Kapitel 7

Brandväggar

7.1 Allmänt om brandvägg

En brandväggs uppgift är att upprätthålla din säkerhetspolicy. Detta genom att släppa igenom viss trafik och spärra annan, beroende på saker som destinationsport, avsändaradress, tcp-flaggor (SYN, ACK, FIN osv). En brandvägg bör sällan vara enda skyddet. Ifall du vill köra servertjänster som webbserver eller mailserver måste dessa givetvis ha öppna portar (t.ex. http:80, smtp:25) vilka brandväggen ska släppa igenom, normalt sett för alla IP om det ska vara publika tjänster. Då bör man se till att hålla serverprogrammen uppdaterade med de senaste säkerhetsfixarna. Å andra sidan, om man på en Linux- eller UNIX-burk inte kör några tjänster kan man klara sig bra utan brandvägg, lyssnar inte datorn på någon port kan den svårligen göras intrång på utifrån.

Normalt har man dock nytta av brandvägg; man kör kanske tjänster som bara ska vara lokalt åtkomliga, eller har flera datorer (kanske med osäkra operativsystem) som ska dela på en internetförbindelse. En brandvägg kan även skydda mot rena misstag, som att root startar någon osäker demon (medvetet eller omedvetet). Man måste tänka på att säkerhet är en mångfasetterad och ständigt pågående process, inte bara ett program man installerar och sen glömmer bort eller en liten låda med blinkande lysdioder.

7.2 Brandväggen som följer med

Vill du köra den brandvägg som är standard i Red Hat så kan du konfigurera den med `sudo system-config-firewall-tui` (eller `system-config-firewall` i X). Deras brandvägg räcker bra för en server med till exempel enbart SSH och webbserver, även för enkel NAT. Men vill du ha mer avancerade funktioner bör du lära dig att skriva egna regler.

7.3 Netfilter/iptables

I Linux heter kärnans del av brandväggen egentligen Netfilter, men oftast kallar man alltihop iptables, efter vad kommandot för att ange reglerna heter.
Iptables har stöd både för tillståndslös- (stateless) och tillståndsfiltrering (stateful inspection). Varje kedja består av ett antal regler. Standardtabellen filter har tre kedjor: INPUT, OUTPUT och FORWARD. INPUT är trafik in till processer på den lokala maskinen, OUTPUT är kedjan för trafik genererad lokalt. Kedjan FORWARD är för paket som går in på ett interface och ut på ett annat, dvs när datorn fungerar som en router.

Med `iptables -P INPUT DROP` anger man att standardpolicyn för tabellen INPUT ska vara att tyst droppa paketen (utan att skicka tcp-RST eller något icmp-meddelande) Ett exempel på syntaxen för en regel är:

```
iptables -A INPUT -p TCP -i eth0 -s 192.168.44.20
——dport 22 -j ACCEPT
```

vilket tillåter inkommande SSH från ett specifikt IP-nummer och endast på ett visst nätverkskort (eth0).

En regel i iptables har oftast två delar: något man kollar och något man gör om det matchar. Som regeln ovan matchar man efter protokoll, interface, källadress och destinationsport. Om alla dessa matchar hoppar man till målet (-j target) ACCEPT och inga flera regler kontrolleras. Skulle paketet kommit från ett annat IP skulle man gått vidare till nästa rad osv tills någon matchar (first-match-wins), och om ingen regel stämmer in blir det standardpolicyn som gäller (DROP ifall man kör med deny-all).

7.4 State/tillstånd

Begreppet state översätter vi med tillstånd. Det gäller främst TCP som i sig är ett förbindelseorienterat protokoll (stateful, connection-oriented), men även viss UDP kan av iptables tolkas som att ha tillstånd (även om det egentligen är separata paket). Vid anslutning med TCP skickar klienten först ett paket med flaggan SYN, servern svarar med SYN och ACK med nästa byte den förväntar sig, klienten svarar med en ACK av detta paket. Därmed är trevägshandskakningen avklarad och förbindelsen upprättad, vilket ses som "established" i netstat. Även iptables använder ordet ESTABLISHED för detta läge.

Tillståndet där förbindelsen håller på att upprättas, SYN mottagen men SYN, ACK ej sänt, kallas av iptables för NEW. Det är det man ofta vill stoppa eller tillåta beroende på port, IP, interface eller liknande. Det finns även ett tillstånd RELATED, det är mer komplicerat och används för att tillåta protokoll som öppnar en ny förbindelse, t.ex. FTP i port mode. Även ett svar på utgående ping (ICMP echo request) tolkar den som ett relaterat paket.

