Copyright (c) 1997 IEEE. See full copyright notice at Table of Contents

Securing the Nimrod Routing Architecture

Karen E. Sirois
BBN Corporation
70 Fawcett St.
Cambridge, MA 02138
ksirois@bbn.com

Stephen T. Kent
BBN Corporation
70 Fawcett St.
Cambridge, MA 02138
kent@bbn.com

Abstract

This paper describes the work undertaken to secure Nimrod, a complex and sophisticated routing system that unifies interior and exterior routing functions. The focus of this work is countering attacks that would degrade or deny service to network subscribers. The work began with an analysis of security requirements for Nimrod, based on a hybrid approach that refines top-down requirements generation with an understanding of attack scenarios and the capabilities and limitations of countermeasures. The countermeasures selected for use here include several newly developed sequence integrity mechanisms, plus a protocol for shared secret establishment. A novel aspect of this work is the protection of subscriber traffic in support of the overall communication availability security goal.

1. Introduction

Today the Internet is used for a wide range of applications, from email to video teleconferencing. Simple connectivity and "best effort" delivery have been adequate for many applications, but as more sophisticated applications arise, they generate greater demands on network resources, e.g., requirements for bandwidth reservations. New routing architectures, e.g., ones that offer quality-of-service (QoS) routing, are emerging to meet these demands. Security plays a crucial role in these emergent architectures. To provide appropriate service to a diverse set of subscribers, and to provide this service across administrative boundaries and in the face of hostile attacks, security mechanisms must be introduced into routing systems.

Network security concerns have traditionally focused on end-to-end protection, providing authentication, confidentiality, integrity and in some cases non-repudiation, for subscriber traffic. However, the end-to-end security techniques employed to achieve these security services do not protect subscriber traffic from attacks aimed at disrupting or degrading communications (violating QoS guarantees), i.e., denial of service attacks. End-to-end security services can be satisfied even in the face of less than correct operation of a routing system. In contrast, a routing infrastructure that is not operating correctly is, by definition, offering degraded service to at least some subscribers.

This paper focuses on securing the routing infrastructure from denial of service attacks in Nimrod, a new routing architecture. Relatively little previous work has focused on securing routing infrastructures in the face of denial of service attacks. A notable exception is Perlman's thesis [Perlman], which developed a routing infrastructure that is robust in the face of Byzantine failure[1]. It guarantees that two non-faulty nodes can communicate if a non-faulty path connects them; it does not guarantee any level of service for subscriber traffic, e.g., delays are acceptable as long as the packets arrive. More recently there has been work incorporating digital signatures into OSPF, an interior routing protocol [MB]. This work provides authentication and connectionless integrity on an end-to-end basis for routing control messages within an autonomous system. Providing security for this one aspect of the routing infrastructure is appropriate; corruption of routing updates affords high leverage for an attacker, degrading the ability of the routing infrastructure to generate correct routes[2].

Nimrod is a routing system with many components. To provide comprehensive protection against denial of service attacks, it is necessary to secure each element of the system. Providing security for only one element of a system, e.g., routing updates, still leaves it vulnerable to other forms of attack that may prove equally debilitating.

[1] Byzantine failures refer to the situation where a legitimate element of a system exhibits deliberately malicious behavior.

[2] Correct routes are those that are consistent with the constraints of the internetwork and the service requirements of the user.

For example, one novel aspect of the security features developed for Nimrod is the protection of subscriber traffic, as part of meeting availability requirements.

The methodology described in this paper was used to define security requirements for the Nimrod routing architecture, but it also is applicable to other routing systems. Similarly, the countermeasures described below meet specific requirements of the Nimrod protocols, but they are applicable to a variety of other protocols as well.

1.1 Attacks on Routing Infrastructure

Protecting the routing infrastructure from denial of service attacks requires an understanding of the types of attacks that could be levied against the infrastructure. This understanding is an important input to the security requirements generation process, described later in this paper.

There are several generic ways by which an attack can be levied against a routing infrastructure, and multiple ways to characterize such attack profiles. For example, one can classify an attack based on the point in the routing infrastructure against which the attack is launched, based on the tools needed to effect the attack, or based on the aspect of the infrastructure that the attack exploits. Examples of attack target points include inter-router links or routers themselves; examples of aspects of the routing infrastructure that may be attacked include inter-router control traffic, subscriber traffic flows, and routing algorithm stability. The following discussion explores attacks against a routing infrastructure from two different perspectives: points that can be attacked and the way attacks can be effected

A denial of service attack may be directed against any of several different aspects of a routing system. It is worth examining these classes of attacks because of the implications of the countermeasures needed to counter each class of attack.

Inter-router control traffic presents an obvious target. If an attacker manipulates this traffic as it passes between two routers, or is flooded from one router to others within a Nimrod node, then the correct operation of the routing system cannot be guaranteed and degradation of service will result. The range of attacks than can be levied against this traffic includes modification or deletion of individual messages, re-ordering or duplication of messages, generation of spurious messages, or imposition of delays on message transmission.

All but the last of these attacks is readily detected on a point-to-point basis with minimal overhead. In a multicast (e.g., flooding) environment, all of these attacks are costly to detect in terms of algorithm performance and bandwidth. Detecting control traffic delays can be easy on some point-to-point links, but is generally costly due to clock synchronization problems, especially for multicast traffic.

Another attack approach involves taking control of an element of the system, e.g., a router or a network monitoring center. If one of these elements is compromised by an attacker, then the traffic generated by this rogue element cannot be distinguished from that of a benign instance of the element. This makes this type of attack qualitatively different and much harder to deal with. In this case, the best one can hope for is an ability to limit the damage caused by the rogue element and to provide means to aid in identifying and isolating the element. Here the goal is to detect this sort of attack quickly, and then to identify and isolate the affected elements.

