Packet Filtering in an IP Router

Bruce Corbridge, Robert Henig, Charles Slater – Telebit Corporation

ABSTRACT

By using existing information in packet headers, routers can provide system administrators a facility to manage network connections between computers. Host address, network number, interface, direction, protocol, and port number are parameters that may be used to implement an access control policy.

We present experiences developing the packet filtering facility in the *NetBlazer* dial-up IP router. We address the sometimes conflicting design goals of efficient performance and ease of administration by choosing internal data structures that simplify per packet lookup and then devoting 90 per cent of our code to implementing commands that maintain these tables in manner that is easy for system administrators.

Introduction

Wide area networks provide remote sites convenient access to local networks. With this increased convenience comes the often complex problem of unauthorized access to network resources. Packet filtering in an IP router can be used to manage this complexity by controlling which hosts and which services may be accessed from remote locations.

In a typical application, host address filters allow remote stations to log in to a host that is known to have carefully administered usernames and passwords but prevent access to hosts that are less secure. Protocol filters allow logins, ftp and mail, but deny remote access to X11 or NFS.

All IP routers do packet filtering to reduce network load. Broadcast packets, packets for which the router does not have a route, packets with bad IP headers, and packets that have been bouncing around over too many gateways (packets with TTL = 0) are not forwarded [2].

Routers from several manufacturers can use IP source and destination address to provide administrative control of which hosts or networks may communicate with each other. Some also use UDP or TCP port or ICMP message type to control what applications network connections are used for [4, 14, 15].

Techniques for Secure Internet Gateways

Packet filtering is one of two common techniques for implementing a secure internet gateway. The other is an application-layer gateway that provides proxy access to the Internet. A third technique uses a combination of a packet filtering gateway and an authentication server.

Packet Filtering Gateway

Digital Equipment Corporation's *screend* is packet filter that runs as a UNIX process [13]. Packets are filtered on input. The decision to accept or deny a packet may be based on host address, subnet or application (port or ICMP type code). *Screend* supports one-way or bi-directional filters, source or destination addresses, and wildcards (accept **any** value). Figure 1 shows examples of filter specifications.

If the keyword **notify** is appended to a **reject** specification, *screend* will send an ICMP destination unreachable message to the source of a rejected packet. If the keyword **log** is included, *screend* will log each application of that packet specification.

Most of the code in *screend* is devoted to parsing the configuration file and building internal data structures. The syntax for the configuration language is specified with a BNF and implemented using *lex* and *yacc*.

Screend uses a cache of recently used packet descriptions and decisions to reduce lookup time.

Application-Layer Proxy

In the application-layer gateway used at AT&T Bell Laboratories described by Cheswick [6], two specialized machines are used: *inet* and *r70*. Only *inet* is visible to the outside world. It contains a very limited amount of secret information. For

```
from host xx.lcs.mit.edu tcp port 3
    to host score.stanford.edu tcp port telnet reject;
between host sri-nic.arpa and any accept;
```

Figure 1: Examples of Filter specifications

inbound connections, a challenge response authentication service is provided by *r70*. Most user accounts use this service rather than passwords. *Inet* is used to provide anonymous ftp and a store and forward mail router. In the outbound direction customized applications such a *ptelnet* and *pftp* send connect to inet via datakit or via Ethernet through *r70*. To hosts external to Bell Labs, these connections appear to come from *inet*.

The advantage of the application layer gateway is the addresses of hosts on the internal network are completely hidden from the external network. The disadvantages are that it is more complicated and it runs slower. The *inet* gateway is a MIPS M/120. File transfer rates peak at 44Kb per second. This is more than enough a for 56Kbps link, but not fast enough to take advantage of a T1 (1.54 Mbps) link.

Packet Filtering Gateway and Authentication Server

Several sites use a combination of a packet filtering gateway and an authentication server to secure an internet connection. The MITRE Corporation uses a *cisco* router to limit the number of hosts exposed to the Internet [8]. Applications such as telnet, rlogin, and ftp have been modified to use a *SecureID* smartcard system. Connections from the Internet are validated with a challenge/response system.

Strategies for Packet Filtering Gateways

There are two benefits from filtering packets: reduction in unneeded packet traffic and protection from unwanted, perhaps malicious use of network resources. How effective routers are at providing these benefits is largely a function of the flexibility and usability of the tools provided to the system administrator.

