Network Working Group
Request for Comments: 1636
Category: Informational

R. Braden
ISI
D. Clark
MIT Laboratory for Computer Science
S. Crocker
Trusted Information Systems, Inc.
C. Huitema
INRIA, IAB Chair

June 1994

Report of IAB Workshop on

Security in the Internet Architecture

February 8-10, 1994

Status of this Memo

This memo provides information for the Internet community. This memo does not specify an Internet standard of any kind. Distribution of this memo is unlimited.

Abstract

This document is a report on an Internet architecture workshop, initiated by the IAB and held at USC Information Sciences Institute on February 8-10, 1994. This workshop generally focused on security issues in the Internet architecture.

This document should be regarded as a set of working notes containing ideas about security that were developed by Internet experts in a broad spectrum of areas, including routing, mobility, realtime service, and provider requirements, as well as security. It contains some significant diversity of opinions on some important issues. This memo is offered as one input in the process of developing viable security mechanisms and procedures for the Internet.

Table of Contents

1. INTRODUCTION

The Internet Architecture Board (IAB) holds occasional workshops designed to consider long-term issues and strategies for the Internet, and to suggest future directions for the Internet architecture. This long-term planning function of the IAB is complementary to the ongoing engineering efforts performed by working groups of the Internet Engineering Task Force (IETF), under the leadership of the Internet Engineering Steering Group (IESG) and area directorates.

An IAB-initiated workshop on the role of security in the Internet Architecture was held on February 8-10, 1994 at the Information Sciences Institute of the University of Southern California, in

Marina del Rey, California. This RFC reports the results of the workshop.

In addition to the IAB members, attendees at this meeting included the IESG Area Directors for the relevant areas (Internet, Transport, Security, and IPng) and a group of 15 other experts in the following areas: IPng, routing, mobility, realtime service, and security (see Appendix for a list of attendees). The IAB explicitly tried to balance the number of attendees from each area of expertise. Logistics limited the attendance to about 30, which unfortunately meant that many highly qualified experts were omitted from the invitation list.

In summary, the objectives of this workshop were (1) to explore the interconnections between security and the rest of the Internet architecture, and (2) to develop recommendations for the Internet community on future directions with respect to security. These objectives arose from a conviction in the IAB that the two most important problem areas for the Internet architecture are scaling and security. While the scaling problems have led to a flood of activities on IPng, there has been less effort devoted to security.

Although some came to the workshop eager to discuss short-term security issues in the Internet, the workshop program was designed to focus more on long-term issues and broad principles. Thus, the meeting began with the following ground rule: valid topics of discussion should involve both security and at least one from the list: (a) routing (unicast and multicast), (b) mobility, and (c) realtime service. As a basis for initial discussion, the invitees met via email to generate a set of scenarios (see Appendix) satisfying this ground rule.

The 30 attendees were divided into three "breakout" groups, with each group including experts in all the areas. The meeting was then structured as plenary meetings alternating with parallel breakout group sessions (see the agenda in Appendix). On the third day, the groups produced text summarizing the results of their discussions. This memo is composed of that text, somewhat rearranged and edited into a single document.

The meeting process determined the character of this document. It should be regarded as a set of working notes produced by mostly-autonomous groups, containing some diversity of opinions as well as duplication of ideas. It is not the output of the "security community", but instead represents ideas about security developed by a broad spectrum of Internet experts. It is offered as a step in a process of developing viable security mechanisms and procedures for the Internet.

2. OVERVIEW

2.1 Strategic and Political Issues

Despite the workshop emphasis on architectural issues, there was considerable discussion of the real-politik of security.

For a number of years, the IETF, with IAB backing, has worked on developing PEM, which provides email security with a great deal of functionality. A question was repeatedly raised at the workshop: why has user acceptance of PEM been slow? A number of answers to this question were suggested.

(a) High-quality implementations have been slow in coming.

(b) The use of a patented technology, the RSA algorithm, violates social conventions of the Internet.

(c) Export restrictions dampen vendor enthusiasm.

(d) PEM currently depends upon a certificate hierarchy for its names, and certificates form a new and complex name space. There is no organizational infrastructure in place for creating and managing this name space.

(e) There is no directory infrastructure available for looking up certificates.

The decision to use X.500 has been a complete failure, due to the slow deployment of X.500 in the Internet. Because of UDP packet size restrictions, it is not currently feasible to store certificates in the DNS, even if the DNS were expanded to hold records for individual email users.

It seems probable that more than one, and possibly all, of these reasons are at work to discourage PEM adoption.

The baleful comment about eating: "Everything I enjoy is either immoral, illegal, or fattening" seems to apply to the cryptography technology that is required for Internet security.

2.2 Security Issues

Almost everyone agrees that the Internet needs more and better security. However, that may mean different things to different people. Four top-level requirements for Internet security were identified: end-to-end security, end-system security, secure QOS, and secure network infrastructure.

A. End-to-End Security

One requirement is to support confidentiality, authentication and integrity for end-to-end communications. These security services are best provided on an end-to-end basis, in order to minimize the number of network components that users must trust. Here the "end" may be the end system itself, or a proxy (e.g., a firewall) acting on behalf of an end system.

For point-to-point applications, the workshop felt that existing security techniques are well suited to support confidentiality, authentication and integrity services efficiently. These existing techniques include symmetric encryption applied on an end-to-end basis, message digest functions, and key management algorithms. Current work in these areas in the IETF include the PEM and Common Authentication Technologies working groups.

The group favored a strategic direction for coping with export restrictions: separate authentication from privacy (i.e., confidentiality). This will allow work to proceed on authentication for the Internet, despite government restrictions on export of privacy technology. Conversely, it will allow easy deployment of privacy without authentication, where this is appropriate.

The workshop explored the implications of multicasting for end-to-end security. Some of the unicast security techniques can be applied directly to multicast applications, while others must be modified. Section 6.2 contains the results of these discussions; in summary, the conclusions were:

a) Existing technology is adequate to support confidentiality, authentication, and integrity at the level of an entire multicast group. Supporting authentication and integrity at the level of an individual multicast source is performance-limited and will require technology advances.

b) End-to-end controls should be based on end system or user identifiers, not low level identifiers or locator information. This requirement should spawn engineering work which consists of applying known key distribution

and cryptographic techniques.

B. End-System Security

Every host has its own security defenses, but the strength of these defenses depends upon the care that is taken in administering them. Careful host security administration means plugging security holes in the kernel and applications as well as enforcing discipline on users to set good (hard to crack) passwords.

Good security administration is labor-intensive, and therefore organizations often find it difficult to maintain the security of a large number of internal machines. To protect their machines from outside subversion, organizations often erect an outer security wall or "perimeter". Machines inside the perimeter communicate with the rest of the Internet only through a small set of carefully managed machines called "firewalls". Firewalls may operate at the application layer, in which case they are application relays, or at the IP layer, in which case they are firewall routers.

The workshop spent considerable time on the architecture of firewall routers. The results are contained in Section 3.

C. Secure QOS

The Internet is being extended to provide quality-of-service capabilities; this is the topic called "realtime service" in the workshop. These extensions raise a new set of security issues for the architecture, to assure that users are not allowed to attach to resources they are not authorized to use, both to prevent theft of resources and to prevent denial of service due to unauthorized traffic. The resources to be protected include link shares, service classes or queues, multicast trees, and so on. These resources are used as virtual channels within the network, where each virtual channel is intended to be used by a particular subset or "class" of packets.

Secure QOS, i.e., protection against improper virtual channel usage, is a form of access control mechanism. In general it will be based on some form of state establishment (setup) that defines authorized "classes". This setup may be done via management configuration (typically in advance and for aggregates of users), or it may be done dynamically via control information in packets or special messages (typically at the time of use by the source or receiver(s) of the

flow/data). In addition to state establishment, some form of authentication will be needed to assure that successive packets belong to the established class. The general case to be solved is the multicast group, since in general the multicast problem includes the two-party case as a subset. The workshop developed an approach to the secure QOS problem, which appears in Section 4 below.

D. Secure Network Infrastructure

Network operation depends upon the management and control protocols used to configure and operate the network infrastructure, including routers and DNS servers. An attack on the network infrastructure may cause denial-of-service from the user viewpoint, but from the network operators' viewpoint, security from attack requires authentication and integrity for network control and management messages.

Securing the routing protocols seems to be a straightforward engineering task. The workshop concluded the following.

a) All routing information exchanges should be authenticated between neighboring routers.

b) The sources of all route information should be authenticated.

c) Although authenticating the authority of an injector of route information is feasible, authentication of operations on that routing information (e.g., aggregation) requires further consideration.

Securing router management protocols (e.g., SNMP, Telnet, TFTP) is urgent, because of the currently active threats. Fortunately, the design task should be a straightforward application of existing authentication mechanisms.

Securing DNS is an important issue, but it did not receive much attention at the workshop.

2.3 DNS Names for Certificates

As noted in Section 2.1, work on PEM has assumed the use of X.509 distinguished names as the basis for issuing certificates, with public-key encryption. The most controversial discussion at the workshop concerned the possibility of using DNS (i.e., domain) names instead of X.509 distinguished names as (at least) an interim basis for Internet security.

The argument in favor of DNS names is that they are simple and well understood in the Internet world. It is easy for a computer operating in the Internet to be identified this way, and users who receive email on such machines already have DNS mailbox names. In contrast, introducing X.509 distinguished names for security will add a new layer of names. Most importantly, there is an existing administrative model for assigning DNS names. There is no administrative infrastructure for assigning X.509 distinguished names, and generating them may be too complex for early acceptance. The advocates of DNS names for certificates hope that using DNS names would encourage the widespread use of security in the Internet. It is expected that DNS names can be replaced later by a more capable naming mechanism such as X.509-based certificates.