The injection of spurious (including duplicated, legitimate) subscriber traffic is another means of denying service. This injection of spurious subscriber traffic may affect only a single subscriber or, in a system without bandwidth reservation, many subscribers. In principle, spurious traffic is detectable as not authentic, but the performance limitations characteristic of high quality authentication algorithms may make this impractical in many instances. To detect and eliminate duplicated, legitimate traffic, routers must maintain detailed state for each traffic flow, which also may impose performance problems that would make such filtering infeasible. These sorts of attacks can be managed on a point-to-point basis, but are difficult to address in the face of compromised routing system elements.

The routing algorithms employed in a system typically are designed to operate correctly in the face of benign failures of routers or links. However, it may be possible to exploit latent vulnerabilities in these algorithms by creating system state changes that exceed the operating parameters envisioned by the designers. Because state changes result in generation of control traffic by system elements, manipulation of inter-router control traffic (even if detected) may be sufficient to degrade service by forcing routing elements to devote bandwidth and processing to propagation of such changes. Because the design assumptions for routing algorithms often are not specified, it is difficult to determine the extent to which a system may be vulnerable to these sort of attacks. To evaluate the impact of such attacks, it is usually necessary to examine the algorithm implementation in detail and to simulate the operation of a system employing the algorithm. This level of analysis is outside the scope of the current work.

A network management center (NMC) is an attractive target with regard to denial of service attacks. An attacker who can masquerade as an NMC can devastate network operation by issuing commands to deactivate or loop links, shut down routers, etc. Since most routers accept downloaded code, subversion of an NMC allows an attacker to substitute code that subverts routing functions in a wide variety of ways.

This paper explicitly does not address attacks levied via NMC protocol control paths. We assume, instead, that a future version of SNMP[3] will incorporate suitable security countermeasures to address these attacks. However, the security model adopted here does allow for the possibility that a router may be subverted, resulting in introduction of hostile code into the router. This might be effected via a variety of means, e.g., a successful attack on a network management protocol[4], physical overrun of the router site, or overrun of an NMC.

2. Nimrod routing architecture

The Nimrod routing system is currently being developed and standardized in the IETF, after initial development by BBN under DARPA funding. It is based on a routing architecture that is designed to provide service-specific routing, i.e., routes are generated based not only upon network topology, but also the service requirements of the subscriber traffic and the service offerings of network nodes. Nimrod is a departure from the existing Internet routing paradigm in that it unifies interior and exterior routing[5].

Nimrod is a scaleable routing architecture designed to operate in a large, heterogeneous, and dynamic internetwork. It provides mechanisms to limit the amount of routing information that needs to be maintained throughout an internetwork while at the same time allowing entities to acquire enough information to generate routes based upon the service requirements of subscriber traffic. An overview of Nimrod, its architectural elements and protocols is given below. A more complete description can be found in [RS].

[3] SNMP v2 was supposed to incorporate such countermeasures, but the working group failed to agree on the details of the security mechanisms as this protocol was being finalized. Hence the Internet community must wait for a future version of SNMP (or another network management protocol) to provide these security services.

[4] In fact, SNMP is not used to download software into routers, so it is appropriate to consider these attacks irrespective of the future of SNMP-based security mechanisms.

[5] A routing protocol executing within the confines of an autonomous system is referred to as an interior routing protocol, e.g., OSPF [RFC 1247]. An exterior routing protocol executes between autonomous systems, e.g., BGP [RFC 1654]

2.1 Overview

The Nimrod routing network is represented as a set of basic entities called <u>nodes</u> and <u>endpoints</u> as illustrated in Figure 1. Associated with each Nimrod entity is a set of attributes that characterize the entity. These attributes[6] are important for routing and are stored in Nimrod's distributed routing databases.

A node is implemented by a set of contiguous internetwork physical assets conceptually clustered together. Nimrod nodes can be clustered into larger nodes, and so on, resulting in a hierarchic organization of nodes with a single top-level universal node containing all other entities. The assets in a node include routing entities and the communication links that connect them, as well as various ancillary components.

Nimrod endpoints are traffic sources and destinations that are visible to other Nimrod entities through association with one or more Nimrod nodes.

Nimrod uses link state maps to generate routes for forwarding subscriber traffic. Although the route generation algorithm is not specified by Nimrod, the algorithm chosen has to be able take as input maps and subscriber service requirements and produce routes that are consistent with both. The maps contain local information about connectivity and service offerings for clusters throughout the internetwork. The maps are maintained locally (i.e., within the node they pertain to) and are automatically distributed within the parent cluster and to more distant clusters upon request (or, if subscribed to, when the topology changes). Restricting distribution of the maps reduces the amount of routing information that must be forwarded and maintained throughout the internet.

[6] Examples of entity attributes are the entity’s locator (like an address), its globally unique *identifier* and the services it offers (for nodes only).