Routing Table Solutions

All IP routers decide to route or not route a packet based on the result of routing table lookup. In principle, the routing table could be used to decide to which destinations packets may be routed and to which they may not. This solution is secure if only static routes are used. Commonly used routing protocols such as RIP are not secure [9].

Some routers can choose from which source address they will accept RIP information [4, 11]. This helps secure against acquisition of incorrect routing information that was accidentally provided. However, because RIP information is passed in an easily forged datagram an unauthorized user to fool a gateway listening to RIP into adding a route.

A solution to this problem is to maintain a destination filter table of permit and deny rules in the same format as the routing table. In addition to routing lookup, the router looks in the filter table for the output interface and sends the packet only if a permit rule is found. In the case where the external gateway has only two interfaces (e.g., one 56Kbps and one Ethernet), this method can be used to limit traffic from outside the organization to a particular set of hosts. Because the filter table is separate from the routing table, it is secure against routing protocol packets and can only be updated by a user with system administrator privileges.

Version 1.0 of Telebit Corporation's *NetBlazer* provides such a destination filter facility. Because we were able to reuse the routing table maintenance and lookup code, it took us less than a week to implement. One benefit to system administrators that this approach provides is the routing and filter commands have a consistent syntax.

Input and Output Filtering

Filtering only on the output interface is often less than optimal. Consider a router that has a 56Kbps interface to the external network and several fast local area network (LAN) interfaces to internal networks. To control the flow of traffic without knowledge of the input interface requires filters be applied to LAN interfaces. Time spent in filter table lookup tends to reduce LAN-to-LAN packet throughput. If instead filtering is done only on the 56Kbps interface both in the input direction and the output direction, the same security objectives can be achieved without slowing down LAN-to-LAN routing.

Source Address Filtering

Some organizations apply one authentication scheme to connections within the internal network and another to connections from outside the network. Connections are considered internal if the source-destination pair is within the organization's internal address space. The integrity of this assumption is improved by applying a filter to the external interface that rejects packets in which the source and destination address are both in the internal network. This prevents an external host from avoiding more rigorous authentication by masquerading as an internal host.

Protocol Port Filtering

By looking at the destination port, the router can control which daemons can be accessed. Each of the TCP services, *smtp*, *nntp*, *ftp-data*, *ftp*, *finger*, *telnet*, *login*, and *shell* begins by connecting to a well-known socket which is listening to a port reserved for that service. The same is true for UDP destined for Sun RPC, RIP, and Domain daemons. By restricting the set of destination ports that may be accessed from the external network, system administrators may control which services may be accessed from the external network. One could, for example, deny external UDP access to Sun RPC (e.g., NFS) and *routed* but allow domain name service by limiting UDP access to port 53. One could allow external access for mail and netnews by limiting TCP access to ports 25 and 119.

Special Cases

IP allows some special cases which make security through packet filtering a challenge: These include source routed packets and fragments.

Source routed packets may slip through a filtering router by appearing to be destined for an authorized host and then being forwarded to an unauthorized host. *Screend* solves this problem by not routing source routed packets [13].

Except for the first one, fragments of a packet do not contain the next level protocol information needed to do port filtering. Passing subsequent fragments is relatively harmless since it is difficult to compromise a system without sending it complete packets. On the other hand, one could flood a network with fragments. One option is to simply pass or reject fragments based on an address filter rule. This is what the *NetBlazer* does. Another is to simulate the reassembly algorithm by keeping a cache of previously seen fragments and match the decision for subsequent fragments with the one made for the first. This is what *screend* does.

The NetBlazer's IP Packet Filtering Facility

In addition to working correctly and efficiently we wanted the the *NetBlazer's* filter facility to be easy to use. We chose an internal data structure which made the per packet lookup processing simple and then put a lot of effort into providing commands to maintain these tables. To reduce code size and insure consistency, all filter commands use the same parsing function. Yet more than 50 per cent of the lines of filter source code are devoted to parsing command lines, almost 40 per cent are devoted to updating the tables, and only 10 per cent are devoted to the per packet lookup.

One-line Commands

Other implementations use configuration languages to create a filter configuration file. In these implementations, rules are applied to packets in the order in which they were entered [4, 13]. We wanted administrators to be able to update the filter table with one-line commands in much the same way that they update the routing table and we wanted the search process to be order of entry independent.

