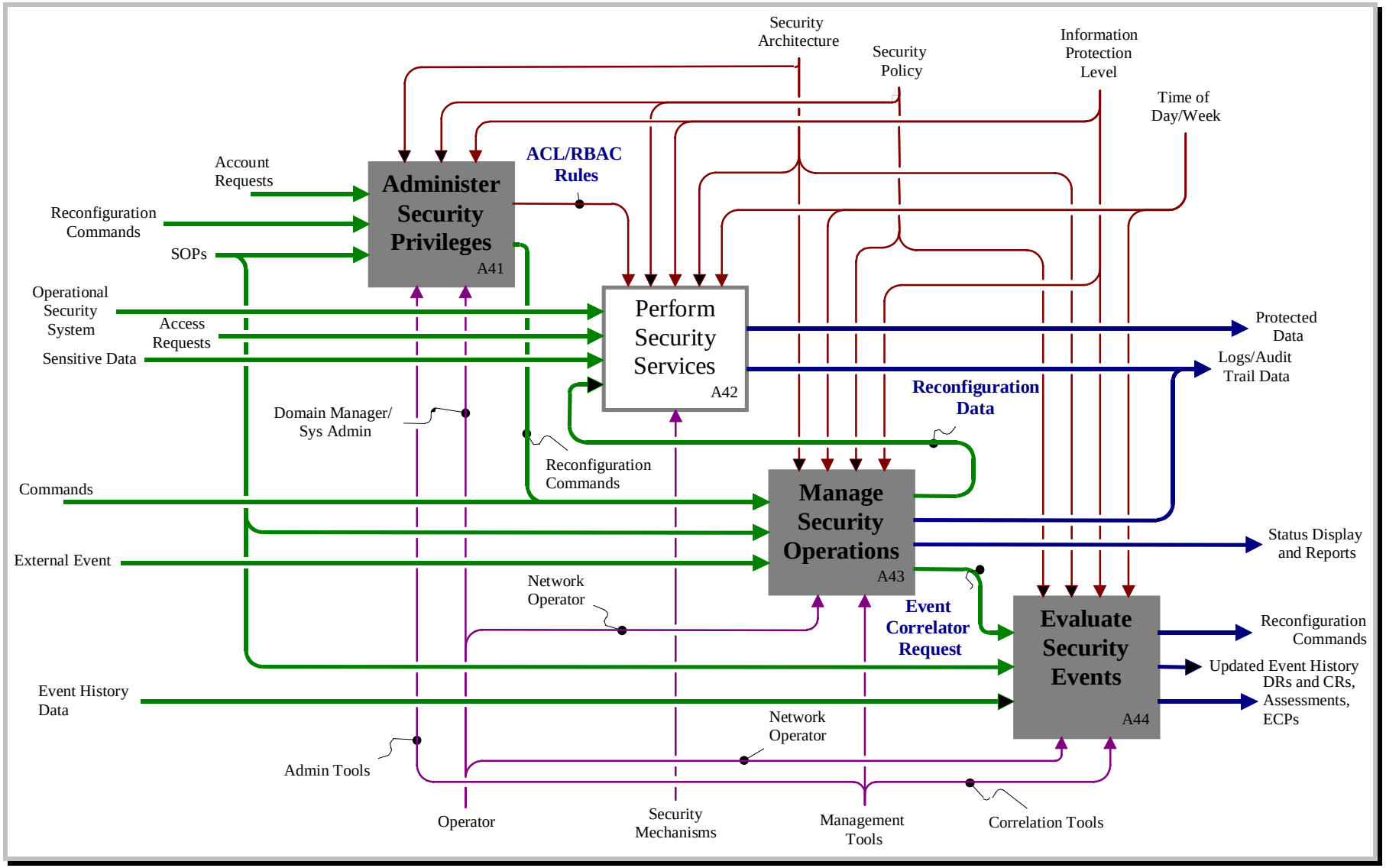

Figure 16. Operate Security System IDEF$_0$ Diagram

applications via an SNMP agent. Thus, we discuss only the three processes that correspond to our SM phases. Each of these phases has data inputs entering on the left, outputs exiting on the right, controls/constraints on the top, and methods/resources on the bottom, as shown. Although our focus is on the management functions, it is important to remember that the objective of security management is to develop user confidence that protection services do reliably change sensitive input data into protected data output.

Another important distinction in **Figure 16** is the difference between the off-line security administration function (establish privileges and controls), the real-time operations functions, and the linkages between them through the assessment function. Although each has different inputs, outputs, constraints, and methods, but there is also a strong degree of commonality among the data flows. The required data flows between the phases are the motivation for the design of the security MIBs and SMIP discussed later. Of particular interest in the next level of detail are the processes within the three management phases. The administration, operational and assessment sub-functions are shown in **Figure 17, Figure 18**, and **Figure 19,** respectively.

4.2.1.1. Administration

The administration breakdown in **Figure 17** shows three parts to the process. First, all requests for new access or requests for modification of existing privileges or access rules must be validated against the current security policy. If the security policy rules are very explicit and are captured in a structured fashion (e.g., Security Policy Specification Language), this may be an automated function. Whereas changes to the top-level security policy may be controlled by the senior security manager, validation of user privileges and device configuration changes may be delegated to a designated subordinate security or domain manager. In the past, validation of change requests with policy was frequently manual and sometimes somewhat arbitrary. If policies are not documented in precise detail, consistent decisions on compliance are difficult to achieve and verify automatically. This is a critical area and additional work in automatic validation and policy management tools is warranted to reduce the tedious and error-prone task of checking for consistency and completeness.

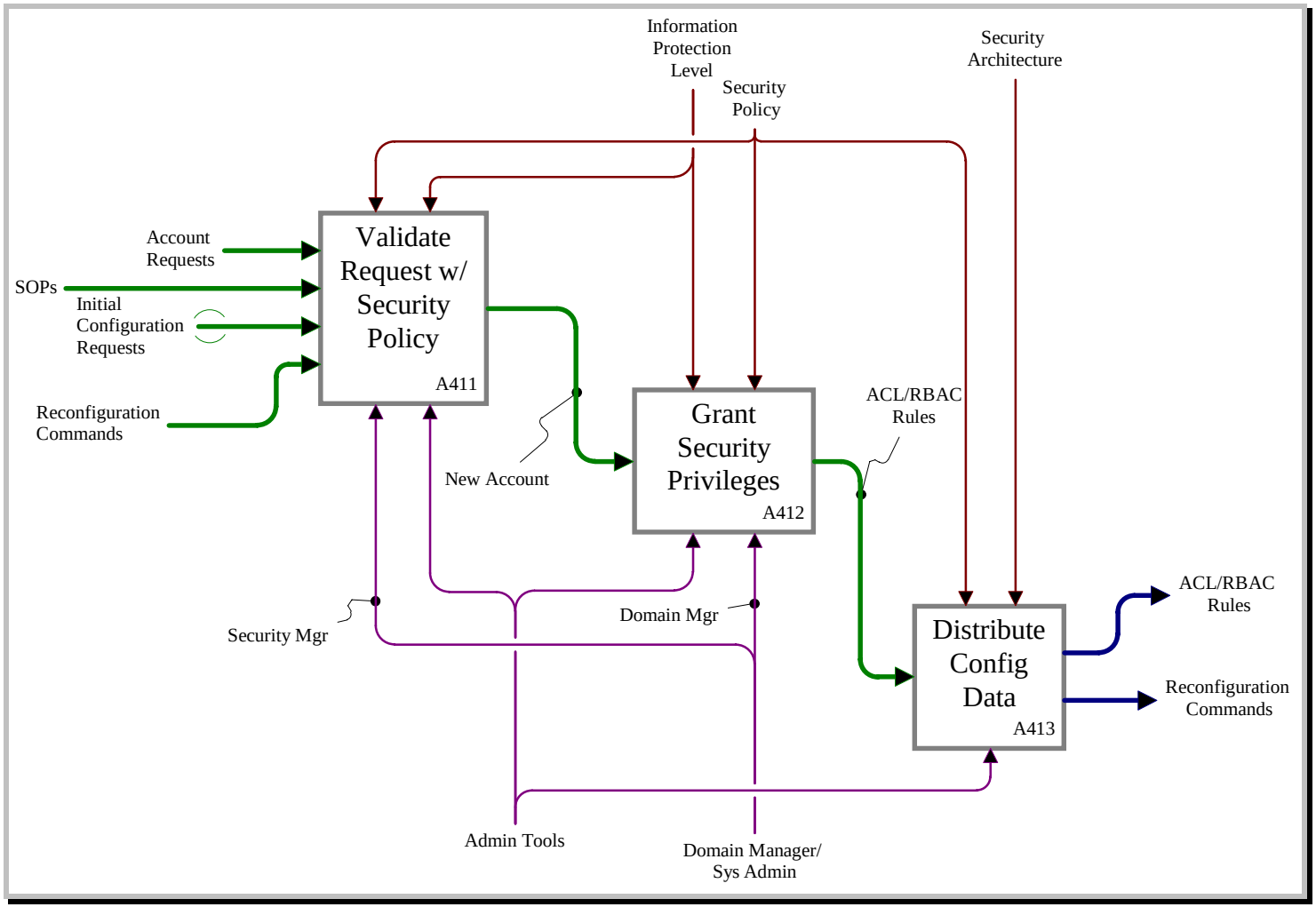

Figure 17. Administer Security Privileges IDEF0 Diagram

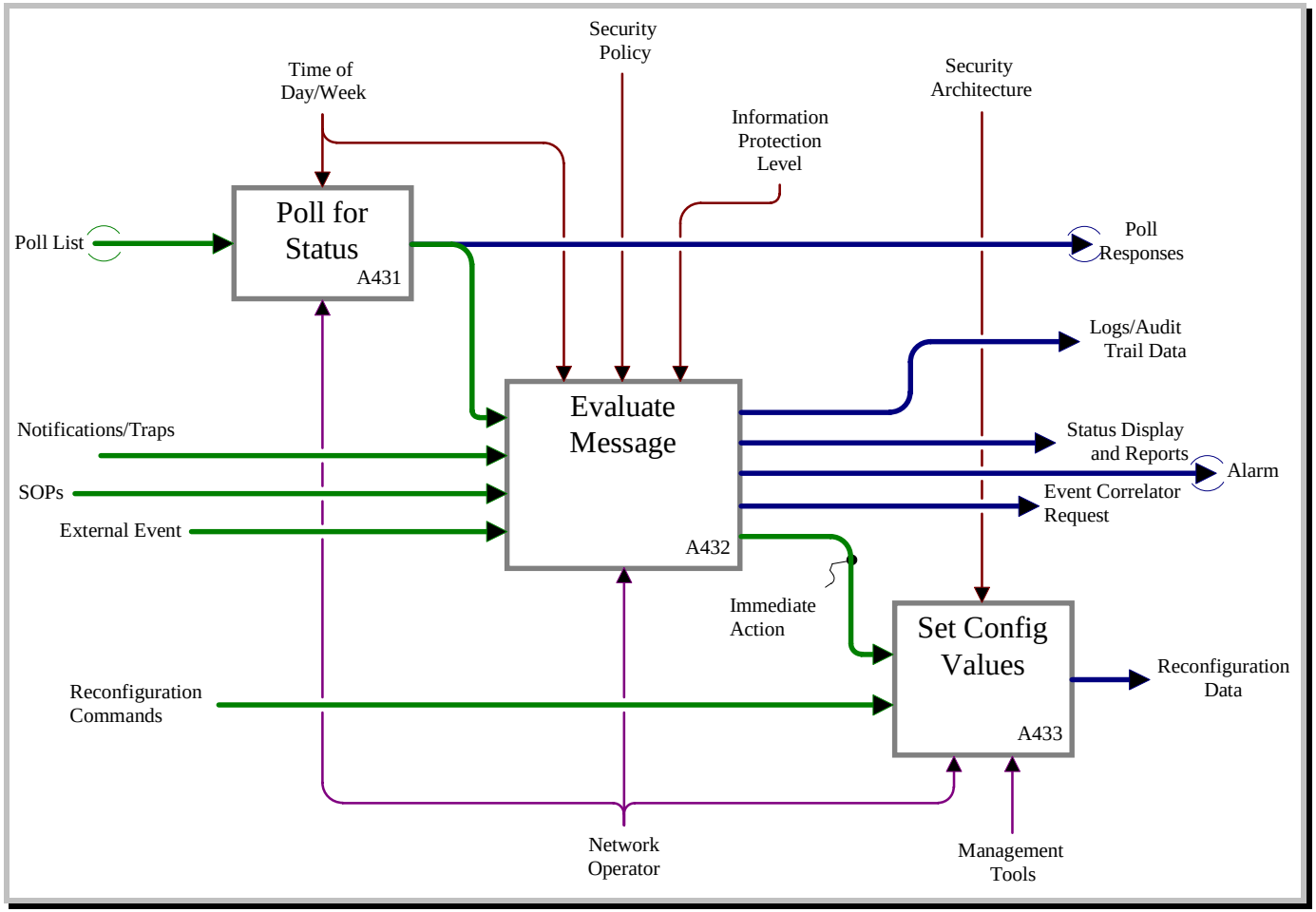

Figure 18.Manage Security Operations IDEF0 Diagram

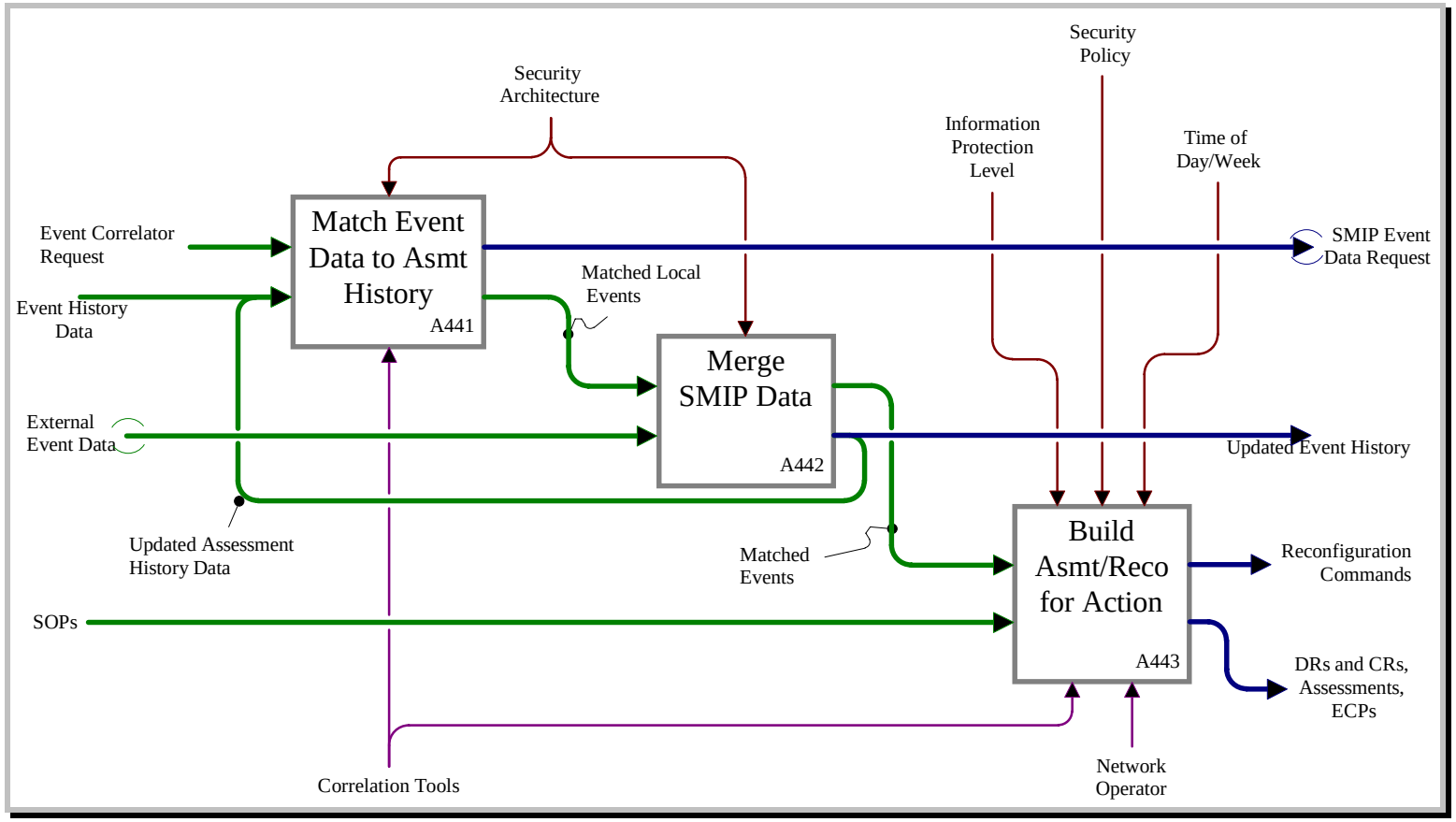

Figure 19. Evaluate Security Events IDEF$_0$ Diagram

The second step in the administration phase is the process of granting privileges. Clearly, validation from step one is necessary, but the diagram also shows that other factors such as the current Information Protection Level may affect the access control rules that are generated. In addition, granting of privileges may involve functional domain managers or system administrators, not always a subordinate security manager. The degree of flexibility and authority given to these administrator roles may also vary by the IPL or at the discretion of the senior security manager.

In the third process block (A413), "Distribute Configuration Data", the resulting Access Control List (ACL) or Role-Based Access Control (RBAC) rules are distributed to the implementing security mechanisms. This is necessary because access control, event logging, and implementation of other security services occur in the security applications and they must be configured reliably and securely. All security applications need access control with respect to their own management functions. This remote configuration would presumably use SNMPv3 Set commands. Other security applications implement external access control services and need more extensive ACL/RBAC configuration capabilities. The SMIB addresses the first access control capability, while the Firewall MIB (discussed later in Section 5.3.1) provides an example of more advanced access control functionality.

4.2.1.2. Operations

The "Manage Security Operations" function is the primary interactive element that monitors and controls security applications. A system operator initiates ad hoc actions while automated scripts guide routine operations. As seen in **Figure 18**, monitored nodes are routinely polled for status based on a polling list and the time of day, or by operator initiation. Other command inputs, such as the "Set Configuration Data" from the administration phase, are transferred to the appropriate security application via SNMP Set commands. Incoming event messages (poll responses or notifications) to the operations function trigger one or more actions that range from ignoring it, to log and display, or to an immediate configuration action. As indicated in Chapter Two, the IPL, event type and severity, and operator responses can all affect the outcome. Frequently, a referral action to the event correlator allows more complex assessment of the event. One of the unique features of this approach is that all compatible

applications can generate a security event, not just dedicated security applications and probes. This offers a great opportunity for innovation by general application vendors, not just those who build IDS and firewalls, to expand the view of the security operator.

4.2.1.3. Assessment

Functions of the assessment phase are allocated mostly to the event correlator module. A correlation request provides the basic who, what, where, and when of the event, so that a simple local match can be attempted. As depicted in **Figure 19**, the assessment function is expected to identify similar events from the local audit history or cooperating CSSA node histories that might indicate suspicious activity. SMIP procedures define how event data external to a SM control entity is requested and shared by SM entities. The last assessment step shown in **Figure 19**, "Build Assessment/ Recommendation for Action", is treated here as a "black box". Interfaces are defined for the specified inputs, outputs, controls, and resources so that internal functions can be updated or replaced without disrupting other areas. Implementation could be a rule-based expert system, a learning-capable neural network, or other advanced assessment system. The methods and logic used in the correlation function for sorting, matching, and ranking the security significance of events and sequences of events is a rich topic of research in itself and is not considered further here. Our interest is only that a standard interface be established to permit other SM functions to interface with the correlation tool. Depending on implementation by developers, the management module to correlator interface may be defined as an API or distributed computing protocol. We discuss the correlator interface further in Section 4.3.2.

4.2.2. Data Flow View

An alternative to the process flow discussion is to trace the use and sharing of data elements by the SM functional elements. Rather than going through the IDEF diagrams again, we tabulate the essential details in **Table 5** below. In several cases, data elements are created in one SM phase and used in one or more phases. The IPL parameter is a good example of a widely used value that is an obvious candidate to include into the core Security MIB.

Table 5. Data Usage and Sharing

Data Element	Created By	Updated By	Constrained By	Used by
ACL/RBAC Rules	Administration	Administration	Security Policy	Operations
Event History	Security Applications (Operations)	Event Correlator (Assessment)	Matched Events	Assessment (Event Correlator)
External SMIB Data	Security Applications (Operations)	Administration Manager		Assessment (Event Correlator)
Information Protection Level	Senior Security Manager (Admin)	Event Correlator with Operator	Security Policy	All
Logs/Audit Trails	Administration, Security Applications (Operations)	Event Correlator (Assessment)		Assessment (Event Correlator)
New Account Requests	External Entities	Administration	Security Policy	Administration
Notifications	Security Applications (Operations)	N/A	MIB Definitions	Assessment (Event Correlator)
Polling List	Security Operator	Security Operator Event Correlator	IPL	Operations (Event Manager)
Reconfiguration Data	Event Correlator (Assessment)	N/A	IPL, Security Policy	Administration
Security Policy	Senior Security Manager (Admin)	Senior Security Manager	N/A	Administration
Sensitive Data	External Users	Security Applications (Operations)	Security Policy	External Users (as Protected Data)
SNMP Commands	Administration, Security Operator, (Operations), Event Correlator (Assessment)	N/A	N/A	Operations
Status Reports/Display	Event Manager (Operations)	Event Correlator (Assessment)	IPL	Operations (Security Operator)

4.3. Objective Architecture

In an ideal world of unconstrained budgets, schedules, and technical expertise, perfect network security might be achievable, although no resources or customers may remain to do useful work. In the worst of all worlds (from a networking point of view), perfect network security is achievable simply by disconnecting from the network. Our objective architecture aims somewhere between these two extremes, with preference for open standards, layered defenses, effective detection, and adaptive response to intrusions. To achieve such challenging goals, a sophisticated support structure is required. **Figure 20** (a more detailed version of **Figure 6)** illustrates how a group of security capabilities may satisfy a set of solid foundation for further work. Three of the six CSSA infrastructure elements shown in **Figure 20** (SNMP,

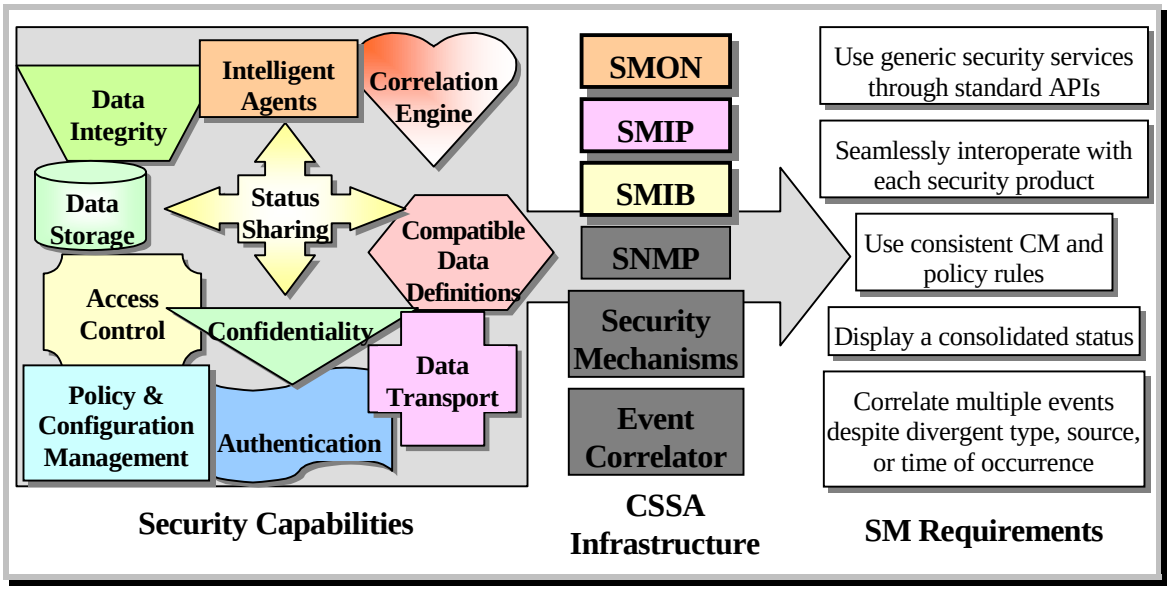

Figure 20. Function of CSSA Infrastructure

Security Mechanisms, and event correlator) are existing or in development by others. requirements[3] via the CSSA infrastructure. Each SM capability maps onto a CSSA infrastructure element that meets one or more requirements. **Table 6** is an expansion of each Capability–Infrastructure–Requirement mapping. By identifying a component to satisfy each SM requirement, we may create a SM toolkit that is useful in the future. The other three items (SMON, SMIB, and SMIP) are the primary focus of the CSSA concept in this dissertation. We discuss each of the CSSA components next, but concentrate on the designs and engineering decisions related to the SMIB, SMON, and SMIP components.

