Domain Based Internet Security Policy Management

John Zao, Luis Sanchez, Matthew Condell, Charles Lynn, Matthew Fredette, Pamela Helinek, Pajesh Krishnan, Alden Jackson, David Mankins, Marla Shepard, Stephen Kent
< *jzao | lsanchez | mcondell | clynn | mfredette | phelinek | pkrishnan | ajackson | dmankins | mshepard | skent @bbn.com* >
GTE / BBN Technologies Inc.

Abstract*. As security devices and protocols become widely used on the Internet, the task of managing and processing communication security policies grows steeply in its complexity. This paper presents a scaleable, robust, secure distributed system that can manage communication security policies associated with multiple network domains and resolving the policies — esp. those that specify the use of IP-AH/ESP security protocols — into security requirements for inter-domain communication.*

Technology innovation includes a formal model for IPsec policy specification and resolution, a platform independent policy specification language and a distributed policy server system. The formal model consists of a hierarchical domain model for IPsec policy enforcement and a lattice model of IPsec policy semantics. The policy specification language enables users to specify IPsec policies using the formal model regardless of the make of the security devices. The policy servers maintain the security policies in a distributed database, and negotiate the security associations for protecting inter-domain communication. Both the policy database and the policy exchange protocol are protected from passive and active attacks. Several UNIX implementations are available for non-commercial uses.

1. INTRODUCTION

Network security technologies are quickly being deployed over the Internet to support electronic commerce and virtual private networking services. In particular, security gateways, commonly known as firewalls, are installed along the perimeters of enterprise internets to enforce data origin authentication and access control, and security protocols, e.g., IP security (IPsec) protocols and Transport Level security (TLS/SSL) protocols, are used to provide end-to-end and hop-to-hop confidentiality, integrity and authentication protection to the Internet traffic. While these security measures have made the Internet a little safer to use, they also create significant management problems. For example, the deployment of security gateways fragments the Internet into security domains, each of which enforces different security policies. The use of security protocols necessitates establishing security associations among communicating end-points and en-route security gateways based on the security requirements of communication. Currently, the Internet Key Exchange protocol [1] can be used to negotiate the necessary security associations; however, a scaleable and general method is still unavailable for deducing the security requirements based on the security policies associated with the end-points and the gateways. In this paper, we present the principles and the design of a distributed policy management system capable of managing the security policies for multiple security domains and resolving them into the security requirements for inter-domain communication.

1.1. Requirement Analysis

We shall begin with a study of the functional requirements of Internet security policy management. Internet security policies are often characterized by the following properties:

- The policies are specified locally but can affect the security requirements of global communications since they are applicable to all communications to and from the network entities managed by the authorities.
- The policies may be specified by several authorities including network administration and users of communication applications. The policies may be classified into *mandatory* and *discretionary* policies respectively.
- The policies may be established for multiple protocol layers including data link, network, transport and application layers.
- The policies may be enforced by different security agents of heterogeneous origins: communication applications, host operating systems, security gateways, secure tunneling and link encryption devices.

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

Along with these characteristics, there lie the general expectations of safe policy management and assured policy enforcement.

Assured Policy Enforcement. The expectation usually implies the following requirements:

- The specification of security policies should be interpreted consistently by different security devices. This requirement implies the needs for a common policy specification language and a standard policy semantics accepted by both network administrations and application developers.
- The methods of managing the security policies should be applicable to networks of different sizes and topologies. In view of the scale and the growth rate of the global Internet, the management system must be distributed in design with decentralized management and enforcement components.
- The effects of security policy enforcement should remain unaffected by route changes (unless the policies are intended to interact with routing conditions). This requirement is a strong motivation for developing the concepts of domain-based policy management and boundary policy enforcement. By requiring all the security devices along a domain boundary to enforce the same set of security policies, we ensure that the traffic in and out of the domain will receive the same treatment regardless of the route it takes.

Safe Policy Management. This expectation can be translated into the following security requirements:

- *Secure Communication*—Security policy communication should be protected by integrity, data-origin authentication and confidentiality services.
- *Authorized Negotiation*—Policy management agents that participate in policy exchanges must be authorized to represent specific security domains and perform the required operations.
- *Selective Disclosure*—Security policies are sensitive information; even the authorized agents should be allowed to access only the policies necessary to accomplish their objectives.

1.2. Solution Approach

Our solution to the Internet security policy management problem is to develop a distributed system of *security policy servers*, each of which is authorized to maintain the security policies applicable to a cluster of network entities and to negotiate the security services with other servers needed to protect the communication involving those network entities. The security policy servers shall be deployed according to the hierarchical structure of security domains: each cluster of entities subjected to a common set of policies shall be grouped into a security domain; a collection of security domains managed by a common administration shall be associated with a security policy server. The security policies of each domain shall be enforced by the security devices installed along the boundary of the domain. Figure 1 shows the architecture of the domain based policy management system. The architecture consists of four components:

- *Security Policy Domains* — A formal model for security policy specification and enforcement that enables policies to be associated with non-overlapping domains in multiple hierarchies and enforced by the security devices located at domain boundaries; this model significantly reduces the complexity of policy management by transforming it from an inter-device process to an inter-domain process.
- *Security Policy Servers* — A distributed system of servers that maintain, exchange and process security policies associated with security domains in order to determine necessary security associations to protect inter-domain communication.
- *Security Enforcement Agents* — A set of security devices installed along the boundary of a security domain to enforce security policies of the domain.

