NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS 19980414 141

A MANAGEMENT SYSTEM FOR HETEROGENEOUS NETWORKS (MSHN) SECURITY ANALYSIS

by

John Paul English

September, 1997

Thesis Advisor:	Cynthia E. Irvine
Thesis Co-Advisor:	Taylor Kidd

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE September 1997	3. REPORT TYPE AND DATES COVERED Master's Thesis

4. TITLE AND SUBTITLE A Management System for Heterogeneous Networks (MSHN) Security Analysis	5. FUNDING NUMBERS
6. AUTHOR(S) English, John Paul	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution unlimited.	12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

A team of interdisciplinary experts funded by DARPA is in the process of developing a Resource Management System termed MSHN (a Management System for Heterogeneous Networks). MSHN's primary function is to accept a sequence of jobs, and intelligently determine what jobs should be executed on which machines and when. It is designed to take both machine affinity and loads into account, thus providing superior performance and Quality of Service (QoS). The current prototype of MSHN does not provide protection against the threats of inadvertent disclosure and corruption of sensitive information and resources. A rigorous security analysis of MSHN is the first step required to successfully incorporate security into the MSHN project. The approach taken was to analyze MSHN's architecture, information flow diagrams and user interfaces and explain how fundamental security concepts may be applied to MSHN. By exercising the MSHN simulator, this work was able to expose many security weaknesses and outline conceivable methods of exploitation. As a result of this effort, a security policy tailored to MSHN is proposed, a functional breakout process based on the principle of least privilege between common user interface capabilities and administration capabilities is provided, and finally design recommendations for the incorporation of security into MSHN are presented

14. SUBJECT TERMS MSHN, Security Analysis, Quality of Service, Security Policy, Interface Assessment	15. NUMBER OF PAGES 108
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFI- CATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)

Prescribed by ANSI Std. 239-18

DTIC QUALITY INSPECTED 3

A MANAGEMENT SYSTEM FOR HETEROGENEOUS NETWORKS (MSHN) SECURITY ANALYSIS

John P. English
Lieutenant, United States Navy Reserve
B.S., Massachusetts Maritime Academy, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
September 1997

Author: [illegible]
John P. English

Approved by: Cynthia E. Irvine
Cynthia E. Irvine, Thesis Advisor

[illegible]
Taylor Kidd, Thesis Co-Advisor

[illegible]
Ted Lewis, Chair, Department of Computer Science

ABSTRACT

A team of interdisciplinary experts funded by DARPA is in the process of developing a Resource Management System termed MSHN (a Management System for Heterogeneous Networks). MSHN's primary function is to accept a sequence of jobs, and intelligently determine what jobs should be executed on which machines and when. It is designed to take both machine affinity and loads into account, thus providing superior performance and Quality of Service (QoS). The current prototype of MSHN does not provide protection against the threats of inadvertent disclosure and corruption of sensitive information and resources. A rigorous security analysis of MSHN is the first step required to successfully incorporate security into the MSHN project.

The approach taken was to analyze MSHN's architecture, information flow diagrams and user interfaces and explain how fundamental security concepts may be applied to MSHN. By exercising the MSHN simulator, this work was able to expose many security weaknesses and outline conceivable methods of exploitation.

As a result of this effort, a security policy tailored to MSHN is proposed, a functional breakout process based on the principle of least privilege between common user interface capabilities and administration capabilities is provided, and finally design recommendations for the incorporation of security into MSHN are presented.

LIST OF FIGURES

LIST OF SYMBOLS, ACRONYMS, AND ABBREVIATIONS

ACM	Access Control Matrix
CTCPEC	Canadian Trusted Computer Product Evaluation Criteria
DAC	Discretionary Access Control
DARPA	Defense Advanced Research Projects Agency
DoD	Department of Defense
EOT	Editor Object Types
GUI	Graphical User Interface
I&A	Identification and Authentication
ITSEC	Information Technology Security Evaluation Criteria
MAC	Mandatory Access Control
MSHN	Management System for Heterogeneous Networks
MSP	Multipolicy Security Policy
NCAR	National Center for Atmospheric Research
NCCOSC	Naval Command and Control, Ocean Surveillance Center
NCSC	National Computer Security Center
NIH	National Institutes of Health
NRaD	Naval Research and Development
QoS	Quality of Service
RMS	Resource Management Systems
RVM	Reference Validation Mechanism
SSF	Security Supporting Functions
TCB	Trusted Computing Base
TCSEC	Trusted Computer System Evaluation Criteria
VHM	Virtual Heterogeneous Machine

ACKNOWLEDGEMENT

I would like to gratefully acknowledge the guidance, motivation, and wisdom provided by Cynthia Irvine, my thesis advisor. I would also like to extend my gratitude to my co-advisor, Taylor Kidd, for his valuable assistance and determination in helping me complete this project. Lastly I would like to recognize the loving patience, sacrifice, and constant devotion of my wife Lynn and my daughter Maggie. Without their support, this thesis and my studies could not have been completed.

I. INTRODUCTION

A. PURPOSE

As computing resources diversify and data becomes distributed throughout the nation and the world, a growing common need exists: the ability to intelligently manage a distributed, heterogeneous computational network and its corresponding resources. A team of interdisciplinary experts funded by DARPA (Defense Advanced Research Projects Agency) is in the process of developing a scheduling framework termed MSHN[1] (a Management System for Heterogeneous Networks). MSHN's primary function is to accept a sequence of jobs, and determine what jobs should be executed on which machines and when. MSHN will incorporate innovative approaches to scheduling and apply advanced job monitoring capabilities to achieve superior performance and meet Quality of Service (QoS) requirements. This distinguishes it from the more traditional Resource Management Systems (RMSs).

One of the many issues being addressed as part of the design and prototyping of MSHN is security. This thesis provides a first step in understanding how security can be incorporated into MSHN. Fundamental security concepts applicable to MSHN are presented. Security policies in the context of heterogeneous systems are discussed. This work examines security vulnerabilities in the user interface of MSHN's predecessor, and concludes with a discussion of the security weaknesses of MSHN's architecture.

[1] MSHN is pronounced "Mission"

The results of this research are twofold. First, we furnish sufficient background on MSHN and how security objectives, principles, and policies may be applied. Also, we provide guidance as to where future studies should focus in order to permit MSHN to balance the enforcement of the security policy against satisfying other QoS obligations.

B. BACKGROUND

MSHN is a program that is building upon the experiences of the SmartNet scheduling framework. SmartNet's genesis was from a paper written in 1991 by Richard Freund, entitled *"SuperC or Distributed Heterogeneous HPC"* [Ref. 1]. This paper viewed the scheduling of multiple and independent compute intensive tasks as a linear programming problem. Later, Freund continued his initial work by forming a design and research team at the Naval Command and Control, Ocean Surveillance Center, Research, Development, Testing and Evaluation Division (NRaD), leading to the creation of SmartNet in 1993. SmartNet became operational in early 1994. Its team has consisted at times of upwards of 25 members, some doing research, others development, and still others product support.

SmartNet has and is being used by many government agencies, including NIH (National Institutes of Health), DARPA, NCAR (National Center for Atmospheric Research), and the US Navy. Presently, SmartNet is known as a scheduling framework for managing tasks in heterogeneous environments. Initially designed for coordinating computationally bound HPC tasks, it has been expanded and generalized to operate in the more typical distributed environment generally in use today.

C. SCOPE, LIMITATIONS, AND ASSUMPTIONS

The security architecture, like the overall architecture of MSHN, is currently under development. The foundation for this thesis is based on the designers' vision for MSHN's architecture, capabilities, and usage.

D. ORGANIZATION OF THESIS

This section provides an overview of each chapter's contents and purpose.

1. Introduction

Chapter I discusses the purpose, background, and scope of this thesis.

2. MSHN Profile

Chapter II describes MSHN's objectives. MSHN's purpose, components, and configurations are explained in detail. The innovations and unique characteristics that MSHN embodies, through its predecessor SmartNet, are presented. The chapter concludes by outlining the configuration of MSHN that will be used for study throughout this thesis.

3. Overview of Computer Security

Chapter III provides a brief review of computer security as it is applicable to the analysis of MSHN. Fundamental security terms, concepts, and goals are introduced. The chapter addresses such topics as security objectives, security models, and security functions and mechanisms. The chapter will end with a discussion of the Trusted Computer System Evaluation Criteria, which provides a basis for gauging the confidence with which a security policy is correctly enforced by commercial products. This chapter will assist the reader in gaining a rudimentary understanding of computer security and build a basis for the subsequent security analysis of MSHN.

4. Quality of Service

Chapter IV explores the notion of Quality of Service (QoS). Its definition and quantification are examined. We explain how MSHN may support QoS requirements for applications taking advantage of MSHN's resource management services. How security can be viewed as a service, how security requirements affect other services, and the implications of having multiple security policies will be discussed.

5. Security Policy and Interpretation for MSHN

Chapter V discusses security policies that could be adopted by MSHN. Statements of intent with regard to the control of access to information and its dissemination are declared and expounded upon. This chapter is an essential step in transforming MSHN into a secure system.

6. Interface Analysis

Chapter VI analyzes the user interface of MSHN's predecessor, SmartNet, from a security perspective. Vulnerabilities that may lead to the denial of service, the corruption of data and applications, and the unintended disclosure of sensitive information are exposed. This chapter provides guidance for the development of MSHN's interface such that security relevant interface functions will be clearly separated from those other functions that are security neutral.

7. Conclusions

Chapter VII provides a summary of this research and gives recommendations for the direction of future research toward the incorporation of security into MSHN.

II. MSHN PROFILE

This chapter provides an overview of the Management System for Heterogeneous Networks (MSHN). Although MSHN is still in the design phase of its development, it's objective, requirements and basic architecture are well founded. For the ease of writing this portrait of MSHN, we will use the present tense. The majority of this chapter is based on several sources. These include the documentation for SmartNet [Refs. 2, and 3], the unpublished notes of one of the designers [Ref. 4], and discussions held at a MSHN Investigator Meeting [Ref. 5].

A. SETTING

1. Purpose

MSHN is a scheduling framework for managing jobs and resources in a heterogeneous computational environment. Given a set of jobs, MSHN determines where and when each job should execute in order to run a set of jobs while maximizing some performance criteria, such as executing them in the smallest possible amount of time. It achieves its superior performance through its comprehensive view of the virtual heterogeneous machine (VHM) and its intimate knowledge of the jobs being scheduled. "The VHM is the set of machines and resources MSHN is installed to operate with." [Ref. 4] MSHN's view of the VHM encompasses not only resource loads and job progress, but resource capability and affinity as well. MSHN is able to use this view to determine and implement the best schedule satisfying the requirements of the jobs it executes. MSHN possesses user interfaces enabling the performance power of the heterogeneous environment to be harnessed.

2. Scheduling Framework

MSHN is neither a scheduler nor a Resource Management System (RMS) but a more robust composition called a scheduling framework. The scheduler is a limited component of this greater system. Its only function is to decide where to run each job. It depends on other mechanisms to gather and provide the necessary information, and to implement the schedules it generates. An RMS incorporates a basic scheduler and, in addition, the ability to execute jobs and monitor their progress. It customarily applies a load balancing methodology for deciding where jobs should execute. A Scheduling Framework possesses the qualities of an RMS and a Scheduler but also contains a larger spectrum of functionality.

The broad range of capabilities possessed by a Scheduling Framework distinguishes it from an RMS. MSHN offers many different scheduling and search strategies for managing resources and jobs. It provides an interface to the user for monitoring the state of the VHM and the jobs being executed on the VHM. It is able to learn by accumulating a history of performance data and can make intelligent decisions based upon on that history. MSHN also is able to deal with the uncertainty that is intrinsic in distributed environments. From a developer's perspective, perhaps MSHN's greatest asset is its modular design, which facilitates its ready adaptation to different operating environments.

3. Innovations of MSHN

The MSHN project is a departure from past approaches to distributed computing. The following six distinct innovations make MSHN unique and contribute to its increased performance, functionality and flexibility. These are:

- its ability to recognize and exploit the heterogeneity present in modern distributed computing architectures,
- its development and use of what are termed an application's Compute Characteristics,
- its ability to track and account for uncertainty,
- its ability to account for the sharing of resources in a distributed environment,
- its separation of the optimization criteria from the search engine, and
- the methods it employs to search the scheduling space for a satisfactory solution to the optimization criteria.

All computer architectures have different capabilities. A given architecture provides varying degrees of processing performance, data storage capacity, and data transmission ability. In addition, some architectures are better suited to handle particular types of applications than are others. "The MSHN team was aware of such performance differences and hypothesized that a distributed collection of machines with diverse architectures would be able to provide a collective performance equal to that of the best machine." [Ref. 4] MSHN embodies this philosophy and is designed to leverage the heterogeneity inherent in different computer architectures.

The runtimes of most computer jobs are not very predictable. "Runtime distributions typically have a very wide variance and are multi-modal in nature." [Ref. 3]

This unpredictability complicates and undermines the effort to optimally schedule a series of jobs. The MSHN development team recognizes this problem and is devising a scheme to address this challenge through the use of what are called Compute Characteristics. A job's runtime distribution can be divided into pieces delineated by these Compute Characteristics. "Compute Characteristics are most easily defined in terms of deterministic jobs executing in a quiescent system with no wait." [Ref. 3]

"The distributed environment is inherently non-deterministic. Machines are operating asynchronously, sharing resources, and executing a host of different jobs simultaneously." [Ref. 3] The developers of MSHN are able to account for this uncertainty and use their knowledge of it to increase the performance of MSHN.

The sharing of resources, such as memory, the central processing unit, and disk space, is a fundamental concern in of distributed processing. MSHN builds on the previous research that has been done in these areas, and also is pioneering the allocation of other resources, in particular, the network-based resources. "The MSHN team initiated the generalization of this work to include other shared distributed resources such as file servers and memory." [Ref. 3]

The MSHN scheduler is designed modularly. This design allows the introduction of additional components, termed optimization engines, containing new optimization criteria so long as they satisfy the interface requirements of the scheduler. In addition, MSHN permits the development of sophisticated optimization criteria that can utilize the information available in the MSHN database. The MSHN database is large and contains data useful to a broad range of optimization strategies.

The MSHN scheduler not only contains multiple optimization criteria but also several search engines. "The search engine explores the solution space for a good schedule as defined by the criteria in the optimization engine." [Ref. 3] The MSHN scheduler is also designed to allow the rapid development and integration of new search engines. The interface requirements for a search engine are subject to the same scheduler interface characteristics required of an optimization engine. Search engines in MSHN implement greedy, fast greedy, and evolutionary programming-based algorithms.

B. SYSTEM DESCRIPTION

1. Components

MSHN's architecture is divided into four different modules (see Figure 1). These are the Scheduler, Database, Learning/Accounting Process, and Controller. The Scheduler's function is to decide where and when jobs are to be executed, taking into account the types and availability of computational resources. The Database stores all the information required by MSHN to intelligently schedule and execute pending jobs. The Learning/Accounting process module allows MSHN to gather historical and statistical data for its scheduling and job management functions, and to track cost accounting data. The Controller is the main organizing process for MSHN, coordinating and implementing most of its internal activity and external interactions.

The Scheduler module is the scheduling mechanism of MSHN. Its purpose is to schedule the submitted jobs such that the best possible performance is achieved. Performance is defined in terms of user and administrator QoS metrics and the relative weighting of these metrics. For example, a common administrator QoS metric is the time it takes to execute all the jobs in the queue. MSHN achieves its high level of performance by matching jobs with the machines and resources that are best suited to process that job. In addition to the above-mentioned QoS metrics, the Scheduler decides this suitability by also taking into account sequencing, concurrency, cost, machine and resource dependencies, and the state of the VHM. The Scheduler relies on the MSHN database to supply this required information. "In truth, the MSHN Scheduler is really a family of scheduling algorithms, each designed to optimize system performance based upon different optimization criteria and constraints." [Ref. 4] The user can select which optimization and search engines to deploy or leave this decision to MSHN. This family of schedulers is not static. New optimization and search engines can be easily added and existing ones enhanced.

The MSHN Database stores and provides information about the past, current, and (estimated) future state of the MSHN environment. It maintains a record of the progress of active jobs and the location of the data they require. It also maintains a historical record of the performance and system requirements of submitted jobs, of the loads and states of all the resources available or in use, and of the global VHM.

The Learning/Accounting process has two primary functions. The Accounting function records accounting information and costs associated with the jobs being managed by MSHN. The Learning function produces a wide variety of experiential data concerning the performance of jobs and resources. The Learning function is one of the primary components that enables MSHN to make intelligent decisions. Using a variety of statistical and filtering techniques, the Learning function can measure and provide the Scheduler and Controller with both directly and indirectly measurable statistical quantities.

The Controller is the center of the MSHN Scheduling Framework. It is responsible for most of the initiation and control of MSHN's actions and the overall management of its components. Its duties and functions are numerous. These include:

- regulating interaction with the user via the interface,
- requesting schedules,
- recognizing scheduling and rescheduling events,
- maintaining accurate predictions of resource and VHM loads,
- updating state information in the MSHN database,
- monitoring job progress,
- maintaining the job queues,
- making sure jobs don't violate their cost limits,
- initiating data movement,
- executing, terminating, blocking, and migrating jobs,
- adding and removing machines and resources from its VHM, and
- communicating and maintaining consistency with other MSHN Frameworks in other domains.