Man använder tillståndsinspektion genom att använda -m state, ofta har man en gemensam regel:

```
iptables -A FORWARD -m state
——state ESTABLISHED,RELATED -j ACCEPT
```

och en rad för varje port/tjänst man vill tillåta, t.ex. om man har en webbserver innanför brandväggen:

```
iptables -A FORWARD -i eth0 -o eth1 -p tcp
——dport 80 -m state ——state NEW -j ACCEPT
```

(ifall det inte är publik adress innanför krävs även DNAT för detta, mera nedan)

7.5 NAT

NAT betyder Network Address Translation. Det finns både SNAT och DNAT. Source-NAT (SNAT) är den vanligaste. Ifall en intern dator skickar paket ut mot Internet skriver brandväggen om källadressen i IP-huvudet så paketet ser ut att komma från brandväggens publika IP-nummer och en ledig port på denna. Mappningen mellan internt IP/port och ex-

ternt IP/port sparar kärnan, så när det kommer ett svar från t.ex. en webbserver så skickar iptables tillbaka till rätt intern dator på rätt port så klientprogrammet inte märker att adressen har översatts. Exempel på detta är:

```
iptables -A FORWARD -m state ——state NEW -i ! eth0
-j ACCEPT
```

Denna regel i FORWARD-kedjan tillåter nya anslutningar inifrån men inte från det yttre interfacet. Sen

```
iptables -t nat -A POSTROUTING -o eth0
-s 192.168.44.0/24 -j SNAT ——to $MYIP
```

där man använder tabellen nat och kedjan POSTROUTING på det utgående interfacet eth0 för alla källadresser på det privata (RFC1918) nätet. MYIP är en variabel för brandväggens externa IP-nummer, den lägger man till i början av skriptet, t.ex:

```
MYIP=194.47.250.234
```

NAT kräver att man har raden med `——state ESTABLISHED,RELATED` i FORWARD-kedjan, eftersom Netfilter måste hålla reda på vilka tillstånd de olika NAT:ade förbindelserna är i. Det krävs även att ip-forwarding är påslagen i kärnan, det gör man via:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

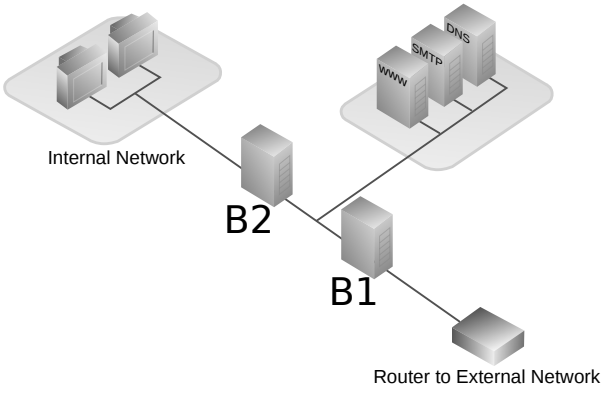
DNAT, destination NAT, används för att ge åtkomst till interna servrar utifrån, t.ex. om man har angett raden med `——dport 80` ovan och vill ha DNAT till en webbserver på 192.168.44.30 skriver man:

```
iptables -A PREROUTING -t nat -d $MYIP -p tcp
——dport 80 -j DNAT ——to 192.168.44.30:80
```

Detta gör att trafik in till port 80 på brandväggen skrivs om så den kan adresseras på det interna LAN:et.

7.6 DMZ

På företag vill man ofta separera publika servrar från resten av nätverket. Då kan man sätta upp ett DMZ, en “demilitarized zone”, eller perimeternätverk. Detta kan man göra genom att ha två brandväggar. Sett utifrån först en brandvägg B1 med både SNAT och DNAT, med ”port forwarding” via DNAT till en eller flera servrar i DMZ. Dessa servrar har B1 som default gateway. På detta LAN sitter även ena nätverkskortet på brandvägg B2, som har sitt andra interface på det LAN där användarnas arbetsstationer sitter. B2 kör endast SNAT. Detta gör att om någon kan göra intrång på en server och på så vis passera B1 kommer de inte vidare genom B2. Servrar som bara används internt kan placeras mellan B1 och B2 (i DMZ) ifall dess tjänster behöver kommas åt från de andra servrarna där, eller innanför B2 om de bara behöver kommas åt från arbetsstationerna.