The basic argument against DNS names as a basis for security is that they are too "weak". Their use may lead to confusion in many instances, and this confusion can only grow as more organizations and individuals attach to the Internet. Some commercial email systems employ numeric mailbox names, and in many organizations there are uncertainties such as whether "bumber@foo.edu" belongs to Bill Umber or Tom Bumber. While it is feasible to make DNS names more descriptive, there is a concern that the existing infrastructure, with millions of short, non-descriptive names, will be an impediment to adoption of more descriptive names.

It was noted that the question of what name space to use for certificates is independent of the problem of building an infrastructure for retrieving those names. Because of UDP packet size restrictions, it would not be feasible to store certificates in the DNS without significant changes, even if the DNS were expanded to hold records for individual email users.

The group was unable to reach a consensus on the issue of using DNS names for security; further discussion in the Internet community is needed.

3. FIREWALL ARCHITECTURE

3.1 Introduction

A firewall may be used to isolate a specific connected segment of Internet topology. When such a segment has multiple links to the rest of the Internet, coordinated firewall machines are required on all the links.

Firewalls may be implemented at different layers in the protocol stack. They are most commonly implemented at the application layer by forwarding (application) gateways, or at the IP (Internet) layer by filtering routers. Section 3.2 discusses application gateways. Section 3.3 concerns Internet-layer firewalls, which filter IP datagrams entering or leaving a security perimeter.

The general architectural model for a firewall should separate policy, i.e., determining whether or not the requester of a service should be granted access to that service, from control, i.e., limiting access to resources to those who have been granted access.

3.1.1 The Use for Firewalls

Firewalls are a very emotional topic in the Internet community. Some community members feel the firewall concept is very powerful because firewalls aggregate security functions in a single place, simplifying management, installation and configuration. Others feel that firewalls are damaging for the same reason: they provide "a hard, crunchy outside with a soft chewy center", i.e., firewalls foster a false sense of security, leading to lax security within the firewall perimeter. They observe that much of the "computer crime" in corporate environments is perpetrated by insiders, immune to the perimeter defense strategy. Firewall advocates counter that firewalls are important as an additional safeguard; they should not be regarded as a substitute for careful security management within the perimeter. Firewall detractors are also concerned about the difficulty of using firewalls, requiring multiple logins and other out-of-band mechanisms, and their interference with the usability and vitality of the Internet.

However, firewalls are a fact of life in the Internet today. They have been constructed for pragmatic reasons by organizations interested in a higher level of security than may be possible without them. This section will try to outline some of the advantages and disadvantages of firewalls, and some

instances where they are useful.

Consider a large organization of thousands of hosts. If every host is allowed to communicate directly with the outside world, attackers will attempt to penetrate the organization by finding the weakest host in the organization, breaching its defenses, and then using the resources of that host to extend the penetration further within the organization. In some sense, firewalls are not so much a solution to a security problem as they are a reaction to a more basic software engineering/administration problem: configuring a large number of host systems for good security. If this more basic problem could be solved, firewalls would generally be unnecessary.

It is interesting to consider the effect that implementing a firewall has upon various individuals in the organization. Consider first the effect upon an organization's most secure host. This host basically receives little or no extra protection, because its own perimeter defenses are as strong or stronger than the firewall. In addition, the firewall will probably reduce the connectivity available to this host, as well as the reliability of the communications path to the outside world, resulting in inconvenience to the user(s) of this host. From this (most secure) user's point of view, the firewall is a loss.

On the other hand, a host with poor security can "hide" behind the firewall. In exchange for a more limited ability to communicate with the outside world, this host can benefit from the higher level of security provided by the firewall, which is assumed to be based upon the best security available in the entire organization. If this host only wants to communicate with other hosts inside the organization, the outside communications limitations imposed by the firewall may not even be noticed. From this host's viewpoint, better security has been gained at little or no cost.

Finally, consider the point of view of the organization as a whole. A firewall allows the extension of the best security in the organization across the whole organization. This is a benefit (except in the case where all host perimeter defenses in the organization are equal). Centralized access control also becomes possible, which may be either a benefit or a cost, depending upon the organization. The "secure" hosts within the organization may perceive a loss, while the "unsecure" hosts receive a benefit. The cost/benefit ratio to the organization as a whole thus depends upon the relative numbers of "secure" and "unsecure" hosts in the organization.

Consider some cases where firewalls do not make sense. An individual can be thought of as an organization of one host. The security of all the host(s) is thus (trivially) identical, and by definition the best available to the organization. In this case the choice of firewall is simple. Does this individual wish to communicate with the outside or not? If not, then the "perfect" firewall is implemented (by complete disconnection). If yes, then the host perimeter will be the same as the firewall perimeter, so a firewall becomes unnecessary.

Another interesting case is an organization that consists of individuals with few shared interests. This might be the case of a service provider that sells public access to the network. An unrelated community of subscribers should probably be considered as individuals, rather than an organization. Firewalls for the whole organization may make little sense in this case.

To summarize, the benefit of a firewall depends upon the nature of the organization it protects. A firewall can be used to extend the best available protection within the organization across the entire organization, and thus be of benefit to large organizations with large numbers of poorly administered hosts. A firewall may produce little or no perceived benefit, however, to the individuals within an organization who have strong host perimeters already.

3.2 Application-Layer Firewalls

An application-layer firewall can be represented by the following diagram.

```
C <---> F <---> S
```

Here the requesting client C opens its transport connection to the firewall F rather than directly to the desired server S. One mechanism for redirecting C's request to F's IP address rather than S's could be based on the DNS. When C attempts to resolve S's name, its DNS lookup would return a "service redirection" record (analogous to an MX record) for S. The service redirection record would return the IP address of F.

C enters some authentication conversation to identify itself to F, and specifies its intention to request a specific service from S. F then decides if C is authorized to invoke this service. If C is authorized, F initiates a transport layer connection to S and begins the operation, passing requests and responses between C and

S.

A major advantage of this scenario over an IP-layer firewall is that raw IP datagrams are never passed through the firewall. Because the firewall operates at the application layer, it has the opportunity to handle and verify all data passing through it, and it may be more secure against illicit rendezvous attacks (see below).

Application layer firewalls also have important disadvantages. For full benefit, an application level firewall must be coded specifically for each application. This severely limits the deployment of new applications. The firewall also represents a new point of failure; if it ceases to be reachable, the application fails. Application layer firewalls also may affect performance more than IP-layer firewalls, depending on specific mechanisms in use.

3.3 IP-Layer Firewalls

Our model of an IP-layer firewall is a multi-ported IP router that applies a set of rules to each incoming IP datagram, to decide whether it will be forwarded. It is said to "filter" IP datagrams, based on information available in the packet headers.

A firewall router generally has a set of filtering rules, each of which specifies a "packet profile" and an "action". The packet profile specifies values for particular header fields, e.g., source and destination IP address, protocol number, and other suitable source and destination identifying information (for instance, port numbers). The set of possible information that may be used to match packets is called an "association". The exact nature of an association is an open issue.

The high-speed datagram forwarding path in the firewall processes every arriving packet against all the packet profiles of all active rules, and when a profile matches, it applies the corresponding action. Typical actions may include forwarding, dropping, sending a failure response, or logging for exception tracking. There may be a default rule for use when no other rule matches, which would probably specify a drop action.

In addition to the packet profile, some firewalls may also use some cryptographic information to authenticate the packet, as described below in section 3.3.2.

3.3.1 Policy Control Level

This section presents a model for the control of a firewall router, with some examples of specific mechanisms that might be used.

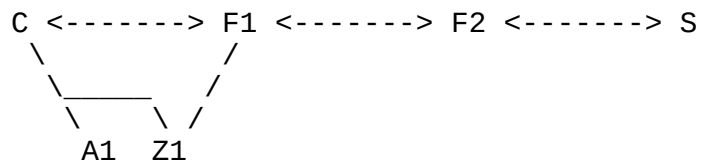
1. A client C attempts to access a service S. (Client here can mean either a person or a process - that also is an issue to be resolved.)

2. The initiation of access to that service may result in an attempt to cross one or more boundaries of protection via firewall router(s).

3. The policy control level sets filters in the firewall router(s), to permit or deny that attempt.

The policy control level consists of two distinct functions, authentication and authorization. Authentication is the function of verifying the claimed identity of a user. The authentication function should be distributed across the Internet, so that a user in one organization can be authenticated to another organization. Once a user is authenticated, it is then the job of the authorization service local to the resource being requested to determine if that user is authorized to access that resource. If authorization is granted, the filter in the firewall can be updated to permit that access.

As an aid to understanding the issues, we introduce a particular detailed mechanism. We emphasize that this mechanism is intended only as an illustrative example; actual engineering of the mechanism will no doubt lead to many changes. Our mechanism is illustrated by the following sketch. Here a user wishes to connect from a computer C behind firewall F1, to a server S behind firewall F2. A1 is a particular authentication server and Z1 is a particular authorization server.

