IPsec Modulation for Quality of Security Service

Evdoxia Spyropoulou Chris Agar Timothy Levin Cynthia Irvine
Naval Postgraduate School
{espyropo, cdagar, levin, irvine}@nps.navy.mil

Abstract

This paper discusses the modulation of security services in response to changes in network conditions or as a result of modified user or application security requirements. First, the notion of security variability and how security can be treated as a dimension of Quality of Service in distributed systems is described. We discuss how security choices presented to users or applications and limits on these choices can be defined and managed through dynamic network policies. A costing framework for managing resource utilization costs due to variant security is presented. And finally, we provide an analysis of how a specific security mechanism can be modulated to provide differing levels of security service in harmony with Quality of Security Service requests and we describe our proof of concept demonstration for such modulation with respect to IPsec.

1. Introduction

Quality of Service (QoS) mechanisms can be beneficial to both the user and the overall distributed system. QoS users benefit by having reliable access to services. The distributed system whose resources are QoS managed benefits by having more predictable resource utilization and, where supported, more efficient resource allocation. We have previously examined how reliability, predictability and efficiency can be enhanced by including security as a real part of QoS, transforming security from an inflexible performance obstacle into a constructive management tool. We have termed the effects of this inclusion, "Quality of Security Service" (QoSS) [10]. The motivation for the work described here is to examine specific methods for managing variability of security services, including choices offered to users, resource cost calculation, and modulation of underlying security mechanisms.

Security variability has been discussed in earlier work. Variable packet authentication rates have been discussed with respect to the management of system performance [17]. A Quality of Protection parameter, which manages the level of protection provided to a message communication stream by an underlying security mechanism, is presented in the GSS-API specification [15]. In our work we are trying to provide a more general framework, applicable to a wide range of policy, processing and networking contexts, as well as diverse security services. We have presented a preliminary security service taxonomy defining the range of security services a QoS mechanism may need to manage [8]. We have also addressed the issues of user interaction with this wide range of security services, and resource cost calculation for variant security [9][18].

For security to be a real part of QoS, security choices must be presented to users. This paper discusses how the limits on these choices can be defined, how we can relate those limits to security policies that can change dynamically, and what the impact of these choices is on resource costs.

Furthermore, for provision of QoSS, the QoS mechanism must be able to modulate related variables to provide predictable security service levels to the users. IPsec is a security mechanism that offers choices for the characteristics of the security services it provides. A trust management system [1][2] can be used in conjunction with the IPsec mechanism to provide policy management for protected traffic in security endpoints [3]. In this paper we present how a QoSS system could be built to utilize a trust management system to store and resolve security range relationships for IPsec and provide variant security according to user QoSS selections.

The remainder of this paper is organized as follows: we give an overview of our ideas for Quality of Security Service in Section 2. Section 3 discusses further the issue of variability in security services and how notions such as network operational modes and security level choices can be addressed. This section also includes a discussion about our framework for calculating the resource costs due to variant security. In Section 4 we illustrate how IPsec can be modulated to respond to QoSS requests and a summary follows in Section 5.

2. Quality of Security Service Background

Quality of Service refers to the ability of a distributed system to provide network and computation services in a way that each user's expectations for timeliness and performance quality are met. Various dimensions of QoS, like accuracy, precision and performance have been described [4][20]. If a Quality of Service dimension is supported, then a user can request a level of service for one or more attributes of this dimension and the underlying QoS control mechanism is capable of entering into agreement to deliver those services at the requested levels.

Users present to the QoS mechanism service level requests, which can be hard and soft requirements [19]. A hard requirement involves fixed service levels that the QoS mechanism must deliver. Soft requirements on the other hand can be seen as describing a range of acceptable service, a variable that the QoS mechanism can manipulate, in order to satisfy the set of current users. The QoS mechanism can offer choices to the user only for aspects of the system over which it controls, and is willing to provide, a range of service (the users would then formulate their hard or soft requests in response to these choices).

Historically, security has been handled rather statically and indirectly compared to traditional QoS attributes (e.g. jitter, deadline, latency). The QoS mechanism could be more effective if variable levels of security services and requirements can be presented to the users or network tasks, providing security choices within acceptable ranges. These ranges result in additional parameters that the underlying system can modulate to successfully meet overall user and system demands.

We use the term Quality of Security Service (QoSS) to refer to the use of security as a quality of service dimension. We have developed a theory of QoSS and a related security-costing framework that supports extension of QoS functionality to embrace existing and emerging security technologies [10][18]. Our goals have been to leverage existing security mechanisms to improve system performance, while maintaining, if not increasing the security of the distributed system.

