To Appear in Proceedings of the 2000 Heterogeneous Computing Workshop, May 2000.

Toward Quality of Security Service in a Resource Management System Benefit Function

Cynthia E. Irvine
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943 USA
email: irvine@cs.nps.navy.mil

Timothy E. Levin
Anteon Corporation
2600 Garden Road
Monterey, CA 93940 USA
email: levin@cs.nps.navy.mil

Abstract

Enforcement of a high-level statement of security policy may be difficult to discern when mapped through functional requirements to a myriad of possible security services and mechanisms in a highly complex, networked environment. A method for articulating network security functional requirements, and their fulfillment, is presented. Using this method, security in a quality of service framework is discussed in terms of "variant" security mechanisms and dynamic security policies. For illustration, it is shown how this method can be used to represent Quality of Security Service (QoSS) in a network scheduler benefit function[1].

1 Introduction

Several efforts are underway to develop middleware systems that will logically combine network resources to construct a "virtual" computational system [4] [7] [8] [15] . These geographically distributed, heterogeneous resources are expected to be used to support a heterogeneous mix of applications. Collections of tasks with disparate computation requirements will need to be efficiently scheduled for remote execution. Large parallelized computations found in fields such as astrophysics [14] and meteorology will require allocation of perhaps hundreds of individual processes to underlying systems. Multimedia applications, such as voice and video will impose requirements for low jitter, minimal packet losses, and isochronal data rates. Adaptive applications will need information about their environment so they can adjust to changing conditions.

User acceptance of these virtual systems, for either commercial or military applications, will depend, in part, upon the security, adaptability, and user-responsiveness provided. Several of the projects engaged in building the middleware to create these networks are pursuing the integration of security [6] [10] [23] and quality of service [1] [17] into these systems. The need for virtual networked systems to both adapt to varying security conditions, and offer the user a range of security choices is apparent.

In the network computing context, users or user programs may request the execution of "jobs," which are scheduled by an underlying control program to execute on local or remote computing resources. The execution of the job may access or consume a variety of network resources, such as: local I/O device bandwidth, internetwork bandwidth; local and remote CPU time; local, intermediate (e.g., routing buffers) and remote storage. The resource usages may be temporary or persistent. As there are multiple users accessing the same resources, there are naturally various allotment, contention, and security issues regarding use of those resources.

The body of rules for resolving network security issues is called the network security policy, whereas the body of rules for resolving network contention and allotment comprise a network management policy (which is sometimes taken to include the network security policy). These policies consist of broad policy jurisdictions, such as scheduling, routing, access control, auditing, and authentication. Furthermore, these jurisdictions can be decomposed, typically, into functional requirements, such as, "users from network domain A must not access site B," and "user C must receive a certain quality of service." The network management and security policies, as mapped through the functional requirements, may be manifested in mechanisms throughout the network, including: host computer operating systems, network managers, traffic shapers, schedulers, routers, switches and combinations thereof. As these mechanisms are distributed and are often obscurely related, there has been some interest in the ability to express and quantify the level of support for security policy and Quality of Security Service (QoSS: managing security and security requests as a responsive "service" for which

1. This work was supported by the DARPA/ITO Quorum program.

quantitative measurement of service “efficiency” is possible) provided in networked systems.

The purpose of this paper is to present the system developed for the MSHN resource management system [8] for describing network security policy functional requirements, to show how QoSS parameters and mechanisms can be represented in such a system, and to provide an example of the use of this system. The remainder of this paper is organized as follows. Section 2 discusses a “security vector” for quantifying functional support of network security policy. Section 3 describes how the security vector can be used for expressing the effects of QoSS in a network-scheduling benefit function; and a conclusion follows in Section 4.