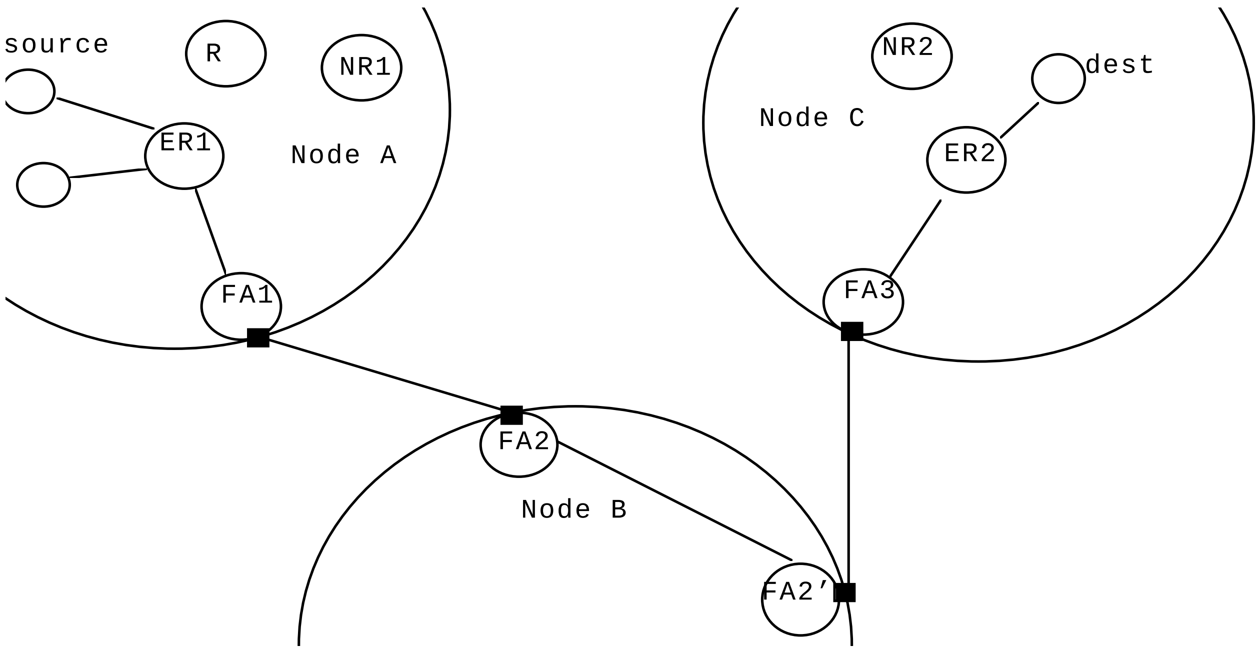

Figure 1 Example of Nimrod Topology with a Subscriber Data Path

The active routing elements in a Nimrod node are called agents. Each agent provides specific functions, e.g map generation and dissemination, and is responsible for a portion of the node's routing databases. There are four types of agents: node representatives, endpoint representatives, route agents and forwarding agents. Nimrod agents do not necessarily map one-to-one to physical devices, e.g., there may be more than one agent implemented in a single physical device. It also is not a requirement that every type of agent be present within every Nimrod node. A node may incorporate multiple instances of some agents and no instances of other agents. The minimum set of agents required for a Nimrod node consists of a forwarding agent (to form neighbor relationships and forward packets) and a node representative (to generate and distribute maps).

It is easiest to understand Nimrod with a typical example of how subscriber traffic is forwarded. As illustrated in Figure 1, the source host (or endpoint in Nimrod terminology) attempts to communicate with the destination endpoint. The source passes traffic to ER 1, its endpoint representative, an entity that participates in Nimrod routing on behalf of its subscribers. The endpoint representative contacts the route agent in its node (R) to obtain a route[7]. Armed with a route, ER 1 attempts to establish a path, using the path management protocol, with a forwarding agent that resides on the boundary to the next node in the path (FA 1). This procedure continues through the forwarding agents in each node along the path until the final destination endpoint representative (ER 2) accepts the path. Traffic can now flow over this path.

2.1.1 Databases

At the core of Nimrod lies a set of distributed databases containing routing information that is constructed, accessed, and acted upon by Nimrod agents. Each node in an internetwork contains its portion of the databases (maintained and acted upon by the agents within the node).

It is crucial that each Nimrod database be consistent with the current state of its portion of an internetwork[8]. Errors in database contents, whether intentionally or unintentionally injected, can result in impaired communications between two endpoints or, in the worst case, completely disrupt communications between all endpoints whose communications rely on those database contents. Database maintenance procedures must include rapid and reliable updating of new information as well as the removal of old information. Each database entry has a finite lifetime and is aged out of the database at the end of its lifetime. This is done to minimize the propagation and use of stale routing information.

[7] The route agent, in the process of generating a route, will obtain maps from the clusters of the internetwork for which it does not have current information.

[8] Databases are not necessarily replicated in each node. Each node contains only the portion of the distributed database that is relevant to it.

