1998

Security Architecture for a Virtual Heterogeneous Machine

Roger Wright; David J. Shifflett; Irvine, Cynthia E.

Proceedings of *Fourteenth Computer Security Applications Conference*, Phoenix, AZ. pp. 167--77, December 1998.

Security Architecture for a Virtual Heterogeneous Machine

Roger Wright
Computer Science Dept.
Naval Postgraduate School
Monterey, CA 93943
rewright@nps.navy.mil

David J. Shifflett
Rolands and Associates Corp.
500 Sloat Avenue
Monterey, CA 93940
dshifflett@gwdi.com

Cynthia E. Irvine
Computer Science Dept.
Naval Postgraduate School
Monterey, CA 93943
irvine@cs.nps.navy.mil

Abstract

We describe security for a virtual heterogeneous machine (VHM). Our security architecture is based upon separation of services into four distinct domains. It is designed to take advantage of operating system support for domains, where available. We have chosen to use emerging public key technology as an interim solution to provide domain separation. A prototype demonstration of our architecture has been developed.

1. Introduction

Science, engineering and defense are witnessing the emergence of a new approach to computation. It depends not on single isolated centers of excellence for high performance computing, but upon a network of computers bound together by an overlying framework that presents to users a powerful virtual heterogeneous machine (VHM) intended to support high throughput computing [19]. Applications may consist of a single thread of execution, a set of sequential jobs, or may be parallelized across several components of the underlying system. Some applications may require a variety of computational and storage resources ranging from desktop PCs for the submission of jobs to supercomputers for computationally intensive tasks.

Efforts are currently underway to define the VHMs which will manage user tasks. A major research area for these systems is the formulation of adaptive scheduling techniques based on task and network properties. Ultimately, the technologies emerging from the creation of an advanced computational infrastructure will be integrated into systems supporting finance, industry, and individual users. In all cases, security is a concern for these systems.

The diversity of the user population employing a VHM implies that one security solution may not be appropriate for all individuals or even for all tasks of one particular individual. We are collaborating in a research project to investigate the properties and services of a VHM and are examining security concerns and technologies in that context. This paper is a report on the progress of that effort.

We will describe how notions of privilege and system integrity can be incorporated into an architecture intended to execute in a heterogeneous computing environment. The remainder of the paper is organized as follows. Section 2. outlines the objectives and high level architecture of the VHM to which our protection architecture is being applied. Section 3. describes the security requirements for the system. Section 4. describes the security architecture we have developed for the VHM. Our prototype implementation and its use of the Intel Common Data Security Architecture, is discussed in section 5. Section 6. will contrast our work with that of several other VHM security efforts and will describe our future plans. Section 7. provides a summary.

2. Management System for Heterogeneous Networks

The intent of the Management System for Heterogeneous Networks (MSHN, pronounced "mission'") is to construct a virtual heterogeneous machine designed to run as an application on a variety of operating systems and hardware platforms [11]. It will provide end-to-end support for applications in distributed and heterogeneous, shared environments. In addition to supporting compute-intensive jobs, MSHN is intended to provide a responsive and flexible execution environment for real-time, interactive, and I/O-intensive tasks. When heterogeneous resources are

This work was supported in part by the DARPA/ITO Quorum Program.

The views expressed in this paper are those of the authors and do not reţect the of Ţcial policy or position of the Department of Defense or the U.S. Government.

simultaneously shared by several applications, each with its own unique quality of service (QoS) requirement, MSHN will be able to efficiently assign resources to the applications so that they will attain their requested QoS.

Each *job* is characterized by code and a dataset. An instance of a job will be the code and a particular dataset and is associated with a particular user. For each job, MSHN will maintain historical data describing the job's *compute characteristics*. When a new request to execute the job with a particular dataset is issued, the scheduler will use the job's compute characteristics as well as dynamic information about network and resource loading. The scheduler will make decisions regarding resource allocation to maximize the QoS for a particular job. For example, suppose job A has historically run in 10 minutes on resource X and in 20 minutes on resource Y. When a request to execute the job is issued the scheduler may decide to wait five minutes so that the job can be run on system X rather than schedule it immediately on system Y. In an open environment, even when using the same resource, consistent QoS for jobs may not be guaranteed. Network congestion and load at the compute resource can impact job performance.

Use of MSHN is optional. In theory, users could attempt to decide where a job might best be executed, however, several projects [14][5] have demonstrated that a scheduling and job management service can provide significant benefits, just as traditional operating systems support applications through local resource management services.

These early resource management system demonstrations lacked the ability to address quality of service requirements in a highly distributed and dynamic heterogeneous environment. MSHN will address these issues. It will monitor the load on a large pool of resources and will advise jobs as to the best strategy for achieving QoS objectives for a particular run. It is anticipated that, despite the extra processing required for MSHN to provide monitoring and advisory services, the performance advantages will outweigh the slight computational tax imposed by the VHM. User involvement in the scheduling and execution of jobs might be streamlined through the use of a special MSHN shell that would hide MSHN processing.

2.1 MSHN VHM

The proposed MSHN VHM [15] consists of five major components, as depicted in Figure 1: a user client, three MSHN core services, and a client on each computational server. (The Audit Server will be discussed in a subsequent section.) An assumption of the MSHN architecture is that the physical machines running MSHN components are not necessarily dedicated to MSHN processing. Thus, jobs on a computational server may be divided into two categories: those taking advantage of the services provided by MSHN and those that do not.