2 Network Security Vector

A *network security policy* can be viewed as an n-dimensional space of functional security requirements. We represent this multidimensional space with a *vector* (S) of security components. Each component (S.c) specifies a boolean functional requirement, whereby the instantiation of a network job either meets (possibly trivially) or does not meet each of the requirements. By convention, a security vector’s components are ordered, so they can be referenced ordinally (S.3) or symbolically (S.c). A component may indicate positive requirements (e.g., communications via node n must use encryption) as well as negative constraints (e.g., users from subnet s may not use node n). Components can also be hierarchically grouped. [22] Requirements for a given security service may be represented by one or more components (indicating a service *sub-vector*), and a security service may utilize functions and requirements of other services and their components.

Some jobs can produce output in different *formats*, where a given format (e.g., high resolution video) might be more resource consumptive than another format (e.g., low resolution video). Formats may have differing security requirements, even within the same job. For example, a video-stream format may require less packet authentication [19] , percentage-wise, than a series of fixed images based on the same data. A “quality of service” scheduling mechanism might choose one format for a job over another, depending on varying network conditions (e.g., traffic congestion). Further, adaptive applications may select formats depending upon changing conditions. For example, IPSec, security association (SA) processing using ISAKMP under IKE can permit complex security choices through an SA payload; and the payload recipient may be given transform choices regarding both Authentication Header and Encapsulating Security Protocol [13] .

2.1 Notation

The set of all jobs is represented by *J*. The set of all formats is represented by *I*. The notation S_{ij} identifies a vector containing the portions of *S* that are applicable to job *j* in format *i*, and $S_{ij}.c$ identifies a given component (*c*) of S_{ij}. The relation of *S* to S_{ij} is clarified further, below. The following are some informal examples of security-vector components:

- S.1: user access to resource is equal to read/write; based on table t
- S.2: % of packets authenticated >= 50, <= 90; inc 10
- S.3: clearance (user) = secrecy/integrity (resource)
- S.4: length of confidentiality encryption key >= 64, <= 256; inc 64
- S.5: authentication header transform *in* {HMAC-MD5, HMAC-SHA}
- S.6: packets from domain A to domain B must be encrypted
- S.7: packets from domain A cannot be sent through domain C

Here, “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, [*S.c*]. In the above examples, [*S.1*] = 1, and [*S.2*] = 5.

2.2 Variant Security Components

When [S.c] > 1, the underlying control program has a range within which it may allow the job to execute with respect to the policy requirement. We refer to this type of policy, and component, as “variant.” Security-variant policies may be used within a resource management context, for example, to effect adaptation to varying network conditions. [18] Also, if the policy mechanism is variant, the control program may offer QoSS choices to the users to indicate their preferences with respect to a given job or jobs. Without variant mechanisms, neither security adaptability by the underlying control program nor QoSS are possible, since fixed policy mechanisms do not allow for changes to security within a fixed job/resource environment. While the expression S.c may contain a compound boolean statement (see Section 2.3), by convention it may contain only one variant clause.

2.3 Component Structure

For use in the examples in this discussion, a component has the following composition (see Table 1 for details):

- component ::= boolean expression, variant-range-specifier ; modifying-clause
- boolean _expression ::= boolean_statement [(or | and) boolean_statement]*
- boolean_statement ::= LHS boolean-operator RHS

Note that it is not the focus here to elaborate on a policy representation language. See other efforts and works in progress [2] [3] [5] [16] .

A given policy component has a *value* which is a boolean *expression.* This component may also have an *instantiated value* with respect to a specific job and format, which is either "true" or "false." A component has a left hand side (LHS), which is the item that is being tested; of course the LHS has a *value* as well as an *instantiated value*. A component also has a right hand side (RHS), which is what the LHS is tested against, as well as zero or more modifying clauses. Similarly to the LHS, the RHS may have a value (or values) which is dependent on the instantiation of the component.

2.4 Dynamic Security Policies

With a dynamic security policy, the value of a vector's components may depend on the network "mode" (e.g., normal, impacted, emergency, etc.), where M is the set of all modes. There is, conceptually, a separate vector for each operational mode, represented as: S^{mode}. Access to a predefined set of alternate security policies allows their functional requirements and implementation mechanisms to be examined with respect to the overall policy prior to being fielded, rather than depending on *ad hoc* methods, for example, during an emergency.