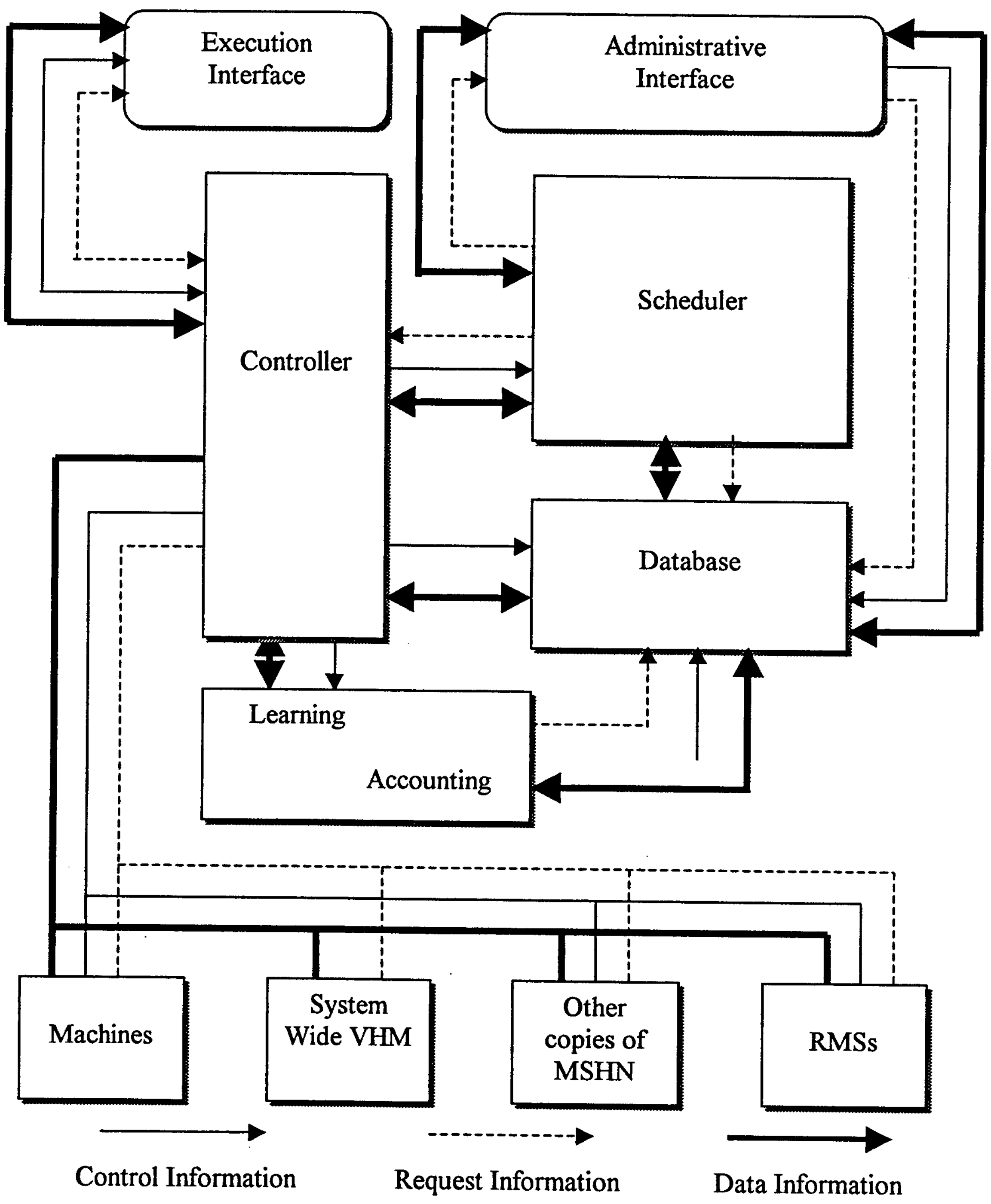


Figure 1. MSHN Basic Architecture [Ref. 3].

2. Configurations

MSHN is designed to operate in one of three different configurations: as a stand-alone environment, as an RMS advisor, or as a coordinator of many RMSs. Depending upon the configuration used, MSHN will exhibit different behavioral characteristics, performance, and capability. For each configuration, different components of Figure 1 are implemented.

MSHN can be configured to function as its own environment. In this configuration, it has explicit control over some of the resources and users of the VHM while having no direct control over others. It is able to accept job requests for execution, identify the correct machine and appropriate time for execution, ensure that data are routed properly, and finally execute the jobs. It has the means to directly interface with the machines and resources it controls, administer the users of MSHN, and monitor the state of the VHM. In this operational mode, the RMS box at the bottom of Figure 1 is left off.

MSHN can also be configured to perform duties as an RMS advisor. In this role, the original scheduling engine of the RMS is replaced by MSHN. MSHN's single purpose is to generate a recommended schedule of job execution. It accomplishes this by accepting from the RMS the lists of jobs to be executed, the VHM state information (i.e., the machines and resources available, and their current loads), and any dependency and constraint information with respect to both the jobs and resources. The RMS accepts the recommended schedule and uses it to coordinate its jobs. In this configuration, MSHN does not directly command the resources of the VHM. It is the RMS, which acts as the controlling agent. This use of MSHN prevents the RMS, and ultimately the user, from taking full advantage of the unique and effective features of MSHN. This is because the majority of the information MSHN requires to optimally schedule jobs is neither available nor tracked by current RMSs (e.g., past performance and resource architecture). In this configuration, the Execution and Administration interfaces of Figure 1 are omitted, their functionality being the responsibility of the RMS being advised, and all the boxes at the bottom of Figure 1 are left off except that entitled "RMS."

The third configuration of MSHN is as an RMS manager. In this form, MSHN takes on the role of a coordinator of multiple RMSs. As a coordinator, MSHN has the capability to migrate jobs from one RMS's domain to that belonging to another, query an RMS on the status of its jobs, and redirect the results of those jobs. These actions enhance the overall performance of the collective RMSs. In this role, MSHN also maintains the ability to interact directly with the user. The user's jobs are submitted directly to MSHN for later delegation to an RMS for execution.

3. Interfaces

There are two classes of MSHN interfaces. One is the internal class consisting of those designed to interface to the people who use MSHN. The other is the external interface that structures and regulates MSHN's interaction with the resources and RMSs of the VHM.

The internal class consists of two distinct human interfaces. One is termed the Execution Interface; the other, the Administration Interface. The Execution interface is provided for the typical user whose concerns focus on MSHN's ability to accept and execute his jobs. The Execution Interface possesses the functionality required for a user to:

- submit an application to be executed,
- provide any special instructions concerning it,
- monitor the application's progress,
- display, direct, or save their application's output, and
- perform rudimentary control functions (e.g., to terminate or dequeue a job).

The Administrative interface is provided for the MSHN administrator. The requirements of the administrator exceed those of a user. The capabilities provided by the Administrative interface allow the administrator to support the correct operation of MSHN. The Administrative interface possesses the functionality required to:

- permit new job records to be placed into the Database,
- permit existing job records in the Database to be updated/modified,
- permit resources to be added/removed from the VHM,
- monitor the VHM (i.e., the load on the resources and the progress of jobs),
- resolve scheduling conflicts, and
- access MSHN's replay, debugging, and diagnostic tools.

The External interfaces of MSHN are used to interact with the resources of the VHM. Depending on the configuration of MSHN, the interfaces may also be used to regulate interactions with the compute facilities, and the corresponding machines at these sites that MSHN controls; the RMSs that it advises; and the collection of RMSs MSHN manages. These interfaces reside with in the MSHN Controller.

C. CONFIGURATION FOR STUDY

The remainder of this study will be restricted to and focus on the stand-alone configuration of MSHN. Also, we will assume that the individual resources of the VHM will not individually possess multilevel security classifications and will also maintain their single security classification.

III. OVERVIEW OF COMPUTER SECURITY

Before continuing, it is necessary to examine the issues that underlie computer security and their impact on MSHN. Computer security is a very complex and broad subject. Security concerns have been in existence since the birth of the computer age and have increased with the growth of the industry. It is not our objective to present an all-encompassing discussion of computer security in this section. However, this chapter is meant to assist the reader in gaining an appreciation for some of the fundamental principles of computer security, to introduce the essential terms and concepts, and to build a basis for a security analysis of MSHN.

A. FUNDAMENTAL SECURITY CONCEPTS

A simplistic but meaningful definition for Computer Security is embodied by the following quote from *Practical Unix & Internet Security*: "A computer is secure if you can depend on it and its software behaves as you expect." [Ref. 6] While this supplies us with a conceptual handle on security, it relies heavily on the user's interpretation of "depend" and "expect." In the rest of this section, we will examine: 1) the fundamental security objectives, the requirements needed to achieve them and their articulation in what is termed a security policy; 2) the definition of a security policy; and 3) the functions and implementation mechanisms needed to meet security policy objectives. Because security policy enforcement is of critical importance in areas such as national defense, the ability to assure that the policy's enforcement is both correct and continuous is closely tied to secure systems development.

1. Security Objectives

The following section defines the three fundamental objectives of computer security, namely confidentiality, integrity, and availability. In many systems, one of these objectives may dominate; in others, they may all have equivalent levels of importance. It is the responsibility of the designer of a system to assess which of these objectives are critical to the user, to prioritize those objectives if necessary, and to make the appropriate design choices in the construction of the system. For each of these security objectives we provide an example in the context of the MSHN architecture.

"Confidentiality (sometimes called secrecy) requires that the information in a computer system and transmitted information be accessible only for reading by authorized parties." [Ref. 7] The regulation of the access to information by authorized users can be decided by other users and implemented as discretionary access controls, such as in Unix. This regulation can also be accomplished by imposing laws and regulations applied to the labeling of that information resulting in mandatory access controls, such as the rules governing classified information within the Department of Defense (DoD). Job characteristics that are stored in the MSHN database, such as a job's past performance on specific machines, may be required to reflect the sensitivity of that job and should not be available to unauthorized users. Classified sites that contain resources available to MSHN should only be accessible by applications possessing the proper security clearance.

"Integrity (sometimes called accuracy) requires that computer system assets and transmitted information may be modified only by authorized parties." [Ref. 7] The integrity objective ensures that a system will maintain the continuing correctness of the information stored in it. If the integrity of MSHN's Learning/Accounting algorithms is not guaranteed, then unapproved alterations in the learning heuristics can cause erroneous updates in the MSHN database. This corrupted data will have a negative effect on the scheduling algorithms that rely on such data to properly schedule user applications. Another concern is the integrity of the system files used by MSHN. The accidental or malicious modification to these files, perhaps via a virus, would cause unwanted operational behavior to occur.

"Availability requires that computer system assets are available to authorized parties when needed." [Ref. 7] The intent of the availability objective is to insure that the system, meaning both its software and hardware, is able to guarantee that the information needed by its users is kept available to those users. A user of MSHN must have confidence in that he will not be denied authorized access to MSHN. He expects to be able to submit his jobs to be scheduled. The user must be confident that his jobs will be executed, that they will finish, and the results returned to him.

2. Security Policy

A security policy is a statement of intent with regard to controlling the access to and the dissemination of information [Ref. 8]. Security policies can be grouped into two fundamental classes: Discretionary Access Control (DAC) policies, and Mandatory Access Control (MAC) policies. DAC controls a subject's access to objects based on the identity of the subject. The controls are discretionary in the sense that a subject with certain access permissions (e.g., "control" access) is capable of passing permissions on to any other subject. For example, Matt and Lynn are engineers at an aerospace company and are both authorized to see engineering documents. Matt may choose to give a project document to Lynn. Here Matt is exercising discretionary access control over an engineering document. MAC regulates access to objects based on immutable sensitivity labels associated with objects at the time of their creation and on the formal authorization (e.g., clearance) of each subject to access information of such sensitivity. [Ref. 9] To continue with the example, Jim works in the same company as Matt and Lynn but is in the marketing department. Engineers are forbidden to give technical documents to the marketing people. So, if Matt gives the same project document to Jim, he will violate the company's mandatory policy and may be fired as a consequence. A more detailed discussion of security policy and how it relates to MSHN will be presented in Chapter V.

3. Reference Monitor Concept

The Reference Monitor Concept resulted from the Computer Security Technology Planning Study conducted in 1972 by James P. Anderson & Company [Ref. 10]. The Reference Monitor Concept provides an abstract ideal with which the actual operation of security mechanisms can be compared and judged. The Reference Monitor Concept provides a basis for addressing the multilevel sharing problem. No plausible alternative to it has been advanced to date. It is believed to represent a necessary and sufficient set of components for controlling access to information. "It scopes a technically coherent subset of the entire computer security problem space without trivializing the importance of addressing other security problems." [Ref. 11] Figure 2 depicts the Reference Monitor Concept.

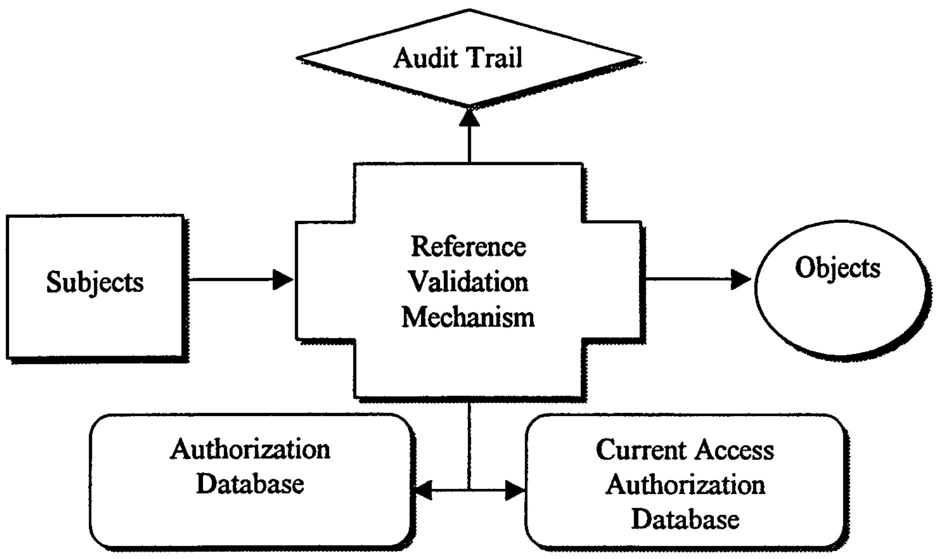

Audit Trail
Subjects
Reference Validation Mechanism
Objects
Authorization Database
Current Access Authorization Database

Figure 2. Reference Monitor Concept.

The Reference Monitor Concept is an abstraction that provides a high level methodology for controlling access to passive entities by active entities. The high level description of the Reference Monitor Concept is formalized in formal security policy models. Formal security policy models are rigorous logical models of security functionality through which the policy can be analyzed and the security aspect of system behavior proven. The Reference Monitor Concept provides a theoretical basis for the design and implementation of mechanisms for the enforcement of the security policy. Three design requirements that must be sought by any implementation of the Reference Monitor Concept are: isolation, completeness, and verifiability. Isolation refers to the requirement that the reference monitor must be tamperproof. This means that the reference validation mechanism (RVM) cannot be subject to an external attack which would modify its policy enforcement properties. Completeness dictates that the reference monitor must always be invoked, viz. every access by every program for data must be mediated. This does not mean that mediation continues once access to an object containing data is granted. That would result in unacceptable performance degradation and is not required to correctly enforce the security policy. Verifiability means the reference monitor must be small enough to be amenable to analysis and tests to assure completeness.

Two fundamental components of the Reference Monitor Concept are objects and subjects. Objects are passive entities that contain or receive information. Some examples of objects are files, directories, keyboards, video displays, printers, system clocks, memory segments, and network nodes. Subjects are active entities that cause information to flow among objects or change the system state. Subjects normally map to people, processes, and devices. The concept of a device as an active entity emerges from the fact that some devices that span multiple security levels must contain logic sufficient to correctly handle variously labeled information.

The Reference Monitor Concept is realized in the imposition of a RVM between subjects and objects as shown in Figure 2. If a subject requires access to an object, then the subject invokes the RVM. The RVM accepts the request for access and consults the authorization database. The content of the database determines if access is granted. If granted, changes to the current access authorization database are made, and the audit trail reflects the transaction.

The Reference Monitor Concept is an ideal, and will always be impossible to achieve in practice. No matter how rigorously security engineering techniques are applied to the development of RVMs, or security kernels as they are alternatively called, imperfections in the software and hardware development process makes it impossible, from a practical point of view, to design a perfect RVM. Flaws and uncloseable covert timing channels may remain to allow unauthorized information flow.

4. Trusted Computing Base

To continue our discussion of the abstraction called the Reference Monitor Concept, we now introduce the idea of a Trusted Computing Base (TCB). It is necessary to discuss this concept because the Trusted Computer System Evaluation Criteria (TCSEC) uses this term in referring to a perimeter delineating the security relevant mechanisms used to enforce the security policy from non-security relevant mechanisms. Thus the TCB may be defined as the smallest isolated subset of the system that encompasses the functions of both the reference monitor implementation and required supporting functions (see Figure 3). The TCSEC defines the TCB as:

> The totality of protection mechanisms within a computer system - including hardware, firmware, and software - the combination of which is responsible for enforcing a security policy. A TCB consists of one or more components that together enforce a unified security policy over a product or system. The ability of a TCB to correctly enforce a security policy depends solely on the mechanisms within the TCB and on the correct input by the system administrative personnel of parameters (e.g., a user's clearance) related to the security policy. [Ref 9]

Many ideas proposed in the Reference Monitor Concept are reflected in the TCB. However, a TCB is quite different. First, the TCB includes additional security supporting functions (SSF) such as password maintenance, providing a security administrator interface, and audit retrieval and analysis. Second, the TCB always refers to an automated system. Finally, a TCB is an implementation and not a high level abstraction.