4.3.1. Compatible Data Definitions - SMIB

A major shortfall, if not the primary inhibitor of security management interoperability, is the lack of agreement on common data definitions to enable common use of security indicators and actuator commands among security applications. Although many security mechanisms may perform certain common actions and define similar parameters, a cross-mapping of similar performance and configuration variables between security applications has not been accomplished. A complete mapping effort would be a

[3] Reference here is to a set of general requirements, rather than a large specification with many specific, detailed requirements.

Table 6. SM Capabilities to Requirements Mapping

Security Capability	CSSA Infrastructure	SM Requirement
Access Control	SNMP	Seamlessly interoperate with each security product
Authentication	SNMP	Seamlessly interoperate with each security product
Confidentiality	SNMP	Seamlessly interoperate with each security product
Correlation Engine	Event Correlator	Display a consolidated status
Compatible Data Definitions	SMIB/SMON	Correlate multiple events despite divergent type, source, or time of occurrence
Data Integrity	SNMP	Seamlessly interoperate with each security product
Data Storage	SMIB	Use generic security services through standard APIs
Data Transport	SNMP	Use generic security services through standard APIs
Intelligent Agents	SMON/SMIP	Use consistent CM and policy rules
Policy & Configuration Management	SMIB	Use consistent CM and policy rules
Status Sharing	SMIP	Display a consolidated status

daunting and probably never-ending task. Nonetheless, we begin a limited effort below to demonstrate that a common core security MIB is feasible.

Table 7 indicates several management objects that may implement similar functions across two or more security mechanisms. Any CSSA-compliant device (agent or manager) would have to implement these as core SMIB data items. For example, all mechanisms should have a remote command interface to permit action commands. The SystemCommand parameter is an authenticated command with integrity checks, which can implement common actions such as master reset, clear counters, start security diagnostics or dump logs. The Information Protection Level is another parameter that is set by the controlling management station. It is used by local security applications to affect how conservatively devices implement their security functions and how much logging/reporting to do. Other control parameters may cause other useful actions such as key management, filter table updates, or revocation of access control list records. On the other hand, read-only parameters may only *report* the health or performance status of security devices. In Chapter Five, we discuss specific portions of the SMIB that implement core management parameters. We expect that after further coordination with interested parties via Internet user groups and through publication of Internet drafts, the final definitions could be formalized as a RFC document.

Table 7. Common Command and Status Parameters

Command / Status Parameter	Format	Mech-anisms	Values/Description
SystemCommand	W	All	Master Reset: Restart/warm boot so that system uptime begins from zero and all values are set to defaults, Clear Counters: Reset all to zero, but system uptime is not changed, Security Diagnostic: Mechanism does a self diagnostic to verify nominal functionality, Dump Events: Bulk delivery of selected events at this device
System Security Administrator	R/W	All	A textual description of the responsible security authority for this device
Information Protection Level	R/W	All	The protective posture designated for the device.
System Security Services	R/W	All	Security services that are configured and available for use
Last Security Event	R	All	The most recent event relevant to security
Time of Last Event	R	All	Time of last security event (date/time)
Time Since Previous Event	R	All	Time since last security event (days, hours, minutes, seconds)
Security Alarm	Trap	All	Sent when a notable security event is detected. The varbind field identifies the warning level.
DES Management Key Update	W	All	DES key for SNMP operations (initial load must be done physically)
MD5/SHA Key Update	W	All	SNMP Authentication/Data Integrity Key (initial load must be done physically)
Log Table	R	All	This is a record of local events that MAY get reported to the security manager depending the IPL and the event type/severity
Traffic Key Update	W	CA, KG, AS, SG	An action command to initiate changeover of keys used for confidentiality must be refreshed periodically
Last Key Update	R	CA, KG, AS, SG	Date of last Traffic Key update
Suspects Table	R/W	AS, CA, FW, VC	List of IP addresses, applications and ports that require extra attention
R/W- Read/Write	AS- Authentication Server CA-Certificate Authority FW- Firewall		KG- Key Generator SG- Security Guard VC- Virus Checker

As compared to the common parameters above, **Table 8** lists selected parameters that correspond to a number of application-specific MIB tree extensions for individual mechanisms introduced back in **Figure 9**. Although core MIB parameters are mandatory for implementation, unique applications may add unique features, as needed, using mechanism-specific MIB plug-ins. Any application-specific parameters that become useful in more than one mechanism (e.g., filter tables for Host-based Access Control and Firewalls) would be candidates for migration into the core SMIB. Chapter Five will discuss specific MIB

Table 8. Mechanism Specific Parameters

Command / Status Parameters	Format	Mecha-nisms	Description
Access Control List	R/W	AC	Rules that mediate access to host-based resource
Directory with Public Key Information	R/W	AS	Permits validation of message origin or encryption of data for specific recipients
Zeroize	W	KM	Purge key(s)
Dirty Word List	R/W	SG	The list of words that the security guard looks for in data to be released
Packet Filter List	R/W	FW	The rules by which protocols and ports at particular IP addresses are opened or blocked based on packet header data
Revocation List	R/W	CA	A list of certificates that are no longer valid.
Virus Profile List	R/W	VC	Virus patterns used to identify viruses

parameters for our proposed Firewall and Security Guard MIBs, which are fully specified in Appendix C and D, respectively.

4.3.2. Correlation Engine

In our CSSA model, the event correlator implements most of the assessment portion of the SM cycle and is an important feature of the assurance capability. Some frameworks consider correlation of events as being a part of the intrusion detection function, which is often built around an audit trail baseline. We take a broader view by considering that the event information for correlation should originate from any security mechanism including, but not limited to, IDS modules. Whereas, both centralized and distributed IDS designs have been suggested [24] [33] [73], our vision of a correlator is only of a module within a management node which may be configured to share information in hierarchical or peer arrangements. Although IDS assessment techniques are beyond the scope of this research, we are interested in defining the interface specification between the correlator and the rest of the management engine.

4.3.2.1. Event Manager to Correlator Interface

As depicted in **Figure 21,** the event correlator interacts with the event manager, the event log, and the administration manager. The assessment function begins when the event manager passes an Event Correlation Request (ECR) (see **Box 3**) to the correlator. The ECR contains the identification of the originating management station, the current IPL, and an urgency indicator, and the original notification.

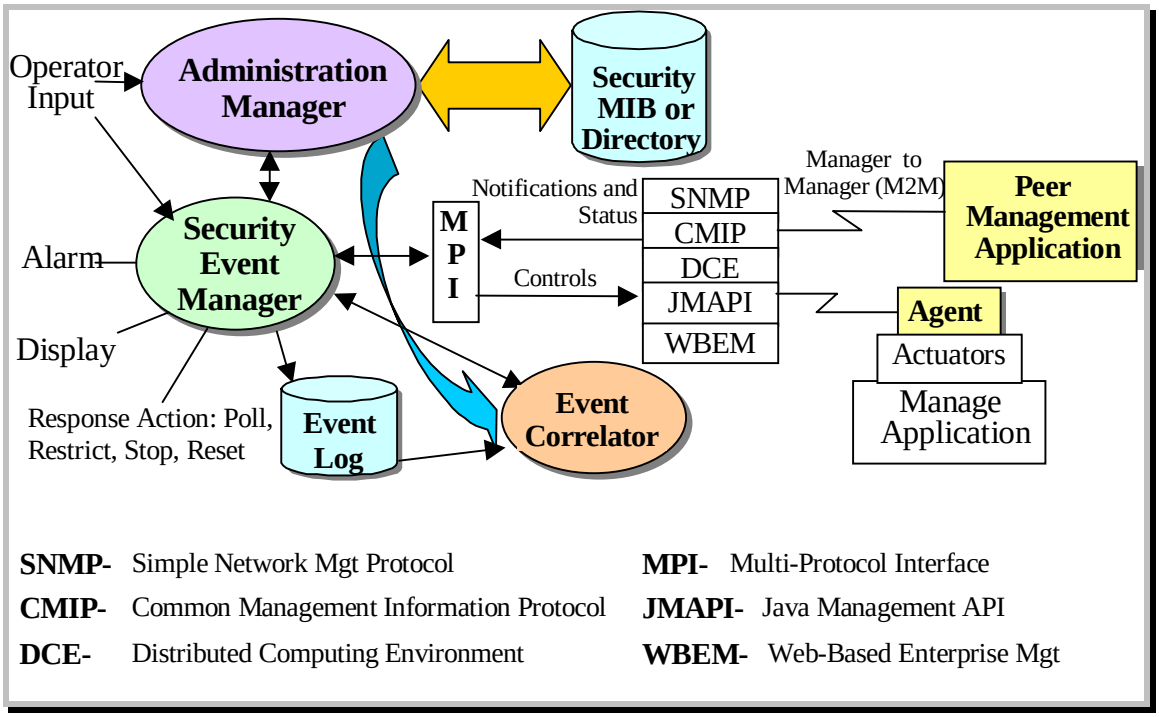

Figure 21. CSSA Infrastructure

The basic assumption upon issuing an ECR is that the event manager does not have enough information or resources to evaluate the event. Based on the event type and severity tags in the SNMP notification, the event manager can determine whether to log, display, or initiate certain response actions. However, when additional context or background processing is needed, the correlator is where it would happen.

4.3.2.2. Correlator Processing

The event correlator has direct, read-only access to the event log in which most, if not all, local security and data processing events are recorded by the event manager or administration manager. The

Box 3. Event Correlation Request Format

Originating Manager:	IpAddress
Current IPL:	Integer (1..5)
Response Urgency:	Integer (0..10)
Event Notification:	SNMPv2 Trap PDU

correlator implementation may also access audit trails and other logs created by non-SNMP processes and mechanisms. If the correlator needs to request data from the local SMIB or from remote sites, it works through the event manager.

The event manager acquires data from the local SMIB through the administration manager. Remote data is obtained via an SMIP Event Data Request (EDR) to a senior node, trusted neighbors, or a list of confederated management nodes. The "confederated" peers are trusted at some level that has been arranged off-line. **Box 5** on page 87 identifies the SMIP EDR format and processing is discussed in Section 6.1.2.

4.3.3. Status Sharing - SMIP

In a reasonably complex security domain with multiple security devices and secure applications located across several networks and distributed systems, computer-assisted remote configuration and monitoring is likely to be the only feasible way to maintain timely visibility into performance and safety[4] of operations. As security attack scenarios become ever more aggressive, subtle, and complex, intelligent assessment capabilities become even more vital. SMIP, introduced in Chapter Two, enables controlled data flows between SM entities responsible to collect, assess and summarize security data. The security data may originate at access control, firewall, IDS, virus checker, or other "traditional" security applications, or new experimental autonomous agents that may not be associated with a security service/mechanism. The key factor is the means to pool the information through common access and distribution of SM data. SMIP defines an external interface between peer management entities to request and update data about current status[5] and historical transactions. SMIP consists of data structures and procedures that effectively make all SMIB data in a security domain accessible through a single virtual database. SMIP supports two types of transactions, security event notifications, and security history queries, which are discussed further in Chapter Six.

---

[4] We use the term safety here in the sense of correct execution of the intended operations.
[5] The definition of *current status* should be configurable by the administrator to a relevant value such as the past 24 hours, last 60 minutes, or most recent 100 events.

Figure 22 shows a high-level view of the activities to distribute and gather event information using SMIP. Each node can generate events (notifications or Traps), but only a management station may also query for information and respond to queries. Note the input filter that nodes use to validate incoming notifications and requests. Unauthorized accesses result in rejections and possibly a new local event. SMIP uses SNMP for data transport, privacy, authentication, and access control functions. As seen in **Figure 23**, SMIP data, SM, and normal network management can use the same protocol stack.

4.3.4. Security Services

Data integrity, access control, confidentiality, and authentication are all essential capabilities for SM. Since many security mechanism options are available, a security services profile is useful to narrow the choices and reduce the likelihood of incompatibility between systems that need to interoperate. This profile will identify the default security services for the CSSA infrastructure. Some security services, such as confidentiality, can be achieved in more than one way (e.g., information hiding versus routing or access control). Where flexibility is needed for special cases, more than one option may be permitted, but one will always be listed as preferred. Additionally, wherever SNMPv3 defines access to a necessary service,

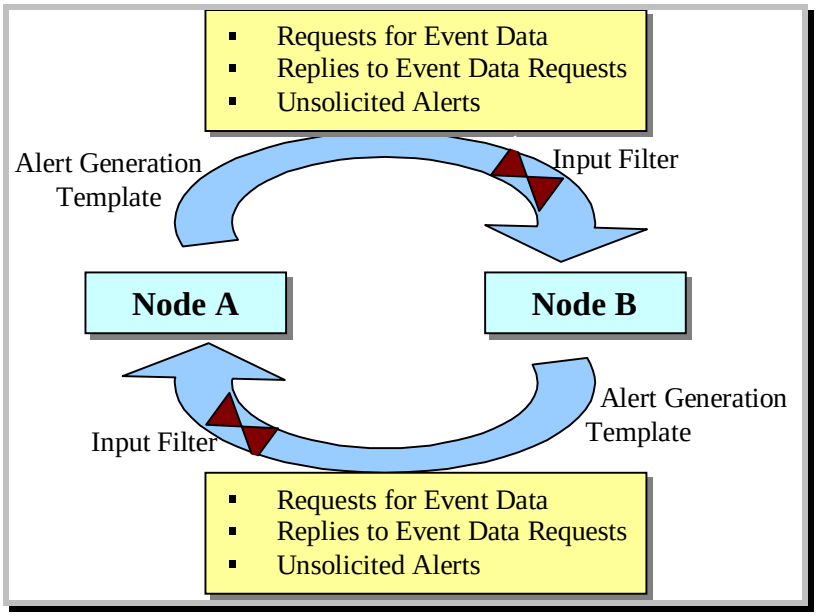

Figure 22. Peer Management Interactions

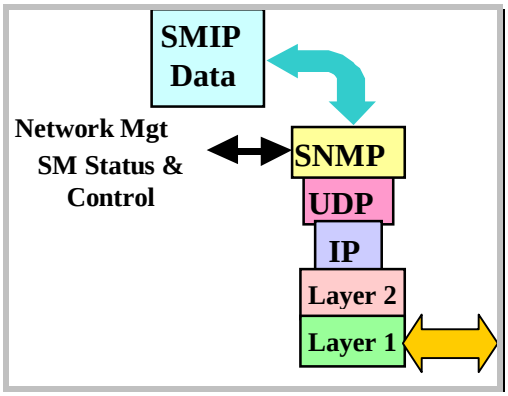

Figure 23. Shared Use of SNMP for Secure Transport

CSSA will follow the SNMP methods and modes. Below, after discussing each security function, we propose the CSSA profile of security services for SM interfaces.

4.3.4.1. Authentication

Authentication is the bedrock feature that provides trusted identity of actors in an automated system. Authentication provides verification of the identity of the user that generated a message and digital signatures confirm that a message originated from a particular user. Although Kerberos is the leading authentication system in use within many organizations, it has certain on-line support implications that could limit functionality in an embattled scenario. Specifically, Kerberos requires parties to interact with a ticket-granting server, which may be difficult when a network runs into trouble or is under attack. The SNMPv3 working group chose the MD5 authentication method with optional use of SHA for message authentication.

4.3.4.2. Access Control/Authorization

The Access Control subsystem within SNMPv3 provides authorization services by means of one or more Access Control Models. RFC 2275 [74] defines the View-based Access Control Model (VACM) for use among SNMPv3 management entities. Access control occurs (either implicitly or explicitly) in an SNMP entity when it processes an SNMP retrieval or modification request message or when a notification originator generates an SNMP notification message.

Since the types of actions allowed by security principals needs to be consistently managed by CSSA policies and standard sets of privileges can be anticipated for common operator activities, a role based access control scheme [66] is envisioned for CSSA administrative and operational functions. This concept also matches well with the VACM approach to define access privileges using defined operations on specific managed object sub-trees.

4.3.4.3. Confidentiality

The variety and incompatibility of currently available privacy mechanisms helps to establish the case that standards are imperative to establish interoperability. Although information hiding increasingly uses public key encryption mechanisms such as Pretty Good Privacy™ (PGP), SNMPv3, uses the Digital Encryption Standard in Cipher Block Chaining mode (DES-CBC). Public keys are very useful for dynamic transactions in which parties may exchange data only once or in an ad hoc manner. On the other hand, SM interactions are more likely to be among established peers and may persist between entities over long periods. Since SNMPv3 is integral to the SM tool kit, we will use the DES-CBC method.

4.3.4.4. Data Integrity

Data integrity provides verification that a received message is not corrupted, (i.e., the message received is the message sent). Since SNMPv3 uses MD5 with optional SHA to ensure protection from incidental or malicious modification of data, we will use the inherent SNMPv3 services.

4.3.4.5. Key Management

Key management is a significant issue considering that every security device or application with a secure SNMPv3 agent needs to share a compatible key with its management node. The use of symmetric DES-CBC keys by SNMPv3 implies the need for a key distribution infrastructure. If a separate MD5 authentication key is defined (as is recommended), then the task of key management doubles. In addition, many nodes may report to more than one control entity, so multiple DES and MD5 keys may be necessary. SNMPv3 defines basic key management within the User Security Model (USM) [6] and its related MIB. SNMP can accommodate security models other than USM, but operational experience will determine if alternatives are necessary. The IETF Security Working Group is working on several related Internet-Drafts

including the Internet Security Association and Key Management Protocol (ISAKMP) [39] and the Internet Key Exchange (IKE) [55]. A Diffie-Hellman key derivation or public key-based distribution strategy can make large, widely distributed networks easier to maintain. The number of security devices and applications, local security policy, and the ease of use of vendor products will influence whether manual or online key distribution and update methods are feasible and/or preferred.

4.3.4.6. Summary of Profile Recommendations

The following standard mechanisms, implemented via SNMPv3 or other APIs, will implement the required security services:

- **Authentication/Data integrity -** HMAC-MD5-96 (mandatory) and HMAC-SHA-96 (recommended)
- **Access Control/Authorization -** RFC 2275 (View-based Access Control Model) (SNMPv3)
- **Confidentiality -** CBC-DES (ANSI X3.92)
- **Key Management -** RFC 2274 (USM with key distribution and key updates using secure one way hash)

4.3.5. Data Transport - SNMP

Since the responsible SM, supervisory functions are remote from the security mechanisms or applications to be observed and controlled, a secure transport of management data is essential. Vital to an adaptive SM concept is the ability to configure remote devices reliably and securely. In addition, the trusted propagation of SMIP data between managers also requires security and trust.

Several candidate protocols for secure transport of management data exist, including the Secure Socket Layer [33] (used by Secure HTTP), Transport Layer Security (TLS) [26], CMIP/Generic Upper Layer Security (GULS) [48], SNMPv3, the Open Group's Distributed Computing Environment (DCE) security [7], IP security, and others. While we will not examine the technical differences, the widespread use of SNMP within the network management community argues for SNMPv3 as the common baseline.

SNMPv3 uses DES encrypted PDUs over UDP/IP. Although this is not a guaranteed delivery mechanism, it is a reasonable compromise for management applications compared to maintaining TCP connections. The lack of a secure Set operation is considered one of the most serious deficiencies of

SNMPv1. This limitation is evident in one group's informal effort to develop a Firewall MIB [38]. The group assumes use of non-secure management tools and side-steps the security issue by defining read-only status with no configurable parameters. Without configuration capabilities, there can be no adaptive controls and therefore, monitoring is the best that can be accomplished. The recent approval of SNMPv3 RFCs as proposed Internet standards with advancement to draft status expected in Spring 1999 should bring SNMPv3 capable agents and managers later in 1999. This will provide a widely accepted IETF standard for transport of management transactions with adequate security for privacy, authentication, data integrity, and access control.

4.3.6. Policy and Configuration Management - SMIB/SMON

From the discussions in Chapter Two and Three, it is clear that improved policy conformance may be achieved by better sharing of data between the administration, operations, and assessment phases. Frincke et al reached a similar conclusion in [34]. To achieve better data sharing of performance and configuration information, we take a two-pronged approach. First, in the SMIB are security policy and administration policy rule tables. In addition, we designed the Security Monitoring MIB as an augmentation of the basic SMIB. The SMON MIB keeps statistics on security events (e.g., port scans and repeated bad log-ins) by host, by a TopN table, and by a Suspects (site or subject) table. Section 5.2 develops the design of the SMON MIB further. Its goal is to enable the value-added data interactions needed between the SM phases to evaluate event information against security policies. Although we provide data structures to implement policy and configuration management, there is significant need for further work on the policy validation and update evaluation processes.

4.3.7. Data Storage/Data Repository

Methods to maintain security data in readily available form are important to SM system performance and functionality. Access control to ensure authorization to view or modify management information is vital for system integrity. SNMPv3 defines the View-based Access Control Model (VACM) in RFC 2275 [74]. SNMP management stations typically use an RDBMS backend component to maintain

large amounts of MIB status and event information. Use of the Lightweight Directory Access Protocol (LDAP) is another option for storing data that may be referenced by several management nodes. With the move to SNMPv3, there needs to be attention given on maintaining secure database operations. Since specific methods for secure data storage and access control strategies are major ongoing research topics in themselves, they are not addressed further in this research.