Variability in security requirements is fundamental to QoSS, and has two distinct characteristics. Variability in *user and application* security requirements allows the underlying control system to be more adaptable in responding to requests for resources, and variability in *system and resource* security requirements allows the distributed system, e.g., through QoS middleware, to offer security choices to users or applications. The availability of user security choices along with support for management of security resources in response to user requests enables QoSS.

Several existing mechanisms and policies allow for security variance. Many of the so-called *fixed* requirements can be seen to actually define only minimums, allowing for a range of solutions. Some examples of security service attributes that provide ranges are the choice of cryptographic algorithm, number of rounds or key length, assurance level or strength of boundary control in a remote environment, or even the capability level of the environment's security administrators.

An example of usage of security ranges could be an environment that offers the user choices of log-on authentication technology. 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 such an environment the user could be granted greater access to resources if he uses higher-assurance authentication [11]. Another example is an intrusion detection system (IDS) which can be run within an effectivity range rather than at a fixed level. There would be a minimal level of IDS processing below which the system would not be allowed to fall, but the IDS would be balanced against performance requirements of the organization's tasks. Thus the IDS might perform with deeper histories when the system is lightly loaded than during peak hours.

QoS can be seen as the modulation of resources to deliver requested services to users, which depends on the control and variability of resources. Similarly, QoSS involves the modulation of security resources, and depends on the control and variability of those security resources. In a typical distributed system, the security restrictions and requirements confronted by a user emanate from many layers, components and services. How can QoS or resource management middleware make sense out of this apparent chaos in attempting to manage the system efficiently? Our approach involves several abstractions: the first is to view all security restrictions as service attributes. The second is to view all security restrictions as a range that defines a set of partially ordered possibilities, where some values are "more secure" than others. Of course, in some cases the range maybe unitary or degenerate, in which case it represents no choice and the related service can be used in only one way.

To understand how these ranges can be used in a layered distributed architecture, consider how a request for execution of a task is passed between different layers, and security services are provided in response to these requests. As this *task sequence* is processed, there are both choices and limits regarding each security restriction or requirement. A *choice* is the security range request passed to the next layer. A *limit* defines which requests from previous layers are acceptable. In the end, if the task is realized by the system, meaning that the various choices and limits have been successfully resolved, the user's expectations for QoSS will have been met. Additionally, the QoS middleware will have had additional latitude, by way of variant security requirements, in fulfilling user and system-wide goals, thereby potentially increasing the availability, predictability, and efficiency of the system.

3. Managing Variability and Costs of Security Services

In this section we will describe briefly our QoSS costing framework that incorporates the ideas of variant security services [18].

A task on the network is presented with various security services: this means that during its execution it may utilize certain security services. A *security service* is a high-level abstract resource providing security functionality such as: authentication, auditing, privacy, integrity, intrusion detection, non-repudiation and traffic flow confidentiality. A security service typically consumes other low level system resources, such as CPU, memory, disk, network bandwidth, and may be implemented by one or more security mechanisms.

Each security service may embody security requirements regarding its use, imposed by the network security policy. This body of rules for resolving network security issues can be decomposed typically into functional requirements. These requirements might be the typical MAC and DAC requirements, or other security constraints, e.g.: encryption available 9pm to 5am, available encryption algorithms, and required key lengths. As we described in the previous section, we view all security requirements as defining a range of permissible behavior. So every task is associated with a vector of security requirements related to the security services that the task invokes.

3.1 User Choices for Security Levels

The security requirements necessitated by the network security policy demand some minimum levels of security be applied for each task, and also indicate the maximum security levels that can be provided by the system (e.g., if only through resource limitations). Selections for QoSS may be provided to users to any degree of security within these limits. Thus, a system can always provide more security, at the user's discretion, than the minimum required by the base security policy, while still complying with that policy.

Still, the complexity of the available variables and the choices associated with the security services and underlying mechanisms, maybe too high for the users or applications to manage without automated support. We can offer a simplified abstraction of security to the user, in the form of security level choices, like "high", "medium", "low". This way the user is not offered all combinations of security parameters for the variant services.

In this approach, the security administrator or system security engineer would pre-set *translation* matrixes, through which the security levels offered to the user are mapped to detailed mechanism invocations. A security level can be mapped to a sub-range within the acceptable range of values for the variant security attribute, or could be mapped to a specific value. In either case the QoS mechanism would be responsible for subsequently assigning security services and resources to the user that would meet the security profile indicated by the translation matrix.

3.2 The Notion of Network Modes

Under different environment conditions we may be willing to accept more (or less) security for a given application. A business executive in a foreign country known for industrial espionage might require strong security for his communications with the headquarters, whereas a company trying to get a proposal submitted within a deadline might decide to forgo security altogether and just get the information out the fastest possible. These decisions change the security policy, but the actual policy arrived at, through an ad hoc decision process, may not be clearly understood.