Client libraries will "wrap" user jobs at the application origination points and at the compute resources. These libraries are intended to provide the services of the MSHN VHM transparently: legacy application code will require no modification to be executed in the MSHN environment. Adaptive applications will benefit from environmental information that may be signalled to them through the MSHN services.

The Resource Requirements Database is used to store initial estimates of per job requirements and to maintain a record of the resources that have historically been consumed by a particular job. This may include the level of service requested by the users as well as priorities for particular classes of applications. As experiential data are accumulated, the resource requirements for the execution of future instances of the job are adjusted.

The Resource Status Server is the most dynamic of those maintained by MSHN. Here information regarding the resource consumption by all currently executing jobs on the VHM is maintained. In addition, client library monitors associated with the network and computational resources provide up-to-date information regarding the availability of resources on the VHM.

The Scheduling Server maintains a set of algorithms used to produce advisory schedules for jobs. If the Resource Status Server flags a job as being bogged down, a request may be issued to the scheduler to seek a better scheduling strategy for the task. That strategy may include task checkpointing and relocation [16].

A MSHN project goal is to provide support not only for traditional static applications, but to enhance the performance characteristics of dynamic and adaptive applications. This is particularly the case in environments where the load on resources is not predictable, but perhaps best characterized probabilistically. Consider, for example, an adaptive application in which a user asks for time-critical processing that will return high precision and high fidelity results. As the application is executed, MSHN will use the client libraries associated with all jobs on the VHM to monitor the resources of the VHM, quantifying and accounting for uncertainty throughout the distributed processing mechanism. Intermediate reports on job status and resource usage are transmitted

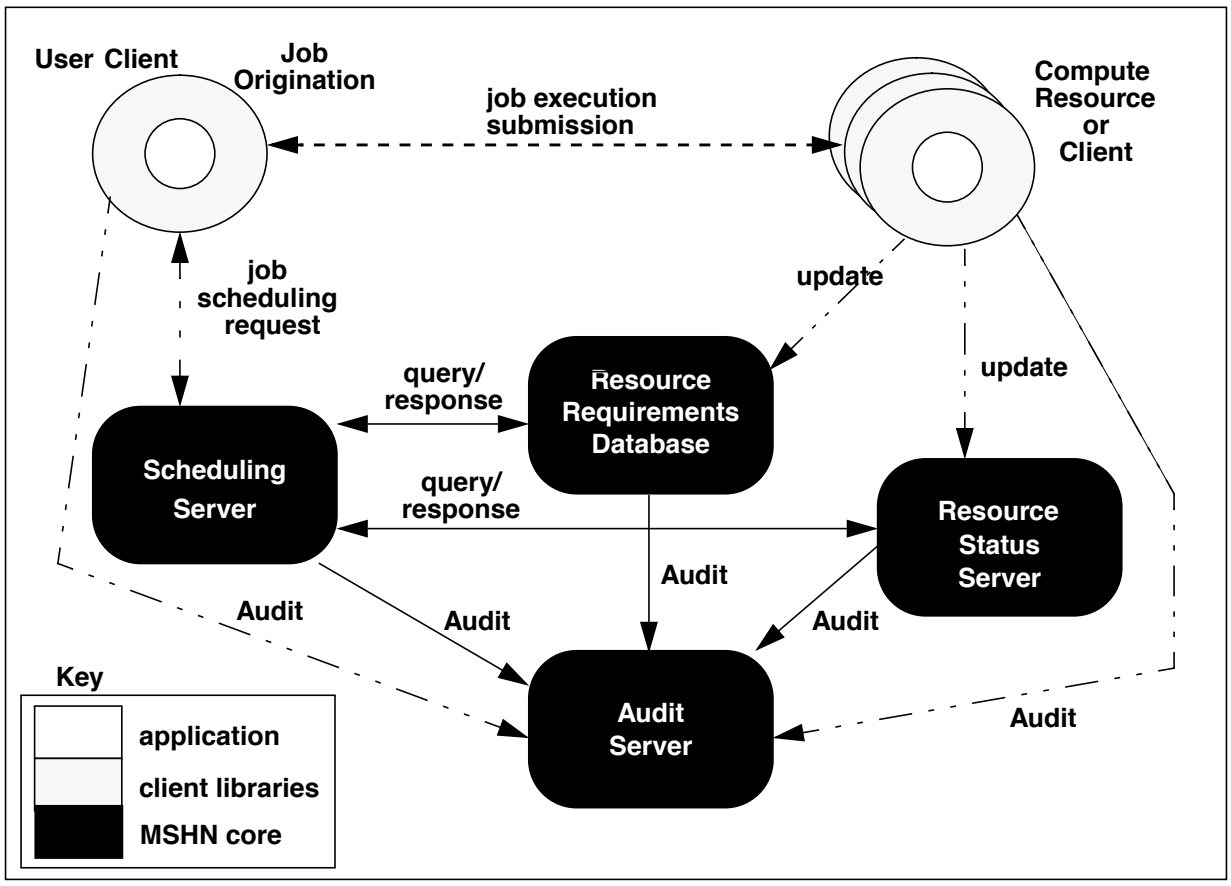

Figure 1. MSHN Architecture

by the client libraries to the Resource Status Server. Should available resources be insufficient to achieve the original time, precision and fidelity objectives, the MSHN Scheduler will trigger a notification to the client library of the affected application. In the case of an adaptive application, the application can adapt to the changed environment, thus satisfying modified, and realistically achievable QoS objectives. Special client libraries could provide adaptivity services to legacy applications. Upon completion of the task, results are returned to the user, while a summary of resource usage by the job is recorded in the Resource Requirements Database.