Initially, every component of S has the same value in each of its modes. Ultimately, components may be assigned different values, depending on the network mode. For example:

- $S^{normal}.a$: % packets authenticated >= 50, <= 90; inc 10
- $S^{impacted}.a$: % of packets authenticated >= 20, <= 50; inc 10
  Note how [*S.a*] changes from 5 to 4 under the impacted mode
- S^{normal}.b: user access to network node = granted; based on table t
- $S^{impacted}$.b: user access to network node = granted; based on table t, OR UID in set of administrators
- $S^{emergency}$.b: UID in set of {administrators, policymakers}

Or, for example, policy makers might decide that the policy should remain in force regardless of network mode:

- $S^{normal}.c$ = $S^{impacted}.c$ = $S^{emergency}.c$: clearance (user) = classification (resource)

Table 1: Simple Component Elements

Element Name	**Example S.1**	**Example S.2**
Value	user access to resource r = RW, based on table t	% of packets authenticated >= 50, <= 90; inc 10
Instantiated value	false	true
Value of LHS	user access to resource r	% of packets authenticated
Instantiated value of LHS	W	70
Boolean operator	=	>=
Value of RHS	RW	50
variant range specifier	none applicable	<= 90
Modifying clause	based on table t	inc 10

If a mode is not specified for a component (e.g., “S.a”), normal mode is assumed. This will be the case (i.e., the mode is unspecified) for the remainder of this discussion.

2.5 Refinements to Security Vector

R is the set of resources $\{r_1 .. r_n\}$. R_{ij} is the subset of *R* utilized in executing job *j* in format *i*.

T_j is the requested completion time of job j.

Security policies may be expressed with respect to principals (user, group or role, etc.,), applications, data sets (both destination and source), formats, etc., as well as resources in R_{ij}.

The definition of S_{ij} is finally refined as follows: S_{ij} is a vector that is an order-preserving projection of *S*, such that a component *c* from *S* is in S_{ij} if and only if the value of *c* depends on format *i*, job *j*, or any *r* in R_{ij}. The number of components in a security vector S_{ij} *is* $[S_{ij}]$.

2.6 Summary of Security Vector

S is a general purpose notational system suitable for expressing arbitrarily complex sets of network security mechanisms. *S* can express variant policies, to allow security expressions of quality of service requests, and can have dynamic security elements to accommodate multiple situation-based policies. In particular, *S* can represent both (1) static security requirements that may be implemented in a system, as well as (2) the results of running a particular job or set of jobs against such static requirements. The latter usage will be examined in the next section, to express QoSS in a resource management system benefit function.

3 Network-Scheduler Benefit Function

As discussed above, various mechanisms exist for managing contention for, and allotment of distributed network resources. One class of these mechanisms attempts to efficiently schedule the execution of multiple (possibly simultaneous) jobs on multiple distributed computers (e.g., the MSHN project [8] [23] [24] [11] [17]), where each job requires a determinable subset of the resources. Of interest is a benefit function for comparing the effectiveness of such job scheduling mechanisms when they are presented with real or hypothetical “data sets” of jobs.

Jobs are assigned *priorities* for use in resolving resource contention and allocation issues. In some systems, a job’s priority may depend upon the particular operating *mode* of the network. [8] Also, the different data formats of a multiple-format job may have different *preferences* (e.g., assigned by a user or “hard wired” as part of the application or job-scheduler database), and different levels of resource usage. [10] [12] A network job scheduler should receive more credit in the benefit function for scheduling high priority and high preference jobs, as opposed to low priority or low preference jobs. That is to say, a scheduler is intuitively doing a better job if important jobs, as judged by priority and preference, receive more attention than unimportant jobs. How much weight the priorities and preferences are given is a matter of network scheduling policy.