With a dynamic security policy, the security restrictions and available security policies can be examined with respect to the overall policy, prior to being fielded. This way the network can support different situational modes and respond to changing environments, by having access to a predefined set of alternate security policies. For example, a military network might have a "crisis" mode indicating that there is a physical threat to the facility, and that command messages which would normally be encrypted and signed need to go out with the highest bandwidth available, disregarding cryptographic security. In such a case, the effects of changes to the security mechanisms would be predefined and limited to meet the desired alternate security policy.

We refer to three example modes: *normal*, *impacted* (e.g., a system can enter this mode when it receives a large amount of simultaneous requests and for efficiency curtails certain optional security services) and *emergency* (in which strong security is required without many choices offered).

Network modes provide alternate mappings for the security requirements or limits to be enforced by the underlying system. The acceptable range of values for a security variable attribute depends on the network “mode” and the selections offered to the users and applications must be within the limits of the mode. Furthermore, network modes can provide alternate mappings for the security levels offered to the users, since "high" security would be translated in a different way if the system is in normal or in emergency mode.

Ranges for different modes can have overlapping values, or certain values may be completely excluded from the permissible set of values of any mode for a security variable. Also, the range need not be continuous (e.g. there could be sets of discrete values).

3.3 Costing for QoSS

Costs in the form of monetary charges (unlimited bandwidth but at a high cost per byte) or performance degradation (for high resolution, processing and downloads times will be long) will influence users' choices for security levels. When cost is very high (e.g., slow response time or image display), users may be willing to accept security (or imagery) that is less than their ideal level of service.

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 QoS control mechanism will suffice. For variant security mechanisms, however, the security overhead will vary, depending on the security vector of the user’s QoS request and any subsequent refinement of the user's choices due to the application or QoS mechanism. To make the appropriate QoSS decisions the underlying system must have access to detailed information about the resource costs for each variant security mechanism.

In our approach, we use a costing framework with cost expressions relative to every security service invoked by the task and its variant security attributes. Each service may access various resources, e.g. CPU, memory and bandwidth. We discriminate between start-up and streaming costs, since the utilization of a resource can be persistent through the task's execution or occur only during a set-up phase. The calculated costs can then be fed to the middleware QoS mechanism for use in its resource allocation and scheduling decisions.

4. Modulation of IPsec

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 to those users. As a proof of concept we want to demonstrate how a specific security mechanism, IPsec, can be modulated to provide different levels for security in response to QoSS requests from users. The following sections give an overview of IPsec and its architecture and describe our approach for modulating IPsec's variant security attributes according to network mode and security level selections.

4.1 IPsec Background

By definition, IP packets inherently have no security. It is therefore simple to spoof, modify, and inspect IP packets without authorization from the sender. IPsec was developed to address this problem by providing a per packet security mechanism that protects datagrams at the network layer. IPsec allows a system to select the required security protocols, determine the algorithms to be authorized for a specific service, and utilize cryptographic keys required by the services. IPsec can protect host-to-host, gateway-to-gateway or host-to-gateway / gateway-to-host packet communications [5].

IPsec provides the following security services that may be combined to meet specific requirements: origin authentication, data integrity authentication, data confidentiality, anti-replay protection and limited traffic flow confidentiality. It uses two methods to protect IP packets: Encapsulating Security Payload (ESP) and Authentication Header (AH). ESP provides data integrity, confidentiality and anti-replay protection. AH provides only data integrity and anti-replay protection (Figure 1).

IPsec additionally implements two modes of packet protection: transport mode and tunnel mode. Transport mode is used to protect upper-layer protocols for host-to-host communications: an IPsec header is inserted between the IP header and the upper layer protocol header. Tunnel mode is used to protect the entire IP packet for gateway communications: entire IP packet is encapsulated in another IP datagram and an IPsec header is inserted between the outer and inner IP headers (Figure 2).

IPsec is fundamentally built around the concept of Security Associations (SA). SAs are security service specifications that define security attributes to be used during communications between peers. More formally, a SA is a "simplex connection that affords security services to the traffic carried by it" and it essentially is "a management construct used to enforce a security policy in the IPsec environment" [12]. There is a set of parameters associated with each SA, which includes, among others: SA lifetime, encryption and/or authentication algorithms and keys, and protocol mode (tunnel/transport).

