Quality of Security Service: An Introduction[1]

Cynthia Irvine
Naval Postgraduate School
Monterey, CA

Tim Levin
Anteon Corporation
Monterey, CA

Abstract. We examine the concept of security as a dimension of Quality of Service in distributed systems. We provide a discussion and examples of user-specified security variables and show how the range of service levels associated with these variables can support the provision of Quality of Security Service. We also discuss various design implications regarding security ranges provided in a QoS-aware distributed system.

Keywords. Quality of Service, Quality of Security Service, variant security, security range.

1 Introduction

Quality of Service (QoS) mechanisms benefit both the user and the overall distributed system. QoS users benefit by having reliable access to services; and the distributed systems whose resources are QoS managed benefit by having more predictable resource utilization and more efficient resource allocation (that is, in systems where allocation efficiency is supported). The motivation for the work described here has been to help determine if this reliability, predictability and efficiency can be enhanced by including security as a real part of QoS. We have termed the effects of this inclusion, "Quality of Security Service" (QoSS).

Inherently, QoS involves user requests for (levels of) services which are related to performance-sensitive variables in an underlying distributed system. For security to be a real part of QoS, then, security choices must be presented to users, and the QoS mechanism must be able to modulate related variables to provide predictable security service levels to those users. This raises the question of whether it makes sense within the context of coherent system security paradigms to provide such security choices to users. It is also of interest to understand how the limits on these choices are defined, and how those limits relate to existing resource security policies.

The purpose of this paper is to provide an overview, rationale and motivation for understanding QoSS and variant security, and how these concepts may benefit future application and system designs. The remainder of this document is structured as follows:

- Section 2 provides background on Quality of Service concepts related to security services;
- Section 3 describes the concept of Quality of Security Service, and provides a discussion of the general "assurability" of application-centric security enforcement mechanisms;
- Section 4 provides a description and rationale for various forms of user and application security "ranges;"
- Section 5 describes some design considerations regarding variant security in distributed multi-tiered systems; and
- Section 6 is a summary discussion.

1.1 Related Work

A *Quality of Protection* parameter is provided in the GSS-API specification [13]. This parameter is intended to manage the level of protection provided to a message communication stream by an underlying security mechanism

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

(or service), “allowing callers to trade off security processing overhead dynamically against the protection requirements for particular messages.” Another early reference to a variable security service is that of Schneck, and Schwan [17], which discusses variable packet authentication rates with respect to the management of system performance. Our work is intended to extend these efforts into a more general framework which is applicable to a wide range of policy, processing and networking contexts, as well as diverse security services.

References to security in the QoS literature can be found in [6], [16], and [21], although little is mentioned there of security variability or use of security as a functional QoS dimension. QoS itself has been extensively discussed in the literature, and we refer the reader to [2] for a thorough review of QoS definitions and architectures.

A *trust management system* [3][4] provides a language and mechanism for specifying security policies and credentials, and may include a *policy server* or compliance checker to resolve questions about access control. The trust management system is not concerned with the nature of the specific policies (e.g., those involving variant security) which it stores and resolves. Nor is the trust management system expressly concerned with QoS issues. However, a QoSS system could be built to utilize a trust management system to store and resolve security range relationships.

2 QoS and Resource Usage Control

The resource usage load on traditional (e.g., not inter-networked) multi-user systems could be understood, simplistically, to be a linear function of the number of users. Similarly, user load could be seen as a function of the number of user terminals configured for the system. Thus, a system administrator could govern the system resource usage load, to a degree, by controlling the number and type of user input terminals (e.g., interactive terminals, modems and card readers). In a distributed and inter-networked environment, system administrators are often without recourse to such straightforward and simplistic resource-usage control approaches, since the number and type of user “terminals” and associated tasks may not be bounded by local (e.g., campus or enterprise) topographies. In some cases, users of system resources may extend across the Internet. The Quality of Service paradigm is designed to help address this problem by providing to users and administrators certain tools for managing resource usage and service levels.

Quality of Service refers to the ability of a distributed system to provide network and computation services such that each user’s expectations for timeliness and performance quality are met. There are several dimensions of Quality of Service described in the literature [6][20], including, accuracy, precision and performance. For a Quality of Service *dimension* to be supported means that users can request or specify a level of service for one or more attributes of these dimensions, and the underlying QoS control mechanism is capable of entering into an agreement to deliver those services at the requested levels. Therefore, the control mechanism must be able to modulate the level of the service to individual subscribers (e.g., users). For example, a network-based multimedia application might be expected to deliver video frames so that the display is jitter-free to some requested level [20],[6].