4.3.8. Management Platform

Another essential component of a fully functional SM system is a secure management platform, including the hardware platform, operating system, and data handling mechanisms. As with data storage operations, the platform is part of the whole secure environment. The function that primarily supports SM is the SNMP management engine. As defined by SNMPv3 standards [41], the management engine includes a dispatcher, a message processing subsystem, a security subsystem, and an access control subsystem. An interface between the management engine and the event correlator is not defined by SNMP, but was discussed Section 4.3.2. In addition to management station concerns, as security applications and their associated agents become more complex, they are more likely to run on systems with general purpose operating systems (OS) as compared to a limited router/hub OS). This leads to concern about trusted OS functionality on the agent side. Obviously, if the underlying OS is weak, the entire application is vulnerable.

4.3.9. Intelligent Security Agents

Another essential element to enable remote management of security applications is the remote agent. The agent is the complement to the SNMP management engine. It verifies authority of access and commands, implements directives from the manager, and reports local information and status using SNMP. An intelligent agent's functionality goes well beyond implementation of basic SNMP protocols. It must take action based on received commands and initiate notifications when defined conditions occur or thresholds are crossed. To do this, agents must be able to access status values from their associated applications. Since agents are responsible for accepting management commands and releasing status

information, they must be included within the Trusted Computing Base (TCB) of a secure application. SNMPv3 agent implementations should become generally available in 1999, but the harder issue is the large integration effort needed to meld the required SM data elements into security applications. The application integration is what will bring the real benefits. That is also why the definition of standard SM data elements needs to be done early, so they can be deployed broadly in a consist way.

Chapter Five—

SMIB/SMON Detailed Design

If SM is to provide a high degree of security assurance across the diverse security applications developed by multiple vendors, the security MIBs must provide a logical and useful organization of security data. In this chapter, we offer more supporting details for the MIB designs presented earlier. We review selected parameters of the SMIB, SMON MIB, Firewall MIB, and Security Guard MIB to give a stronger sense of the critical importance, development rationale and justification of important elements of the MIB designs. Just as the operational security environment results from the combined effect of all security policy layers (see **Figure 24**), so the usefulness of CSSA results from the sum of its parts.

The basic MIB development concept, as laid out in the Structure of Management Information, RFC 1155 [60] and RFC 1902 [15], is to define parameters in a tree hierarchy so that branches may be created independently and extended as appropriate. Although SNMPv3 provides the critical components needed for management entities to communicate and conduct secure transactions with secure agents, ultimately the MIB definitions ensure that the right data are defined to permit active management and analysis of operational security applications. A common core of definitions contained in the SMIB serves as a baseline for product interoperability between management applications and security applications from multiple vendors. Compliance statements give conformance information for "lower bound" MIB implementations. MIB extensions augment basic functionality with new or unique features of vendor implementations. As a contextual reference, one can view the SMIB as containing relatively static configuration and performance information about local devices. The SMIB is broadly implemented in SM

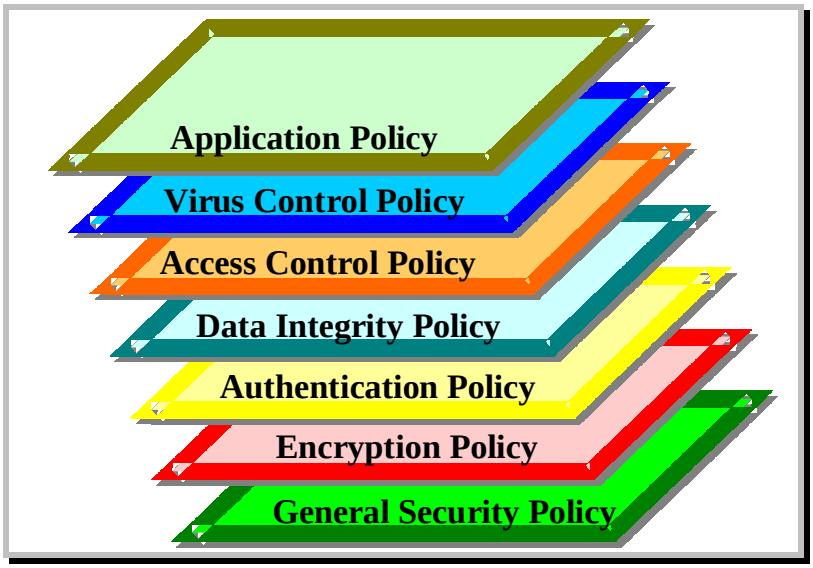

Figure 24. Security Policy Layers

agents across the security domain to track common values. The SMON is an extension to the SMIB. It is implemented more selectively and holds data that is more dynamic, distributed, and oriented on summarizing event statistics.

5.1. SMIB

The initial SMIB discussion in Chapter Four began the MIB definition process at a high level. Now we further discuss how SMIB information may be used. The basic SMIB specified here consists of a few core modules (groups) of parameters that are always required for compliance. Feature extensions for specific security operations permit product differentiation. These plug-in components permit the addition of special purpose definitions required to configure or monitor the unique features of particular security mechanisms. The basic SMIB (see **Appendix A**) consists of the System, Security Policy, Administration Policy, System Log, and SMIB Trap groups. SMON and security application-specific MIB parameters (see **Figure 25**) are defined in separate MIBs that are discussed later in this chapter.

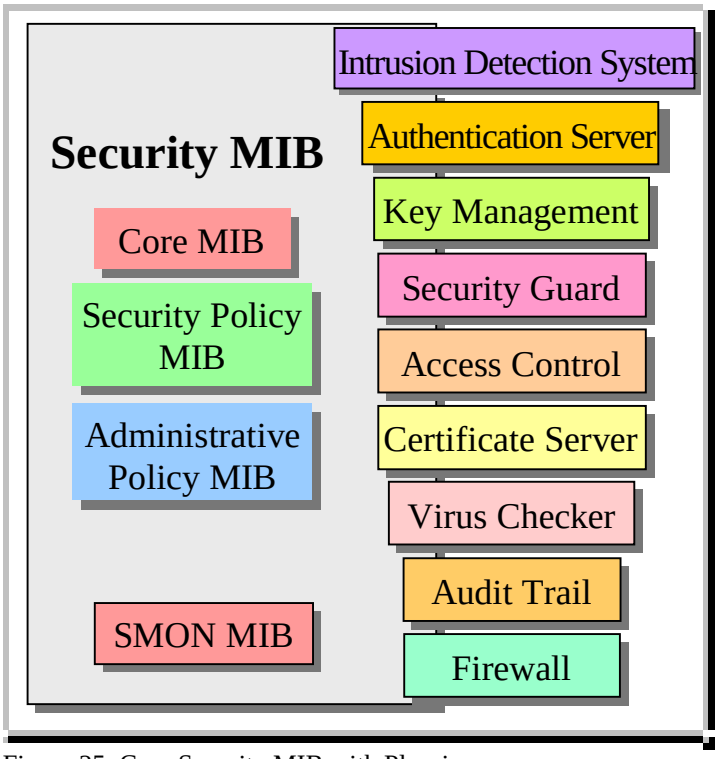

Figure 25. Core Security MIB with Plug-ins

5.1.1. System Group

The System group of the SMIB contains the configuration information that indicates the identification, capabilities, and common settings of the local device. The sysSecName and sysSecDomain are obvious identifiers by which to track each SM node. Likewise, identification of an administrative and security POC for the system is necessary to report anomalies and correct problems. The read-only public key allows an external node to validate the source of notification information that reputedly originated there. Even though notifications are typically authenticated when they are initiated and when propagated as SMIP notifications, it as possible for authentication to be turned off. A lack of traceability through multiple hops may weaken confidence in the validity of the source data. The public key parameter allows the dependent node to directly validate the digital signature in the notification with the reputed source.

The administrative state parameter is a general health indicator, which is intended to be generally accessible by any requester. The sysCommand is a R/W parameter that provides macro control functions as we discussed in Chapter Four. The last successful command is retained here until replaced by a new

command. The next two values are read-only and provide last security event information (type and time) giving a quick look at the last event from any local security applications. The sysTimeSincePreviousEvent gives an idea of the interval between the last event and any previous events.

Next, the current and objective Information Protection Levels (IPL) provide a novel management context for threat and vulnerability. The IPL influences the way that management information is assessed and how applications are controlled. The smibCurrentIPL parameter is the local device IPL status, while the smibObjectiveIPL is the system-wide IPL setting. Differences in the two indicate that either negative events have degraded the local (current) IPL or the objective IPL has been raised and the current IPL has not yet "caught up". The SMIB keepalive parameters help ensure that potential data loss due to an unreliable network is identified. If the control entity does not receive the keepalive packets, it indicates an event. Although Trap messages from a threshold violation may by lost in the network, at least it will be recorded as a local event that can be polled by the control entity.

The next two sets of system level parameters (sysApplTable and SysConfedTable) show information on the security applications and management entities that have been registered at the SM node for management visibility. If a security application or peer manager is not listed here, the application or manager is unknown and untrusted by hosts in this security domain.

The application table identifies the type and version of security applications on a managed node, its security requirements, and initialization parameters. Key management parameters and keys may also be added on a case by case basis if desired. The application security requirements identify the SM needs when interacting with the associated application to protect the system and its users from security breaches.

The confederated host table implements the concept of groups of management hosts that accept information at various trust levels. Hosts in the same group can trust management traffic from others in the group as if it was generated locally. Each external host that is to be treated as a peer will have an entry. The sysConfedType and sysConfedNotificationEable parameters help hosts to filter which messages they will accept. A management host will likely be in more than one group; therefore, more than one entry in the sysConfedTable may have the same hostname and address.

5.1.2. Security Policy (SP) Group

We do not address how to develop security policy specifications in this work; however, to the extent that a security policy can be precisely defined, it is a critical part of the configuration information captured in the SMIB. The SM system uses the security policy to control and enhance the trust in the overall security of the system.

The SP group defines a set of managed objects that uniquely specify the permitted use of system services and resources. The spLastUpdate and spUpdatePOC record the most recent change history. The spRulesTable captures specific rules derived from the security policy. The rules have an implicit "deny all", so unless access is specified, no access to resources is allowed. Each SP rule is identified by an index and text name. The expiration field is required, but may contain all zeros to indicate that no expiration is specified. The target of each rule indicates what resource is being protected. For example, an access control rule may allow/deny a subject or group of subjects access to a host, a directory, database record, or an instance of a MIB parameter. For now, a textual reference is used, but any standard method for referencing computer resources may take precedence. The spRuleSubject field identifies the individual, host, or named group affected by the rule. All subjects must be registered at a locally trusted authentication server (i.e., a peer node in the confederated host table with sysConfedType that includes Accept Authentication Requests). The default value "All" has special meaning in the subject fields, along with the default deny action to apply the implicit rule — "Deny all resources to all subjects". The action and permission fields are inter-related in that the permissions are only valid if the action field is "permit (2)". Permissions are also valid only when the spRuleType is set to "access control (2)". The last spRule field, spRuleParameters, provides any additional parameters needed to implement the rule. **Table 9** portrays an example instance of the spRuleTable. The columns of the spRuleTable are presented here as rows for the sake of readability. Each row of spRuleTable forms a new and distinct rule instance. It is an issue for the security policy administration manager to resolve conflicts and unintended consequences of cascading permissions (i.e., A can access B's resources and B can access C's resources, but A should not have access to C).

Table 9. Example of Security Policy Rules

Parameter Name	Format	Value
SpRuleIndex	INTEGER	1
SpRuleName	DisplayString	DenyBinAccess
SpRuleType	INTEGER	2
SpRuleExpiration	UTCTime	0000000000Z
SpRuleTarget	DisplayString	MainServer.ACME.com:/bin
SpRuleSubject	DisplayString	LocalSysAdmin
SpRuleAction	INTEGER	2
SpRulePermissions	INTEGER	777
SpRuleParameters	DisplayString	N/A

5.1.3. Administration Policy (AP) Group

The AP group contains information related to administrative access and updates to the SMIB (how and by whom). Thus, the AP group contains the second order configuration information that controls the administration and configuration management of SM functions that are in control of security applications. Therefore, AP group information is crucial to the overall protection of system security. The first AP group parameter identifies the senior security officer who may delegate security authority to subordinate security managers. The second parameter defines what officers must approve changes to the top level SMIB.

The administration manager table identifies all SM manager subjects and the permissions they have been granted. Although this may be redundant with general user ACL or RBAC capabilities, the sensitivity of security administration functions argues for special treatment of the administration privileges. The target resources in the adminMgrTable identifies only SM resources that may be accessed and/or modified, since regular access control is defined by user tables in a host access MIB or other relevant security application MIB. An administration role-based access control scheme could be implemented in the AP group; however, that is left as future work.

5.1.4. Security Log Group

This group implements the security logging function for remote security applications. Some applications may implement their local logging or have hooks to transfer logs to a central syslog facility. While not prevented, a common repository of security-relevant events gathered via trusted transactions is important to extend the security manager's view of the world. There is some question whether to place the

log table in the SMIB or the SMON. Since logging is an essential function of all security applications, we consider it appropriate to place the security log group in the core SMIB.

The security log group data definitions identify the log administrator, the maximum log table size, percent of log filled, log start date/time, and logging levels. Other parameters define where the log is to be archived and the interval for sending updates to the archive. The log table records the event time, reporting host, the targeted host, resource, the attacker's host address, and identification, if known. An event type, severity, duration, and security level help the event management to determine the need for further aggregation and distribution.

5.1.5. SMIB Trap Group

A single SMIB trap provides a method for all security applications to report local events to a monitoring host or group. Numerous event types and severity levels provide a great amount of flexibility in determining how to respond to each notification. The destination hosts are identified by the sysConfedNotificationEnable flag in the confederated hosts table. Note that a notification may be sent to a host that has since decided not to accept notifications (possibly due to change in IPL). If some messages are discarded, it is acceptable because notifications are considered "best effort" alerts anyhow.

5.2. SMON MIB

The purpose of the SMON MIB is to efficiently summarize the security conditions within a security domain. The SMON MIB specification (see **Appendix B**) identifies dynamic security states, tracks local and remote security events, and retains relevant status history that support monitoring and reaction to security events that may affect security posture. Because the SMON is oriented toward providing the "big picture" for network security status, specific data elements from a network probe may be aggregated and concentrated for a whole subnet.

Although the original idea for the SMON MIB came from experience with the Remote Monitoring (RMON) MIB, the two have no direct dependencies. The SMON MIB contains similar groups for tabulating events related to individual hosts as well as network security applications. The detected events

on hosts are likely to be quite different from those detected by network security applications. Hosts may be the target of an attack to access resources on the host. On the other hand, security applications typically would not be targeted (and may in fact be invisible) for their resources, but to gain data that could provide leverage into other systems.

The first group within SMON is the Hosts group. It tracks typical host level events that could yield information in tracing an attack. As with remote network monitoring, the individual events may not necessarily mean much, but anomalies or trends over time can support security analysis. For example, log-in attempts, TCP SYN events, and port scans are captured as dynamic parameters as they occur. They are also captured in the HostHistory group to show trends over particular periods. The number and length of intervals, data source and other control information about the history collection is defined in the smonHistoryControlTable.

The next SMON group is the Alarms group. It provides a simple way to trigger events based on rising or falling values or deltas. Various thresholds for other integer-based MIB variables can be configured to help detect and report suspicious activity to a consolidated management site for correlation and assessment. The Security Application Top N group contains a control table and a data table that provide a way to generate reports on changes monitored on the network.

The SMON MIB also implements the Suspects group. This is a table of reported suspicious sources that may be useful in tracking new events against previous activity. After the suspects table ages or grows to a certain point, it is typically archived to a senior management node for future use. By setting the update interval, the table can be pared or flushed as needed.

The last two groups in the SMON MIB are used to implement the SMIP capability. The Notifications group contains the data from SMIP notifications and the Event Data Request group keeps data related to SMIP queries. Each query is recorded in the smonQueryTable until it is satisfied. The idea of standing or persistent queries can be implemented by setting the smonQueryUrgency value to zero. This retains the query indefinitely (or until the smonQueryMaxResponses threshold), so any matching event in the future will be returned to the requester.

The SMON MIB, like other MIBs, permits operational use with a reduced view of the whole management domain. RFC 2275 [74] defines View-based Access Control for SNMPv3; however, the use of subviews can also help limit operator complexity as well as implement division of duties and least privilege controls. As illustrated in **Figure 26**, a network operator may see only network management parameters and only the instances in their assigned network. The administration policy in the SMIB defines the allowed access and whether access to the parameters requires authentication and encryption. Security managers may have a view of all parameter types, but still of only their portion of the system. Higher-ranking network managers and security officers may have correspondingly wider views. In fact, for ease of understanding any operator/manager may have a set of views that are various subsets of their maximum authorized view.

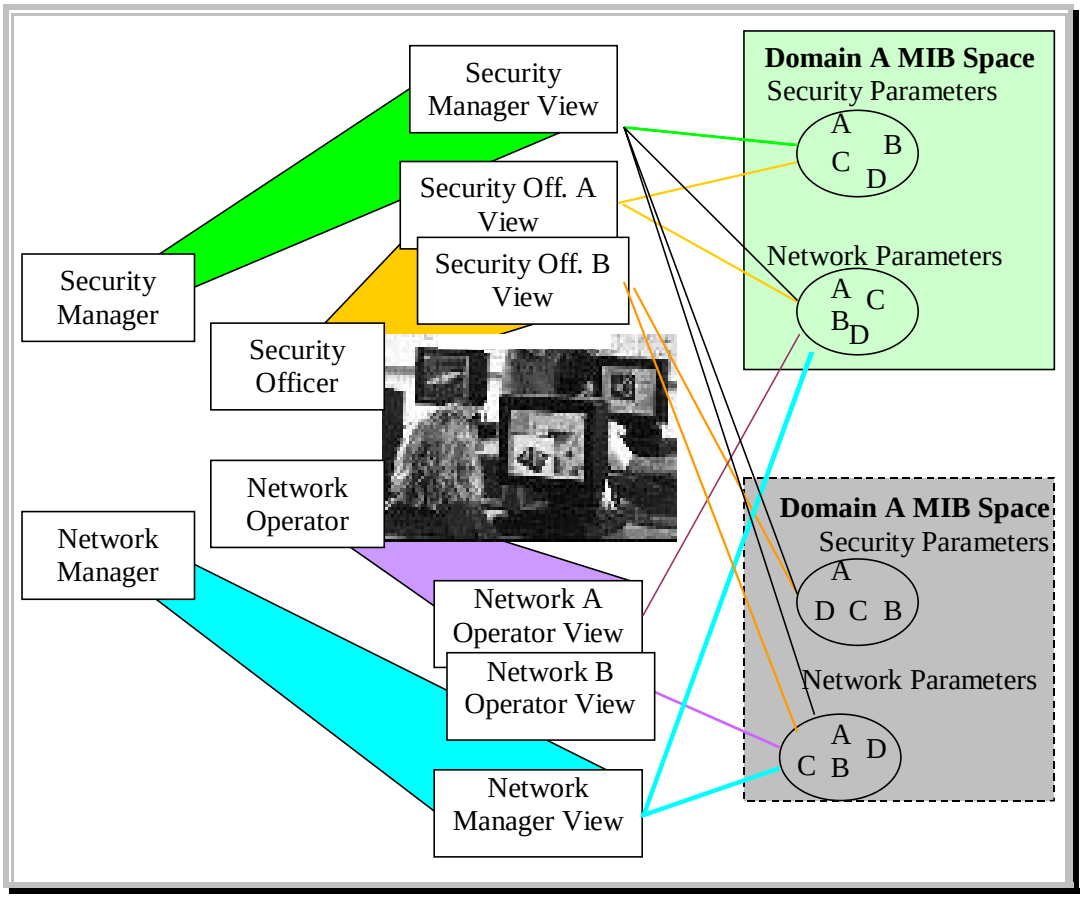

Figure 26. Use of Subviews for Operations

5.3. Application-Specific MIBs

5.3.1. Firewall MIB (FWMIB)

The core SM parameters defined in the SMIB are useful for status and activities that are common to all security devices; however, special features important to firewall operations may require a specific extension to the SMIB. Additionally, since not all firewalls are functionally equivalent, provision is made in the FWMIB specification (see **Appendix C**) for further extensibility and differentiation. In fact, areas that are likely for extension in the future include management parameters for content-based filters and stateful inspections that are becoming common in firewall products.

An outline of a generic Firewall MIB is shown at **Figure 27**. Although no IETF working groups are chartered to develop a firewall MIB, one informal group of vendors has worked on an Internet Draft [38] for a firewall MIB for monitoring only. Another research team [49] has also considered use of SNMP to manage firewalls. Neither effort assumes management security; therefore, they do not allow any Set (configuration) operations. Our proposed FWMIB assumes use of SNMPv3 security, and so does not restrict Set operations. Using our FWMIB, a remote management station can safely adjust[6] performance,

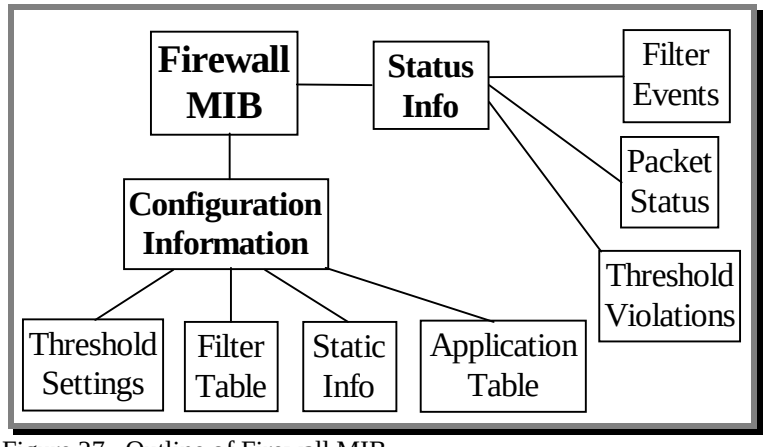

Figure 27. Outline of Firewall MIB

[6] The notion of adaptable performance and security levels is quite new. At least one vendor has announced plans to permit remote configuration of their firewall as either a packet filter or proxy server, based on security/performance tradeoffs [59]. CSSA, however, goes much further in suggesting a true adaptive feedback approach with configurable protection levels.

proxy, and filtering parameters. Our FWMIB consists of four parameter groups including System, PacketFilter, Proxy, and Traps.

5.3.1.1. Firewall System Group

The fwSystem group holds firewall-unique system parameters that are not in the SMIB. This includes a global forwarding and non-IP forwarding switch to give macro level control of packet processing. More specific controls are in the groups that follow.

5.3.1.2. Packet Filter Group

The most basic firewall capability is to pass or reject packets based on TCP/IP header information. This group permits creation, modification, and deletion of filtering rules by an authorized management station. Additional activation factors like active days and times permit flexible configuration of when the rules are in effect. Packet filtering is typically implemented on routers and basic firewalls. Some access control applications such as tcp_wrapper also implement a limited form of host-level packet filtering. In that sense, it is conceivable that the packet filter parameters below could share common definitions with such host-based applications. The implementation would require only modification of tcp_wrapper to read its filter configuration from a MIB file that is created by an agent module. Although such work appears useful and not too difficult, it is reserved for future work. It is apparent that a common method to update filter tables on firewalls, screening routers, and host access control applications using SNMP can greatly assist the security manager/network operator.

