Brandväggar

Grupp 13
Andreas Önnebring
Markus Månsson

21 februari 2008

Brandväggar

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 kan 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.

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 ju inte minst även skydda mot rena misstag, som att root råkar starta någon osäker daemon (medvetet eller omedvetet). Man måste tänka på att säkerhet är en mångfacetterad och ständigt pågående process, inte bara ett program man installerar och sen glömmer bort eller en liten låda med blinkande lysdioder.

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 pakten (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 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 till någon matchar (first-match-wins), och om ingen regel stämmer in blir det standardpolicyn som gäller (DROP ifall man kör med deny-all).

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)

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 och en ledig port på denna. Mappningen mellan internt ip/port och externt 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 för ett C-nät är:
`iptables -A FORWARD -m state --state NEW -i ! eth0 -j ACCEPT`
Först en regel i FORWARD-kedjan som 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.

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

DMZ

På företag vill man ofta separera 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 portforwarding 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 arbetstationer sitter. B2 kör endast SNAT. Detta gör att om någon kan göra intrång på någon 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.

Loggning

För att logga paket finns två alternativa targets, LOG och ULOG. De används som andra targets, fast de avbryter inte kedjan utan returnerar 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 på de flesta Linux-distributioner om man inte ändrar. 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.

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 Microsoft 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 tima 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_filter`
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_broadcasts`
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/accept_source_route`
IP Bogus Error Response Protection:
`echo 0 > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses`
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 sitter remote och uppdaterar brandväggens regler. T.ex. bör man sätta -P ACCEPT innan man gör en -F INPUT (-F är flush, ta bort alla regler i en kedja) (författaren har missat det nån gång...)

Brandväggar i Windows XP

Bakgrund

Windows har länge varit det vanligaste operativsystemet för PC-datorer och har därför varit det främsta målet för attacker över internet. Under mitten av 2003 började nätmasken Blaster härja, vilken under loppet av några minuter infekterade Windows-maskiner utan brandväggar. Senare under året började även nätmasken Sasser florera på nätet, vilken utnyttjade snarlika säkerhetsbrister. Microsoft hade redan fått hård kritik över att deras operativsystem hade ett bristfälligt skydd mot skadlig mjukvara och attacker över internet, och dessa händelser gjorde problemen ännu mer uppenbara.

Under augusti 2004 släppte Microsoft SP2 (Service Pack 2) för Windows XP, vilken förutom alla buggfixar även inkluderade Windows Firewall. Microsoft hade visserligen redan tidigare inkluderat minimala brandväggsfunktioner i Windows, men de flesta användare visste inte att de fanns, och övriga visste knappt var man skulle hitta inställningarna. Microsoft såg det som nödvändigt att göra funktionaliteten mer tydlig.

Funktionalitet

En brandvägg blir inte bättre än den som sätter firewall-reglerna gör den, och majoriteten av användarna utav Windows har väldigt grundläggande eller saknar helt uppfattning om hur en brandvägg skall konfigureras. Microsoft försåg därför Windows Firewall med väldigt minmala inställningsmöjligheter.

Brandväggen ger användaren möjlighet att konfigurera vilken form av inkommande datatrafik som skall tillåtas, men saknar helt funktioner för att kontrollera utgående datatrafik. Brandväggen tillåter i sin grundinställning de vanligaste protokollen, bland annat HTTP, ICMP och DHCP, att köra obehindrat på datorn. I övrigt används en policy om att alla andra program och portar skall nekas såvida inte användaren godkänner datatrafiken. Detta görs genom en "inlärningsfunktion" i Windows Firewall, där så fort ett nytt program eller port tar emot en viss form av trafik, frågas användaren om den vill ta emot datatrafiken samt om den vill tillåtas i fortsättningen. Frågorna är relativt generella för att inte överösa oerfarna användare med frågor om brandväggsinställningar som de inte har någon kunskap om. Den mer erfarna användaren kommer snabbt upptäcka att dessa inställningsmöjligheter inte tillhandahåller några avancerade funktioner.

Reglerna över vilken trafik som tillåts i Windows Firewall sätts antingen genom att knyta samman ett program tillsammans med vilken IP-Range den skall tillåtas ta emot datatrafik från, alternativt att knyta ihop en port med TCP- eller UDP-protokoll samt en IP-Range den skall ta emot denna för. Den senare är således en global regel som gäller för alla program på datorn. Det går att avaktivera alla ICMP-funktioner för att göra datorn "osynlig" på nätverket, men om Skrivar- och Fildelningsfunktionen är aktiverad låser Windows Firewall sig på att tillåta svar på ping-anrop.

Brandväggar från tredje part

För de som vill ha mer avancerade funktioner krävs programvara från tredje part. Dessa ersätter den inbyggda brandväggen i Windows. De exakta funktionaliteterna kan dock variera en aning mellan programmen och kan även ha begränsningar på hur många attribut man kan knyta ihop i en och samma regel.

De flesta programmen tillåter att man kan knyta ihop regler som inkluderar port-range (source och destination), ip-range (source och destination), protokolltyp samt vilken EXE-fil som regeln gäller för på den lokala maskinen. Brandväggsprogrammet ser sedan till att alla trådar som hör till en viss EXE-fil följer förälderns definierade regler. Det finns även de programvarianter som har extra funktionalitet för att knyta ihop regler på specifika MAC-adresser för source och destination.

En del program håller även en checksumma för varje EXE-fil reglerna är knutna till, där syftet är att meddela användaren om att innehållet i EXE-filen har bytts ut av någon form av virus. Detta kan givetvis även ske under normala omständigheter, t ex vid en uppdatering av program som Mozilla Firefox eller Acrobat Reader.

Varje EXE-fil kan ha flera regler knutna till sig och komplexiteten kan således bli väldigt stor i en mjukvarubrandvägg. Är dessutom användaren inte helt säker på vad han/hon gör kan det få oönskade konsekvenser, och även göra att användaren känner en falsk trygghet. Användaren borde sträva efter att ha så få regler som möjligt så att man behåller en acceptabel överskådlighet.

Referenser

[1] S. Suehring & R.L. Ziegler. *Linux Firewalls.* Pearson Education 2006.

[2] G.N. Purdy. *Linux iptables Pocket Reference.* O'Reilly 2004.

[3] Oskar Andreasson. *Iptables Tutorial 1.2.2*
http://iptables-tutorial.frozentux.net/