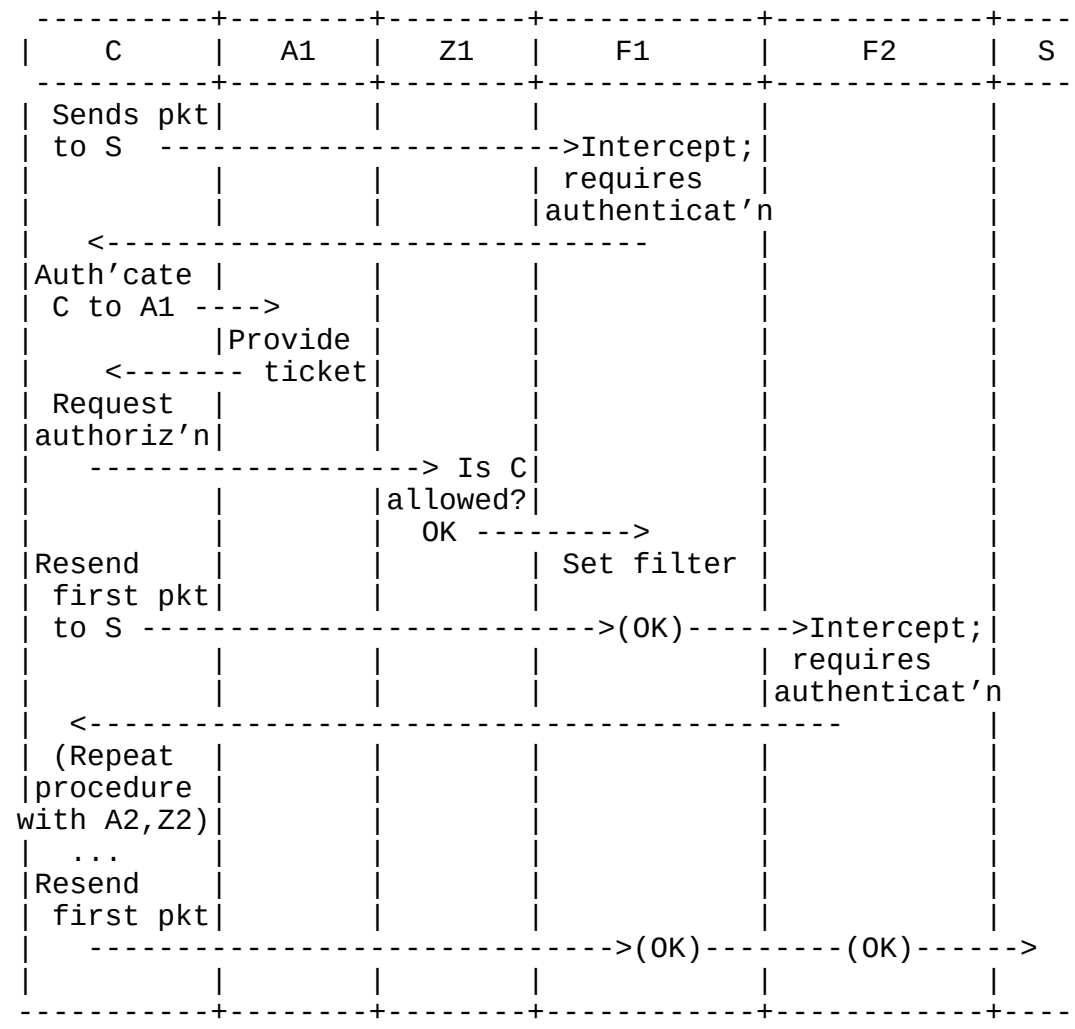


C attempts to initiate its conversation by sending an initial packet to S. C uses a normal DNS lookup to resolve S's name, and uses normal IP routing mechanisms. C's packet reaches

firewall router F1, which rejects the packet because it does not match any acceptable packet profile. F1 returns an "Authentication Required" error indication to C, including a list of authentication/authorization servers that F1 trusts. This indication might be a new type of ICMP Destination Unreachable packet, or some other mechanism for communicating with C.

When C receives the error indication, authenticates itself with A1, one of the authentication servers listed in the error indication, after validating A1's identity. C then requests authorization from server Z1 (using a ticket provided by A1), informs Z1 of the application it wishes to perform, and provides a profile for the packets it wishes to pass through F1. Z1 then performs an authorization function to decide whether to allow C to penetrate F1. If C is to be allowed, Z1 then informs the firewall F1 to allow packets matching the packet profile to pass through the firewall F1.

After C's packets penetrate F1, they may again be rejected by a second firewall F2. C could perform the same procedures with authentication server A2 and authorization server Z2, which F2 trusts. This is illustrated by the following schematic diagram of the sequence of events.



Again, we emphasize that this is only intended as a partial sketch of one possible mechanism. It omits some significant issues, including the possibility of asymmetric routes (see 3.3.3 below), and the possibility that the profiles may be different in the two directions between C and S.

We could imagine generalizing this to an arbitrary sequence of firewalls. However, security requires that each of the firewalls be able to verify that data packets actually come from C. This packet authentication problem, which is discussed in the next section, could be extremely difficult if the data must traverse more than one or possibly two firewalls in sequence.

A firewall router may require re-authentication because:

* it has been added to the path by a routing change, or

* it has timed out the profile entry, or

* it has been newly re-activated, perhaps after a crash that lost its list of acceptable profiles.

If C contacts authentication and authorization servers that S trusts, C may utilize tickets given it by these servers when initiating its use of S, and avoid re-authenticating itself to S.

Although the authentication server A1 and the authorization server Z1 are conceptually separate, they may run on the same computer or router or even be separate aspects of a single program. The protocol that C speaks to an An, the protocol that C speaks to a Zn, and the protocol that Zn speaks to Fn are not specified in these notes. The authentication mechanism used with An and the packet profile required by a firewall Fn are considered matters of policy.

3.3.2 Source Authentication

We next consider how to protect against spoofing the IP source address, i.e., injecting packets that are alleged from come from C but do not. There are three classes of mechanisms to prevent such spoofing of IP-level firewalls. The mechanisms outlined here are also discussed in Section 4.3 below.

o Packet Profile Only

The lowest level of security consists of allowing the IP-layer firewall to filter packets purely on the basis of the packet profile. This is essentially the approach used by filtering routers today, with the addition of (1) authentication and authorization servers to control the filtering profiles, and (2) the automatic "Authentication Required" notification mechanism. This approach provides almost no security; it does not prevent other computers from spoofing packets that appear to be transmitted by C, or from taking over C’s transport level connection to S.

o Sealed Packets

In the second level of security, each packet is "sealed" with a secure hash algorithm. An authentication server Ai

chooses a secret and shares it with the source host S and also with the authorization server Zi, which shares the secret with the firewall Fi. Every packet that C transmits contains a hash value that depends upon both the contents of the packet and the secret value. The firewall Fi can compute the same hash function and verify that the packet was originated by a computer that knew the shared secret.

This approach does raise issues of how much C trusts Zi and Fi. Since they know C's secret, Zi or Fi could spoof C. If C does not trust all Z's and F's in its path, a stronger mechanism (see below) is needed.

A more difficult problem arises in authenticating C's packets when more than one firewall lies in the path. Carrying a separate seal for each firewall that is penetrated would be costly in terms of packet size. On the other hand, in order to use a single seal, all the firewalls would have to cooperate, and this might require a much more complex mechanism than the one sketched in the previous section. Morever, it may require mutual trust among all of the authentication servers Ai and authorization servers Zi; any of these servers could undermine all the others. Another possibility to be investigated is to use hop-by-hop rather than end-to-end authentication of C's packets. That is, each firewall would substitute into the packet the hash needed by the next firewall.

Multi-firewall source authentication is a difficult problem that needs more investigation.

- Packet Signatures

  In the third level of security, each packet is "signed" using a public/private key algorithm. C shares its public key with Zn, which shares it with Fn. In this scenario, C can safely use one pair of keys for all authorization servers and firewalls. No authorization server or firewall can spoof C because they cannot sign packets correctly.

  Although packet signing gives a much higher level of security, it requires public key algorithms that are patented and currently very expensive to compute; their time must be added to that for the hash algorithm. Also, signing the hash generally makes it larger.

3.3.3 Other Firewall Issues

- Performance

  An Internet-layer firewall has the advantage of generality and flexibility. However, filtering introduces a potential performance problem. Performance may depend upon the number and position of the packet fields used for filtering, and upon the number of rules against which a packet has to be matched.

  Denial of service attacks require that the per-packet rule matching and the drop path be able to keep up with the interface speed.

- Multicasting

  To allow multicast traffic to penetrate a firewall, the rule that is needed should be supplied by the receiver rather than the sender. However, this will not work with the challenge mechanism outlined in Section 3.3.1, since "Authentication Required" notifications would be sent to the sender, not to the receiver(s).

  Multicast conversations may use any of the three levels of security described in the previous section, but all firewalls will have to share the same secret with the originator of the data stream. That secret would have to be provided to the receivers through other channels and then passed to the firewalls at the receivers' initiative (in much the same way that resources are reserved at receiver's initiative in RSVP).

- Asymmetric Routing

  Given a client computer C utilizing a service from another computer C through a firewall F: if the packets returning from S to C take a different route than packets from C to S, they may encounter another firewall F' which has not been authorized to pass packets from S to C (unlike F, which has been). F' will challenge S rather than C, but S may not have credentials to authenticate itself with a server trusted by F'.

  Fortunately, this asymmetric routing situation is not a problem for the common case of single homed administrative domains, where any asymmetric routes converge at the firewall.

- Illicit Rendezvous

  None of these mechanisms prevent two users on opposite sides of a firewall from rendezvousing with a custom application written over a protocol that may have been authorized to run through a firewall.

  For example, if an organization has a policy that certain information is sensitive and must not be allowed outside its premises, a firewall will not be enough to enforce this policy if users are able to attach sensitive information to mail and send it outside to arbitrary parties. Similarly, a firewall will not prevent all problems with incoming data. If users import programs and execute them, the programs may have Trojan horses which disclose sensitive information or modify or delete important data. Executable code comes in many, many forms, including PostScript files, scripts for various interpreters, and even return addresses for sendmail. A firewall can detect some of these and scan for some forms of potentially hazardous code, but it cannot stop users from transforming things that look like "data" into programs.

  We consider these problems to be somewhat outside the scope of the firewall router mechanism. It is a matter of the policies implemented by the organization owning the firewalls to address these issues.

- Transparency for Security Packets

  For the mechanisms described above to operate, the "Authentication Required" notification and the authentication/authorization protocol that is used between the client computer and the authentication and authorization servers trusted by a firewall, must be passed by all firewalls automatically. This might be on the basis of the packet profiles involved in security. Alternatively, firewall routers might serve as application-layer firewalls for these types of communications. They could then validate the data they pass to avoid spoofing or illicit rendezvous.