In addition to meeting individual user requirements, a QoSM makes choices that permit it to maximize overall benefit in accordance with its QoS policy. For example, one QoS policy might require that benefit be equally shared among all tasks. This would mean that if network resources were over subscribed all tasks would have a reduction in service. Another policy might state that no service is better than poor service, so that if resources were sufficiently oversubscribed, some tasks would be postponed or terminated. This policy could be extended so that certain tasks would be given priority for guaranteed service during times of resource congestion.

Users present their expectations to the QoS mechanism by way of service level *requests*. These requests can take the form of both *hard* and *soft* requirements [19]. In essence, the system enters into a contract with the user to meet the hard and soft requirements. Hard requirements mandate fixed service levels that the QoS mechanism must deliver if it is to accept the user’s task; whereas, a soft requirement can be considered to define a range of acceptable service, for example, in terms of bandwidth, response time, or image fidelity. Each soft requirement represents a variable which the QoS control mechanism can manipulate in balancing the needs of multiple users. Given latitude in the user’s soft requirements, the more variables that the control mechanism has to manipulate, the easier will be the job of satisfying the set of current users. Conversely, the QoSM can offer choices to the user (in response to which the user may enter hard or soft requests) only for aspects of the system over which it controls,

and is willing to provide, a range of service. For aspects in which there is no such control, only a fixed or "best effort" type of service can be delivered, so QoS concepts (e.g., regarding service level requests) do not apply.

3 Security

The purpose of this section is to provide an analysis of the role of security in a system designed to provide QoS. Security has long been a gleam in the eye of the QoS community: many QoS RFPs and QoS system-design presentation slides have included a place-holder for security, without defining security as a true QoS *dimension* (as above). Some of these presentations have provided access control mechanisms within the QoS framework [14][12], but they have only touched on security as a QoS dimension.

3.1 Quality of Security Service

We believe that QoS mechanisms can be more effective if, like response time and image fidelity, variable levels of security services and requirements can be presented to users or network tasks, providing security choices within acceptable ranges. As described above, these ranges result in additional tools (i.e., parameters) with which the QoSM can successfully meet overall user demands. Furthermore, if user security service requests are defined as ranges, then the underlying system can *adapt* more gracefully to changes in resource availability during the execution of a task, and thereby do a better job at maintaining requested or required levels of service in all of its dimensions. We use the term *Quality of Security Service* to refer to the use of *security* as a quality of service dimension.

To recap, the enabling technology for both QoSS and a security-adaptable infrastructure is *variant security*, or the ability of security mechanisms and services to allow the amount, kind or degree of security to vary, within predefined ranges. This notion of network Quality of Security Service has the potential to provide administrators and users with more flexibility and potentially better service, without compromise of network and system security policies.

3.2 Application-Centric Security

The traditional view of access control was OS-centric. The operating system enforced a policy, to the best of its ability, and ideally, objects never left the control domain of the OS. Policies that were enforced globally and persistently within this domain were considered to be "mandatory," and all others were considered to be "discretionary" [5]. With the advent of distributed/heterogeneous applications, data storage objects, operating systems, and resources, and a plethora of middleware mechanisms for managing those distributed entities, *application-centric access control* has now become common, if not the norm [3]. In this Brave New World, the application itself (perhaps in concert with some middleware mechanisms) enforces access control on its objects, rather than depending for this function on an underlying (e.g., OS and hardware) control mechanism. Thus, network applications have assumed some functions of the traditional OS. If the applications's objects are completely encapsulated, such that the object never leaves the control domain of the application[1], then a global and persistent policy could be said to be enforced, assuming persistence on the part of the application. However, this is a necessary, but not sufficient condition for effective policy enforcement.

Another traditional aspect of policy enforcement was the notion that, to be considered highly effective, access control should be performed at the lowest level(s), including hardware, of a strictly layered system. The reason for allocating access control functions to the lower levels is that it is more feasible, then, to ensure that the mechanisms are non-bypassable, persistently enforced, and small enough to allow thorough analysis (e.g., see [1]). Thus,

1. Note that if the object is allowed to leave the application's domain, then it is more difficult to argue that a global policy is enforced; a component of one such argument for a distributed application is that objects in transit are protected, perhaps by cryptographic mechanisms, to the extent that the object remains, logically, in the control domain of the application.

