NPS-CS-00-004

NAVAL POSTGRADUATE SCHOOL
Monterey, California

The Effects of Security Choices and Limits in a Metacomputing Environment

by

Cynthia E. Irvine
Timothy E. Levin

January 2000

Approved for public release; distribution is unlimited.

Prepared for: Defense Advanced Research Project Agency
Information Technology Organization

NAVAL POSTGRADUATE SCHOOL
Monterey, California 93943-5000

RADM Robert C. Chaplin
Superintendent

R. Elster
Provost

This report was prepared as part of the Naval Postgraduate School Center for Information Systems Security (INFOSEC) Studies and Research (NPS CISR) at the Naval Postgraduate School, as part of a project funded under the Defense Advanced Research Projects Agency/Information Technology Organization grant under the Quorum program.

This report was prepared by:

Cynthia E. Irvine
Cynthia E. Irvine
Assistant Professor

Timothy E. Le
Timothy E. Levin
Senior Research Associate

Reviewed by:

Neil C. Rowe
Neil C. Rowe
Associate Professor
Department of Computer Science

DC Boger
D. C. Boger, Chair
Department of Computer Science

Released by:

D. W. Netzer
Associate Provost and
Dean of Research

REPORT DOCUMENTATION PAGE	Form approved OMB No 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** 31 January 2000	**3. REPORT TYPE AND DATES COVERED** Progress; 6/15/99 - 6/15/00

4. TITLE AND SUBTITLE The Effects of security Choices and Limits in a Metacomputing Environment	**5. FUNDING** MIPR No. 00-E583
6. AUTHOR(S) Cynthia E. Irvine and Timothy E. Levin	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Center for Information Systems Security Studies and Research (NPS CISR) Naval Postgraduate School, 833 Dyer Road, Monterey, CA 93943	**8. PERFORMING ORGANIZATION REPORT NUMBER** NPS-CS-00-004
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) DARPA/ITO 3701 North Fairfax Drive Arlington, VA 22203-1714	**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

11. SUPPLEMENTARY NOTES

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

13. ABSTRACT (Maximum 200 words.)

It is anticipated that the introduction of metacomputing and distributed resource management mechanisms to the Internet and World Wide Web will make available to users and applications a large diversity of previously unavailable network and computing resources. New methods of managing the scheduling and allocation of distributed resources bring into focus new problems and approaches for managing security in those contexts. We present an analysis layered and variable security services and requirements. These services and requirements may be accessed via a network control program such as a Resource Management System (RMS) which is responsible for scheduling resources in distributed heterogeneous environments. The RMS will not present the same "virtual computer/network" to the same job each time it is submitted for execution. Each instance will be comprised of potentially different actual resources with different properties. Our objective is to understand how user and application requirements, characterized as *choices* and *limit*s, can affect the overall security provided. A method is presented for fairly measuring the effectiveness of an RMS in performing security allocation and assignments with respect to security choices made by metacomputer users and applications.

14. SUBJECT TERMS resource management system, quality of security service, metacomputing	**15. NUMBER OF PAGES** 15
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** Unlimited

The Effects of Security Choices and Limits in a Metacomputing Environment

Cynthia E. Irvine
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943

Timothy Levin
Anteon Corp.
Monterey, CA 93940

Abstract

It is anticipated that the introduction of metacomputing and distributed resource management mechanisms to the Internet and World Wide Web will make available to users and applications a large diversity of previously unavailable network and computing resources. New methods of managing the scheduling and allocation of distributed resources bring into focus new problems and approaches for managing security in those contexts. We present an analysis of layered and variable security services and requirements. These services and requirements may be accessed via a network control program such as a Resource Management System (RMS) which is responsible for scheduling resources in distributed heterogeneous environments. The RMS will not present the same "virtual computer/network" to the same job each time it is submitted for execution. Each instance will be comprised of potentially different actual resources with different properties. Our objective is to understand how user and application requirements, characterized as choices and limits, can affect the overall security provided. A method is presented for fairly measuring the effectiveness of an RMS in performing security allocation and assignments with respect to security choices made by metacomputer users and applications. [1]

Keywords: Quality of Security Service, Resource Management System

1 Introduction: Managing Metacomputer Resource Allocation

Metacomputing provides users and applications with access to a virtual machine consisting of a wide range of distributed networking and computing resources (see e.g., [15]). Initially, efforts in metacomputing were focussed on providing transparent access to remote supercomputers for their user communities and support organizations. The advent of standardized protocols for (1) managing production and transmission of multimedia data [16], and (2) the more general distribution and execution of remote code (e.g. via the World Wide

[1] This work was supported under the MSHN Project of the DARPA/ITO Quorum Program

Web, Java, or Jini) may help to enable the vision of metacomputing to extend to devices and computational resources that are generally available on the Internet.