5.3.1.3. Proxy Sessions Group

The trend in firewall proxy support is toward more, rather than fewer, unique proxy services. The number and capabilities of proxy services supported by firewall developers varies widely; however, some vendors reportedly support over 300 unique proxy services [57]. The proxyProtocol object identifies the type of proxy service. The proxySessionTable keeps track of all ongoing proxy activity and defines the type of monitoring being done. Default monitoring levels are based on the SMIB sysObjectiveIPL parameter; however, the proxyMonitoring variable is writeable so the management station can modify its setting. Care is necessary to prevent excessive logging.

5.3.1.4. Firewall Trap

A unique Trap message for firewall-specific notifications has been defined in the Trap group. However, modifications to the SMIB Trap definitions may accommodate the needs and permit a single definition for all security applications.

5.3.2. Security Guard MIB (SGMIB)

The SG scenario in Section 3.3 provided background on a conceptual deployment of a semi-automated information security guard system. Secure remote management using standards-based management tools requires well-defined data definitions. As an operational security application, the SG implements the core SMIB for general management parameters, configuration variables, and logging functions, etc. The SGMIB specification (see **Appendix D**) defines unique management parameters that apply specifically to security guards. It demonstrates a robust, cost-effective alternative to dedicated proprietary management systems for high assurance applications. As was suggested in **Table 4** (Section 3.3), the SGMIB should define both administrative and operational parameters. Unique assessment information related to SG events is forwarded and processed by the control entity. The SGMIB has three management groups, sgAdmin, sgOpsCon, and sgTrap.

5.3.2.1. SG Administration Group

The SG administration group identifies a table of authorized information releasers. Each releaser has authentication, active day/time, and data type parameters that must match the content type and digital signature of a candidate release package. In addition, the SG administration group has a filter table that identifies data fields, values, and modifiers to indicate data content that cannot be released. This table is similar to the content filter applications being added to many firewalls, so some parameters may be applicable in that context as well. In fact, a merger of capabilities of firewalls and security guards appears possible. Some say the only difference is the level of assurance expected of each (i.e., security guards have higher standards of assurance since they connect two different classifications of information).

5.3.2.2. SG Operational Control Group

The Operational Control group permits remote management of the SG application through a master enable field and two status fields that show the percentage of input data storage space and quarantine data storage space that is in use. Another value defines the interval between keepalive packets emitted by the SG to assure communications to the manager is functional. Other operational management objects may be defined as needed.

5.3.2.3. SG Trap Group

The SG Trap group defines two types of housekeeping notifications, one for the quarantine space becoming 80% full and the other for input space becoming 80% full. An acknowledged Keepalive notification using the SNMP InformRequest is defined to support the communications handshake exchange required to assure contact with the manager is maintained.

Chapter Six—

SMIP Detailed Design

The basic sequence of actions that comprise the SMIP notification and event data request is given below. In addition, we present a detailed example of a SMIP notification and a SMIP data query to show how the design is intended to be applied.

6.1.1. SMIP Event Notification

The purpose of SMIP notifications is to distribute current security status between management nodes. When a management node receives an SNMP notification from an agent, the event manager determines which events to propagate based on the Information Protection Level (IPL) and the event characteristics. A local broadcast can be used. Alternatively, an agent may multicast a notification using a table of associated destinations instead of broadcast zones. We refer to this group of peer control entities as *confederated hosts*.

As the IPL increases due to a greater perceived or detected threat, a broader announcement of security events may be sent to a wider band of confederated hosts so that more nodes are aware of the increasing suspicious activity. This is an important reason for using the term "concentric" in the CSSA concept. For example, a management system at IPL1 may notify a default superior node in a management hierarchy only about the most serious events. At IPL3, the system may notify both its immediate neighbors and the superior management node of serious warnings and significant suspicious events. At IPL5, because maximum awareness is desired, the same station may transmit additional, less serious events and the range

of dissemination may increase to all associated sites or partners.

6.1.1.1. SMIP Notification Data Format

SMIP notifications use the acknowledged SNMP InformRequest message format. **Box 4** shows a proposed data structure conveyed by the SMIP notifications and stored in the SMON MIB. All notifications should use authentication and data integrity to assure receivers of the source of the data and its validity. Privacy may be applied selectively according to the local security policies. Care is important in assessing the level of trust to put in non-verifiable, secondary information such as IP addresses and DNS names, since they may be spoofed or subverted. Experimentation and/or simulation efforts may indicate useful limits for maximizing the value of shared information (i.e., more data can cause information overload rather than the intended goal of better situational comprehension).

6.1.1.2. SMIP Notification Example

The SMIP notification is initiated when the event correlator passes the Event Notification parameters to the event manager. The event manager initiates an SNMP message using an InformRequest PDU (see **Table 10**) in the form of:

> **InformRequest** (Date of Event, Reporting Host, Originator Authentication, Type of Event, Severity of Event, Target Host, Target Resource ID, Suspected Attacker Host, Suspected Attacker ID)

Box 4. SMIP Notification Parameters

Date of Event: (Date/Time)
Reporting Host: (IP Address)
Originator Authentication: Digital Signature
Level of Trust: (High, Medium, Low, None)
Type of Event: (Probe, Failed Logins, Email virus, etc.)
Severity of Event: (Informational, Warning, Urgent, Emergency)
Target Host: (IP Address or DNS name)
Target Resouce ID: (USERID or resource name)
Suspected Attacker Host: (IP Address or DNS name)
Suspected Attacker ID: (IP Address, USERID, or Codename)

Table 10. Notifications using the InformRequest-PDU

Field Name	Type	Value	Note
Request-id	Integer32	-	-
Error-status	INTEGER	noError (0), tooBig (1), noSuchName (2), badValue (3), readOnly (4), genErr (5), noAccess (6), wrongType (7), wrongLength (8), wrongEncoding (9), wrongValue (10), noCreation (11), inconsistentValue (12), resourceUnavailable (13), commitFailed (14), undoFailed (15), authorizationError (16), notWritable (17), inconsistentName (18)	-
Error-index	INTEGER	0..max-bindings (0..2147483647)	The varbind that caused the error.
SysUpTime	TimeTicks		1st Varbind
SnmpTrapOID	OBJECT IDENTIFIER	Value, unSpecified, noSuchObject [0], noSuchInstance [1], endOfMibView [2]	2nd Varbind
Additional Varbindings	Varbind	0..maxbindings-2 (0.. 2147483645)	-

The InformRequest PDU requires a Response PDU back from the receiver to confirm the data was accepted. Upon receipt of a new message, the receiver must determine whether to add the information to the local event database. The confederated hosts table in the SMIB identifies associated hosts that have established a level of trust. Notices from untrusted peers may be discarded or kept with a flag indicating it is not from verified partner. Notifications, both local and those received via SMIP, are stored in the SMON MIB, Notifications Group, which is implemented only on management stations.

6.1.2. SMIP Event Data Request

From time to time, the event correlator of a local management entity may need to request external support from an approved group of trusted peers. As the severity and IPL associated with a current event assessment increase, the confirmation of similar occurrences becomes more desirable, especially if the correlator is not able to identify any related local events.

6.1.2.1. SMIP Event Data Request Format

The data request between managers will use the GetBulkRequest PDU with a corresponding Response PDU in the manner defined in sections 4.2.1 through 4.2.4 of RFC 1905 [20]. The basic sequence of protocol actions between management nodes is shown in **Box 5** below. The event correlator typically initiates the request for an expanded search of external SMIB data when local information is

Box 5. SMIP Event Data Request Sequence

Event Data Request: Event Correlator → Event Mgr

SNMP GetBulkRequest: Event Mgr → MPI[1] → Peer Management Station

Remote Management Station Response: Peer Management Station → MPI → Event Mgr

External SMIP Data: Event Mgr → Correlator

Assessment Result: Correlator → Event Mgr

Log Record: Event Mgr → Log

[1] Multi-Protocol Interface (see Figure 21)

insufficient to establish a suspicion or threat level for an event. Based on the IPL, the external request may be to only the immediate workgroup, to a whole DNS domain, or to a defined extranet. Due to security concerns at both sender and receiver, the most likely exchanges are among a pre-arranged group of confederated hosts as noted above.

6.1.2.2. SMIP Event Data Request Example

The SMIP correlation request is a special feature of CSSA that permits a group of hosts to share security event information. However, instead of centralizing all of the data, distributed managers maintain their local event data, and share externally with associates as needed. SM nodes must be able to initiate and respond to SMIP requests. A SMIP request uses the GetBulkRequest PDU (see **Table 11**) under the assumption that the response may consist of zero or more matching events. The GetBulkRequest PDU is especially efficient for cases where many "hits" may occur because it limits the need for many "query-single response" cycles. With a properly formed query, several relevant events may be returned in a single Response PDU. For some cases, several queries may be necessary if there are several relevant fields. In that case, the event correlator has to deal with combining and assessing the several responses.

Table 11. SMIP Event Data Request using the GetBulkRequest PDU

Field Name	Type	Value	Note
Request-id	Integer32		
Non-repeaters	INTEGER	0..max-bindings (0..2147483647)	N
Max-repetitions	INTEGER	0..maxbindings (0.. 2147483647)	M
SysUpTime	TimeTicks		1st Varbind
Additional Varbindings	Varbind	Value, unSpecified, noSuchObject [0], noSuchInstance [1], endOfMibView [2]	R = # of Varbindings-N

6.1.3. SMIP Request Processing

At destination management station, the GetBulkRequest command initiates a search action using selectors provided in the request. Typical selection parameters may include date, IP address, event type, and event severity ranges. Compliant implementations of SMIP must provide at least a baseline query-response capability. **Box 6** below lists some recommended SMIP parameters that should be searchable. The search capability should permit searches based on at least two fields at a time and permit modifiers such as "like", "=", and "/=" (not equal) for appropriate fields (i.e., smonQueryTargetID and smonQuerySourceID). The search parameters are represented in the SMON MIB as columns in the smonQueryTable of the SMIP Event Data Request module in Appendix B, which was discussed above.

The SMIP request from one management node to another to search for a relevant security event in its SMIB is an action that may require a significant processing effort. Our assumption is that since both ends of the transaction are management stations, there will be sufficient computing resources at the destination to sort and filter the response data prior to shipping it over the network to the requester. This is a network-efficient approach, but it requires more capability at the responding node to search and filter responses. The alternative would be to do more processing at the originating management node. That is more appropriate when sending SNMP data polls from managers to a large population of resource-limited agents.

Box 6. SMIP Event Data Request Parameters

Confederated Host List:	List of IpAddress
Date Range:	Start/Stop Dates
Target ID:	IP Address, DNS domain, or Subnet and mask
Source ID:	IP Address, USERID, or Codename
Severity of Event:	(All, Warnings, Urgent, Emergency)
Type of Event:	(Probe, Failed Logins, Email, etc.)
Urgency:	Integer (0..10)
Max Responses:	Integer (1..N)

Chapter Seven—

Conclusions

Our hypothesis for this research has been that remote monitoring and control of security applications can be designed to be just as viable for Security Management as it has become for network management. The CSSA concept has offered a three-phase SM process model and a security MIB data model to show that security applications do have areas of commonality for management activities. Specific proposals for MIB designs and the SMIP with IPLs provide steps and lessons that contribute toward a concept for viable implementations. No single person or organization can dictate the entire SM design; however, we have proposed a foundation that can be modified and extended as needed. Short of deployment of a prototype, this work provides a substantial case that further deployment efforts can lead toward benefits for SM as occurred in network management environments. However, even with a secure communications protocol (SNMPv3) and a SM framework, management platform vendors and security application developers must expend a significant effort to define additional SM parameters and integrate them with their operational systems.

In this final chapter, we evaluate the significance of the CSSA framework, examine migration needs for the implementation of SM, and discuss future research needs to make active security assurance capabilities more effective and pervasive.

7.1. Assessment of Contributions and Component Designs

Of the necessary elements for a complete SM architecture, we have made specific suggestions

regarding the Security MIB, SMON, and the SMIP as part of our CSSA concept. In this section, we consider the value, metrics of success, and alternative issues of these contributions.

7.1.1. CSSA

The overall CSSA concept of a three-phase management cycle with MIB data providing linkage is considered a means to the end goal of extending capabilities needed for a workable SM environment. While the CSSA provides an important organizational framework for SM, the components are the tangible contributions toward SM operations.

7.1.2. SMIB

The primary benefit intended in defining the SMIB is to allow any combination of security applications to be managed by generic tools from a common platform. The idea to define an SMIB to store security-related data is not unique; however, the idea to create a core MIB to which application-specific security MIBs can be added has not been articulated before. The key contribution is the development of a draft SMIB specification that defines common management parameters.

7.1.2.1. Metrics for Success

Not all security applications will make a complete transition to standardized remote management methods (security officers are a hard crowd to convince). Some small or specialized systems may not need generic tools. The main measure of success of SM is the number of vendors that develop products that can be monitored and controlled via SNMPv3. The success of SMIB must be measured against the equivalent SNMPv1 baseline MIB (MIB-II) found in nearly all manageable network devices. Most security applications that are SNMPv3 manageable should use the SMIB to permit visibility of basic common information to any SNMPv3 management node that has authorization. The other sign of success is that new security applications will make their functions manageable via published MIBs and vendors will only define MIB parameters for unique features.

7.1.2.2. Alternatives

The alternative to a commonly defined set of management parameters is redundant, proprietary definitions that do not interoperate or are not fully compatible. For the benefit of the whole security services community, standard management methods enable consolidated views of all security applications. Further integration using the same SNMPv3 infrastructure that supports a large installed base of network management application should greatly ameliorate the high cost of deployment.

7.1.3. SMON

The notion of adapting the RMON concept to the security management context for keeping statistics on network security events is an original concept. It holds a good deal of promise to permit use of common statistics on important event metrics among agents and managers. The SMON MIB has the potential for broad applicability if published as an Internet draft and RFC thereafter.

7.1.3.1. Metrics for Success

As currently defined, the SMON MIB includes only part of the data that may help security managers collaborate against common threats. If SMON is widely implemented, additional extensions will be needed that are tailored to specific SM functions. Initially, publishing the SMON MIB as an Internet draft is important, but support of its security statistic reporting by network management platform vendors and security application developers will be the real measure of success. The first challenge is to implement SM agents to collect and report the security events. The second task is to make available SM applications to access the SMON information and respond appropriately.

7.1.3.2. Alternatives

Just as network management was able to function without RMON for many years, SM can do without SMON. However, just as RMON and remote network probes were a boon to network operators, SMON can enhance the detection and consolidation of security event data to enable security managers/operators to see the "big picture".

7.1.4. SMIP

SMIP is an important companion to the SMON and SMIB to document procedures for distribution of event data between confederated hosts that have agreed to share security information. The exchange of management information amongst management modules and throughout the lifecycle of SM is essential to effective and efficient operations. Implementation of the SMIP or a variant that permits loosely associated security entities to share threat and attack information is one way that the "good guys" can gain the upper hand in the fight against computer abuse.

7.1.4.1. Metrics for Success

The SMIP, as currently defined, is only the start of a wide variety of functions that may help security managers collaborate against common threats. Expanded use and more advanced methods to exchange of additional data types among associated security entities will signal that the initial SMIP concept was the seed of an important SM capability.

7.1.4.2. Alternatives

The ability to propagate security information and access remote event histories extends the vision of security operators. Without SMIP or an equivalent secure tip-off capability, security assessments must be based on limited local data, typically from a few single purpose applications. Although dedicated capabilities can be very useful, the cost-benefit ratio can also be prohibitive.

7.2. Transition Strategies

In this section, we discuss some of the expected impacts of a move toward a secure management environment. The purpose is to establish a point of departure from which to extend the work presented here and to plan for migration. Bluntly put, migration to SNMPv3 and fully functional SM capabilities is not likely to be cheap. The key is to determine the value of the assets being protected and then decide on relative priorities. This permits one to invest appropriately as one would to protect any valuable resource. In a general sense, the core management systems for the network and for the organizational decision

makers are most critical. Of those, the ones that allow or would benefit from remote configuration should get early consideration. All of the types of security identified in **Figure 1** on page 5 should be considered.

Significant costs will be incurred to update software, hardware, personnel training and for application integration efforts. For operating organizations, refitting management platforms, just to manage security applications might be prohibitively expensive. However, if a combined network management and security operations capability is considered, then deployment and operational costs may be shared. If combined with recapitalization of network management systems that require Year 2000 fixes, then justification of deployment costs becomes even more palatable.

Few existing security management tools that now exist are extensible into the CSSA environment. Replacement of management software and possibly hardware platforms is a migration issue in moving toward the objective architecture. Much of the cost for SNMPv3 software development will fall upon management platform, management application, and SNMPv3 agent vendors. However, some major SNMP vendors were "burned" by the demise of SNMPv2 and they may be slow to invest again before a clear market demand for secure management builds up. The application vendors that get started early in adding secure remote management will have a marketing advantage until other vendors can catch up.

Beyond platform and application software development, a major effort to integrate security management capabilities with emerging security and sensitive operational applications must be addressed. Additional MIB development and agent–application interface synthesis is needed by product developers, user organizations and component integrators. A few major vendors may implement the lion's share of SNMPv3 management platforms and a few security application vendors will quickly add remote management features to mainline products. On the other hand, a great many unique and niche applications will require MIB development and application integration.

For most organizations, migration from SNMPv1 capabilities to SNMPv3 will be a gradual process, possibly over several years. Most SNMPv3 managers and proxies will be able to talk down to SNMPv1 agents; therefore, a mixed environment can exist peacefully. Many SNMPv1 devices that support

non-critical functions or resources and applications that do not need privacy or remote configuration will exist indefinitely in the network.

To permit rapid and convenient integration with security management tools, SNMPv3 services need to be interoperable and accessible via standard APIs. The SNMP standards do not define manager or agent side APIs since APIs do not affect external behavior between SNMP entities. That is left as a local implementation matter. Thus, vendor implementations of management platforms and agents will determine the ease of integration between SNMPv3 services and management applications and between SNMPv3 agents and security applications.

7.3. Topics for Further Research

An important outcome of any significant research effort is not only the knowledge gained and the lessons learned, but also the indicators to additional research topics that offer potential for future benefit. The truism that deeper study of a topic seems to bring more questions than answers is maintained in the context of SM. Below we identify the research areas that appear to be particularly promising.

7.3.1. MIBs for Security Applications

Additional work is required to add new security application MIBs and to refine those that are in work. With consensus on SNMPv3 management protocols, the focus of attention should be to agree on widely applicable monitoring and control structures for virus checkers, certificate authorities, etc.

7.3.2. Integrated Simulation Tools

Use of a simulation tool along with audit data, incident logs, or real-time event data has the potential to test a proposed administration state against a range of operational conditions to assess its desirability in the real world.

7.3.3. Automated Security Policies

With better monitoring capabilities and improved assessment and correlation methods, the importance of precise, consistent, and enforceable security policies grows. Development of techniques for

creating and validating automated security policies is needed as a companion effort to the existing IETF work to establish a Security Policy Specification Language [24].

7.3.4. Performance Testing of SNMPv3-based Management

Early SNMPv3 testing is focused more on interoperability than on performance measures, but vendors and users are interested in anticipated responsiveness, traffic loading, processing burden, and memory/storage needs. Both experimental and simulation results are desirable to validate or refute vendor claims of performance.

7.3.5. IDS Assessment Methods

IDS assessment methods and logic as applied and integrated with the correlation function for sorting, matching, and ranking the security significance of events and sequences of events is an ongoing research area. It needs to be more closely integrated with sensors in all security applications.

7.3.6. Role-based Access Control Implementation

We have not developed the Administration Policy Group of the current SMIB to implement RBAC concepts, but the View-based Access Control Model within SNMPv3 makes it appear that the two would match well. The development of a scenario and a prototype implementation could provide insights and clarify whether there are compatibility issues.

7.3.7. Standard APIs for Agent and Manager-Side Interfaces

SNMP standards groups have always considered API definitions (between an SNMP agent and the managed application or the SNMP management engine and the management application) to be an implementation matter. However, consistency and openness grow in importance as applications on both sides of SNMP (agent and manager) become more complex and varied. Since SNMPv3 incorporates several security services into its architecture, access to those services using GSSAPI calls may be feasible. As a minimum, a consistent, normalized approach is desirable.