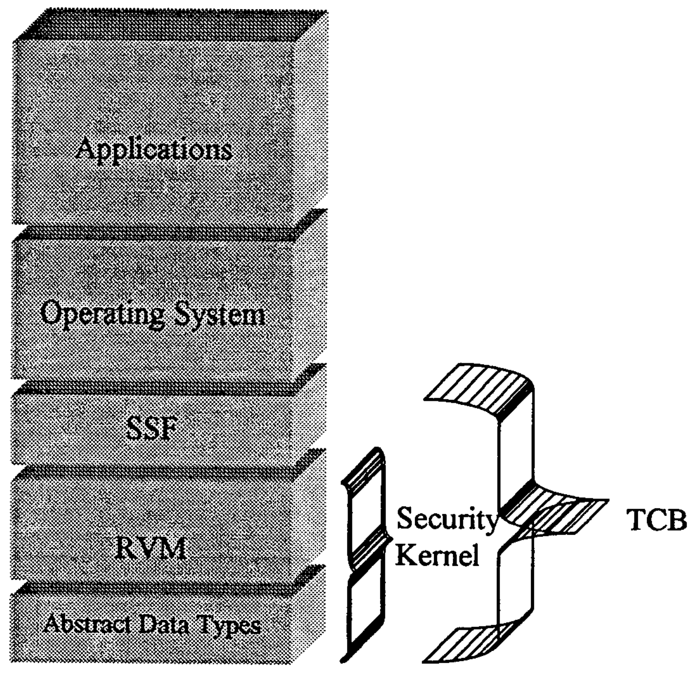


Figure 3. TCB Architecture.

B. SECURITY MODELS

A security model precisely defines the security policy, relating it to the overall behavior of the system. The primary purpose of a security model is to provide a precise mathematical description of a security policy in terms of system level operations designed to successfully implement the policy's requirements [Ref. 12]. A sound security policy model should be precise and unambiguous, easy to comprehend, and deal only with security; it should not constrain the principle function of the system. This section outlines three traditional security models. In a subsequent chapter we will discuss MSHN in the context of a more complex security policy.

1. Graham-Denning Model

The Graham-Denning model [Ref. 13] is a formal description of Protection Rules (see Figure 4) along with an Access Control Matrix (ACM) (see Figure 5), designed to enforce a discretionary security policy. These protection rules relate a set of subjects, a set of objects, and a set of access rights. There are eight Protection Rules.

- **Create an object**. Governs the creation of an object.
- **Delete an object**. Governs the deletion of an object by a subject.
- **Create a subject**. Governs the creation of a subject.
- **Delete a subject**. Governs the deletion of a subject by a subject.
- **Read access right**. Governs the reading of a subject's access right to an object.
- **Delete access right**. Governs the deletion of a subject's access right of an object.
- **Grant access right**. Governs the granting of a subject's access right to an object to another subject.
- **Transfer access right**. Governs the transferring of a subject's access rights of an object to another subject.

The ACM (see Figure 5) is constructed with each row representing a subject and each column representing an object or a subject acting as an object. The model views all subjects as having this dual existence for the purpose of determining whether a subject can exercise control (e.g., delete, read access rights, delete access rights) over another subject. Subjects are also viewed as objects when, in fact, they exhibit the behavior of an object (e.g., an executable program when being managed in memory). Each table cell contains the subject's (row) access rights to the object (column) and reflects the current security status of the system.

There are five primitive access rights: read, write, execute, control, and owner. Each object and subject in the ACM is assigned a distinct owner and controller, respectively. An owner is the subject that has the exclusive owner access right to an object, and a controller is the subject that has the exclusive control right to a subject. The Protection Rules use the control and owner access rights to determine whether the actions listed in Figure 4 may be performed by a given subject.

Command	Pre-Condition	Effect
Create object *O*	Null	Add column for *O* in ACM; Place owner in ACM[x,*O*]
Create subject *S*	Null	Add row for *S* in ACM; Place control in ACM[x,*O*]
Delete object *O*	Owner in ACM[x,*O*]	Delete column *O*
Delete subject *S*	Control in ACM[x,*S*]	Delete row *S*
Read access right of *S* to *O*	Control in ACM[x,*S*] or Owner in ACM[x,*O*]	Copy ACM[*S*,*O*] to x
Delete access right *R* of *S* to *O*	Control in ACM[x,*S*] or Owner in ACM[x,*O*]	Remove *R* from ACM[*S*,*O*]
Grant access right *R* to *S* to *O*	Owner in ACM[x,*O*]	Add *R* to ACM[*S*,*O*]
Transfer access right *R* or *R** to *S* to *O*	R* in ACM[x,*O*]	Add *R* or *R** to ACM[*S*,*O*]

Note: x represents the subject requesting access right(s)

Figure 4. Protection Rules [Ref. 27].

	Objects					
Subjects	S1	S2	S3	O1	O2	O3
S1	Control			Owner	Read Write	
S2		Control	Read			Execute
S3			Control		Owner	

Figure 5. Access Control Matrix [Ref. 27].

2. Bell and LaPadula Model

The Bell-LaPadula model [Ref. 14] is an information flow model identifying allowable paths for information flow in a secure system (see Figure 6). This model was developed and published by D. Bell and L. LaPadula to specifically reflect military (DoD) security policy.

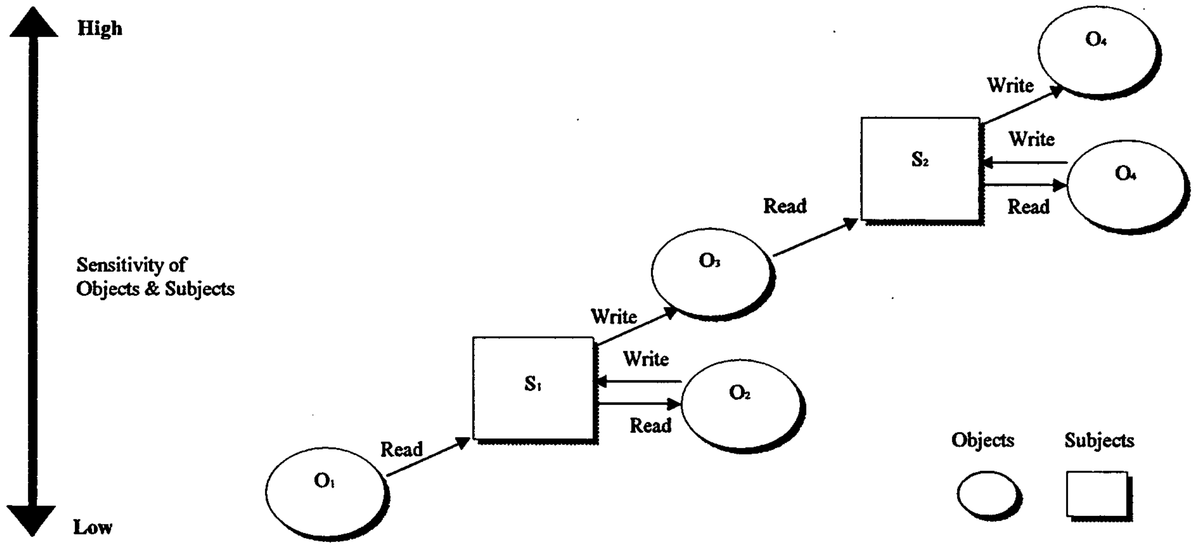


Figure 6. Secure Information Flow.

It models both mandatory and discretionary security policies. The model describes a set of subjects *S* and a set of objects *O* and a binary access class relational operator $\leq$. For example, if $A \leq B$, the access class of the left operand *A* is at the same level or lower than the access class of the right operand *B*. For every subject *s* in *S*, and object *o* in *O* there is a fixed access class *C(s)* and *C(o)*. (*C* is a function that returns the access class of the subject or object.) For mandatory policy enforcement the model presents two properties that must be maintained. These two properties work conjunctively to prevent the disclosure of sensitive information and are defined as follows.

- **Simple Security Property**. A subject *s* may have *read* access to an object *o* only if $C(o) \leq C(s)$.
- ***-Property (Confinement Property).** A subject *s* with *read* access to an object *o* may have *write* access to an object *p* only if $C(o) \leq C(p)$.

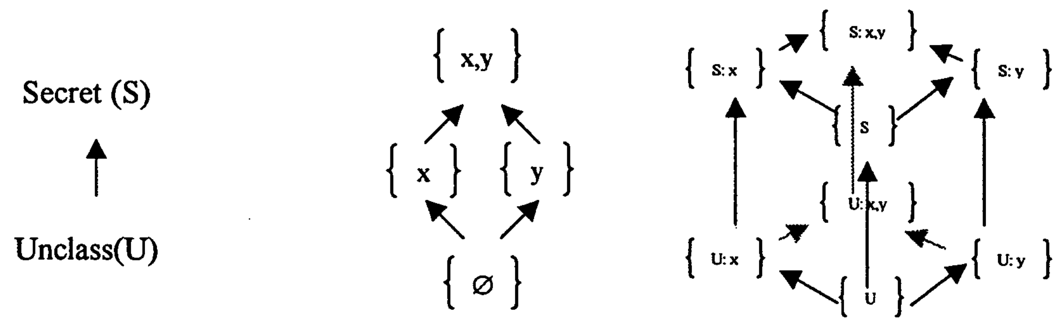
3. Lattice Model

The Lattice Model [Ref. 15] is also an information flow model, and is applicable to Mandatory Access Control security policies. Its unique characteristic is that it is represented using a mathematical structure called a lattice: a finite set of security classes and a flow relation, $\rightarrow$, with least upper bound and greatest lower bound operators. The lattice properties (reflexivity, transitivity, and antisymmetry) permit concise formulations of the security requirements of different systems and facilitate the construction of mechanisms to enforce a security policy. In terms of processes, the lattice properties as described in *A Guide to Understanding Security Modeling in Trusted Systems* [Ref 12] are:

- **Reflexivity.** A process can access any information it possesses. That is, information can always flow from a process to itself.
- **Transitivity.** If information can flow from process P_1 to process P_2 and can flow form P_2 to P_3, then information can flow form P_1 to P_3.
- **Antisymmetry.** If information can flow from a process with label L_1 to a process L_2, and conversely, then $L_1 = L_2$.

"The model provides a unifying view of all systems that restrict information flow, enables classification of them according to security objectives, and suggests some new approaches." [Ref. 15] In this model, each node on the lattice represents a particular security class that is derived from the system's set of security classes. The security classes may be linearly ordered, nonhierarchically ordered, or a combination of both, as shown in the lattices of Figure 7. Information may only flow from one node to another if the following two conditions are met.

- The sending node's hierarchical component of the security class is less then or equal to the receiving node's hierarchical component of the security class.
- The sending node's nonhierarchical component of the security class is a subset of the receiving node's nonhierarchical component of the security class.



Linear Class Lattice **Nonhierarchical Classes Lattice** **Combination Lattice**

Figure 7. Lattice Structures.

C. TRUSTED COMPUTER SYSTEM EVALUATION CRITERIA

In designing a secure system one must consider the security objectives described in the previous section. The question remains as to how to provide a high level confidence to an organization that their particular system does (or will) correctly enforce the organization's security policy. Criteria are intended to provide guidance to system developers. They outline the minimal requirements that must be satisfied in order to achieve a particular level of confidence that the policy will be correctly enforced. There are several sets of criteria to choose from: the Trusted Computer System Evaluation Criteria (TCSEC) [Ref. 9], the Information Technology Security Evaluation Criteria (ITSEC) [Ref. 16], the Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) [Ref. 17], and the Federal Criteria/Common Criteria [Ref 18]. In our analysis of MSHN, we will utilize the TCSEC. This section summarizes the requirements embodied in the TCSEC.

1. History and Purpose

In 1983 the National Computer Security Center published the DoD Trusted Computer System Evaluation Criteria, also called the "Orange Book" after the color of its cover. This document was reviewed and republished in 1985 as DoD standard 5200.28-STD. The National Computer Security Center (NCSC) was formed in January 1981 under the management of the National Security Agency. Its mission was to expand on the work started by the DoD Computer Security Initiative of 1977. The NCSC based the TCSEC upon the evaluation material produced by the National Bureau of Standards and the MITRE Corporation. [Ref. 9]

The TCSEC not only provides a rating structure for security evaluation, but also defines many computer security concepts and provides guidelines for what functionality is necessary and sufficient for a trusted system. The official purpose as stated in the TCSEC is threefold:

- To provide guidance to manufacturers as to what to incorporate in their systems to satisfy the trust requirements of a DoD evaluation,
- To give users a yardstick with which to assess the degree of trust that can be placed in computer systems, and
- To provide a common basis for specifying security requirements in acquisitions.

The TCSEC accomplishes its objectives by defining four broad hierarchical divisions for describing the protection mechanisms that are provided in a given computer system. These divisions are: D (minimal security), C (discretionary protection), B (mandatory protection), and A (verified protection). These broad divisions are further subdivided to reflect varying degrees of security capabilities within each division. These subdivisions are explained later in this section. Each division is defined by the extent to which it meets the Fundamental Computer Security Requirements that follow.

2. Fundamental Computer Security Requirements

The Fundamental Computer Security Requirements of the TCSEC are six areas that delineate what it really means to call a computer secure from a DoD perspective. These requirements define what is needed to control access to information, and to obtain accountability and assurance in a trusted computer system.

a) Security Policy

This requirement states that there must be an explicit and well-defined security policy enforced by the system. This policy is a set of rules used to determine whether a given subject can be permitted to gain access to a specific object. It also mandates that the security policy for systems handling sensitive information (e.g., classified messages) requires mandatory access controls as well as discretionary access controls. [Ref. 9]

b) Marking

This requirement states that access control labels must be associated with objects. Mandatory access control requires that every object be labeled with an identifier reflecting its level of sensitivity. Without such labels, mandatory access control cannot be implemented. [Ref. 9]

c) Identification

This requirement states that individual subjects must be identified before obtaining access to the system. Information access must be mediated on the basis of the identity and authorization of the subject requesting the access. Recall that a subject is defined as an active element that performs some security-relevant action in the system. In addition, it is required that the system provide for the protected safe storage of the identification and authorization information. [Ref. 9]

d) Accountability

This requirement states that audit information must be selectively kept and protected so that actions affecting security can be traced to the responsible party. It introduces the concept of an audit log, the repository in which relevant events are recorded. This log must be protected from unauthorized modification and destruction. [Ref. 9]

e) Assurance

This requirement states that the computer system must contain hardware/software mechanisms that can be independently evaluated and provide sufficient confidence that the system enforces the previous four requirements. As the risk to information increases, the level of confidence in correct policy enforcement must increase. This will require the application of rigorous software engineering methods, minimization of trusted code, and the use of formal methods. [Ref. 9] Configuration management and trusted distribution are also factors that contribute to assurance in high confidence systems.

f) Continuous Protection

This requirement states that the trusted mechanisms that enforce these basic requirements must be continuously protected against tampering and/or unauthorized changes. It claims that no computer system can be considered truly secure if the mechanisms that enforce the security policy are themselves subject to corruption. This requirement states that continuous protection must be provided throughout the computer system's life cycle. [Ref. 9] It is a reflection of the "tamperproofness" explained in the Reference Monitor Concept.

3. Criteria Summary

As stated earlier, the Trusted Computer System Evaluation Criteria defines four broad hierarchical divisions for the protection of computer systems: D, C, B, and A. Division C and B are further decomposed into the following classes: C1, C2, B1, B2, and B3. Each division represents a major improvement in the overall confidence one can place in the system for the protection of sensitive information (see Figure 8). It is important to note that the criteria are cumulative in that each division of the criteria inherits the security requirements of the preceding lower levels. In the TCSEC, each criterion division is presented in detail and lists to what degree it supports the six fundamental security requirements.