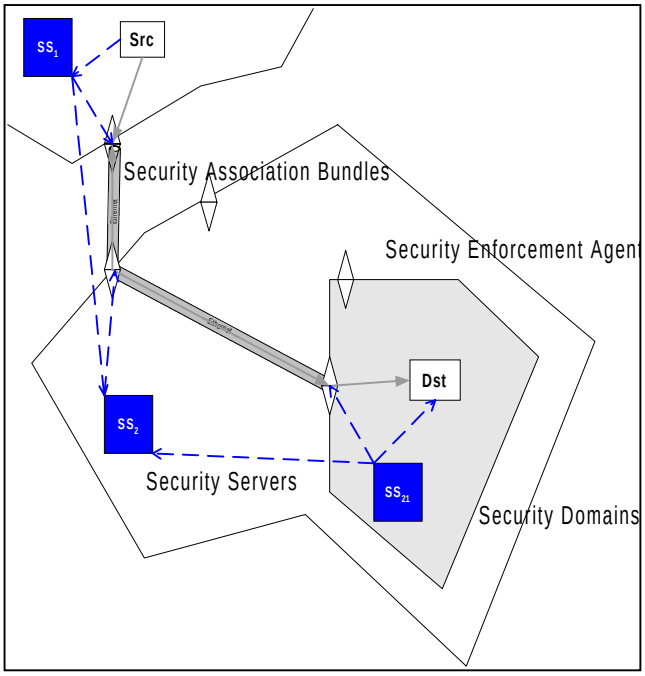

Figure 1. Architecture of Domain Based Security Policy Management System

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

- *Security Association Bundles* — The collections of security associations derived from the policy exchanges conducted by the security policy servers.

Four new technology components have been created to support this architecture:

- *Security Policy Exchange Protocol* — The protocol used for communication among the policy servers and between them and the enforcement agents
- *Security Policy Specification Language* — The common language for expressing security policies
- *IPsec Policy Algebra* — an algebra for deducing the IPsec actions from the policies
- *Security Policy Management Tools* — Graphic tools for viewing and editing security policies

1.3. Paper Overview

In the remaining sections of the paper, we will discuss the formal concept of hierarchical security domains, the design of the security policy server system and the security policy exchange protocol, the security policy specification language and processing algebra, and finally briefly describe the prototype.

2. SECURITY DOMAINS

2.1. Basic Definitions

An *internet security management domain* (or simply a *security domain*) is a connected cluster of communicating entities on the Internet, referred to as the *members* of the domain, that are protected by a common set of *communication security policies* applied by the *security enforcement agents* (or simply a *security policy*) positioned along the *boundary* of the security domain.

Security domains may form a *hierarchy* based on the following rules:

- A security domain may be a subset of another security domain if all the members of the first domain are also the members of the second domain; in that case, the first domain is considered to be at a higher level of the hierarchy than the second domain.
- A security domain may not overlap partially with another security domain, i.e., a security domain can either be a subset of another domain or share no common member with the other domain. Hence, a communicating entity can only be a *direct member* of a single domain and must be an *indirect member* of all the domains containing that domain.

Conceptually[1], every communicating entity on the Internet must be a member of one or more security domain(s) and subjected to the security policies of the domains to which it belongs directly and indirectly. Any inconsistency among the security policies of these domains that are applicable to a common set of entities is considered a *security policy conflict*.

A *communication security policy* (or simply a *security policy*) is a specification of security services that are required to protect specific network traffic in or out of a security domain under specific conditions. Given these domain-based security policies, one can determine the security services required to protect a communication among two or more network entities by combining the services specified by the security policies of the domains that contain these entities, and the policies of the domains traversed by the communication. We refer to this process as the *resolution* of security policies.

The security policies pertaining to a security domain are enforced solely by the security agents on the boundary of the security domain. We refer to this practice as *boundary policy enforcement.* The practice must be *complete* in the following sense: all communications between the members of the domain and the external Internet must be processed by the security agents according to the security policies of the domain; no communication path can exist between the members of a security domain and the rest of the Internet that can bypass the protection of the security agents.

2.2. Usage Rules

Together, the concepts of security domain hierarchies and boundary security enforcement simplify the management of communication security policies in the following ways:

- It provides a simple but powerful abstraction for reasoning about the deployment and the enforcement of security policies. A security policy will be applied to a communication if and only if the traffic traverses the boundary of the security domain that owns the policy. Hence, in order to protect the communication of some network entities, one must enclose them in a security domain and place security agents at all traffic crossing points on the boundary. Conversely, in order to determine the policies applicable to a communication, one simply has to discover the security domains containing the communication

[1] We shall reconcile the conceptual model with practical network configurations in Section 2.2.

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

sources and destinations as well as the domains traversed by the traffic.

- It separates the concerns of policy enforcement from those of packet routing: given the practice of boundary policy enforcement is complete then the same set of security policies will be applied to the communication traffic regardless of the routing choice for the traffic.

However, because boundary policy enforcement precludes the application of security policies to the intra-domain communication, the practice shall affect the way we group network entities into security domains. Following are two possible groupings of entities.

2.2.1 Grouping of Local Subnets

The first case corresponds to the typical setting of a firewall protected internet segment: one or more *internet subnets* are surrounded by security gateways that guard all the connections between the subnets and the external Internet. In this setting, the security gateways are the security enforcement agents; the hosts, bridges and routers installed on the subnets are the members of the domain. The security policies specify not the protection to be applied at the communication end-points, but those to be enforced at the security gateways which function as inter-mediary agents.

In some special settings, the subnets may be surrounded by security gateways that perform different functions and enforce different policies. These settings can be modeled as a collection of overlapping security domains with the same members but different security agents/gateways. Each group of security gateways that enforce the same policies should define the boundary of a distinct domain. As explained in §2.2.3, the overlapping domains shall only be used for formal reasoning; in actual practice, they shall be collapsed into a single physical security domain guarded by multiple groups of security agents.

