Global Information Assurance Certification Paper

The Packet Filter:
A Basic Network Security Tool
Author: Dan Strom

What is a packet filter?
According to the internet.com webopedia, packet filtering is "controlling access to a network by analyzing the incoming and outgoing packets and letting them pass or halting them based on the IP address of the source and destination. Packet filtering is one technique, among many, for implementing security firewalls."[i] Packet filtering is both a tool and a technique that is a basic building block of network security. It is a tool in that it is an instrument that aids in accomplishing a task. It is a technique because it is a method of accomplishing a task.

In the context of a TCP/IP network, a packet filter watches each individual IP datagram, decodes the header information of in-bound and out-bound traffic and then either blocks the datagram from passing or allows the datagram to pass based upon the contents of the source address, destination address, source port, destination port and/or connection status. This is based upon certain criteria defined to the packet filtering tool. The leading IP routers, including Cisco, Bay, and Lucent, can be configured to filter IP datagrams. Many operating systems can be configured for packet filtering. Packet filtering can be added to *nix operating systems. Support for packet filtering via *ipchains* is included by default in the Linux kernel. Windows NT and Windows 2000 support packet filtering. Virtually all commercial firewalls support packet filtering. Some commercial firewalls also have the capability of filtering packets based upon the state of previous packets (stateful inspection).

Why use a packet filter?
Packet filtering generally is inexpensive to implement. However it must be understood that a packet filtering device does not provide the same level of security as an application or proxy firewall. All except the most trivial of IP networks is composed of IP subnets and contain routers. Each router is a potential filtering point. Because the cost of the router has already been absorbed, additional cost for packet filtering is not required.

Packet filtering is appropriate where there are modest security requirements. The internal (private) networks of many organizations are not highly segmented. Highly sophisticated firewalls are not necessary for isolating one part of the organization from another. However it is prudent to provide some sort of protection of the production network from a lab or experimental network. A packet filtering device is a very appropriate measure for providing isolation of one subnet from another.

How does a packet filter work?
All packet filters function in the same general fashion. Operating at the *network layer* and *transport layer* of the TCP/IP protocol stack, every packet is examined as it enters the protocol stack. The network and transport headers are examined closely for the following information:

- **protocol** (IP header, network layer) – In the IP header, byte 9 (remember the byte count begins with zero) identifies the protocol of the packet. Most filter devices have the capability to differentiate between TCP, UPD, and ICMP.
- **source address** (IP header, network layer) – The source address is the 32-bit IP address of the host which created the packet.
- **destination address** (IP header, network layer) – The destination address is the 32-bit IP address of the host the packet is destined for.
- **source port** (TCP or UDP header, transport layer) – Each end of a TCP or UDP network connection is bound to a *port*. TCP ports are separate and distinct from UDP ports. Ports numbered below 1024 are reserved – they have a specifically defined use. Ports numbered above 1024 (inclusive) are known as ephemeral ports. They can be used however a vendor chooses. For a list of "well known" ports, refer to RFP1700. The source port is a pseudo-randomly assigned ephemeral port number. Thus it is often not very useful to filter on the source port.
- **destination port** (TCP or UDP header, transport layer) – The destination port number indicates a port that the packet is sent to. Each service on the destination host listens to a port. Some well-known ports that might be filtered are 20/TCP and 21/TCP - ftp connection/data, 23/TCP - telnet, 80/TCP - http, and 53/TCP - DNS zone transfers.
- **connection status** (TCP header, transport layer) – The connection status tells whether the packet is the first packet of the network session. The ACK bit in the TCP header is set to "false" or 0 if this is the first packet in the session. It is simple to disallow a host from establishing a connection by rejecting or discarding any packets which have the ACK bit set to "false" or 0.

The filtering device compares the values of these fields to rules that have been defined, and based upon the values and the rules the packet is either passed or discarded. Many filters also allow additional criteria from the *link layer* to be defined, such as the network interface where the filtering is to occur.