MSHN jobs consist of both code and the data set to be processed by that code. The need to define the job both in terms of its code and data set, or at least a characterization of the data set, arises from quality of service objectives for each job. Consider for example, code that will compute Fast Fourier Transforms (FFTs) on seismic data. Clearly the granularity and duration of the data will affect the amount of processing required. Hence, despite the fact that the code applied to both a short event and a long one is the same, the jobs differ in terms of the computational resources they will consume. As noted above, a variety of QoS requirements may also be associated with a job. Of course, a group of security requirements may also be bound to an instance of a job as a result of the particular code or data involved. A detailed discussion of application security requirements will be given in the next section.

3. Security Requirements

The MSHN security mechanisms are intended for an environment in which users elect to use MSHN services to enhance the quality of service they receive on their jobs. Users request advisory schedules from MSHN core services. When the schedule is returned, jobs are submitted to compute resources under user control and executed in the context of user accounts. Dynamic and summary job status information is reported to MSHN core services via client libraries which "wrap" the user's unaltered job. During job

execution, the MSHN scheduler may send revised scheduling information to the client libraries. Under certain conditions, a job may be moved to a different compute resource. If the job is adaptive, information relayed by the MSHN scheduler permits the job to modify its runtime characteristics in order to achieve desired quality of service goals.

This sketch of MSHN activity shows that the VHM depends upon communication between distributed hosts: those providing MSHN core services and those executing client libraries on behalf of users. In the remainder of this section we will present our approach to providing for the integrity of MSHN core components and client services within the networked environment. In addition, we will describe how the MSHN core supports security requirements such as integrity and confidentiality for the jobs it manages.

3.1 General Constraints and Assumptions

Certain MSHN design goals affect several choices regarding the security architecture. These are:

- Use of MSHN is optional. An advisory schedule obtained from the MSHN Scheduler is optional. A user may still choose to execute a job using MSHN clients at compute resources, but the compute resources selected by the user may not correspond to those suggested by the MSHN Scheduler.
- MSHN will not require modification of existing applications.
- MSHN will not require modification to the underlying operating system of any system it uses. This includes MSHN user clients, servers hosting MSHN core components, and MSHN services on compute resources.
- No MSHN component will have arbitrary control over any host. By this we mean that execution of MSHN will not require supervisory or, in Unix terminology, "root" privilege.
- It is assumed that users do not wish to subvert their own jobs with bogus run-time parameters, which would also result in corruption of the Resource Requirements Database. (Users can, of course, always elect to execute their jobs without the benefit of MSHN services.)
- User jobs are not required to reside permanently at compute resources. Code and data can either be bundled and sent to compute resources with the request for execution. Eventually, the MSHN client libraries are envisioned to be capabile of fetching code and data for a job from appropriate repositories.

Assumptions applicable to the security architecture are:

- A user will not attempt to corrupt his own jobs. If a user is concerned that his jobs will be corrupted by other jobs he may be executing on a compute resource, then the best remedy is to establish a separate principle for the execution of jobs managed by MSHN. The underlying operating system, through its mechanisms for the separation of user processes is expected to support this objective.
- Users will be required to log in when accessing compute resources, thus execution on resources will be by a principle representing the user rather than the MSHN core.
- For each user, MSHN will maintain a list of compute resources for which the user is authorized.
- MSHN will rely upon the existence of a public key infrastructure (PKI). The security architecture depends upon the existence and integrity of standard directory services such as X.500 or LDAP and the provision of standardized certificates such as those defined by X.509 v3. All elements of the MSHN architecture must have facilities to support certificate-based public key technology. Support for symmetric key cryptography is also required in most applications of public key cryptography.
- It is assumed that the MSHN client libraries do not contain malicious code.
- Compute resources will be known to the MSHN core. The list of resources will be administratively updated.

3.2 Policy

The security policy we wish to enforce for the MSHN components is as follows.