Whereas current distributed systems and internet technology may import mobile code for local execution (e.g., via Java applet), or request that remote code be executed in its native (fixed) environment (e.g., via servelets or object request brokers), metacomputing expands this paradigm to include execution of mobile code utilizing a wide range of possible remote resources. In some sense, recent Jini (Sun) and Universal Plug-and-Play (MS) technologies enable some metacomputing functions, but they may lack the ability to optimize multi-task scheduling or Quality of Service, or to adapt to changing resource availability.

The resources available on a metacomputing virtual machine are both local and remote; are implemented in hardware as well as software; and include processing, storage, and display devices. The heterogeneity [4], multiplicity and remoteness of these resources provides various management, scheduling and security challenges [5, 2]. Of specific concern for this paper is the fact that the metacomputer presents too many variables and choices for users or applications to manage without automated support.

Resource Management Systems (RMSs) are designed to provide efficient, automated management and allocation decisions for metacomputer resources [9, 8]. Allocation decisions involve matching requirements to capabilities and attributes for security [3], completion time, computational environment [12], network bandwidth, etc. The efficiency of an RMS in providing these type of decisions can be measured with respect to various user and system goals, for example, *quality of service* (QoS) specifications and system allocation policies [11, 16]. The relationship of the RMS to the metacomputer is shown in Figure 1, where P indicates compute or processor resources; N, network bandwidth; and D, data storage and data staging components.

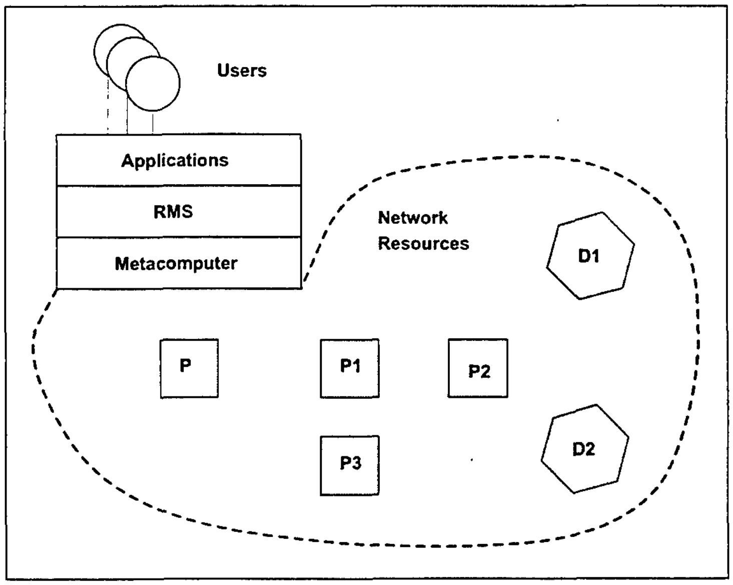

Figure 1: Metacomputing and the RMS

1.1 QoS Choices

Quality of Service refers to the ability of a system to provide services such that user expectations for timeliness and performance quality are met. For example, a multimedia application should deliver video frames so that the display is jitter-free [16, 6]. Quality of service can be provided at several levels within the overall system. The notion of translucence, where components can adapt to changing conditions at one or more other levels, results in a problem that is both horizontal, viz. distributed across the network; and vertical, viz. distributed within the stack. Finally quality of service requirements may change in systems supporting dynamic policies based upon current operating modes, e.g. normal, impacted or crisis [3].

Users have expectations with respect to the security services they are provided. These expectations may include both functional and assurance characteristics. With respect to security for a particular job, a user might require a minimum level of both functional mechanism and assurance. The ability of the network to meet these requirements is measured in terms of *Quality of Security Service* (QoSS) [3]. The notion of network Quality of Security Service expands the network service choices available through the metacomputer, providing administrators and users with more flexibility and potentially better service, without compromise of network and system security policies.