3.3.4 Firewall-Friendly Applications

Firewall routers have problems with certain communication patterns where requests are initiated by the server, including callbacks and multiple connections (e.g., FTP). It was

suggested that it would be useful to have guidelines to application designers to help them to build 'firewall-friendly applications'. The following guidelines were suggested:

1) no inbound calls (the xterm problem),

2) fixed port numbers (no portmapper or tcpmux),

3) integral redirection is good (application gateways),

4) no redirection in the protocol,

5) 32 bit sequence numbers that are crypto-strong random #'s, and

6) fixed length and number of header fields.

Type fields are good, but they may not be needed if there are fixed port numbers.

3.3.5 Conclusions

Compared to an application-layer firewall, an IP-layer firewall scheme could provide a number of benefits:

- No extra authentication is required for end hosts.
- A single authentication protocol can be used for all intended applications.
- An IP-layer firewall causes less performance degradation.
- An IP-layer firewall may be able to crash and recover state without disturbing open TCP connections.
- Routes can shift without disturbing open TCP connections.
- There is no single point of failure.
- It is independent of application.

However, there are substantial difficult design issues to be solved, particularly in the areas of multiple firewalls, assymmetric routes, multicasting, and performance.

4. SECURE QOS FORWARDING

When the Internet supports special qualities-of-service (QOS) for particular packet flows, there will be a new set of security problems. There will be a need to authenticate and authorize users asking for those QOS values that are expensive in network resources, and it will be necessary to prevent theft of these resources and denial-of-service attacks by others. This section contains a conceptual model for these problems, which we may call secure QOS forwarding. The issues here differ from end-to-end security and firewalls, because QOS forwarding security may need to be enforced at every router along a path.

It was noted that this is not a new problem; it was stated and solved in a theoretical way in a thesis by Radia Perlman.

4.1 The Requirement for Setup

Setup is an essential part of any QOS mechanism. However, it may be argued that there are also good engineering reasons for setup in any Internet-layer security mechanism, even without QOS support. In the abstract, one could imagine a pure datagram model in which each IP packet separately carried the necessary authorizations for all the stages in the forwarding path. Realistically, this is not practical, since the security information may be both unacceptably large and computationally demanding for inclusion in every packet. This seems to imply the need for some form of state setup for security.

Thus, we presume a two stage process that moves somewhat away from the pure datagram model. In the first stage, the setup stage, some state is established in the routers (and other network elements) that describes how a subsequent stream of packets is to be treated. In the second stage, the classification stage, the arriving packets are matched with the correct state information and processed. The terminology in use today calls these different state descriptions "classes", and the process of sorting "classification".

Setup can take many forms. It could be dynamic, invoked across the network by an application as described above. The setup process could also be the manual configuration of a router by means of a protocol such as SNMP or remote login. For example, a network link, such as a link across the Atlantic, might be shared by a number of users who purchase it jointly. They might implement this sharing by configuring a router with specifications, or filters, which describe the sorts of packets that are permitted to use each share. Whether the setup is

dynamic or manual, short-lived or semi-permanent, it has the same effect: it creates packet classes in the router and defines how packets are to be classified as they arrive.

Much of the current research on extensions to IP for QOS, such as realtime service, has assumed an explicit setup phase and a classification stage. The setup stage is accomplished using protocols such as RSVP or ST-II, which also specify how the subsequent classification is to be done. Security at the setup stage would thus simply be an extension to such a protocol. It should be noted that there are alternative proposals for realtime QOS, based on an implicit setup process.

4.2 Securing the Setup Process.

To secure the setup process, we require that a setup request be accompanied by user credentials that provide a trustworthy assurance that the requester is known and is authorized to make the request in question. We refer to the credentials used in the setup phase as the high-level identification (HLID).

A simple version of this authorization would be a password on the management interface to a router (the limitations of such a password scheme are well known and not the issue here). In the case of setup requests made by individual applications, some user-specific authorization must be assumed.

While there could be any number of ways to organize the HLIDs, the objective of scaling suggests that a global framework for user naming and authentication would be useful. The choice of naming framework is discussed further in Section 5. Note that this discussion, which concerns controlling access to network resources and security devices, is distinct from end-to-end authentication and access control; however, the same authentication infrastructure could be used for both.

In general, while significant engineering effort will be required to define a setup architecture for the Internet, there is no need to develop new security techniques. However, for the security aspects of the classification process, there are significant problems related to performance and cost. We thus focus on that aspect of the overall framework in more detail.

Above, we defined the high-level ID (HLID) as that set of information presented as part of a setup request. There may also be a "low-level ID" (LLID), sometimes called a "cookie", carried in each packet to drive classification. In current proposals for IP extensions for QOS, packets are classified based on existing

packet fields, e.g., source and destination addresses, ports, and protocol type.

It is important to note that the LLID is distinct from the address of the user, at least conceptually. By stressing this distinction we make the point that the privileges of the user are not determined by the address in use. If the user's address changes, the privileges do not.

The LLID in a packet acts as a form of tag that is used by some or all routers along a path to make decisions about the sort of QOS that shall be granted to this packet. An LLID might refer to a data stream between a single source-destination address pair, or it might be more general and encompass a range of data streams. There is no requirement that the LLID embody a syntax that permits a router to discern the QOS parameters that it represents, but there also is no prohibition against imposing such a structure.

We propose that an IP datagram contain one LLID, which can be used at various stages of the network to map the packet to a class. We reject the alternative that the packet should have a variable number of LLIDs, each one for a different point in the net. Again, this is not just a security comment, but it has security implications.

The attributes of the LLID should be picked to match as broad a range of requirements as possible.

* Its duration (discussed below) must match both the needs of the security protocol, balancing robustness and efficiency, and the needs of the application, which will have to deal with renewal of the setup when the LLID expires. A useful end-node facility would be a service to renew setup requests automatically.

* The degree of trust must be high enough to meet the most stringent requirement we can reasonably meet.

* The granularity of the LLID structure must permit packet classification into classes fine-grained enough for any resource selection in the network. We should therefore expect that each separate stream of packets from an application will have a distinct LLID. There will be little opportunity for aggregating multiple streams under one LLID or one authenticator.

4.3 Validating an LLID

At a minimum, it is necessary to validate the use of an LLID in context, i.e., to ensure that it is being asserted in an authorized fashion. Unauthorized use of an LLID could result in theft of service or denial-of-service attacks, where packets not emitted by an authorized sender are accorded the QOS treatment reserved for that sender (or for a group of which the sender is a member). Thus, use of an LLID should be authenticated by routers that make QOS decisions based on that LLID. (Note that not all routers may "pay attention" to the LLID.)

In principle, the validity of an LLID assertion needs to be checked on every packet, though not necessarily at every router; it may be possible to restrict the checks to security perimeters. At those routers that must validate LLIDs, there is an obvious concern over the performance impact. Therefore, a router may adopt a less rigorous approach to LLID validation. For example, a router may elect to sample a data stream and validate some, but not all, packets. It may also elect to forward packets first and perform selective validation as a background activity. In the least stringent approach, a router might log selected packets and validate them as part of an audit activity much later.

There are several candidate techniques for validating the use of LLIDs. We have identified three basic techniques, which differ in terms of computational performance, bandwidth overhead, and effectiveness (resistance to various forms of attack).

* Digital Signatures

  The first technique entails the use of public key cryptography and digital signatures. The sender of each packet signs the packet (header and payload) by computing a one-way hash over the packet and transforming the hash value using a private key associated with the LLID. The resulting authenticator value is included in the packet header. The binding between the public key and the LLID is established through a connection setup procedure that might make use of public keys that enjoy a much longer lifetime. Using public key technology yields the advantage that any router can validate a packet, but no router is entrusted with data that would enable it to generate a packet with a valid authenticator (i.e., which would be viewed as valid by other routers.) This characteristic makes this technique ideal from the standpoint of the "principle of least privilege."

Public key cryptosystems such as RSA have the advantage that validation of a signature is much faster than signing, which reduces the router processing burden. Nonetheless, this approach is not likely to be feasible for anything other than selective checking by routers, given current public key algorithm performance.

* Sealing

The next technique is based on the use of the same type of one-way hash function used for digital signatures, but it does not require signing the hash value. Here the sender computes a one-way hash with a secret quantity (essentially a "key") appended to the packet. This process is an example of what is sometimes referred to more generically as cryptographic "sealing." The inclusion of this key at the end of the hash computation results in a hash value that is not predictable by any entity not possessing the key. The resulting hash value is the authenticator and is included in the packet header. A router validates a packet by recomputing the hash value over the received packet with the same secret quantity appended. If the transmitted hash value matches the recomputed hash value, the packet is declared valid. Unlike the signature technique, sealing implies that all routers capable of verifying a seal are also capable of generating (forging) a seal. Thus, this technique requires that the sender trust the routers not to misuse the key.

This technique has been described in terms of a single secret key shared between the sender and all the routers that need to validate packets associated with an LLID. A related alternative strategy uses the same authenticator technique, but shares the secret key on a pairwise basis, e.g., between the sender and the first router, between the first router and the next, etc. This avoids the need to distribute the secret key among a large group of routers, but it requires that the setup mechanism enable Router A to convince his neighbor (Router B) that Router A is authorized to represent traffic on a specific LLID or set of LLIDs. This might best be done by encapsulating the packet inside a wrapper that both ends of the link can validate. Once this strategy is in place, it may even be most efficient for routers to aggregate traffic between them, providing authentication not on a per-LLID basis, since the router pairs are prepared to "trust" one another to accurately represent the data stream LLIDs.

For a unicast data stream, the use of pairwise keying between routers does not represent a real change in the trust

required of the routers or of the setup mechanism, because of the symmetric sharing of the secret key. However, for a multicast connection, this pairwise keying approach is superior in that it prevents a router at one point in a multicast tree from being able to generate traffic that could be inserted at another point in the tree. At worst, a router can generate spurious, but authenticatable, traffic only for routers "below" it in the multicast tree.