- Only authorized elements of the MSHN core services, such as the MSHN Scheduler, can query the Resource Status Server and the Resource Requirements Database for job status information. (Authentication and Access Control)
- Communications between the elements of the MSHN core services should be protected from unauthorized modification and disclosure. (Communications Integrity and Confidentiality)
- Status and summary information provided by client libraries to the Resource Status Server and Resource Requirements Database should be identifiable to the granularity of a particular run of a user job. This will prevent bogus updates to MSHN databases by malicious jobs outside of MSHN's purview. It will also permit the identification of rogue jobs which may have been corrupted and are misbehaving. (Authentication, Access Control and Denial of Service)
- Status and summary information provided by client

libraries to the MSHN core databases should be protected from unauthorized modification and disclosure while in transit. (Communications Integrity and Confidentiality)

- Dynamic scheduling advice provided by the MSHN Scheduler to client libraries supporting jobs at compute resources should be identifiable on a per job basis. It is not sufficient for the client library to know that scheduling information has, in fact, come from the MSHN Scheduler. In addition, the client library must be able to ascertain that the scheduling information is intended for the particular job being supported by the client library. (Authentication)
- Dynamic scheduling advice provided by the MSHN Scheduler to client libraries supporting jobs at compute resources should be protected from unauthorized modification and disclosure while in transit. (Communications Integrity and Confidentiality)
- Information maintained in MSHN core databases should be available only to authorized users. Some earlier resource management systems [14] had no notion of access control. The emphasis in these efforts was on scheduling and performance, not security, so everyone was privileged and had access to all job status information.

  A mechanism must be implemented to limit access to job status information. It is expected that there may be tension between limiting access to this information and providing statistically significant accumulations of runs to improve scheduling decisions.
- For each job, historical data on that job can be restricted to a set of authorized users. At first glance, this does not appear to be very hospitable. Why should historical information be restricted when it will help everyone run the job better? Suppose there is a job used for battle planning and that its execution statistics are stored in the Resource Requirements Database. Also suppose that an enemy wants to find out about this job. He could pretend to be submitting a request to run the battle planning job and would obtain from the scheduler advice on where to position his run. Even though he might not have a copy of the code and data, the historical data could yield information regarding the best systems to use for executing the job, duration of the run, and other exploitable information.

  Thus, in the interest of encouraging use of MSHN, a user can always request that MSHN provide a schedule for his instance of the job. Whether that schedule reflects historical information would depend upon the user's access to the historical data associated with other users' runs.

3.3 Accountability

Individual accountability is essential in any system intended to securely process information on behalf of users. Accountability permits liability to be established in the case of violations of policy. If the objective is to establish liability such that the perpetrator of an illicit act can be punished, then accountability must be sufficient to provide proof that the accused perpetrator is at fault.

Since MSHN is running as an application on operating systems that may or may not provide an adequate foundation for our security objectives, we must temper our accountability requirements. Corruption of an underlying operating system lacking penetration resistance could result in corruption of MSHN accountability mechanisms. It would be difficult to send someone to prison for violating MSHN security policy and it would be difficult to successfully litigate contractual agreements based upon the chain of assurance and accountability evidence provided by a VHM dependent on intrinsically weak platforms. So accountability in MSHN can act as a deterrent to abuse by relatively law abiding users, but it will not stop the determined opponent.

Accountability in MSHN will depend upon the identification and authentication of users, first at their local clients and subsequently to the MSHN core service and compute servers. Authentication and audit will provide the mechanisms to tie the user's identification to the system activities he has invoked and create a record of those activities considered security relevant.

Non-repudiation is described as a mechanism such that the sender of a communication is unable to deny that the message was sent and the receiver is unable to deny its receipt. The audit mechanism can serve as a trusted third party providing non-repudiation services for MSHN users. Clearly, a third party beyond MSHN is needed if non-repudiation services are required for transactions between MSHN and its users.

3.4 Assurance

Security enforced for MSHN components does not depend upon enforcement mechanisms built into MSHN itself, but instead upon underlying policy enforcement mechanisms that may be provided by operating systems and perhaps high assurance platforms. This is in concert with the philosophy [12] that more privileged mechanisms are

- more likely to embody the fundamental notions of the reference monitor concept [4] and

- more likely to have undergone scrutiny by qualified and unbiased personnel able to assess the assurance claims made about the mechanism.

The security architecture for the VHM will be designed to take advantage of mechanisms provided by more privileged underlying components. Our objective is to provide the intermediate interfaces necessary so that underlying medium to high assurance mechanisms can be used to protect communications associated with the VHM and user jobs as information is moved across unprotected networks.

We wish to prevent unauthorized updates to critical MSHN databases. Suppose that MSHN core components execute on dedicated servers. Then the likelihood of corruption of MSHN core databases by malicious applications is substantially reduced as is the possibility of exfiltraton of information by malicious software. In such an environment, the use of session keys provides a reliable mechanism to bind the components into a virtual domain.

How does the notion of privilege work in MSHN? Ideally, we would like to have each platform in the VHM provide a minimum of four protection domains: one for the user's application, one for MSHN client resource libraries and services, and one for MSHN protection services. Layered beneath these three domains would be more privileged domains associated with the underlying OS and its libraries. We envision certain essential public key technologies located in highly privileged domains. These include the key distribution centers and the certificate authorities.