Appendix A

Security MIB (SMIB)

```
 -- ************************************************
-- * Copyright 1999 Philip C. Hyland All Rights Reserved      *
-- *                                                          *
-- *                  Security MIB Descriptions               *
 -- ************************************************

SECURITY-MIB DEFINITIONS ::= BEGIN

IMPORTS
mib-2, DisplayString                                    FROM RFC1213-MIB
-- RFC 1902
MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, IpAddress, enterprises, experimental,
Counter32, Integer32                                    FROM SNMPv2-SMI
-- RFC 1903
TEXTUAL-CONVENTION, TimeStamp                           FROM SNMPv2-TC
-- RFC 1904
MODULE-COMPLIANCE, OBJECT-GROUP                         FROM SNMPv2-CONF;

-- Upon publication as RFC, delete this comment and the line following this comment and change the
-- reference of { smibDraft 1 } (above) to { mib-2 X }.  This will result in changing:
-- 1 3 6 1 3 4400 2220 to:              [4400↔GMU and 2220↔sMIB]
-- 1 3 6 1 2 1 sMIB
-- This will make it easier to translate prototypes to the standard namespacebecause the lengths of the OIDs
-- won't change.
sMIB MODULE-IDENTITY
LAST-UPDATED "9812150000Z"
ORGANIZATION "George Mason University"
CONTACT-INFO
      "   Philip C. Hyland
          Postal: George Mason University
          4400 University Ave
          Fairfax, VA 22060
          Tel: 703-993-1499
          Fax: 703-993-xxxx

          E-mail: phyland@gmu.edu"
DESCRIPTION  "The core security MIB module for management of security applications."
::= { experimental gmu sMIB }
```

```
gmu       OBJECT IDENTIFIER ::= { experimental 4400 }
 sMIB     OBJECT IDENTIFIER ::= { gmu 2220 }
-- Complete sMIB designation: ( 1 3 6 1 3 4400 2220 )
-- iso.org.dod.internet.experimental.gmu.sMIB

smibSys          OBJECT IDENTIFIER ::= { sMIB 1 }
smibSP           OBJECT IDENTIFIER ::= { sMIB 2 }
smibAP           OBJECT IDENTIFIER ::= { sMIB 3 }
smibLog          OBJECT IDENTIFIER ::= { sMIB 4 }
smibTrap         OBJECT IDENTIFIER ::= { sMIB 5 }
```

-- ******** The System Security Group **********

```
sysSecName       OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "A unique textual identification for this security system."  ::= { smibSys 1 }

sysSecDomain     OBJECT-TYPE
SYNTAX DisplayString (~Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The unique DNS-like identity for a security domain that contains associated hosts."
::= { smibSys 2 }

sysAdminContactOBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
MAX-ACCESS  read-only
STATUS  Mandatory
DESCRIPTION  "The name and location information (phone, room, address) of the contact person
regarding operations of this system." ::= { smibSys 3 }

sysSecMgr        OBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
MAX-ACCESS  read-only
STATUS  Mandatory
DESCRIPTION  "The name and contact information for the security manager responsible for the security
operations of this system." ::= { smibSys 4 }

sysPublicKey     OBJECT-TYPE
SYNTAX  OctetString (Size (0..255))
MAX-ACCESS  read-only
STATUS  Mandatory
DESCRIPTION  "The public key of the SM entity that permits a confederated station to check the validity
of information from this system." ::= { smibSys 5 }

sysAdminState    OBJECT-TYPE
SYNTAX  INTEGER {Invalid (0), Okay (1), Initializing (2), Self-testing (3), DES Key Invalid (4),
Authentication Key Invalid (5), Other (Unknown) (6) }
MAX-ACCESS  read-only
STATUS  Mandatory
```

DESCRIPTION "The public (non-sensitive) status of the SM entity that permits an external station to check the health of this system." ::= { smibSys 6 }

sysCommand OBJECT-TYPE
SYNTAX INTEGER { Cold Boot (1), Warm Boot (2), Reset to Factory Default Configuration (3), Reset to Local Default Configuration (4), Clear Counters (5), Security Diagnostics (6), Dump TopN Events (7) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "When operations appear inconsistent or reset is otherwise deemed necessary, the system manager can use this parameter to get to a known state. In the read mode, it indicates the last successful command" ::= { smibSys 7 }

sysLastSecEventType OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The type of the last security event." ::= { smibSys 8 }

sysLastSecEventTime OBJECT-TYPE
SYNTAX timeticks
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The time since agent startup of most recent event relevant to security."
::= { smibSys 9 }

sysTimeSincePreviousEvent OBJECT-TYPE
SYNTAX timeticks
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The time from the most recent event to the one previous to that."
::= { smibSys 10 }

smibObjectiveIPL OBJECT-TYPE
SYNTAX INTEGER { Invalid (0), Minimal (1), Moderate (2), Medium (3), High (4), VeryHigh (5) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The desired Information Protection Level. It is used by entities as needed to set the level of trap generation or indicate the destination of notifications." ::= { smibSys 11 }

smibCurrentIPL OBJECT-TYPE
SYNTAX INTEGER { Invalid (0), Minimal (1), Moderate (2), Medium (3), High (4), VeryHigh (5) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The local device view of the proper Information Protection Level." ::= { smibSys 12 }

smibKeepAliveInterval OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Identifies the time (in seconds) between keepalive packets required from the agent to indicate proper functioning." ::= { smibSys 13 }

smibKeepAliveThreshold OBJECT-TYPE

```
SYNTAX  INTEGER
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "Identifies the number of keepalive cycles that do not complete successfully before
initiating a severe TRAP."  ::= { smibSys 14 }

sysApplTable      OBJECT-TYPE
SYNTAX SEQUENCE OF SysApplEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A list of system entries and their corresponding properties." ::= { smibSys 15 }

sysApplEntry      OBJECT-TYPE
SYNTAX SysApplEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION  "A system entry that shows which security applications have registered with this SM
station."
INDEX { sysAppIndex } ::= { sysApplTable 1 }

SysApplEntry ::= SEQUENCE {
sysApplIndex              INTEGER,
sysApplName               DisplayString,
sysApplVersion            DisplayString,
sysApplHost               IpAddress,
sysApplServerProtocol     INTEGER,
sysApplServerPortNum      INTEGER,
sysApplType               INTEGER,
sysApplSecReqmts          INTEGER,
sysApplParam              DisplayString  }

sysApplIndex      OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "A unique code for this table of system application attributes."  ::= { sysApplEntry 1 }

sysApplName      OBJECT-TYPE
SYNTAX  OBJECT IDENTIFIER
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "A unique identification for the security application." ::= { sysApplEntry 2 }

applVersion       OBJECT-TYPE
SYNTAX  DisplayString (Size (0..15))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The specific release and revision level of this application in the format XX.xx  for major
and minor version and release numbers." ::= {sysApplEntry 3 }

sysApplHost       OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS  read-write
```

STATUS Mandatory
DESCRIPTION "The IP address of the host on which the security application runs."
::= { sysApplEntry 4 }

applServer Protocol OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The protocol identity as defined by the Internet Assignened Numbers Authority (IANA) that is used by this application. Use of multiple protocols requires additional rows in the table" ::= {sysApplEntry 5 }

applServerPortNum OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The port number used by this application. Use of multiple ports requires additional rows in the table" ::= {sysApplEntry 6 }

sysApplType OBJECT-TYPE
SYNTAX INTEGER {Invalid (0), Incoming Packet Filter (1), Duplex Packet Filter (2), SMTP proxy (3), TCP proxy (4), TELNET proxy (5), HTTP proxy (6), FTP proxy (7), rlogin proxy (8), WAIS proxy (9), X11 proxy (10),), SNMP proxy (11), NNTP proxy (12), plug-gw proxy (13), IRC proxy (14) , tcp_wrapper (15), authentication server (16), Security Guard (17), Key Management (18), Certificate Authority (19), Intrusion Detection System (20), Audit Trail System (21), Virus Checker (22), Access Controller (23), Key Generator (24) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The functional type of the security application running on the device. If more than one security application runs on one device, there will be multiple application table entries with the same host address." ::= { sysApplEntry 7 }

sysApplSecReqmts OBJECT-TYPE
SYNTAX INTEGER { Encryption (1), Authentication (2), Data Integrity Code (4), Incoming Filter (8), Outgoing Filter (16) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The bitwise sum of these values identify the security needs of the associated application to protect the system and its users from security breaches." ::= {sysApplEntry 8 }

sysApplParam OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The specific parameters that this application uses or requires such as key IDs, modes of operation, etc. May vary depending on security requirements indicated above." ::= {applEntry 9 }

sysConfedTable OJECT-TYPE
SYNTAX SEQUENCE OF SysConfedEntry
MAX-ACCESS not-accessible
STATUS Mandatory

DESCRIPTION "A list of confederated systems (for SMIP) and their corresponding properties."
::= { smibSys 16 }

sysConfedEntry OBJECT-TYPE
SYNTAX SysConfedEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A system entry shows the security management nodes that have been registered with this SM station for SMIP exchanges."
INDEX { sysConfedIndex } ::= { sysConfedTable 1 }

SysConfedEntry ::= SEQUENCE {
sysConfedIndex INTEGER,
sysConfedHostName DisplayString,
sysConfedHostAddress IpAddress,
sysConfedHostGroup DisplayString,
sysConfedType INTEGER,
sysConfedNotificationEnable INTEGER,
sysConfedDESKeyID INTEGER,
sysConfedAuthKeyID INTEGER }

sysConfedIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "A unique code for this table of confederated SM systems and attributes."
::= { sysConfedEntry 1 }

sysConfedHostName OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "A unique textual identification for the confederated SM node."
::= { sysConfedEntry 2 }

sysConfedHostAddress OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The IP address of the host on which the security application runs."
::= { sysConfedEntry 3 }

sysConfedHostGroup OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The name of the group with which this host is associated."
::= { sysConfedEntry 4 }

sysConfedType OBJECT-TYPE
SYNTAX INTEGER { Trust Traps for non-critical actions (1), Trust Traps fully (2), Trust SMIP-Notifications for non-critical actions (4), Trust SMIP-Notifications fully (8), Accept SMIP-queries when IPL=1 (16), Accept SMIP-queries when IPL=2 (32), Accept SMIP-queries when IPL=3 (64), Accept

SMIP-queries when IPL=4(128), Accept SMIP-queries when IPL=5 (256), Accept Authentication Requests (512) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The types of interactions the remote system agrees to accept. Value is the bitwise sum of each applicable 'powers of two' term"
DEFVAL { 0 } ::= { sysConfedEntry 5 }

sysConfedNotificationEnable OBJECT-TYPE
SYNTAX INTEGER { Disable (0), Enable (1) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The switch to send notifications to this host according to the setting of sysConfedType and the IPL." ::= { sysConfedEntry 6 }

sysConfedDESKeyID OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The unique identifier of the DES key associated with this security application."
::= { sysConfedEntry 7 }

sysConfedAuthKeyID OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The unique identifier for the authentication key associated with this security application."
::= { sysConfedEntry 8 }

-- ******** The Security Policy Group *************

spLastUpdate OBJECT-TYPE
SYNTAX UTCTime
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The time of the most recent security policy update." ::= { smibSP 1 }

spUpdatePOC OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The name and contact information for the security manager/officer made the most recent security updates of the security policy." ::= { smibSP 2 }

spRuleTable OBJECT-TYPE
SYNTAX SEQUENCE OF SpRuleEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A list of security policy entries and their corresponding properties." ::= { smibSP 3 }

spRuleEntry OBJECT-TYPE
SYNTAX SpRuleEntry
MAX-ACCESS not-accessible

```
STATUS Mandatory
DESCRIPTION "A list of security policy rules and their corresponding properties."
INDEX { spRuleIndex } ::= { spRuleTable 1 }

SpRuleEntry ::= SEQUENCE {
spRuleIndex       INTEGER,
spRuleName        DisplayString,
spRuleType        INTEGER,
spRuleExpiration UTCTime,
spRuleTarget      DisplayString,
spRuleSubject DisplayString,
spRuleAction      INTEGER
spRulePermissions          INTEGER
spRuleParameters           DisplayString }

spRuleIndex       OBJECT-TYPE
SYNTAX            INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS            Mandatory
DESCRIPTION  "A unique code for this table of security policy rules." ::= { spRuleEntry 1 }

spRuleName        OBJECT-TYPE
SYNTAX  DisplayString (Size (0..127))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The common name for this security rule." ::= {spRuleEntry 2 }

spRuleType        OBJECT-TYPE
SYNTAX INTEGER {Invalid (0), Authentication (1), Access Control (2), Confidentiality (3), Data
Integrity (4), Non-Repudiation (5)}
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The category of the security policy rule being defined." ::= { spRuleEntry 3 }

spRuleExpirationDate       OBJECT-TYPE
SYNTAX  UTCTime
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The time at which this security rule becomes invalid." ::= { spRuleEntry 4 }

spRuleTarget      OBJECT-TYPE
SYNTAX            DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The textual identification of the resource to which this security rule applies.  If an ASN.1
or similar encoding for computer resources is available, that may be used instead.  The word 'All' has
special meaning.  SP rule defaults are to deny all resources to all subjects."
DEFVAL { "All" } ::= { spRuleEntry 5 }

spRuleSubject     OBJECT-TYPE
SYNTAX            DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
```

DESCRIPTION "The textual identification of the entities against which this security rule applies. Typically, this would be a domain or IP address and subnet mask. The word 'All' has special meaning. SP rule defaults are to deny all resources to all subjects."
DEFVAL { "All" } ::= { spRuleEntry 6 }

spRuleAction OBJECT-TYPE
SYNTAX INTEGER { Invalid (0), Deny (1), Permit (2), Require (3), Other (4) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "An encoding of the subject or subject group to which this security rule applies. SP rule defaults are to deny all resources to all subjects."
DEFVAL { 1 } ::= { spRuleEntry 7 }

spRulePermissions OBJECT-TYPE
SYNTAX INTEGER (000..777)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The UNIX-style access to resources permitted by this security rule."
DEFVAL { 000 } ::= { spRuleEntry 8 }

spRuleParameters OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The additional configuration information needed to apply this security rule."
::= { spRuleEntry 9 }

-- ******** The Administration Policy Group *********

adminSeniorSecOfficer OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "A unique textual identification for the senior security officer for this system."
::= { smibAP 1 }

adminUpdatePolicy OBJECT-TYPE
SYNTAX INTEGER {Invalid (0), Senior Security Officer Alone (1), SSO plus one SM (2), Any Two Security Managers (3) }
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION " An encoding of the approval authority needed to update the security MIB configuration parameters." ::= { smibAP 2 }

adminMgrTable OBJECT-TYPE
SYNTAX SEQUENCE OF AdminMgrEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A list of security policy entries and their corresponding properties." ::= { smibAP 3 }

adminMgrEntry OBJECT-TYPE
SYNTAX AdminMgrEntry
MAX-ACCESS not-accessible

```
STATUS Mandatory
DESCRIPTION "A list of security policy rules and their corresponding properties."
INDEX { adminMgrIndex } ::= { adminMgrTable 1 }

AdminMgrEntry ::= SEQUENCE {
adminMgrIndex   INTEGER,
adminMgrID      DisplayString,
adminMgrType    INTEGER,
adminMgrExpiration      UTCTime,
adminMgrTarget DisplayString,
adminMgrAction INTEGER
adminMgrPermissions     INTEGER
adminMgrParameters      DisplayString,
adminMgrDescription     DisplayString }

adminMgrIndex   OBJECT-TYPE
SYNTAX          INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS          Mandatory
DESCRIPTION  "A unique code for this table of administration manager rules." ::= { adminMgrEntry 1 }

adminMgrID      OBJECT-TYPE
SYNTAX  DisplayString (Size (0..127))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The common name for this manager." ::= {adminMgrEntry 2 }

adminMgrType    OBJECT-TYPE
SYNTAX INTEGER { Security Operator (1), Security Manager (2), Senior Security Officer (3) }
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The category of the security person for which rules being defined."
::= { adminMgrEntry 3 }

adminMgrExpirationDate  OBJECT-TYPE
SYNTAX  UTCTime
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The time at which these security rules become invalid." ::= { adminMgrEntry 4 }

adminMgrRuleTarget      OBJECT-TYPE
SYNTAX          DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The textual identification of the resource to which this security rule applies (protects or
opens).  If an ASN.1 or similar encoding for computer resources is available, that may be used instead."
::= { adminMgrEntry 5 }

adminMgrAction OBJECT-TYPE
SYNTAX          INTEGER { Invalid (0), Deny (1), Permit (2), Require (3), Other (4) }
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "An encoding of the effect of this security rule." ::= { adminMgrEntry 6 }
```

adminMgrPermissions OBJECT-TYPE
SYNTAX INTEGER (000..777)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The UNIX-style access permitted if security rule is related to access."
::= { adminMgrEntry 7 }

adminMgrParameters OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The additional configuration information needed to apply this security rule."
::= { adminMgrEntry 8 }

adminMgrDescription OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The textual description of the purpose and intent of this management rule."
::= { adminMgrEntry 9 }

********** **The Security Log Group** **********

sysLogAdministrator OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The textual identity and contact information of the person responsible for the security log." ::= { smibLog 1 }

sysLogMaxEntries OBJECT-TYPE
SYNTAX INTEGER32
STATUS Mandatory
MAX-ACCESS read-write
DEFINITION " The maximum number of entries in the sysLogTable." ::= { smibLog 2 }

sysPercentLogFilled OBJECT-TYPE
SYNTAX Gauge
STATUS Mandatory
MAX-ACCESS read-only
DEFINITION " The state of the log capacity due to recording log entries."
DEFVAL { 0 }::= { smibLog 3 }

sysLogStart OBJECT-TYPE
SYNTAX UTCTime
STATUS Mandatory
MAX-ACCESS read-write
DEFINITION " The beginning of the current log as identified by the earliest log entry."
::= { smibLog 4 }

sysLogServer OBJECT-TYPE

SYNTAX IpAddress
STATUS Mandatory
MAX-ACCESS read-write
DEFINITION " The host where the log entries are to be directed. Typically it is the local host."
DEFVAL { 127.0.0.1 }::= { smibLog 5 }

sysLogDir OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The directory for the log file for the security application." ::= { smibLog 6 }

sysLogLevel OBJECT-TYPE
SYNTAX INTEGER { Other (1), FW Threshold: Packet Counters/Rules (2), Packet Source Address/Rule for Warnings (4), Packet Header/Rule for Warnings (8), Full Packet/Rule for Warnings (16), Packet Source Address/Rule for Alerts (32), Packet Header/Rule for Alerts (64), Full Packet/Rule for Alerts (128), FW Threshold: NumberProxySessions (256), FW Threshold: ProxySessionsByService (512), FW Threshold: Proxy Warnings (1024), FW Threshold: ProxyAlerts (2048) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The desired logging level for the assorted events." ::= { smibLog 7 }

smibLogInterval OBJECT-TYPE
SYNTAX INTEGER (0..65535)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The maximum number of seconds to wait for additional log data before writing data to the log server."
DEFVAL { 60 } ::= { smibLog 8 }

sysLogTable OBJECT-TYPE
SYNTAX SEQUENCE OF SysLogEntry
STATUS Mandatory
MAX-ACCESS not-accessible
DEFINITION " A list of local log entries that have been reported to this management entity as related to system security events." ::= { smibLog 9 }

sysLogEntry OBJECT-TYPE
SYNTAX SysLogEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A system log entry contains attributes that define security events that have been reported to the system security manager."
INDEX { sysLogIndex, sysLogTargetHost, sysLogAttackHost, sysLogEventTime, sysLogEventSeverity, sysLogEventType } ::= { sysLogTable 1 }

SysLogEntry ::= SEQUENCE {
sysLogIndex INTEGER32,
sysLogEventTime UTCTime,
sysLogReportingHost IpAddress,
sysLogTargetHost IpAddress,
sysLogTargetResourceID DisplayString,
sysLogAttackHost IpAddress,

```
sysLogAttackerID        DisplayString,
sysLogEventSeverity     INTEGER,
sysLogEventType         INTEGER,
sysLogMessage           DisplayString,
sysLogEventDuration     TimeTicks
sysLogSecLevel          snmpSecurityLevel }

sysLogIndex     OBJECT-TYPE
SYNTAX  INTEGER32 (1..2147483647)
STATUS  Mandatory
MAX-ACCESS  read-only
DEFINITION  "A unique value for referencing each logged system event." ::= { sysLogEntry 1 }

sysLogEventTime         OBJECT-TYPE
SYNTAX          UTCTime
STATUS  Mandatory
MAX-ACCESS  read-only
DEFINITION  " The time the event report originated." ::= { sysLogEntry 2 }

sysLogReportingHost     OBJECT-TYPE
SYNTAX  IpAddress
STATUS  Mandatory
MAX-ACCESS  read-only
DEFINITION  " The source host from which the event report originated." ::= { sysLogEntry 3 }

sysLogTargetHost        OBJECT-TYPE
SYNTAX  IpAddress
STATUS  Mandatory
MAX-ACCESS  read-only
DEFINITION  "The apparent target of the reported incident." ::= { sysLogEntry 4 }

SysLogTargetResourceID  OBJECT-TYPE
SYNTAX  DisplayString (Size (00..255))
STATUS  Mandatory
MAX-ACCESS  read-write
DEFINITION  "The textual identity of the target host resources, accounts, etc that were attacked"
::= { sysLogEntry 5 }

sysLogAttackHost        OBJECT-TYPE
SYNTAX  IpAddress
STATUS  Mandatory
MAX-ACCESS  read-only
DEFINITION  " The host from which the incident appears to have originated." ::= { sysLogEntry 6 }

sysLogAttackerID        OBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
STATUS  Mandatory
MAX-ACCESS  read-write
DEFINITION  "The textual identity of the attacker as far as is known.  This value may be updated by the
event correlator based on related events" ::= { sysLogEntry 7 }

sysLogEventSeverity             OBJECT-TYPE
```

SYNTAX INTEGER { invalid (0), severe (1), serious (2), warning (3), informational (4), minor (5), administration (6) }
STATUS Mandatory
MAX-ACCESS read-write
DEFINITION "Indicates the seriousness of a particular incident." ::= { sysLogEntry 8 }

sysLogEventType OBJECT-TYPE
SYNTAX INTEGER { Invalid (0), FW1 (1), FW2 (2), FW3 (3), FW4 (4), FW5 (5), VC1 (6), VC2 (7), VC3 (8), VC4 (9), VC5 (10), SG1 (11), SG2 (12), SG3 (13), SG4 (14), SG5 (15), AS1 (16), AS2 (17), AS3 (18), AS4 (19), AS5 (20), KM1 (21), KM2 (22), KM3 (23), KM4 (24), KM5 (25), AC1 (26), AC2 (27), AC3 (28), AC4 (29), AC5 (30), IDS1, (31), IDS2 (32), IDS3 (33), IDS4 (34), IDS5 (35) }
STATUS Mandatory
MAX-ACCESS read-only
DEFINITION "This field specifies the type of event that occurred. The five most common events of each reporting security application is listed, but additional types and applications can/will be defined"
::= { sysLogEntry 9 }

sysLogMessage OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The value of this object contains the system message that normally would be generated if displayed to the console." ::= { sysLogEntry 10 }

sysLogEventDuration OBJECT-TYPE
SYNTAX TimeTicks
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The amount of time that the logged incident appears to have lasted. An initial event report will list this parameter as zero, however, an event correlator can make associations between subsequent/previous reports and may explicitly modify this parameter" ::= { sysLogEntry 11 }

sysLogEventSecurity OBJECT-TYPE
SYNTAX snmpSecurityLevel
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The required security protection for distribution of the event." ::= {sysLogEntry 12 }

********** **The SMIB Trap Group** **********

sysSecTrap NOTIFICATION-TYPE
OBJECTS {
sysLogIndex,
sysLogEventTime,
sysLogReportingHost,
sysLogEventType,
sysLogEventSeverity }
STATUS current
DESCRIPTION "Security event notification from a security application." ::= { smibTrap 1 }