Note that the use of network management fault isolation techniques, e.g., sampling router traffic statistics at different points along a data stream, should permit post hoc detection of packet forgery attacks mounted by rogue routers along a data stream path. Use of this technique could provide a deterrent to such activity by routers, further arguing for the pairwise keying approach.

The sealing technique is faster than the digital signature technique, because the incremental hash calculation (including the appended secret quantity) is much faster than the cryptographic transformation required to sign a hash. The processing burden is symmetric here, i.e., the sender and each router devote the same amount of processing power to seal a packet and to verify the seal. Also, a sealed hash may be smaller than a signed hash, even if the same function is used in both cases. (This is because the modulus size of the public key signature algorithm and any ancillary parameters tend to increase the size of the signed hash value.) Moreover, one could use a hash function with a "wide" value and truncate that value, if necessary to reduce overhead; this option is not available when the authenticator is a signed hash value.

As a variant on this technique, one could imagine a "clearinghouse" that would receive, from the sender, the secret key used to generate and validate authenticators. A router needing to validate a packet would send a copy of the packet to the clearinghouse, which would check the packet and indicate to the router whether it was a valid packet associated with the LLID in question. Obviously, this variant is viable only if the router is performing infrequent, selective packet validation. However, it does avoid the need to share the authenticator secret among all the routers that must validate packets.

For both of these techniques, there is a residual vulnerability to denial-of-service attacks based on replay of valid packets during the lifetime of a data stream. Unless

packets carry sequence numbers and routers track a sequence number window for each data stream, an (external) attacker can copy valid packets and replay them. It may be easiest to protect against this form of attack by aggregating all traffic between a pair of routers into a single flow and providing replay protection for the flow as a whole, rather than on a per data stream basis.

* Temporary Passwords

The final technique explored in the workshop takes a very different tack to packet validation. The preceding techniques compute a function of the bits in a packet and transform that value in a fashion that prevents an intruder from generating packets with valid authenticators. The ability to generate packets with valid authenticators for a given LLID requires access to a secret value that is available only to the sender, or to the sender and to routers participating in a given data stream.

In contrast, this third technique calls for the authenticator to be a short term, secret quantity that is carried in the packet header, without benefit of further protection. In essence, this technique incorporates a short term "password" into each packet header. This approach, like its predecessor, requires that all of the routers validating the LLID be privy to this authenticator. Moreover, the authenticator is visible to any other router or other equipment along the path, and thus this technique is much more vulnerable than the previous ones.

Here the same authenticator may be applied to all packets with the same LLID, since the authenticator is not a function of the packet it authenticates. In fact, this suggests that it is feasible to use the LLID as the authenticator. However, adopting this tack would not be consistent with the two previous techniques, each of which requires an explicit, separate authenticator, and so we recommend against this optimization.

Nonetheless, the fact that the authenticator is independent of the packet context makes it trivial to generate (forge) apparently authentic packets if the authenticator is intercepted from any legitimate packet. Also, if the authenticator can be guessed, an attacker need not even engage in passive wiretapping to defeat this scheme. This latter observation suggests that the authenticator must be of sufficient size to make guessing unlikely, and making the

LLID and the authenticator separate further supports this requirement.

The major advantage of this approach is one of performance. The authenticator can be validated very quickly through a simple comparison. Consistent with the need to protect against guessing attacks, the authenticator need not consume a significant amount of space in the packet header.

The use of a sequence number visible to the routers is an interesting technique to explore to make these somewhat vulnerable methods more robust. If each stream (each source of packets) numbers its packets, then an intruder attempting to use the network resource must delete the legitimate packets, which in many cases would be difficult. Otherwise, the router being attacked would notice duplicate sequence numbers and similar anomalies. The exact details of the numbering would have to be worked out, since for the legitimate stream packets might be lost, which would cause holes in the sequence space.

We do not consider here the issues of collusion, in which a user with a given LLID and authenticator deliberately shares this with another unauthorized user. This possibility should be explored, to see if there is a practical advantage to this act, and thus a real threat.

4.4 Dynamics of Setup

- Duration of LLID's

  A key question in the use of LLIDs is how long they remain valid. At one extreme, they last only a very short time, perhaps seconds. This limits the damage that can be done if the authenticator for the LLID is stolen. At the other extreme, LLIDs are semi-permanent, like credit card numbers. The techniques proposed above for securing the LLID traded strength for efficiency, under the assumption that the peril was limited by the limited validity of the LLID.

  The counterbalancing advantage of long-term or semi-permanent LLIDs is that it becomes practical to use primitive setup techniques, such as manual configuration of routers to establish packet classes. This will be important in the short run, since deployment of security and dynamic resource allocation protocols may not exactly track in time.

We conclude that the correct short-term action is to design LLIDs under the assumption that they are fairly short lived, and to tolerate, in the short run, a longer period of validity. This would imply that we will get an acceptable long-term mechanism in place, which operationally will have a lower level of security at first. As we get better tools for automatic setup, we can shorten the duration of validity on a individual basis, without replacing mechanism in the packet forwarding path.

o Setup Latency

The tradition of the Internet is not to impose any setup latency in the communication path between end nodes. This supports the classic datagram model for quick transactions, etc., and it is a feature that should be preserved.

For setup that is done "in advance", either through a management interface or by an end-node in the background, the issue of latency does not arise. The latency issue occurs for dynamic reservations made in response to a specific application request.

We observe that while latency is a key issue, it is not materially influenced by security concerns. The designers of resource reservation protocols such as RSVP and ST-II are debating the latency of these protocols today, absent security. Adding an authenticator to the request message will increase the processing needed to validate the request, and might even imply a message exchange with an authentication service, but should not substantially change the real time of the setup stage, which might already take time on the order of a round-trip delay. But the design of the high level authentication and authorization methods for the setup protocol should understand that this process, while not demanding at the level of the per-packet processing, is still somewhat time-critical.

One way of dealing with an expensive setup process is to set up the request provisionally and perform the validation in the background. This would limit the damage from one bad setup request to a short period of time. Note, however, that the system is still vulnerable to an attack that uses a sequence of setup requests, each of which allows unauthorized usage for at least a short period of time.

Note also that a denial-of-service attack can be mounted by flooding the setup process with invalid setup requests, all

of which need to be processed and rejected. This could prevent a valid user from setting up any state. However, denial-of-service attacks based upon flooding leave very large "finger prints"; they should not normally be an important threat. If it is a problem, it may be possible to incorporate a mechanism at the level of setup processing that is equivalent to "fair queueing", to limits the damage from a flooding attack at the packet level.

4.5 Receiver-Initiated Setup

Recent work on a QOS extension for the Internet, embodied in the RSVP protocol, uses the model that the receiver will reserve resources. This scheme is consistent with the current IP multicast paradigm, which requires the receiver to join the multicast group. The receiver reserves the resources to insure that the multicast traffic reaches the receiver with the desired QOS. In this case, it is the credentials (the HLIDs) of the receivers that will be presented to the setup phase.

Note that receiver initiation requires an explicit setup phase. Suppose setup were implicit, driven by pre-existing fields in the packet. Then there would be no way to associate a packet with a particular receiver, since in multicast, the address of the receiver never appears in the packet.

Further, it is impossible in this case to perform a setup "in advance", unless the sender and the receiver are very tightly co-ordinated; otherwise, the receiver will not know in advance what LLID will be in the packet. It is certainly impossible, in this case, for the receiver to set up "semi-permanent" reservations for multicast traffic coming to it. This, again, is not a security issue; the problem exists without adding security concerns, but the security architecture must take it into account.

4.6 Other Issues

4.6.1 Encrypting Firewalls and Bypass

Our view of security, both end node and network protection, includes the use of firewalls, which partition the network into regions of more or less trust. This idea has something in common with the encrypting-firewall model used in the military/intelligence community: red (trusted) networks partitioned from black (untrusted) networks. The very significant difference is that, in the military model, the partition uses an encryption unit that encodes as much as possible of the packet for its trip across the black network to

another red network. That is, the purpose of the encryption unit, among others, is to provide a very high degree of protection against disclosure for data housed within the red networks. In contrast, our version of a firewall is more to protect the trusted (red) region of the network from outside attacks. It is concerned both with what comes in and with what goes out. It does permit communication between a node on the trusted and nodes in the untrusted parts of the network.

We would like to be able to adapt our model of secure QOS to the case of military-style encrypting firewalls. However, this use of encryption raises a problem with our model of secure resource management, discussed above, which was based on a two-stage process of setup and classification. This model is problematic because it requires information to pass from the red region to the black region in the clear. This information includes both the setup packets themselves, if setup is done dynamically from the end node, and the classification fields (the LLIDs) in the data packets. Obviously, this information cannot be encrypted when leaving the red region of the network, since it would then be meaningless to the black net, so that the black network would be unable to make resource allocation decisions based on it.

To make this sort of control scheme work, it is necessary for the encryption device to be programmed to permit certain packets and fields in packets to pass through the encryptor in the clear. This bypass of the encryption is considered highly undesirable. In a high security situation, the process generating the bypassing information might be corrupted, with the result that information that should be controlled is removed from the secure network by hiding it in the bypassed fields of the packets.

We concluded, however, that this bypass problem is not insurmountable. The key idea, as in all cases of bypass, is to limit, rather than wholly outlaw, the information passing in the clear. To limit the information needed for bypass, one can either perform the setup as a management function totally within the black environment, or divide the process into two stages. The first stage, again totally in the black context, defines a limited number of setup situations. The second stage involves sending from the red net a very small message that selects one request to be instantiated from among the pre-defined set.

Perhaps the more difficult issue is the LLID in the packet header. If the LLID is an explicit field (as we have discussed

so far, but see below), it represents a new field in each packet, with perhaps as many as 32 bits. Again, the solution is to limit the way this field can be used. When the end-node performs a setup, it will specify the value of the LLID to be used. This fact can be observed by the red/black encryption unit, which can then limit the components of this field to the values currently in use. To further improve the situation, the encryption unit might be able to aggregate a number of flows onto one flow for the purpose of crossing the black net, which would permit a further reduction in the number of distinct LLIDs that must escape the red region.