4. MSHN Security Architecture

MSHN security is based upon the establishment of the following domains:

1. MSHN Core Domain
2. Client Library Domain
3. Application Domain

Each domain will authenticate itself to other domains using certificate-based technology. The basis for believing the certificates is the integrity provided at

Keys	Description	Key Type	Creator	User	Key Life	Purpose
K_m	MSHN Core Session Key	S	MSHN Core Master	MSHN Core Servers	per core instance	• high speed transmission of message traffic between core components
K_j	Job Session Key	S	MSHN Core	Client Library of job	per task	• identifies MSHN-relevant communications to MSHN Core • transmit audit records • authentication of per-job status information to MSHN Core
K_c	Client Library Session Key	S	MSHN Core	Client Library of Job	per task	• per-task intra-client domain management functions
K_a	Application Session Key	S	MSHN Core	Task Clients on behalf of application	per task	• application-level task communications • receive job execution results
$C_{pub/pri}$	MSHN Core Server Key	P	KDC or user	MSHN servers; one per server	KDC defined	• permits clients to authenticate communications to MSHN servers • core session key distribution • digital signatures
$R_{pub/pri}$	Computer Resource Client	P	KDC or server administrator	used by all MSHN compute resources	KDC defined	• receive job session key
$U_{pub/pri}$	User client key	P	KDC or user	Client Library	KDC defined	• receive job session key • user authentication

Table 1: Key Usage in the MSHN Security Architecture

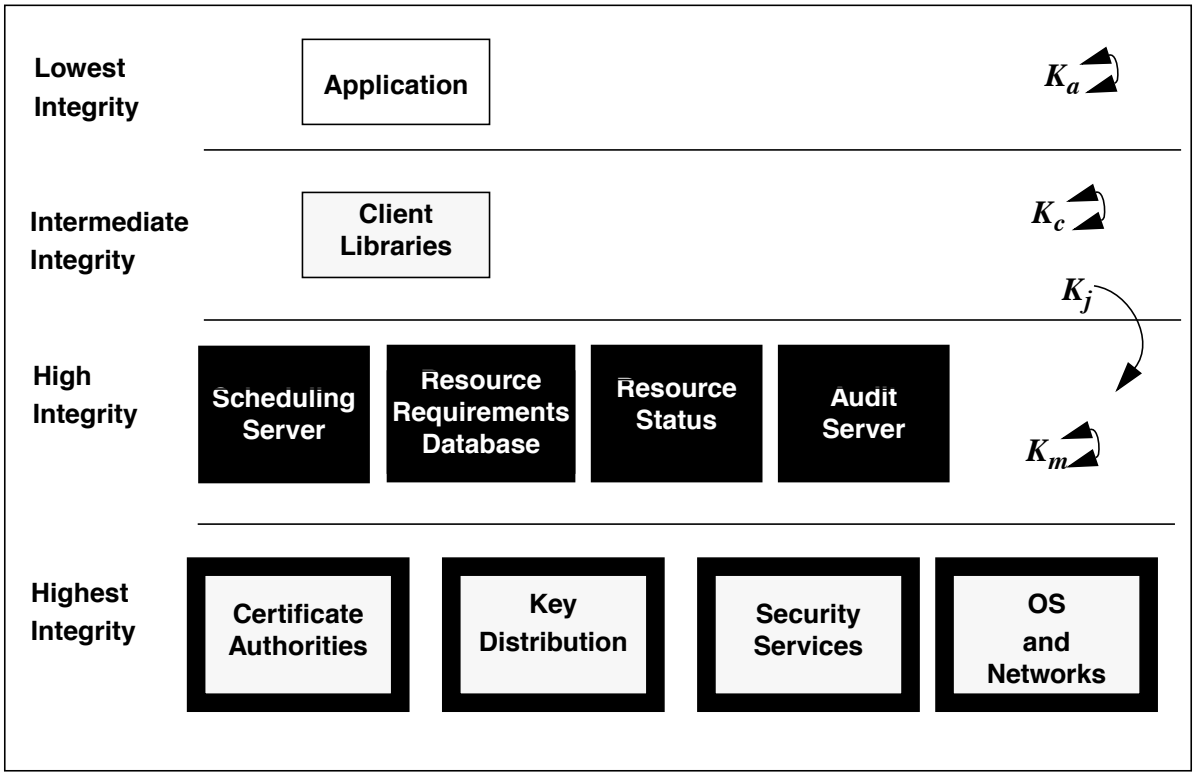

Figure 2. MSHN Security Domain Architecture

the certificate authorities, hence the certificate authorities are the most trusted components of our VHM security architecture. User tasks will be separated on a per-task basis using appropriate keys supplied to each domain. Within MSHN, the highest integrity domain, viz. the MSHN Core, will manage the creation and allocation of session keys used by lower integrity domains. Thus the client domains are issued per-task keys when a job is scheduled and these keys are used for dynamic task management and Core database updates. In addition to being isolated on a per-task basis, each domain is able to distinguish communications from members at its integrity from those of entities of greater or lesser integrity. The MSHN domains are illustrated in Figure 2. Policy is allocated to each of these domains and they are organized hierarchically. This is in keeping with notions of *balanced assurance* for distributed systems [13].

4.1 MSHN Core Domain

The MSHN core domain contains the basic MSHN core services: the Scheduling Server, the Resource Status Server, and the Resource Requirements Database. For the security architecture, we have added an Audit Server. The correct function of these servers is essential to insuring that jobs receive useful scheduling advice; the corruption of either their data or algorithms would render MSHN potentially useless or even detrimental to job performance and quality of service. Thus we assign to these system elements a high integrity level.

The MSHN Scheduling Server will make scheduling decisions based upon the conjunction of authorization information and other scheduling information. When a scheduling request is issued to MSHN, the Scheduling server will first determine the systems accessible by the user. This list of systems will be input to the scheduling algorithms and an advisory schedule will result. Thus for the same job and all other factors being equal, two users could have different schedules depending upon their access to compute resources.