END

Appendix B

SMON MIB

The Security Monitoring MIB (SMON) is a distributed MIB that gathers security event information that is best retained at the collection location. Although the SMON idea came from RMON, the two have no other direct connection, although a number of RMON monitoring parameters have been adapted to SMON. SMON uses SNMPv3 agents attached to security applications. These security applications may provide end user services or be deployed as dedicated security probes (e.g., IDS) to permit central visibility of remote security capabilities and status.

```
-- ************************************************
-- * Copyright 1999 Philip C. Hyland All Rights Reserved *
-- *                                                    *
-- *              Security Monitoring MIB Descriptions  *
-- ************************************************

SMON-MIB DEFINITIONS ::= BEGIN

IMPORTS
mib-2, DisplayString                                    FROM RFC1213-MIB
-- RFC 1902
MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, IpAddress, enterprises, experimental,
Counter32, Integer32                                    FROM SNMPv2-SMI
-- RFC 1903
TEXTUAL-CONVENTION, TimeStamp                           FROM SNMPv2-TC
-- RFC 1904
MODULE-COMPLIANCE, OBJECT-GROUP                         FROM SNMPv2-CONF;

-- Upon publication as RFC, delete this comment and the line following
-- this comment and change the reference of { smonDraft 1 }
-- (above) to { mib-2 X }.
-- This will result in changing:
-- 1 3 6 1 3 4400 2221 to:          [4400↔GMU and 2221↔smon]
-- 1 3 6 1 2 1 smon
-- This will make it easier to translate prototypes to the standard namespace
```

```
-- because the lengths of the OIDs won't change.

smonMIB MODULE-IDENTITY
LAST-UPDATED "9812150000Z"
ORGANIZATION "George Mason University"
CONTACT-INFO
      "   Philip C. Hyland
          Postal: George Mason University
          4400 University Ave
          Fairfax, VA 22060
          Tel: 703-993-1499
          Fax: 703-993-xxxx
          E-mail: phyland@gmu.edu"
DESCRIPTION  "The security monitoring MIB module for management of security applications."
::= { experimental gmu smonMIB }
gmu       OBJECT IDENTIFIER ::= { experimental 4400 }
smonMIB           OBJECT IDENTIFIER ::= { gmu 2221 }
-- Complete smonMIB designation: ( 1 3 6 1 3 4400 2221 )
-- iso.org.dod.internet.experimental.gmu.smonMIB

SnmpSecurityLevel ::= TEXTUAL-CONVENTION
STATUS       current
DESCRIPTION "A Level of Security at which SNMP messages can be sent or with which operations are
being processed; in particular, one of:
  noAuthNoPriv - without authentication and without privacy,
  authNoPriv   - with authentication but without privacy,
  authPriv     - with authentication and with privacy.  These three values are ordered such that
noAuthNoPriv is less than authNoPriv and authNoPriv is less than authPriv."
SYNTAX       INTEGER { noAuthNoPriv(1), authNoPriv(2), authPriv(3) }

smonHosts                 OBJECT IDENTIFIER ::= { smonMIB 1 }
smonHistoryControl        OBJECT IDENTIFIER ::= { smonMIB 2 }
smonHostsHistory          OBJECT IDENTIFIER ::= { smonMIB 3 }
smonAlarms                OBJECT IDENTIFIER ::= { smonMIB 4 }
smonSecApplTopN           OBJECT IDENTIFIER ::= { smonMIB 5 }
smonSuspects              OBJECT IDENTIFIER ::= { smonMIB 6 }
smonNotifications         OBJECT IDENTIFIER ::= { smonMIB 7 }
smonEventDataRequest      OBJECT IDENTIFIER ::= { smonMIB 8 }

-- ********** The Hosts Group **************************
smonHostsTable  OBJECT-TYPE
SYNTAX SEQUENCE OF SmonHostsEntry
MAX-ACCESS not-accessible
STATUS mandatory
DESCRIPTION  "A list of host statistics entries." ::= { smonHosts 1 }

smonHostsEntry  OBJECT-TYPE
SYNTAX  SmonHostsEntry
MAX-ACCESS  not-accessible
STATUS mandatory
DESCRIPTION  "A collection of security statistics kept for a particular host."
INDEX { smonHostsIndex } ::= { smonHostsTable 1 }
```

```
SmonHostsEntry ::= SEQUENCE {
smonHostsIndex  INTEGER,
smonHostsDataSource      DisplayString,
smonHostsLogins          Counter,
smonHostsBadLogins       Counter,
smonTCPSYNs   Counter32,
smonPortScans   Counter32 }

smonHostsIndex  OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION  "The value of this object uniquely identifies this smonHosts entry."
::= { smonHostsEntry 1 }

smonHostsDataSource      OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION  "This object identifies the source of the data in this entry.  As a minimum this should
include the IP address and a resource/user sub-identifier." ::= { smonHostsEntry 2 }

smonHostsLogins          OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION  "This is the total number of login attempts for this host." ::= { smonHostsEntry 3 }

smonHostsBadLogins       OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION  "This is the total number of failed login attempts for this host." ::= { smonHostsEntry 4 }

smonTCPSYNs   OBJECT-TYPE
SYNTAX  Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION  "The total number of TCP SYNs received at this host." ::= { smonHostsEntry 5 }

smonPortScans   OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION  "The total number of port scans received at this host." ::= { smonHostsEntry 6 }
```

-- ******** The History Control Group **********************

```
smonHostsControlTable OBJECT-TYPE
SYNTAX SEQUENCE OF SmonHostsControlEntry
MAX-ACCESS not-accessible
STATUS mandatory
```

DESCRIPTION "A list of host history control entries." ::= { smonHistoryControl 1 }

smonHostsControlEntry OBJECT-TYPE
SYNTAX HostsControlEntry
MAX-ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A list of parameters that set up a periodic sampling of statistics."
INDEX { hostsControlIndex } ::= { hostsControlTable 1 }

SmonHostsControlEntry ::= SEQUENCE {
HostsControlIndex INTEGER,
HostsControlDataSource OBJECT IDENTIFIER,
HostsControlBucketsRequested INTEGER,
HostsControlBucketsGranted INTEGER,
HostsControlInterval INTEGER,
HostsControlOwner OwnerString,
HostsControlStatus EntryStatus }

HostsControlIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "An index that uniquely identifies an entry in the smonHostsControl table. Each such entry defines a set of samples at a particular interval on the device." ::= { smonHostsControlEntry 1 }

hostsControlDataSource OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "This object identifies the source of the data for which historical data was collected. As a minimum this should include the IP address and a resource/user sub-identifier, which may be an object identifier string. This object may not be modified if the associated hostsControlStatus object is equal to valid (1)." ::= { smonHostsControlEntry 2 }

hostsControlBucketsRequested OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The requested number of discrete time intervals over which data is to be saved. The probe should set hostsControlBucketsGranted as closely to this object as is possible for the particular probe implementation and available resources."
DEFVAL { 50 } ::= { smonHostsControlEntry 3 }

HostsControlBucketsGranted OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of discrete sampling intervals over which data shall be saved. This should be as close to the requested value as possible for the particular implementation and available resources. This value should not be lowered except as a result of a modification to the associated hostsControlBucketsRequested object. When the actual number of buckets is less than the value of this object at the end of each sampling interval, a new bucket will be added to the media-specific table. When the number of buckets reaches the value of this object, the oldest bucket shall be deleted a new bucket

needs to be added. When the value of this object changes to a value less than the current value, the oldest of these entries shall be deleted by the agent so that their number remains less than or equal to the new value of this object. When the value of this object changes to a value greater than the current value, the number of entries may be allowed to grow." ::= { smonHostsControlEntry 4 }

hostsControlInterval OBJECT-TYPE
SYNTAX INTEGER (1..3600)
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The interval in seconds over which the data is sampled for each bucket. Consider the minimum time in which any counter could overflow on a particular media type and set the hostsControlInterval object to a value less than this interval. This object may not be modified if the associated hostsControlStatus object is equal to valid (1)."
DEFVAL { 1800 } ::= { smonHostsControlEntry 5 }

hostsControlOwner OBJECT-TYPE
SYNTAX OwnerString
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The entity that configured this entry and is therefore using the resources assigned to it."
::= { smonHostsControlEntry 6 }

hostsControlStatus OBJECT-TYPE
SYNTAX EntryStatus
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The status of this hostsControl entry. Each instance will be deleted if this hostsControlEntry is not equal to valid (1)." ::= { smonHostsControlEntry 7 }

-- ******** The Hosts History Group **********

smonHostsHistoryTable OBJECT-TYPE
SYNTAX SEQUENCE OF SmonHostsHistoryEntry
MAX-ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A list of historical host statistics entries." ::= { smonHostsHistory 1 }

smonHostsHistoryEntry OBJECT-TYPE
SYNTAX SmonHostsHistoryEntry
MAX-ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A historical collection of security statistics kept for a particular host."
INDEX { smonHostsHistoryIndex } ::= { smonHostsHistoryTable 1 }

SmonHostsHistoryEntry ::= SEQUENCE {
smonHostsHistoryIndex INTEGER,
smonHostsHistorySampleIndex INTEGER,
smonHostsHistoryIntervalStart TimeTicks,
smonHostsHistoryDataSource DisplayString,
smonHostsHistoryLogins Counter,
smonHostsHistoryBadLogins Counter,
smonHistoryTCPSYNs Counter32,
smonHistoryPortScans Counter32 }

smonHostsHistoryIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "The value of this object uniquely identifies this smonHostsHistory entry." :
:= { smonHostsHistoryEntry 1 }

smonHostsHistorySampleIndex OBJECT-TYPE
SYNTAX INTEGER32 (1..2147483647)
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "An index that uniquely identifies the particular sample this entry represents among all samples associated with the same smonHostsControlEntry. This index starts at 1 and increases by one as each new sample is taken." ::= { smonHostsHistoryEntry 2 }

smonHostsHistoryIntervalStart OBJECT-TYPE
SYNTAX TimeTicks
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "The value of sysUpTime at the start of the interval over which this sample was measured. If the time of day is known, the first sample should start when the next hour begins. This may require a delay since each sample must be of the same interval. Samples currently being collected are not accessible until the end of the interval." ::= { smonHostsHistoryEntry 3 }

smonHostsHistoryDataSource OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "This object identifies the source of the data in this entry. As a minimum this should include the IP address and a resource/user sub-identifier." ::= { smonHostsHistoryEntry 4 }

smonHostsLogins OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "This is the total number of login attempts for this host."
::= { smonHostsHistoryEntry 5 }

smonHostsHistoryBadLogins OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "This is the total number of failed login attempts for this host."
::= { smonHostsHistoryEntry 6 }

smonHistoryTCPSYNs OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "The total number of TCP SYNs received at this host." ::= { smonHostsHistoryEntry 7 }

smonHistoryPortScans OBJECT-TYPE

SYNTAX Counter
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "The total number of port scans received at this host." ::= { smonHostsHistoryEntry 8 }

-- ******** The Alarms Group **********

-- Implementation of the Alarm group is optional. The Alarm Group requires the implementation of the
-- smonEvent group. The Alarm group periodically takes statistical samples from variables in the probe
-- and compares them to thresholds that have been configured. The alarm table stores configuration entries
-- that each define a variable, polling period, and threshold parameters. If a sample is found to cross the
-- threshold values, an event is generated. Only variables that resolve to an ASN.1 primitive type of
-- INTEGER (INTEGER, Counter, Gauge, or TimeTicks) may be monitored in this way. This function
-- has a hysteresis mechanism to limit the generation of events. This mechanism generates one event as a
-- threshold is crossed in the appropriate direction. No more events are generated for that threshold until
-- the opposite threshold is crossed. In the case of a sampling a deltaValue, a probe may implement this
-- mechanism with more precision if it takes a delta sample twice per period, each time comparing the sum
-- of the latest two samples to the threshold. This allows the detection of threshold crossings that span the
-- sampling boundary. Note that this does not require any special configuration of the threshold value. It is
-- suggested that probes implement this more precise algorithm.

smonAlarmTable OBJECT-TYPE
SYNTAX SEQUENCE OF SmonAlarmEntry
MAX-ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A list of alarm entries." ::= { smonAlarms 1 }

smonAlarmEntry OBJECT-TYPE
SYNTAX SmonAlarmEntry
MAX-ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A list of parameters for periodic checking of alarm conditions."
INDEX { smonAlarmIndex } ::= { smonAlarmTable 1 }

SmonAlarmEntry ::= SEQUENCE {
smonAlarmIndex INTEGER,
smonAlarmInterval INTEGER,
smonAlarmVariable OBJECT IDENTIFIER,
smonAlarmSampleType INTEGER,
smonAlarmValue INTEGER,
smonAlarmStartupAlarm INTEGER,
smonAlarmRisingThreshold INTEGER,
smonAlarmFallingThreshold INTEGER,
smonAlarmRisingEventIndex INTEGER,
smonAlarmFallingEventIndex INTEGER,
smonAlarmOwner OwnerString,
smonAlarmStatus EntryStatus }

smonAlarmIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS mandatory

DESCRIPTION "An index that uniquely identifies an entry in the alarm table. Each entry defines a diagnostic sample at a particular interval for an object on the device." ::= { smonAlarmEntry 1 }

smonAlarmInterval OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The interval in seconds between which data is sampled and compared with the rising and falling thresholds. Care should be taken in the case of deltaValue sampling to set the interval short enough that the sampled variable is very unlikely to increase or decrease by more than 2^31 - 1 during an interval. This object may not be modified if the associated alarmStatus object is equal to valid (1)."
::= { smonAlarmEntry 2 }

smonAlarmVariable OBJECT-TYPE
SYNTAX OBJECT IDENTIFIER
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The object identifier of the particular variable to be sampled. Only variables that resolve to an ASN.1 primitive type of INTEGER (INTEGER, Counter, Gauge, or TimeTicks) may be sampled. During a set operation, if the supplied variable name is not available in the selected MIB view, a badValue error must be returned. If the variable name of an established alarmEntry is no longer available in the selected MIB view, the probe must change the status of this alarmEntry to invalid (4). This object may not be modified if the associated alarmStatus object is equal to valid (1)." ::= { smonAlarmEntry 3 }

smonAlarmSampleType OBJECT-TYPE
SYNTAX INTEGER { absoluteValue(1), deltaValue(2) }
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The method of calculating the difference between the sample value and the thresholds. For absoluteValue (1), the selected variable will be compared directly with the thresholds at the end of the sampling interval. For deltaValue (2), the last sample will be subtracted from the current value, and the difference compared with the thresholds. This object may not be modified if the associated alarmStatus object is equal to valid (1)." ::= { smonAlarmEntry 4 }

smonAlarmValue OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-only
STATUS mandatory
DESCRIPTION "The value of the statistic during the last sampling period. If the sample type is deltaValue, this value will be the difference between the beginning and end of the period. If the sample type is absoluteValue, this value will be the sampled value at the end of the period. This is the value that is compared with the rising and falling thresholds. The value is not available until the period is completed, but remains available until the next period completes." ::= { smonAlarmEntry 5 }

smonAlarmStartupAlarm OBJECT-TYPE
SYNTAX INTEGER { risingAlarm (1), fallingAlarm (2), risingOrFallingAlarm (3) }
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The alarm that may be sent when this entry is first set to valid. If the first sample after this entry becomes valid is greater than or equal to the risingThreshold and alarmStartupAlarm is equal to risingAlarm (1) or risingOrFallingAlarm (3), then a single rising alarm will be generated. If the first sample after this entry becomes valid is less than or equal to the fallingThreshold and alarmStartupAlarm is

equal to fallingAlarm (2) or risingOrFallingAlarm (3), then a single falling alarm will be generated. This object may not be modified if the associated alarmStatus object is equal to valid (1)."
::= { smonAlarmEntry 6 }

smonAlarmRisingThreshold OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "A threshold for the sampled statistic. When the current sampled value is greater than or equal to this threshold, and the value at the last sampling interval was less than this threshold, a single event will be generated. A single event will also be generated if the first sample after this entry becomes valid is greater than or equal to this threshold and the associated alarmStartupAlarm is equal to risingAlarm (1) or risingOrFallingAlarm (3). After a rising event is generated, another such event will not be generated until the sampled value falls below this threshold and reaches the alarmFallingThreshold. This object may not be modified if the associated alarmStatus object is equal to valid (1)." ::= { smonAlarmEntry 7 }

smonAlarmFallingThreshold OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "A threshold for the sampled statistic. When the current sampled value is less than or equal to this threshold, and the value at the last sampling interval was greater than this threshold, a single event will be generated. A single event will also be generated if the first sample after this entry becomes valid is less than or equal to this threshold and the associated alarmStartupAlarm is equal to fallingAlarm (2) or risingOrFallingAlarm (3). After a falling event is generated, another such event will not be generated until the sampled value rises above this threshold and reaches the alarmRisingThreshold. This object may not be modified if the associated alarmStatus object is equal to valid (1)." ::= { smonAlarmEntry 8 }

smonAlarmRisingEventIndex OBJECT-TYPE
SYNTAX INTEGER (0..65535)
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The index of the smonEventEntry that is used when a rising threshold is crossed. The smonEventEntry identified by a particular value of this index is the same as identified by the same value of the smonEventIndex object. If there is no corresponding entry in the smonEventTable, then no association exists. In particular, if this value is zero, no associated event will be generated, as zero is not a valid event index. This object may not be modified if the associated smonAlarmStatus object is equal to valid (1)."
::= { smonAlarmEntry 9 }

smonAlarmFallingEventIndex OBJECT-TYPE
SYNTAX INTEGER (0..65535)
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION "The index of the smonEventEntry that is used when a falling threshold is crossed. The smonEventEntry identified by a particular value of this index is the same as identified by the same value of the smonEventIndex object. If there is no corresponding entry in the smonEventTable, then no association exists. In particular, if this value is zero, no associated event will be generated, as zero is not a valid event index. This object may not be modified if the associated smonAlarmStatus object is equal to valid (1)."
::= { smonAlarmEntry 10 }

smonAlarmOwner OBJECT-TYPE
SYNTAX OwnerString
MAX-ACCESS read-write

```
STATUS mandatory
DESCRIPTION  "The entity that configured this entry and is therefore using the resources assigned to it."
::= { smonAlarmEntry 11 }

smonAlarmStatus OBJECT-TYPE
SYNTAX EntryStatus
MAX-ACCESS read-write
STATUS mandatory
DESCRIPTION  "The status of this alarm entry." ::= { smonAlarmEntry 12 }
```

-- ******** The Security Application Top N Group **********************

```
-- Implementation of the  SMON Top N group is optional.  The  SMON Top N group requires the
-- implementation of the SMIB Security Applications group.  The SMON Top N group is used to prepare
-- reports that describe the hosts that top a list  ordered by one of their statistics.  The available statistics are
-- samples of one of their base statistics, over an interval specified by the management station.  Thus, these
-- statistics are rate-based.  The management station also selects how many such hosts are reported.
-- The smonSecApplTopNControlTable is used to initiate the generation of such a report.  The management
-- station may select the parameters of such a report, such as which statistic, how many
-- hosts, and the start and stop times of the sampling.  When the report is prepared, entries are created in the
--  smonSecApplTopNTable associated with the relevant  smonSecApplTopNControlEntry.  These entries
-- are static for each report after it has been prepared.

smonSecApplTopNControlTable OBJECT-TYPE
SYNTAX SEQUENCE OF  SmonSecApplTopNControlEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION  "A list of top N security application control entries." ::= {  smonSecApplTopN 1 }

smonSecApplTopNControlEntry OBJECT-TYPE
SYNTAX  SmonSecApplTopNControlEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION  "A set of parameters that control the creation of a report of the top N hosts according to
several metrics.  For example, an instance of the smonSecApplTopNDuration object might be named
smonSecApplTopNDuration.3"
INDEX { smonSecApplTopNControlIndex } ::= { smonSecApplTopNControlTable 1 }

SmonSecApplTopNControlEntry ::= SEQUENCE {
smonSecApplTopNControlIndex    INTEGER (1..65535),
smonSecApplTopNSecApplIndex       INTEGER (1..65535),
smonSecApplTopNRateBase        INTEGER,
smonSecApplTopNTimeRemaining   INTEGER,
smonSecApplTopNDuration        INTEGER,
smonSecApplTopNRequestedSize   INTEGER,
smonSecApplTopNGrantedSize     INTEGER,
smonSecApplTopNStartTime       TimeTicks,
smonSecApplTopNOwner           OwnerString,
smonSecApplTopNStatus          EntryStatus }

smonSecApplTopNControlIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
```

ACCESS read-only
STATUS mandatory
DESCRIPTION "An index that uniquely identifies an entry in the smonSecApplTopNControl table. Each such entry defines one top N report prepared for one interface." ::= { smonSecApplTopNControlEntry 1 }

smonSecApplTopNSecApplIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-write
STATUS mandatory
DESCRIPTION "The security application table for which a top N report will be prepared on behalf of this entry. The table identified by a particular value of this index is associated with the same table as identified by the same value of smonSecApplIndex. This object may not be modified if the associated smonSecApplTopNStatus object is equal to valid(1)." ::= { smonSecApplTopNControlEntry 2 }

smonSecApplTopNRateBase OBJECT-TYPE
SYNTAX INTEGER {
smonSecApplTopNInPkts(1),
smonSecTopNOutPkts(2),
smonSecTopNInOctets(3),
smonSecTopNOutOctets(4),
smonSecTopNOutErrors(5),
smonSecTopNOutBroadcastPkts(6),
smonSecTopNOutMulticastPkts(7) }
ACCESS read-write
STATUS mandatory
DESCRIPTION "The variable for each security application that the smonSecApplTopNRate variable is based upon. This object may not be modified if the associated smonSecApplTopNStatus object is equal to valid (1)." ::= { smonSecApplTopNControlEntry 3 }

smonSecApplTopNTimeRemaining OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION "The number of seconds left in the report currently being collected. When this object is modified by the management station, a new collection is started, possibly aborting a currently running report. The new value is used as the requested duration of this report, which is loaded into the associated smonSecApplTopNDuration object. When this object is set to a non-zero value, any associated smonSecApplTopNEntries shall be made inaccessible by the monitor. While the value of this object is non-zero, it decrements by one per second until it reaches zero. During this time, all associated smonSecApplTopNEntries shall remain inaccessible. At the time that this object decrements to zero, the report is made accessible in the smonSecApplTopNTable. Thus, the smonSecApplTopN table needs to be created only at the end of the collection interval."
DEFVAL { 0 } ::= { smonSecApplTopNControlEntry 4 }

smonSecApplTopNDuration OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION "The number of seconds that this report has collected during the last sampling interval, or if this report is currently being collected, the number of seconds that this report is being collected during this sampling interval. When the associated smonSecApplTopNTimeRemaining object is set, this object shall be set by the probe to the same value and shall not be modified until the next time the

smonSecApplTopNTimeRemaining is set. This value shall be zero if no reports have been requested for this smonSecApplTopNControlEntry."
DEFVAL { 0 } ::= { smonSecApplTopNControlEntry 5 }

smonSecApplTopNRequestedSize OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-write
STATUS mandatory
DESCRIPTION "The maximum number of hosts requested for the top N table. When this object is created or modified, the probe should set smonSecApplTopNGrantedSize as closely to this object as is possible for the particular probe implementation and available resources."
DEFVAL { 10 } ::= { smonSecApplTopNControlEntry 6 }

smonSecApplTopNGrantedSize OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION "The maximum number of applications in the top N table. When the associated smonSecApplTopNRequestedSize object is created or modified, the probe should set this object as closely to the requested value as is possible for the particular implementation and available resources. The probe must not lower this value except as a result of a set to the associated smonSecApplTopNRequestedSize object. Applications with the highest value of smonSecApplTopNRate shall be placed in this table in decreasing order of this rate until there is no more room or until there are no more applicationss."
::= { smonSecApplTopNControlEntry 7 }

smonSecApplTopNStartTime OBJECT-TYPE
SYNTAX TimeTicks
ACCESS read-only
STATUS mandatory
DESCRIPTION "The value of sysUpTime when this top N report was last started. In other words, this is the time that the associated smonSecApplTopNTimeRemaining object was modified to start the requested report." ::= { smonSecApplTopNControlEntry 8 }

smonSecApplTopNOwner OBJECT-TYPE
SYNTAX OwnerString
ACCESS read-write
STATUS mandatory
DESCRIPTION "The entity that configured this entry and is therefore using the resources assigned to it."
::= { smonSecApplTopNControlEntry 9 }

smonSecApplTopNStatus OBJECT-TYPE
SYNTAX EntryStatus
ACCESS read-write
STATUS mandatory
DESCRIPTION "The status of this smonSecApplTopNControl entry. If this object is not equal to valid (1), all associated smonSecApplTopNEntries shall be deleted by the agent."
::= { smonSecApplTopNControlEntry 10 }

smonSecApplTopNTable OBJECT-TYPE
SYNTAX SEQUENCE OF SmonSecApplTopNEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A list of top N security application entries." ::= { smonSecApplTopN 2 }

smonSecApplTopNEntry OBJECT-TYPE
SYNTAX SmonSecApplTopNEntry
ACCESS not-accessible
STATUS mandatory
DESCRIPTION "A set of statistics for a security application that is part of a top N report. For example, an instance of the smonSecApplTopNRate object might be named smonSecApplTopNRate.3.10"
INDEX { smonSecApplTopNReport, smonSecApplTopNIndex } ::= { smonSecApplTopNTable 1 }