Packet filter example
Suppose that you want to create a simple packet-filtering only firewall based upon Linux. You have two network interface cards installed and configured for the two IP subnets. You have packet forwarding enabled between the network interfaces. At this point, you have a Linux-based router. Now you are ready to make it into a packet-filter firewall. (Further information on creating the firewall can be found in the Linux IPCHAINS-HOWTO document.) Supposing that this is your primary firewall between your internal network and the Internet, then you might want to allow only www connections originating internally and reject all else. Your ipchains configuration might look something like this:

```
ipchains –A int-ext –p tcp –dport www –j ACCEPT
ipchains –A int-ext –j REJECT
```

The first line adds the ability for connections on port 80 (www) to be accepted and pass from the internal interface to the external interface. It is added to the int-ext chain (sometimes this is referred to as the access control list).
Line two is the catch-all. It rejects all other packets.

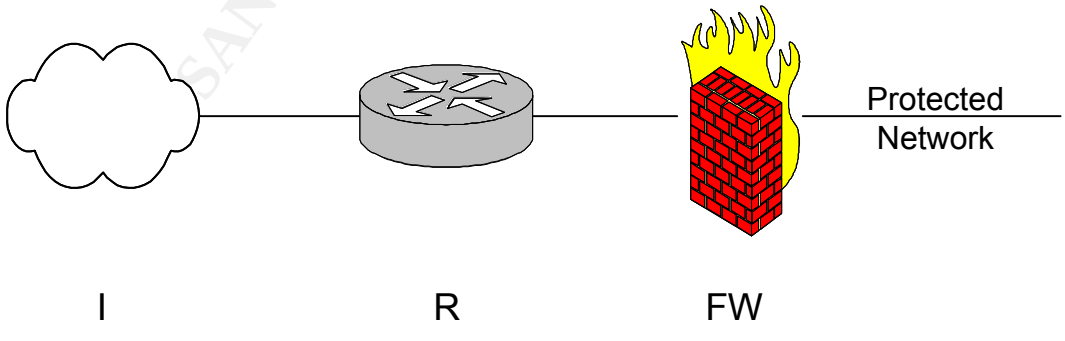
Although this example is extremely simplified, it illustrates a couple of points. First, for a packet to pass, it must be explicitly defined. Second, it is a good idea to have a "catch-all" rule which rejects all packets not explicitly allowed.

Limitations of packet filters
Packet filters do not inspect the payload of the packet. They do not read the data and make decisions based on the contents. Dangerous forms of permissible traffic may pass through the filter undetected. A virus in an e-mail attachment will pass if SMTP/POP connections are allowed.

Packet filters are not stateful. They do not remember a packet once it has passed. Conversation streams cannot be reconstructed to determine if a connection attempt is malicious. As a result, an assault based upon some packet fragmentation scheme, is difficult to prevent using a packet filter only.

Packet filtering does not deal well with the quirks of certain protocols. FTP is a good example of this. The FTP command stream is established on port 21/TCP and the data stream is on port 20/TCP. The client uses random high TCP ports. The data connection from the ftp server has a source port of 20 and tries to connect to a high destination port number. There are ways around this, but the best all-around solution seems to be an FTP proxy server.

Applications of packet filtering
A packet filtering device can be the first-line of defense in the network and used to block in-bound packets of specific types from ever reaching the protected network. This is known as *ingress filtering*. Although not a robust firewall, it can be used to reduce the load on the proxy or application firewall. The following diagram illustrates a simple example of using the packet filter and proxy or application firewall.



In this illustration, the protected network is connected to the Internet (I) via a router (R) and a firewall (FW). If, for instance, the protected network does not offer any DNS

services to the Internet, then the router can be configured to block in-bound TCP and UPD traffic to port 53. This eliminates extraneous requests for DNS servicing on a network that does not offer DNS. The firewall should still be configured to disallow ports 53/UDP and 53/TCP, but because the router is filtering, there will not be any DNS requests or zone transfer requests hitting the firewall. This same approach should be used for other protocols also. A general philosophy on ingress filtering would be to block all ports that are not being used to supply services to the Internet.