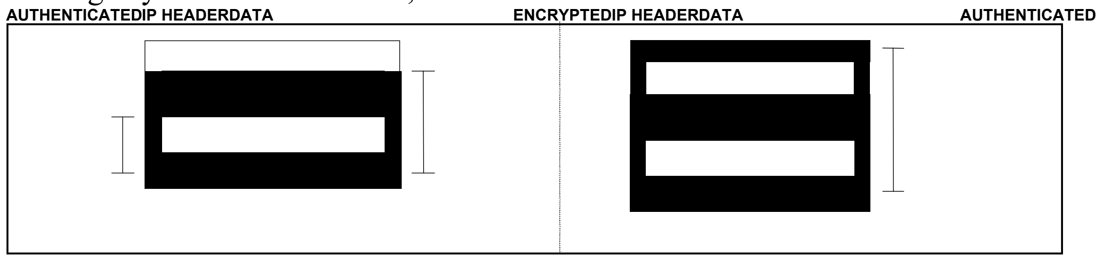

Figure 1: Encapsulated Security Protocol vs. Authenticated Header Protocol

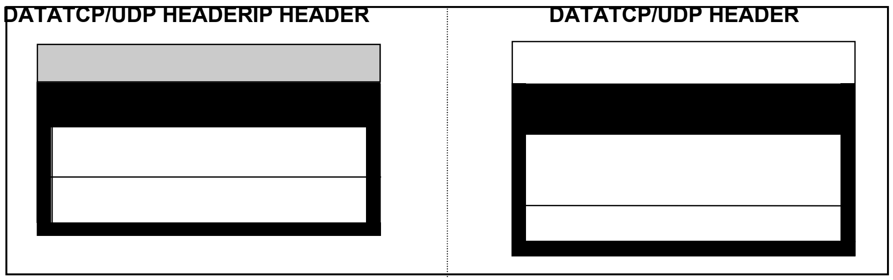

Figure 2: Tunnel Mode vs. Transport Mode

The established SAs are stored in the Security Association Database (SAD). The SAs can be generated manually, but that approach does not scale well. The Internet Key Exchange (IKE) along with the Internet Security Association and Key Management Protocol (ISAKMP) address the problem of establishing and maintaining SAs through the use of an automated daemon. IKE establishes shared security parameters and authenticated keys between communicating peers, it negotiates SAs for peer IPsec communications and it dynamically populates the SAD. IKE uses the ISAKMP format for negotiation messages with the peer. ISAKMP defines the method of peer negotiations including message format and state transitions [7][16].

Rules that define communication authorizations between peers based on SAs, i.e. what action should be taken on a per-packet basis (e.g., packet should be dropped, requires IPsec protection, IPsec protection expected, etc…), are stored in the Security Policy Database (SPD). The SPD must be able to distinguish between traffic requiring IPsec protection, traffic protected by IPsec protection and traffic allowed to by-pass IPsec security mechanisms. If a policy exists for a packet in SPD but no SAs currently are found in the SAD, a negotiation process must be initiated with the remote peer to create valid SAs [12].

IPsec provides protection for packets against tampering and eavesdropping, whereby protection is either granted for a peer-to-peer communication or bypassed. However, IPsec does not include mechanisms for specifying more granular security policy issues like which hosts are authorized for sessions with certain other entities or whether hosts are authorized to exchange specified kinds of traffic.

4.2 IPsec and Trust Management

We are currently using OpenBDS's implementation of IPsec [13][6]. This implementation addresses the issue of managing the policies that control which host is allowed to establish SAs with another host and what kind of characteristics the SAs should have, by including a trust management system, KeyNote and ensuring through it, that newly created SAs agree with a local security policy.

KeyNote is a simple and flexible trust-management system that provides a standard interface used by applications to determine if proposed potentially dangerous actions comply with the local security policies. It is formally characterized by actions, principals, policies, credentials and a compliance checker. KeyNote is composed of assertions which are small highly structured programmable predicates that limit the actions principals are authorized to perform. Simply put, KeyNote determines whether SAs between specific peers (including characteristics like encryption algorithms, keys, and communication protocol) comply with local security policy. A distinction should be made between SPD