2.2.2 Grouping of Hosts and Applications

Since security policies can only be enforced at the boundary of a security domain, policies applicable to a communication end-point—be it a layer in the protocol stack, a software application or a hardware module—must be associated with a security domain that encloses the software and hardware components protected by the policies. In this setting, the security enforcement agents are the security modules that perform the services prescribed by the security policies. Network, transport and application layer security can all be modeled in this case. Multiple security domains may also be formed within a single host if different policies may be applied to various user applications and/or at different protocol layers.

2.2.3 Simplification of Domain Hierarchy

The strict definition of security domains [§2.1] seems to imply that a large number of security domains must be established over the entire Internet before domain based policy management can be useful. This is not the case: the system not only can be deployed incrementally but can also be used with a small number of security domains (similar to those established by the network administration). The following techniques can be used to transform the *logical domain model* described above into a *physical domain architecture*.

Collection of end-point domains. A large number of logical security domains may be established when one introduces a policy to a group of communication end-points. Since the policy is expected to protect individual end-points, a domain must be created for each end-point. One way to combine these single-entity domains into one domain is to use a *pointer* for identifying the security enforcement agents. Instead of providing the explicit identity of the security agent, the policy shall use a pointer to refer to the end-point that matches with the policy specification. This will allow all the end-points protected by the same policies to be included in one security domain and use the same pointer as a reference to individual security agents.

Collapse of overlapping domains. Security domains with same members but different security agents [§2.2.1] can be collapsed into one domain by grouping the security agents that enforce the same policies. Again, a pointer must be used to refer to the security agents: the policies that are enforced by the same group of security agents shall use a pointer to refer to any agent inside the group (but not to the ones outside); policies that are enforced by different group of agents shall use different pointers with disjoint coverage. Each of the pointers thus represent a distinct overlapping domain.

3. SECURITY POLICY SYSTEM

3.1. Distributed Server Architecture

The Security Policy Server System (SPS) is a distributed system which provides hosts and security gateways with the policy information required to establish a secure communication end-to-end through possibly multiple security gateways. The Security Policy System provides one or more automated mechanisms for hosts to discover primary and secondary security gateways relevant in an end-to-end

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

communication. Using the Security Policy System, hosts can validate the identity of security gateways and verify that the gateways in question are authorized to represent the source or destination host that they claim to represent.

SPS is comprised of Policy Servers (PS), Policy Clients (PC), Master Files and SPS Databases. Master Files contain local policies and other particular information about a security domain. Local policy information combined with non-local policies (policies outside the boundaries of the security domain) form the SPS Databases. Policy Servers receive request messages from policy clients and other policy servers, process them, and provide the appropriate policy information to the requestor based on the request and access control rules. The servers also maintain the SPS databases by loading local and non-local policy information received through SPS exchanges. Policy Clients generate requests for policy information and transform the replies into the appropriate format required by the application using SPS.

Policy Servers and Clients use the Security Policy Protocol (SPP) to exchange policy information. SPP transports policy information from the SPS Database where this information resides to security gateways and hosts. This protocol provides hosts with the policy information needed to establish security associations with security gateways in the communication path to other hosts, without requiring a-priori knowledge of the identities of the security gateways.

3.2. Policy Databases

In SPS, every security domain must maintain a database containing the policy information for that domain. Security domains could be as small as one host or as large as several networks. This database, called the SPS Database, is comprised of three logical databases: (1) the Local Policy Database, (2) the Cache Database, and (3) the Security Domain Database. The Local Policy Database contains all the policies for the security domain. It is populated with information coming from the Master File of the security domain. The Cache Database contains local and external policies received from other security domains via SPS exchanges.

The Security Domain Database contains a list of all hosts, security gateways, and policy servers that are part of the security domain. Compliant SPS implementations of a policy client and server do not need to implement these three databases separately. However, the information contained in each one of them must exist. The Local Database and the Cache Database must keep a distinct sets of policies since it is not possible to revert cached policy information into Local Database policy information after the cached items expire. While it is not necessary to standardize the format of the database used, the SPS database must contain a minimum set of information.

3.3. Security Policy Protocol (SPP)

In Policy Based Security Management (PBSM), policy clients and servers exchange information using the Security Policy Protocol (SPP) illustrated in Figure 2. SPP defines how the policy information is exchanged, processed, and protected by clients and servers. The protocol also defines what policy information is exchanged and the format used to encode this information.

SPP is flexible and can adapt as policies and topologies change since it does not require a-priori knowledge of the identities of the security gateways along the end-to-end path of a communication. The protocol depends solely upon routing of an appropriate message to discover the identities of the security gateways. In SPP a client may send a query message to a pre-configured local policy server to determine the security policies required for a particular communication. The local server, if it does not have an answer to the query, sends a query to the final destination of the communication.

SPP proxies running on security gateways along the route to the destination will intercept the query and forward it to the policy server for that gateway. That server may then either answer the query only if it has a policy for the communication and if it is authoritative over the final destination. A policy server is authoritative over a host if it asserts policy information for it and can present cryptographic credentials indicating the server's right to make policy statements on behalf of the host. If the policy server is not authoritative over the final destination, it forwards the query to the final destination. This process continues hop-by-hop (server to server) until the query reaches the authoritative policy server for the final destination of the communication.