4.2 Client Library Domain

The client libraries will occupy a domain of intermediate integrity. A copy of the client libraries will be available to each user's job. Although users might share the text of client libraries, viz, the client library code, when executing on the same machine, each instance will have separate data. Client libraries will also be able to securely communicate with client

libraries on other compute servers. This will permit coordination of parallel processing or checkpointing and movement of an application to a different compute server.

4.3 Application Domain

This domain is intended for user jobs. Each application will be distinct. Distributed, security-aware applications may contain their own end-to-end mechanisms or may use underlying security services.

A legacy application need not become "security aware" in order to be protected. When users request scheduling and process management services of MSHN, they may include protection services as part of the QoS specification for their jobs. The scheduling advisor will determine the best way to meet those protection needs within the VHM accessible by the user. Along with the scheduling recommendations returned to the user, algorithms and keys held by the underlying layers will be available to protect communications for distributed applications. We note that exposure of keys to applications is not intended: either keys are cryptographically protected or completely hidden by the underlying mechanism.

4.4 Mechanisms

Cryptographic mechanisms will be used to isolate and protect the elements of the MSHN architecture in "lightweight" domains. Table 1 describes the major keys used in our design. Key types are for symmetric, S, and public/private, P, cryptography. We intend to support updates of session keys during job runs. The frequency of these updates will be defined as part of the quality of service for security. It is expected that, as the public key infrastructure matures, key distribution centers (KDCs) will issue public/private key pairs. In the interim, individual systems may be required to act as their own KDCs.

4.4.1 Core Keys. The MSHN core will be assigned its own set of keys. Each MSHN core server will have a public/private key pair, $C_{pub/pri}$. These key pairs will be issued by a key distribution center (KDC) and can be veriTed by a certiTcate authority (CA) for inter-component authentication.

We believe that symmetric key cryptography provides the most efficient technique for the frequent communications that will be required between MSHN core components. Core components associated with a particular instance of MSHN share a MSHN-wide session key, K_m. This key permits the use of fast symmetric algorithms for the efficient transmission of core-relevant data.

A question that arose in our analysis was how to handle replicas of MSHN core services. This might occur in situations where the system is highly distributed and replicas of one or more MSHN core services are placed at strategic points in the network. Since it is desirable that a client be able to send updates to any one of the replicated servers of a particular type, e.g., Resource Status Server, with a consistent result across the entire system, the session keys of clients could be available to all status servers. Alternatively, dynamic key exchange algorithms could provide protected communications between core components.

4.4.2 Client and Application Keys. The MSHN Core will be responsible for issuing session keys for each job. These keys will be protected enroute to the compute resource using public key algorithms with $R_{pub/pri}$. There will be three session keys associated with each job: a Job Session Key, K_j, a Client Library Session Key, K_c, and an Application Session Key, K_a.

The Job Session Key will be used by the compute client to transmit status information back to the MSHN Resource Status Server and to the Audit Server. It will be used by the MSHN Scheduler to transmit advisory information dynamically to the user client and the compute client. For example, information regarding dynamic job adaptation can be transmitted to the compute client. A database tying user IDs to job IDs and session keys will be distributed across relevant MSHN core servers such as the Scheduling Server and the Audit Server.

The Client Library Session Key will be used for intra-domain management of clients associated with a particular job both at the user's platform and at the compute resources.

The Application Session Key is provided for the benefit of the application. It represents the keys and algorithms proposed by the MSHN core to meet the protection requirements specified by the user for the particular instance of the job run.

4.4.3 User Authentication. Individual users will have public/private keys, $U_{pub/pri}$. User's public keys will be stored in certiTcates which are available to all elements of the system through a directory service. The MSHN Scheduler will be able to authenticate the user using public key technology. The user will digitally sign messages using his private key and the scheduler will vali-

date the source of the signed message with the user's certiȚed public key.

5. Prototype Demonstration

A prototype demonstration of the MSHN security architecture has been implemented [20]. In keeping with the emphasis on commercial-off-the-shelf PCs embraced by the Navy's IT-21 initiative [7], our demonstration is implemented on three Windows NT Platforms. One platform served as the user interface from which jobs were submitted. A second platform hosted the MSHN Core database and scheduling services, each executing as a separate process. The last platform played the role of a compute resource. (Clearly, in a complete implementation compute resources might be high performance processors.)

For this demonstration we chose all certificates to be homogeneous and selected X.509 v3 certificates [6] for use throughout the VHM for authentication mechanisms.

5.1 Security Layer

One of our principle concerns has been the provision of security services within the context of a well engineered, modular architecture. Toward this end, we have defined a set of security services to be provided to MSHN core components and clients. This thin service layer allows us to achieve the support for heterogeneous platforms sought by the project. Also, it permits us to postpone API choices so that services presented by several APIs can be used, such as CDSA [3] and GSS-API [1]. Thus, a variety of security implementations can be used and these will be transparent to the VHM core, clients, and applications.