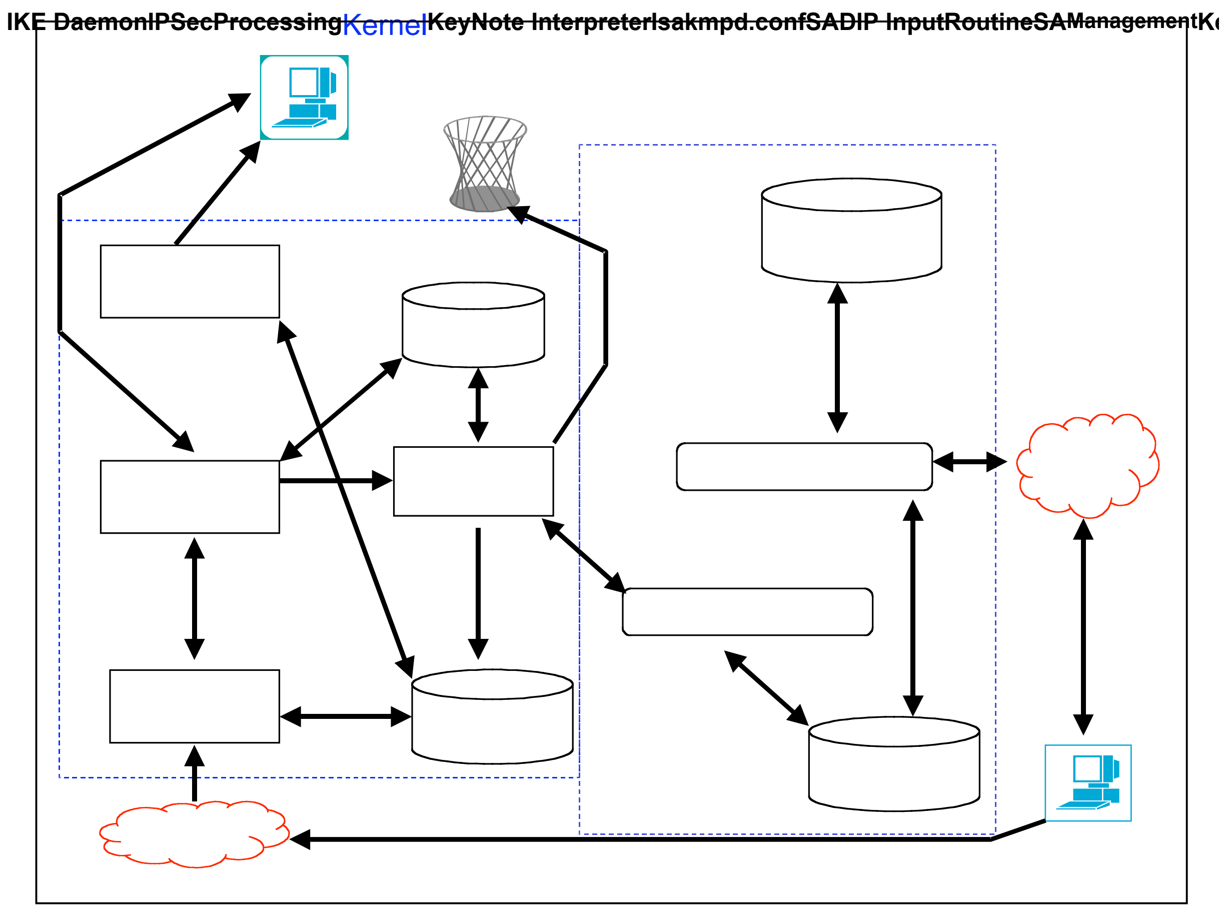
IKE DaemonIPSecProcessingKernelKeyNote InterpreterIsakmpd.confSADIP InputRoutineSAManagementK

Figure 3: IPsec and Trust Management

and KeyNote policy. SPD policy (per packet basis) is a subset of KeyNote policy (peer to peer basis). For performance reasons, SPD is used as a kernel cache for packet policies [3].

When peers negotiate security communication parameters, KeyNote is referenced on both sides to verify compliance with local security policies. SAs will be dynamically created for authorized communication and inserted into the SAD. Unauthorized communications will be refused.

The basic data packet flow through a KeyNote-controlled IPsec system implemented on OpenBSD is as follows:

Output processing: A packet arrives from a high-level protocol. The SPD is consulted to determine if the packet requires IPsec protection. If protection is not required, the packet is forwarded to the external network. If IPsec protection is required, the packet is forwarded to the IPsec processing module, where the SAD is consulted for SA specifics for the packet. If an SA exists for the packet, the appropriate security transformations are applied to the packet and it is forwarded to the external network. If no SA exists, the SA management module is triggered. The SA management module consults the SAD to reverify that no SA exists. If one does, the packet is dropped. The KeyNote database is consulted using packet information via the KeyNote interpreter to determine if the packet should be accepted, dropped, or needs IPsec protection. In the event of the requirement for IPsec protection, the IKE daemon consults its configuration data (from