The policy server ultimately responsible (authoritative) over the final destination sends a reply message destined to the policy server that sent the query. This is repeated on a hop-by-hop basis until the policy server that originated the query receives its reply. When a policy server receives a reply, it must verify a cryptographic signature on the message and the chain-of-trust. The signature proves the integrity and authenticity of the message. The chain-of-trust provides cryptographic proof that every server that has processed this reply is authorized to be involved

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

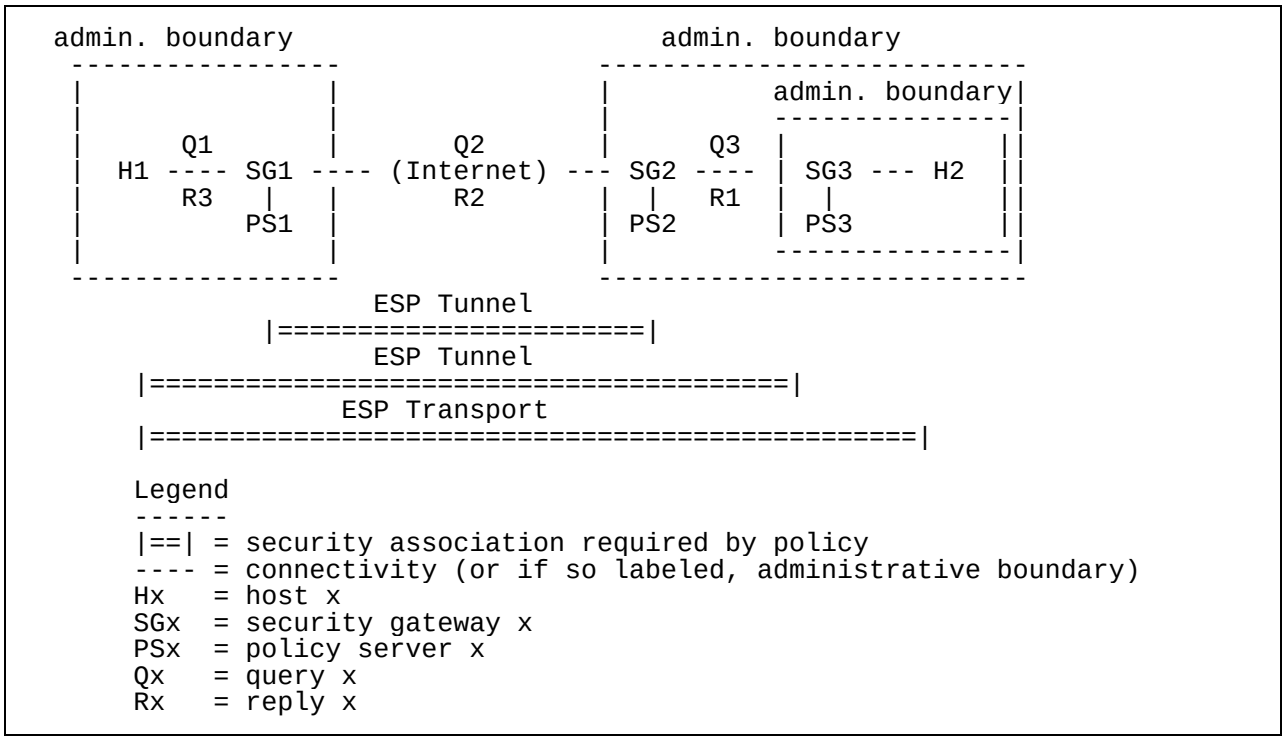

Figure 2. Illustration of SPP Operation

in the communication. The chain-of-trust proves that all the servers involved are either directly or indirectly authoritative over the source or the destination of the communication.

Policy servers communicating through SPP must also resolve their local policies with the policies received in reply messages before accepting and incorporating them into their local policy databases. The resolution process intersects the two sets of policies. SPP implementations perform this process to: (1) find the common policy elements, (2) resolve any ambiguities in how the policies may be enforced; and (3) determine if policies must be sent to a gateway controlled by the server to provide it with the information necessary to allow the communication. The server then replies with the merged policy and sends any necessary signals to its gateways and clients.

All policies exchanged using SPP must be decorrelated. Two policies are decorrelated if there exists at least one selector in both policies for which their values do not overlap or intersect. Due to the local nature of policies, decorrelation is necessary to permit policy servers to properly cache policies.

3.3.1 SPP Message Format

An SPP message consists of a message header and a payload. The SPP header is present in every message. It contains fields identifying the type of message, the status of the message, the number of queries and/or record payloads, and the identity of the host requesting policy information. The header also includes a time-stamp field that provides anti-replay protection. Following the header there might be zero or more SPP payloads. Currently, there are three payload types defined in SPP: Query, Record, and Signature payloads.

SPP has six distinct message types. Query messages contain a specific request for policy information. Reply messages include policy records that answer specific policy queries. Policy messages include policy information and are utilized for up-/ downloading security policies to and from a policy server. Policy Acknowledgment messages are utilized to acknowledge corresponding policy messages but do not contain policy information. Policy servers exchange bulk policy information between servers using transfer messages. Finally, policy servers use keep alive messages to inform security gateways and/or other monitoring devices of the status of the server.

SPP messages are authenticated using either IPsec [3] or another security mechanism. SPP provides a basic security mechanism that can be used to provide authentication and integrity to its messages when other security mechanisms are not in use. The SPP authentication is especially useful when traversing heterogeneous domains and the identity of the policy

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

server authoritative for the destination is unknown. These services are provided using digital signatures.

SPP carries signatures in the signature payload. The signature is calculated over the entire SPP message. When this service is used, the entity (host, policy server, or security gateway) verifying the signature must have access to the public key that corresponds to the private key used to sign the SPP message. Certificate fetching is out of the scope of SPP. However, SPP provides a simple certificate fetching mechanism for entities that elect to use it as an alternative to other mechanisms. SPP supports several Public Key certificates formats.

3.3.2 SPP Operation

The following example illustrates how the Security Policy Protocol operates and interacts with IPsec. It describes, step-by-step, the process from the initial application call to the establishment of the necessary security associations. We assume the policy servers have been loaded with policies for their security domains and the policies have been appropriately decorrelated.