1.2 Security Ranges

As introduced in [3, 11], the security requirements presented to a network application can allow a *range* of security behavior. For example, a security policy for a hypothetical sub-network requires IP packet encryption. In this sub-net, a commercial multimedia application exports digital images (e.g., movies, or high-resolution fine art images). However, recognizing that the application in this specific environment can tolerate a media stream which is *periodically* encrypted (viz., one yielding a suitably obscured image, which would render a stolen image unsalable), the policy may only require that a range of from 80% to 100% of the packets should be encrypted. (Note that in some risk models, such a periodic encryption method might require fortified protection against cryptanalysis. In addition, care must be taken to ensure that in five repeated transmissions the entire unencrypted image is revealed.)

Collaborative applications, for which video teleconferencing with shared electronic white boards and application suites represent current technology, present another example in which security choices are available to the participants. Suppose that today one party in the group is located at organizational headquarters while another is a "road-warrior" participating from a hotel room in a foreign country known for government support of corporate espionage. Clearly the security requirements and choices of the road-warrior will be quite different than those chosen tomorrow when all participants will be in "friendly" territory. When a remote user is involved, collaborators may demand increased levels of both confidentiality and integrity support.

Consider the security administrator's or the user's motivation in agreeing to or specifying a range of security protection. As with multimedia image resolution, users will generally desire the greatest amount of security (or image fidelity) available, but this desire is tempered by cost. Cost may may take the form of monetary charges or performance degradation, for example. When cost is very high (e.g., slow image display), users may be willing to accept degraded security or imagry, instead.

Yet, once a user (or security officer) decides on the minimum level of security required for a given application, why would they ever agree to more security, if it increases their cost? For one, an application may have variable data formats, which may have correspondingly variable security requirements. A degraded image might require less security, and conversely, the enhanced image might be more security sensitive. To illustrate this example, a range of fidelity/security is shown in Table 1:

Table 1: **Security Ranges**

Fidelity	Security	Performance
high	high	low
medium	medium	medium
low	low	high

Another example is that the underlying system might support different situational modes. For some modes (e.g., "emergency"), the user or administrator may be willing to accept more (or less) security for a given application. The management of mode and security-level negotiation is handled automatically by some resource management systems [9].

Yet another scenario is that the underlying control program may have more flexibility to execute the job quickly, or at all, if the user can live with a range of security requirements. For example, transmission paths may go through a wide range of security environments. So the user specifies: do what you need to do, but give me at least "this much" security. The application might even execute faster with more security; regardless, the RMS manages the security allocation within the bounds specified by the user.

From the system's point of view, as opposed to that of the user, security variability provides another tradeoff factor, allowing the system to be more flexible in providing QoS for the system as a whole.

Here are some other examples of security ranges with examples of how the ranges could be characterized:

- strength of cryptographic algorithm
  - e.g., RSA, DES, etc., where strength might be measured in terms of the work factor associated with a brute force attack
- Length of cryptographic key
  - characterized by bit-length
- percentage of packets authenticated [14]

  - characterized by percentage of total (e.g., a multimedia environment might tolerate a percentage of data modification or loss)

- Security functions present in destination job-execution environment
  - characterized by operating system or boundary control security policy enforcement mechanisms
- Confidence of policy-enforcement in remote login environment
  - characterized by 3rd-party evaluation
- robustness of authentication mechanism
  - here the range might span weak password, strong password, biometric, and smart cards with on-board display and input interfaces

As one last example, consider a network consisting of various subnets. One of these subnets could be known to be toxic to the interests of the host enterprise, as in a subnet of nodes within a hostile country. Now, the toxic subnet could be identified by ID or by a security rating, and application or enterprise policies could prohibit routing through, execution within, or logon to such a subnet, by specifying allowed or disallowed sets of subnets.

1.3 Goal: Effective Security with regard to QoS Choices

As stated above, network services may allow security ranges. These ranges provide the RMS with additional variables to consider in scheduling and balancing its various requirements. The RMS may allow users and applications to indicate a choice or preference within any of the security ranges. As with other QoS factors, the RMS may modify the level of security service within this range in order to balance other factors, e.g., completion time.