The *NetBlazer* IP filter facility contains the following one-line commands:

permit	adds a permit filter to an interface
deny	adds a deny filter to an interface
delete	deletes a filter from and interface
flush	remove all filters from an interface
lookup	tests a packet specification against the filter table
list	displays the filter table for an interface

What You See Is How It Works

When the *NetBlazer* administrator lists the filter table it displays the rules in the order in which they are applied. The *NetBlazer* also provides a lookup command which takes as arguments a source and destination address, an input interface, and optionally a protocol and destination port number. The command displays the results of input filter, routing, and output filter table lookup.

Lookup Order

Traditionally, a 32-bit IP address has been considered to have three components: network, subnet, and host [1]. Phil Karn's *ka9q* views an IP address simply as network and host with the network portion being of a variable length that is defined by the subnet mask [7]. Routing lookup is done by searching the network numbers with the longest subnet masks first. In this scheme host routes are treated as network numbers with 32-bit subnet masks. Routes are specified with a convenient *network/bits* syntax. For example

```
route add 143.191.10/24 en0
```

routes subnet 143.191.10 via the interface en0. When a packet comes in the first 24 bits of the destination address are compared to 143.191.10. If they match, the packet is sent out via en0.

The same lookup policy and specification syntax is used in the *NetBlazer's* filter facility. Except for the default behavior each filter rule has a network specification associated with it. Filter rules can specify a source network, a destination network, or a source network and a destination network (a source-destination pair). If the */bits* field is omitted, it's value is assumed to be 32 and the rule is applied only if the address is an exact match.

A Millisecond in the Life of an IP Packet

When an IP packet enters the the *NetBlazer,* the first test is to see if it was a hardware broadcast. Information in broadcast packets may be consumed by the *NetBlazer,* but it does not forward them. The second step is to determine if the packet is a valid IP packet. The packet is then tested against the input interfaces input filters (if any). If permission to route the packet is granted, routing table lookup is done. Having found a route to the destination address, the *NetBlazer* now knows the output interface. Output filter table lookup determines whether to send or reject the packet. Thus the *NetBlazer* forwards a packet only when the following conditions are true: (1) not a broadcast, (2) valid IP packet, (3) permitted by input filters, (4) a route to the destination address exists, (5) permitted by output filters.

NetBlazer Examples

The following examples apply to a *NetBlazer* with a synchronous interface (*syn0*) to the *Alternet* and several local area network connections.

The filter commands shown in Figure 2 would limit inbound access to internal network to ftp, mail, news, and domain name service requests directed to the host ftp.telebit.com.

The *NetBlazer* permits shortening commands to the shortest unique abbreviation and specifying an IP address instead of a host name. The `permit` command in figure 2 could have been specified as:

```
p 143.191.3.1 syn0 tcp 20 21 25 53 119 i d
```

The list command displays the filter table shown in Figure 3.

The filter facility has an implied on/off switch which operates on a per interface per direction basis. Because no filters exist in the output direction, outbound traffic is not filtered. When the first filter is created, a complementary default behavior is created at the same time. Adding an outbound filter enables filtering in the outbound direction. For example, to forbid the transmission of any UDP packets from the 143.191 network:

```
deny 143.191/16 syn0 udp output source
```

The list command now displays the filter table shown in Figure 4.

This has two undesirable side effects: (1) domain name service from ftp.telebit.com is no longer available to the Alternet and (2) the default of *deny* is now applied to all outbound traffic. In a prototype that Telebit showed to some customers, there were separate defaults for input and output. Having more than one default behavior was sometimes useful, but often very confusing. The side effects can be corrected with the following two commands:

```
deny 143.191/16 syn0 udp !=53 output source
permit any syn0 output source
```

The list command now shows the results displayed in Figure 5.