The flip side of ingress filtering is *egress* filtering. Just as ingress filtering blocks in-bound traffic, egress filtering filters traffic leaving the network. Egress filtering might be used, for instance, to limit the connections that may be established by a mail server on the protected network to hosts on the Internet. In this case, connections might be restricted to port 25/TCP. Another use of egress filtering might be to limit users on the protected network so they can only access http (port 80/TCP) on the Internet.

One of the most important uses of ingress and egress filtering is in combating denial of service attacks. RFC2827 specifies best current practices for ingress filtering with respect to denial of service attacks that employ IP source address spoofing. ICSA and The SANS Institute have also issued recommendations for ingress and egress filtering to fight DoS attacks. The following guidelines should be adhered to:

Ingress – Inbound filtering

1. Block packets destined for services that are not being offered to the Internet.
2. Block addresses that have a source IP address of:
    a. Illegal addresses – e.g. 0.0.0.0/8 (CIDR notation)
    b. Broadcast address – 255.255.255.255/32
    c. RFC1918 reserved addresses – Private networks use these. There should not be any traffic attempting to access your network with these as source addresses. The IANA reserved blocks are:
        10.0.0.0 - 10.255.255.255
        172.16.0.0 - 172.31.255.255
        192.168.0.0 - 192.168.255.255
    d. Multicast, if multicast is not being used.
    e. Loopback – 127.0.0.0/8
    f. ICMP broadcast per RFC 2644 – This will keep a site from being used as a **smurf amplifier**.
    g. UDP echo
3. Block packets from outside the filtering device with a source IP address the same as your internal networks. This blocks packets with spoofed IP addresses. This means that you must know what address space is used internally.

Egress – Outbound filtering

1. Block traffic with an invalid source IP address. This keeps a denial of service attack using IP address spoofing from originating on the internal network. The filter should only allow traffic to leave your network with a source IP address

that is valid on your internal networks. The purpose here is to keep a denial of service attack from originating on the private network.

Summary

This article has addressed packet filtering.

1. We saw that packet filtering is a means to impose control on the types of traffic permitted to pass from one IP network to another.
2. We noted that there are good reasons to use a packet filter.
3. The packet filter examines the header of the packet and makes a determination of whether to pass or reject the packet based upon the contents of the header.
4. Packet filters have limitations.
5. Finally, applications of packet filters were discussed and some guidelines for combating denial of service attacks were set forth.

Bibliography

Boran, Sean. "IT Security Cookbook." 19 Sept. 2000. URL: http://www.boran.com/security/ (21 Sept. 2000).

Carnahan, Lisa J. and Wack, John P. "Keeping Your Site Comfortably Secure: An Introduction to Internet Firewalls." 9 February 1995. URL: http://csrc.ncsl.nist.gov/nistpubs/800-10/ (21 Sept. 2000).

CERT Coordination Center. "Packet Filtering for Firewall Systems." 12 February 1999. URL: http://www.cert.org/tech_tips/packet_filtering.html (19 Aug 2000).

Chapman, D. Brent. "Network (In)Security Through IP Packet Filtering." 1992. URL: http://www.greatcircle.com/pkt_filtering.html (19 Aug 2000).

Ferguson, P. and Senie, D. "RFC 2827: Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing." May 2000. URL: http://www.faqs.org/rfcs/rfc2827.html (21 Sept 2000).

McCown, J.D. "Applying Filtering Router Technology to DdoS Mitigation." URL: http://www.icsa.net/html/communities/ddos/alliance/guide.shtml (21 Sept 2000).

Russell, Rusty. "Linux IPCHAINS-HOWTO." V1.0.8. 4, July, 2000. URL: http://www.linuxdoc.org/HOWTO/IPCHAINS-HOWTO.html (21 Sept. 2000).

SANS Institute. "Help Defeat Denial of Service Attacks: Step-by Step." V 1.41, 23 March 2000. URL: http://www.sans.org/dosstep/index.htm (21 Sept 2000).

SANS Institute. "How To Eliminate The Ten Most Critical Internet Security Threats." V1.27. September 2000. URL: http://www.sans.org/topten.htm (21 Sept 2000).