The details of this proposal, including some important issues such as the time duration of LLIDs in this case, must be considered further. However, the initial conclusion that bypass can be incorporated into a general resource control framework is very encouraging, since it suggests that both military and commercial forms of security can be built out of the same building blocks.

4.6.2 The Principle of Consistent Privilege

A well understood principle of security is the principle of least privilege, which states that a system is most robust when it is structured to demand the least privilege from its components.

A related rule we observe is the principle of consistent privilege. This can be illustrated simply in the case of denial of service, where it is particularly relevant. For a particular route, no assumption of service can be justified unless we trust the routers to deliver the packets. If a router is corrupted and will not forward packets, the only solution is to find another route not involving this router. We do not concern ourselves here with protocols for finding new routes in the presence of a corrupted router, since this topic is properly part of another topic, securing the network infrastructure. We only observe that either we will get service from the router or we will not. If the router is corrupted, it does not matter how it chooses to attack us. Thus, as long as the router is part of a forwarding path (most generally a multicast forwarding tree), we should not hesitate to trust it in other ways, such as by giving it shared resource keys or LLID verifiers.

This illustrates the principle of consistent privilege. This principle is exploited in the scheme for hop-by-hop or pairwise use of secrets to validate LLIDs in a multicast tree. If a

single key is issued for the whole tree, then the privilege is not consistent. We only need to trust a router with respect to the nodes "below" it in the tree. If it fails to forward traffic, it can affect only those nodes. But if we give it the group key, then it can generate bogus traffic and inject it into the tree at any point, affecting traffic for other parts of the tree. If, on the other hand, we use pairwise keys, then a corrupt node can only generate bogus traffic with the key for traffic it would directly receive, which is the part of the tree it could damage anyway.

Another requirement we must place on the network concerns routing. If a firewall is in place, we must trust the routing architecture not to bypass that firewall. One way to accomplish this is to eliminate any physical path between the regions other than those that go through the firewall. Operational experience will be required to see if this simple physical limit is an acceptable constraint.

4.6.3 Implicit LLID's

We stress the importance of a strong conceptual distinction between the addresses in a packet and the LLID which is used to classify the packet. The conceptual distinction is important, but under limited circumstances it may be possible to overload some of the packet fields and create an LLID from the current packet header. For example, current packet classifiers for IPv4, which are not secure but which seem to work for classifying the packets into service classes, use a number of the packet fields together as a form of LLID: the source and destination IP addresses and ports plus the protocol type.

This sort of "implicit" LLID must be short-lived, especially if the host can change its IP address as it moves. But if the LLID is established by some sort of dynamic setup protocol, it should be possible reestablish the LLID as needed.

The current IPv4 header has no authenticator field to validate the LLID. An authenticator field could be optionally carried in an option; adding it gives robustness to network reservations. Any of the schemes described above for creating an authenticator could be used, except that if the simple password-style authenticator is used, it must be an explicit separate field, since the LLID cannot be picked randomly.

4.6.4 Security without Setup

As we describe this architecture, the setup phase is an essential part of the sequence. This suggests that the current Internet, which has no setup protocols, cannot be secured against denial-of-service attacks. It is important to explore the limits of this point. As we stressed above, setup can occur in many ways. Routers today offer management options to classify packets based on protocol types and other fields found in the header, and to use this classification to create a few fair queueing classes that can prevent one class from overloading the net to the exclusion of the others.

There are two problem here. The first is that for a setup done using a management interface, the secret that is shared among the source and the routers to validate the LLID must remain valid for a long time, and it must be manually configured. The second problem is that the granularity of the categories may be coarse. However, it has been proposed, in a thesis by Radia Perlman, that a router might create a separate fair queueing class implicitly for each source address. This approach, which uses the addresses as an implicit LLID, must have some form of authenticator for robustness. But if the LLID can be trusted, this scheme provides classification of traffic based only on an implicit setup operation. The granularity of classification is not sufficient to provide any QOS distinction. The only objective is to prevent the traffic from one source from flooding the net to the exclusion of another.

4.6.5 Validating Addresses

We make a claim here that if the LLID and the addresses in the packet are conceptually distinct, and if there is a suitable means to validate the LLID, then there is no reason to validate the addresses. For example, a packet constructed with a false source address does not seem to represent any security problem, if its LLID can be validated.

An exception to this might possibly lie in communication with mobile hosts, but it will require a complete model of threats and requirements in the mobile environment to be sure. However, we make the claim, as a starting point for discussion, that if LLIDs are distinguished from addresses, many of the security concerns with mobility are mitigated and perhaps removed. This point should be validated by more detailed consideration of the mobility problem.

4.6 Conclusions

a) It is important to conceptually separate a LLID (Low-Level IDentifier) carried in a packet from addresses in the packet.

b) There will be a single LLID carried in each packet. Although this might imply some additional state in the routers than if multiple LLIDs were used, using only one LLID choice is more scalable.

c) Hop-by-hop LLID authentication mechanisms might provide a highly scalable approach that limits the distribution of secrets. However, the robustness limitations must be investigated thoroughly.

d) Statistical sampling or after-the-fact detection mechanisms may be employed by routers to address performance concerns.

5. AN AUTHENTICATION SERVICE

The purpose of an authentication service is simply to verify names, or more precisely to verify the origin of "messages". It differs from the authorization service, which determines what services are available to an authenticated name. We expect that authentication will be an Internet-wide service, while authorization will be specific to the resources to which access is being authorized.

This "identification" function can be used in several contexts, for example:

* One-time passwords: "it is really <huitema@inria.fr> that is responding to this challenge".

* Access to a firewall: "it is really <huitema@inria.fr> that is trying to send data to host-A at port-a".

There are many Internet objects that we may want to name, e.g.,:

domain names:	sophia.inria.fr
machine names:	jupiter.inria.fr
service names:	www.sophia.inria.fr (in fact, a data base)
users:	huitema@sophia.inria.fr

processes: p112.huitema@sophia.inria.fr
p112.sophia.inria.fr

universal resource locators:
http//www.sophia.inria.fr:222/tmp/foobar

One could be tempted to believe that the authentication service will only be concerned with naming humans, as only humans are "responsible"; a process obtains some access rights because it is acting on behalf of a person. However, this is too reductive and potentially misleading. We may have to authenticate "machines" or hardware components. For example:

* When a machine boots it needs to access resources for configuring itself, but it is not yet "used" by a person; there is no user.

* On a "distributed processor", component CPUs may need to authenticate each other.

Machines do differ from users; machines cannot keep their "secrets" in the same way that people do. However, there is a big value in having a simple and extensible name space.

5.1 Names and Credentials

We make the hypothesis that the authorization services will generally use "access control lists" (ACLs), i.e., some definition of a set of authorized users. A compact way to represent such a set would be to allow "wildcard" authorizations, e.g., "anybody at <Bellcore.com>", or "any machine at <INRIA.FR>". The authentication service should be designed to facilitate the realization of the authorization service and should support "wildcards".

However, wildcards are not general enough. Assuming that we have a hierarchical name space, a wildcarded entry is limited to the naming hierarchy. For example, a name like <huitema@sophia.inria.fr> could be matched by the wildcard <*@sophia.inria.fr> or <*.inria.fr> or <*.fr>. This is useful as long as one stays at INRIA, but does not solve the generic problem. Suppose that an IETF file server at CNRI is to be accessible by all IAB members: its ACL will explicitly list the members by name.

The classic approach to naming, as exemplified in the X.500 model, is to consider that people have "distinguished names". Once one has discovered such a name through some "white pages" service, can

use it as an access key in a global directory service.

An individual may acquire authorizations from a variety of sources. Using a pure, identity-based access control system, the user would have to acquire multiple identities (i.e., distinguished names), corresponding to the roles in which she is authorized to access different services. We discuss this approach in the next section.

An alternative approach is for the user to have a very small number of identities, and to have the grantors of authorizations issue (signed) credentials granting permissions to the user, linked to her ID. These additional signed credentials are known as "capabilities". The user can then establish her identity through a generic identity credential, e.g., an X.509 certificate, and can establish authorization by presenting capabilities as required. This is somewhat analogous to a person acquiring credit cards linked to the name on a driver's license, and presenting the appropriate credit card, plus the license for picture verification of identity.

5.2 Identity-Based Authorization

Let's open the wallet of an average person: we find several "credit cards" in it. We all have many "credit cards", e.g., company cards, credit cards, airline frequent flyers memberships, driver licenses. Each of these cards is in fact a token asserting the existence of a relation: the bank certifies that checks presented by the bearer will be paid, the traffic authorities certifies that the bearer has learned how to drive, etc. This is an example of identity-based authorization, in which an individual is given different names corresponding to different relations entered into by that individual.

If we imagine that the name space is based upon DNS (domain) names, then for example, the person mentioned above could be authenticated with the names:

customer@my-big-bank.com

customer@frequent-flyer.airline.com

The model we used here is that "the name is an association". This is consistent with name verification procedures, in which that one builds a "chain of trust" between the user and the "resource agent". By following a particular path in the trust graph, one can both establish the trust and show that the user belongs to an "authorized group".

The existence of "multiple names" for a person may or may not imply the existence of an "equivalence" relation. It may be useful to know that <huitema@sophia.inria.fr> and <huitema@iab.isoc.org> are two names for the same person, but there are many cases where the user does not want to make all his tokens visible.

5.3 Choosing Credentials

Let's consider again the example of Christian Huitema accessing a file at CNRI. He will have to interact with INRIA's outgoing firewall and with CNRI's incoming controls. Regardless of whether authorization depends upon capabilities or upon multiple association names, a different credential may be needed in each firewall on the path. For example, assuming multiple names are used, he will use an INRIA name, <huitema@sophia.inria.fr>, to be authorized by INRIA to use network resources, and he will use an IAB name, <huitema@iab.isoc.org>, to access the file server. Thus comes an obvious problem: how does he choose the credential appropriate to a particular firewall? More precisely, how does the computer program that manages the connection discover that it should use one credential in response to INRIA's firewall challenge and another in response to CNRI's request?