SmonSecApplTopNEntry ::= SEQUENCE {
SmonSecApplTopNReport INTEGER (1..65535),
smonSecApplTopNIndex INTEGER (1..65535),
smonSecApplTopNAddress OCTET STRING,
smonSecApplTopNRate INTEGER }

smonSecApplTopNReport OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "This object identifies the top N report of which this entry is a part. The set of security applications identified by a particular value of this object is part of the same report as identified by the same value of the smonSecApplTopNControlIndex object." ::= { smonSecApplTopNEntry 1 }

smonSecApplTopNIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
ACCESS read-only
STATUS mandatory
DESCRIPTION "An index that uniquely identifies an entry in the smonSecApplTopN table among those in the same report. This index is between 1 and N, where N is the number of entries in this table. Increasing values of smonSecApplTopNIndex shall be assigned to entries with decreasing values of smonSecApplTopNRate until index N is assigned to the entry with the lowest value of smonSecApplTopNRate or there are no more smonSecApplTopNEntries."
::= { smonSecApplTopNEntry 2 }

smonSecApplTopNAddress OBJECT-TYPE
SYNTAX OCTET STRING
ACCESS read-only
STATUS mandatory
DESCRIPTION "The physical address of this security application." ::= { smonSecApplTopNEntry 3 }

smonSecApplTopNRate OBJECT-TYPE
SYNTAX INTEGER
ACCESS read-only
STATUS mandatory
DESCRIPTION "The amount of change in the selected variable during this sampling interval. The selected variable is this security application's instance of the object selected by smonSecApplTopNRateBase." ::= { smonSecApplTopNEntry 4 }

-- ******** The Suspects Group **********************

smonSuspectsUpdate OBJECT-TYPE
SYNTAX TimeStamp
MAX-ACCESS read-only

STATUS Mandatory
DESCRIPTION "This identifies the last time the suspect table was archived and/or initialized."
::= { smonSuspects 1 }

smonSuspectsUpdateAction OBJECT-TYPE
SYNTAX INTEGER { invalid (0), reset table (1), log summary report and reset table (2), send notification, log summary report and rest table, (3), pare table (4), log summary report and pare table (5), send notification, log summary report and pare table, (6) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "This identifies the update action to take at the end of the next update interval."
::= { smonSuspects 2 }

smonSuspectsUpdateInterval OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "This identifies the number of minutes (abou 45 days max) before Suspects table is flushed or pared per the smonSuspectsUpdateAction parameter." ::= { smonSuspects 3 }

smonSuspectsTable OBJECT-TYPE
SYNTAX SEQUENCE OF SmonSuspectsEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A list of suspicious user entries." ::= { smonSuspects 4 }

smonSuspectsEntry OBJECT-TYPE
SYNTAX SmonSuspectsEntry
MAX-ACCESS not-accessible
STATUS Mandatory
INDEX { smonSuspectsIndex } ::= { smonSuspectsTable 1 }

SmonSuspectsEntry ::= SEQUENCE {
smonSuspectsIndex INTEGER,
smonSuspectID DisplayString,
smonSuspectsSourceHost IpAddress,
smonSuspectsTargetHost IpAddress,
smonSuspectsTargetProtocol INTEGER,
smonSuspectsTargetPort INTEGER,
smonSuspectsAlarmLevel INTEGER,
smonSuspectsIncidentsToday Counter32,
smonSuspectsIncidentsThisWeek Counter32,
smonSuspectsIncidentsThisMonth Counter32 }

smonSuspectsIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "Index identifying each item in this table." ::= { smonSuspectsEntry 1 }

smonSuspectsID OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-only

STATUS Mandatory
DESCRIPTION "The textual identification of a suspicious user being tracked." ::= { smonSuspectsEntry 2 }

smonSuspectSourceHost OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The source host address of the suspected subject (user, application, site) being tracked."
::= { smonSuspectsEntry 3 }

smonSuspectsTargetHost OBJECT-TYPE
SYNTAX IpAddress)
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The targeted host of the suspected subject being tracked. Multiple entries are possible."
::= { smonSuspectsEntry 4 }

smonSuspectsTargetProtocol OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The targeted protocol number (as defined by the Internet Assigned Numbers Authority) of the suspected subject being tracked. Multiple entries are possible." ::= { smonSuspectsEntry 5 }

smonSuspectsTargetPortNum OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The targeted port number of the suspected subject being tracked. Multiple entries are possible." ::= { smonSuspectsEntry 6 }

smonSuspectsAlarmLevel OBJECT-TYPE
SYNTAX INTEGER { Unknown (0), lowest (1).. highest (10) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The suspicion level of the suspected subject being tracked." ::= { smonSuspectsEntry 7 }

smonSuspectsIncidentsToday OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The number suspicious events in past 24 hours for this suspect being tracked."
::= { smonSuspectsEntry 8 }

smonSuspectsIncidentsThisWeek OBJECT-TYPE
SYNTAX Counter32
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The number suspicious events in past 7 days for this suspect being tracked."
::= { smonSuspectsEntry 9 }

smonSuspectsIncidentsThisMonth OBJECT-TYPE
SYNTAX Counter32

MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The number suspicious events in past month for this suspect being tracked."
::= { smonSuspectsEntry 10 }

-- ******** The Notifications Group **********

-- This group is required to be implemented only in SMON nodes that are management nodes.

smonNotificationsTable OBJECT-TYPE
SYNTAX SEQUENCE OF SmonNotificationsEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A list of the notifications received at this node." ::= { smonNotifications 1 }

smonNotificationsEntry OBJECT-TYPE
SYNTAX SmonNotificationsEntry
MAX-ACCESS not-accessible
STATUS Mandatory
INDEX { smonQueryIndex } ::= { smonNotificationsTable 1 }

SmonNotificationsEntry ::= SEQUENCE {
SmonNotificationIndex INTEGER,
SmonEventDate UTCTiime,
smonReportingNode Ipaddress,
smonOriginAuth DisplayString,
smonLevelOfTrust INTEGER,
smonEventType INTEGER,
smonEventSeverity INTEGER,
smonTargetID IpAddress,
smonTargetResouceID DisplayString,
smonAttackSource IpAddress,
smonAttackerID DisplayString }

smonNotificationIndex OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "Unique identifier for each notification entry." ::= { smonNotificationEntry 1 }

smonEventDate OBJECT-TYPE
SYNTAX UTCTime
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The date and time of occurrence for each notification entry."
::= { smonNotificationEntry 2 }

smonReportingNode OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Unique identifier for each notification entry." ::= { smonNotificationEntry 3 }

SmonOriginAuth OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The digital signature of the original notification creator." ::= { smonNotificationEntry 4 }

smonLevelOfTrust OBJECT-TYPE
SYNTAX INTEGER (0..10)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Indicator of trust level for each notification entry." ::= { smonNotificationEntry 5 }

smonEventType OBJECT-TYPE
SYNTAX INTEGER { Probe (1), Failed Login (2), Email Bomb (4), Virus (8), TCP SYN (16), General DOS (32), Address Spoofing (64), Replay (128), Masquerade (256) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Event category for each notification entry." ::= { smonNotificationEntry 6 }

smonEventSeverity OBJECT-TYPE
SYNTAX INTEGER { Informational (1), Warnings (2), Urgent (4), Emergency (8) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Unique identifier for each notification entry." ::= { smonNotificationEntry 7 }

smonTargetID OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The subject of the event for each notification entry, usually the same is the reporting host." ::= { smonNotificationEntry 8 }

smonTargetResourceID OBJECT-TYPE
SYNTAX DisplayString
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "identifier for the targeted resource associated with each notification entry."
::= { smonNotificationEntry 9 }

smonAttackSource OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Unique identifier for each notification entry." ::= { smonNotificationEntry 10 }

smonAttackerID OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory

```
DESCRIPTION  "Unique textual identifier for suspected attacker associated with a notification entry."
::= { smonNotificationEntry 11 }
```

-- ********** The SMIP Query (Event Data Request) Group **********

```
-- This group is  required to be implemented only in SMON nodes that are management nodes.  The
-- processing and storage requirements for SMIP queries assume more that a minimal device configuration.

smonQueryTable OBJECT-TYPE
SYNTAX  SEQUENCE OF  SmonQueryEntry
MAX-ACCESS  not-accessible
STATUS  Mandatory
DESCRIPTION  "A list of the active queries." ::= { smonEventDataRequest 1 }

smonQueryEntry OBJECT-TYPE
SYNTAX  SmonQueryEntry
MAX-ACCESS  not-accessible
STATUS  Mandatory
INDEX  { smonQueryIndex } ::= { smonQueryTable 1 }

SmonQueryEntry ::= SEQUENCE {
smonQueryIndex INTEGER,
smonQueryStartDate  UTCTime,
smonQueryEndTime  UTCTime,
smonQueryTargetID  DisplayString,
smonQueryTargetIDModifier  INTEGER,
smonQuerySourceID  DisplayString,
smonQuerySourceIDModifier  INTEGER,
smonQueryEventType  INTEGER,
smonQueryEventSeverity  INTEGER,
smonQueryUrgency INTEGER,
smonQueryMaxResponses INTEGER }

smonQueryIndex              OBJECT-TYPE
SYNTAX  INTEGER32  (1.. 2147483647)
MAX-ACCESS  not-accessible
STATUS  Mandatory
DESCRIPTION  "Unique identifier for each query entry." ::= { smonQueryEntry 1 }

smonQueryStartDate        OBJECT-TYPE
SYNTAX  UTCTime
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The earliest event records desired for this query entry.  An empty field means look at all
records from the beginning" ::= { smonQueryEntry 2 }

smonQueryEndTime smonQueryStartDate    OBJECT-TYPE
SYNTAX  UTCTime
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The latest date of event records to be matched for this query entry.  An empty field
means look at all records until the end" ::= { smonQueryEntry 3 }
```

smonQueryTargetID OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The IP address, DNS name, or subnet and mask of the the attacked system."
::= { smonQueryEntry 4 }

smonQueryTargetIDModifier OBJECT-TYPE
SYNTAX INTEGER { Like (1), Equal To (2), Not Equal To (3) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The search string modifier to be used with the smonQueryTargetID parameter above."
::= { smonQueryEntry 5 }

smonQuerySourceID OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The IP address, DNS name, or subnet and mask of the suspected attacker's system." ::= { smonQueryEntry 6 }

smonQuerySourceIDModifier OBJECT-TYPE
SYNTAX INTEGER { Like (1), Equal To (2), Not Equal To (3) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The search string modifier to be used with the smonQuerySourceID parameter above."
::= { smonQueryEntry 7 }

smonQueryEventType OBJECT-TYPE
SYNTAX INTEGER { Probe (1), Failed Login (2), Email Bomb (4), Virus (8), TCP SYN (16), General DOS (32), Address Spoofing (64), Replay (128), Masquerade (256) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The IP address, DNS name, or subnet and mask of the attacked system. Combinations are allowed by summing the power of two values for each desired match " ::= { smonQueryEntry 8 }

smonQueryEventSeverity OBJECT-TYPE
SYNTAX INTEGER { Informational (1), Warnings (2), Urgent (4), Emergency (8) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The level of the potential event damage to search for. Combinations are allowed by summing the powers of two value for each desired match." ::= { smonQueryEntry 9 }

smonQueryUrgency OBJECT-TYPE
SYNTAX INTEGER { Persistent (0), Low (1)..High (10) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The level of responsiveness desired in processing this search."
::= { smonQueryEntry 10 }

smonQueryMaxResponses INTEGER }
SYNTAX INTEGER { 1..65535}
MAX-ACCESS read-write

STATUS Mandatory
DESCRIPTION "The volume of responses to search for and return. This has an obvious effect on the relative responsiveness as requested above." ::= { smonQueryEntry 11 }

END

Appendix C

Firewall MIB

The Firewall MIB (FWMIB) is an application-specific extension of the Security MIB (SMIB) defined in **Appendix A**. It is designed to monitor and control firewall functions remotely through secure SNMPv3 transactions. This MIB definition considers work in progress by Grall and others [33] on a read-only Firewall MIB, which assumes only insecure SNMP interactions. Derivations from that work are clearly noted.

There can be significant variations in firewall capabilities, so all parameters except those in the fwSystem group are optional. If a group is implemented, the entire group must be supported. In addition, additional extensions for new and vendor-specific features are expected. The formal ASN.1 FWMIB definition begins on the following page.

```
 -- ************************************************
-- * Copyright 1999 Philip C. Hyland All Rights Reserved      *
-- *                                                          *
-- *                        Firewall MIB Descriptions         *
 -- ************************************************

FIREWALL-MIB DEFINITIONS ::= BEGIN

IMPORTS
enterprises, IpAddress, Counter, TimeTicks          FROM RFC1155-SMI
mib-2, DisplayString                                FROM RFC1213-MIB
TRAP-TYPE                                           FROM RFC-1215
MODULE-IDENTITY, OBJECT-TYPE, experimental,
Counter32, Integer32                                FROM SNMPv2-SMI
TEXTUAL-CONVENTION, TimeStamp                       FROM SNMPv2-TC
MODULE-COMPLIANCE, OBJECT-GROUP                     FROM SNMPv2-CONF;

-- Upon publication as RFC, delete this comment and the line following
-- this comment and change the reference of { fwmibDraft 1 }
-- (above) to { mib-2 X }.
-- This will result in changing:
-- 1 3 6 1 3 4400 2222 to:          [4400↔GMU and 2222↔fwMIB]
-- 1 3 6 1 2 1 fwMIB
-- This will make it easier to translate prototypes to the standard namespace
-- because the lengths of the OIDs won't change.

-- fwMIB MODULE-IDENTITY
--  LAST-UPDATED "9812150000Z"
-- ORGANIZATION "George Mason University"
--  CONTACT-INFO
--      "    Philip C. Hyland
--           Postal: George Mason University
--           4400 University Ave
--           Fairfax, VA 22060
--           Tel: 703-993-1499
--           Fax: 703-993-xxxx
--           E-mail: phyland@gmu.edu"
--
-- DESCRIPTION
--  "The MIB module for management of firewalls."

gmu      OBJECT IDENTIFIER ::= { experimental 4400 }
fwMIB  OBJECT IDENTIFIER ::= { gmu 2222 }
-- Complete fwMIB designation: ( 1 3 6 1 3 4400 2222 )

-- Textual conventions for this MIB module

DayGroup ::= TEXTUAL-CONVENTION
STATUS   current
DESCRIPTION "The set of days that may indicate the start or stop of a service."
SYNTAX INTEGER { Sunday (1), Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6),
Saturday (7), Weekdays (8), Weekends (9), Holidays (10) }
```

TimesofDay ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION "Hour groupings that may be used to start or stop security services."
SYNTAX INTEGER { invalid (0), All Hours (1), Bankers' Hours (e.g., 9-5) (2), Extended Business (e.g., 7AM-8PM) (4), Awake Hours (e.g., 6AM-11PM) (5) }

-- ********** Groups in Firewall MIB **********

fwSystem OBJECT IDENTIFIER ::= { fwMIB 1 }
fwPacketFilter OBJECT IDENTIFIER ::= { fwMIB 2 }
fwProxy OBJECT IDENTIFIER ::= { fwMIB 3 }
fwTraps OBJECT IDENTIFIER ::= { fwMIB 4 }

-- ********** The Firewall System Group **********

-- Global firewall configuration parameters and identity objects.

fwSysForwardingOBJECT-TYPE
SYNTAX INTEGER { active (1), inactive (2), active until Severe warning (3) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "This variable determines globally, whether the firewall can forward packets according to the defined packet filter and proxy policies. If a severe alarm occurs, the forwarding state may be changed to inactive, as a fail-safe mode." ::= { fwSystem 1 }

fwSysNonIpAction OBJECT-TYPE
SYNTAX INTEGER { forward (1), discard (2) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "Identifies what to do with packets received which are not IP packets (i.e. IPX, DECnet, etc.)." ::= { fwSystem 2 }

-- ********** The Packet Filter Group **********

pfTabUpdate OBJECT-TYPE
SYNTAX TimeStamp
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The last time and date of a successful table update." ::= { fwPacketFilter 1 }

pfTablePOC OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The name, contact data and public key of the person doing the latest table update."
::= { fwPacketFilter 2 }

pfTableSize OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The maximum number of entries allowed in the filter table."::= { fwPacketFilter 3 }

```
pfTable  OBJECT-TYPE
SYNTAX SEQUENCE OF PfEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A list of packet filter entries. The number of entries is given by the value of pfNumber."
::= { fwPacketFilter 4 }

pfEntry  OBJECT-TYPE
SYNTAX PfEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION  "A packet filter entry contains attributes that define acceptance/rejection of packets at a
particular interface."
INDEX { pfIndex } ::= { pfTable 1 }

PfEntry ::= SEQUENCE {
pfIndex                     INTEGER,
pfIn/Out_Flag               INTEGER,
pfAck_Flag                  INTEGER,
pfProtocol                  INTEGER,
pfAllow/Deny_Flag           INTEGER,
pfSource_Address            IpAddress,
pfSource_PortLB             INTEGER,
pfSource_PortUB             INTEGER,
pfSource_operator           OCTET STRING,
pfDest_Address              IpAddress,
pfDest_Port                 INTEGER,
pfStatus                    INTEGER,
pfActiveTimes1              TimesofDay,
pfActiveDays1               DayGroup,
pfRemoteDef                 INTEGER,
pfRespMsgFlag               INTEGER.
pfPollInterval              INTEGER.
PfDroppedPktsSinceLastPoll      Counter,
pfTop10SRCsSinceLastPoll        DisplayString ,
pfTop10DropSRCsSinceLastPoll    DisplayString }

PfIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..65355)
MAX-ACCESS  read-only
STATUS  Mandatory
DEFINITION  "A unique value for each filter rule. Its value ranges between 1 and 65535.  The value for
each interface must remain constant at least from one re-initialization of the entity's network management
system to the next re- initialization." ::= { pfEntry 1 }

pfIn/Out_Flag      OBJECT-TYPE
SYNTAX  INTEGER,
MAX-ACCESS  read-write
STATUS  Mandatory
DEFINITION  "The direction of packet flow (with respect to the router) on which to apply filter rules."
::= { pfEntry 2 }

pfAck_Flag         OBJECT-TYPE
```

SYNTAX INTEGER,
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "If the pfAckFlag is set, it indicates that the ack bit in TCP packet headers must be set."
::= { pfEntry 3 }

pfProtocol OBJECT-TYPE
SYNTAX INTEGER { TCP(1), UDP(2), ICMP(3) }
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Specifies the type of protocol to which the filtering rule applies." ::= { pfEntry 4 }

pfAllow/Deny_Flag OBJECT-TYPE
SYNTAX INTEGER (Allow (1), Deny (2) }
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Determines whether the present rule specification is permissive or exclusive."
::= { pfEntry 5 }

pfSourceAddress OBJECT-TYPE
SYNTAX IpAddress
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The source host identity to which the rule applies." ::= { pfEntry 6 }

PfSourcePortLB OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The source port lower bound to which the rule applies. The value zero has the special meaning the '<' SourceOperator should applied before the value in pfSourcePort_UB."
::= { pfEntry 7 }

pfSourcePortUB OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The source port upper bound to which the rule applies. The value of zero has the special meaning that the '>' operator should applied before the port value in pfSourcePortLB" ::= { pfEntry 8 }

pfDestAddress OBJECT-TYPE
SYNTAX IpAddress,
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The destination host identity to which the rule applies." ::= { pfEntry 9 }

pfDestPortLB OBJECT-TYPE

SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The destination port lower bound to which the rule applies. The value of zero has the special meaning that the '<' operator should applied before the port value in pfSourcePortUB."
::= { pfEntry 10 }

pfDestPortUB OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The destination port upper bound to which the rule applies. The value of zero has the special meaning that the '>' operator should applied before the port value in pfSourcePortLB."
::= { pfEntry 11 }

pfStatus OBJECT-TYPE
SYNTAX INTEGER { active (1), invalid (2), create-and-wait (3), create-and-go (4), destroy (5) }
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Indicates whether the rule is active, being created or to be deleted. Create-and-wait results in a new row instance, but indicates that data therein is unreliable. Create-and-go is a one step row addition." ::= { pfEntry 12 }

pfActiveTimes1 OBJECT-TYPE
SYNTAX TimesofDay
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The hours that the filter should be active if the pfStatus = 'active'." ::= { pfEntry 13 }

pfActiveDays1 OBJECT-TYPE
SYNTAX DayGroup
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The days that the filter should be active if the pfStatus = 'active'." ::= { pfEntry 14 }

pfdropped OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-only
STATUS Mandatory
DEFINITION "For deny rules only. Indicates how many packets matched this 'deny' rule and were thus dropped." ::= { pfEntry 15 }

pfRemoteDef OBJECT-TYPE
MAX-ACCESS read-only
STATUS Mandatory
DEFINITION "Indicates that a filter has been defined upstream, thus requiring consideration when new changes are made or if the rule results from an external update request." ::= { pfEntry 16 }

pfRespMsgFlag OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Indicates whether to send a response message to the source of a dropped packet."