The security services layer consists of the following interfaces exported to MSHN core services and client libraries:

mshn_sl_init Initialize databases for MSHN security layer and create any necessary security contexts.

mshn_sl_create_cert Create a certificate with the specified parameters

mshn_sl_get_cert Obtain a certificate for the specified principle

mshn_sl_cert_verify Check that the specified certificate is valid

mshn_sl_cert_revoked Determine if the specified certificate has been revoked

mshn_sl_get_public_key Return the public key for the principle associated with the specified certificate

mshn_sl_get_private_key Return the private key for the specified subject. Note that, in the implementation, the private key is not actually returned, but a handle to the key is provided and this handle is used as a parameter to encryption or decryption functions. Thus the private key is never exposed.

mshn_sl_put_audit Write a record to the audit file or server

mshn_sl_encrypt Encrypt the specified data using the specified key and algorithm

mshn_sl_decrypt Decrypt the specified data using the specified key and algorithm

mshn_sl_sym_key_gen Create a key to be used with symmetric encryption and decryption.

mshn_sl_asym_key_gen Create a public/private key pair to be used with asymmetric encryption and decryption. Only information about the private key is exported, the key itself is not exposed.

mshn_sl_msg_digest Hash the specified buffers and generate a message digest.

mshn_sl_sign Produce a signature for the specified data in two steps: create a message digest and then encrypt the message digest.

mshn_sl_sig_verify Verify a digital signature for a specified data-signature pair.

5.2 Common Data Security Architecture

The Intel Common Data Security Architecture (CDSA) has been used as the basis for a proof of concept demonstration of the MSHN core services.

CDSA is intended to provide a basic security infrastructure for use with personal computers. It is a layered architecture and is intended to be modular, portable and adaptable. It has currently been implemented on Windows NT, a system of particular interest to the U.S. Navy as a result of its IT-21 initiative. An implementation of CDSA by RSA Security is underway and will provide the components on Unix platforms. In the interim, we have implemented a subset of the MSHN Security functions on a Unix platform using SSLeay encryption software[22] (a public implementation of SSL [9]).

CDSA consists of three primary layers:

- a set of system security services
- the Common Security Services Manager (CSSM)
- add-in security modules.

The add-in security modules are organized by the CSSM so that service provider interfaces (SPIs) can be defined for each module job manager.

Services are separated into four categories:

cryptographic services: An add-in cryptographic service module can provide the following functions

- bulk encryption and decryption
- creation and verification of digital signatures

- cryptographic hash creation
- key generation
- random number generation
- encrypted storage of private keys. We note here that ultimately the protection of keys, must depend upon a system-based protection mechanism.

trust policy services: The intent of the trust policy services is to provide access control over various system activities. Certificates are presented to the trust policy manager and access decisions are made according to information contained in the certificate.

certificate services: For a PC, the certificate services provide a self-contained certificate management mechanism. This module provides support to create new certificates, create certificate revocation lists (CRLs), sign and/or verify certificates and CRLs, import and export certificates created using other formats, extract information such as public keys from certificates, reinstate revoked certificates and search CRLs. The flexibility of these services with respect to certificate format makes them particularly attractive during the current period of evolving standards. For our current work, we have chosen the X.509 v3 format.

data store services: These modules are intended to provide secure and persistent storage for non-volatile information such as certificates and CRLs. Clearly in a system where assurance is of significant importance, data store services would be relegated to an underlying high assurance TCB and the access control mechanisms of the TCB would be used to ensure that only authorized users (or processes acting on behalf of those users) had modify access to security-critical databases.

6. Other Work and Future Plans

The Sigma project is addressing heterogeneity on the enclave level with its exploration of Internet Interoperability Protocol gateways to support Common Object Request Broker Architecture (CORBA) [2] interoperability [17]. This work is not directly integrated into a VHM management system and provides only a highly coarse level of access control and enclave protection. However, this effort could be viewed as complementary to ours as the MSHN security mechanisms could be used in an environment supporting Sigma controls.

Globus [8] is using a proxy-based system to permit computation on behalf of users on remote resources. The mechanism bears a remarkable similarity to Kerberos [18] and currently suffers from several scalability and trust domain drawbacks. Our approach provides a more straight forward path toward accountability.

Legion [21] claims to permit users to define policies at the granularity of objects. It is *ad hoc* with respect to privilege in that privilege is minimized everywhere -- no element of the system is responsible for any element other than itself. In addition, Legion does not provide for certificate authorities or key distribution centers. Instead, Legion-specific unique object identifiers are required globally to support key management. It appears that instead of supporting interoperability with respect to key management, Legion is dictating a homogeneous approach.

6.1 Future Work

Quality of service and security can sometimes be regarded as conflicting rather than complementary goals [10]. During crises, it is likely that demands upon a system may be significantly higher than during normal operations. This may mean that tasks may be required to adapt in order to provide a minimal level of service. We plan to explore the possibility of providing information to users and administrators regarding the performance costs incurred by using security measures commensurate with the sensitivity and reliability associated with the task. Hence the security requirements of different tasks can be met with measures of differing "strengths." At this juncture, we are not suggesting that security measures can be relaxed for a job with an assigned sensitivity and reliability.

Another area of future activity will be in the development of an audit mechanism for the VHM. As noted earlier, we plan to use a combination of audit and digital signatures as the cornerstone of non-repudiation services in our VHM.