There are many possible answers. The program could simply pass all the user's credentials and let the remote machine pick one. This works, but poses some efficiency problems: passing all possible names is bulky, looking through many names is long. Advertising many names is also very undesirable for privacy and security reasons: one does not want remote servers to collect statistics on all the credentials that a particular user may have.

Another possibility is to let the agent that requests an authorization pass the set of credentials that it is willing to accept, e.g., "I am ready to serve CNRI employees and IAB members". This poses the same privacy and security problems as the previous solutions, although to a lesser degree. In fact, the problem of choosing a name is the same as the generic "trust path" model. The name to choose is merely a path in the authentication graph, and network specialists are expected to know how to find paths in graphs.

In the short term, it is probably possible to use a "default name" or "principal name", at least for local transactions, and to count on the user to "guess" the credential that is required by remote services. To leave the local environment we need only the local credentials; to contact a remote server we need only the destination credentials. So we need one or maybe two credentials,

which may be derived from the destination. It will be very often the case that the generic credential is enough; then wildcards; then "FTP provided" tokens.

6. OTHER ISSUES

6.1 Privacy and Authentication of Multicast Groups

Multicast applications are becoming an increasingly important part of Internet communications. Packet voice, video and shared whiteboard can be powerful productivity tools for users. For these applications to have maximum value to their users, a variety of security services will be required.

Existing techniques are directly applicable to providing privacy for a private teleconference. If each member of the conference shares a single key for a symmetric encryption algorithm (such as DES), existing point-to-point security techniques can be extended to protect communication within the group from outsiders.

However, slight modifications to existing techniques are required to accommodate the multicast environment. Each packet will require independent cryptographic processing to ensure that packets from multiple sources can be independently decrypted by the numerous receivers, particularly in the presence of lost packets. N-party authentication and key management will be required to establish the shared key among the proper group members. This can be done by extending existing two-party key management techniques pairwise. For example, the conference manager may provide the key to each member following individual authentication; for example, this could be implemented trivially using PEM technology. The overhead experienced by each host computer in the conference will be similar to that of existing point-to-point encryption applications, This overhead is be low enough that, today, software encryption can offer adequate performance to secure whiteboard and voice traffic, while hardware encryption is adequate for video.

The nature of multicast communication adds an additional requirement. Existing multicast conferences provide gradual degradation in quality as the packet loss rate increases. To be acceptable, authentication protocols must tolerate lost packets. Techniques to accomplish this efficiently need to be developed. One initial sketch is outlined below. Engineering work will be required to validate the practicality of this approach.

The use of symmetric encryption provides the members of the conference with effective protection from outsiders. However, because all members of the conference share a single key, it does not provide a means of authenticating individual conference members. In principle, existing techniques, based on one-way hash functions coupled with digital signatures based on asymmetric encryption algorithms, can provide individual authentication. One-way hash functions such as MD5 are comparable in cost to symmetric encryption. However, digital signatures are considerably more costly, both in computation and in communication size. The degree of overhead depends on the quality of authentication required.

In summary, realtime authentication at the granularity of group membership is easy and cheap, but individual authentication is costly in time and space. Over time, the costs of both communications and processing are expected to decline. It is possible that this will help make authentication at the level of individual conference participants. There are two conflicting trends: (1) increasing CPU speeds to provide symmetric encryption, and (2) increasing communication data rates. If both technologies increase proportionally, there will be no net gain, at least if the grain size is measured in terms of bits, rather than as a period in seconds.

The group felt that the correct approach to end-to-end controls is the use of encryption, as discussed above. The alternative is to control the ability of a user to join a multicast group as a listener, or as a speaker. However, we are not comfortable with the level of assurance that we can offer if we attempt to ensure end-to-end semantics using these means. Any passive penetration of the network, i.e., any wire-tap, can compromise the privacy of the transmitted information. We must acknowledge, however, that problems with deployment of encryption code and hardware, and especially problems of export controls, will create a pressure to use the tools described in Section 4 to implement a form of end-to-end control. Such a decision would raise no new issues in security technology. The shared key now used for encrypting the data could instead be used as the basis for authenticating a multicast group join request. This would require modification of the multicast packet format, but nothing more. Our concern is not the technical difficulty of this approach, but the level of assurance we can offer the user.

6.2 Secure Plug-and-Play a Must

Plug-and-play is the ability to plug a new device into a network and have it obtain the information it needs to communicate with other devices, without requiring any new configuration information. Secure plug-and-play is an important Internet requirement, and a central architectural issue is whether it can be made to scale well.

For plug-and-play operation, a new machine that is "plugged" into the network needs to:

(1) Obtain an locator so it can communicate with other devices

(2) Register or obtain a name to be identified by (e.g., machine name)

(3) Discover services available on the network (e.g., printers, routers, file servers, etc.)

(4) Discover other systems on the network so it can communicate with them.

In some environments, no security mechanisms are required because physical security and local knowledge of the users are sufficient protection. At the other end of the spectrum is a large network with many groups of users, different types of outside connections, and levels of administrative control. In such environments, similar plug-and-play capabilities are needed, but the new device must be "authenticated" before it can perform these functions. In each step in the discovery process the new device must authenticate itself prior to learning about services.

The steps might be:

- Obtain a HLID from a smart card, smart disk, or similar device.

- Authenticate itself with the first plug-and-play server using its HLID, to register a name and to find the location of other services.

- Discover services available on the network (e.g., printers, routers, file servers, etc.) based on its HLID.

- Discover other systems on the network so it can communicate with them.

The problem of taking a system out of the box and initially configuring it is similar to the problem of a mobile or portable machine that a human wants to connect to a local network temporarily in order to receive services on that network. How can the local network authenticate the human (and therefore the human's machine) and know which services this visiting machine is permitted to use?

The human must be endowed with a high level identifier (HLID) which acts as his/her passport and can be verified by the local network. This high level identifier must be globally unique and registered/assigned by some recognized authority.

When the human plugs the machine onto a local net, the machine identifies itself to the net with the human's high level identifier. If local net has a policy of permitting anyone to plug and play on its network, it will ignore the HLID and assign an address (locator), permitting the visitor unrestricted access and privileges. More likely, the local net will authenticate the HLID prior to granting the visitor an address or any privileges.

At this point, the HLID has only authenticated the visitor to the local network; the issue of which services or resources the visitor is entitled to use has not been addressed. It is desirable to develop a low-overhead approach to granting authentications to new users. This will help in the case of visitors to a site, as well as new users joining a facility.

6.3 A Short-Term Confidentiality Mechanism

Authentication has customarily been achieved using passwords. In the absence of active attacks, the greatest threat to computer system security may be the ease with which passwords can be "snooped" by the promiscuous monitoring of shared-media networks. There are known security techniques for achieving authentication without exposing passwords to interception, for example the techniques implemented in the well-known Kerberos system. However, authentication systems such as Kerberos currently operate only in isolation within organizational boundaries. Developing and deploying a global authentication infrastructure is an important objective, but it will take some years. Another useful approach in the short term is the use of a challenge-response user authentication scheme (e.g., S/Key).

One of the groups explored another interim approach to guarding passwords: introducing a readily-used confidentiality mechanism based on an encrypted TCP connection. This would operate at the IP level to encrypt the IP payload, including the TCP header, to

allow the nature as well of the contents of the communication to be kept private. It could be implemented to provide either "strict" protection (the connection fails if the other side cannot decrypt your data stream) or "loose" protection (falling back to non-private TCP if decryption fails).

Loose protection would allow interoperability with older hosts in a seamless (non-user-intrusive) manner.

One-time keys may be exchanged during the SYN handshake that starts the TCP connection. Using one-time keys avoids a need for infrastructure support and does not require trust between the organizations on the two ends of the connection. Tieing the key exchange to the SYN handshake will avoid the possibility of having the connection fully open without knowing the state of encryption on both ends of the connection. Although it may still be theoretically possible to intercept the SYN exchange and subvert the connection by an active "man-in-the-middle" attack, in practice such attacks on TCP connections are quite difficult unless the routing protocols have been subverted.

The keys could be exchanged using a new option that specifies the key exchange protocol, the data encryption algorithm, and the key to be used to decrypt the connection. It could be possible to include multiple options in the same SYN segment, specifying different encryption models; the far end would then need to acknowledge the option that it is willing to use. In this case, the lack of an acknowledgement would imply disinterest in decrypting the datastream. If a loose privacy policy were in force, the connection could continue even without an acknowledgment. The policy, "strict" or "loose", would be set by either the user or the default configuration for the machine.

One must however observe that a TCP option can carry only a limited amount of data. Efficient protection against crypto-analysis of the Diffie-Hellmann scheme may require the use of a very long modulus, e.g., 1024 bits, which cannot be carried in the 40 bytes available for TCP options. One would thus have either to define an "extended option" format or to implement encryption in a separate protocol layered between TCP and IP, perhaps using a version of "IP security". The detailed engineering of such a solution would have to be studied by a working group.

A TCP connection encryption mechanism such as that just outlined requires no application changes, although it does require kernel changes. It has important drawbacks, including failure to provide privacy for privacy for UDP, and the great likelihood of export control restrictions. If Diffie-Hellman were used, there would

also be patent issues.

7. CONCLUSIONS

As a practical matter, security must be added to the Internet incrementally. For example, a scheme that requires, as a precondition for any improvement, changes to application code, the DNS, routers and firewalls all at once will be very hard to deploy. One of the reasons the workshop explored schemes that are local to the IP layer is that we surmise that they might be easier to deploy in practice.

There are two competing observations that must shape planning for Internet security. One is the well known expression: "the best is the enemy of the good." The other is the observation that the attacks are getting better.