The following filter command prevents an external host from spoofing the authentication server by pretending to be a host on the 143.191.1 network and sending a host route in a RIP packet to the gateway.

```
deny    143.191.1/24    143.191.1/24    syn0
```

The list command shows the results displayed in Figure 6.

```
filter
permit ftp.telebit.com syn0 tcp 20 21 25 53 119 input dest
```

Figure 2: Filter commands limiting inbound access

```
Source          Destination     Interface  Protocol  I/O   Permit/Deny
                143.191.3.1/32  syn0       TCP       In    Permit
                                port rules: =20 =21 =25 =53 =119
Default                         syn0                 In    Deny
```

Figure 3: Result of list command

```
Source          Destination     Interface  Protocol  I/O   Permit/Deny
                143.191.3.1/32  syn0       TCP      In     Permit
                                port rules: =20 =21 =25 =53 =119
143.191.0.0/16                  syn0       UDP      Out    Deny
Default                         syn0                       Deny
```

Figure 4: Filter table after forbidding UDP packets

```
Source          Destination     Interface  Protocol  I/O   Permit/Deny
                143.191.3.1/32  syn0       TCP      In     Permit
                                port rules: =20 =21 =25 =53 =119
143.191.0.0/16                  syn0       UDP      Out    Deny
                                port rules: !=53
any                             syn0                Out    Permit
Default                         syn0                       Deny
```

Figure 5: Listing after correcting side effects

Because input or output was not specified, the *NetBlazer* created both an input filter and an output filter. The list command displays filter table entries in the order in which they are searched. Input filter lookup is done first then routing table lookup, then output table lookup. The most significant network number (the ones with the most bits) are searched first.

Performance

Studies indicate that traffic through routers tends to flow between pairs of addresses [10, 16]. While a cache size of two entries, Heimlich observes a hit rate of 0.48 doing wide area routing and 0.38 doing LAN-to-LAN routing. With a cache size of 16 entries, hit rates exceed 90 per cent [10].

The *NetBlazer* has a simple two-entry cache in which routes to the source address and destination address of the last packet are stored. If address-only filters are used, the appropriate filter tables are first checked. If permission to route the packet is granted the address is cached. The cache is not used when port filtering is enabled on either the input or output interface. Our observations of *NetBlazers* used to do Ethernet-to-Ethernet routing internally at Telebit find cache hit rates typically between 80 and 90 per cent. So far, we have not seen hit rates below 25 per cent.

Throughput was measured between two Ethernets with a filter table size containing four entries. The *NetBlazer* CPU is a 16-Megahertz Intel 386/SX. One way traffic from one host to another via the *NetBlazer* was varied until the maximum number of packets routed by the *NetBlazer* was observed. While this was being done, 100 ICMP Echo Requests and 100 ICMP Echo Replies per second were sent between a second pair of hosts to generate background traffic.

Maximum Total Packets Per Second Throughput

Type of Filters	PPS	Hit Rate (Per Cent)
Address and Protocol	320	-
Address Only	440	77
None	470	81

Background Traffic 100 64-byte pings per second (200 PPS total)
Foreground Traffic: 1500-byte packets

Future Work

The *NetBlazer* needs to provide more flexibility in the way it deals with ICMP. It should distinguish between different ICMP packet types and provide customer selectable notification options including no notification and a choice of Destination Unreachable type including the new RFC1122 defined types shown in Figure 7.

We would like to spoof TCP connections with the *NetBlazer* and map one connection in to two. This would hide the the Internal Network from the outside world without requiring modification of application software. Options to filter on protocol source port and to log filter decisions are needed.

Conclusions

By providing powerful, flexible filters, the *NetBlazer* minimizes the number of interfaces the system administrator must deal with. One-line commands make it easy to modify, list, and test the filter set.

The *NetBlazer* uses simple internal data structures to provide security filters while maintaining a performance level that is at least 50 per cent as fast as routing without filters. A global two-entry cache can provide average hit rates that range from 25 to 90 per cent. By making this cache two entries per interface, performance can be further improved.

```
Source          Destination     Interface  Protocol  I/O   Permit/Deny
                143.191.3.1/32  syn0       TCP       In      Permit
                                port rules: =20 =21 =25 =53 =119
143.191.1.0/24  143.191.1.0/24  syn0                 In      Deny
143.191.1.0/24  143.191.1.0/24  syn0                 Out     Deny
143.191.0.0/16                  syn0       UDP       Out     Deny
                                port rules: !=53
any                             syn0                 Out     Permit
Default                         syn0                         Deny
```

Figure 6: Listing after correcting for spoofing

```
 9 = communication with destination network administratively prohibited
10 = communication with destination host administratively prohibited
```

Figure 7: Two new RFC1122 defined types

Author Information

Bruce Corbridge received an electronic engineering degree from DeVry Institute of Technologies in 1974. Over the past sixteen years, he has been employed as a test engineer and technical writer for several companies in Silicon Valley, including ISS Sperry Univac, Diablo Systems and Convergent Technologies. He is currently working as a technical writer in the Network Products division at Telebit Corporation in Sunnyvale, CA. Reach him via U.S. Mail at Telebit Corporation; 1315 Chesapeake Terrace; Sunnyvale, CA 94086-1100. Reach him electronically at uunet!telebit!bac or bac@telebit.com.

