2006-00-00

Quality of Security Service: Adaptive Security

Levin, Timothy E.; Irvine, Cynthia E.; Spyropoulou, E.

Springer

Handbook of Information Security, Vol.3, pp 1016-1025, ed. H. Bidgoli, John Wiley and Sons, Hoboken, NJ, 2006, pp. 1016-1025
http://hdl.handle.net/10945/7145

Quality of Security Service: Adaptive Security

Timothy E. Levin and Cynthia E. Irvine, *Naval Postgraduate School*
Evdoxia Spyropoulou, *Technical Vocational Educational School of Computer Science of Halandri*

INTRODUCTION

The purpose of this chapter is to provide an overview, rationale, and motivation for understanding *quality of security service* (QoSS). Just as with the *quality of service* (QoS) mechanisms from which they are derived, QoSS mechanisms benefit both the subscriber (e.g., individual user or enterprise) and the overall distributed system. Users benefit by having reliable access to services, and the distributed systems whose resources are QoSS-managed may benefit by having more predictable resource utilization by users and more efficient resource allocation. Thus, the QoSS vision is to transform security from a performance obstacle into an adaptive, constructive network management tool.

Motivation

Most of today's distributed and highly populated computing environments, as exemplified by the Internet, face challenges with security as well as with the management and availability of resources. Bandwidth in mobile and wireless environments is limited; batteries have limited lifetimes; Internet service providers must be proactive to constrain high bandwidth users; and many users struggle with download times and access to networks. Every day, we are reminded in our e-mail and in various news media about the significant security vulnerabilities in our computers and networks. Network administrators have a constant battle with the configuration of firewalls, routers, and user workstations. Furthermore, resource usage patterns and the security or threat environment of these general-purpose networks are in constant flux. Intelligent, adaptive, automated mechanisms are needed to help manage network resources for availability, performance, and security.

Background

Historically, in a community of users with relatively homogeneous computing behaviors, resource usage load on multiuser systems could be understood, simplistically, to be a linear function of the number of users or the number of user terminals configured for the system.

Thus, a system administrator could govern the system resource usage load, to a large degree, by controlling the number of allowed simultaneous user input terminals (e.g., interactive terminals, modems, and card readers). In a distributed and internetworked environment, system administrators are often without recourse to such straightforward and simplistic resource-usage control approaches, because the number and type of user "terminals" and associated tasks may not be bounded by local (e.g., campus or enterprise) topographies and the resource requirements of different tasks may vary widely. In some cases, users of a given system resource may extend across the Internet. The QoS paradigm is designed to help address this problem by providing to users and administrators certain tools for managing resource usage and service levels.

QoS 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 QoS described in the literature (Chatterjee, Sabata, & Sydir, 1998; Vendatasubramanian & Nahrstedt, 1997), including accuracy, precision, and performance. For each QoS dimension, users can request or specify a level of service for one or more attributes, and the underlying QoS control mechanism (QoSM) is capable of delivering those services at the requested levels. Levels of service may be specified absolutely (e.g., megabytes of bandwidth) or statistically (e.g., 90% availability). The control mechanism must be able to modulate the level of the service to individual subscribers (e.g., users or enterprises). 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.

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 oversubscribed, all tasks would have a reduction in service. Another policy might be that no service is better than poor service, so that if resources were

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

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 (Stankovic, Supri, Ramamritham, & Buttazo, 1998). 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 QoSM 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 that the QoSM can manipulate in balancing the needs of multiple users. Given latitude in the user's soft requirements, the more of these variables that the control mechanism has to manipulate, the easier will be its job of adapting to changing resource availability and 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 that it controls and is willing to provide a range of service. For aspects in which there is no such control, only a fixed level of service can be delivered, thus adaptive QoS concepts do not apply.

QUALITY OF SECURITY SERVICE

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 research and development "request for proposals" and QoS system-design presentation slides have included a placeholder for security, without defining security as a true QoS *dimension* (as discussed in the previous section). Some of these presentations have provided access control mechanisms within the QoS framework (Sabata, et al., 1997; Lee, et al., 1998), but they have only touched on security as a QoS dimension.