a) Division D: Minimal Protection

Division D only contains the Class D. This division is reserved for all computer systems that have been evaluated but fail to meet the requirements for a higher evaluation class. Class D systems cannot be expected to provide any real security or even protect against human error. [Ref. 9]

b) Division C: Discretionary Protection

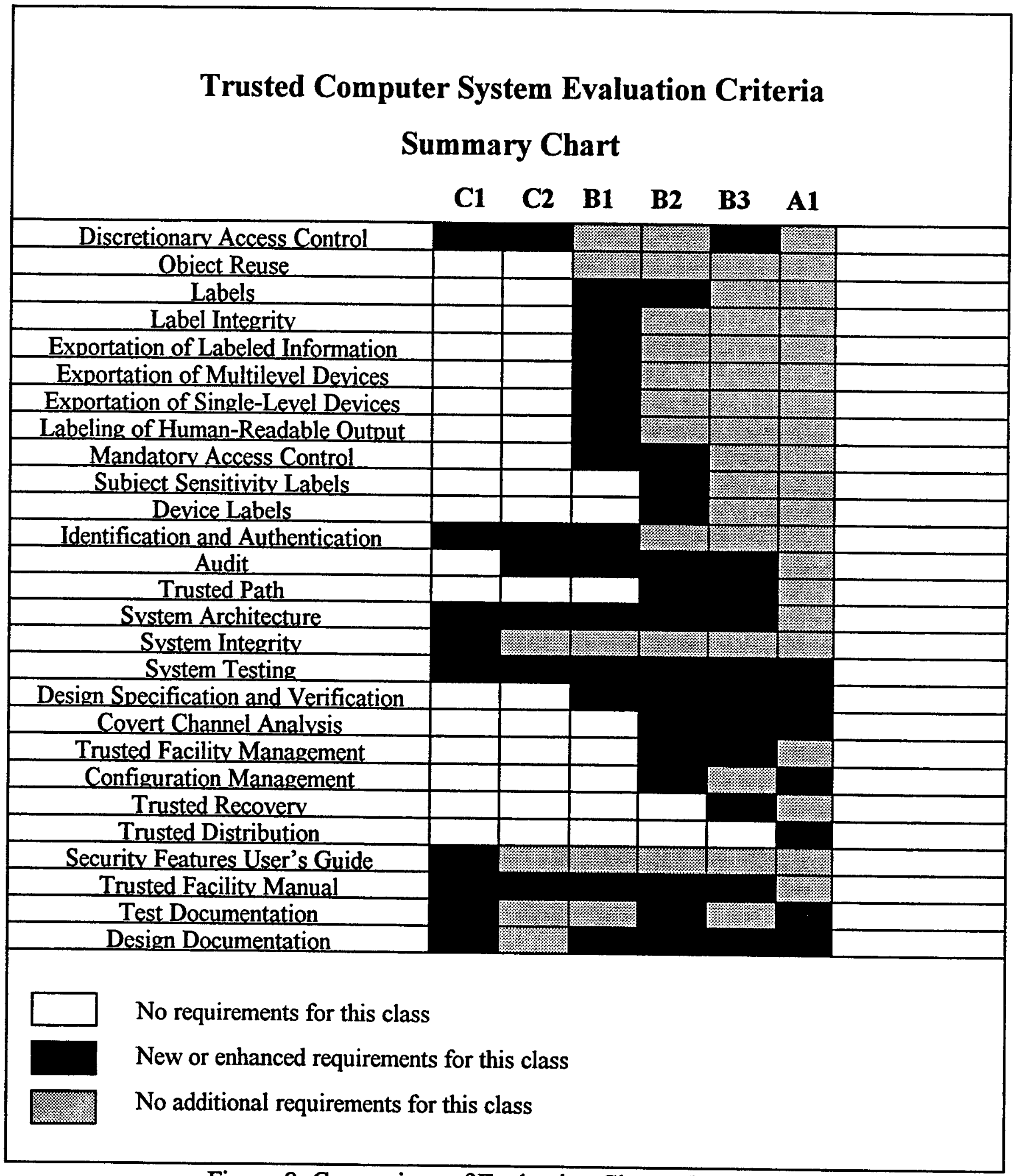
Division C contains Class C1 and Class C2. Systems in this division provide confidence to the organization that the Trusted Computing Base is enforcing a discretionary access control policy. Class C1 nominally satisfies discretionary security requirements by separating users and data. Some credible controls capable of enforcing access limitations on an individual basis are incorporated. Class C2 enforces a more granular form of discretionary access control, making users accountable for their actions through login procedures, the auditing of security-relevant events, and the isolation of system resources. [Ref. 9]

c) Division B: Mandatory Protection

Division B contains Class B1, Class B2, and Class B3. Class B1 systems feature an informal statement of the security policy model, provide for data labeling, and require mandatory access control over named subjects and objects. Class B2 systems require the TCB to be based upon a clearly defined and documented formal security model, and requires that the enforcement of discretionary and mandatory access controls be extended to all subjects and objects. Additional software engineering requirements are introduced making this class relatively resistant to penetration [Ref. 9]. The final class in this division, Class B3, dictates that the TCB substantially implement the Reference Monitor Concept requirements. From a practical perspective, it is the minimization of the TCB that contributes most to assurance by reducing the complexity of the TCB, and the number of components that must be evaluated for correctness. [Ref. 9]

d) Division A: Verified Protection

Division A contains the Class A1. Class A1 systems are functionally equivalent to B3 systems; however, the implementation of formal design specification and verification techniques is required. This results in a high degree of assurance that the TCB is correctly implemented. Also, additional configuration management requirements, such as trusted distribution, are added into the criteria of this class.



Trusted Computer System Evaluation Criteria

Summary Chart

	C1	C2	B1	B2	B3	A1
Discretionary Access Control	■	■	▨	▨	■	▨
Object Reuse	□	□	▨	▨	▨	▨
Labels	□	□	■	■	▨	▨
Label Integrity	□	□	■	▨	▨	▨
Exportation of Labeled Information	□	□	■	▨	▨	▨
Exportation of Multilevel Devices	□	□	■	▨	▨	▨
Exportation of Single-Level Devices	□	□	■	▨	▨	▨
Labeling of Human-Readable Output	□	□	■	▨	▨	▨
Mandatory Access Control	□	□	■	■	▨	▨
Subject Sensitivity Labels	□	□	□	■	▨	▨
Device Labels	□	□	□	■	▨	▨
Identification and Authentication	■	■	■	▨	▨	▨
Audit	□	■	■	■	■	▨
Trusted Path	□	□	□	■	■	▨
System Architecture	■	■	■	■	■	▨
System Integrity	■	▨	▨	▨	▨	▨
System Testing	■	■	■	■	■	■
Design Specification and Verification	□	□	■	■	■	■
Covert Channel Analysis	□	□	□	■	■	■
Trusted Facility Management	□	□	□	■	■	▨
Configuration Management	□	□	□	■	▨	■
Trusted Recovery	□	□	□	□	■	▨
Trusted Distribution	□	□	□	□	□	■
Security Features User's Guide	■	▨	▨	▨	▨	▨
Trusted Facility Manual	■	■	■	■	■	▨
Test Documentation	■	▨	▨	■	▨	■
Design Documentation	■	▨	■	■	■	■

□ No requirements for this class

■ New or enhanced requirements for this class

▨ No additional requirements for this class

Figure 8. Comparison of Evaluation Classes [Ref. 9].

IV. QUALITY OF SERVICE

In the MSHN proposal document [Ref 19], the MSHN team has identified several "Problem Areas" that require detailed research and examination in order to make MSHN a success. These Problem Areas include such topics as: Exploiting Heterogeneity, Accounting for Non-determinism, Resource Sharing, Fault Tolerance, and Managing Quality of Service (QoS). Security is one of the primary QoS objectives to be provided by MSHN. As such, this chapter will explore how QoS is defined, what services make up the QoS domain, and how security impacts the other QoS services. The results of this analysis will later assist us in developing a security policy for MSHN.

A. QUALITY OF SERVICE PERSPECTIVE

QoS is a difficult concept to explain in definable terms. Webster's dictionary defines *Quality* as degree or grade of excellence, and *Service* as an act of helpful activity. These lay definitions help us to gain a general understanding of the Quality of Service notion. Unfortunately, questions still remain. Who determines what helpful activities of a system are meaningful, and what metrics should be used in evaluating those activities? The user must ultimately decide the answers to these two questions. If the user is satisfied with the system, then the designers have done their job in ensuring that their system provides a high degree of Quality of Service. However, the word "satisfied" is an extremely vague term. A designer must somehow be able to identify those activities (services) that comprise the QoS domain, and devise some procedure to quantify the degree to which those services are supported. The following is a more logical methodology for making the identification and measuring of such services tractable (see Figure 9). This approach is very similar to the Requirement Analysis phase of the System Engineering Process [Ref. 20].

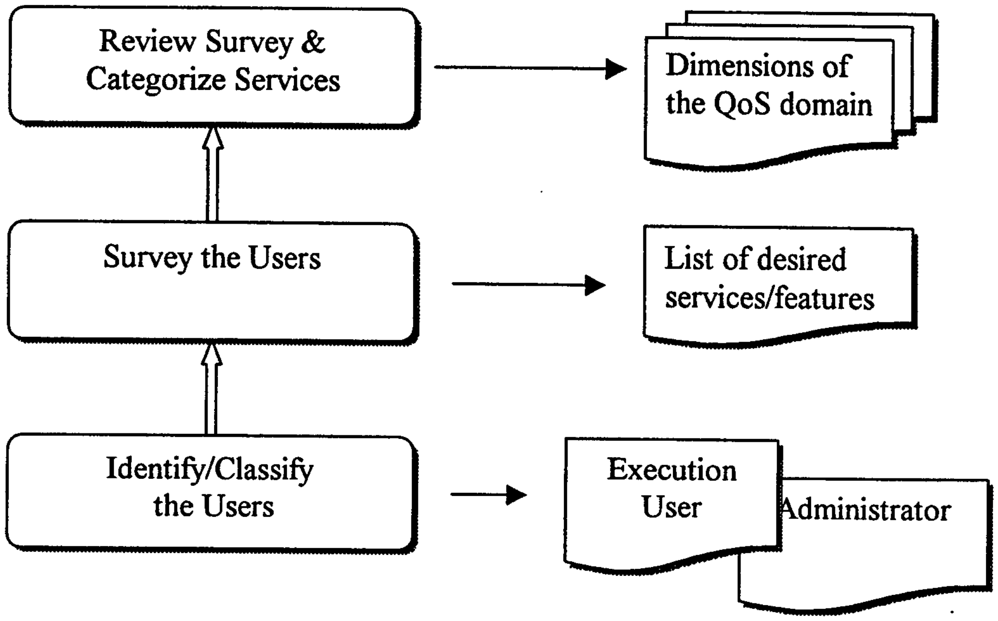

Review Survey & Categorize Services
Dimensions of the QoS domain
Survey the Users
List of desired services/features
Identify/Classify the Users
Execution User
Administrator

Figure 9. Process of QoS Identification.

The first task is to identify and classify the users of the system. MSHN envisions having two general types of users, Execution Users, and Administrators. Execution Users will be the people who submit their applications to MSHN, and rely on MSHN to properly schedule and execute their jobs. Administrators will be responsible for ensuring that MSHN is properly configured and operating correctly. These two types of users clearly will have different objectives, reasons for using the system, expectations of performance and capability, and interaction requirements. This leads us to the next phase of our methodology.

The next task is to survey what each type of user expects from MSHN. This is the step where the user articulates what features, properties, and system characteristics are important to him. Of course the survey responses of the user will be very subjective and will differ on an individual basis, but it is essential to solicit these opinions. The users are the ones who define the properties required for each service. In MSHN, the Execution User may desire a simple and clear interface such as a Graphical User Interface (GUI), while the Administrator would prefer a powerful and efficient command line interface. This is an example of conflicting requirements that must be analyzed and resolved by the designers. Once the survey is complete, the designers must review the list of services.

The last task is for the designers to examine this user-generated list of desired services to determine if, and to what extent, MSHN can supply these services. Although this list will contain specific descriptions of desired services, one may find that these services may be grouped into general categories or dimensions. Creation of a multidimensional classification system for services will assist the designers in having MSHN meet the needs of the users. The following is a list of dimensions that make up the QoS domain. It is certainly not an all-inclusive list but does contain those services that are common to most user requirements.

1. Functionality

Functionality is probably the most important service a system can provide. Functionality is what the system can do for the user. If the system provides only limited capability, or if what it can do is irrelevant to the user's needs, then the system is useless. One could measure this Quality of Service by simply comparing the user's requirements to what the system offers. MSHN definitely will provide the much-needed service of intelligently managing distributed heterogeneous computational and network resources. Its functionality will include the capability for users to be able to submit a job, monitor a job's progress, input specifications and special information related to a job, display a job's output, receive a job's data, and terminate or dequeue a job.

2. Timeliness

Timeliness is a concern to almost all users. In today's fast food society, the patience of the typical user is finite. The speed at which the system can process an application may be of paramount importance to the user. In a real time system, a delay in processing can cause catastrophic effects resulting in the loss of data, equipment and possibly even lives. Looking at the other end of the job spectrum (e.g., a simple word processor application), excessive processing time, when seen in the light of the more critical effects above, is merely an annoyance to the user. Besides viewing timeliness as a goal of QoS, it is also an economic issue. Time is a precious resource that can be quantified in monetary terms. Excessive processing time experienced by users can be detrimental to an organization's operational budget and mission. MSHN's principle goal is to effectively and efficiently schedule applications thereby meeting the QoS goals of its users (e.g., reducing the required time to execute a user's jobs).

3. Throughput

Throughput is a service that deals with quantity. The amount of information the system can process, display, transmit, and store determines the level of throughput a system provides. Some factors that affect throughput are the capacity of the system's memory and secondary storage, the network channel bandwidth, the rating of the modem, and the system architecture. Throughput, in the context of MSHN, will also depend upon how many users can access MSHN simultaneously, how many resources will be at the user's disposal, and the limitations MSHN imposes on user job submissions.

4. Dependability

A system is dependable if it is highly available, has a very small recovery time, is capable of providing uninterrupted services, and assures its users that it solves the intended problem. Dependability also implies that the system consistently achieves an expected level of performance. This QoS objective is often taken for granted. The user does not often recognize or appreciate this highly important service until it is absent. One of the ways to build dependability into a system is to concentrate on its fault tolerance. Fault tolerance is the ability to recover from hardware and software component failures without performing incorrect actions. This field of study incorporates many underlining issues such as halting failures, fail-stop failures, timing failures, and Byzantine failures [Ref. 21]. The MSHN project recognizes this critical area and has dedicated time and manpower to resolve problems associated with dependability and fault tolerance.

5. Security

Security is the ability of the system to enforce a specific policy to protect data, services, and resources against misuse. This misuse may come from unauthorized users, malicious programs (e.g., viruses and Trojan Horses) or unintentional user/software errors. This service is highly coveted by systems that process sensitive, financial, and military information. Computer security is a diverse and complex subject and was discussed in greater detail in the previous chapter of this thesis. Because of MSHN's potential military and commercial application, the designers are determined to incorporate security into its architecture.

6. Ease of Use

The last QoS element to be discussed is ease of use. If it is easy to use, the users will have a natural affinity for the system. Ease of use also results in increasing user productivity. Minimizing the time spent on unproductive user interaction with the system allows for more beneficial work to be done. In addition, the user is less apt to make mistakes (e.g., input erroneous data) when a simple, clear interface is supplied. A system that requires a knowledgeable and proficient user will restrict the number of people who can use that system. Implementing training programs, supplying reference material, and creating on-line tutorials can resolve this problem. However, these approaches can be costly and time consuming. MSHN's predecessor, SmartNet, is intended for an expert user, but MSHN plans to expand its clientele. Because of this larger and more diverse anticipated user population, the MSHN project will pay particular attention to its interface and ease of use.

B. SECURITY SERVICE IMPACT

It is imperative to realize that the services that comprise the QoS domain are not orthogonal. These services interact with and are influenced by each other. Compromises and trade-offs will take place in any system's design. Introducing security into a system will definitely affect the other QoS elements that the system provides. The magnitude of that an impact depends on the types of security policies that must be enforced and the security mechanisms applied in the system. MSHN will not be an exception. Integrating security with other QoS requirements will certainly impact MSHN's functionality, timeliness, throughput, dependability, and ease of use.

Incorporating security will rescope the functionality of MSHN. MSHN will not be an open system. Restrictions will be placed on a user's access to files, applications, features, and resources. Their accesses will be controlled by their security clearances and the permissions they possess. One of the most significant changes will be the user's view of the virtual machine. Users will only be allowed to see those resources to which they are authorized to access. This view may be only a subset of the entire virtual machine. These changes to MSHN's functionality will reflect the required security enhancements.

Security will also cause a fundamental change in the timeliness and throughput provided by MSHN. Security will be considered an attribute of the application. The scheduler will have to be modified to select resources based on their security attributes in addition to their other attributes. One ramification is that a more ideal resource with respect to some other QoS attribute may not be selected for use due to the resource's inability to meet the application's security requirements. Another timeliness issue results from the processing overhead of invoking the reference monitor. Every time a subject requests access to an object, the reference validation mechanism mediates the request. This will take time and may lead into a bottleneck situation. The exact cost of this additional time is difficult to evaluate. At the present stage of MSHN's development, the subjects and objects of the system are yet to be defined. The granularity of the objects and subjects could be high or low. The final determination of this granularity would have a significant effect on the number of invocations of the reference monitor.

Security will strengthen the dependability of MSHN. As stated earlier in this chapter, dependability is the stability and availability of the system. One of the reasons a system can become unstable is the corruption of critical operating system files. By restricting access to these files and implementing a program integrity security policy, the probability of this occurring is reduced [Ref. 22]. To ensure availability of the system, countermeasures can be put into place to guard against denial of service attacks. One such countermeasure would be to give MSHN the ability to limit a user's utilization of the system. The length of a particular user session might be bounded. Users might be given a quota on the number of applications that may be submitted to MSHN. A particular user application could be allotted a specified quota of time to run on a machine. User identity and the current system load could determine these time allotments. This apportionment will help ensure that MSHN and its corresponding resources continue to be available to all users.

Working with a secure version of MSHN will change how the user interacts with MSHN. No longer will the user have unchecked access to the system and the system's resources. The user will have to become familiar with the added security mechanisms that MSHN will contain. These include an identification and authentication login process, the setting of session levels, and the stipulations of working with discretionary access controls. Much of this could be automated to preserve MSHN's ease of use (e.g., by using smartcards, biometrics, or other easy-to-use authentication techniques). Operating in a multilevel security climate, the user will be subject to certain restrictions and limitations. For example, if the user wants to submit his application to a resource that is strictly dominated by the user's session level, he must change his session level. This must be done to preserve the confidentiality of sensitive information. This may become burdensome if the system contains many security levels and is heavily compartmentalized. The user may have to change the session level frequently to accomplish his duties. Along with requiring new system behavior, security will also generate additional administrative duties. The configuration and maintenance of user accounts, the reviewing of audit trails, and the security training of users are but a few of the added responsibilities that a secure system requires.

C. MANAGING THE IMPACT

One of the greatest concerns of the designers of MSHN is the detrimental impact that the enforcement of a stringent security policy might have on MSHN's ability to meet QoS requirements. This is a heightened concern during certain operating conditions when performance (i.e., the combination of timeliness and throughput) and resource accessibility are paramount. An example of this would be a military setting where the survivability of forces depends on MSHN scheduling and executing a critical application in a timely manner. When operating within the constraints of an established security policy, MSHN may fail to meet this requirement. The most appropriate resource to process this critical application might be inaccessible due to an inadequate number of secure channels or to the unavailability of personnel cleared to access this resource. In such a scenario the information and resources of MSHN's domain are protected but at the cost of lives. This is unacceptable.