Robert Henig received an BSCS from Northeastern University in 1984. He then spent six years working for Intel Corporation in network programming and network management roles. He joined Telebit Corporation in Sunnyvale, CA in January, 1991 to develop software for the NetBlazer. Reach him via U.S. Mail at Telebit Corporation; 1315 Chesapeake Terrace; Sunnyvale, CA 94086-1100. Reach him via electronic mail at uunet!telebit!rhenig or rhenig@telebit.com.

Charles Slater received an MS in Social Science from the California Institute of Technology in 1980. After spending 10 years in support organizations helping various Silicon Valley companies work around flaws in network products, he decided to go to work for a manufacturer to see if he could do a better job than his former vendors. He has spent a little more than a year Telebit Corporation in Sunnyvale, CA writing software for the NetBlazer. Reach him via U.S. Mail at Telebit Corporation; 1315 Chesapeake Terrace; ; Sunnyvale, CA 94086-1100. Reach him via electronic mail at either uunet!telebit!cslater or cslater@telebit.com.

References

[1] Braden, R.T., "Requirements for Internet Hosts – Communication Layers RFC 1122", October 1989

[2] Braden, R.T.; Postel, J.B., "Requirements for Internet gateways. RFC 1009", June 1987

[3] Bradner, Scott O., "Testing Multiprotocol Routers: How Fast Is Fast Enough?", Data Communications, February 1991, pp 70-86.

[4] cisco Systems, Inc., "Gateway System Manual/Software Release 8.2", Menlo Park, CA, November 1990.

[5] Carlin, Jerry M., "Internet Gateway Security Checklist", USENIX Security II Workshop, Summer 1990, pp. 145-147.

[6] Cheswick, Bill, "The Design of a Secure Internet Gateway", USENIX Anaheim Conference Proceedings, Summer 1990, pp 233-237.

[7] Ford, G. E., "Beginner's Guide to TCP/IP on the Amateur Packet Radio Network Using the KA9Q Internet Software", Version 1.0, May 9, 1990

[8] Goldberg, David S., "The MITRE User Authentication System", USENIX Security II Workshop, Summer 1990, pp. 1-4.

[9] Hedrick, C.L. "Routing Information Protocol", RFC 1058, June 1988

[10] Heimlich, Steven A., "Traffic Characterization of the NSFNET National Backbone", USENIX Washington, D.C. Conference Proceedings, Winter 1990, 0.25i7-227.

[11] Honig, Jeffrey C., "Gated(8)" manual, Cornell Theory Center, Cornell University, Ithaca, NY January 1989.

[12] McNeill, Keith, "SUMMARY: Need firewall telnet/ftp gateway", Electronic Newsgroup: alt.security, May 8, 1991.

[13] Mogul, Jeffrey C., "Simple and Flexible Datagram Access Controls", USENIX Baltimore Conference Proceedings, Summer 1989, pp 0.25i3-221.

[14] Mogul, Jeffrey C., "Re: well-behaved firewalls", Electronic Newsgroup: comp.protocols.tcp-ip, June 25, 1991.

[15] Nussbacher, Henry, "Comparison of Multiprotocol Routers", Version 1.7, Electronic Mailing List: tcp-ip@nic.ddn.mil, November 1990

[16] Paxson, Vern, "Measurements and Models of Wide Area TCP Conversations", Computer Systems Engineering Department, Lawrence Berkeley Laboratory, University of California, Berkeley, CA, LBL-30840, May 1991.

[17] Postel, J.B., "Internet Protocol. RFC 793", September 1981.

[18] Postel, J.B., "Internet Message Control Protocol. RFC 792", September 1981.

[19] Postel, J.B., "Transmission Control Protocol RFC 793", September 1981.

[20] Postel, J.B., "User Datagram Protocol. RFC 768", August 1980.