Inherently, QoS involves user requests for (levels of) services that 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 premise of QoSS is that QoS mechanisms can be more effective if variable levels of *security* services and requirements can be presented to users or network tasks. In this approach, the "level of service" must be within an acceptable range, may be absolute or statistical, and can indicate degrees of security with respect to assurance (integrity, confidentiality, privacy, authentication, etc.), mechanistic strength, administrative diligence, and so forth. For example,

- security level of user $\geq$ security level of resource,
- length of confidentiality encryption key ≥ 64 and ≤ 256, and
- percentage packets authenticated ≥ 50 and ≤ 90.

As described for QoS, these ranges result in additional parameters with which the QoSM can successfully meet overall user and system demands, as well as balance costs and projected benefits to specific users/clients (see the following discussion of QoSS cost framework). Furthermore, the broader solution space with which the QoSM has to work in the security realm means that it can adapt more gracefully to dynamic changes in resource availability and thereby do a better job at maintaining requested or required levels of service in all of its dimensions. The term *quality of security service* refers 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 QoSS has the potential to provide administrators and users with more flexibility and potentially better service, without compromise of network and system security policies.

Security Ranges

For many, security is thought to be binary: either you have it or you do not. Without some minimum level of security, a system will be considered inadequate for user requirements. Yet, if a user's as well as the system's minimum requirements are met, can there not be some choice with respect to what is adequate? The answer is "yes." As an initial example, suppose that a user requires medium assurance of policy enforcement at any end system where a distributed task will be executed. If the assurance levels of potential target platforms range between medium and high assurance, there is a choice. In fact, if the medium assurance system is oversubscribed 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.

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 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 lower priced 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 or

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

scenario, why would they ever agree to more security, if it increases their cost? In general, the increase in cost will be acceptable to the user only if it is accompanied by a commensurate increase in some other associated level of service, such as

- likelihood of task completion;
- performance factors, such as latency and throughput;
- storage/output device features, such as supported media or format; and
- data features, such as color, accuracy, and precision.

In other words, more security and higher costs will be acceptable if it results in an increase or stasis in the overall satisfaction with the task invocation (see discussion of "benefit functions" in Irvine and Levin (2000), Jensen, Locke, & Tokuda (1985, and Kim, et al., (2000); thus, users could be motivated to consider security ranges above their established minimums. For example, an application may have variable data formats, which have correspondingly variable security requirements and costs. Perhaps a degraded image requires less security, and conversely, the enhanced image requires more security. Thus, a user might prefer heightened security if it is accompanied by greater image fidelity, even through the cost is higher.

An example taken from a popular military novel will help to illustrate our point. Suppose that high-, medium-, and low-resolution satellite images of enemy troop movements are available. A different type of optical equipment produces each type of image. To help keep confidential the optical technologies used to obtain the images, the images themselves are classified at high, medium, and low sensitivity levels, respectively. For analysis of enemy troops in tactical planning, any resolution image will suffice, however, the low-resolution images are from old, slow equipment, and use of high-resolution images is restricted to emergencies (here, part of the cost of using the high resolution is the need to justify its usage at a later date). Therefore, the tactical commander issues a request for troop movement images with fidelity in the range of low to high, in the following priority order: medium, low, high (such that high would be used only when the other resolutions are not available). Thus, we have a situation in which the user would prefer medium security but will also accept low or high security images, depending on what is available.