For illustration, we introduce a simple benefit function, *B*, to measure how well a scheduler meets the goals of user preference and system priorities (see [4] , [12] and [21] for other approaches). This function averages preference (*p*) and priority (*P*) (use of a priority and preference in measuring network effectiveness have been introduced for the MSHN project [10]).

$$B = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m_j} X_{ij}(P_{ij} + p_{ij})}{2n}$$

Where the characteristic function *X* is defined for *i, j* as:

Xij = 1 if format i was successfully delivered to job *j* within time T_j, else 0

and at most one format is completed per job:

$$\forall j \in J \left(\sum_{i=1}^{m_j} X_{ij} \le 1 \right)$$

Jobs and formats are defined as above.

P_j is the priority of job *j*

$$0 \le P_j \le 1$$

The formats for a job are assigned preferences (*p*) by the user such that:

$0 <= p <= 1$

m_j is the number of {format, preference} pairs assigned for job *j*

p_{ij} is the preference the user has assigned to format *i*, job *j*

the preferences for a job add up to 1:

$$\forall j \in J : \sum_{i=1}^{m_j} p_{ij} = 1$$

This approach assumes that users will assign preference values that correspond to resource usage, since we want the benefit function to indicate a higher value when the scheduler succeeds in scheduling “harder” jobs [12] .

3.1 Adding Security to the Benefit Function

We wish the benefit function to reflect the effectiveness and restrictions of the security policy. First, we define the characteristic security function *Z*, for *i* and *j*:

$Z_{ij} = 1$ if the instantiated value of all components in S_{ij} are true, else 0

The numerator of the benefit function is multiplied by *Z*, so that no credit is given for jobs that fail to meet the security requirements:

$$B = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m_j} X_{ij} Z_{ij} (P_j + p_{ij})}{2n}$$

Now, for variant components, we wish to be able to give less credit to the scheduler for fulfilling less "difficult" security requirements. One algorithm for expressing this is for each instantiated component (*c*) in S_{ij} to be assigned a security completion token (*g*) where $0 \le g \le 1$. g_c will indicate the completion token corresponding to component *S.c*. g_c is defined to be the "percentage" of [*S.c*] met or exceeded by the *instantiated value* of the component's LHS (notated as *S.c"*):

$g_c = S.c" / [S.c]$

To illustrate the calculation of g_1, for component S.1:

S.1: % of packets authenticated >= 50, <= 90; inc 10
[*S.1*] = 5 (the number of quanta in *S.1), S.1"* = 3 (the job achieves the 3rd quantum (70))
$g_1 = 3/5 = 0.6$

For invariant components, $g = 1$ or $g = 0$. A token (*g*) whose value is 0 represents a job "failing" the component's security policy. Recall that *Z* will be 0 when the job/format fails to meet the requirement of any security component, meaning that the function returns no benefit value for that job/format. We introduce a function (*A*) which averages the tokens of a job:

$A_{ij} = (g_1 + g_2 + .. + g_n)/n$
where n = $[S_{ij}]$ -- the number of components in S_{ij}
and $(0 \le A_{ij} \le 1)$

Averages, such as *A*, over many different elements can tend to minimize the difference that is seen between different data sets. Therefore, we weight the tokens (*g*) assigned to individual security components to give more credence to components that are "more important" than others, e.g., reflecting network management policies. Each g_n has a corresponding integer weight (w_n), $w_c \ge 0$. So *Aij* becomes:

$A_{ij} = (g_1 w_1 + g_2 w_2 + .. + g_n w_n)/(w_1 + w_2 + .. + w_n)$
again $(0 \le A_{ij} \le 1)$

In the final expression of the network benefit function, *A* is added to the numerator, providing an average of security, priority and preference.

$$B = \frac{\sum_{j=1}^{n} \sum_{i=1}^{m_j} X_{ij} Z_{ij} (P_j + p_{ij} + A_{ij})}{3n}$$

$0 \le B \le 1$, where 1 indicates the maximum scheduling effectiveness.

3.2 Applicability

This technique for quantifying the variant security instantiated by a resource management system is being used in the MSHN project as a factor in representing the effectiveness of its resource assignments [10] . In the MSHN design, the security requirements of network resources (represented by *S*) are stored in a Resource Requirements Database. This database is consulted during the resource scheduling phase to effectively match jobs to resources. We expect that this measurement technique could also be applied to other resource management systems, such as Condor [15] and Globus [7] .