regardless of how well formed or misused was an application, if the enforcement layers were well formed, the policy enforcement could be ensured. Modern distributed applications do not necessarily have these two properties (dependency layering, and access control implemented at the lowest levels). A network application typically depends on an untrusted operating system for access to resources, and we suggest that no application under these conditions can be considered to enforce a security policy with high effectivity (assurance). Neither is dependency layering a fundamental design consideration in many modern distributed or object-oriented applications and systems. As a result, the distributed application needs to be analyzed very carefully to understand whether or not it has the capability, by virtue of its design, to enforce a policy; for without understanding the dependency layering, it will not be clear on which other modules the application depends, nor will it be clear if there are fatal (e.g., circularly dependent) or semantically undefined execution sequences. Therefore, under the conditions described here, much more design analysis may be involved in understanding the degree to which a distributed system is capable of enforcing a security policy, than was required to analyze a traditional layered system.

We present in the following pages some thoughts about how certain QoS aspects of application-centric access control security can be understood and managed. This is not to say that this approach ameliorates the design analysis problems of application-centric access control. On the contrary, we would reiterate that each such system needs careful design review to understand the effectiveness of its security mechanisms. Hopefully, the security abstractions presented here will aid in such analyses.

4 Security Ranges

The notion of security ranges may, at first, seem strange or even an oxymoron. For many, security is thought to be binary: either you have it or you don’t. On a gross scale, this is true. Without some minimum level of security, a system will be considered inadequate for user requirements. Yet if a user’s minimum requirements are met, can there not be some choice with respect to what *is* adequate? Our answer is "yes." As an initial example, suppose that a user requires medium assurance at end systems where a distributed task will be executed. If potential target platforms range between medium and high assurance, there is a choice. In fact, if the medium assurance system is over-subscribed while the high assurance system is idle, the user may realize better overall service by electing to execute the task on the high assurance processor.

Consider the security administrator's or the user's motivation in agreeing to or specifying a range of security. As with multimedia image resolution, users will generally desire the greatest amount of security (or image fidelity) available, but this desire is generally tempered by cost. The cost may take the form of monetary charges (unlimited bandwidth but at a high cost per byte) or performance degradation (for high resolution, processing and download times will be long), for example. When the cost is very high (e.g., slow response time), users may be willing to accept security (or imagery) that is less than their ideal level of service. Thus, the user/administrator’s *acceptable* security would range from a *minimum* to an *ideal*. A system that is sufficiently flexible may be able to impose performance degradations on others when an application that is willing to pay enough or has the highest priority is introduced. By indicating a range within which they are willing to operate, the poorer or lower priority tasks will still be able to run rather than being terminated or rejected.

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, the level of security might be tied to another desirable service level, such as image fidelity. An application may have variable data formats which have correspondingly variable security requirements, as shown in Table 1. Here, the degraded image requires less security, and conversely, the enhanced image requires more security. So a user might welcome heightened security

if it is tied to her desire for more image fidelity.

Table 1: Security Choice Related to Fidelity Choice

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

An example taken from a popular military novel will help to illustrate our point. Suppose that high, medium and low resolution images of enemy troop movements are available. Here we will assume that resolution and fidelity are equivalent. To protect the technologies used to obtain both the high and medium resolution images, correspondingly high and medium confidentiality mechanisms are required to protect the images, which will be handled as two levels of classified information. The images will be useful for military purposes only for a limited time, because within a few days the battle will be over. Those planning the battle strategy consider medium confidentiality to be sufficient to protect the images since even medium strength security mechanisms are considered sufficient to protect the information for a few days. However, to be useful for tactical planning high resolution is required. This means that the military strategists must employ high confidentiality mechanisms. Thus we have a situation in which the tactical field commander would be happy enough with medium security but her requirement for high resolution (or high fidelity) imposes a requirement for high security. The combined requirements for high fidelity and high security consume processing and network resources to produce low overall performance.

From the above example, we can also observe that if the fidelity of the images is diminished, the requirements for security are also reduced. If the images become fuzzy enough, little or no security is required. In this case, resource usage will be low and overall performance will be high.

An integrity example may also be useful. Suppose that a surgeon is performing a delicate brain operation remotely. To ensure that only the precise brain locations are affected, high fidelity is required. Additionally, there is a requirement for high integrity to ensure that the video stream is not tampered with by malicious entities who might wish to ruin the operation and render the patient a vegetable. Secrecy is not a requirement, yet to achieve the performance required, a hardware mechanism that provides both a high level of secrecy and integrity could be used. In this case the operation achieves high secrecy as a bonus resulting from fidelity and integrity requirements.