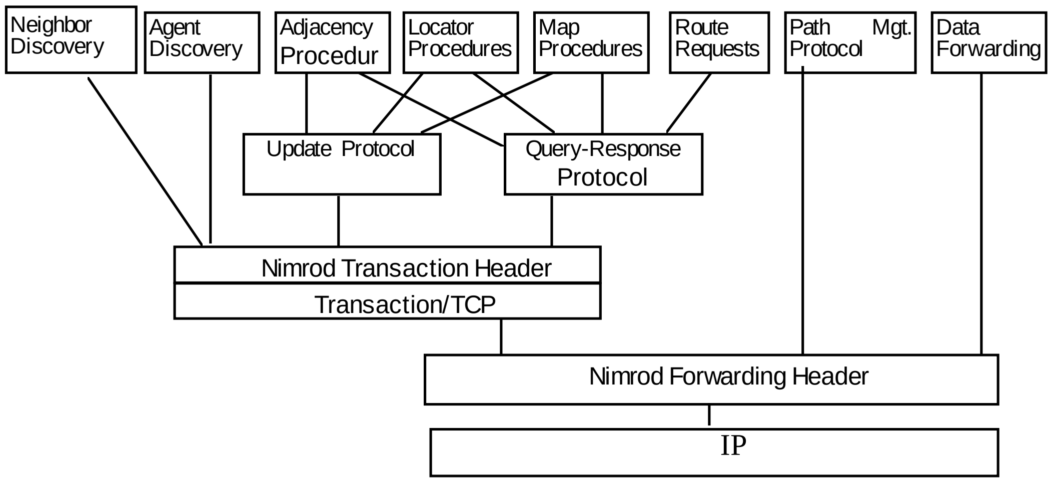

Figure 2. Nimrod Protocol Structure

2.2 Routing protocols

Nimrod uses several different routing protocols and procedures to collect and distribute routing information throughout an internetwork as shown in Figure 2. There are two layers of protocols used to build the routing infrastructure. At the lowest layer, an agent within a node uses "discovery" protocols to discover other agents in its node and to learn of external connectivity with other nodes. At a higher layer are the routing applications and procedures. Most of the routing applications and procedures are built upon just two underlying protocols, a query-response protocol and an update (constrained flooding) protocol. There is an additional protocol used to establish paths throughout the internetwork, the path management protocol.

The query-response protocol is a simple, two-way exchange of data between a pair of agents. It is used to obtain or release locators[9], create or terminate adjacencies,[10] and to request maps[11] or routes from the appropriate agents.

The update protocol is a multi-point delivery protocol used to update database contents in an efficient manner. The flooding nature of the protocol is carefully constrained by involving only a few agents per update. It is used to update locator, adjacency and map databases.

[9] Locators are like addresses and are necessary for routing. They have the property that they identify a location in the Nimrod routing network for this entity, showing the position in the clustering hierarchy.

[10] Adjacencies are neighbor relationships between nodes and are necessary for nodes to communicate.

[11] Maps, both *basic* and *abstract* (representing different levels of detail), are constructed by node representatives and describe the connectivity specifications(i.e., connectivity and service offerings) of the node.

The path management protocol is used to install and remove forwarding state information (for forwarding subscriber traffic) in agents along a route connecting source endpoints to destination endpoints. There are 5 types of messages in this protocol: SETUP, ACCEPT, STATUS, ACK and TEARDOWN, each used for a different phase of path establishment. The SETUP and ACCEPT messages are used to establish a path and TEARDOWN is used as its name indicates, to teardown a path. STATUS messages are used to obtain status about a path. The ACK message is a point-to-point message used to provide reliable transmission for the other protocol messages and to change the path label in the event of a collision[12].

There are two discovery protocols: neighbor discovery and agent discovery. Neighbor discovery is a point-to-point protocol used by each agent to discover its physical neighbors (participating in Nimrod) and to maintain connectivity over time. Agent discovery is a multi-point delivery protocol used to discover all agents within a node and to maintain reachability information about the agents. They are generated by each agent in a node and forwarded only by the forwarding agent.

3. Security requirements

The security requirements described here are defined using standard terminology from [IS 7498-2]. The philosophy used in determining the security requirements for the Nimrod routing protocols is a hybrid approach. It is primarily top-down, driven by the notion of correct operation of the protocols. Since a top-down approach

[12] Path labels are not globally unique. A path label collision occurs when a setup message is received by a forwarding agent that already has the same label in its forwarding database for another path.

often generates very coarse-grained requirements, an analysis of attack characteristics (described in Section 1.1) and of security countermeasure characteristics was employed to refine the requirements generated from the top-down perspective.

Data origin authentication is a security service that is required by all the Nimrod routing applications. The semantics associated with processing Nimrod control messages are dependent on the correct identification of the initiator in the transaction. If this identification is made incorrectly, the processing of the control message and subsequent action taken by the Nimrod agent can cause disruptions in the communications within the internetwork[13]. For some Nimrod control traffic (path management, agent discovery and update protocols), multicast authentication is required. For these protocols, a message is generated by one agent and sent unaltered[14] to many destination agents.

In addition to data origin authentication, some Nimrod routing applications require some form of access control, identity-based and/or rule-based. The actions performed by agents are often dependent upon the type and/or the identity of the initiating or transit agent. For instance, nodes control access to their resources by forming adjacencies with other nodes based upon the identity of the other node and a predetermined set of rules about with whom to form those adjacencies.

To maintain accurate and up-to-date routing data, all of the routing applications require integrity. Connectionless integrity guarantees that received data have not been altered en route. Corruption of any of the routing databases can lead to a disruption in service, hence this is a critical requirement. Sequence integrity also is required by each routing application. This service protects the recipient of routing protocol messages against replay and re-ordering attacks. The requirements for sequence integrity differ among the various routing protocols within Nimrod. Some protocols require a strict ordering of messages with no gaps, while other protocols accept out of order messages so long as replays are detected and the messages are timely. Undetected replay of valid messages can cause a variety of problems, including terminated adjacencies, corrupted databases, bandwidth reservations, etc. Any of these disruptions could result in denial of service to communicating endpoints.

Non-repudiation is a security requirement for only some of the routing applications. It is generally used to deal with Byzantine failures. In this context, a network management center functions as a third party to detect the node or agent that is maliciously sending incorrect data (and denying that it is the source of this data). For example, a compromised node representative may respond to some agent queries with correct maps of its node and to other agents with incorrect maps. The network monitoring center can detect the difference and identify the failing agent, so long as suitable non-repudiation measures are implemented.

For most of the routing applications, confidentiality is considered a secondary security requirement. In general the information that can be obtained by a passive listener to these protocol messages is limited in scope and can be acquired by other means. The map procedures are the one exception, and even in this case the information contained in the map protocol messages is limited. Maps contain information about small portions of the internetwork and are distributed in a limited area. An attacker would have to acquire maps from many different nodes throughout the internetwork to piece together a picture of the virtual Nimrod topology.