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 harm the patient. Secrecy is not a requirement, yet if the only secure communication channel available provides both a high level of secrecy and integrity, the operation is provided 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:

- Destination subnets could be classified by risk factor with respect to routing through, execution on, or logging on to nodes in those subnets (ISO, 1989). 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.
- Some environments may offer the user choices of logon authentication technology. For example, a user may log on with a standard password, a one-time password (crypto challenge–response), a public key smart card, a biometric device, 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 (Juneman, 1997).
- Some underlying systems support different *situational modes*. For some modes (e.g., normal, impacted, emergency), the user or administrator may be willing to accept more (or less) security. 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 Management System for Heterogeneous Networks (MSHN) resource management system is an example of a QoSM system in which the management of mode versus security requirement was designed to be handled automatically (Hensgen, et al., 1999; Irvine & Levin, 1999a).
- The security policy for a hypothetical commercial subnetwork 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 stakeholders 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 (Schneck & Schwan, 1998) 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 (and consequently dropped), it might be possible for the display program to show a *completely* bogus image, utilizing the remaining 20%.
- The number of "rounds" performed in a cryptographic transformation algorithm, such as the advanced encryption standard, could be used as a QoSS variable, to the extent that more rounds consume more resources and provide more security.
- An administrator may choose to run an intrusion detection system (IDS) within an effectivity *range* rather than

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

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 processing to conserve resources.

- Another variable packet authentication scheme (Xie, Irvine, & Colwell, 1999) would be to authenticate only a certain percentage of each packet. The higher percentage of authentication could be used, for example, to protect against steganographic exfiltration of sensitive data.

From these examples, it is apparent that the notion of security ranges is useful and, in some cases, already evident in existing systems.

QoSS MODEL

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

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, nonrepudiation, and traffic flow confidentiality (Irvine & Levin, 1999b). A security service typically consumes other low-level *system resources* such as central processing unit (CPU) power, memory, disk space, and network bandwidth. For example, the common data security architecture (CDSA) (Sargent, 1998) 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 mandatory and discretionary requirements or other security constraints, for example, 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*.

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 other system or network layers, results in a problem that is both horizontal, viz. distributed across the network, and vertical, viz. distributed within the layers of programs within a given network node. 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 or another task. The task utilizes various network system services and other resources. This utilization may be intermediated by different QoS middleware mechanisms, 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.
- The QoSM submits the application to the system. Note that 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.

More complex invocations are possible, such as chains of applications leading to the intermediation by the QoSM, chains of sequences, and so forth. However, the simplification of a user/application/QoSM/system model appears general enough to address the security concerns of the more complex cases.

Security requirements may be established or refined by any or all of the following: 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, and so forth.

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

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

choice from the previous provider, for example, a given application will not accept a range greater than 60–100% from the user.

Security Range Relationships

An important aspect of the notions of "range" and the relationship between ranges is how two elements of a range compare. In the packet authentication rate example, the "rate" variable is measured on a linear scale, so one choice, or element, has a natural relationship to another choice: one of greater than, less than, or equal. To be most general, we would also like to allow two choices within a range to be incomparable, resulting in a partial ordering. A natural interpretation of a security range is that the elements on one end of the range are "more secure than" the elements on the other end.

A *security requirement* is more formally defined, as follows: for each variant security requirement there is a set of elements, from which choices and limits are selected, that are partially ordered by a "security" relation. This relation is called "dominates," meaning that one element is more secure than a second if and only if the first element dominates the second. Each defined or selected security range for the requirement is a sublattice of that set such that the maximum of the range dominates the minimum. One range is contained "within" a second range, if and only if the maximum of the first dominates the maximum of the second, and the minimum of the second dominates the minimum of the first. For two ranges to intersect means that the maximum of each dominates the minimum of the other.

Table 1 shows the limits and choices of security requirement providers in a task invocation sequence.

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, because there is no next provider to whom a choice might be given. With so many requirement ranges at different points in the sequence, one may wonder how these ranges relate to each other.

The following relationships appear to be inherent in a task invocation sequence involving cooperating entities:

1. The maximum of each limit and choice range dominates 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 that 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 that 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 do not intersect, there does not exist a value that could satisfy both ranges; this would disallow a task from being successfully invoked.

Table 1 Security Limits and Choices

	User	Application	Middleware	System
Choice range provided	Yes	Yes	Yes	Service level
Limit range enforced	No	Yes	Yes	Yes

These relationships are illustrated in Figure 1. Because the choices and limits are partially ordered and thus functionally 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, whereas the dynamic database could record current network conditions (e.g., available capacities) and choices. Thus, when a new task request enters the system, the QoSM can compute its execution strategy. Note that the complexity of this computation is "NP-complete" (Knuth, 1974), and extensive work exists on heuristic scheduling techniques, for example, that by Siegel and Ali (2000).