We desire to be able to measure the level of security provided by the RMS in managing the tasks and resources for which it is responsible. Our general approach will be to summarize the level of security service supplied across all scheduled tasks. This metric should give maximum credit to the RMS when it maximizes the security provided to the overall network (i.e., the sum of the network applications). Additionally, we would like to factor user and application security choices into the RMS measurement model so that, if the user asks for, and is provided, less security service than the maximum for a given range, the RMS is not penalized with respect to the metric.

The rest of this paper provides a description of how user and application choices in the context of QoSS can be understood to affect the overall service provided by an RMS. A set of definitions and conceptual framework for reasoning about the QoSS problem is introduced in Section 2. A more formal presentation of choices and restrictions within the context of an RMS is presented in Section 3. Finally, our conclusions and future work are found in Section 4.

2 Network System Model

In order to reason about the defined problem we will first establish a definitional framework.

2.1 Security Resources, Services and Requirements

A network *system* is the infrastructure consisting of the totality of network-accessible resources and security services. A *security service* is a high-level abstract resource providing security functionality such as: authentication, auditing, privacy, integrity, intrusion detection, non-repudiation, and traffic flow confidentiality [3]. A security service typically consumes other low-level system resources such CPU, memory, disk, and network bandwidth. For example, the Common Data Security Architecture (CDSA) [10, 13] describes modules each of which contain specific security mechanisms to provide some of these services.

Each such resource and service may embody security *requirements.* A requirement may *restrict* the availability of a resource to an external entity. Some restrictions might be the typical MAC and DAC requirements, or other security constraints, e.g.: encryption available 9 P.M. to 5 A.M., range of available encryption algorithms, and range of required key lengths.

An implemented security mechanism functions as either a service or a requirement. For example, suppose that a user wishes to access a database within a protected subnet. To access this subnet all packets must be digitally signed using HMAC-SHA. The need to digitally sign all packets entering the subnet is a *requirement* imposed in order to access the database. The user may choose to not access the database because of this imposed requirement. In contrast, if a user has a requirement for data authenticity, he will choose to access the database resource within the protected subnet and will choose to use a local packet authentication *service* in order to transmit packets with the required signatures.

2.2 Variant Security

To be general, we will define that all security requirements have a *range* of permissible behavior. That is, a range may be *unitary*, or degenerate, in which case it represents no choice. Where a range represents a choice, the requirement is termed *security variant.* All system security services are security variant: since they are invoked at the discretion of the user or application, the range is at least binary (i.e, invoked or not invoked). Some requirements are unitary, while others are variant.

2.3 Task Sequences

In the theory of metacomputing, applications may be broken up into subtasks each of which may be executed on different topographical network elements, the results of which are in some way logically joined by the metacomputer [1]. Depending on the metacomputing mechanism used, the topographical structure and the location of specific elements may be more or less transparent to the end user. For the work discussed in this paper, we make the simplifying assumption that a *task* is an application invoked by a user, and each such distributed subtask (if present) is a logically separate *task*. The task utilizes various network system services and resources. The utilization is intermediated by the RMS. Thus, a task is invoked in a sequence:

- the user activates the application through some interface with an application manager (OS, browser, etc.);
- the application is intermediated by the RMS; and
- the RMS submits the application to the system.

We call this the *task invocation sequence:*

$$user \Rightarrow application \Rightarrow RMS \Rightarrow system^2$$

Security requirements may be established or refined by any or all of: the user, the application, the RMS, and the system. We designate these entities as *security requirement providers.*

As an example of how a requirement can be refined within the task invocation sequence, consider how a typical application offers the user a choice for some service. If the user does not indicate a choice, the application uses some default value. If the user chooses a range, the application invokes itself with a particular value within that range (the application's choices may be managed by a handler or wrapper). Similarly, the RMS may refine the application's choice, for example, to optimize metacomputer performance, load balancing, etc.

In a task invocation sequence, the request is passed from left to right, from a *previous* requirement provider, and to the *next* provider. A security choice for each variant security requirement is logically included with each request step. The choice may be implicit or explicit. For example, if no explicit choice is made, then it may be implicit that the choice is to not limit or modify the security options proffered at that step.