[i] http://webopedia.internet.com/TERM/p/packet_filtering.html

Last Updated: June 26th, 2017

Upcoming Training

SANS London July 2017	London, United Kingdom	Jul 03, 2017 - Jul 08, 2017	Live Event
Cyber Defence Japan 2017	Tokyo, Japan	Jul 05, 2017 - Jul 15, 2017	Live Event
SANS Munich Summer 2017	Munich, Germany	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Cyber Defence Singapore 2017	Singapore, Singapore	Jul 10, 2017 - Jul 15, 2017	Live Event
SANS Los Angeles - Long Beach 2017	Long Beach, CA	Jul 10, 2017 - Jul 15, 2017	Live Event
Community SANS Phoenix SEC401	Phoenix, AZ	Jul 10, 2017 - Jul 15, 2017	Community SANS
Mentor Session - SEC401	Ventura, CA	Jul 12, 2017 - Sep 13, 2017	Mentor
Mentor Session - SEC401	Macon, GA	Jul 12, 2017 - Aug 23, 2017	Mentor
Community SANS Atlanta SEC401	Atlanta, GA	Jul 17, 2017 - Jul 22, 2017	Community SANS
SANSFIRE 2017	Washington, DC	Jul 22, 2017 - Jul 29, 2017	Live Event
SANSFIRE 2017 - SEC401: Security Essentials Bootcamp Style	Washington, DC	Jul 24, 2017 - Jul 29, 2017	vLive
Community SANS Fort Lauderdale SEC401	Fort Lauderdale, FL	Jul 31, 2017 - Aug 05, 2017	Community SANS
SANS San Antonio 2017	San Antonio, TX	Aug 06, 2017 - Aug 11, 2017	Live Event
SANS Boston 2017	Boston, MA	Aug 07, 2017 - Aug 12, 2017	Live Event
SANS Prague 2017	Prague, Czech Republic	Aug 07, 2017 - Aug 12, 2017	Live Event
Community SANS Omaha SEC401*	Omaha, NE	Aug 14, 2017 - Aug 19, 2017	Community SANS
SANS New York City 2017	New York City, NY	Aug 14, 2017 - Aug 19, 2017	Live Event
SANS Salt Lake City 2017	Salt Lake City, UT	Aug 14, 2017 - Aug 19, 2017	Live Event
Community SANS Trenton SEC401	Trenton, NJ	Aug 21, 2017 - Aug 26, 2017	Community SANS
Community SANS San Diego SEC401	San Diego, CA	Aug 21, 2017 - Aug 26, 2017	Community SANS
Virginia Beach 2017 - SEC401: Security Essentials Bootcamp Style	Virginia Beach, VA	Aug 21, 2017 - Aug 26, 2017	vLive
SANS Virginia Beach 2017	Virginia Beach, VA	Aug 21, 2017 - Sep 01, 2017	Live Event
SANS Adelaide 2017	Adelaide, Australia	Aug 21, 2017 - Aug 26, 2017	Live Event
SANS Chicago 2017	Chicago, IL	Aug 21, 2017 - Aug 26, 2017	Live Event
Mentor Session - SEC401	Minneapolis, MN	Aug 29, 2017 - Oct 10, 2017	Mentor
SANS Tampa - Clearwater 2017	Clearwater, FL	Sep 05, 2017 - Sep 10, 2017	Live Event
SANS San Francisco Fall 2017	San Francisco, CA	Sep 05, 2017 - Sep 10, 2017	Live Event
Mentor Session - SEC401	Edmonton, AB	Sep 06, 2017 - Oct 18, 2017	Mentor
SANS Network Security 2017	Las Vegas, NV	Sep 10, 2017 - Sep 17, 2017	Live Event
Community SANS Albany SEC401	Albany, NY	Sep 11, 2017 - Sep 16, 2017	Community SANS
Community SANS Dallas SEC401	Dallas, TX	Sep 18, 2017 - Sep 23, 2017	Community SANS