The following are some more examples of the use of real and hypothetical security ranges.

- Collaborative applications, such as video teleconferencing with shared electronic white boards, and application suites, may present communication security choices to participants. For example, if a group member is participating in the collaboration from a hotel room in a foreign country known for government support of corporate espionage, his security requirements and choices will be quite different than if he were in “friendly” territory. These security choices may form a range from which the user or application can select, and can include different levels of authentication, confidentiality, and integrity.
- Destination subnets could be classified by risk factor with respect to routing through, execution on, or logging on to nodes in those subnets. Users, applications or enterprise-wide mechanisms could request of middleware control mechanisms that communications or tasks executed on the user’s behalf utilize a specific *risk range* of subnets (e.g., the user’s QoSS specification might include the request to use any “high to very high” security subnets for this invocation).
- Some environments may offer the user choices of log-on authentication technology. For example, a user may log on with a password, a one-time password (crypto challenge-response), a public-key smart card, a biometric, or some combination of these. In these environments, the user could be granted greater access to resources (e.g., a higher classification of data) if he uses higher-assurance authentication [11].
- Another example is that the underlying system supports different *situational modes*. For some modes (e.g., normal, impacted, emergency), the user or administrator may be willing to accept more (or less) security for a

given application. A commander under attack at a foreign embassy might require the highest communication security; whereas a commander under attack on the battlefield might declare, “damn the security, full speed ahead!” The MSHN resource management system is an example of a system in which the management of mode vs. security requirements is designed to be handled automatically [8][9].

- The security policy for a hypothetical commercial sub-network requires outgoing IP packet encryption. In this environment, a multimedia application exports digital images (e.g., high resolution fine art images). However, recognizing that the stake-holders in this specific environment can tolerate a media stream which is *partially* or *periodically* encrypted (viz, one yielding a suitably obscured image, which would render a stolen image unusable by the vast majority of its target market), 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 the entire unencrypted image is not revealed in repeated transmissions.)
- Variable packet authentication [17] is a corollary to the preceding confidentiality scenario. In this case, the sender or recipient might be satisfied if (only) a certain percentage of the packets in an image stream were authenticated (e.g., 80% to 100%). Depending on the threat model and the packet-checking algorithm, to detect attacks attention may need to be paid to the ratio of good to bad packets: if all of the packets were bogus, and only 80% were checked, it might be possible for the display program to show a completely bogus image, with the remaining 20%.
- An administrator may choose to run an intrusion detection system within a range rather than at a fixed level. There would be a minimal level of IDS processing below which the system would not be permitted to fall, but the IDS would be balanced against performance requirements of the organization’s tasks. Thus the IDS might perform more thoroughly (with deeper histories) when the system is lightly loaded than during peak hours. The administrator might also choose to set an upper limit to IDS performance.
- Another variable packet authentication scheme [22], would be to authenticate only a certain percentage of each packet; this might have applicability for image display, especially considering that the low order bits of each byte are not very significant, visually, in some display protocols.

The following are some example security variables, with characterizations of how they could be specified or measured:

- Strength of cryptographic algorithm, e.g., RSA, DES

  measured in terms of the work factor associated with a brute force attack

- Length of cryptographic key

  characterized by bit-length

- 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 third-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

From these examples, it is apparent that the notion of security ranges is useful and, in some cases, already evident in existing systems. Thus, we can conclude that it is reasonable to consider such ranges within the context of a QoS manager.

5 System Considerations

This section presents some observations about how variant security can be viewed in a distributed system which provides QoS support.

5.1 Security Resources, Services and Requirements

A network *system* is defined as the totality of network-accessible resources. 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 [10]. 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) [15] describes modules, each of which contain specific security mechanisms to provide some of these services.

Each resource (including security services) may embody security *requirements* regarding its use. 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.

To be general, we state that all security requirements define 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 called *security variant*.

5.2 Task Sequences

Quality of service can be provided at several levels within the overall system. The notion of translucence, by which components can adapt to changing conditions at one or more other system or network layers, results in a problem that is both horizontal, viz. distributed across the network; and vertical, viz. distributed within the stack. In the following discussion, the management of QoSS can be seen to have both horizontal and vertical interactions, depending on the implementation of the various components.