2.4 Security Limits and Choices

Each requirement provider may specify a choice range for each variant requirement in a given task invocation. For example, the user selects a range of 50 - 80% for packet authentication rate. This choice is passed to the next provider (viz., the application) in the sequence. Additionally, each requirement provider may have a *requirement limit range* outside of which it will not accept a request. The limit applies to the request choice from the previous provider, e.g., a given application will not accept a range wider than 60 - 100% from the user. We consider this *limit* to be valid only if the provider enforces it.

[2] Note that it is an implementation detail whether the RMS returns advisory parameters to the application and the application *invokes* the system, or the RMS submits the application with those parameters directly to the system. For simplicity, we assume, here, that the RMS submits the application to the system.

2.5 Range Relationships Inherent in Task Sequences

Table 2 shows the various limits and choices we have identified for security requirement providers.

Table 2: **Security Limits and Choices**

	User	Task	RMS	System
Choice Range	requirement	requirement	requiremnt	Service Level
Limit Range	n/a	requirement	requirement	requirement

Notice that the user does not have an effective limit range, as he has no *previous provider* upon whom to enforce such a range. Also, the system choice range is the level of service ultimately provided by the system in response to the request. This is a unitary range, since there is no *next* provider to whom a choice might be given.

The question arises as to how these ranges relate to each other. We present the following relationships as intuitively inherent in task sequences. However, it is not clear that these relationships are (precisely) necessary or sufficient for that purpose; rather, they are provided to explore the semantics of security ranges in task sequences.

- Each provider's choice must be within its own limit.

  This restriction reflects the natural semantics of choices and limits, in that it is natural to respect one's own limits.

- Each choice must be within the previous choice in the sequence

  This reflects a natural protocol to respect the choice of the previous requirement provider: a requirement provider will try to fulfill the request of a previous provider. For example in a quality of service context, a service provider may accept a request if it can be realized, but it will not proceed with wholly divergent parameters.

- Each choice must be within the next limit in the sequence

  This relation is the consequence of our definition that limits are enforced by their providers. This restriction intuitively means that requests which are out of bounds will be rejected.

- The limits of each provider in a task sequence must all intersect

  This is a consequence of the need for a choice to be within its own limit, and within the next limit, as well as within the previous choice.

2.6 Further Concepts of Operation

Requirement choices and limits may be *unitary* or *nil*. *Unitary* means only one value (i.e., no choice) is passed to the next requirement provider in the task sequence. A *nil* range denotes no restriction is imposed in the task sequence, and would likely denote acceptance of the previous choice range, such that the previous choice is transparently passed through to the next provider.

3 Formal Representation

In this section we present a formal representation of the framework developed in Section 2.

3.1 Goals

The purpose of formalizing the task sequence model introduced above is to *precisely* characterize security choices for support of QoSS in metacomputing environments. Furthermore, we wish to provide *generality* such that the model does not limit designs and implementations of the basic concepts, and we want to provide *consistency* such that the model does not require self-contradictions in derived implementations.

3.2 Range Definitions and Operations

A *range* is a set of elements which defines the possible choices of a variant security requirement. More than just a set, the elements of a range are related, because some are more secure than others. We will use the operator $\geq$ (*dominates*) to partially order the elements of a range with respect to relative security. The dual of the $\geq$ operator is the $\leq$ operator.

These are some example orderings based on such an operator:

- $BlackNet \geq RedNet \geq WWW$
  - Since $BlackNet \geq RedNet \geq WWW$, we see that subsets of a network may be partially ordered by set inclusion.
- % of packets encrypted
  - This is a linear partial ordering based on numeric value.
- $3DES \geq DES \geq caesar\_cipher$
  - The strength of encryption algorithms are ordered by crypto-analytic work factor.

The functions max and min are universal upper and lower bounds on a range:

$$\forall e : element(e \in r \rightarrow max(r) \geq e)$$
$$\forall e : element(e \in r \rightarrow min(r) \leq e)$$

The *enclosure* operator $\gg$ means range r *encloses* range s, as follows:

$$r \gg s \dashrightarrow max(r) \geq max(s)\; and\, min(s) \geq min(r)$$

The *intersection* operator $\wedge$ means range r and range s intersect:

$$r \wedge s \dashrightarrow max(r) \geq min(s)\; and\, max(s) \geq min(r)$$

The *sequence* function seq(p) provides the following linear ordering on the security providers (p) of a task sequence:

$$seq(user) < seq(application) < seq(RMS) < seq(system)$$

We also represent the set of security requirement providers:

$$\{u, a, r, s\}$$

This set represents (respectively) the user, application, RMS and system.