QoSS APPLIED

In this section, two aspects of applying QoSS to distributed systems are presented: a framework for quantifying the cost of QoSS resource selections, such as would be utilized by a QoSM in making resource allocation decisions, and an examination of how a specific security mechanism can be modulated to provide differing levels of security service in response to QoSS requests.

Costing

Security comes at a cost. If a particular security mechanism is "fixed" (i.e., always applied) then the overhead for the mechanism is part of the normal cost of running the task and the normal costing mechanism used by the QoSM will suffice. For variant security mechanisms, however, the security overhead will vary depending on the user's QoS request. Some task invocations will utilize little, if any, of the variant mechanism and other invocations may utilize the mechanism at an increased level. Also, the scheduler may adapt security support, while maintaining any minimum system security policy requirements, to schedule the tasks most efficiently. The QoSM must calculate how much the use of the security mechanism will increase the cost of the task, according to the specific security "level" requested. For this reason the QoSM must have access to detailed information about the resource cost (as well as the task's requested QoS) for each variant

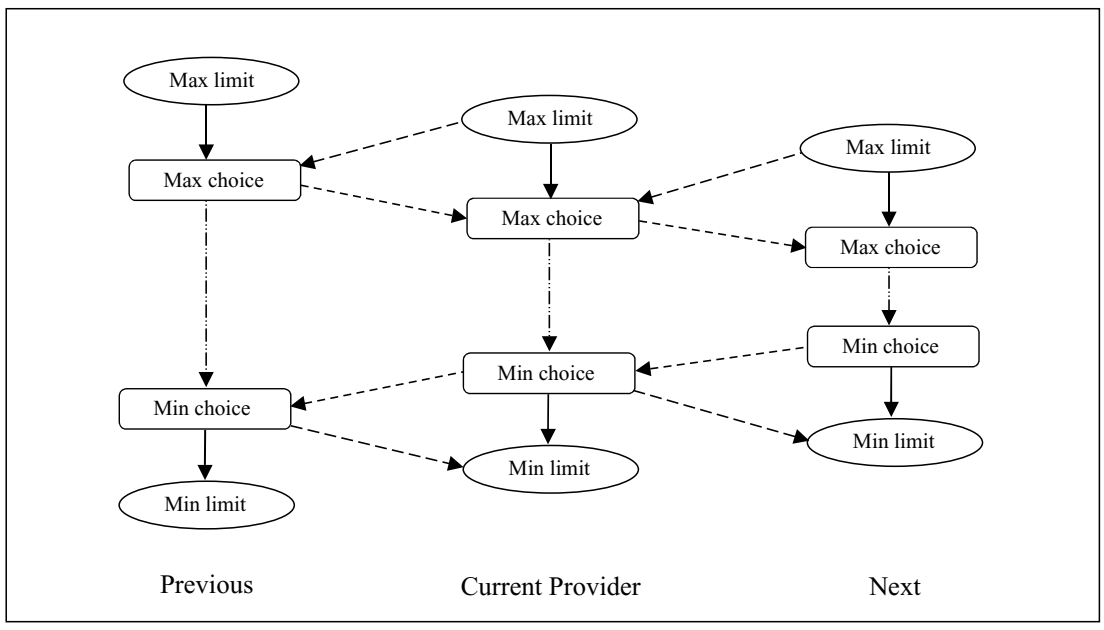

Figure 1: Relationships of limits and choices.

security mechanism. Near-optimal solution selection for task schedules depends on the accurate estimation of per-task, per-resource cost of security (Irvine & Levin, 1999a).

This approach for quantifying the costs related to a task's security requests (Spyropoulou, Levin, & Irvine, 2000) refers to costs relative to every security service invoked by the task. Each service may access CPU time, memory, bandwidth, disk space, and so forth. The resource usages may be temporary or persistent, providing discrimination between start-up and streaming costs.