A *task* is an application invoked by a user. The task utilizes various network system services and other resources. This utilization may be intermediated by different QoS middleware mechanisms (QoSMs), such as: QoS-aware object request brokers and application servers, distributed resource management systems, and various network traffic managers. In these multiple-tiered environments, a task is invoked in a *task invocation sequence*:

- the user activates the application through some interface with an application manager (OS, browser, etc.);
- the application is intermediated by the QoSM; and
- the QoSM submits the application to the system[1].

Security requirements may be established or refined by any or all of: the user, the application, the QoSM, and the system; we call these entities *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 may use a default value. If the user chooses a range, the application may invoke itself with a particular value within that range. Similarly, the QoSM may refine the application's choice, for example, to optimize the overall *system* (user population) performance, perform load balancing, etc.

---

1. It is an implementation detail whether the QoSM returns advisory parameters to the application and the application invokes the system, or the QoSM submits the application with those parameters directly to the system. For simplicity, we assume, here, that the QoSM submits the application to the system.

5.3 Security Limits and Choices

In a task invocation sequence, the request is passed from a *previous* requirement provider 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. As with requirements, all security choices define a *choice range*, which may be unitary. Thus, each requirement provider specifies 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.

For each variant security requirement, each requirement provider may also have an explicit requirement *limit range* (again, unitary or variant) 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.

5.4 Security Range Relationships

Table 2 on page 7 shows the various limits and choices we have identified for security requirement providers in a task invocation sequence.

Table 2: Security Limits and Choices

	User	Application	Middleware	System
Choice Range provided	Yes	Yes	Yes	Service Level
Limit Range enforced	No	Yes	Yes	Yes

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.

With so many requirement ranges at different points in the sequence, how do these ranges relate to each other? The following relationships appear to be inherent in a task invocation sequence:

1. The maximum of each limit and choice range *dominates*[1] the minimum of that range.

2. Each provider's choice range must be within its own limit range. This restriction reflects the natural protocol to respect one's own limits.

3. Each choice range must be within the previous choice range 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 parameters which are divergent from (outside) the user’s request.

4. Each choice range must be within the next limit range in the sequence. This restriction means that requests which are out of bounds will be rejected.

5. The limit ranges of each provider in a task sequence must all *intersect*. This is a consequence of the need for a choice to be within the provider’s own limit, and within the next limit, as well as within the previous choice. Obviously, if two ranges in a task invocation sequence don’t intersect, there does not exist a value which could

1. For each variant security requirement there is a set of elements which are partially ordered by a “security” relation (dominates), and each range is a sub-lattice of that set such that the maximum of the range is more secure than the minimum. One range is contained “within” a second range, if and only if the max of the first dominates the max of the second, and the min of the second dominates the min of the first. For two ranges to intersect means that the maximum of each dominates the minimum of the other.

satisfy both ranges; this would disallow a task from execution.

These relationships are illustrated in Figure 1.