1. User application attempts to send a message from H1 to H2 (e.g., finger somebody@H2)
2. IPsec on H1 gets the packet and finds a policy for it in the Security Policy Database (SPD).
3. H1 does not have a security association for the communication and asks the Key Management Protocol to establish one.
4. The policy client in H1 is queried for the policy governing the communication between H1 and H2.
5. H1's policy client creates an SPP query, Q1, and sends it to PS1, its configured policy server.
6. PS1 receives the query. Its security domain database indicates that it is not authoritative over H2 so it checks its cache to see if it has a cached answer. For this example, it does not, so it creates a new SPP query, Q2, with the query and sends it to H2.
7. SG2 intercepts Q2 and forwards it to PS2.
8. PS2 receives the query. Its security domain database indicates that it is not authoritative over H2 and PS2 determines it wants to be involved in the communication. It checks its cache to see if it has a cached answer. For this example, it does not, so it creates a new SPP query, Q3, with the query and sends it to H2.
9. SG3 intercepts Q3 and forwards it to PS3.
10. PS3 receives the query. It checks its security domain database and determines that it is authoritative over H2 so it will send a reply. It checks its cache to see if it has a cached answer. For this example, it does have one cached from previous information sent to it by H2. The cached policy indicates ESP transport must be done with H2 and ESP tunnel must be done with SG3.
11. PS3 creates two messages. One is an SPP reply message, R1, with the policy indicating the required security associations, a security server record indicating PS3 is authoritative over H2, and PS3's certificate in a certificate record. The reply is sent to PS2. The second message is an SPP policy message to signal SG3 that it will need a security association with H1.
12. PS2 receives the R1. It verifies that PS3 is authoritative over H2. It looks up its local policy and resolves it with the policy in the reply. It caches the resolved policy and creates two messages. One is the reply to PS1, R2, that contains the resolved policy, PS3's security server and certificate records and a security server record that states that PS2 is authoritative over PS3 and PS2's certificate. The second message is an SPP policy message to signal SG2 that it will need a security association with H1.
13. PS1 receives the R2. It verifies that PS2 is authoritative over PS3 and PS3 is authoritative over H2, forming a valid chain of trust. It looks up its local policy and resolves it with the policy in the reply. It caches the resolved policy and creates R3, a reply to H1. R3 contains the resolved policy (the security server and certificate records are no longer needed). It also creates a policy message to signal SG1 that it will need a security association with SG2.
14. The policy client receives the R3 and returns it to the application that queried for it. The policy indicates that the three security associations must be established and they must be established in a particular order.
15. The Key Management Protocol is given this information and first creates a security association with SG3 which it can use to set up the security association with H2.
16. Finally, the original message from the application can proceed using the security associations.

4. SECURITY POLICY SPECIFICATION

4.1. Security Policy Specification Language

In order to support vendor and platform independent communication security policy specification, we developed the Security Policy Specification Language (SPSL) [7] by borrowing the syntax from IETF Routing Policy Specification Language (RPSL) [11]. The language was designed

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

originally for specifying IPsec and ISAKMP policies, but its extensible syntax allows new object classes to be added for specifying policies of other security protocols. The language has the following distinct features.

4.1.1 IPsec/ISAKMP Policy Specification

In SPSL, a communication security policy is defined as a binding between a list of communication conditions and a corresponding list of security actions. If a communication matches with one of the conditions then the list of actions must be taken to protect the communication. In an IPsec/ISAKMP policy, a communication condition is specified as a tuple of selector values. SPSL supports an extended selector value set, which includes all the selectors mentioned in IPsec architecture document [5]. The actions, on the other hand, can effect three different kinds of operations:

- They may specify simple packet filtering actions: discard, pass or forward the packet (via tunneling) to a designated network entity.
- They may use IPsec tunnels or transports to pass the packet. The possible security mechanisms for protecting the tunnels and the transports are specified as ISAKMP proposals allowed by ISAKMP-IPsec DOI [10].
- They may also specify security proposals necessary for protecting ISAKMP exchanges.

4.1.2 Node/Domain Based Policy Associations

SPSL supports two different ways to associate security policies with network entities. They are known as node based and domain based policy association.

In *node based policy association*, security policies are bounded to individual network nodes and security devices. The policies associated with a network node specify the protection for the communications to and from the node. These policies are expected to be enforced by the security modules in the node. The policies associated with a security device (formally known as a security enforcement agent) specify the protection for the communications passing through these agents. Either the source or the destination of the communication must be among the nodes that the agent is authorized/expected to protect. Both the network nodes and the security enforcement agents should manage their own policies.

In *domain based policy association*, security policies are bounded to a hierarchical security domain. A security domain is defined as a connected set of network entities that are protected by security enforcement agents (SEA) placed on every communication path going through the domain perimeter. Every security enforcement agent of the domain works to enforce the common set of security policies associated with the domain. The security domains may be completely disjoint or contained in one another, and thus form multiple domain hierarchies. The policies associated with a domain are managed by one or more special agents known as security servers (SS) of the domain. These servers may be distinct network entities or co-located with the nodes or the security enforcement agents of the domain.

SPSL defines special object classes to support the two policy associations: classes node, node-set, gateway and gateway-set are defined to be used in node based policy association while classes gateway, gateway-set and domain are to be used in domain based policy association. A mixed use of two associations is also permitted with the node class objects to be regarded as one-node domain objects.

Multiple Distributed Policy Enforcement Points

SPSL allows explicit selection of security enforcement agent(s) of a security policy. The choices can be specific interfaces of end nodes, en-route security gateways (SG) and/or network nodes identified by IP addresses or DNS names. If the selection is generic such as local/remote security gateways then the exact enforcement agent will be reached by routing. If the selection is specific as by giving IP addresses or DNS names then the enforcement agent may be reached via IP tunneling.