We may have to consider having a polymorphic security policy to cope with this situation. This polymorphic security policy would have the ability to modify itself based on the working environment in which the system is operating. Under certain conditions, where the transmission of information is more crucial than its protection, a lenient security policy could be instituted. When the focus shifts back to information protection, a more rigid security policy is reinstated. At the conceptual level, an alteration of the security policy (no matter of what magnitude) is really a replacement of one policy for another. Thus a polymorphic security policy is really nothing more than multiple serial polices that the system has to manage. The difficulties and implications of a multipolicy system are discussed in Chapter V.

V. SECURITY POLICIES

A security policy contains the rules and procedures that will regulate how a system's active entities, acting as surrogates for users, manage, protect, and distribute information. The formulation of a system's security policy is the first step in building a secure system. For the proper design of MSHN it is critical that such a policy be formally stated. At the current time, no such policy exists. This chapter discusses two fundamental types of security policies that may be applied to MSHN, namely, DAC and MAC, as well as a flexible multipolicy based on these. For the first two policies, the chapter focuses on the identification and the infrastructure of enforcement mechanisms that may be applied to support these policies within MSHN. The chapter concludes with a discussion of a multipolicy security policy for MSHN, emphasizing such a policy's characteristics, the related problem areas that apply to MSHN's design with respect to implementing such a policy, and recommendations for resolution of these problems. This information will hopefully assist the designers in creating the appropriate security policy for MSHN and building effective mechanisms to support this policy.

A. ACCESS CONTROL POLICY CATEGORIES

Access control policies can be delineated into two fundamental types. They are termed identity-based policies and label-based policies. These two classes are separated and characterized by the methods and criteria they use to determine a subject's access to objects.

1. Identity Based

Identity based policies permit or deny access based solely upon the identity of the subject. Another name for this class of policies is Discretionary Access Control (DAC) policies. In certain applications areas (e.g., government and military), this type of policy may be expanded to include additional access rules based on the "need to know" of the user. For other applications (e.g., commercial and academic computing) identity based controls are simply presented as a mechanism, available to serve whatever discretionary access control needs such users might have.

2. Label Based

Labeled-based policies emerge from the assignment of trust, in the form of clearances, to users and sensitivity levels to information. Within the computer system, subjects act as surrogates for users and objects are information containers. Each are assigned immutable access classes. Comparison of subject and object access classes permits the mediation of access rights by subjects to objects. This class of policies is also termed Mandatory Access Control (MAC) policies. Enforcement of a MAC security policy is required of those U.S. government systems that are used to process classified or other specially categorized sensitive information [Ref. 9].

B. ENFORCEMENT MECHANISMS OF A DAC SECURITY POLICY

Discretionary security policies are so named because they apply discretionary access control mechanisms to control the access to information. They are probably the most common of enforcement policies. For example, they can be found in UNIX, Microsoft NT, and Novell operating systems. To better understand discretionary security policies, it is necessary to examine the requirements, characteristics, and varying forms of discretionary access control mechanisms that support such policies.

1. Requirements

All implementations of DAC security policies share the ability and the supporting infrastructure to perform the following fundamental operations. It is important to note that satisfying this set of requirements is not sufficient even for obtaining a class C1 TCSEC evaluation. However, it is beneficial to identify a subset of "core requirements" so that we may better understand the mechanisms that support DAC security policies. The first fundamental operation is that access to objects is based upon user identity. Secondly, it must be possible for authorized users to grant and revoke authorization, via some means, to objects under their administrative control. Thirdly, it must be possible for programs acting for users to grant and revoke authorizations. Lastly, the system must support the creation and deletion of objects. The exact discretionary access control mechanism used to satisfy these four requirements depends on the techniques employed, the defined access types, and the control models implemented. [Ref. 11]

2. Discretionary Access Control Mechanisms

There are five commonly used mechanisms that support a DAC security policy. They are termed capabilities, profiles, access control lists (ACLs), protection bits, and passwords. This section provides an overview of each of these mechanisms and highlights some of the advantages and disadvantages of applying them to a system.

The capabilities mechanism uses a protected identifier (the capability) that is assigned to both objects and subjects and used to determine access. A subject is only granted access to a particular object if it possesses the proper capability for that object. Two fundamental properties of the capabilities mechanism are that a capability can be passed from one subject to another, and that the capability may not be altered or fabricated without the mediation of the operating system TCB. Capabilities mechanisms are useful in enforcing the least privilege principle and providing dynamically changeable domains. The problem with the capabilities mechanism approach is that a passing of a capability is not recorded. It is difficult to assess who has access to what objects. [Ref. 23]

The profiles mechanism associates with each user a list of protected objects. This list delineates what objects the user possesses and the type of access he has to those objects. There are several disadvantages to this mechanism. If the user has access to many protected objects, the profile list can become very big and difficult to manage. Creating, deleting, and changing the permitted access to protected objects requires many operations since multiple user profiles must be updated. As in the case for the capabilities mechanism, using a profiles mechanism complicates the ability of the system to determine who has access to an object. [Ref. 23]

The ACLs mechanism takes an approach opposite to that used by the profiles and capabilities mechanisms. The ACL mechanism associates each protected object with a list of identities (e.g., users and groups). This list is referred to as the object's ACL. The access modes allowed for each identity are kept in the ACL. An advantage of this type of mechanism is that the list need not be excessively long if groups are used. Groups are a way of grouping multiple users into a single list entry. All members of a group share the privileges of that group. The use of groups introduces the problem of conflicts between individual user access rights and group rights. For example, a user may be granted only read access to an object but through group membership be given both read and write access to that object. This conflict must be resolved by a precedence schema to evaluate group and user access privileges in an ACL. [Ref. 23]

The protection bits mechanism is a degenerate form of the ACL mechanism. This method uses protection bits associated with objects instead of a list of users who may access an object. An example of the use of this technique is found in the UNIX operating system. In UNIX, the protection bits are grouped into three fields: owner, group, and public. The fields contain the access rights respectively for the owner of the object, a group, and the public (i.e., all users of the system). Each field is further subdivided into three bits, namely read, write, and execute. The value of a given bit indicates the authorization for the associated access right. For example, if the read bit of the group field is set to 1, the members of the group have read access to the object. An advantage of this technique is that it is easy to implement and manage. The disadvantage is that it lacks ability to conveniently control the access to an object at the granularity of a single user. [Ref. 23]

The password mechanism utilizes passwords to mediate access to each object with particular rights. A subject requesting access to the object must supply the correct password in order to gain access. The difficulties associated with this protection mechanism are the daunting demand put upon the user to remember all the passwords, the requirement for the selection of strong passwords, the need for changing passwords often, and the ramifications of revoking a user's access rights. [Ref. 23]

3. Access Modes

The access modes associated with an object specify what specific operations a subject can apply to that object. Numerous types of access modes are used by various discretionary access control mechanisms, but most can be found to be derivative of a few simplified access modes. The following are those basic access modes as described in *A Guide to Understanding Discretionary Controls in Trusted Systems* [Ref. 23].

- **Read.** This access mode allows an object to be read but not changed in any way. On most systems the read mode also allows the object to be copied.
- **Write.** Subjects are allowed to modify, add, or delete the contents of an object in any manner but does not allow the user to view the object.
- **Write-Append.** Subjects are allowed to expand an object but not allowed to change the previous contents of or view the object.
- **Execute.** Subjects are allowed to run the object as an executable file.
- **Delete.** Subjects are allowed to delete an object.
- **Null.** No access permissions are granted. It is used to allow the exclusion of a particular user in an ACL.
- **Control.** The subject is allowed to pass access permission for an object and to set the access modes to the object for other subjects.
- **Control with passing ability.** This is identical to the "control" access mode with the exception that the holder can pass his control permission to other users.

Of the modes described above, read and write are fundamental. Other access modes are constructed using additional mechanisms and combinations of read and write access.

4. Control Management Models

How control permissions (control, and control with passing ability) are managed in a system further dictates how information is regulated. A system may take a lenient posture and allow all users to have control permissions. This results in a very dynamic system where access changes to objects occur frequently. Alternatively, a system may allow only one user to have control permission. This results in a fairly static environment where changes to access are centrally controlled. Typically, systems apply one of four control models. They are termed hierarchical, concept of ownership, laissez faire, and centralized.

The hierarchical model implements a tree structure to manage control permissions. Objects are mapped to the nodes of a tree. If a subject has control permissions to an object at a particular node, it can control the access to objects located on all descendent nodes. The advantage of this model is that the mapping of users to the nodes can mimic the organizational structure of a large enterprise. Therefore, control can be placed at the most trusted and appropriate level.

The concept of ownership model requires that only one user is the owner of an object, in most systems this being the creator of that object. The owner is the only one with control permissions to the object. He is not able to pass that control to any other user without transferring his ownership rights as well. This eliminates any confusion concerning who controls access to each object, but also places the burden on the owner to grant and revoke access to the object by other users.

The laissez-faire model permits any user with "control with passing ability" permission to an object to exercise that right without interference from the system. This enables the user possessing such a right to pass that permission on to any other user he deems appropriate. This can result in an object having multiple controllers, each with the ability to modify its access rights. The major disadvantage of this model is that it is difficult to track the propagation of access rights because there are no constraints placed on the control of right passing.

The centralized model is similar to the concept of ownership model with the exception that there is only one owner for all of the objects in the system. Normally, in most systems, the administrator is that user. No other user can possess control permission to any of the objects in the system because the "control with passing ability" access mode does not exist. The advantage of this model is its tight control of access permissions. However, like the concept of ownership model, a significant burden is place on the controlling user to satisfy requests for access to objects by the users of the system.

5. Fundamental Flaw

The fundamental flaw of a DAC security policy is that it is vulnerable to Trojan Horses. "A Trojan Horse is a computer program with an apparently or actually useful function that contains additional (hidden) functions that surreptitiously exploit the legitimate authorizations of the invoking process to the detriment of security or integrity." [Ref. 24] A Trojan Horse may cause actions, including the transfer and modification of data, that the user is unaware of and normally would not authorize. In most systems, programs that are executed by a user inherit all the rights of that user. Because it is hidden from the user but possesses that user's access rights, a Trojan Horse is able to exploit a DAC system.

C. ENFORCEMENT MECHANISMS OF A MAC SECURITY POLICY

Mandatory security policies are so named because they are global and persistent. The majority of systems requiring MAC policy enforcement are U.S. Government systems. The data that are handled by systems using MAC mechanisms are generally sensitive. This data sensitivity generally results from the topic of the information or its source. To better understand mandatory security policies it is necessary to examine the principle requirements of MAC mechanisms, the labels they employ, and trusted subjects.

1. Requirements

The following are required features of MAC mechanisms. There exists a finite system of labels that are given to objects and subjects. The labels must not be modifiable by normal system users or by subjects operating on their behalf. There exists a relation, the dominance relation, that partially orders the labels. This partial ordering is described in the Lattice model [Ref.15] presented in Chapter III. The two fundamental access modes, read and write, are granted in accordance with the Simple Security Property and the Confinement Property. The Simple Security Property allows a subject to read an object only if the label of the subject dominates the label of the object. The Confinement property allows a subject to write to an object only if the label of the object dominates the subject.

2. Labels

Sensitivity labels are used to provide the identification of both system users and data stored within the system. "A user's sensitivity label specifies the sensitivity level, or level of trust, associated with that user; it's often called a clearance. A file's sensitivity label specifies the level of trust that a user must have to be able to access that file" [Ref. 25]. The sensitivity labels that are used by MAC policy enforcement mechanisms generally consist of two components. These components are classifications and compartments.

The standard classifications traditionally used by the DoD military model represent a hierarchical relationship (see Figure 10), whereas the compartments used represent a non-hierarchical relationship (see Figure 11). The system of classifications is said to be hierarchical because the classification labels can be arranged in a linear sequence of increasing dominance. Compartments are considered to be a set. When a particular access class both hierarchically dominates and contains all of the compartments of another access class, it dominates that class. The labels on data, or objects, may consist of both a classification and a compartment category. The object's sensitivity label must have a single classification component from one of the hierarchical classification categories. In addition to the classification component, the object's sensitivity may have zero or more compartment components. In order for the user, or subject operating on his behalf, to be granted access to an object, the user's sensitivity label must dominate the object's sensitivity label.

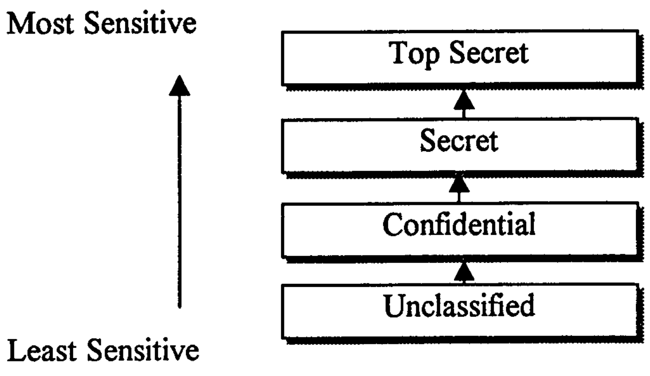


Figure 10. Classifications.

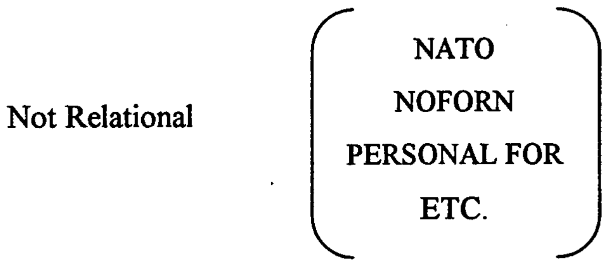


Figure 11. Compartments.

3. Trusted Subjects

A trusted subject is a subject that is permitted by the system's reference validation mechanism to violate the Confinement property. In basic terms, a subject is allowed to read information at a higher access class and write it to a lower class. Trusted subjects are internal to the Trusted Computing Base and used to perform necessary functions that would otherwise be prohibited. An example of one of these functions is the downgrading of information.

D. MULIPOLICY SECURITY POLICY

Multipolicy Security Policies (MSPs) coordinate the enforcement practices of multiple security policies when they coexist in the same system. They are metapolicies, meaning that they govern the implementation and interactions of policies. MSPs are responsible for determining a policy's domain (i.e., what objects and subjects apply to each security policy), integrating one policy with another, and resolving conflicts between the various security policies. How MSPs can be viewed, how they work, and what are some of the special concerns associated with MSPs are the topic of this section.

Potentially, the MSHN development team, driven by the purpose and QoS goals of MSHN, might be faced with the need to develop a MSP. MSHN will employ numerous resources that will be allocated to support the execution of user applications. If a resource has a security policy incompatible with that of MSHN's, a MSP must be present in MSHN to allow that resource to be useable. The second catalyst for a MSHN MSP is the possibility, proposed at a MSHN Investigators meeting [Ref. 5], of MSHN having the ability to alter its security policy to improve performance. This security rheostat implies that MSHN will have to enforce several security policies at any given time.

1. Viewing the Multipolicy system

Multipolicy systems can be characterized in one of two ways. The first view is of multiple individual security policies working together in a predictable way independent of the state of the system. The other is of policies working together with predictable behavior dependent on the state of the system.

The first view may be thought of as a composition of policies resulting in one unifying traditional security policy. The reason to describe this unifying security policy as traditional is because of its consistent behavior. The set of rules for controlling the access and dissemination of information as defined by the unifying security policy is unalterable and is consistent throughout all states of the system. This immutability of the security policy is a prevalent, and more often required, attribute in traditional security policies. An example of this view is found in systems at TCSEC class B3 and higher. These systems enforce discretionary, mandatory, and supporting policies. Each policy has its own set of rules and security goals, but they work in harmony to achieve a consolidated and consistent security policy for the system they are protecting.

We introduced the other view in the previous chapter, namely that of a polymorphic security policy. To review, a polymorphic security policy is one in which multiple individual security policies are enforced in concert. The system is able to manage these policies in such a manner that from the external perspective it appears as if there is only one security policy (i.e., a super policy) being enforced at a given time. This super policy may be a combination of the individual policies or it just may be the one security policy that is being enforced at that moment. This view differs from the unifying view in that this super policy may change during the course of the system's operation. By changing the security policy and corresponding protection mechanisms, the behavior of the system has, in effect, been changed. The tranquility property associated with the label used to enforce traditional MAC security policies is no longer valid. A polymorphic security policy suggests that we shift our way of thinking about the security policy objectives, requirements, and rules of operation.

Both of these views may be applied to MSHN. The unifying security policy view describes how MSHN should operate with resources that follow different security policies. The polymorphic security policy view permits the implementation of a "security rheostat" mechanism into MSHN.

2. Mechanics of Multipolicies

Modeling the Multipolicy Machine [Ref. 26], by David Bell, illustrates a method of treating the difficulties associated with a system that supports multiple security policies. The notions of what he terms policy combination, policy conflict, conflict resolution, and policy precedence are discussed. These notions map very well to our two views of MSPs and are, in fact, the motivation for the conception of the two views.

The majority of the concepts presented by Bell can be used to describe the workings of what we term our unifying security policy view. He describes a process for representing the multipolicy. This process uses a "policy combiner," essentially a mapping function, to fuse all of the policies into a single policy (i.e., a unifying policy). This unifying policy associates every calculation (request for access) with a value (e.g., must, may, cannot). If there is a conflict (i.e., one policy allows a particular action while an other does not) between a security policy and the unifying policy, it is resolved by one of two methods. The first is the selection of another policy combiner that does not produce conflicts. The other is through the process of policy attenuation, where the policy in conflict is asked if it can accept the decision of the unifying policy. If it cannot, another policy combiner is used and the process repeats itself until all conflicts are resolved.