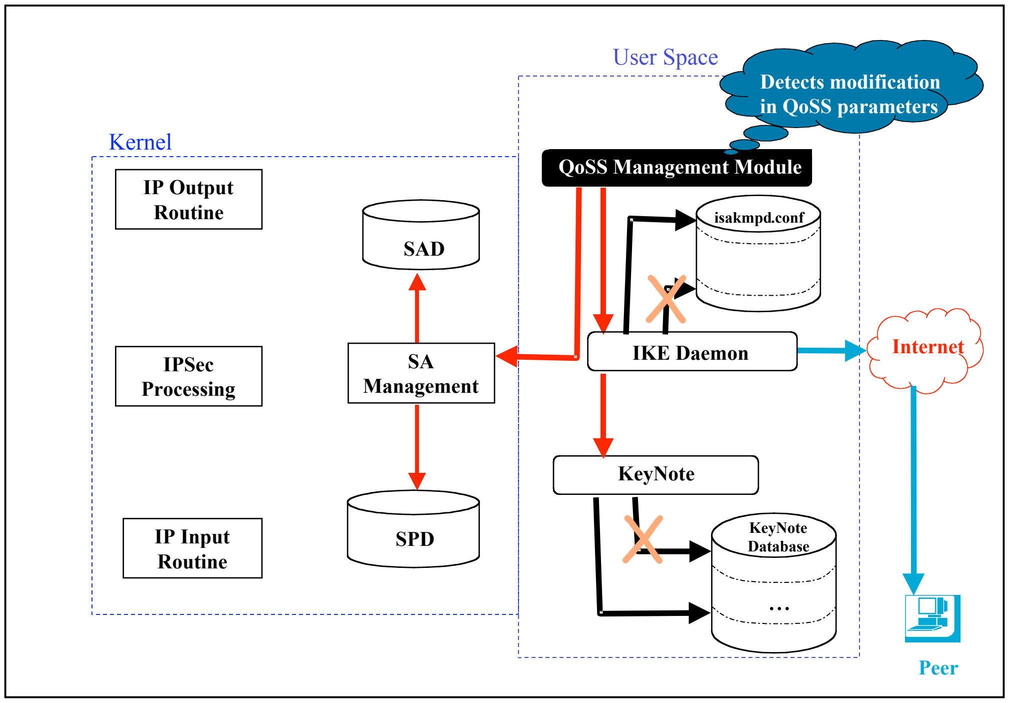

Figure 4: QoSS awareness for IPsec

the isakmpd.conf file) for remote peer communication parameters and forms the proposals it will send to the peer. IKE then negotiates security parameters with the remote peer. The remote peer will consult its own KeyNote Policy and reply with a proposed setting of secure communication parameters. The KeyNote database will be consulted again to ensure that the parameters presented by the remote peer comply with local policy. If both systems are in agreement, an SA will be created, updating the SAD and SPD. At completion the original packet is discarded [3]. These interactions can be seen schematically in Figure 3.

Input processing: Input processing is similar to output processing except that the SPD is consulted to determine whether the packet should be forwarded for further processing, by-pass IPsec input process or be dropped. If the packet is forwarded for further processing, the SAD is consulted to decapsulate the packet. If no current SA exists, the SA management module is initiated as in the output processing.

4.3 QoSS Awareness for IPsec

Two entities that wish to communicate with each other using IPsec negotiate through IKE/ISAKMP and KeyNote and establish IPsec SAs, as described previously. These SAs are used until their negotiated lifetime expires (assuming that no violent interruption or discard of SAs takes place). There is no provision for a response to changing operational conditions: the characteristics of the negotiated SAs cannot respond to dynamic modifications of the environment's security requirements, for example they cannot adapt to changes in threat conditions, critical time transmissions, and network congestion/traffic.

There are two main issues that need to be handled if we want to provide QoSS awareness for IPsec:

- the SAs that the IKE daemon negotiates, and the local policy for them enforced by the Trust

Management System, should be in accordance with the system's QoSS parameters: this means that if there is a change, for example, in a "network mode" or a "security level" parameter, the set of SA characteristics we are willing to propose and accept should reflect these changes.

- Furthermore, if there is a change in a QoSS parameter, currently active SAs should be renegotiated to conform to the current set of security requirements (as expressed by the local policy).

Using the foundation that the IPsec trust management architecture described in the previous paragraph provides, we apply our QoSS ideas by using the scheme shown in Figure 4. The QoSS Management Module, which we have added, is responsible for fetching the current selections for the network mode and the security level. The SA settings proposed (IKE daemon's configuration file) and the local KeyNote policy file are selected based on current QoSS selections.

If the network mode changes to reflect a modification in the system status, or if we just want to execute the same application but with higher security, then the QoSS Management Module detects the modification and updates the IKE daemon's configuration data and the local security policy. It then sends a notification to the daemon. The daemon in response uses from then on the new configuration data and local policy for SA negotiations. Furthermore if there exist currently active SAs, it removes them and notifies the peer's daemon that these SAs are no longer valid, so that renegotiation of SAs can proceed.