Authentication and Authorization Mechanisms

SPSL defines special object classes for supporting the following security services:

- *Data integrity of policy specification*—Every policy object is protected at least by keyed MD5 data integrity and data origin authentication.
- *Data origin non-repudiation of policy specification*—Policy objects may be protected with public key signatures, which offer non-repudiation proof to the issuer(s) of the policies.
- *Authentication and authorization of policy management entities*—Management objects such as maintainers, person and roles all have public key certificates so that they may issue policies and/or identify themselves to the security management system for access control purposes.

With these services, users of SPSL policy specifications can always verify the integrity and the origin of the policies and allow only authorized personnel to maintain and/or access the policies.

4.1.3 Object Classes

SPSL uses the object programming paradigm. It defines a small set of object classes that maintain

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

policy information in their attributes. There are no executable methods in the object classes. New classes can be created as needed based on a syntactic rule similar to inheritance in object-oriented languages: a new class may have all the attributes of an existing class in addition to its own attributes. Objects in a SPSL file are distinguished and referred to by the unique values of their first attribute, which is known as the key attribute.

The existing classes in SPSL can be divided into the following four categories:

Primitive Data—They contain basic or atomic data elements used in policy specification, e.g., object-name, ipv4-address, integer-range, date, etc.

Management Agents—They contain relevant information of the management entities; the existing classes in this category are maintainer (mntner), person, role, and certificate (cert).

Network Entities—They depict the network elements that are relevant to policy specification; the existing classes are node, node-set, gateway, gateway-set, policy-server and domain.

Policies—They are the classes that contain the policy specification; there are only two classes at the moment: policy class specifies general packet filtering rules and IPsec-policy class specifies IPsec selectors and actions; objects of policy class may appear in two forms for short or long policy specification.

Each class has a set of attributes which store information about the objects of the class. Attributes can be mandatory (man) or optional (opt). A mandatory attribute must be defined for all objects of the class, and an optional attribute may be skipped. Attributes can also be single valued (s-v) or multiple valued (m-v). A single valued attribute may only be used once per object. A multiple valued attribute may be used more than once per object. Each object is uniquely identified by the key attribute of its class.

An SPSL object is textually represented as a list of attribute-value pairs. Each attribute-value pair is written on a separate line. The object's representation ends when a blank line (i.e., a line containing only whitespace characters) is encountered. The default character set is ISO 8859-1 (Latin-1). An object's specification may contain comments. A comment may appear anywhere in an object definition. It starts at the first "#" character on a line and ends at the first end-of-line character. Figure 3 shows a few sample SPSL objects.

4.2. Algebraic Semantics of IPsec Policies

In addition to defining a language for specifying security policies, we also developed an algebra for computing the *composition* and the *difference* of IPsec policies in particular. The algebra can be used in two different ways to process IPsec policies. First, it can be used to perform *policy decorrelation*, a process that can transform a set of policies into a corresponding set of uncorrelated policies. The decorrelated policies have a desirable property that only one of them may match with a specific communication condition; thus, these policies can be processed and used independent from one another. The policy resolution algebra can also be used to build an IPsec policy resolver—a functional module that accepts queries for the security requirements of end-to-end communications and returns the applicable policies to be enforced by specific agents.

4.2.1 Lattice Model of IPsec Policies

The IPsec policy algebra is based upon the set theoretic relations existing among the policy conditions and the partial order relations among the policy actions. As mentioned, every IPsec policy specifies a clause of *communication conditions*, expressed in terms of the values of seven *selectors* (source and destination IP addresses, source and destination port numbers, transport layer protocol identifier, user identifier, and security label) and a clause of *security actions*. Each selector may take a specific value or the "wildcard" value ANY. Source and destination IP addresses may also take ranges of address values. On the other hand, the action clause specifies whether an IP datagram should be *discarded*, *bypassed* without IPsec processing, or *passed* with IPsec processing when it matches with the policy condition. If IPsec processing is necessary then the action clause also specifies a typed list of security mechanisms (such as follows) to be used in datagram processing:

```
(esp (DES HMAC_MD5) or (DES HMAC_SHA))
or ((esp DES), (ah (HMAC_MD5) or
(HMAC_SHA)))
```

An IPsec policy can be modeled as a mapping from its *condition clause*, which is a direct product of seven selector value sets $S_{i=1..7}$, to its *action clause*, which is a typed list of IPsec-ISAKMP descriptors of security mechanisms:

$$P = \overset{7}{\underset{i=1}{\times}} S_i \rightarrow A.$$

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

```
policy-name:    P11
descr:          incoming TELNET connection to SD1 hosts
domain:         SD1
member-of:      P_SD1
direction:      inbound
dst:            199.100.2.1      # H1 Address
src:            ANY
xport-proto:    TCP
action:         permit
IPsec-proto:    esp transport
IPsec-mech:     esp cipher DES keylen 56 integrity HMACMD5 keylen 128
tech-c:         mcondell
admin-c:        lsanchez
mnt-by:         SS1_MNT
changed:        mcondell@ir.bbn.com 19980529
signature:      <mnter-name><cert-name><signature-alg><signature-value>

sec-serv:       SS1
descr:          primary security server of SD1
name:           sec-server-1.bbn.com
alias:          cotton.bbn.com
ifaddr:         128.89.0.2 masklen 16

sec-agent:      SG1
descr:          security gateway 1 protecting SD1
name:           bbn-firewall.bbn.com
ifaddr:         128.89.0.1 masklen 16
ifaddr:         192.2.1.83 masklen 24

sec-domain:     BBN.COM:SD1
descr:          BBN IR security domain
coverage:       128.89.5/24
sec-serv:       SS1, SS1a
sec-agent:      SG1, SG1a, SG1b, SG1c
sec-policy:     P_SD1
```

Figure 3. Policy Objects in SPSL

Based on the current data model of IPsec policies [6], we deduce that A is a *lattice* with the *top* element T being the NULL action or the least upper bound of conflicting actions and the *bottom* element $\perp$ being the UNDEFINED action lower in order than all actions defined. The lattice model of the policy actions enables us to compose two actions by finding their least upper bound using the *join* operation $\uparrow$ of a partially ordered set.