Figur 7.1: *DMZ*

7.7 Loggning

För att logga paket finns två alternativa targets, LOG och ULOG. De används som andra targets, men de avbryter inte kedjan utan bara fortsätter till raden nedanför. Vill man bara logga allt som passerar till kedjans policy avslutar man med:

```
iptables -A INPUT -j LOG
```

Vill man logga allt som stannar vid en viss regel skriver man samma matchning med target LOG på raden ovanför. T.ex:

```
iptables -A INPUT -p tcp -s 207.46.197.32
——dport 22 -j LOG
iptables -A INPUT -p tcp -s 207.46.197.32
——dport 22 -j DROP
```

Om man både vill logga och spärra SSH-anslutning från den IP-adressen. Allt med target LOG skickas vidare till vanliga syslog, med facility kernel och priority warning. Detta betyder med andra ord att det skrivs till /var/log/messages eller motsvarande fil. Det kan bli en hel del rader i loggarna.

Target ULOG skickar istället till en separat process, ulogd, skild från systemets vanliga loggning. Ofta struntar man helt i loggning, det som stoppas kommer ju inte in ändå, och logga allt som passerar skulle bli för mycket. Det kanske inte är så intressant att se hur ofta man portscannas. Men som felsökning kan en rad med `-j LOG` vara mycket värdefull.

7.8 Diverse tips

Vill man matcha mer än en port finns det två sätt:

```
iptables -A INPUT -p tcp ——dport 6000:6063 -j DROP
```

om de är i följd, eller:

```
iptables -A INPUT -p tcp -m multiport
——dport 80,443 -j ACCEPT
```

för portar som inte är i följd.

För att ta bort en regel använder man `-D` istället för `-A`. `-A` lägger alltid till i slutet av kedjan (append), det är oftast det man vill. Det finns `-I` för att lägga till först och `-R` för replace, men skriver man alla reglerna i ett skript blir det mest lättläst med `-A` hela tiden.

Man bör tänka på att DNS oftast använder UDP, men byter till TCP vid långa svar:

```
iptables -A INPUT -p UDP ——dport 53 -j ACCEPT
iptables -A INPUT -p TCP ——dport 53 -j ACCEPT
```

Ifall man vill använda hostname i sina regler måste man tillåta DNS-trafik innan detta, men att använda IP-nummer eller nät med CIDR-notation rekommenderas, då DNS kan gå att lura i vissa fall.

Även om man kanske normalt sett tillåter allt annat ut genom brandväggen bör man stoppa SMB-fildelning:

```
iptables -A FORWARD -o $OD -p tcp -m multiport
−−dport 139,445 -j REJECT
iptables -A FORWARD -o $OD -p udp -m multiport
−−dport 137,138 -j REJECT
```

(där $OD är utgående interfacet) Använder man `-j REJECT` skickas icmp-port-unreachable, istället för att tyst slänga paketet som `-j DROP` gör. Detta gör att klienten slipper vänta på att TCP ska tajma ut. Det brukar rekommenderas att använda DROP mot Internet och REJECT mot sitt LAN. Det tar mycket längre tid att portscanna vid DROP.

Kärnan kan hjälpa till med en hel del annat än Netfilter, det finns flera parametrar till TCP/IP-stacken som man bör sätta i sitt brandväggsskript

Viss anti-ipspoofing: `echo 1 > /proc/sys/net/ipv4/conf/all/rp_filt`
Slå på syn-cookies (skydd mot syn-flooding attacker): `echo 1 >/proc/sys/net/ipv4/tcp_syncookies`
Stänga av ICMP echo-request till broadcast adresser (Smurf amplifier): `echo 1 >/proc/sys/net/ipv4/icmp_echo_ignore_broadcas`
Stänga av ICMP redirects
`echo 0 >/proc/sys/net/ipv4/conf/all/accept_redirects`
Stänga av source route: `echo 0 >/proc/sys/net/ipv4/conf/all/accep`
IP Bogus Error Response Protection: `echo 0 > /proc/sys/net/ipv4/icmp_i`

Vissa, men inte alla, av dessa har rätt värde som standard, men det skadar inte att slå på och av dem i skriptet.

Man bör tänka sig för om man är fjärrinloggad via SSH och uppdaterar brandväggens regler. T.ex. bör man sätta `-P INPUT ACCEPT` innan man gör en `-F INPUT`

`-F` är flush, ta bort alla regler i en kedja. Det är inte lämpligt om policyn är DROP. Jag har missat det nån gång och fått besöka datorn eller ringa kollega.

Man kan även använda cron eller at för att automatiskt återställa tidigare inställningar vid fjärradministration av `iptables`.

7.9 Lästips

- S. Suehring & R.L. Ziegler. *Linux Firewalls.* Pearson Education 2006.
- G.N. Purdy. *Linux iptables Pocket Reference.* O'Reilly 2004.
- Oskar Andreasson. *Iptables Tutorial* http://www.frozentux.net/documents/iptables-tutorial/