The idea of policy evolution raised by Bell can be applied to our polymorphic security policy view. He also does not view the swapping of one security policy for another as a complete procedural replacement. Instead, the system recognizes and enforces all of the security policies defined by the designers. How the system is able to interact with the multipolicies is governed by a concept he terms policy precedence. Each policy is associated with a precedence value. There are two distinct forms of the policy precedence concept, absolute-precedence and conflict precedence. Absolute precedence allows for a specified policy to dominate all others. The system recognizes the policy with the highest precedence and ignores all others. To the outside world, this gives the appearance that the system is a single policy machine. To change the security policy that the system is abiding by, one would change the precedence attributes of the various policies, maximizing the precedence of the desired security policy. The other form, conflict precedence, allows for the combing of the multiple security policies into one policy. If there is a conflict between policies, the policy with the higher precedence is favored. [Ref. 26]

3. Polymorphic Security Policy Implications

Policy evolution provides insight as to how to model a multipolicy system but it does not reveal the negative implications of changing the dominant security policy. Switching security policies may have a disastrous affect on the operation of a system.

This section discusses these issues and provides some recommendations to help mitigate the negative consequences of policy evolution.

The security policy determines how information and computing resources are to be used. Change the security policy (and the applicable enforcement mechanism) and information flow may occur that was previously restricted. Sensitive information may be revealed, previously protected files may be modified, and system vulnerabilities may be exposed. These actions are allowed when a less restrictive security policy is dominant but become violations when a more confining security policy is reinstated. An important question is whether the reinstated more restrictive security policy concerns itself with the past access decisions of the previous governing security policy? This question is left to the designers of the system to answer.

If the designers do decide to include a mechanism to correct the actions of the previous security policy, they will be faced with numerous difficulties. The foremost is that of trying to revoke access to, and possession of, information to which subjects may no longer be authorized to have access in accordance with the new security policy. Problems associated with this scenario include identifying such information, locating it, and effecting its recovery. A subject may have made several copies of the information and may have sent it around the world via the Internet. There is no way to provide complete assurance that all the information will be reclaimed.

Another obstacle to switching the security policy deals with the access attributes of objects (e.g., permissions or labels) and how such attributes translate from one security policy to another. For example, if a DAC policy is enforced, objects are not marked with a label indicating its sensitivity. If that policy is replaced with a MAC policy, how does MAC adjudicate access without the existence of such labels?

To resolve these issues the following must be considered. Selection of the security policies must be carefully and methodically undertaken. One must fully comprehend the potential effects and ramifications of each policy. Maintaining an audit trail throughout all changes of policy is essential for determining what actions were taken. This information may prove to be invaluable and necessary in resuming a secure state. The protection of all the security mechanisms (e.g., auditing log, RVM, encryption devices, and trusted subjects) must be maintained with every change of policy. There must be assurance that the governing mechanisms are working properly and are not corrupted due to a policy change and consequent actions. Lastly, the object labels must be compatible for all of the security policies. This may result in having one standard label that is associated with each object and maintained throughout all security policy changes.

VI. INTERFACE ANALYSIS

Interfaces are important, they reflect, facilitate, and mediate the functionality of the system. To most users, the interface is the system. The interface development process is an integral part of the system design process. This chapter analyzes the interface of SmartNet version 2.6. This study is not conducted from the customary usability viewpoint but instead from a security perspective. By taking this approach, we hope to gain insight about the security issues pertaining to a scheduling framework. This chapter begins with a discussion of the aspects of the interface we are examining and the basis of the examination. An abridged overview of SmartNet's interface is provided followed by the identification of the security weaknesses and vulnerabilities of the interface. The chapter concludes with recommendations for resolving the exposed security liabilities. The lessons learned and suggestions noted will assist in the development of the MSHN interface(s).

A. SECURITY STANDPOINT

The approach of our study of the SmartNet interface will be from a security standpoint. This security perspective can best be illustrated by the security objectives described in Chapter III. These objectives insure the confidentiality, integrity, and availability of information. These objectives reflect the security of the system. Features that either support or conflict with these security properties are the focus of our attention.

To discuss the confidentiality and integrity of SmartNet, we need to look at the SmartNet interface-allowable actions that result in a user gaining access to information, applications, and resources. It is important to note that these actions can be autonomously caused by SmartNet as well as by the user interacting with the interface. An example of instigation of such an action is in the transmission of messages to the user about the states of currently running applications. This supply of information is not explicitly requested by the user but is sent automatically. Actions caused by the user include the activation of menus, the entering of data in dialogue boxes, and the opening of files. These actions, no matter how complex, can be reduced to some combination of two types of accesses, reading, and writing. Reading is the transfer of information from one entity to another, while writing is the modification of information.

In support of insuring confidentiality and integrity objectives, we need to look at how SmartNet's mechanisms assure the proper identification and authentication of its users and assure the certainty of transmissions. The form of the mechanisms, their strength (resistance to subversion), and, most importantly, the user-machine interface presented by these mechanisms are of interest to this study. In a network system, these mechanisms will involve cryptographic communication protocols for intercomputer security as well as those involved with internal computer security.

In looking at availability, we need to determine those actions permitted by the interface that may result in a denial of service attack. Denial of service attacks may be caused directly by a user's action, such as overloading SmartNet with numerous spurious jobs, or by less direct means that manifests themselves in a more covert fashion. An example of a less obvious attack is one in which a user corrupts the historical data that SmartNet collects resulting in a serious degradation in SmartNet's performance.

B. SMARTNET INTERFACE

The SmartNet interface consists of three primary Graphical User Interface (GUI) processes. They are the Editor, the Monitor, and the Runner. Each of the primary GUI processes has multiple "views" that the user can access (see Figure 12). Each view has its own unique, purpose, appearance, and functionality. These views allow the user to interact with and manage the SmartNet environment. In SmartNet all users have the same privileges. SmartNet assumes that the user is knowledgeable about the local site's machine and network characteristics visible from that site, and about the remote machines that can be accessed [Ref. 2]. This high level of user aptitude is required so that he may accurately enter the compute characteristics of his applications. SmartNet's ability to perform effectively depends on the accuracy of this data. The user has access to all the functionality of all the interfaces.

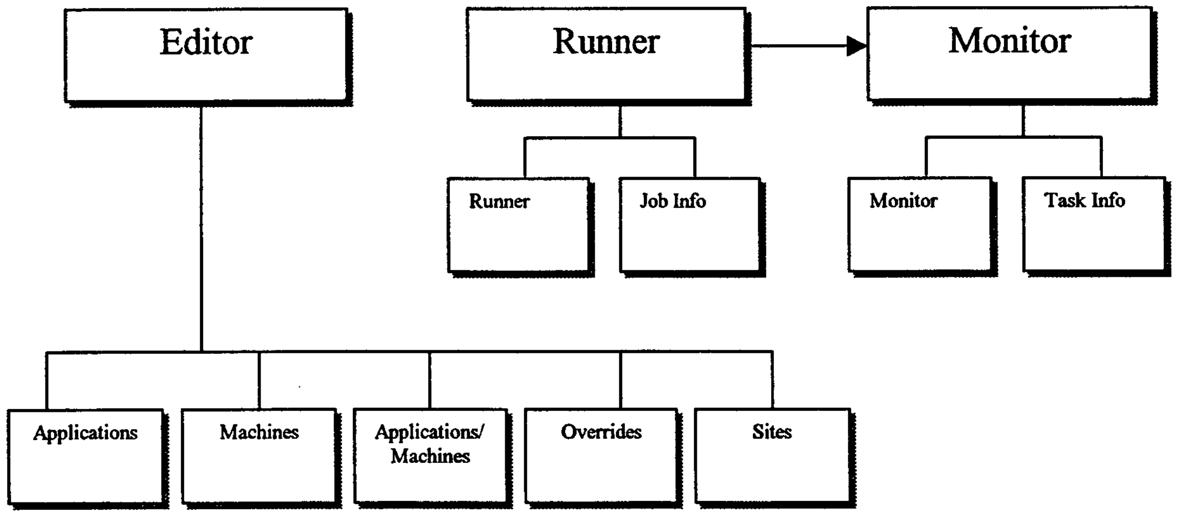


Figure 12. Structure of SmartNet's Interfaces.

1. Editor Interface

The Editor allows the user to interactively edit the SmartNet database, which contains information about the applications (user jobs), machines (potential resources), sites (location where machines reside), and networks from which SmartNet generates its schedules. The Editor provides the abilities to both query and send updates to the database. The Editor consists of five distinct views (see Figures 13 through 17). These are the Applications window, Machines window, ApplicationMachines window, Overrides window, and the Sites window. Each Editor window serves a specific purpose and provides the user with the supporting functionality listed in Figure 18.

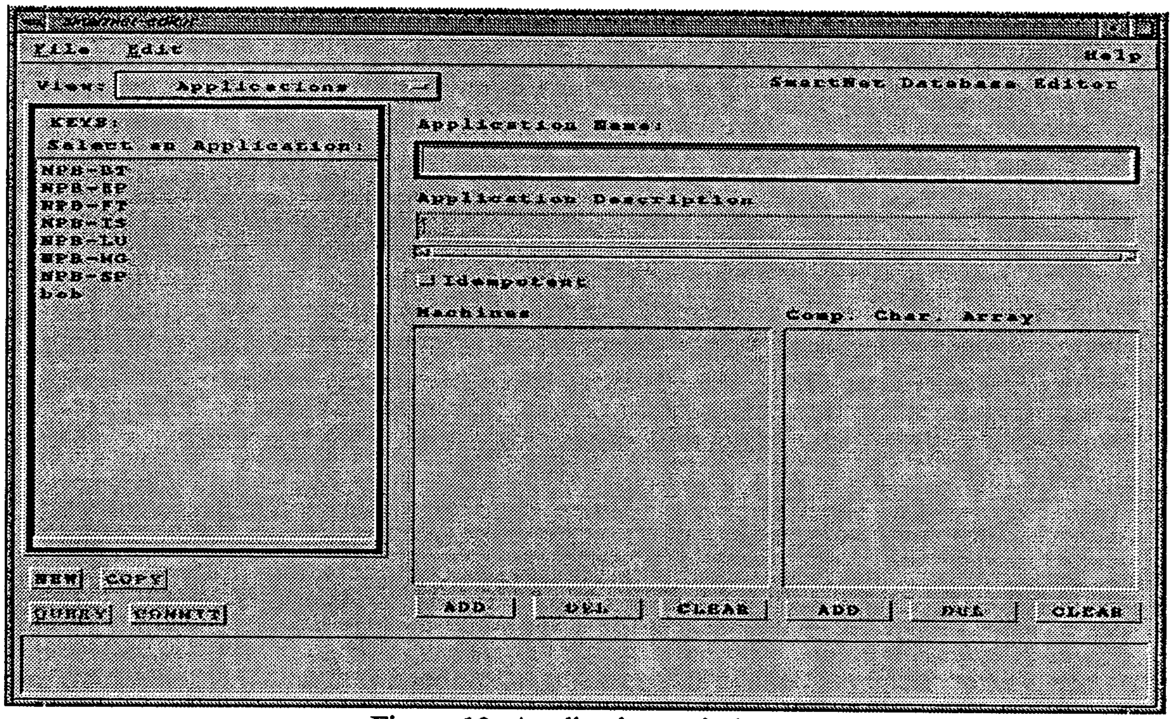
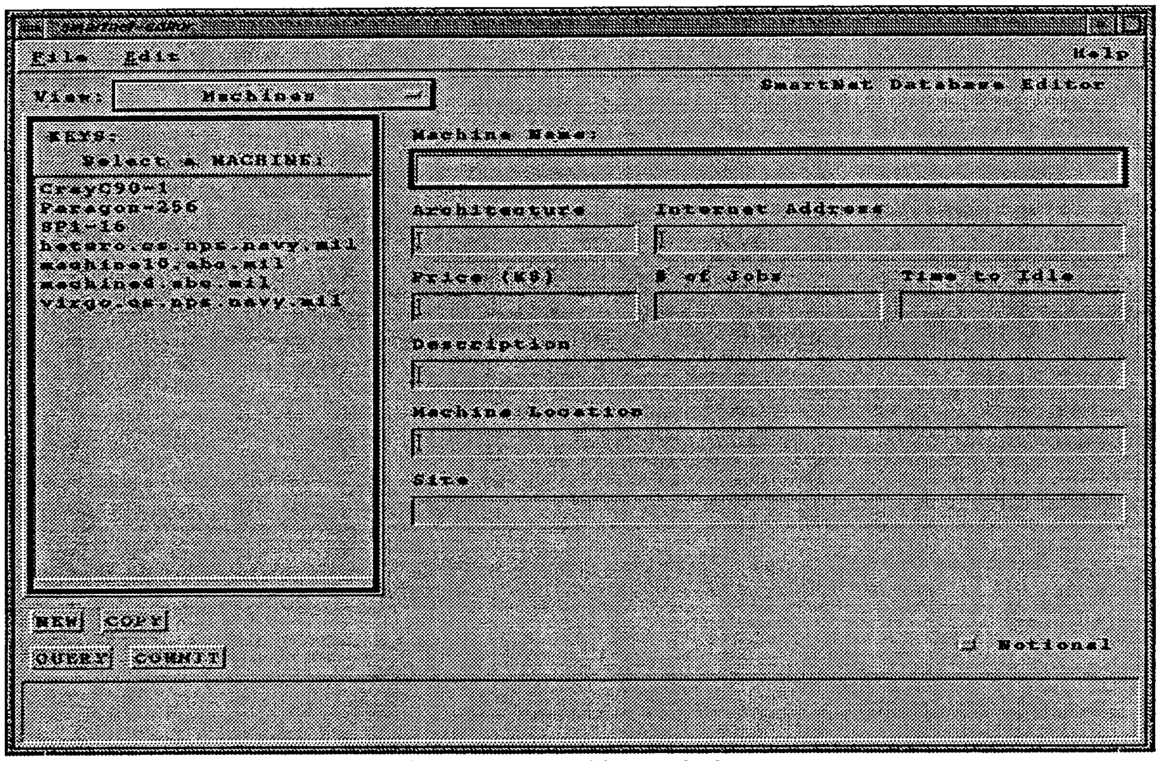
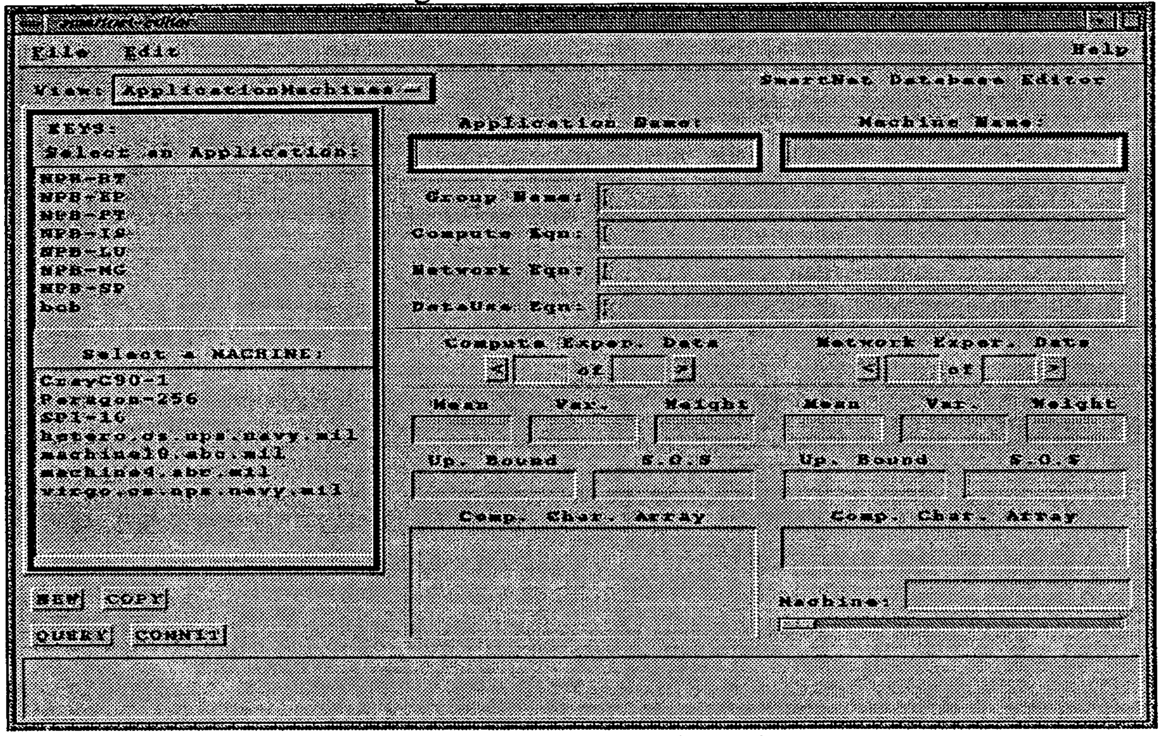


Figure 13. Applications window.



View: Machines

Figure 14. Machines window.



View: ApplicationMachines

Figure 15. ApplicationMachines window.

