1 Introduction to Firewalls

Steve Woodall [1] has written an excellent project report on firewall design principles. I'd suggest you all refer to Steve's article for a nice introduction to the topic and the references therein. I will however supplement Steve's material on how one goes about designing a packet filter; as we know filtering is done on the basis of one or more of the following:

1. Source IP address.
2. Destination IP address.
3. Source port no.
4. Destination port no.
5. ACK bit set or not.
6. The interface on which the packet arrives/leaves.

The following section on a sample firewall illustrates the fact that designing a packet filter from first principles is not always easy, and there is always potential attackers can exploit to gain access to your private network.

2 Example of a sample Firewall

Most of the material in this section is taken from Chapter 8 of Zwicky et al. [2]. Let us start with a sample example that illustrates some of the important considerations in firewall design. Let us say that McMaster University wants to allow inbound and outbound SMTP (so that people in Mac can send and receive electronic mail) and nothing else. Kartik is assigned the task of coming with a packet filter and he comes up with the following: A few points regarding Table 1 are in order:

- Rules A and B allow inbound SMTP connections (incoming email). This is used by the people in Mac to receive email from others.
- Rules C and D allow outbound SMTP connections (outgoing email). This is used by the people in Mac to send out email.

Rule	Direction	Source Address	Dest. Address	Protocol	Dest. Port	Action
A	In	External	Internal	TCP	25	Permit
B	Out	Internal	External	TCP	> 1023	Permit
C	Out	Internal	External	TCP	25	Permit
D	In	External	Internal	TCP	> 1023	Permit
E	Either	Any	Any	Any	Any	Deny

Table 1: Packet Filter 1

- Rule E is the default rule that applies if all else fails.

We will assume in this example that the packet filter examines the rules in order, i.e., it starts at the top until it finds a rule that matches the packet, and then in takes the action specified.

Now let us consider some sample packets to see how Kartik's filter works. Let's say that your host has IP address 172.16.1.1,and that someone is trying to send you mail from a remote host with IP address 192.168.3.4. Furthermore, let's say the sender's SMTP client uses port 1234 to talk to your SMTP server, which listens at port 25 for incoming email. The TCP connection has the following characteristics filter and is allowed due to rules A and B in the packet filter. What about your outgoing email from you to them?. Let us say that our

Rule	Direction	Source Address	Dest. Address	Protocol	Dest. Port	Action
1	In	192.168.3.4	172.16.1.1	TCP	25	Permit (A)
2	Out	172.16.1.1	192.168.3.4	TCP	1234	Permit (B)

SMTP client uses port 1357 to talk to their SMTP server; this TCP connection is allowed too (see the table below) due to the rules A and B. Now suppose someone

Rule	Direction	Source Address	Dest. Address	Protocol	Dest. Port	Action
3	Out	172.16.1.1	192.168.3.4	TCP	25	Permit (C)
4	In	192.168.3.4	172.16.1.1	TCP	1357	Permit (D)

from the outside world (say host 192.168.3.4) attempts to open a connection from port 5150 on his end to the web proxy server on port 8080 on machine 172.168.1.1 It turns out that this TCP connection is allowed too (see table below) due to the rules D and B. Here rules B and D together end up allowing all connections where both ends are using ports above 1023. This was not what McMaster initially intended. Kartik is reprimanded, and given a chance to set things right!. Kartik decides to include the source pot in making the filtering decisions too. So he comes up the following packet filter (see Table 2; compare with Table 1). In this case packets 5 and 6 which got through are rejected due to the default rule, as can be seen from the following table. Things are looking pretty rosy for Kartik until a smart attacker strikes!. Let's say this attacker uses port 25 as the client port on his end (he does this by killing the SMTP server normally listening at this port and uses this port for is client needs). He then attempts to open a connection to the web proxy server (listening at port 8080) on machine 172.16.1.1. As you can see from the following table, this TCP connection gets through (creating nightmares for Kartik :(). So what can Kartik do?. After reading Chapter 15 of Comer on TCP connections, Kartik realizes that the ACK (acknowledgement) bit could also be used as a filtering criterion.

Rule	Direction	Source Address	Dest. Address	Protocol	Dest. Port	Action
5	In	192.168.3.4	172.16.1.1	TCP	8080	Permit (D)
6	Out	172.16.1.1	192.168.3.4	TCP	5150	Permit (B)

Rule	Direction	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	Action
A	In	External	Internal	TCP	> 1023	25	Permit
B	Out	Internal	External	TCP	25	> 1023	Permit
C	Out	Internal	External	TCP	> 1023	25	Permit
D	In	External	Internal	TCP	25	> 1023	Permit
E	Either	Any	Any	Any	Any	Any	Deny

Table 2: Packet Filter 2

So, Kartik generates the following packet filter (see Table 3; also compare with Tables 1 and 2). This prevents packet 7 (the attacker attempting to open a connection to your web proxy server), as follows. Remember the ACK bit is not set in packet 7 as it opens the TCP connection (the first handshake).

Now what if our attacker get around this by simply setting the ACK bit on the first packet?. In this case, packet 7 will get past the filters, but the destination will believe the packet belongs to an existing connection (the packet is instead trying to open a new connection here). When the destination tries to match the packet with the supposed existing connection (using the sequence nos in the packet), it will fail because there isn't one in the first place, and the packet will be rejected.

Ending: Kartik gets that raise from McMaster University that he has always wanted, and peace reigns on planet Mac :)

Supplementary Reading:

References

[1] S. Woodall, *Firewall design principles*, Project Report, 4C03, 2004. http://optlab.cas.mcmaster.ca/~kartik/sfwr4c03/projects/SteWoodall-Project.pdf

[2] E.D. Zwicky, S. Cooper, and D.B. Chapman, *Building Internet Firewalls*, 2nd edition, O'Reilly & Associates, 2000.

Rule	Direction	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	Action
5	In	192.168.3.4	172.16.1.1	TCP	5150	8080	Deny (E)
6	Out	172.16.1.1	192.168.3.4	TCP	8080	5150	Deny (E)

Rule	Direction	Source Address	Dest. Address	Protocol	Source Port	Dest. Port	Action
7	In	192.168.3.4	172.16.1.1	TCP	25	8080	Permit (D)
8	Out	172.16.1.1	192.168.3.4	TCP	8080	25	Permit (C)

Rule	Dir.	Source Add.	Dest. Add.	Prot.	Source Port	Dest. Port	ACK set	Action
A	In	External	Internal	TCP	> 1023	25	Any	Permit
B	Out	Internal	External	TCP	25	> 1023	Yes	Permit
C	Out	Internal	External	TCP	> 1023	25	Any	Permit
D	In	External	Internal	TCP	25	> 1023	Yes	Permit
E	Either	Any	Any	Any	Any	Any	Any	Deny

Table 3: Packet Filter 3

Rule	Dir.	Source Add.	Dest. Add.	Prot.	Source Port	Dest. Port	ACK set	Action
7	In	192.168.3.4	172.16.1.1	TCP	25	8080	No	Permit (D)