Forwarding subscriber data in an internetwork has security requirements analogous to those of the path management protocol. Data is forwarded along paths that were established using the path management protocol. Data origin authentication is required to authenticate that the traffic is associated with a valid user of the path. Connectionless and sequence integrity are clearly security requirements for data forwarding as well. Data packets manipulated en route could have many implications, e.g., modification to the Nimrod header of a packet, in particular to the path identifier, could cause the packet to be routed incorrectly.

Table 1 summarizes the security services required for each routing application within Nimrod. The rows correspond to the Nimrod protocols and procedures and the columns correspond to the security services. An “X” in the cell indicates that the security service is required for this protocol or procedure. An asterisk (*) indicates that the security service is considered a secondary requirement.

[13] For example, a map update received from an invalid source posing as a valid agent, will cause the receiving agent to update it’s map database with incorrect information. Ultimately this incorrect map database will cause incorrect routes to be generated disrupting communications between endpoints.

[14] There are selected fields in the protocol messages that may be altered by each transit agent along the path of the message.

	Data Origin Authentication	Peer Entity Authentication	Rule-Based Access Control	Identity-Based Access Control	Connectionless Integrity	Sequence Integrity	Confidentiality	Non-Repudiation
Neighbor Discovery	X				X	X		
Agent Discovery	X				X	X		
Locator Procedures	X			X	X	X	X*	
Map Procedures	X				X	X	X	X
Adjacency Procedures	X			X	X	X	X*	
Route Requests	X				X	X	X*	X
Data Forwarding	X				X	X		
Path Setup/Accept	X		X	X	X	X	X*	X
Path Teardown	X		X	X	X	X	X*	X
Path Status/Ack								

Table 1: Nimrod Security Service Requirements

4. Countermeasure design

This section details the strategy for the countermeasures proposed for Nimrod, based upon the requirements described above. The designers of the Nimrod routing architecture have made some provisions for security. These provisions have been incorporated into the countermeasure design where appropriate, with additional specifications for algorithm choices, timestamp validation mechanisms and new features where required.

Because several of the routing applications use the same base protocols (i.e., the query-response and update protocols), and because they exhibit similar security requirements, the countermeasures used to provide these security services will be implemented in the base protocols. This avoids duplication of mechanism and provides consistency for all of the routing applications. Consistent with this strategy, countermeasures have been designed for the query-response, update, path management, neighbor discovery and agent discovery protocols.

In all of the protocols, sequence integrity checks are performed before authentication checks. The reasoning for this ordering is that if the sequence integrity check fails, there is no reason to perform a more CPU intensive authentication check.

4.1 IPSEC Protection

Neighbor discovery and transport of subscriber traffic are viewed as point-to-point communications requiring data origin authentication, connectionless integrity and anti-replay protection. The chosen countermeasures for this context are keyed hashes with windowed sequence numbers. A keyed hash provides a relatively fast and flexible means of ensuring connectionless integrity and data origin authentication. Windowed sequence numbers provide replay protection, while allowing out of order packets to be accepted. When a packet arrives, it is accepted if its sequence number is within the window and has not previously been processed. If the sequence number is a duplicate or is out of range, the packet is rejected. By using IPSEC-ESP [Hughes][15] (in tunnel mode) to protect all inter-agent traffic, one can avoid replicating these functions in the Nimrod protocols. Applying IPSEC to the data traveling between neighboring agents requires encapsulation of all Nimrod packets including subscriber traffic.

In order to employ agent-to-agent IPSEC protection, neighboring agents must maintain shared secrets (keys). Key management is necessary to establish and update the shared secrets. Since none of the Nimrod protocols is designed to accommodate key management exchanges, a separate protocol has been developed for this purpose. This protocol is a two-way exchange that provides a uniform "application" interface when implemented using any of three different public-key algorithms: RSA, Diffie-Hellman, or KEA. Details of the protocol can be found in [BBN 8173]

4.2 Digital Signatures

Digital signatures are specified for Nimrod protocols that require multicast authentication and integrity and/or non-repudiation services (i.e., the path management, update and agent discovery protocols). The identity established through the use of signatures[16] also provides the basis for access control decisions. The RSA algorithm, which provides signature verification that is much faster than signature generation, is preferred in

[15] Although IPSEC-ESP did not originally incorporate replay protection, there is now an anti-replay option using sequence numbers. We propose to employ this variant, incorporating the integrity and authentication facilities, but without invoking the confidentiality facilities.

[16] The key management protocol used to establish shared secrets also provides validated identities for access control decisions.

these contexts, since each message is signed once and verified numerous times. Digital signatures provide data origin authentication and connectionless integrity on an end-to-end basis for these routing control messages, independent of the point-to-point security services offered through IPSEC-ESP.

Public key cryptography is used both for digital signature generation and validation, and for establishing shared secrets. Thus there must be a means for acquiring public keys of communicating agents. The approach adopted for this design makes use of X.509 v3 certificates. Nimrod nodes are clustered in a hierarchic manner, making it easy to construct a corresponding X.509 certificate hierarchy. The subject and issuer alternate name extensions in X509 version 3 certificates allow for a variety of choices for names. In the case of Nimrod the DNS name maybe the most meaningful choice.

In some Nimrod protocols, selected fields in a message are modified by each hop, e.g., the phase field (which determines the action to be performed) in the update message or the tracing and monitoring trailer on other protocol messages. A digital signature cannot protect such data on an end-to-end basis. However, this data is critical to the routing of these messages[17]. The use of IPSEC-ESP on a hop-by-hop basis protects the whole message from being modified on any link that the it traverses. This mutable data is still subject to modification by a malicious agent along a path; however, the cost of signing and verifying signatures on a hop-by-hop basis (which would aid in detecting such Byzantine attacks) is too great to make such countermeasures practical.