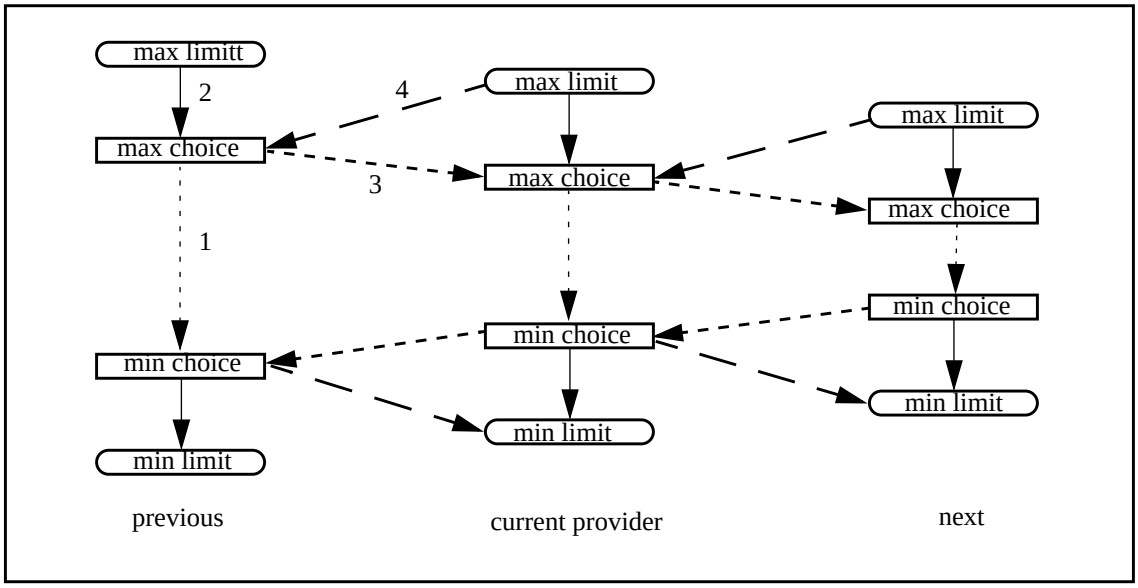

FIGURE 1. Relationships of Limits and Choices

Because the choices and limits are partially ordered and consequently comparable, it is possible for a security service selection algorithm to be encoded. A QoSM would maintain databases of static and dynamic resource characteristics. In the static database, limits might be recorded while the dynamic database could record current network conditions and choices. Thus when a new job enters the system, the QoSM can compute its execution strategy. We note that this is an NP-complete problem and extensive work exists on heuristic scheduling techniques, e.g. [18].

6 Summary

Our goal has been to provide an understanding of QoSS and variant security, and to determine whether these concepts can be useful in improving security service and system performance in QoS-aware distributed systems.

We described the general requirements for system attributes to participate in the provision of Quality of Service, and described how certain security attributes might meet these requirements. We then described various forms of user and application security “ranges” and showed how these ranges can make sense in relation to existing security policies, when those ranges are presented as user choices. Finally we described security ranges as forming a coherent system of relationships in a distributed multi-tiered system.

Our conclusion is that it may be possible for security to be a semantically meaningful dimension of Quality of Service without compromising existing security policies. Further study is needed to understand the effectiveness of QoSS in improving system performance in QoS-aware systems.

References

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

772806).

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

[3] Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A., The KeyNote Trust-Management System, version 2, RFC 2704, September 1999. ftp://ftp.ietf.org/internet-drafts/draft-blaze-itef-trustmgmt-keynote-02.txt

[4] Blaze, M., Feigenbaum, J., Ioannidis, J., and Keromytis, A., The Role of Trust Management in Distributed System Security, to appear in Secure Internet Programming: Security Issues for Mobile and Distributed Objects, ed. Jan Vitek and Christian Jensen, Springer=Verlag Inc., New York, NY.

[5] Brinkley, D.L. and Schell, R. R., Concepts and Terminology for Computer Security, in Information Security: An Integrated Collection of Essays, ed. Abrams, Jajodia and Podell, IEEE Computer Society Press, Los Alamitos, CA, 1995, pp. 40-97.

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

[7] 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

[8] Hensgen,D., Kidd, T., St. John, D., Schnaidt, M., Siegel, H.J., Braun, T., Kim, J-K, Ali, S., Cynthia Irvine, Tim Levin, Prasanna, V., Bhat, P., Freund, R., and Gherrity, M., 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., A Note on Mapping User-Oriented Security Policies to Complex Machanisms and Services, NPS Technical Report, NPS-CS-99-008

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

[11] Juneman, R. R., Novell Certificate Extension Attributes--Novel Security Attributes: Tutorial and Detailed Design. Version 0.998, Novell, Inc. 122 East 1700 St., Provo, UT, August 1997.

[12] 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.

[13] Linn, J., Generic Security Service Application Program Interface, IETF Request for Comments: 1508, September 1993

[14] 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.

[15] Sargent, R., "CDSA Explained: An Indispensable Guide to Common Data Security Architecture",The Open Group, Reading, Berkshire, UK, 1998.

[16] Schantz, R. E. “Quality of Service,” to be published in “Encyclopedia of Distributed Computing,” 1998.

[17] 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.

[18] Siegel, H..J. and Ali, S., Techniques for Mapping Tasks to Machines in Heterogeneous Computing Systems, to appear in Journal of Systems Architecture, Special Issue on Heterogeneous Distributed and Parallel Architectures: Hardware, Software and Design Tools.

[19] Stankovic, J. A., M. Supri, M., Ramamritham, K., and Buttazo, G. C., "Deadline Scheduling for Real-time Systems, Kluwer Academic Publishers, Norwell MA, 1998, pp. 13-22.

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

[21] Welch, L., Shirazi, B., and Ravindran, B., "DeSiDeRaTa: QoS Management Technology For Dynamic, Scalable, Dependable, Real-Time Systems," 15th Symposium on Distributed Computer Control Systems (DCCS’98), IFAC, Sept. 1998.

[22] Xie, G., Irvine, C., and Colwell, C., A Protocol for High Speed Packet Authentication, NPS Technical Report, NPS-CS-99-001, August 1999.