Finally, it should noted that the principle of least privilege, which was mentioned above, may be in contradiction to the principle of least cost.

7.1 Suggested Short-Term Actions

The general recommendation for short-term Internet security policy was that the IETF should make a list of desirable short-term actions and then reach out to work with other organizations to carry them out. Other organizations include regionals, which may be in a good position to provide site security counseling services to their customers, vendors and other providers, and other societies. We should also give input to the US government to influence their posture on security in the direction desired by the community.

A suggested preliminary list of short-term actions was developed.

- Perform external diagnostic security probes

  Organizations should be encouraged to use CRACK and other tools to check the robustness of their own passwords. It would also be useful to run a variety of security probes from outside. Since this is a very sensitive issue, some care needs to be taken to get the proper auspices for such probing.

Useful probe tools include:

ISS: Klaus (GA)
SATAN: Farmer Venema
ICEPICK: NRL

- Determine Security-Risk Publication Channels

  What channels should be used for disseminating information of security risks?

- Encourage use of one-time passwords.

  Available packages: S/Key, SecurID, Enigma, Digital Pathways.

- Develop and publish guidelines for protocol developers, for security-friendliness and firewall-friendliness.

- Control topology to isolate threats

- Set privacy policy:

  * Always

  * As much as possible

- Bring Site Security Handbook up to date

- Support use of Kerberos

The subject of the "Clipper chip" came up several times, but there was not sufficient discussion of this very complex issue for this grouip to reach a recommendation. It has been observed that there are a number of quite differing viewpoints about Clipper.

- Some people accept the government’s Clipper proposal, including key escrow by the US government and the requirement that encryption be in hardware.

- Some people don’t mind key escrow by the government in principle, but the object to the hardware requirement.

- Some people don’t mind key escrow in principle, but don’t want the government to hold the keys. They would be comfortable with having the organization which owns the data hold the keys.

- Some people don’t want key escrow at all.

- Some people don't mind the hardware or the key escrow, but they don't think this will be acceptable to other countries and thus will not work internationally.

This report takes no position on any of these viewpoints.

7.2 Suggested Medium-Term Actions

These actions require some protocol design or modification; however, they use existing security technology and require no research.

- Authentication Protocol

  There is a problem of the choice of technology. Public key technology is generally deemed superior, but it is patented and can also induce relatively long computations. Symmetric key technology (Needham-Schroeder algorithm, as used in Kerberos) has some technical drawbacks but it is not patented. A system based on symmetric keys and used only for authentication would be freely exportable without being subject to patents.

- Push Kerberos

  Engineering is needed on Kerberos to allow it to interoperate with mechanisms that use public key cryptography.

- Push PEM/RIPEM/PGP...

- Develop an authenticated DNS

- Develop a key management mechanism

- Set up a certificate server infrastructure

  Possible server mechanisms include the DNS, Finger, SNMP, Email, Web, and FTP.

- Engineer authentication for the Web

7.3 Suggested Long-Term Actions

In this category, we have situations where a threat has been identified and solutions are imaginable, but closure has not been reached on the principles.

- Executable Apps
- Router sabotage counter-measures
- Prevent Byzantine routing.
- Proxy Computing
- Decomposition of computers
- Are there "good" viruses?

APPENDIX A -- Workshop Organization

The following list of attendees indicates also the breakout group to which they were assigned.

Breakout Groups

Group I.1 Leader:
1 Christian Huitema, INRIA (IAB)

1 Steve Bellovin, AT&T
1 Bob Braden, ISI (IAB)
1 John Curran, NEARNET
1 Phill Gross, ANS (IETF/IAB)
1 Stev Knowles, FTP Software (Internet AD)
1 Barry Leiner, USRA (IAB)
1 Paul Mockapetris, ISI
1 Yakov Rekhter, IBM (IAB)
1 Dave Sincoskie, Bellcore (IAB)

Group I.2 Leader:
2 Steve Crocker, TIS (Security AD)

2 Jon Crowcroft
2 Steve Deering, PARC
2 Paul Francis, NTT
2 Van Jacobson, LBL
2 Phil Karn, Qualcomm
2 Allison Mankin, NRL (Transport AD, IPng AD)
2 Radia Perlman, Novell
2 John Romkey, ELF (IAB)
2 Mike StJohns, ARPA (IAB)

Group I.3 Leader:
3 Dave Clark, MIT

3 Deborah Estrin, USC
3 Elise Gerich, Merit (IAB)
3 Steve Kent, BBN (IAB)
3 Tony Lauck, DEC (IAB)
3 Tony Li, CISCO
3 Bob Hinden, Sun (IESG->IAB liaison, Routing AD)
3 Jun Murai, WIDE (IAB)
3 Scott Shenker, PARC
3 Abel Weinrib, Bellcore

The following were able to attend only the third day, due to a conflicting ISOC Board of Trustees meeting:

Scott Bradner, Harvard (IPng AD)
Jon Postel, ISI (IAB)

The workshop agenda was as follows.

Tues Feb 8

9:00 - 10:30 Plenary
Discuss facilities, meeting goals, agenda, organization. Establish some minimal common understandings. Assign scenarios to Breakout I groups.

10:30 - 13:00 Breakout I meetings
Each breakout group examine one or more scenarios and formulate a list of design questions. Lunch available on 11th floor.

13:00 - 15:00 Plenary
Report, discuss. Collate and shorten list of design issues. Organize Breakout II groups to work on these issues.

15:00 - 17:30 Breakout IIa meetings
Work on design issues.

Wed Feb 9

9:00 - 10:00 Plenary
Report, discuss.

10:00 - 13:30 Breakout IIb meetings
More work on design questions, develop list of requirements.

13:30 - 14:30 Plenary
Report, discuss.

15:30 - 17:30 Breakout III groups

Thurs Feb 10

9:00 - 9:30 Plenary

9:30 - 11:00 Breakout Groups (wrapup)

11:00 - 12:00 Plenary
Discuss possible short-term security recommendations

13:00 - 14:00 Plenary -- Discuss short-term security issues

14:00 - 14:30 Plenary -- Presentation by Steve Bellovin

14:30 - 16:00 Plenary -- Long- and Medium-term Recommendations

The following scenarios were used as a starting point for discussions. It distinguished security-S (security as a service to the end systems) from security-M, security as a mechanism to support other services. The workshop was intended to be primarily concerned with interactions among the following different *services*:

- Security-S
- Routing
- Multi-destination delivery (mcast-S)
- Realtime Packet scheduling (realtime)
- Mobility
- Accounting

(and maybe large-scale?)

These categories were then applied to the following scenarios:

S1. Support a private teleconference among mobile hosts connected to the Internet. [Security-S, mcast-S, realtime, mobility]

S2. The group in S1 is 1/3 the Internet, i.e., there are VERY severe scaling problems. [Security-S, mcast-S, realtime, mobility, large-scale]

S3. Charge for communication to support a video teleconference. [Accounting, realtime, mcast-S]

S4. I am travelling with my laptop. I tune in to radio channel IP-RADIO, pick-up the beacon and start using it. Who gets the bill? Why do they believe this is me? Is "me" a piece of hardware (IP address) or a certified user (PEM certificate)? [Mobility, accounting (, realtime, mcast-S)]

S5. A Politically Important Person will mcast an Internet presentation, without danger of interruptions from the audience.

S6. The travel industry wants to use Internet to deliver tickets to customer premises directly in a secure way, but the customer has only dial-up capability. [Security-S, mobility]

S7. I am traveling with my laptop and this friendly host is running the autoconfiguration protocol. I immediately get an address as "mac1.friendly.host.com". (What is the difference between my laptop and a bona fide autoconfigured local station?) [Security-S, mobility]

S8. Multiple people are connected to a subnetwork providing mobility (e.g., cellular, packet radio). The subnetwork is connected to multiple places in the "fixed" backbone. How can routing be done efficiently? [Routing, mobility]

The following scenarios that were suggested do not fit into the primary thrust of the workshop, generally because they are single-issue topics. Most of them are pure security topics and are concerned with the security perimeter. The last two do not fit into our classification system at all.

S9. XYZ corporation has two major branches on opposite ends of the world, and they want to communicate securely over the Internet, with each branch having IP-level connectivity to the other (not through application gateways).

S10. I am visiting XYZ corporation, with my laptop. I want to connect it to their LAN to read my email remotely over the Internet. Even though I am inside their corporate firewall, they want to be protect their machines from me.

S11. XYZ corporation is trying to use the Internet to support both private and public networking. It wants to provide full connectivity internally between all of its resources, and to provide public access to certain resources (analogous of anonymous ftp servers)

S12. The travel industry wants to use Internet to deliver tickets to customer premises directly in a secure way.

S13. Some hacker is deliberately subverting routing protocols, including mobile and multicast routing. Design counter measures.

S14. Part of the Internet is running IPv4 and part is running IPng (i.e. the Internet is in transition). How can we assure continued secure operation through such a transition?

S15. A corporation uses ATM to connect a number of its sites. It also uses Internet. It wants to make use of the ATM as its primary carrier, but also wants to utilize other networking technologies as appropriate (e.g., mobile radio). It wants to support all

media (data, voice, video).

Security Considerations

This memo is entirely concerned with security issues.

Authors' Addresses

Bob Braden [Editor]
USC Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90292-6695

Phone: (310) 822-1511
EMail: Braden@ISI.EDU

David Clark
MIT Laboratory for Computer Science
545 Technology Square
Cambridge, MA 02139-1986

Phone: 617-253-6003
EMail: ddc@lcs.mit.edu

Steve Crocker
Trusted Information Systems, Inc.
3060 Washington Road (Rte 97)
Glenwood, MD 21738

Phone: (301) 854-6889
EMail: crocker@tis.com

Christian Huitema
INRIA, Sophia-Antipolis
2004 Route des Lucioles
BP 109
F-06561 Valbonne Cedex
France

Phone: +33 93 65 77 15
EMail: Christian.Huitema@MIRSA.INRIA.FR