To represent the security requirement provider choices and limits discussed above, we introduce the structure **b** in Table 3. **b** consists of two subvectors: one for choice ranges (**b.c**) and one for limit ranges (**b.l**).

Table 3: **Choices and Limits Represented in Structure b**

Entity	Choice	Limit
User	**b.c.u** - user choice range	**b.l.u**- no user limit
Appln	**b.c.a** - application choice range	**b.l.a** - application limit range
RMS	**b.c.r** - RMS choice range	**b.l.r** - RMS limit range
System	**b.c.s** - system response	**b.l.s** - system limit range

3.3 Expression of System Security

Previous work has provided an expression for security requirements in a network environment [3]. Briefly, a security vector **S** represents the security requirements involving a task executing in a network environment. A security vector component, **S.component**, contains a boolean statement regarding security requirements for a given service or resource.

Examples of security vector components are:

$$\begin{aligned}
\textbf{S.a} &= \text{level (user)} \geq \text{level(resource)}\\
\textbf{S.b} &= \text{length of confidentiality encryption key} \geq 64, \leq 256;\ inc\ 64\\
\textbf{S.c} &= \text{\% packets authenticated} \geq 50, \leq 90,\ inc\ 10\\
\textbf{S.d} &= \text{authentication header transform in } \{\text{HMAC-MD5, HMAC-SHA}\}
\end{aligned}$$

Each **S.component** has at most one variant requirement. Requirements of a given security service may span several vector components (indicating a service *sub-vector*).

In **S.c**, "inc 10" indicates that the range from 50 through 90 is quantized into increments of 10, viz: 50, 60, 70, 80, 90. Later, we will need to indicate the number of quantized steps in the component; to do this, one more notational element is introduced, $|\ \mathbf{S.c}\ |$. In the above examples, $|\ \mathbf{S.a}\ |= 1$, and $|\ \mathbf{S.c}\ |= 5$.

$$|\ \mathbf{S.c}\ | \quad = \quad \text{number of quanta in } \mathbf{S.c}$$

When $|\ \mathbf{S.c}\ |> 1$, the underlying control program has a $range$ within which it may allow the task to execute with respect to the policy requirement. This range corresponds to the ranges in the structure **b** discussed above, but to which range in **b** does **S.c** correspond? First, we relate a **b** structure to each security vector component **S.c**, as follows:

$$\mathbf{S.c} \models \mathbf{b}$$

We wish to measure the effectiveness of the RMS decision/management strategy as reflected in the ultimate system choices (b.c.s), but we need to provide a measurement "yardstick." The effects of providers who are earlier than the RMS in the task invocation sequence

$$user \Rightarrow application \Rightarrow RMS \Rightarrow system$$

determine the reduced/restricted requirements perceived by the RMS. That is to say, the user and the application may narrow the requirement range such that the requirements are less restrictive[3] than the system maximum, (max(**b.l.s**)) and more restrictive than the system minimum (min(**b.l.s**)), so we will use the reduction immediately before **b.c.r** (i.e., **b.c.a**) as the security metric against which RMS effectiveness is to be measured: the value of the requirement **S.c** is defined to be $\mathbf{S.c} \models \mathbf{b.c.a}$.

3.3.1 Expression of Range Relationships

We can now restate the range relationships described previously, using S and b.

```
$\forall$ $S$:security_vector, c:service_component(
    /* each provider's choice must be within its own limit */
    $\forall$ $p$:provider (
        $S.c \models b.l.p \gg S.c \models b.c.p$)
    & $\forall p_1, p_2$:provider(
        /* each choice must be within the previous choice in the sequence */
        $seq(p_1) < seq(p_2) \rightarrow S.c \models b.c.p_1 \gg S.c \models b.c.p_2$
        /* each choice must be within next limit in the sequence */
        & $seq(p_1) < seq(p_2) \rightarrow S.c \models b.l.p_2 \gg S.c \models b.c.p_1$
```

[3] Here more restrictive means more security.