For example, the Confidentiality on the Network Connection service, using a symmetric algorithm such as Twofish (Juneman, 1997) for data encryption, would require some extra processing during startup for the initialization of S boxes. This is a one-time cost during the establishment phase of the service. On the other hand, bandwidth costs for the confidentiality service are streaming costs only, in the form of extra bytes per packet because of the encryption algorithm.

In a QoS system every application would have its resource costs modeled as shown in Table 2, where cost expressions are functions of security variables. A given security variable may be a factor in more than one cost expression.

Units of measure can vary, for example, CPU costs can be measured in clocks (or clocks/packet), memory costs in bytes, and bandwidth costs in bytes (or bytes/packet). In another approach, all measures could be unitless and normalized within a common framework. A careful description of the semantics of the units with respect to each security service would then be required.

After the individual costs are calculated, their intended use is as input to a QoSM for efficient scheduling of tasks when treating security as a QoS dimension.

Examination of Resource Modulation

For security to be a real part of QoS, security choices must be presented to users, and the QoS mechanism must be able to modulate related variables to provide predictable security service levels. Consider, for example, how a specific security mechanism, IPsec, can be externally modulated to provide different levels of security in response to QoSS requests from users or the QoSM. One approach is to manage detailed variant security attributes according to abstract *network mode* and user *security level* selections.

The variables associated with many security services are too complex for average users or application developers to understand or manage without assistance. A simplified abstraction of security, in the form of security level choices such as "high," "medium," and "low," can be provided. The applications, either alone or combined with the QoSM, can then manage a mapping of security levels to detailed variant attributes, and users will not have to contend with the complexity of these choices, as shown in Table 3.

Table 2 Hypothetical CPU Cost Formulas for FTP Security

	CPU Cost Types	
FTP Security Services	**Startup (clocks)**	**Streaming (clocks/packet)**
Network integrity	a ∗ KEY_LENGTH + b	c ∗ INTEGRITY_RATE
User authentication	d ∗ e (AUTHENTICATION_TYPE) + f	0

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

Table 3 Hypothetical Security Attributes per Security Level

	Security Level Choice		
	Low	**Medium**	**High**
Packet integrity rate	0.6	0.8	1
Symmetric key length	56	96	128
Type of authentication	None	Password	Biometric & password

Table 4 Hypothetical IPsec Security Attributes per Security Level

	Security Level		
Channel	**Low**	**Medium**	**High**
telnet	No IPsec processing	ESP processing with DES	ESP processing with 3DES
finger	No IPsec processing	AH processing with HMAC-MD5	AH processing with HMAC-SHA
ping	No IPsec processing	No IPsec processing	No IPsec processing

Similarly, abstract policies regarding system limits for different situational modes and changing environments can be provided. For example, a military network or Internet service provider might have an "emergency" mode that indicates that there is a physical threat to the facility, or "congested" mode in which there is more traffic than the current capacity. In both of these cases, the system limits (and choices) for variant security attributes might be different than for a "normal" mode. Again, the applications, QoSM, and/or system could maintain a mapping from the abstract modes to specific security attribute settings, so that once the mapping has been established, users and administrators would not have to contend with the complexity of these choices.

Two entities that wish to communicate with each other using IPsec negotiate to establish IPsec *security associations* (SAs), which define the detailed security characteristics of the communication channels to be used, such as encryption and/or authentication algorithm, SA lifetime, and cryptographic keys. These SAs are used until their negotiated lifetime expires (assuming that no interruption or discard of the channel takes place). The two main issues for utilizing IPsec in a QoSS framework are as follows:

- The SAs should conform to the settings represented by the abstract QoSS parameters. The set of SA characteristics resulting from an SA negotiation should reflect the mappings determined for "network mode" and "security level."
- If there is a change in a QoSS parameter, currently active SAs should be stopped and renegotiated to conform to the new security mappings.