::= { pfEntry 17 }

pfPollInterval OBJECT-TYPE
SYNTAX INTEGER (0..255)
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Indicates in minutes, the period between statistical record points." ::= { pfEntry 18 }

pfDroppedPktsSinceLastPoll OBJECT-TYPE
SYNTAX Counter
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Indicates whether to send a response message to the source of a dropped packet."
::= { pfEntry 19 }

pfTop10SRCsSinceLastPoll OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "A textual listing of the ten most active sources of packets since the last checkpoint."
::= { pfEntry 20 }

pfTop10DropSRCsSinceLastPoll OBJECT-TYPE
SYNTAX DisplayString (Size (0..255))
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "A textual listing of the ten most active sources of dropped packets since the last checkpoint." ::= { pfEntry 21 }

-- ******** The Proxy SessionsGroup **********************

-- Proxy Management Parameters

proxyMaxSessions OBJECT-TYPE
SYNTAX INTEGER
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The maximum number of entries in the proxySessionTable.” ::= { fwProxy 1 }

proxySessionTable OBJECT-TYPE
SYNTAX SEQUENCE OF ProxyEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DEFINITION "Each row entry represents an active proxy session being processed through the proxy server.” ::= { fwProxy 2 }

proxyEntry OBJECT-TYPE
SYNTAX ProxyEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A proxy entry contains attributes that define indicate the current status of a particular session."
INDEX { proxyIndex, proxySourceAddress } ::= { proxySessionTable 1 }

```
ProxyEntry ::= SEQUENCE {
proxyIndex              INTEGER,
proxySourceAddress      IpAddress,
proxySourceProtocol     INTEGER,
proxySourcePort         INTEGER,
proxyDestAddress        IpAddress,
proxyDestProtocol       INTEGER,
proxyDestPort           INTEGER,
proxySessionStatus      INTEGER,
proxyType               INTEGER,
proxyStartTime          TimeTicks,
proxyDomainName         OCTET STRING,
proxyMonitoring         INTEGER }

proxyIndex      OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-only
STATUS  Mandatory
DEFINITION  "A unique value for referencing each proxy session.  The value for each interface must
remain constant at least from one re-initialization of the entity's management system to the next re-
initialization." ::= { proxyEntry 1 }

proxySourceAddress      OBJECT-TYPE
SYNTAX  IpAddress
MAX-ACCESS  read-only
STATUS  Mandatory
DEFINITION  " The source host identity from which the proxy service originates."  ::= { proxyEntry 2 }

proxySourceProtocol     OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS  Mandatory
DEFINITION  "The source protocol number (as defined by the Internet Assigned Number Authority) from
which the proxy session originates."  ::= { proxyEntry 3 }

proxySourcePort OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS  Mandatory
DEFINITION  "The source port number from which the proxy session originates."  ::= { proxyEntry 4 }

proxyDestAddress        OBJECT-TYPE
SYNTAX  IpAddress
MAX-ACCESS  read-only
STATUS  Mandatory
DEFINITION  " The destination host to which the proxy service connects."  ::= { proxyEntry 5 }

proxyDestProtocol       OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS  Mandatory
```

DEFINITION "The destination protocol number (as defined by the Internet Assigned Number Authority) to which the proxy session connects." ::= { proxyEntry 6 }

proxyDestPort OBJECT-TYPE
SYNTAX INTEGER (1..65535)
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "The destination port number to which the proxy session connects." ::= { proxyEntry 7 }

proxySessionStatus OBJECT-TYPE
SYNTAX INTEGER {opening (1), active (2), invalid (3) }
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Indicates the processing state of a particular session." ::= { proxyEntry 8 }

proxyType OBJECT-TYPE
SYNTAX INTEGER { SMTP proxy (1), TCP proxy (2), TELNET proxy (3), HTTP proxy (4), FTP proxy (5), rlogin proxy (6), WAIS proxy (7), X11 proxy (8), snmpv1 proxy (9), SNMPv2 proxy (10), NNTP proxy (11), plug-gw proxy (12), IRC proxy (13), tcp_wrapper (14) }
MAX-ACCESS read-write
STATUS Mandatory
DEFINITION "Specifies the type of proxy service of the current session." ::= { proxyEntry 9 }

proxyStartTime OBJECT-TYPE
SYNTAX TimeStamp
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The time that the current session began (allows session duration to be derived by subtracting the proxy start time from the current time." ::= { proxyEntry 10 }

proxyDomainName OBJECT-TYPE
SYNTAX OCTET STRING
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The domain name of the external entity associated with the current session."
::= { proxyEntry 11 }

proxyMonitoring OBJECT-TYPE
SYNTAX INTEGER { None (0), Connection Logging (1), Header Logging (2), Full Packet Logging (3) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "The level of monitoring associated with the current session." ::= { proxyEntry 12 }

-- ******** The FW Trap Group **********

fwSecurityAlarm TRAP-TYPE
ENTERPRISE fwMIB
VARIABLES { fwAlarmLevel }
DESCRIPTION "This trap is sent when a notable security event is detected. The varbind field in the trap identifies the warning level." ::= 49

END

Appendix D

Security Guard MIB

The security guard MIB expects implementation of MIB-II and the core Security MIB to provide certain administrative and operational performance information. Without MIB-II, remote management operations are still possible, but SMIB is mandatory.

```
 -- ************************************************
-- * Copyright 1999 Philip C. Hyland All Rights Reserved      *
-- *                                                          *
-- *                Security Guard MIB Descriptions           *
 -- ************************************************

SECURITY-GUARD-MIB DEFINITIONS ::= BEGIN

IMPORTS
mib-2, DisplayString                                    FROM RFC1213-MIB
-- RFC 1902
MODULE-IDENTITY, OBJECT-TYPE, NOTIFICATION-TYPE, IpAddress, enterprises, experimental,
Counter32, Integer32                                    FROM SNMPv2-SMI
-- RFC 1903
TEXTUAL-CONVENTION, TimeStamp                           FROM SNMPv2-TC
-- RFC 1904
MODULE-COMPLIANCE, OBJECT-GROUP                         FROM SNMPv2-CONF;

-- Upon publication as RFC, delete this comment and the line following
-- this comment and change the reference of { sgmibDraft 1 }
-- (above) to { mib-2 X }.
-- This will result in changing:
-- 1 3 6 1 3 4400 2223 to:          [4400↔GMU and 2223↔sgMIB]
-- 1 3 6 1 2 1 sgMIB
-- This will make it easier to translate prototypes to the standard namespace
-- because the lengths of the OIDs won't change.

sgMIB MODULE-IDENTITY
LAST-UPDATED "9812150000Z"
ORGANIZATION "George Mason University"
CONTACT-INFO
```

" Philip C. Hyland
Postal: George Mason University
4400 University Ave
Fairfax, VA 22060
Tel: 703-993-1499
Fax: 703-993-xxxx
E-mail: phyland@gmu.edu"
DESCRIPTION "The security guard MIB module for management of security guard applications."
::= { experimental gmu sgmib }

gmu OBJECT IDENTIFIER ::= { experimental 4400 }
sgMIB OBJECT IDENTIFIER ::= { gmu 2223 }
-- Complete sgMIB designation: (1 3 6 1 3 4400 2223)
-- iso.org.dod.internet.experimental.gmu.sgMIB

-- Textual Conventions
DayGroup ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION "The set of days that may indicate the start or stop of a service."
SYNTAX INTEGER { Sunday (1), Monday (2), Tuesday (3), Wednesday (4), Thursday (5), Friday (6), Saturday (7), Weekdays (8), Weekends (9), Holidays (10) }

TimesofDay ::= TEXTUAL-CONVENTION
STATUS current
DESCRIPTION "Hour groupings that may be used to start or stop security services."
SYNTAX INTEGER { invalid (0), All Hours (1), Bankers' Hours (e.g., 9-5) (2), Extended Business (e.g., 7AM-8PM) (4), Awake Hours (e.g., 6AM-11PM) (5) }

sgAdmin	OBJECT IDENTIFIER	{ sgmib 1 }
sgOpsCon	OBJECT IDENTIFIER	{ sgmib 2 }
sgTrap	OBJECT IDENTIFIER	{ sgmib 3 }

-- ********** The Administrative Control Group **********

sgReleaserTable OBJECT-TYPE
SYNTAX SEQUENCE OF SgReleaserEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION ".List of authorized persons who may verify the data packages are releasible "
::= { sgAdmin 1 }

sgReleaserEntry OBJECT-TYPE
SYNTAX SgReleaserEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION "A releaser entry contains attributes that identify who, when, where releases are allowed."
INDEX { sgReleaserIndex } ::= { sgReleaserTable 1 }

SgReleaserEntry ::= SEQUENCE {

```
sgReleaserIndex  INTEGER,
sgReleaserName  DisplayString,
sgReleaserKeyID DisplayString,
sgReleaseDays    DayGroup,
sgReleaseTime    TimesofDay,
sgReleaseDataTypes        INTEGER }

sgReleaserIndex  OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "Unique identification of the row/record of a person having release authority."
::= { sgReleaserEntry 1 }

sgReleaserName  OBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The name of the person having release authority." ::= { sgReleaserEntry 2 }

sgReleaserKeyID OBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The authentication key of the person having release authority." ::= { sgReleaserEntry 3 }

sgReleaserDays   OBJECT-TYPE
SYNTAX  DayGroup
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The days that the person having release authority can exercise that authority
::= { sgReleaserEntry 4 }

sgReleaserTime   OBJECT-TYPE
SYNTAX  TimesofDay
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The times of day that the person having release authority can exercise that authority."
::= { sgReleaserEntry 5 }

sgReleaserDataTypes        OBJECT-TYPE
SYNTAX  INTEGER { TBD-*match with MIME types or application data extensions* }
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The data types this person has authority to release.  Value is the bitwise sum of all
allowed types or a text listing of all allowed extension types. " ::= { sgReleaserEntry 6 }

sgFilterTable       OBJECT-TYPE
SYNTAX SEQUENCE OF SgFilterEntry
MAX-ACCESS  not-accessible
STATUS  mandatory
DESCRIPTION " This table lists the names, formats, and values for formatted data fields under review that
are non-releasable.  A modifier field helps define value ranges." ::= { sgAdmin 2 }
```

```
sgFilterEntry       OBJECT-TYPE
SYNTAX  SgFilterEntry
MAX-ACCESS not-accessible
STATUS Mandatory
DESCRIPTION  "A guard filter entry contains attributes that define acceptance/rejection of packets at the
interface."
INDEX { sgFilterIndex } ::= { sgFilterTable 1 }

SgFilterEntry ::= SEQUENCE {
sgFilterIndex       INTEGER,
sgFieldName         DisplayString,
sgFieldType         INTEGER,
sgFieldValue        INTEGER,
sgFieldModifier     INTEGER }

sgFilterIndex       OBJECT-TYPE
SYNTAX  INTEGER (1..65535)
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "Unique identification of the sgFilterEntry row." ::= { sgFilterEntry 1 }

sgFieldName       OBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The name of the data field being filtered.  May be empty for unformatted data"
::= { sgFilterEntry 2 }

sgFieldType       OBJECT-TYPE
SYNTAX INTEGER { invalid (0), int (1), alphanumeric (2), binary (3) }
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The data type being evaluated for release." ::= { sgFilterEntry 3 }

sgFieldValue       OBJECT-TYPE
SYNTAX  DisplayString (Size (0..255))
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The value of the data field being filtered.  Data is interpreted and compared based on the
data type identified above." ::= { sgFilterEntry 4 }

sgFieldModifier   OBJECT-TYPE
SYNTAX INTEGER { invalid (0), Greater Than (1), Less Than (2), Equal To (3), Not Equal (4), Contains
(5), Does Not Contain (6) }
MAX-ACCESS  read-write
STATUS  Mandatory
DESCRIPTION  "The data modifier to identify ranges/matches for data being evaluated for release."
::= { sgFilterEntry 5 }
```

-- ********** The Operational Control Group **********

```
sgReleaseEnable  OBJECT-TYPE
```

SYNTAX INTEGER { on (1) or off (0) }
MAX-ACCESS read-write
STATUS Mandatory
DESCRIPTION "This switch sets the release mechanism on or off. " ::= { sgOpsCon 1 }

sgQuarantineSpace OBJECT-TYPE
SYNTAX Gauge
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "This shows the percentage of space filled within the quarantine area. When it is full, no further reviews are permitted. The rate at which this space is filled is also an indicator of internal QC status" ::= { sgOpsCon 2 }

sgInputSpace OBJECT-TYPE
SYNTAX Gauge
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "This shows the percentage of space filled within the input area. When it is full, no further data can be accepted for review. The rate at which this space is filled is also an indicator of device status and performance" ::= { sgOpsCon 3 }

sgKeepAliveInterval OBJECT-TYPE
SYNTAX INTEGER (1..60)
MAX-ACCESS read-only
STATUS Mandatory
DESCRIPTION "The number of seconds between keepalive checks between the SG agent and the management station. Just before the interval expires, the agent sends a keepalive Trap to the manager. If a matching response is not received before the time expires, no more data can be released from the SG until normal communications/operations are verified." ::= { sgOpsCon 4 }

-- ******** The SG Trap Group **********************

sgQspace@80 NOTIFICATION_TYPE
OBJECTS {
sgQuarantineSpace
sgEventTime,
sgReportingHost,
STATUS current
DESCRIPTION "Threshold event notification of low quarantine space from SG application."
::= { sgTrap 1 }

sgInput@80 NOTIFICATION_TYPE
OBJECTS {
sgInputSpace
sgEventTime,
sgReportingHost,
STATUS current
DESCRIPTION "Threshold event notification of low nput space from SG application." ::= { sgTrap 2 }

sgKeepAlive NOTIFICATION_TYPE
OBJECTS {
sgEventTime,
sgEventType

sgReportingHost,
STATUS current
DESCRIPTION "An acknowledged keepalive event notification from SG application to the manager. A reply is required before further SG data can be released." ::= { sgTrap 3 }

END

List of References

List of References

[1] ANSI X9.17: American National Standard for Financial Institution Message Authentication (Wholesale), 1985.

[2] Ballardie, Anthony J., “A New Approach to Multicast Communication in a Datagram Internet”, Ph.D. Dissertation, University of London, May 1995.

[3] Banning, Debra Lynn, “A Distributed Audit System Using Network Management Protocols”, Master’s Thesis, California State University, Long Beach, 1992.

[4] Bartal, Yair, Mayer, Alain, Nissim, Kobbi, and Wool, Avishai, "Firmato: A Novel Firewall Management Toolkit", *IEEE Symposium on Security and Privacy'99*, Oakland, CA, May 1999.

[5] Bierman, A. and Iddon, R., "Remote Network Monitoring MIB Protocol Identifiers", Request for Comments: 2074, January 1997.

[6] Blumenthal, U. and Wijnen, B., User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3)", Request for Comments: 2274, January 1998.

[7] Bruce, Glen and Dempsey, Rob, *Security in Distributed Computing: Did You Lock the Door?*, Prentice-Hall, 1997.

[8] CAE Specification, "Common Security: CDSA and CSSM", Document Number C707, ISBN 1-85912-194-2, The Open Group, 1997.

[9] Case, J., Fedor, M., Schoffstall, M., and Davin, J., "A Simple Network Management Protocol (SNMP)", Request for Comments: 1157, May 1990.

[10] Case, J., Harrington, D., Presuhn, R., and Wijnen, B., "Message Processing and Dispatching for the Simple Network Management Protocol (SNMP)", Request for Comments: 2272,January 1998.

[11] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., " Coexistence between Version 1 and Version 2 of the Internet-standard Network Management Framework", Request for Comments: 1908, January 1996.

[12] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., " Introduction to Community-based SNMPv2", Request for Comments: 1901, January 1996.

[13] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., " Management Information Base for Version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1907, January 1996.

[14] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., " Manager-to-Manager Management Information Base", Request for Comments: 1451, April 1993.

[15] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., " Structure of Management Information for Version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1902, January 1996.

[16] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Coexistence between version 1 and version 2 of the Internet-standard Network Management Framework", Request for Comments: 1452, April 1993.

[17] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Conformance Statements for Version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1904, January 1996.

[18] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Management Information Base for version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1450, April 1993.

[19] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Protocol Operations for version 2 of Simple Network Management Protocol (SNMPv2)", Request for Comments: 1448, April 1993.

[20] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Protocol Operations for Version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1905, January 1996.

[21] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Textual Conventions for SNMPv2", Request for Comments: 1903, January 1996.

[22] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1449, April 1993.

[23] Case, J., McCloghrie, K., Rose, M., and Waldbusser, S., "Transport Mappings for Version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1906, January 1996.

[24] Condell, M., Lynn, C. and Zao, J., " Security Policy Specification Language", Internet Draft <draft-ietf-ipsec-spsl-00.txt>, work in progress, October 21, 1998.

[25] Crosbie, Mark, “Active Defense of a Computer System Using Autonomous Agents”, Purdue Univ., http://www.purdue.cs.edu/homes/spaf/tech-reps/9508.ps.

[26] Dierks, T. and Allen, C., " The TLS Protocol Version 1.0", Internet-Draft <draft-ietf-tls-protocol-06.txt>, work in progress, November 12, 1998.

[27] Diffie, W. and Hellman, M., "New Directions in Cryptography", *IEEE Transactions on Information Theory*, vol. IT-22, no. 6 (1976), pp. 644-654.

[28] Elkins, M., "MIME Security with Pretty Good Privacy (PGP)", Request for Comments: 2015, October 1996.

[29] Endersz, G, et al, "Key Management and the Security Management in Open Systems: The SAMSON Prototype", *Proceedings of the IFIP SEC '95: 11th International Conference on Information Security*, May 1995.

[30] Escamilla, Terry, *Intrusion Detection: Network Security Beyond the Firewall*, Wiley and Son, 1998.

[31] FIPS 171: U.S. Department of Commerce, *Key Management Using ANSI X9.17*, Federal Information Processing Standards Publication 171, 1992.

[32] FIPS 46-1: U.S. Department of Commerce, *Data Encryption Standard*, Federal Information Processing Standards Publication 46-1, 1988.

[33] Frier, A., Karlton, P. and Kocher, P. "The SSL 3.0 Protocol", Netscape Communications Corp., Nov 18, 1996.

[34] Frincke, Deborah, et al, "A Framework for Cooperative Intrusion Detection", *21st National Information System Security Conference Proceedings*, October 1998.

[35] Galvin, James and McCloghrie, Keith, "Administrative Model for version 2 of the Simple Network Management Protocol (SNMPv2), Request for Comments: 1445, April 1993.

[36] Galvin, James and McCloghrie, Keith, "Security Protocols for version 2 of the Simple Network Management Protocol (SNMPv2)", Request for Comments: 1446, April 1993.

[37] Gong, Li and Shocham, Mencham, "Elements of Trusted Multicasting", *Proceedings 1994 International Conference on Network Protocols*, p. 23-30, IEEE Computer Society, Los Alamitos, CA, 1994.

[38] Grall, C., "Firewall Management Information Base", Internet-Draft <draft-grall-firewall-mib-01.txt>, work in progress, 20 April 1998.

[39] Harkins, D. and Carrel, D., "The Internet Key Exchange (IKE)", Internet-Draft <draft-ietf-ipsec-isakmp-oakley-08.txt>, work in progress, June 1998.

[40] Harreld, Heather, "'Cyber-radar' will alert feds to CPU break-ins", *Federal Computer Week*, p. 22, September 22, 1997.

[41] Harrington, D., Presuhn, R., and Wijnen, B., "An Architecture for Describing SNMP Management Frameworks", Request for Comments: 2271, January 1998.

[42] Hochmuth, Phil, "Firewall market blazing, up 143%", *Network World*, August 24, 1998.

[43] Hyland, Philip and Sandhu, Ravi, "Concentric Supervision of Security Applications: A New Security Management Paradigm", *Annual Computer Security Applications Conference*, December 1998.

[44] Hyland, Philip and Sandhu, Ravi, "Management of Network Security Applications", *21st National Information System Security Conference Proceedings*, October 1998.

[45] ISO/IEC 7498-2, Information Technology - Open Systems Interconnection - Basic Reference Model - Part 2: Security Architecture, International Standards Organization, 1988.

[46] ISO/IEC 7498-4, Information Technology - Open Systems Interconnection - Basic Reference Model - Part 4: Management Framework, International Standards Organization, 1989.

[47] ISO/IEC 9596, Information Technology – Common Management Information Protocol Specification, International Standards Organization, 1989.

[48] ISO/IEC 11586-1, Information Technology - Open Systems Interconnection - Generic Upper Layers Security Overview, Models and Notation, International Standards Organization, 1989.

[49] Izumi, Yutaka, Nakai, Tomoya, Yamaguchi, Suguru, and Oie, Yuji, "Design and Implementation of an Agent System for Application Service Management", URL: http://130.75.2.13/inet98_proc/1e/1e_3.htm

[50] LaBarre, Lee, ISO/CCITT to Internet Management Coexistence (IIMC): ISO/CCITT to Internet Management Security (IIMCSEC), Internet Draft, MITRE, Feb 1994.

[51] Lechner, S., "SAMSON: Management of Security in Open Systems", *Computer Communications*, Sep 1994.

[52] Levi, D., Meyer, P., and Stewart, B., "SNMPv3 Applications ", Request for Comments: 2273, January 1998.

[53] Linn, J., "Generic Security Service Application Program Interface, Version 2", Request for Comments: 2078, January 1997.

[54] Lunt, T., et al, A Real-time Intrusion Detection Expert System (IDES) - Final Report, SRI International, Menlo Park, CA, Feb. 1992.

[55] Maughan, D., Schertler, M., Schneider, M., Turner, J., "Internet Security Association and Key Management Protocol", Internet-Draft <draft-ietf-ipsec-isakmp-10.txt>, work in progress, July 3, 1998.

[56] McCloghrie, Keith and Galvin, James, "Party MIB for version 2 of the Simple Network Management Protocol (SNMPv2))", Request for Comments: 1447, April 1993.

[57] Messner, Ellen, "Burned by firewalls", *Network World*, September 21, 1998.

[58] Muftic, Sean, et al, *Security Architecture for Open Distributed Systems*, Wiley and Sons, New York 1993.

[59] Nelson, Matthew, "Two faces for the firewall", *InfoWorld*, Vol. 20, Issue 41, 12 October 1998.

[60] Rose, M. and McCloghrie, K., "Structure and Identification of Management Information for TCP/IP-based Internets", RFC: 1155, May 1990.

[61] Rose, Marshall T., *The Simple Book: An Introduction to Management of TCP/IP-based Internets*, Prentice Hall Series in Innovative Technology, Prentice-Hall, 1991.

[62] Rushby, John, "Critical System Properties: Survey and Taxonomy", *Reliability Engineering and System Safety*, 43(2): 189-219, 1994.

[63] Rushby, John, "Kernels for Safety?", *Safe and Secure Computing Systems*, pp. 210-220, Blackwell Scientific Publications, 1989.

[64] SAMSON URL: http://www.darmstadt.gmd.de/TKT/security/samson/samson.html

[65] SAMSON GDMO SMIB URL: http://www.darmstadt.gmd.de/TKT/security/samson/results_from_CD/GDMO/gdmo.html

[66] Sandhu, R., Coyne, E., Feinstein, H., & Youman, C., " Role-Based Access Control Models", *IEEE Computer*, Volume 29, Number 2, February 1996.

[67] Schiffman, A., and Rescorla, E., "The Secure HyperText Transfer Protocol", Internet-Draft <draft-ietf-wts-shttp-06.txt>, work in progress, 6 June 1998.

[68] Stallings, William, *SNMP, SNMPv2 and CMIP: the Practical Guide to Network Management Standards*, Addison-Wesley, 1993.

[69] Systems Management: Management Protocols API (XMP), ISBN 1-85912-027-X, The Open Group, 1994.

[70] TRUMPET URL: http://ascom.cica.fr/TRUMPET/

[71] Waldbusser, S., Remote Network Monitoring Management Information Base, Request for Comments: 1757, February 1995.

[72] Waldbusser, S., Remote Network Monitoring Management Information Base Version 2 using SMIv2, Request for Comments: 2021, January 1997.

[73] White, Gregory B., Fisch, Eric A., and Pooch, Udo W., “Cooperating Security Managers: A Peer-Based Intrusion Detection System”, *IEEE Network*, pp. 20-23, January/February 1996.

[74] Wijnen, B., Presuhn, R., and McCloghrie, Keith, "View-based Access Control Model (VACM) for the Simple Network Management Protocol (SNMP)", Request for Comments: 2275, January 1998.

[75] WILMA URL: http://ftp.ldv.e-technik.tu-muenchen.de/dist/WILMA/WHAT_IS_WILMA.html

CURRICULUM VITAE

Philip C. Hyland was born on January 29, 1956, in Rochester, New York, and is a citizen of the United States. He graduated from Wayne Central High School, Ontario Center, New York, in 1974. He received his Bachelor of Science in Applied Science and Engineering in 1978 from the United States Military Academy, West Point, New York and served as a Signal Corps officer in the United States Army for over 11 years. He is a Vietnam-era and Persian Gulf veteran and has earned numerous awards for meritorious service. He is currently a lieutenant colonel in the U.S. Army Reserves. He earned a Master of Science in Computer and Electronic Engineering from George Mason University in 1987 and has been employed at TASC, Inc. as a Communications System and Network Engineer since 1989. He has published two papers entitled "Management of Network Security Applications" and "Concentric Supervision of Security Applications: A New Security Management Paradigm.