Although query-response is a point-to-point protocol, a digital signature is specified here too, to provide authentication and connectionless integrity. Some of the application procedures (e.g., locator and adjacency requests) have the property that the recipient of the query is not known by the query issuer[18]. In this case, the query is passed to transit agents along the way, e.g., the boundary forwarding agents connecting parent to child, and these agents direct the query to an appropriate node representative. Establishing a shared secret between two parties when they are not known to each other in advance entails communication beyond the existing protocols. The added delay associated with that communication might significantly impact the protocol and it might not be amortized over many subsequent message exchanges between the same parties. (In contrast, the added protocol messages used to establish shared secrets between neighbors represent a small cost that is amortized over many later messages.)

Using digital signature technology, a query can be authenticated without the query issuer knowing the identity of the respondent *a priori*. The identity of the query issuer is contained in the message (as the endpoint identifier of the originator), which allows the recipient to obtain[19] the corresponding public key and thus authenticate the message. The messages are signed by the query issuer and verified by the recipient.

The path management ACK, unlike the other path management protocol messages, is hop by hop with no forwarding. It acknowledges the receipt of each of the other path management protocol messages on a per hop basis. Since these messages are point-to-point, it is tempting to consider establishing a shared secret between each pair of agents along a path, using that secret with a keyed hash for connectionless integrity and data origin authentication. However, this approach could be prohibitive, because communicating pairs of agents for path establishment are not necessarily neighbors in terms of normal Nimrod neighbor discovery relationships. Figure 3 illustrates the problem with an example.

The setup for the top level path 3, from source A to destination E, travels as data on the paths labeled 1 and 2. The SETUP and subsequent ACK are processed by agents A (the originator), C and E (the destination) and are switched (fast path forwarding) by agents X and Y. Thus the "neighbors" for these messages are not adjacent Nimrod elements which would already share secrets for agent-to-agent protection of control and subscriber traffic (via IPSEC-AH). Because the communicating agents for path establishment could be any agents, it could be very costly to establish and maintain shared secrets between <u>each</u> possible pair of agents that participate in the path management protocol.

[17] In the case of a locator request, the query issuer does not have a locator (hence the request) and is not yet in any map. The recipient of the query, therefore, cannot obtain a route for sending the response and must use the traced route (the trailer data in the query) to send the response.

[18] For example, an agent requiring a locator sends a query to the node representative of its parent. It may not know exactly where or who this agent is, it merely knows the type of agent it is communicating with.

[19] Alternatively, the query issuer might pass its certificate, or a full certification path in the message, thus anticipating the signature validation requirements of the recipient.

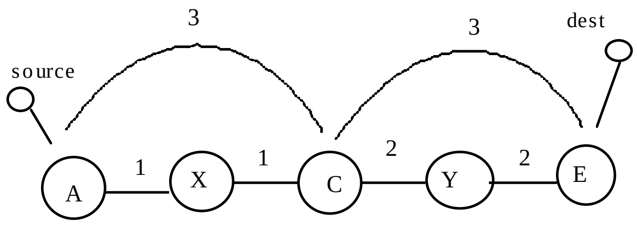

Figure 3: Nimrod Path Establishment Example

Another option is to not protect ACKs at all. In general they do not contain critical information, with the exception of one specific ACK (described below). Paths have a limited lifetime (a parameter of path establishment) and are eventually torn down if path establishment failed for some reason, e.g., if only a partial path was created. The path establishment procedure has an overall acknowledgment (the ACCEPT message) that is authenticated by a digital signature. There is one specific ACK, however, that contains critical information and should be protected. This ACK contains replacement path label information. Path labels are not globally unique in the network. The recipient of a path SETUP must verify that no other path exists in its database with the same label. If a path label collision occurs, the ACK sent to acknowledge the SETUP contains a new label that the previous hop must use to forward traffic over this path.

The use of IPSEC-ESP for agent-to-agent protection protects against counterfeit, corrupted, and replayed ACKs on a per link basis (between neighboring agents). The ACKs are protected as they traverse each link, but are not protected against a failure of an agent that lies between the communicating pair of agents. The overhead associated with digitally signing and verifying each ACK or establishing shared secrets between each possible pair of agents in the internetwork to protect against failure of an agent could be prohibitive.

However, an ACK that contains replacement path label information is critical; replaying or counterfeiting this ACK by a rogue agent has a more widespread effect causing traffic to be routed over incorrect paths. These specific ACKs should be infrequent, (path label collisions are expected to be rare), thus digitally signing them should not have a significant performance implication for path establishment. Hence this designs calls for digitally signing only those ACKs that contain replacement path label information, in conjunction with a timestamp window mechanism and saved hash values (explained below).

4.3 Timestamps

Many of the Nimrod protocols have an additional sequence integrity requirement: messages must be timely and not replayed. Routing databases must be consistent with the state of the internetwork. Incorporating old, out of date information from a replayed (or significantly delayed) message into the routing databases can cause incorrect routes to be generated. Timestamps are recommended to establish message timeliness where this is a critical requirement.

Two primary mechanisms using timestamps are defined to provide sequence integrity for the Nimrod protocols (in addition to the sequence numbers used with IPSEC). Each protocol has different requirements for sequence integrity and has its own unique characteristic, thus different mechanisms are chosen. Choosing the appropriate mechanism depends upon the type of attacks being countered, and the characteristics of the particular protocol. In each of the mechanisms, described below, a value representing the local time is placed in the message by the sending agent when the message is generated. A battery-backed clock is recommended in each router or agent, to avoid the possibility of attacks levied through time synchronization protocols. Such clocks are available in several cryptographic modules, e.g., the Fortezza card [Fortezza].