While different schedulers could be compared with respect to the individual components of *B*, a summary function such as *B* would be useful to automate and normalize the comparison process. Additionally, we expect that the security component (viz, *A*) in an operational system would be complex enough to evade effective manual analysis.

4 Discussion and Conclusion

A security vector has been presented for describing functional requirements of network security policies. It has been shown that this vector can be used for representing security with respect to both quality of service and a network scheduler benefit function.

We are involved in ongoing work to organize the security vector into a "normal form" with sub-vectors or hierarchies corresponding to security policy jurisdictions (such as: access control, auditing, and authentication) and to incorporate a costing methodology for security components, such as can be provided to a resource management system [9] . We are working to develop a means of adjusting the preference expression with a notion of the corresponding resource usage [12] . We are considering how to expand the security benefit function (*A*) to reflect user quality of security service choices within the range allowed by variant security components, and to reflect performance implications of redundant security mechanisms.

The organizational security policy [20] governing the network may allow individuals or principals representing them to override rules represented by invariant security vector components. For example, a military commander might decide to forgo cryptographic secrecy mechanisms for a job in an emergency (e.g., to improve network performance), even though the system has not been configured with "dynamic" or "variant" security mechanisms, as defined herein. From the perspective of the security vector S and the benefit function, this is a *change* to *or violation of the computer security policy.* It is recommended that this type of policy change be audited.

References

[1] Aurrecoechea, C., Campbell, A., and Hauw, L. "A Survey of Quality of Service Architectures", Multimedia Systems Journal, Special Issue on QoS Architectures, 1996.

[2] Badger, L., Stern, D. F., Sherman, D. L., Walker, K. M., and Haghighat, S. A., "Practical Domain and Type Enforcement for Unix," Proceedings of 1995 IEEE Symposium on Security and Privacy, 1995, Oakland, Ca., pp. 66-77

[3] Blaze, M., Feigenbaum, J., and Lacy, J., "Decentralized Trust Management," in Proceedings of 1996 IEEE Symposium on Security and Privacy, May 6-8, 1996, Oakland, Ca., pp 164-173

[4] Chatterjee, S., Sabata, B., Sydir, J. "ERDoS QOS Architecture," SRI Technical Report, ITAD-1667-TR-98-075, Menlo Park, CA, May 1998.

[5] Condell, M., Lynn, C. and Zao, J. "Security Policy Specification Language," INTERNET-DRAFT, Network Working Group, July 1, 1999, ftp://ftp.ietf.org/internet-drafts/draft-ietf-ipsec-spsl-01.txt, Expires January, 2000

[6] Foster, I, N. T. Karonis, N. T., Kesselman, C., Tuecke, S. Managing Security in High-Performance Distributed Computing. Cluster Computing 1(1):95-107, 1998.

[7] Foster, I., and Kesselman, C., Globus: A Metacomputing Infrastructure Toolkit. Intl J. Supercomputer Applications, 11(2):115-128, 1997.

[8] Debra Hensgen, Taylor Kidd, David St. John, Matthew C. Schnaidt, H. J. Siegel, Tracy Braun, Jong-Kook Kim, Shoukat Ali, Cynthia Irvine, Tim Levin, Viktor Prasanna, Prashanth Bhat, Richard Freund, and Mike Gherrity, An Overview of the Management System for Heterogeneous Networks (MSHN), 8th Workshop on Heterogeneous Computing Systems (HCW '99), San Juan, Puerto Rico, Apr.1999

[9] Irvine, C., and Levin, T., Toward a Taxonomy and Costing Method for Security Metrics, Annual Computer Security Applications Conference, Phoenix, AZ, Dec. 1999

[10] Kim, Jong-Kook, Hensgen, D., Kidd, T., Siegel, H.J., St.John, D., Irvine, C., Levin, T., Porter, N.W., Prasanna, V., and Freund, R., A QoS Performance Measure Framework for Distributed Heterogeneous Networks, Proceedings of the 8th Euromicro Workshop on Parallel and Distributed Processing, Rhodos, Greece, January 2000.