4.3.1 A Proof of Concept Demonstration

Using our task model and the QoSS aware IPsec, we are able to provide different IPsec processing to applications according to system QoSS settings. We are also able to display responsiveness to a system status change by adjusting the values of security variables, like encryption algorithm used for ESP processing and authentication algorithm used for AH processing.

As a proof of concept we have developed a demonstration in which the kind of IPsec processing on the traffic originating from three specific applications varies in response to a *Security Level* QoSS parameter.

Assume that we have three example applications in our system: finger, telnet and ping.

- *ping* does not utilize any security services.
- *telnet* may use the confidentiality service provided by the ESP protocol of IPsec. One of the security attributes for which we have choices is the encryption algorithm, it could be any of the: *DES, 3DES*, *RC4*, *IDEA*, *CAST*, BLOWFISH, *3IDEA*, or *AES*.

  Our security policy could say that we can only utilize these algorithms: DES, 3DES, AES and a further refinement in the policy, that takes into account the notion of network modes, could say:

  -in *Normal* Mode: DES, 3DES, AES

  -in *Impacted* Mode: no encryption, DES, 3DES

  -in *Emergency* Mode: AES (in this case the range is degenerate).

  So the system is in one of the above modes and the user/application could request any of the available by the mode choices for the encryption algorithm. We

Table 1: IPsec Processing for different security levels

Security Level / Application	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

could go ahead and do the mapping to Security Levels for each Network Mode and we illustrate this for the *Impacted* Mode:

-*Low* Security Level in *Impacted* Mode: no encryption

-*Medium* Security Level in *Impacted* Mode: DES

-*High* Security Level in *Impacted* Mode: 3DES.

- *finger* may use the integrity service provided by the AH protocol of IPsec. One of the security attributes for which we have choices is the authentication algorithm, it could be any of the: *HMAC-MD5, HMAC-SHA, HMAC-RIPE-MD, DES-MAC, or KPDK.* So if our policy for the Impacted Mode says that available choices are: no authentication, HMAC-MD5, HMAC-SHA, the security levels could be mapped as:

  -*Low* Security Level in *Impacted* Mode: no authentication

  -*Medium* Security Level in *Impacted* Mode: HMAC-MD5

  -*High* Security Level in *Impacted* Mode: HMAC-SHA.

If our policy is as described above and our system is in Impacted Mode, we apply different IPsec processing to the applications in response to the Security Level parameter, as seen in Table 1: we apply no IPsec processing to the traffic of any of the applications when the Security Level is *Low*. If we switch to *High* security, *finger* traffic is authenticated with HMAC-SHA and we encrypt *telnet* traffic with 3DES.

More IPsec settings could change as a result of our selections, for example: the set of hosts we are willing to communicate with using IPsec, the SA lifetimes, the key lengths or rounds for variable key-size / variable round algorithms, the employment of transport or tunnel mode.

4.3.2 Discussion - Future Work

Currently we are using predefined sets of alternate IKE configuration data and local security policies that describe the characteristics we want our SAs to have for each <network mode, security level> pair and we make active the proper set of files through our QoSS module. This technique is not a mature solution and it has obvious scalability and maintenance problems: the addition of other QoSS parameters or the further refinement of possible network modes (such as adding a "crisis" mode) or security levels (e.g. including INFOCON levels) would increase noticeably the number of necessary sets of files. And if we decided to update our security policy and requirements for any QoSS parameter this would mean individual modification of those files.

We are working on identifying an architecture that would allow the automated daemon and the trust management system to take in QoSS parameters (like network mode, security level) and deploy the proper policy for those SA characteristics that are negotiable. We describe below proposed additional functionality we are currently researching to potentially include in our set-up.

Expand KeyNote to include Network Mode and Security Levels: First of all the KeyNote policy would include additional attributes corresponding to QoSS parameters. In order to enable the IPsec mechanism to handle further parameterization and to provide more granularity in security controls, we intend to expand KeyNote's parameter list to include QoSS discriminators, like *network mode* and *security level*.

As a result, the IKE daemon, when querying KeyNote to check if the SA under negotiation is in agreement with the local policy, would have to include in its query the current selections for the QoSS parameters. So a mechanism for the IKE daemon to retrieve the network mode and security level values would be included.

Furthermore the IKE daemon would generate its proposals according to the current QoSS settings. Two techniques could be utilized:

Dynamic IKE configuration data. Enhance the current implementation of the isakmpd.conf file to incorporate a range of peer communication parameters relevant to security level and network mode. This would solve the awkward file version replacement technique currently in use.