/* all limit ranges intersect */
& $S.c \models b.l.p_1 \wedge S.c \models b.l.p_2$))

3.3.2 Expression of Effective Security

To see how effectively the RMS applies variant security to tasks, we need to have an expression for the level or difficulty of security "met" by a task invocation, with respect to the required security range.

From above, $|< range >|$ is the number of quanta in "range."

Let $< response >:< range >$ indicate the ordinal of the quantum in a "range" achieved by a system "response." For example, if a range was from 3 to 12 in increments of 3 then $\mid (3,6,9,12) \mid = 4$. The ordinal of the quantum for 6 is 2, i.e. $6 : (3,6,9,12) = 2$, and the ordinal of the quantum for 9 is 3, i.e. $9 : (3,6,9,12) = 3$.

Then we associate a token $(g.c)$ with each c in S, and define $g.c$ to be the $fraction$ of the required security met by a task invocation:

$$g.c = \frac{< response >:< range >}{|< range >|}$$

We know that

$$requirement = \mathbf{b.c.a} \text{ (security choice of the application)}$$

and

$$met = \mathbf{b.c.s} \text{ (service level provided by system)}$$

So, for example, given **S.c** such that **S.c** $\models$ **b** is defined as follows ...

b.l.s	-system limit range	=	% packets authenticated $\geq 50, \leq 90$, *inc* 10
b.c.u	-user choice range	=	% packets authenticated $\geq 50, \leq 90$
b.c.a	-appl'n choice range	=	% packets authenticated $\geq 50, \leq 80$
b.c.r	-RMS choice range	=	% packets authenticated $\geq 50, \leq 70$
b.c.s	-system response	=	% packets authenticated $= 70$

(Here, 70 is the third quantum in a range that spans from 50 to 80 in increments of 10.) then we can state:

$$g.c = \frac{b.c.s : b.c.a}{\mid b.c.a \mid} = \frac{3}{4} = 0.75$$

Notice that $g.c = (0 \; or \; 1)$ for invariant components.

With g.c in hand, we introduce a function (A) which averages the tokens of a task [3]:

$$A = \frac{(g_1 + g_2 + .. + g_n)}{n}$$

where n equals the number of components in S. (Optionally, one could add weight coefficients to each g to indicate relative importance with respect to the security policy.)

Since an RMS manages the execution of many tasks over a period of time, we wish to examine its effectiveness with respect to a group of tasks. If A is derived against n tasks, then A_S is an expression for the overall effective security delivered by the RMS to the n tasks:

$$A_S = \frac{\sum_{j=1}^{n} A_j}{n}$$

$0 \leq A_S \leq 1$, where 1 indicates the maximum scheduling effectiveness.

4 Conclusion

We have presented a framework for analysis of security with respect to Quality of Security Service, user security choices, and the "task invocation sequence" used in a metacomputer scheduling mechanism (e.g., an "RMS"). This framework characterizes the relationships of security requirements of users, applications, RMSs, as well as the underlying security resources and services. We then presented a formalization of the framework, and extended it to include an expression for the efficiency of an RMS with respect to security, which tends to the maximum as security "met" approaches the maximum of the application's choice.

4.1 Future Work

We are currently working on several aspects of security regarding the characterization of RMS security, security choices, and benefit functions. We are also working on incorporating *variant* security costing techniques into a research prototype of an RMS [9, 3, 17].

We would like to understand the completeness of the range relationships identified for task invocation sequences.

We are working to better understand the nature of contradictions which might exist within or between security vector components. For example, does there exist a normal form of S whose derivation discovers or eliminates contradictions? One such contradiction involves a security component whose provider limits (e.g., application limit and RMS limit) do not intersect; another involves two interdependent components of a security vector (e.g., a communication authentication service which utilizes a digital integrity service), whose limits for a given requirement (e.g., the required ranges for encryption key lengths) do not intersect.

We would like to incorporate into the benefit function the effects of parallel and redundant security mechanisms [7].

References

[1] Ammar Alhusaini, Viktor K. Prasanna, and C.S. Raghavendra. A Unified Resource Scheduling Framework for Heterogeneous Computing Environments. In *Proceedings of the Heterogeneous Computing Workshop*, pages 156–165, San Juan, PR, April 1999. IEEE Computer Society Press.