Lacking a Unix implementation of CDSA, a Unix version of our prototype was created using SSLeay. In a fully layered architecture, SSL might depend upon CDSA for security support.

7. Summary

This paper has described a security architecture intended to support a virtual heterogeneous machine. A consistent set of choices must be made to construct data structures and mechanisms to use the underlying environment effectively. We build assurance into the system by relying on underlying mechanisms rather than constructing our own.

Our architecture for the MSHN VHM consists of four domains: the underlying operating sytsem, MSHN Core Services, Client Services, and Application. Domains start with their own identification and authentication evidence provided by key distribution

centers and certificate authorities. When the VHM is started, the Core Services establish their own cryptologically defined domain. Client Services submit job requests to the Core and are provided with session keys which establish per-session domains that permit communication between Clients and Core Services. Using underlying services, Clients create domains on behalf of Applications, to which they provide handles to keys for application-level communications security.

Future work will integrate quality of service mechanisms into this architecture and explore the creation of high assurance audit mechanisms.

Acknowledgements

We wish like to thank Timothy Levin for his critical reading of the manuscript and helpful comments.

References

[1] Global Security Services Application Programming Interface, RFC 2078.

[2] The Common Object Request Broker: Architecture Specification, December 1993.

[3] Common Data Security Architecture Specification, Intel Corporation, Santa Clara, CA release 1.2 edition February 1998. http://developer.intel.com/ial/security/.

[4] Anderson, J.P., Computer Security Technology Planning Study. Technical Report ESD-TR-73-51, Airforce Electronic Systems Division, Hanscom AFB, Bedford, MA, 1972. (Also available as Vol. I, DITCAD-758206, Vol. II DITCAD-772806).

[5] Bricker, A., Litzkow, M.,l and Livny, M., Condor Technical Summary, Technical Report CS-TR-92-1069, Computer Science Department, University of Wisconsin, Madison, WI, January 1992.

[6] CCITT. Recommendation X.509: The Directory - Authentication Framework, 1989. also ISO/IEC 9594-8.

[7] Clemmins, A., IT-21: The path to Information Superiority, CHIPS, July 1997.

[8] Foster, I., Karonis, N., Kellelman, C., Koenig, G., and Tuecke, S., A Secure Communications Infrastructure for High-Performance Distributed Computing, In *Proceedings 6th IEEE Symposium on Distributed Computing*, pages 125-136. IEEE Computer Society Press, 1997.

[9] Freier, A., Karlton, P., and Kocher, P., *The SSL Protocol, Version 3.0*, Netscape Communications, March 1996. URL: http://home.netscape.com/eng/ssl3/index.html6.

[10]Greenberg, I., Lunt, T., Clark, R., and Wells D., Secure Alpha: Security Policy and Policy Interpretation for a Class B3 Multilevel Secure Real-Time Distributed Operating System, Technical Report ELIN A003, SRI International, Menlo Park, CA, April 1993.

[11]Hensgen, D., and Kidd, T., MSHN design documents. note, February 1998.

[12]Irvine, C. E., Acheson, T.B., Thompson, M. F., Building Trust into a Multilevel File System, In *Proceedings 13th Nation Computer Security Conference*, pages 450-459, Washington, DC, October 1990.

[13]Irvine, C.E., Schell, R. R., and Thompson, M.F., Using TNI Concepts for the Near Term Use of High Assurance Database Management Systems. In *Proceedings Fourth RADC Database Security Workshop*, Little Compton, RI, April 1991.

[14]Kidd, T., Hensgen, D., Freund, R., Moore, L., SmartNet: A Scheduling Framework for Heterogeneous Computing, In *International Symposium On Parallel Architectures, Algorithms, and Networks*, Beijing, China, June 1996.

[15]Kresho, J.P., Quality Network Load Information Improves Performance of Adaptive Applications, M.S. thesis, Naval Postgraduate School, Monterey, CA September 1997.

[16]Litzkow, M., Tannenbaum, T., Basney, J. and Livny, M., Checkpoint and Migration of UNIX Processes in the Condor Distributed Processing System, Technical Report CS-TR-97-1346, Computer Science Department, University of Wisconsin, Madison, WI, April 1997.

[17]Sebes, J. and Benzel, T. C. V, SIGMA: Security for Distributed Object Interoperability Between Trusted and Untrusted Systems, Technical draft paper, Trusted Information Systems, Mountain View, CA 1996.

[18]Steiner, J., Neumann, B. and Schiller, J., Kerberos: An authentication System for Open Network Systems, Usenix Conference Proceedings, pages 191-202, 1998.

[19]Stevens, R., Woodward, P., DeFanti, T., and Catlett, C., From the I-Way to the National Technology Grid, Comm. A.C.M., 40(11):51-60, 1997.

[20]Wright, R., Integrity Architecture and Security Services Demonstration for Management System for Heterogeneous Networks, M.S. Thesis, Naval Postgraduate School., Monterey, CA, June 1998.

[21]Wulf, W., Wang, C., and Kienzle, D., A New Model of Security for Distributed Systems, UVa CS Technical Report CS-95-34, University of Virginia, Richmond, VA, August 1995.

[22]Young, E. A., SSLeay, June 1997. ftp://ftp.psy.uq.oz.au/pub/Crypto/SSL.