4.2.2 Composition Operation

The composition of two IPsec policies $(\circ : \wp \times \wp \to \wp)$ is a *commutative* and *associative* operation that produces a new policy specifying the resultant actions to be taken when the datagram states match with the conditions of *both* input policies. The operation is performed by processing the condition clauses and the action clauses separately. The composition of two policy conditions is performed simply as the *intersection* of the direct products of their selector value sets:

$$\left(\underset{k}{\times} S_{ik}\right) \circ \left(\underset{k}{\times} S_{jk}\right) = \left(\underset{k}{\times} S_{ik}\right) \cap \left(\underset{k}{\times} S_{jk}\right) = \underset{k}{\times} \left(S_{ik} \cap S_{jk}\right)$$

The composition of two action clauses A_i, A_j is performed as the join operation $\uparrow$ defined for the partially ordered set of IPsec policy actions:

$$A_i \circ A_j = A_i \uparrow A_j$$

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

4.2.3 Difference Operation

The *difference* operation $\left(-:\wp\times\wp\rightarrow\wp\right)$ is a *non-commutative* operation that takes *two* IPsec policies and produces a new policy which specifies the actions to be taken when the condition of the *second* policy is excluded from that of the *first* policy. Policy difference can also be performed by processing the condition clauses and the action clauses separately.

Let $P_1=\left(C_1\rightarrow A_1\right)$ and $P_2=\left(C_2\rightarrow A_2\right)$ be two policies, and $P_{21}\equiv P_2-P_1=\left(C_{21}\rightarrow A_{21}\right)$ be one of their differences then the formulas to compute the condition C_{21} and the action A_{21} of the difference policy are

$C_{21}=C_2-C_1$ and $A_{21}=A_2$.

5. PROTOTYPE IMPLEMENTATION

PBSM is comprised of five implementation modules: the parser for the Security Policy Specification Language (SPSL), the Security Policy System (SPS), the Security Management Agent (SMA), the Certificate Infrastructure (CI) and the Policy Management Tool (PMT). The following sections briefly cover key implementation details for each module.

5.1. SPSL Parser

Security Policy Specification Language (SPSL) is a language designed to express security policies, security domains, and the entities that manage the policies and domains. SPSL currently supports policies for packet filtering, IP security (IPsec), and ISAKMP exchanges, however, it may easily be extended to express other types of policies.

SPSL uses the object paradigm although it is neither an object-oriented nor a type based language. The language defines a small set of classes, which can instantiate objects maintaining data relevant to policy specification. The data are contained in the attributes defined in the object classes. There are no executable methods in the classes. The parser is implemented using Bison and Flex.

5.2. Security Policy System (SPS)

The Security Policy System (SPS) is a main component of PBSM. It takes requests from clients (hosts and security gateways) inside/outside a security domain and supplies them with the security policies needed to establish an end-to-end communication across multiple heterogeneous security domains. SPS is comprised of the Security Server Daemon (SSD) and the Security Policy Resolver (SPR).

The Security Server Daemon is the server portion of the Security Policy System and it resides in the security servers. The Security Policy Resolver is the client portion of the Security Policy System and it consists of a set of library calls used to request policy information. Client SPRs at hosts and/or security gateways communicate with Security Server Daemons using the SPP protocol.

The implementation of the SPS consists of developing the library calls that form the Security Policy Resolver, the functions that make the Security Policy Agent, the development of the decorrelation algorithm and the composition algebra processing. The last three components form the Security Server Daemon. The Security Policy Protocol (SPP) is implemented as part of both the Security Policy Resolver and the Security Server Daemon.

5.2.1 Security Server Daemon

The Security Server Daemon (SSD) is the server portion of the Security Policy System and it resides in any host functioning as a security server. The server daemon loads policies for a security domain from its master file into the SPS database. It replies to queries from policy clients and communicates with peer server daemons from other security domains to request policy information related to hosts outside its security domain.

The SSD processing is simple in nature. First, the SSD loads configuration parameters from a configuration file. It then calls the SPSL parser to verify that the policies in the Master files are correct. Upon successful parsing completion the SSD calls the policy decorrelator to eliminate any overlaps among the security policies in the Master File. Once the policies are decorrelated the SSD loads the policies into the SPS database.

After loading the policies, the SSD remains in a loop waiting for messages from client Security Policy Resolvers or peer Security Server Daemons. SSDs establish the validity of incoming messages, make SPS database searches, interface with the Policy Management Agent and generate appropriate replies as needed. The SSD has the ability to identify conflicts and resolve ambiguities between the local and imported policies. It may also take advantage of any ambiguities to help optimize the use of security associations within its security domain.

5.2.2 Security Policy Resolver

The Security Policy Resolver (SPR) is a set of library routines on PBSM compliant hosts and gateways that acts as an interface between the host and its security servers. These routines provide both synchronous and asynchronous interfaces that may be

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

used by applications to request services from the Security Server Daemons. The SPR communicates with the security servers using the Security Policy Protocol.