4.3.1 Timestamps with Hash Cache

A timestamp window in conjunction with a cached hash value check is recommended for the update, query-response and path management protocols. With a timestamp window, the sender of a message includes a timestamp based on his local time reference. The recipient of a message compares the timestamp in the message with its local time reference[20]. If the timestamp falls within the local acceptance window, the message can be accepted. This windowing allows a protocol to accept messages that might be delayed in transit, while still putting a limit on how old the data can be. If the timestamp window check accepts the message, then a hash is computed. (This hash is the same one that is used for the authentication check.) The hash for the message is compared to a list of hashes received from this source

[20] Since clocks throughout a network may not be tightly synchronized, it may be necessary for each router/agent to maintain an offset for each neighbor, to reflect differences in local clock values and permit wider variance without dramatically increasing the size of the window.

within the time window. If the hash is the same as any saved hash then the message is a duplicate and it is rejected; otherwise the message is accepted and its hash value is saved.[21] The hashes need be saved only for as long as the corresponding messages are acceptable by the windowing mechanism. When the corresponding message is too old, it will be rejected by the timestamp window mechanism and no comparison need be done on the hash, and the hash can be flushed from the cache.

Saving the hash values has the advantage of allowing a window of time for accepting messages without the risk of replays within that window. Clocks within the internetwork do not need to be finely synchronized, which allows them to be adjusted without causing the communicating partners of routers/agents to reject valid messages. This mechanism does require additional memory and processing. A router or agent must cache the hash values for each message received within a window of time for each possible source. If the messages for a particular routing protocol are frequent, this cache could become sizable and the additional processing used to compare each incoming message's hash value to each hash in the cache can in itself cause delays.

If the receiving router loses state and the hash cache is lost, the residual vulnerability is very small. An adversary can replay only those messages with timestamps within the window, and each message can be replayed only once. After each message has been received, it is added into the new cache and will be rejected on further attempts to replay it.

4.3.2 Increasing Timestamps

An alternative to the windowing with hash check mechanism is recommended for the agent discovery protocol, because of the frequency of these messages. In this version of a sequence integrity mechanism, the recipient of a timestamped message makes two checks on the timestamp in the message. It first uses the windowing mechanism described above (to verify the message is current) and then compares the timestamp to the timestamp of the last (authenticated) message it received from this source. If the timestamp in the message is greater than the saved value and is current, the message is accepted and its timestamp becomes the saved timestamp for this source. The recipient must not save this timestamp as the new reference point until the message is completely validated. This mechanism prevents a router or agent from incorporating any old information into its routing databases and prevents it from accepting replays.

There are two cases where the recipient does not have a saved timestamp for this particular source. The first time a message is received from a source, or any time the recipient loses state for a source (e.g., the saved timestamp is lost after a crash). These cases are considered equivalent in terms of how the recipient should respond; the recipient will use only the timestamp window mechanism under these circumstances. If the message is accepted by the windowing mechanism and validated, the timestamp from the message is saved and the agent reverts to the increasing timestamp mechanism (with this timestamp as the initial value). The initial state and post crash state for the sender are no different than the steady state, unless the clock in the sender is modified at restart. Modification of the clock in the sender or recipient can have varying effects as described below.

4.3.2.1 Clock adjustments

This timestamp mechanism is potentially vulnerable with regard to clock synchronization. There are circumstances when the clock in an agent is adjusted because of clock drift. Clocks set back because of forward clock drift at the recipient should have no effect, as the recipient uses a window of time for timeliness and the saved timestamp for comparison. Adjustments for normal clock drift are done to re-synchronize the agent's clock; thus messages received from other agents should fall within the acceptable window of time. There is a small vulnerability that a sender's clock has also drifted and the difference between the two clocks (after adjusting the recipient's) is now greater than the window. Messages will be rejected until the sender's clock is also adjusted. This vulnerability can be mitigated by choosing an appropriately sized window, and reporting the event to the NMC.

Clocks set back on the sending machine can effect communications during the time it takes the sender's clock to catch up to its pre-adjusted value. During this period, messages (sent to agents with which the sender has previously communicated) may be rejected, because the saved timestamp in these machines is greater than the new timestamp in the message. This causes a disruption in communications between these two agents for a period of time. This is not critical for agent discovery messages. When there is a disruption, the receiving agent will have outdated information for the period of time it takes the clock to catch up, However, because these messages are sent frequently, and are not generated based only upon changes in topology, as soon as the clock has reached it's pre-adjusted value the receiving agent will again start

[21] Kerberos uses a similar timestamp mechanism for sequence integrity. It uses the timestamp windowing check for message recency but instead of the hash of the message, it uses the timestamp itself to check for duplicates [NT].

accepting messages. It is also expected to be rare that a clock is adjusted in an agent at the same time the topology in the node containing the agent has changed.

Effects on communications because of adjustment for backward clock drift, in either the sender or the recipient, are similar to adjustment for forward drift at the recipient. A clock that experiences backward drift is adjusted by setting the clock ahead. Adjusting for normal clock drift re-synchronizes the clock to the correct time. Recall that a recipient uses a time window (to establish timeliness) and a saved timestamp for comparison. Regardless of whether the recipient or sender has had their clock adjusted forward, messages should still fall within the window and the new timestamp received should always be greater than the saved value. Still, a residual vulnerability exists if the adjustment for normal drift is such that the difference between the clock values on the two machines is greater than the window size. In this case, the recipient will reject (valid) messages until the window catches up with the adjusted clock.