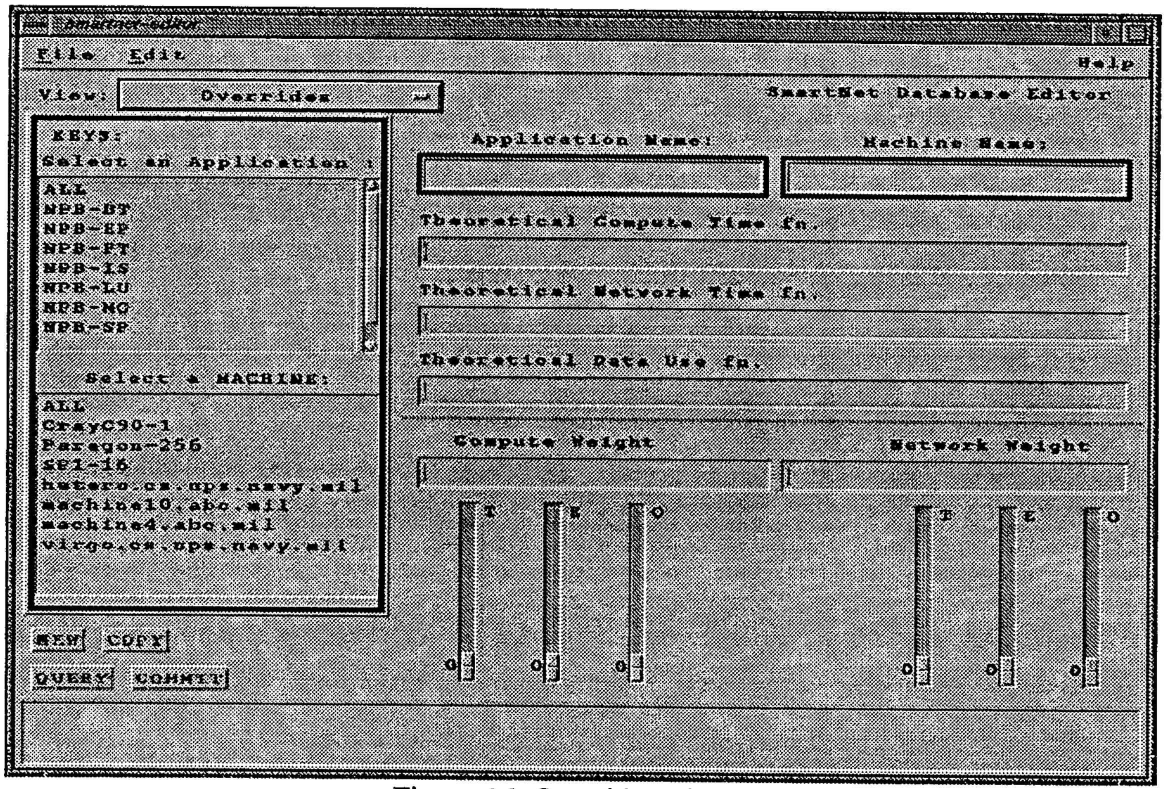

View: Overrides

Figure 16. Overrides window.

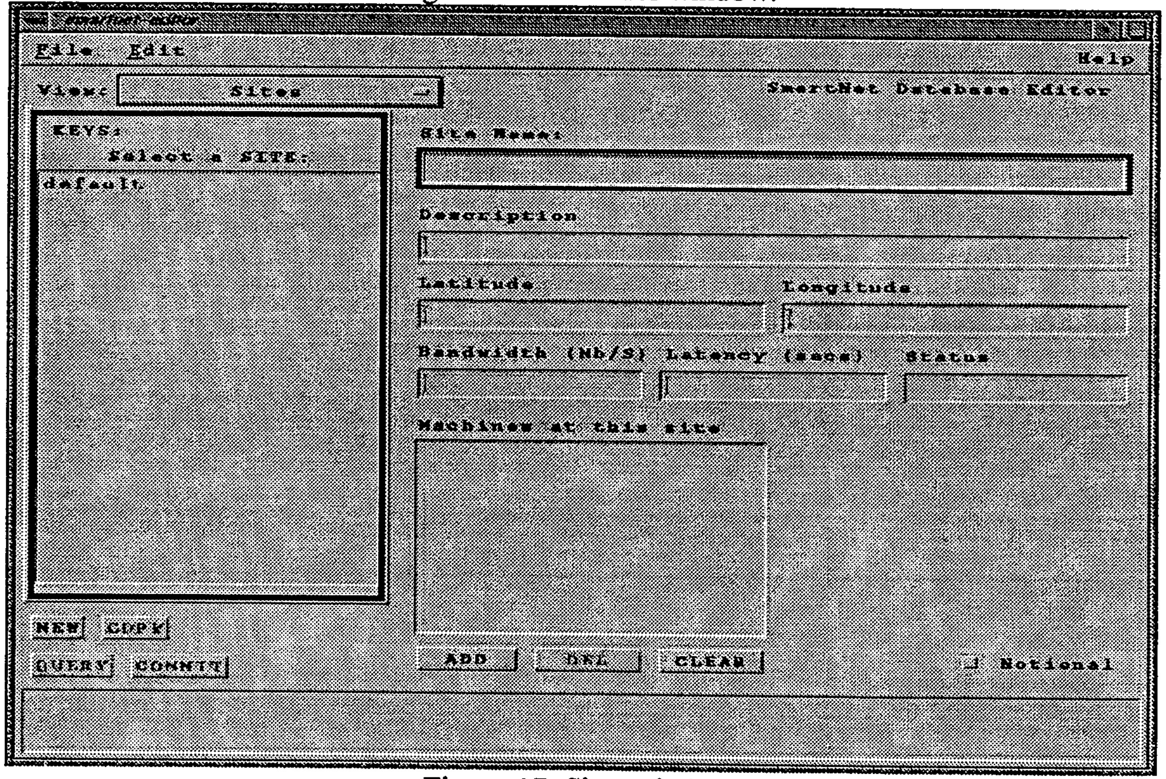

View: Sites

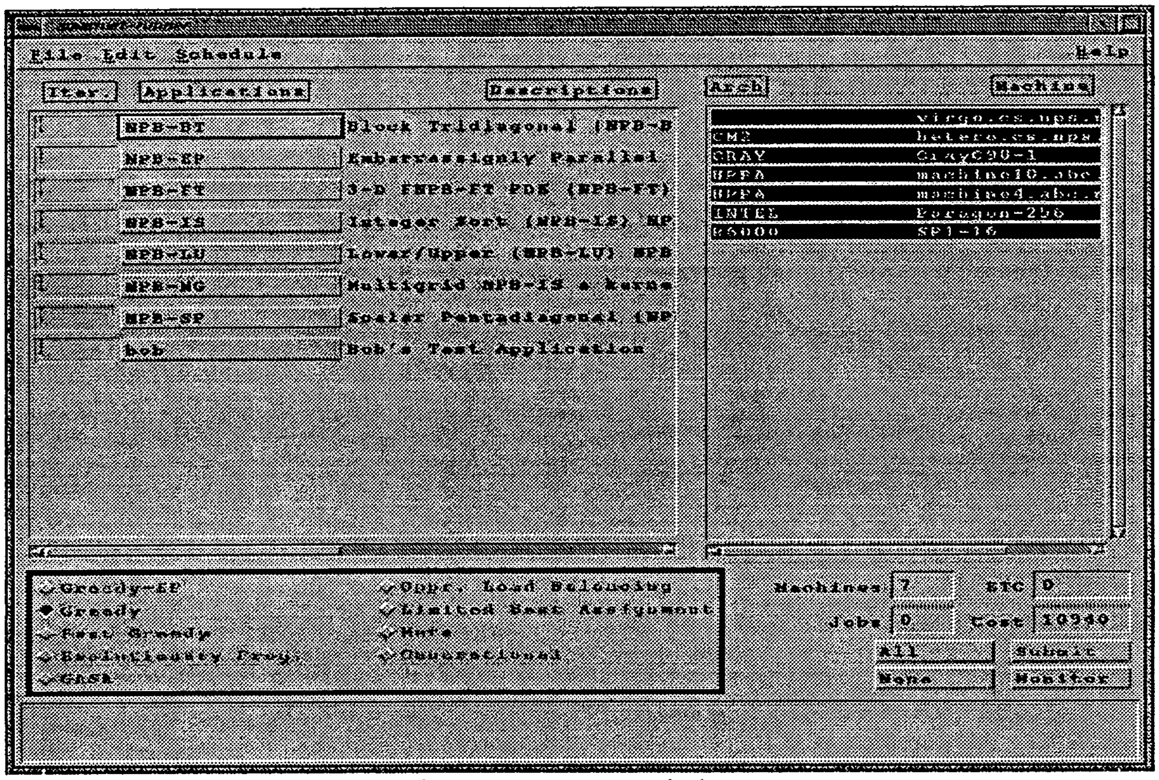
Figure 17. Sites window.

Applications	**Purpose:** Allows the user to describe applications to be scheduled and identify machines.
Capabilities: • Add *Applications* Editor Object Type (EOT) • Read the name of applications • Enter application's textual description • Indicate idempotentcy of application	• Enter application's *Compute Characteristics* • Modify application's *Compute Characteristics* • Delete machine on which application can run • Add machine on which application can run
Machines	**Purpose**: Allows the user to specify the attributes of the machines to be schedules.
Capabilities: • Add *Machines* EOT • Enter machine's architecture description • Enter machine's purchase price • Read number of jobs scheduled or running on a machine • Read time until all jobs scheduled are completed • Enter machine's textual description	• Enter machine's location within a site • Enter machine's site location • Indicate fictitious or real machine • Read the name of machines
AppMach	**Purpose**: Allows the user to view and edit the compute characteristics and experiential data associated with the user application and machine combination to be scheduled.
Capabilities: • Add *ApplicationMachine* EOT • Read name of applications • Enter group name of application machine combination • Read *Mean* value for computer and network experiential data records set • Read *Variance* value for computer and network experiential data record set • Read *Weight* value for computer and network experiential data record set • Read *Upper Bound* value for computer and network experiential data record set • Read *Sum of Squares* value for computer and network experiential data record set • Read *Compute Characteristics* for computer and network experiential data record set • Read *Counter* for computer and network experiential data record set • Enter *Computer Equations* and *Network Equations*	• Enter *DataUse Equations* • Read name of machines
Overrides	**Purpose**: Allows the user to set parameters for the user application to be scheduled.
Capabilities: • Read names of applications • Read names of machines • Enter execution, network, and data use function of the ETC	• Modify overall weighting of the compute information • Modify overall weighting of the network information
Sites	**Purpose**: Allows the user to describe a machine's site characteristics
Capabilities: • Add *Sites* EOT • Enter site description • Enter site latitude and longitude • Add, delete machines at site	• Indicate fictitious or real machine • Enter latency • Read status of site • Enter bandwidth

Figure 18. Editor Windows Capabilities.

2. Runner Interface

The Runner permits the user to schedule and run jobs on a given VHM. It also has the capability to generate only a schedule (without executing it) for planning purposes. The Runner interface consists of two views, the SmartNet Users Guide [Ref. 2] terms these views as the Runner window (see Figure 19) and the "Job Info" window (see Figure 20). The Runner window lets the user select the applications to be executed, specify the number of iterations each job will execute, select the machines that compose the VHM, and choose a scheduling algorithm. The "Job Info" window, which is a subwindow of the Runner window, allows the user to enter more specific information concerning all the applications. This information encompasses dependency, priority, compute characteristic values, and command line entries for each application. While operating in the Runner interface the user is able to access the Monitor interface.



Applications	Descriptions
NPB-BT	Block Tridiagonal (NPB-B
NPB-EP	Embarrassingly Parallel
NPB-FT	3-D FNPB-FT PDE (NPB-FT)
NPB-IS	Integer Sort (NPB-IS) NP
NPB-LU	Lower/Upper (NPB-LU) NPB
NPB-MG	Multigrid NPB-IS a kerne
NPB-SP	Scaler Pentadiagonal (NP
bob	Bob's Test Application

Arch	Machine
	virgo.cs.nps.n
CM5	hetero.cs.nps
CRAY	CrayC90-1
HPPA	machine10.abc
HPPA	machine4.abc.n
INTEL	Paragon-256
RS6000	SP1-16

Greedy-EP
[✓ Greedy]
Fast Greedy
Evolutionary Prog.
GASA
Oppr. Load Balancing
Limited Best Assignment
Mars
Unrestricted

Machines 7 ETC 0
Jobs 0 Cost 10940

Figure 19. Runner window.

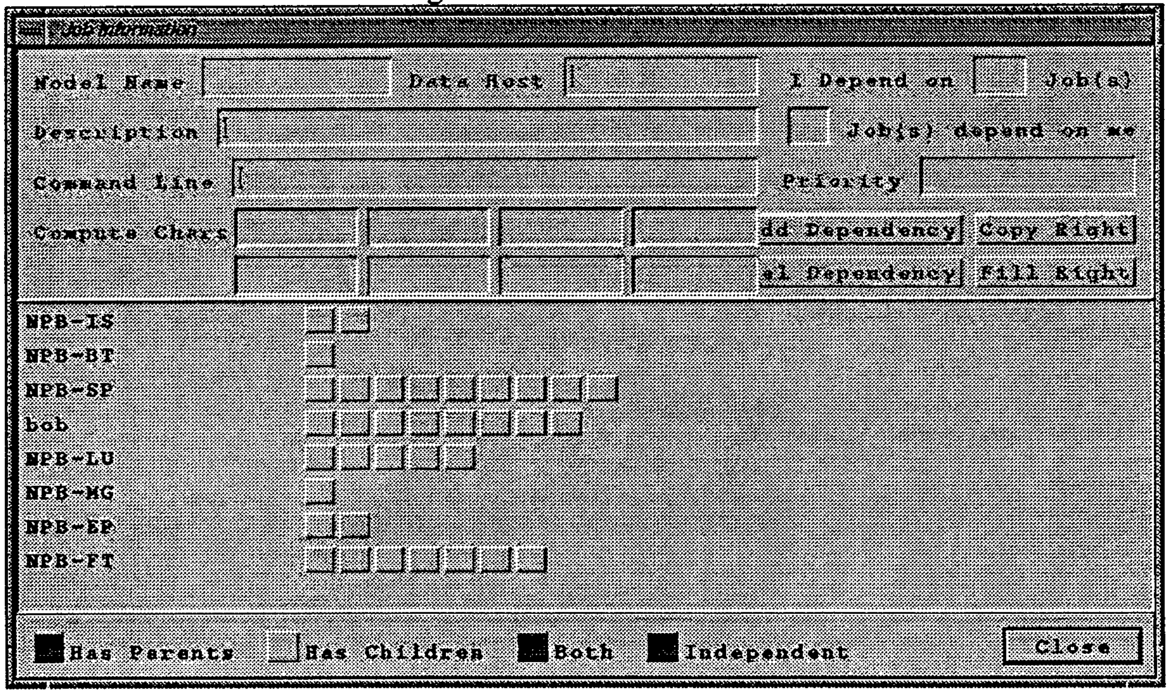


Figure 20. "Job Info" window.

3. Monitor Interface

The Monitor presents the user with a real time look at the jobs that are currently running and scheduled to be run by SmartNet. It is strictly a passive interface. The user is not able to manipulate the state of SmartNet in any way. The Monitor consists of two views, the Monitor window (see Figure 21), and the "Task Info" window. The Monitor window displays, in bar graph form, all jobs currently scheduled and indicates the machines on which they are scheduled to run. Each bar represents a job. The jobs currently executing are distinguished from waiting jobs by a flashing bar. The "Task Info" window is activated from the main window by clicking a job's bar. The "Task Info" window displays additional information pertaining to that job such as name, duration, start time, and status.

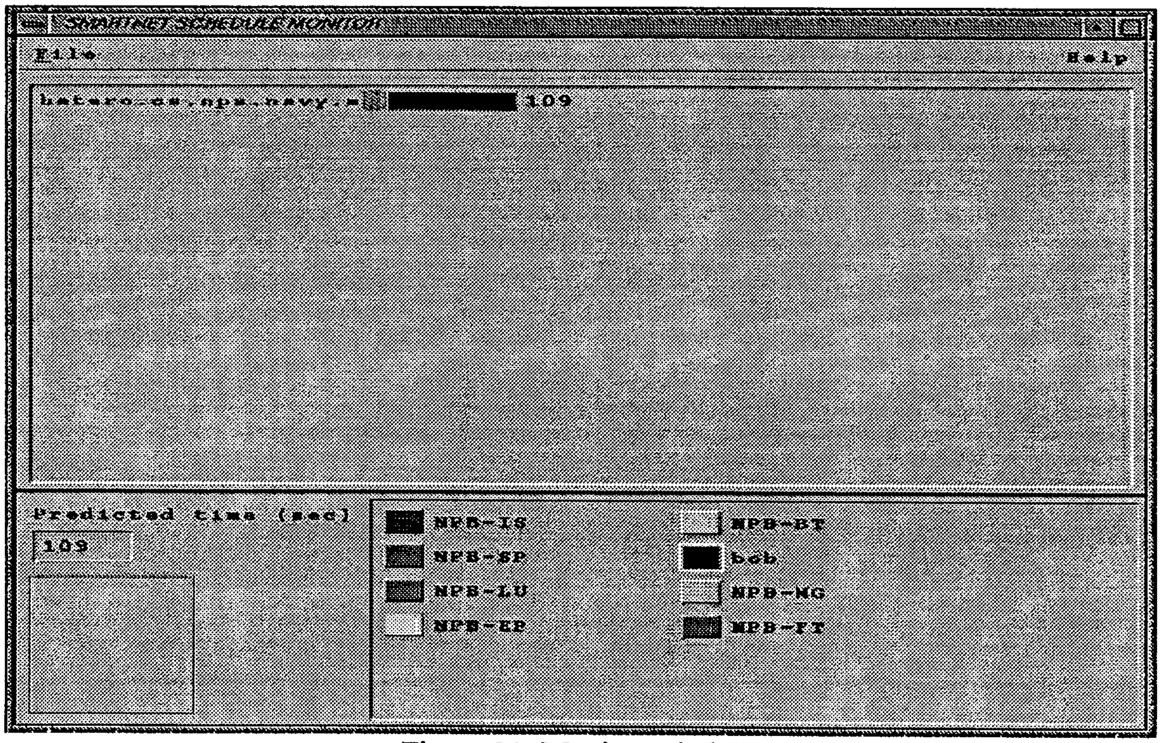


Figure 21. Monitor window.