The Security Policy Resolver is configured with the IP addresses of the Security Servers that it should use to resolve policies. The host on which the SPR resides may be included in this list. At least one Security Server must be configured. Multiple servers may be configured, as in DNS, to allow secondary servers to be accessed in the event that a primary server fails. These configured servers are the default servers that the synchronous interface will use to make policy queries. This information may also be used by applications to determine the appropriate Security Server Daemon to attempt with the asynchronous interface.

5.3. Security Management Agent

The Security Management Agent is an application that intercepts calls from the IPsec engine to a Key Management Protocol (KMP) and vice versa. It serves as the initiator of the policy resolution process in PBSM. When an application requires IPsec the IP datagram is sent to the IPsec engine in the kernel (for native software implementations). The IPsec engines check for an existing Security Association (SA) for that particular communication. If one SA exists in the Security Association Database it uses it otherwise the engine signals the KMP to establish one.

The SMA intercepts the engine signal, asks the SPR for policy information regarding the communication in question and waits for a response from the system. Once the response arrives, the SMA passes such response to the KMP and the process continues as usual. If the SMA doesn't receive a response within a time window it forwards the original request to the KMP and the process continues as usual. As a possible optimization, the SMA can intercept the engine's signal and forward it to the KMP while using it to initiate the policy resolution process with the SPR.

5.4. Certificate Infrastructure

The implementation of the Certificate Infrastructure (CI) consists of the development of the Certificate Infrastructure daemon and the CI database tools. The CI daemon (certd) maintains the database of certificates and CRLs, performs validation and external fetches, and provides certificates or certificate data such as keys, or Certificate Extensions to requesting applications. The CI daemon is a separate process that should be run in the background as root.

The application may request a certificate, key or a set of certificate extensions from the CI. It gives the CI a set of parameters which allows the CI database to search for the requested certificate. If the CI finds one or more certificates that match the input parameters, it returns them all to the application. The more optional input parameters available to the application to configure, the more likely the CI is to be able to identify one correct certificate.

A series of tools were developed for testing and supporting the Certificate Infrastructure daemon or certd. These include tools for inserting certificates and Certificate Revocation Lists (CRLs) into the certificate database, reading certificates from filenames, printing the database contents and validating certificates.

5.5. Policy Management Tools

The Policy Management Tools provide a graphical user interface to create SPSL files. Additionally, they provide an interface to several tools that may be useful for helping system administrators correctly develop their local policies, including tools to interact with the local security server and to provide consistency checks on policies.

6. Future Work and Conclusions

The work presented in this paper represents our first attempt in developing a framework for managing and processing communication security policies. The work may be extended in the following directions.

Policy managing for both network and application layer security. Existing work was focused entirely on the policies governing IPsec AH-ESP protocol uses. It would be meaningful to investigate the use of this system to negotiate the security requirements for both TLS/SSL protection between end hosts and IPsec tunneling among intermediate security gateways.

Algebraic semantics for Virtual Private Networking policies. The algebra discussed in §4.2 specifies the algebraic rules to compose and resolve IPsec policies written according to the simple Pereira-Bhattacharya model. It would be useful to extend the rule set in order to incorporate policy actions relevant to virtual private network (VPN) operation.

Policy management for group communication. The SPP protocol discussed in §3.3 performs only policy negotiation for two-party communication. It would be valuable to extend the protocol for supporting group communication.

Finally, it may be necessary to divide the policy management task into two sub-tasks of *policy discovery and negotiation* and *policy specification and processing.* The first task depends the use of multi-

0-7695-0490-6/99 $10.00 (c) 1999 IEEE

party protocols and distributed databases while the second requires the knowledge of language semantics and program analysis. There are plenty of work remained to be done in both areas.

7. REFERENCES

[1] D. Harkins, D. Carrel. "The Internet Key Exchange (IKE)" *RFC-2409.* Internet Society, Network Working Group, Nov. 1998.

[2] D. Maughan, M. Schertler, M. Schneider, J. Turner. "Internet Security Association & Key Management Protocol (ISAKMP)" *<draft-ietf-IPsec-isakmp-07>*, Internet Society IPsec Working Group, Feb. 97

[3] S. Kent, R. Atkinson. "IP Authentication Header" *RFC-2402.* Internet Society, Network Working Group, Nov. 1998.

[4] S. Kent, R. Atkinson. "IP Encapsulating Security Payload (ESP)" *RFC-2406.* Internet Society, Network Working Group, Nov. 1998.

[5] S. Kent, R. Atkinson. "Security Architecture for the Internet Protocol" *RFC-2401*. Internet Society, Network Working Group, Nov. 1998.

[6] R. Pereira, P. Bhattacharya. "IPsec Policy Data Model" *Internet Draft <draft-ietf-IPsec-policy-model-00>*, Internet Society, IP Security Working Group, Feb. 1998.

[7] M. Condell, C. Lynn, J. Zao. "Security Policy Specification Language" *Internet Draft <draft-ietf-IPsec-spsl-01>*. Internet Society, IP Security Working Group, July 1999.

[8] L. Sanchez, M. Condell. "Security Policy System" *Internet Draft <draft-ietf-IPsec-sps-00>*. Internet Society, IP Security Working Group, Nov. 1998.

[9] J. Zao. "Partially Ordered Composition and Search of IP Security Policies" *To be published in IEEE JSAC special issue on network security*, Feb. 2000.

[10] D. Piper. "The Internet IP Security Domain of Interpretation for ISAKMP" *RFC-2407*. Internet Society, Network Working Group, Nov. 1998.

[11] C. Alaettinouglu, T. Bates, E. Gerich, D. Karrenberg, D. Meyer, M. Terpstra, and C. Villamizer. "Routing Policy Specification Language (RPSL)". *RFC 2280.* Jan. 1998.

0-7695-0490-6/99 $10.00 (c) 1999 IEEE