Although the usual IPsec management mechanisms have no provision for response to changing external requirements for a given communication channel, its framework is adaptable enough to allow for the dynamic management of SAs, such as would result from changes to network mode or security level selections.

One approach (described in detail in Spyropoulou, Agar, Levin, & Irvine, 2002) is to use a *trust management system* (Blaze, Feigenbaum, Ioannidis, & Keromytis, 1999) and an external *dynamic parameter console* to modulate the detailed IPsec security attributes. The dynamic parameter console interacts with the external environment, such as an administrator or an intrusion detection system, to receive the current selections for the network mode and security level, and to pass the selections to the trust management system. The trust management system keeps track of the current abstract settings, defines the mapping of levels and modes to detailed IPsec settings, and also acts as policy arbiter regarding allowed IPsec settings and connections (see Table 4). IPsec and the trust management system are instrumented to perform renegotiation of SAs when the abstract settings change.

If the network mode changes to reflect a modification in the system status, or if the user level changes to indicate a desire for higher security, then the dynamic parameter console notifies the trust management system and IPsec. The trust management system is updated with the new policy information for use with new SA negotiations. Furthermore, if there exist currently active SAs based on the changed policies, it removes them and initiates a renegotiation of those SAs.

QoSS AND APPLICATION-CENTRIC SECURITY

The historical view of access control to resources was OS-centric with respect to the activities of applications and other programs that the OS hosted. 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" (Brinkley & Schell, 1995). 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 (e.g., see Blaze, Feigenbaum, Ioannidis, & Keromytis, 1999). 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 application's objects are completely encapsulated, such that the object never leaves the control domain of the application, 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. (Note that if the object is allowed to leave the application's domain, then it is more difficult to

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

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, such that the object remains, logically, in the control domain of the application.)

Another aspect of traditional OS policy enforcement was the notion that, to be considered highly effective, access control should be performed at the lowest layers(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 unable to be bypassed, persistently enforced, and small enough to allow thorough analysis (e.g., see Anderson, 1972). Without understanding the dependency layering, it will not be clear on which other modules a module depends, nor will it be clear if there are fatal (e.g., circularly dependent) or semantically undefined execution sequences.

In the OS paradigm, 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). For example, a network application, which has been allocated the responsibility for enforcing a security policy, typically depends on an untrusted operating system for access to resources, and dependency layering is not a fundamental design consideration in many modern distributed applications and systems.

Under the conditions described here for distributed systems, 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, nondistributed, layered system. The QoSS approach may be applicable in certain distributed systems that utilize application-centric access control concepts. This is not to say that this approach ameliorates the design analysis problems of application-centric access control. On the contrary, it is important to reiterate that each such system needs careful design review to understand the effectiveness of its security mechanisms. It is hoped that the security abstractions presented here will aid in such analyses.

RELATED WORK

The *OSI Basic Reference Model Security Architecture* document (ISO, 1989) provides information and analysis about network communications security services and mechanisms, including a mapping of security services to mechanisms and OSI layers, and describes the behavior of lower layers when responding to security service requests. This analysis provides a good summary of network security services from the perspective of protection of communications. The QoSS approach is intended to include security services other than those specific to communications protection, and the QoSS service model is more oriented to the *n*-tier architectural framework rather than the OSI protocol stack. Additionally, the OSI work does not address the constructive management of security variability.

A *quality of protection* parameter is provided in the GSSAPI specification (Linn, 1993). This parameter is intended to manage the level of protection provided to a message communication stream by an underlying security mechanism (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 (1998), which discusses variable packet authentication rates with respect to the management of system performance. The QoSS work presented here 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 Chatterjee, Sabata, and Sydir (1998), Aurrecoechea, Campbell, and Hauw (1996), and Welch, Shirazi, and Ravindran (1998), 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 the reader is referred to Aurrecoechea, Campbell, and Hauw (1996) for a thorough review of QoS definitions and architectures.

A *trust management system* (Blaze, M., Feigenbaum, J., Ioannidis, J., & Keromytis, 1999; 2001) 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) that 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.