Peer communication parameter retrieval from KeyNote. This technique would eliminate the need to address the previous improvement. An interface would be developed to retrieve peer communication parameters from KeyNote that currently reside in isakmpd.conf. Since KeyNote maintains all policy information, it would be more efficient to simply retrieve policy related information from one area. This would solve concerns regarding dynamic policy changes and updates required throughout the various files and databases. This is the approach that the OpenBSD project is inclined to adopt in the future [14].

SPD Initial Policy Caching Routine. Currently, SPD must be manually populated with packet policy entries to allow an initiating peer to establish IPsec communications with another peer. This does not scale well in a dynamic environment. We intend to research methods that will enable the SPD to be dynamically populated by KeyNote to avoid direct access by users to the policy databases.

Develop a Policy Editor for KeyNote. Currently all modifications to the KeyNote Database require raw data input involving an intimate knowledge of the syntax language. To allow for dynamic updates and usability, we intend to develop a Policy Editor that will provide the user with an interface to modify the KeyNote policy files.

5. Summary

In this paper, we presented our concepts for variant security and QoSS and examined how these ideas can be applied to the modulation of existing security mechanisms. The ultimate goal of this work is to improve security service and system performance in QoS-aware distributed systems, by making security variability accessible to both users/applications and middleware systems.

We presented a QoSS framework for managing:

- security choices presented to the users in an abstract form
- limits on the choices presented to users in accordance with network operational status
- resource utilization costs due to variant security
- adjustment of underlying security mechanisms.

Finally, as a proof of concept, we illustrated how a complex security mechanism like IPsec, can be modulated to provide variant security in harmony with QoSS requests.

Our future work with IPsec is described in Section 4.3.2. We plan to extend our efforts for QoSS modulation to other security mechanisms. We also plan to conduct more experiments and measurements to help us understand better the impact of QoSS on the performance of applications under various network operational modes and high level policies.

6. References

[1] Blaze, M., Feigenbaum, J., Ioannidis, J. and Keromytis, A.D., "The KeyNote Trust Management System Version 2", Internet RFC 2704, Internet Engineering Task Force, September 1999.

[2] Blaze, M., Feigenbaum, J., Ioannidis, J. and Keromytis, A.D., "The Role of Trust Management in Distributed Systems Security", Secure Internet Programming: Issues in Distributed and Mobile Object Systems, Springer-Verlag Lecture Notes in Computer Science State-of-the-Art series, Berlin 1999, pp. 185 - 210.

[3] Blaze, M., Ioannidis, J. and Keromytis, A.D., "Trust Management for IPsec", Proc. of the Internet Society Symposium on Network and Distributed Systems Security 2001, San Diego, CA, February 2001, pp. 139-151.

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

[5] Doraswamy, N., Harkins D., "IPsec, The New Security Standard for the Internet, Intranets, and Virtual Private Networks", New Jersey: Prentice Hall, PTR, 1999.

[6] Hallqvist, N. and Keromytis, A. D., "Implementing Internet Key Exchange (IKE)", Proc. of the USENIX 2000 Annual Technical Conference, Freenix Track, San Diego, CA, June 2000, pp. 201 - 214.

[7] Harkins, D. and Carrel, D., "The Internet Key Exchange (IKE)", Internet RFC 2409, Internet Engineering Task Force, November 1998.

[8] Irvine, C. and Levin, T., "Toward a Taxonomy and Costing Method for Security Services", Proc. of the Computer Security Applications Conference, Phoenix, AZ, December 1999, pp. 183-188.

[9] Irvine, C. and Levin, T., "A Note on Mapping User-Oriented Security Policies to Complex Mechanisms and Services", Technical Report NPS-CS-99-08, Naval Postgraduate School, Monterey, CA, June 1999.

[10] Irvine, C. and Levin, T., "Quality of Security Service", Proc. of New Security Paradigms Workshop 2000, Cork, Ireland, September 2000, pp. 91-99.

[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] Kent. S. and Atkinson, R., "Security Architecture for the Internet Protocol", Internet RFC 2401, Internet Engineering Task Force, November 1998.

[13] Keromytis, A. D., Ioannidis, J., and Smith, J.M., "Implementing IPsec", Proc. of the IEEE Global Internet (GlobeCom) 1997, Phoenix, AZ, November 1997, pp. 1948-1952.

[14] Keromytis, A.D, personal communication, March 14, 2001.

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

[16] Maughan, D., Schertler, M., Schneider, M. and Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", Internet RFC 2408, Internet Engineering Task Force, November 1998.

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

[18] Spyropoulou, E., Levin, T., and Irvine, C., "Calculating Costs for Quality of Security Service", Proc. of the Computer Security Applications Conference, New Orleans, LA, December 2000, pp. 334-343.

[19] Stankovic, J.A., 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., November 1997.