SECTION 1

WHITE PAPER ON OSI PACKET FILTERING

Walt Lazear (lazear@gateway.mitre.org)

INTRODUCTION

The capabilities for a TCP/IP router to restrict or filter traffic based on network address or application type is used in electronic network perimeter defense strategies (so-called firewalls). Many router vendors have access control commands to define which protocols or applications can be exchanged between which networks or hosts. The router implementing these commands forms a security boundary that allows only certain hosts on the inside to converse (in perhaps restrictive ways) with hosts on the outside.

No such capabilities for OSI protocols will be available in COTS products until late 1992 or early 1993. Even then, the capability is expected to be limited to OSI NSAP address filtering, without further restrictions based on OSI application type, because of the complexities of the OSI protocols.

This document discusses some of the issues in filtering OSI packets by application type. It assumes that restriction by NSAP address is possible and in place. It seeks to explore the upper layer conversations, characterize their components, and highlight the open issues with filtering them. The appendix contains results of on-line monitoring of OSI conversations for various applications.

SECTION 2

FILTERING REQUIREMENTS

2.1 General

The initial step is to characterize the flow of traffic to be filtered and to identify the source of the required information for filtering. TCP traffic contains enough information in each packet to filter the packets individually. The small number of protocol layers makes this task straightforward. Each packet contains not only the source and destination IP addresses, but also the TCP port numbers that identify the probable application (ignoring subversion and collusion for the time being). SInce the application runs directly on top of TCP, no further info is required before making a filtering decision.

The OSI protocol stack has twice the number of layers and may have sub-conversations occurring at each layer before the next higher layer is allowed to converse. Thus, each lower layer may require that packets be allowed through the filter until the topmost (application) layer conversation is begun. OSI (CLNP) packets may have the information spread across several packets and may require that a conversation of packets be monitored before a particular application can be identified. The conversation needs to be described fully before filter implementation is attempted.

2.2 Protocol Analogy

An analogy to the OSI filtering problem would be trying to make a router keep certain TELNET users off the network. TELNET merely carries characters for a conversation between a user and a host, usually a character at a time. Even if we postulate a word at a time transmission (for simplicity) and a standard way of specifying the login prompt (again, for simplicity), the problem is not simple. We have to allow a TCP connection to begin (several handshake packets) and then spot the login prompt going towards the user before we can intercept the login answer (the user's ID). This answer would then be compared against the list of “bad guys”. If the user were to be denied, the router would have to throw away all packets with the source and destination addresses and ports found in the login answer packet. The TCP connection would eventually time out and be dropped by the end hosts.

Further complicating the issue is the ability in OSI protocols to use locally-chosen selector numbers at each protocol level for contacting an application. The conventions are designed to be advertised in the X.500 Directory, and offer a more flexible and autonomous environment than the globally-registered “well-known” port numbers used in TCP. Some way to specify the selectors at each layer must be designed. In the TELNET example, you might have to implement a way of saying what the login prompt actually looked like, so the router could look for it (e.g., “Login” versus “Enter User ID”).

2.3 Inbound Connections

Connections initiated outside the security boundary and directed towards a host inside need to be checked for several items. The destination NSAP address must match an “approved” list of hosts inside the boundary. The protocol Selectors or application contexts requested must be checked against an approved list, since these determine the application or service being accessed. These Selectors are under local control, so the inside host can define the Selectors and they can be configured manually into the filtering router.

2.4 Outbound Connections

The above discussion has only addressed connections initiated from outside the perimeter. That filtering depends on an organization locally selecting the access points (protocol Selectors) and configuring these into the filtering scans. The situation for connections initiated from within the security perimeter is entirely different. In this case, the destination Selectors are chosen by the remote site. The initiating user may learn of the Selectors through use of the X.500 directory or through private correspondence with the destination system administrator. It is unreasonable and overly-constrictive to expect each destination host to have its Selectors “registered” with the filtering router. This process would be administratively burdensome and error-prone. Thus, outbound connections can easily be limited by source NSAP, but not by Selectors.

SECTION 3

OPEN ISSUES

There are a number of issues associated with the filtering process that have not been explored in detail. They are presented here for completeness and to spur discussion.

3.1 Selector Values

The local organization needs to define Selector values for the services being offered across the security boundary. This allows unambiguous determination of the legal conversations. At issue is which Selectors at which protocol layers are sufficient. Are presentation, session, transport, and network all required?

3.2 Protocol Daemon Control

How can a boundary host control which user can start an OSI daemon on which Selector? OSI has no notion of “reserved” port numbers. Should there be a mechanism such as the UNIX (not TCP) “privileged port”, which requires super-user privileges to run a network daemon for a certain range of selectors. Such a mechanism is operating system specific and non-COTS at this time.