CONCLUSION

This chapter provides an overview of QoSS and variant security and demonstrates that these concepts can be useful in improving security service and system performance in QoS-aware distributed systems. The general requirement for system attributes to participate in the provision of QoS, as well as an approach for how certain security attributes might meet these requirements, is described. Various forms of user and application security "ranges" are shown to make sense in relation to existing security policies when those ranges are presented as user choices.

It is evident that for a distributed multitiered system, security ranges can form a coherent system of relationships and security can be a semantically meaningful dimension of QoS without compromising existing security policies. Further study is needed to understand the effectiveness of QoSS in improving system performance in QoS-aware systems.

GLOSSARY

Quality of Service Dimension A response attribute of a distributed system that a user or program can request or specify, and that the system is capable of delivering at or near those requested levels. Examples are bandwidth and response time.

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

Quality of Security Service Refers to the use of security as a quality of service dimension.

Security Service A high-level abstract resource providing security functionality such as authentication, auditing, privacy, integrity, intrusion detection, nonrepudiation, and traffic flow confidentiality. A security service typically consumes other low-level system resources and may be implemented by one or more security mechanisms.

Variant Security The ability of security mechanisms and services to allow the amount, kind, or degree of security to vary, within predefined ranges.

CROSS REFERENCES

See *Auditing Information Systems Security; Computer and Network Authentication; Security and Web Quality of Service; Security Policy Guidelines.*

REFERENCES

Anderson, J. P. (1972). *Computer security technology planning study* (Technical Report ESD-TR-73-51). Bedford, MA: Air Force Electronic Systems Division. Hanscom AFB. (Also available as Vol. 1, DITCAD-758206; Vol. 2, DITCAD-772806.)

Aurrecoechea, C., Campbell, A., & Hauw, L. (1996). A survey of quality of service architectures. *Multimedia Systems Journal*, 6(3), 138–151.

Blaze, M., Feigenbaum, J., Ioannidis, J., & Keromytis, A. (1999, September). *The KeyNote Trust-Management System version 2* (RFC 2704). Retrieved April 7, 2005, from http://ietf.org/rfc/rfc2704.txt?number=2704.

Blaze, M., Feigenbaum, J., Ioannidis, J., & Keromytis, A. (2001). The role of trust management in distributed system security. In J. Vitek & C. Jensen (Eds.), *Secure Internet programming: Security issues for mobile and distributed objects* (pp. 185–210). New York: Springer-Verlag.

Brinkley, D. L., & Schell, R. R. (1995). Concepts and terminology for computer security. In M. Abrams, S. Jajodia, & H. Podell (Eds.), *Information security: An integrated collection of essays* (pp. 40–97). Los Alamitos, CA: IEEE Computer Society Press.

Chatterjee, S., Sabata, B., & Sydir, J. (1998). *ERDoS QoS architecture* (SRI Technical Report ITAD-1667-TR-98-075). Menlo Park, CA: SRI International.

Hensgen, D., Kidd, T., St. John, D., Schnaidt, M., Siegel, H. J., Braun, T., Kim, J.-K., et al. (1999). An overview of the management system for heterogeneous networks (MSHN). In V. Prasanna (Eds.), *Proceedings of the Heterogeneous Computing Workshop* (pp. 184–198). Los Alamitos, CA: IEEE Computer Society.

International Organization for Standardisation (ISO; 1989). *Information processing systems—open systems interconnection—basic reference model—part 2: Security architecture*. International Standard, no. 7498-2. Switzerland.

Irvine, C., & Levin, T. (1999a). *A note on mapping user-oriented security policies to complex mechanisms and services* (NPS Technical Report NPS-CS-99-008). Monterey, CA: Naval Postgraduate School.

Irvine, C., & Levin, T. (1999b). Toward a taxonomy and costing method for security metrics. *Proceedings of the Annual Computer Security Applications Conference* (pp. 183–188. Los Alamitos, CA: IEEE Computer Society.