4.4 Access Control and Non-Repudiation

In addition to authentication, many of the higher level routing applications require a form of identity-based access control (IBAC). Using the identity of the sender of a message (verified by a digital signature), access control lists are used to enforce IBAC for specific network resources. This is a straightforward extrapolation of existing practice, but making use of strong authentication technology.

In general, strong non-repudiation requires the use of a trusted third party timestamp. This requirement adds a significant overhead in bandwidth as each control message requiring non-repudiation would be sent to this third party by the sender of the message and the recipient of the message for timestamping and caching. A less stringent form of non-repudiation, useful for detecting rogue agents, has been selected for use with Nimrod. Where non-repudiation is required by the protocol, the participating agents (who are signing and verifying the routing control messages) cache these messages themselves, e.g., in a circular buffer. This adds an additional storage requirement but no additional bandwidth. If anomalous behavior indicative of a rogue agent is detected, a network management center can retrieve these caches (and verify the message signatures) to help identify the rogue agent(s).

4.5 Subscriber Traffic

Subscriber traffic should comprise the vast majority of all traffic carried in a Nimrod environment. Data is forwarded from endpoint to endpoint by the forwarding agents, using paths established by the path management protocol. Security requirements for subscriber traffic, with regard to the overall goal of protecting communication availability for this traffic, are data origin authentication, connectionless integrity, and sequence integrity.

In principle, data origin authentication and connectionless integrity could be achieved by signing each subscriber packet, and having the signature checked by each agent along the path. However, the processing overhead associated with signature generation and validation for every data packet would be prohibitive, slowing communications to an unacceptable level of performance[22]. To prevent replays of valid traffic, each packet also would require a sequence number and each agent would have to track the sequence numbers of the packets it has processed, e.g., relative to a window. Maintaining this state for each flow traversing an agent would also impose a significant processing (and perhaps storage) burden.

Instead, Nimrod will rely on IPSEC-ESP with replay protection to provide data origin authentication, and connectionless and sequence integrity, on a agent-to-agent basis. This mechanism provides protection against attacks that inject spurious or duplicated traffic onto a link; such traffic is detected and discarded, preventing it from consuming network bandwidth and other routing resources at other points in the net. Because this protection is provided only on an agent-to-agent (vs. an end-to-end) basis, there remains a vulnerability in that a rogue agent can inject spurious traffic into paths that traverse it.

At this time, the authors are aware of no acceptable (from a performance perspective) solution to eliminate this residual vulnerability. The IPSEC approach adopted here effectively counters denial of service attacks directed against the inter-agent links, and it does so at an acceptable cost. Digital signatures and end-to-end sequence numbers, although more rigorous countermeasures, are sufficiently costly in terms of resource usage that their adoption might constitute a form of denial of service.

5. Summary

This paper presented a security architecture for Nimrod, oriented towards countering attacks that would result in degradation or denial of service to Nimrod subscribers. The architecture is derived from a security requirements analysis. This analysis was driven by a top-

[22] Also, a signature would add at least 40-100 bytes to each packet, and this constitutes a significant bandwidth overhead for small packets.

down model of “correct operation” and refined by knowledge of capabilities and limitations of both attacks and countermeasures. The countermeasures developed or selected for the design address most, but not all of the requirements. In particular, the design is still vulnerable to Byzantine agents that inject spurious (or duplicate) subscriber traffic on paths that traverse these agents. Nonetheless, the design does address a wide range of attacks directed against inter-agent links, and (more highly leveraged) Byzantine agent attacks directed against the Nimrod control protocols.

6. Acknowledgments

This work is sponsored by The Air Force Materiel Command and The Defense Advanced Research Projects Agency (DARPA)/ITO under contract #F30602-95-C-0297, Internet Routing Infrastructure Security.

The authors would like to thank Pam Helinek for her contributions to the design of the Nimrod countermeasures, and Martha Steenstrup for her guidance on the Nimrod architecture and her review of this paper.

7. References

1. [BBN 8141] S. Kent, K. Sirois, D. Ellis, “Internet Routing Infrastructure Security Requirements Analysis,” BBN Report No. 8141, January 1996.
2. [BBN 8173] S. Kent, D. Ellis, P. Helinek, K. Sirois, N. Yuan, “Internet Routing Infrastructure Security Countermeasures”, BBN Report 8173, July 1996.
3. [Hughes] J. Hughes, “Combined DES-CBC, HMAC and Replay Prevention Security Transform”, Internet Draft, September, 1996.
4. [RFC1992] I. Castineyra, N. Chiappa, M. Steenstrup, “ The Nimrod Routing Architecture”, August 1996
5. [Fortezza] National Security Agency, Workstation Security Products, “Fortezza Application Implementors Guide,” revision 1.52, March 5, 1996.
6. [IS 7498-2] Information Processing Systems - OSI Reference Model - Part 2: Security Architecture, ISO/IEC document JTC 1/SC 21, 1988.
7. [MB] S.L. Murphy, M.R. Badger, “Digital Signature Protection of the OSPF Routing Protocol”, Proceedings of the Internet Society Symposium on Network and Distributed System Security”, Pages 93-102, San Diego, CA, February 1996
8. [NT] B. Clifford Neuman, Theodore Ts’o, “Kerberos: An Authentication Service for Computer Networks”, IEEE Communications Magazine, pp 33-38, September 1994
9. [RS] S. Ramanathan, M Steenstrup, “Nimrod Functionality and Protocol Specifications, Version 1”, Work in Progress, February 1996.
10. [Perlman] Radia Perlman, “Network Layer Protocols With Byzantine Robustness,” MIT/LCS/TR-429, October, 1988.