3.3 Connection Establishment

Preliminary monitoring shows that there is an exchange of PDUs before the application is identified. This exchange is to establish a transport layer connection. There is concern that allowing PDU exchanges gratis between systems involves a security risk. NSAP level filtering can reduce the risk to a path between an outside system and an approved inside system. But the concern is that the specification might allow user data in the preliminary PDUs.

One approach is to insert a proxy at the filtering router that would intercept and respond positively to the transport connection request, but not notify the destination system until the next exchange of PDUs reveals a “legal” application. This appears infeasible because part of the transport information exchange is definition of connection identifiers for each end. Thus, the identifier would have to be supplied by the proxy and, if the proxy is to step out of the way for the rest of the conversation, it would have to be replaced by the real identifier from the destination system. Changing identifiers in mid-conversation will probably break most implementations.

3.4 Connection Re-Use

One of the concepts of OSI layering is that a lower-layer connection may be re-used by an upper-layer protocol. That is, a transport connection might be left open between two systems as a type of permanent circuit that is used again and again by the session layer (for different applications). The efficiency gained is that connection establishment handshaking is avoided for each new session. The security concern is that a filtering router would have to monitor every packet for a session-closing command (or a new session-opening command). Such a command would indicate that another application is being started and needs to be checked for legality. That is the theory. In practice, we do not know if implementations simply close the transport connection every time an application terminates (although we suspect so because of the simplicity of such an approach).

3.5 Asymmetric Routing

If there is more than one path from a source to a destination, dynamic routing can send inbound packets along one path and outbound along another. Thus, instead of a two-way street, you have two one-way streets. Filtering that depends on seeing both sides of a conversation cannot exist in this asymmetric environment. Unfortunately, it is the source that gets to choose how to reach the destination and a variety of inputs may go into that decision (only some of which are under the destination's control). Indeed, the source may launch in one direction (towards a northern entrance to an organization, for example), but intermediate routing may change the direction (to the southern entrance), based on information unknown to either source or destination. One way to defeat asymmetric routing is to limit to one the number of inbound routes, but this can have undesirable performance effects.

One way to combat the half-conversation view is to have the various routers participating in protecting an organization's boundary exchange information about the state of conversations. This introduces overhead and its own synchronization problems, but allows the rest of the routing domain to function normally. Another approach is to restrict entrance to a boundary to a single point, thus forcing all conversation through that point. This restricts redundancy and flexibility of routing paths, but does not require another infrastructure protocol to synchronize a group of boundary routers.

3.6 Dynamic Routing

A variation on the asymmetric routing problem occurs when an established conversation changes routers. That is, if we postulate a conversation that has been going through a single (filtering) router, we could envision that the router could go down. The conversation could then be routed to another (filtering) router through the use of dynamic routing protocols. Unfortunately, the new router has not seen the establishment phase of the conversation and must make a decision about the middle-of-the-conversation PDUs that have just appeared. One choice is to block the conversation and make it go through the establishment phase again. This defeats the transparency of dynamic routing, but is limited making routing static only at the boundary routers (a conversation must be routed continuously through a single router). The rest of the Internet (or organization) can be as dynamic as desirable. Another choice is to leave it alone and assume that someone else has blessed the establishment phase. This does not sit well as part of a formal security posture.

3.7 PDU Fragmentation

Each protocol layer may have the concept of splitting a chunk of user data into suitably-sized pieces. This can greatly complicate the task of filtering software. It may force the recognition and concatenation of PDU fragments at each layer before meaningful comparisons can be made. For example, the information conveyed in an association can be too large to fit in a single PDU. This can mean that the application conversation being started cannot be identified for another sequence of packets and that state information about the conversation must be kept even longer. Likewise, if one keys off the Transport connection and records the reference numbers for source and destination, one can match on these when checking packets. Unfortunately, when Network PDUs are segmented (fragmented) the Transport reference numbers only occur in the first Network segment. It's unclear what implementations do in the similar position with IP fragmentation (TCP port numbers being only in the first fragment).

3.8 Performance

Filtering adds some overhead to the normal PDU forwarding process. By preliminary inspection, there would seem to be approximately ten to twenty times the amount of detail work to unwrap OSI packets as there is with TCP packets. Part of the work is decoding ASN.1 encoding, part is the number of fields to be decoded, and part is keeping track of multiple PDUs until a conversation has revealed what application is really using the connection. In addition, there needs to be a recovery or time-out mechanism for partial conversations (that don't complete or where only one side is seen).