Irvine, C., & Levin, T. (2000). Toward quality of security service in a resource management system benefit function. Proceedings of the Heterogeneous Computing Workshop (pp. 133–139). Los Alamitos, CA: IEEE Computer Society.

Jensen, E., Locke, C., & Tokuda, H. (1985). A time-driven scheduling model for real-time operating systems. In *Proceedings of the IEEE Real-Time Systems Symposium* (pp. 112–122). Los Alamitos, CA: IEEE Computer Society.

Juneman, R. R. (1997). Novell certificate extension attributes. In *Novel security attributes: Tutorial and detailed design, version 0.998*. Provo, UT: Novell.

Kim, J., Hensgen, D., Kidd, T., Siegel, H., St. John, D. Irvine, et al. (2000). A QoS performance measure framework for distributed heterogeneous networks. In *Proceedings of the Eighth Euromicro Workshop on Parallel and Distributed Processing* (pp. 18–27). Rhodes, Greece.

Knuth, D. E. (1974). A terminological proposal, SIGACT News *6*(1), 12–18. ISSN:0163-5700. New York, NY: Association for Computing Machinery.

Lee, C. A., Stepanek, J., Michel, B. S., Kesselman, C., Lindell, R., Sonnwook Hwang, et al. (1998). The quality of service component for the Globus metacomputing system. In *Proceeding of the 1998 International Workshop on Quality of Service* (pp. 140–142). Los Alamitos, CA: IEEE Computer Society.

Linn, J. (1993). *Generic security service application program interface* (RFC 1508). Retrieved April 7, 2005, from http://ietf.org/rfc/rfc1508.txt?number=1508.

Sabata, B., Chatterjee, S., Davis, M., Sydir, J., & Lawrence, T. (1997). Taxonomy for QoS specifications. In *Proceedings of the Third International Workshop on Object-Oriented Real-Time Dependable Systems* (pp. 100–107). Los Alamitos, CA: IEEE Computer Society.

Sargent, R. (1998). *CDSA explained: An indispensable guide to common data security architecture.* Reading, Berkshire, UK: The Open Group.

Schneck, P. A., & Schwan, K. (1998). *Dynamic authentication for high-performance networked applications* (Technical Report GIT-CC-98-08). Atlanta, Georgia: Georgia Institute of Technology College of Computing.

Siegel, H. J., & Ali, S. (2000). Techniques for mapping tasks to machines in heterogeneous computing systems. *Euromicro Journal of Systems Architecture, 46*(8), pp. 627–639.

Spyropoulou, E., Agar, C., Levin, T., & Irvine, C. (2002). IPsec modulation for quality of security service. In *Proceedings of the International System Security Engineering Association Conference*. Herndon, VA: International System Security Engineering Association.

Spyropoulou, E., Levin, T., & Irvine, C. (2000). Calculating costs for quality of security service. In *Proceedings of the 16th Computer Security Applications Conference* (pp. 334–343). Los Alamitos, CA: IEEE Computer Society.

Stankovic, J. A., Supri, M., Ramamritham, K., & Buttazo, G. C. (1998). *Deadline scheduling for real-time systems* (pp. 13–22). Norwell MA: Kluwer Academic Publishers.,

P1: NFR
JWBS001C-205.tex WL041/Bidgoli WL041-Bidgoli.cls April 22, 2005 20:8 Char Count= 0

Vendatasubramanian, N., & Nahrstedt, K. (1997). An integrated metric for video QoS. *Proceedings of the ACM International Multimedia Conference* (pp. 371–380). New York, NY: Association for Computing Machinery.

Welch, L., Shirazi, B., Ravindran, B. & Bruggeman, C. (1998). DeSiDeRaTa: QoS management technology for dynamic, scalable, dependable, real-time systems. In *Proceedings of the 15th Symposium on Distributed Computer Control* Systems (pp. 7–12). Internation Federation of Accountants, New York, NY.

Xie, G., Irvine, C., & Colwell, C. (1999). *A protocol for high speed packet authentication* (NPS Technical Report NPSCS-99-001). Monterey, CA: Naval Postgraduate School.