[2] M. Blaze, J. Feigenbaum, and J. Lacy. Decentralized trust management. In *Proceedings 1996 IEEE Symposium on Security and Privacy*, pages 164–173, Oakland, CA, May 1996. IEEE Computer Society Press.

[3] blind. This reference is blinded for peer review, January 3000.

[4] Tracy D. Braun, Muthucumaru Maheswaran, Howard Jay Siegel, Noah Beck, Ladislau Boloni, Albert I. Reuther, James P. Robertson, Mitchell D. Theys, , and Bin Yao. A Taxonomy for Describing Matching and Scheduling Heuristic for Mixed-Machine Heterogeneous Computing Systems. In *Workshop on Advances in Parallel and Distributed Systems (in proceedings of the IEEE Symposium on Reliable Distributed Systems)*, pages 330–335, West Lafayette, IN, October 1998.

[5] Paul Carff. When is a simple model adequate for use in scheduling in mshn? Master's thesis, Naval Postgraduate School, Monterey, CA, March 1999.

[6] S. Chatterjee, B. Sabata, and J. Sydir. Erdos qos architecture. Technical Report ITAD-1667-TR-98-075, SRI International, Menlo Park, CA, May 1998.

[7] Saurav Chatterjee and Michael Brown. A graceful adaptation scheme for secure multimedia traffic. Technical Report ITAD-1667-PA-99-008, SRI International, Menlo Park, CA, 1999.

[8] I. Foster and C. Kesselman. Globus: A metacomputing ingrastructure toolkit. *International Journal of Supercomputer Applications*, 11(2):115–128, 1997.

[9] Debra Hensgen, Taylor Kidd, David St. John, Matthew C. Schnaidt, J.J. Siegel, Tracy Braun, Jong-Kook Kim, Shoukat Ali, Cynthia Irvine, Tim Levin, Victor Prasanna, Prashanth Bhat, Richard Freund, and Mike Gherrity. An Overview of the Management System for Heterogeneous Networks (MSHN). In *8th Workshop on Heterogeneous Computing systems*, pages 184–198, San Juan, PR, April 1999.

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

[11] Jong-Kook Kim, Debra Hensgen, Taylor Kidd, H. J. Siegel, David St. John, Cynthia Irvine, Timothy Levin, N.W. Porter, Victor Prasanna, and Richard Freund. A QoS Performance Measure Framework for Distributed Heterogeneous Networks. In *Proceedings of the 8th Euromicro Workshop on Parallel and distributed Processing*, pages 18–27, Rhodes, Greece, 2000.

[12] Muthucumaru Maheswaran, Shoukat Ali, Howard Jay Siegel, Debra Hensgen, and Richard Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Proceedings of the Heterogeneous Computing Workshop*, pages 30–44, San Juan, PR, April 1999. IEEE Computer Society.

[13] Richard Sargent. *CDSA Explained: An Indispensable Guide to Common Data Security Architecture*. The Open Group, Reading, Berkshire, UK, 1998.

[14] P. Schneck and K. Schwan. Dynamic authentication for high performance networked applications. Technical Report GIT-CC-98-08, Georgia Institute of Technology, College of Computing, Atlanta, GA, 1998.

[15] L. Smarr and C. Catlett. Metacomputing. *Communications of the A.C.M.*, June 1992.

[16] N. Vendatasubramanian and K. Narstedt. An integrated metric for video qos. In *A.C.M. International Multimedia Conference*, Seattle, WA, November 1997.

[17] Lonnie Welch, Michael W. Masters, Leslie Madden, David Marlow, Philip Irey, Paul Werme, and Behrooz Shirazi. A Distributed System Reference Architecture for Adaptive QoS and Resource Management. In Jose Rolim, editor, *Proceedings of 11th IPPS/SPDP'99 Workshops*, pages 1316–1326, Berlin, April 1999. Springer.

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, Code 013 2
Naval Postgraduate School
Monterey, CA 93943-5100

3. Research Office, Code 09 1
Naval Postgraduate School
Monterey, CA 93943-5138

5. Dr. Cynthia E. Irvine 10
Code CS/Ic
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5118