C. VULNERABILITY AND WEAKNESSES

Our examination of the security inadequacies of the SmartNet interface is organized by reviewing each of the three primary interfaces: the Editor, Runner, and the Monitor. Security problems common to all of the interfaces are noted as well as those distinct to a particular interface. Specific examples of features that expose SmartNet to potential security threats are highlighted and their consequences explained.

1. Common Security Problems

There are several security weaknesses shared by all of the three primary interfaces. The first is that there is no identification and authentication procedure. Users are not required to uniquely identify themselves to SmartNet. As result, control over the access to information, either by discretionary or mandatory means, cannot be accomplished. Subject identity and associated rights must be established for the proper mediation of information access. Without this feature it is impossible to uphold any of the security policies and assign accountability to users for their actions.

The second mutual security deficiency is the all-encompassing capability of any user to view the information pertaining to any job, and to manage all of the applications and resources of SmartNet. Jerome Saltzer and Michael Schroeder expostulate on the design principle of "least privilege." [Ref. 27] This principle states that every program and user of the system should have the least set of privileges necessary to complete their job. The "least privilege" principle limits the damage that can result from human accident or program error. In a military setting, this principle can be compared to the security rule of "need-to-know."

The third weakness is the absence of an audit trail. "An audit trail is a chronological record of system activities that is sufficient to enable the reconstruction, reviewing, and examination of the sequence of environments and activities surrounding or leading to an operation, a procedure, or an event in a transaction from its inception to final results." [Ref. 28] So long as the audit mechanism is not subverted, the audit trail will allow a penetration of SmartNet to be detected and future penetrations deterred by revealing any misuse of the system.

The final security weakness is that information transmitted via the network is not encrypted and therefore susceptible to interception. This could lead to replay attacks (the passive capture of a data unit and its subsequent retransmission), masquerade attacks (where one entity pretends to be another), and the modification of data.

2. Editor Security Problems

The overarching security problem with the Editor interface is that it allows the user to modify all aspects of all the applications and resources of the VHM. This user control of vital information governing the proper execution of applications is unchecked by SmartNet. The user is able to directly affect SmartNet's ability to schedule by modifying information stored in the SmartNet database pertaining to jobs that may or may not belong to him.

Specifically, the user can alter a number of application and resource parameters that will have adverse effects on the operation of SmartNet. The user can change the compute characteristics of an application in the Applications window causing SmartNet to improperly schedule the application. He may redirect transmissions (e.g., job execution commands or job results) by changing the Internet address of the machine in the Machines window. He may create bogus applications, machines, and sites, in the Applications, Machines, and Sites windows, respectively, to confuse the other users of SmartNet. He may change the SmartNet server to site bandwidth settings in the Sites window causing SmartNet to miscalculate the impact of the network. He may input unrealistic values for the Compute and Network weight thereby altering the results of the SmartNet's Estimated Time to Completion function. All of the above actions are not restricted to either SmartNet nor its users.

3. Runner Security Problems

The Runner interface restricts the user to only those operations relating to the execution of his jobs. However, the user does have the capability, via the interface, to indirectly affect the other users of SmartNet. The functionality of the Runner interface exposes SmartNet to three security vulnerabilities: the denial of service, the corruption of historical data, and the exploitation of the resources of the VHM.

The first security vulnerability of the Runner interface is that it allows a user to conduct a denial of service attack. A user may interrupt the operation of SmartNet by resetting the server. This is done by using the Schedule pull down menu on the Runner window and selecting Reset Server. This will cancel all of SmartNet's currently running and scheduled jobs. This means that all jobs, including those of other users, are terminated. The user may also conduct a denial of service attack through a less overt method. This is achieved by submitting an unusually large number of jobs for execution. He may target an individual machine or a selected group of machines by selecting them as the execution platforms. This overloading of jobs will cause an excessive delay in the processing of other users' jobs.

Unlike the Editor, the Runner does not allow the user to modify the SmartNet database directly, but it is possible to do so indirectly. The SmartNet database contains a historical track record of the performance and needs of jobs. Each time a job is executed, SmartNet records runtime information. It uses that information in the future scheduling of that specific executable. A user could submit to SmartNet a series of a particular executable and provide bogus parameteric and characteristic information (i.e., compute characteristics and dependency information). This bogus information would result in poor performance for that job. SmartNet would incorporate this corrupt data into the collective job information contained in its database. This corruption of the job's runtime will adversely affect what SmartNet sees as the characteristics of that job, and so, the ability of SmartNet to schedule that job.

The last security weakness is that the Runner interface exposes resources to potential exploitation. Specifically, the "Job Info" window permits the user to enter commands that will be received at the resource tasked to execute that job. The purpose of this feature is to associate the application and its related parameters to the resource tasked to execute that application. This information is required for the execution of the job at the resource. The user could use this feature to enter in commands intended for harmful purposes.

4. Monitor Security Problems

The Monitor's interface function, to provide a real time overview of the VHM, poses two security problems. These problems are violations of the Simple Security Property (a property of the Bell and LaPadula security model discussed in Chapter III) and facilitation of inference.

The Monitor interface allows a user to view the application names, start times, durations, job identifications, predicted times to finish, status, number of iterations, and the assigned machines. The Monitor interface also displays information about the machines running the applications, such as their machine name and network address. This information may be sensitive and therefore should only be available to users with the proper clearance and need to know. This ability for a user, in particular one without the proper clearance, to read sensitive information violates the Simple Security Property. A less overt manner in which information may be illegally passed between a high user; that is, one operating at a high security level (e.g., secret), and a low user, one operating at a low security level (e.g., unclassified), is through a prearranged signaling protocol. The high user may signal the low user by submitting a job to a specific machine. The number of jobs that are submitted to that machine, the time submitted, and the name of the job itself could all convey information to the low user.

The last security problem is that the Monitor interface supports inference. Inference is "the occurrence when a user is able to deduce information to which they do not have privilege from information to which they do have privilege." [Ref. 24] For example, a low user observes that SmartNet has tasked the machines at a military high security site with an extraordinary number of jobs. The low user may be able to deduce that the military is utilizing the resources managed by SmartNet to prepare for an impending conflict. The military's order to maintain silence about the operation and preserve the element of surprise is subverted.

D. RECOMMENDATIONS

This sections outlines recommendations to counteract the previously noted security weaknesses in SmartNet's interfaces and operation principles. The first three suggestions are required features of all systems rated Class C2 and higher in the TCSEC classifications [Ref. 9]. The last recommendation addresses the separation of user from system administrative capabilities in SmartNet's interface.

The first recommendation for a more secure SmartNet is the insertion of an Identification and Authentication (I&A) mechanism into the system and the interfaces. An I&A procedure involves the user establishing a communication path to SmartNet, identifying himself, and then supplying one or more authentication elements as proof of his identity. SmartNet, using the claimed identity and authentication elements as parameters, would then validate the supplied information against that contained in an authentication database (e.g., a password file). If satisfied with the verification process, SmartNet establishes the user's session. SmartNet must also protect the I&A data. I&A data transmitted during a login session is vulnerable to interception (like most transmissions). Protection of this transmission could include physically securing the wires between the user and the I&A mechanism (not feasible for a configuration of SmartNet that includes resources that are dispersed throughout the country) or applying a cryptographic protocol to the transmission. Implementation of an Identification and Authentication process will ensure that only authorized users have access to SmartNet. It will also facilitate the incorporation of the remaining recommendations. [Ref. 29]

The second suggestion is the development and implementation of an Audit Trail mechanism. The audit trail will serve five primary functions. It will allow the review of patterns of access to individual objects, the discovery of repeated attempts to bypass the protection mechanisms, and the discovery of users assuming greater privileges (e.g., a user assuming the system administrator role). The audit trail will act as a deterrent against a perpetrator habitually attempting to bypass SmartNet's protection mechanism and also improve SmartNet's ability to control the damage if such attempts are successful. [Ref. 28]

The third recommendation is the enforcement of discretionary access controls. This will provide a measurable degree of information control within SmartNet. Further explanation of DACs and techniques for their implementation are provided in Chapter V.

The last suggestion is to differentiate the user capabilities from the administrative capabilities. It is clear from this study that the average user (one who uses SmartNet to schedule his jobs) does not need all the capability that the current interface provides. However, the system administrator, as the one who ensures SmartNet operates correctly, does in fact need all of the current capability to carry out his responsibilities.

To accomplish this separation the following process is recommended. Determine the minimum set of functionality that the user requires in order to submit, configure, and execute his applications. Resolve those operations that conflict with the security principles, such as those noted in the previous section. This resolution may require one or all of the following: the elimination of some capabilities, the placement of constraints on the allowable input, or the implementation of some type of error checking procedure (either automatically or via human intervention). Next, those features that the user deems "nice to have" should be reviewed for possible inclusion into the user's capability domain. Here, particular attention should be given to their security implications. A similar process is then conducted for the administrator's capabilities. The conclusion of this process will result in two distinct interfaces, one for the user and one for the administrator.

VII. CONCLUSIONS

A. CONCLUSIONS

MSHN is a program that is building upon the experience of SmartNet. SmartNet was designed and constructed as an open system. Its implicit security policy makes no distinction between users, and their access to information and resources is not constrained by the system. This simplified system development and maximized functionality, facilitating the creation of a high performance, flexible, and capable scheduling framework. The users of SmartNet enjoyed an all-encompassing ability to control and view the entire VHM.

One may be led to believe that the lack of security in SmartNet is acceptable under certain operating environments. For example, in a controlled environment where the users are trusted to do the right thing, the information processed is not of a sensitive nature, and the connectivity is well-known and regulated, users accept the absence of security. Our discussion here indicates that these beliefs are unfounded. Infection by Trojan Horses and viruses, mistakes caused by users, or deliberately malicious activity can occur and adversely affect the ability of SmartNet to effectively schedule and execute user jobs. Security does more than protect the dissemination of information. Security ensures the proper operation of systems by providing those systems with an expected level of secrecy, integrity and availability.

B. RECOMMENDATIONS

The following section presents recommendations and future directions for the incorporation of security into MSHN.

The MSHN design team must first clearly identify the expected customer base and their intended use for MSHN. From an economic perspective, MSHN must satisfy the needs of the user in order to be commercially successful. It is user requirements that ultimately shape the design and functionality of the system. Specifically, it is their security policy requirements that will lead the design team in determining the degree to which MSHN will accomplish the three security objectives described in Chapter II, namely confidentiality, integrity, and availability.

This leads to the next recommendation: that a security policy must be defined for MSHN. As Chapter IV states, it is critical to the proper design of MSHN to formally state the rules and procedures that will regulate how MSHN manages, protects, and disseminates information. The articulation of a policy is the first step in building a secure system. It will characterize the behavior, capability, and the trust of the system. It is counterproductive to proceed with the application of security mechanisms without this vital governing statement of intent.

Independent of the type of security policy applied to MSHN, it is sound practice to incorporate into MSHN's functionality the ability to identify and authenticate users, and the ability to construct an audit trail. As explained in Chapter VI, these two security services will deter the penetration of MSHN by unauthorized users and will contribute to the system's ability to make users accountable for their actions. Through the audit trail, it will be possible to determine which users are using the system improperly.

Another recommendation emerging from Chapter VI is that MSHN should adopt the philosophy of "Least Privilege." Users of MSHN should only have the ability to modify those aspects of applications and machines that are part of their responsibility or concern. Unconstrained malicious actions by some subset of users will impact the correct operations of MSHN and its ability to schedule jobs appropriately. There must be some access control mechanism instituted in MSHN to insure the integrity of the system's management database.

The last recommendation is that the MSHN design team should continue research and examination of the multipolicy security policy issues presented in Chapter V. MSPs will enhance the adaptability of MSHN and potentially increase the number of resources that can be utilized by MSHN. This will increase the effectiveness and marketability of MSHN. However, multipolicies are a relatively unexplored area. Major challenges remain to be addressed for multipolicies to be successfully applied to MSHN as discussed in Chapter V.

C. SUMMARY

The MSHN team has committed to incorporate security into MSHN to promote its viability, marketability, and trustworthiness. In this thesis we have undertaken the first steps in realizing this commitment. We have demonstrated how fundamental security objectives, principles and policies may be applied to MSHN. We have qualified the notion of QoS. We illustrated the role of security and its influence over the other services in the QoS domain. We have stressed the importance and purpose of a security policy, and presented a discussion of the types of security policies and the available mechanisms for supporting such policies. We have concluded with a security analysis of the SmartNet interface such that the vulnerabilities noted might be avoided in the design of the MSHN interface as well as throughout the architecture.

LIST OF REFERENCES

1. Freund R., "SuperC or Distributed Heterogeneous HPC", *Computing Systems in Engineering*, 2(4): 349-355, 1991.

2. Naval Command, Control, and Ocean Surveillance Center Research, Development, Test and Evaluation Division, *SmartNet User Guide V2.6*, June 1996.

3. Freund R., and others, "SmartNet: A Scheduling Framework for Heterogeneous Computing", *International Symposium on Parallel Architecture, Algorithms, and Networks (ISPAN '96)*, July 1996.

4. Kidd T., unpublished notes 1994-1995.

5. MSHN Investigator Meeting, Naval Postgraduate School, Monterey CA., August 25 –26 1997.

6. Garfinkle S., and Spafford G., *Practical Unix & Internet Security*, O'Reilly & Associates Inc., Sebastopol, California, 1996.

7. Stallings W., *Network and Internetwork Security Principles and Practice*, Prentice Hall, Englewood Cliffs, New Jersey.

8. SRI International, *Secure Distributed Data Views*, RADC-TR-89-313 Volume I, December 1989.

9. National Computer Security Center, *DoD Trusted Computer System Evaluation Criteria*, Department of Defense, DoD 5200.28-STD, December 1985.

10. Anderson J., *Computer Security Technology Planning Study*, ESD-TR-73-51, Vol. I, AD-758206, ESD/AFSC, Hansom AFB MA, October 1972.

11. Shockley W, Class Notes, CS4605, Naval Postgraduate School, Monterey CA., October 1996.

12. National Computer Security Center, *A Guide to Understanding Security Modeling Trusted Systems*, NCSC-TG-010, October 1992.

13. Graham G. S., and Denning P. J., "Protection-Principles and Practices", *Proceedings of the 1972 Spring Joint Computer Conference*, Montvale NJ., AFIPS Press, pp 417-429, 1972.

14. MITRE Report MTR 2547 Vol. 2, *Secure Computer Systems: Mathematical Foundations and Model*, by D. Bell and L. LaPadula, November 1973.

15. Denning D., "A Lattice Model of Secure Information Flow", *Communications of the ACM*, Volume 19, Number 3, May 1976.

16. Department of Trade and Industry, *Information Technology Security Evaluation Criteria (ITSEC)*, London 1991, Harmonized Criteria of France, Germany, the Netherlands, and the United Kingdom.

17. Canadian System Security Centre, *The Canadian Trusted Computer Product Evaluation Criteria (CTCPEC) Version 3.0e*, Communications Security Establishment, Government of Canada, January 1993.

18. NIST, *Common Criteria for IT Security Evaluation*, CCEB-96, January 1996.

19. Freund R., Hensgen D., and Kidd T., MSHN Proposal Document, to Defense Advanced Research Projects Agency, 1996.

20. IEEE Standards Department, *IEEE P1220 Standard for Systems Engineering*, New York, NY, 1994.

21. Birman K., *Building Secure and Reliable Network Applications*, Manning Publications CO, NewYork, 1996.

22. Shirley L., and Schell R., "Mechanism Sufficiency Validation by Assignment", *Symposium on Security and Privacy*, April 1981.

23. National Computer Security Center, *A Guide to understanding Discretionary Access Control in Trusted Systems*, NCSC-TG-003, September 1987.

24. National Computer Security Center, *Glossary of Computer Security Terms*, NCSC-TG-004, October 1988.

25. Russel D., and Gangemi G. T., *Computer Security Basics*, O'Reilly & Associates, Inc, July 1992.

26. Bell D. E., "Modeling the Multipolicy Machine", *Proceedings New Security Paradigms Workshop*, Little Compton, RI, pp 2-9, August 1994.

27. Saltzer J., and Schroeder M., "The Protection of Information in Computer Systems", *Proceedings of the IEEE*, Vol. 63 No.9, September 1975.

28. National Computer Security Center, *A Guide to Understanding Identification and Authentication in Trusted Systems*, NCSC-TG-017, September 1991.

29. National Computer Security Center, *A Guide to Understanding Audit in Trusted Systems*, NCSC-TG-001 Ver-2, June 1988.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
8725 John J. Kingman Rd., Ste 0944
Ft. Belvoir, VA 22060-6218

2. Dudley Knox Library 2
Naval Postgraduate School
411 Dyer Rd.
Monterey, CA 93943-5101

3. ECJ6-NP 1
HQUSEUCOM
Unit 30400 Box 1000
APO, AE 09128

4. Chairman, Code CS 2
Computer Science Department
Naval Postgraduate School
Monterey, CA 93943-5000

5. Dr. Cynthia Irvine 2
Computer Science Department Code CS/Ic
Naval Postgraduate School
Monterey, CA 93943-5000

6. Dr. Taylor Kidd 1
Computer Science Department Code CS/Kt
Naval Postgraduate School
Monterey, CA 93943-5000

7. LT John P. English, USNR 2
290 Adams Street
Milton, MA 02186

8. Dr. Blaine W. Burnham 1
R23
National Security Agency
9800 Savage Road
Fort George G. Meade, MD 20755-6000