[11] Lee, C. Kesselman, C., Stepanek, j., Lindell, R., Hwang, S., Scott Michel, B., Bannister, J., Foster, I., and Roy, A. The Quality of Service Component for the Globus Metacomputing System. Proc. 1998 International Workshop on Quality of Service, Napa California, pp. 140-142, May, 1998.

[12] Levin, T., and Irvine C., An Approach to Characterizing Resource Usage and User Preferences in Benefit Functions, NPS Technical Report, NPS-CS-99-005

[13] Maughan, D., Schertler, M., Schneider, M., and Turner, J. Internet Security Association and Key Management Protocol, RFC 2408, http://info.internet.isi.edu/in-notes/rfc/files/rfc2408.txt

[14] Ostriker, J., and Norman. M. L., Cosmology of the Early Universe Viewed Through the New Infrastructure. C.A.C.M. 40(11):85-94.

[15] Raman, R., Livny, M., Solomon, M., "Matchmaking: Distributed Resource Management for High Throughput Computing," Proceedings the 7th IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago, Ill.

[16] Ryutov, T. and Neuman, C. Access Control Framework for Distributed Applications. INTERNET-DRAFT, CAT Working Group, USC/Information Sciences Institute, draft-ietf-cat-acc-cntrl-frmw-00.txt, August 07, 1998, Expires February 1999, http://gost.isi.edu/info/gaa_api.html

[17] Sabata, B., Chatterjee, S., Davis, M., Sydir, J., Lawrence, T. "Taxonomy for QoS Specifications," Proceedings the Third International Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'97), February 5-7, 1997, Newport Beach, Ca., pages 100-107

[18] Schantz, R. E. "Quality of Service," to be published in "Encyclopedia of Distributed Computing," 1998.

[19] Schneck, P. A., and Schwan, K., "Dynamic Authentication for High-Performance Networked Applications," Georgia Institute of Technology College of Computing Technical Report, GIT-CC-98-08, 1998.

[20] Stern, D. F., On the Buzzword "Security Policy", Proceedings of 1991 IEEE Symposium on Security and Privacy, 1991, Oakland, Ca., pages 219-230.

[21] Vendatasubramanian, N. and Nahrstedt, K., "An Integrated Metric for Video QoS," ACM International Multimedia Conference, Seattle, Wa., Nov. 1997.

[22] Wang, C. and Wulf, W. A, "A Framework for Security Mea-

surement." Proc. National Information Systems Security Conference, Baltimore, MD, pp. 522-533, Oct. 1997.

[23] Wright, R., Integrity Architecture and Security Services Demonstration for Management System for Heterogeneous Networks, Masters Thesis, Naval Postgraduate School, Monterey, CA, Sept. 1998.

[24] Wright, R., Shifflett, D., and Irvine, C. E., "Security Architecture for a Virtual Heterogeneous Machine." Proc. Computer Security Applications Conference, Scottsdale, AZ, Dec. 1998, pp 167-17

Biographies

Cynthia E. Irvine is Director, Naval Postgraduate School Center for INFOSEC Studies and Research and an Assistant Professor of Computer Science at the Naval Postgraduate School. Dr. Irvine holds a Ph.D. from Case Western Reserve University. She has over thirteen years experience in computer security research and development. Her current research centers on architectural issues associated with applications for high assurance trusted systems, security architectures combining popular commercial and specialized multilevel components, and the design of multilevel secure operating systems.

Timothy E. Levin is a Senior Research Associate at the Naval Postgraduate School Center for INFOSEC Studies and Research. He received a BS degree in Computer and Information Science from the University of California at Santa Cruz, 1981. He has over fifteen years experience in computer security research and development. His current research interests include management and quantification of heterogeneous network security in the context of Resource Management Systems, development of costing frameworks and scheduling algorithms for the dynamic selection of QoS security mechanisms, and the application of formal methods to secure computer systems.