IPTABLES
från grunden

Anders Sikvall, Sommarhack 2015

Linux brandvägg

- Brandväggen kallas netfilter
- Den är inbyggd och har funnits länge i kernel
- Alltså inte en fristående applikation
- Den manipuleras med
  - ipfwadm
  - ipchains
  - iptables
  - ufw

ipfwadm

- Äldsta verktyget för att styra netfilter
- Inte stateful - kan inte hantera
  - Paket som är relaterade andra förbindelser
  - Inte lätt med forwarding och routing
  - Avancerad firewalling svårare
- Ganska begränsad
- Ger ändå hyggligt skydd för de flesta

ipchains

- Tog över från ipfwadm när den ej räckte
- Inte heller stateful
- Kan göra avancerade routes
- Har accounting och QoS-möjligheter
- Loggning till syslog
- Klarar NAT och MASQUERADE

iptables

- Nya delar i netfilter i 2.4 kernel krävde bättre verktyg
- Allt som ipchains har men mycket mer
- Kan fatta beslut baserat på saker som antal paket per minut
- Kan sätta upp komplicerade kedjor av regler med hopp baserat på tidigare händelser
- Effektiv pre-routing och post-routing
- Kan tagga paket med särskilda tags för att senare relatera dem till anda paket
- Massor med MODES

ufw

- Står för “uncomplicated firewall”
- Mycket enkel att använda
- Ger effektivt skydd för de flesta hempulare
- Svårare att göra avancerade lösningar
- “Leksak” säger de som borde veta

Iptables behöver inte vara svårt

- **Tre steg:**
  - Sätt upp en policy för vad som händer om ingen regel matchar paketen
  - Sätt upp regler för att blockera eller tillåta paket som matchar reglerna
  - Logga det som du behöver logga för att ha koll
- **Därefter kan vi börja labba med mer avancerade saker**
  - Routing
  - NAT
  - Masquerade

Kedjor och tabeller

- Iptables har kedjor som innehåller regeltabeller
- Tre inbyggda sådana finns ALLTID
  - INPUT - paket med denna dator som destination
  - OUTPUT - paket med denna dator som källa
  - FORWARD - paket som routas genom denna dator
- Egna kedjor skapas efter ändamål
- När en kedja är skapad kan man fylla den med regler…
- …eller tabellen om man så vill

Observera!

En vanlig missuppfattning är att paket går igenom mer än en kedja. Detta är inte sant om man inte gör -j (jump).

Paket med destination denna dator passerar INPUT

Paket som härrör från denna dator men har destination till en annan dator passerar OUTPUT.

Paket som routas genom denna dator passerar varken input eller outpu utan endast FORWARD precis som svaren på dessa paket.

Även vana användare glömmer bort detta ibland.

POLICY

Tre sorters policy finns:

DROP - kastar paket som ej matchar någon regel

ACCEPT - accepterar alla paket som inte matchar

DENY - svarar med felmeddelande (via ICMP)

En policy appliceras när ingen annan regel matchar paketet.

POLICY

- Skillnaden mellan DROP och DENY är att den som skickar ett paket får ett “connection denied” om man får DENY medan om man får ett DROP får man inget svar alls.
- Till slut timar TCP ut om man kör det
- UDP bara “kom inte fram”
- Connection deny meddelas med ICMP

Skapa din första brandvägg

Rensa alla gamla regler och öppna allt:

```
# iptables -P INPUT    ACCEPT
# iptables -P OUTPUT      ACCEPT
# iptables -P FORWARD  ACCEPT
# iptables -F    ; Rensa alla regler
# iptables -X    ; Radera alla user chains
```

Skapa din första brandvägg

Nu kan vi börja med att sätta en default policy.

```
# iptables -P INPUT   DENY
# iptables -P OUTPUT  DENY
# iptables -P FORWARD DENY
```

Obs, om du gör detta på en host över SSH tappar du förbindelsen. Detta stoppar all nätverkstrafik.

Regler för tillåtna tjänster

```
# iptables -A INPUT  -p icmp -j ACCEPT
# iptables -A OUTPUT -p icmp -j ACCEPT
```

Ovanstående accepterar all ICMP-trafik in och ut ur burken. Många stänger av ICMP av rädsla att den kan vara farligt.

Det är det som regel inte.

Internet är trasigt utan ICMP så var snäll och tillåt det.

Tillåt trafik till och från webserver

Öppna trafiken till och från webservern

```
# iptables -A INPUT  -p tcp --dport http -j ACCEPT
# iptables -A OUTPUT -p tcp --sport http -j ACCEPT
```

Nu kan man prata med webservern.

Det var väl inte så svårt?

http kan ersättas med portnummer 80 om man vill

Öppna för SSH

Så öppnar vi för SSH

```
# iptables -A INPUT  -p tcp --dport 22 -j ACCEPT
# iptables -A OUTPUT -p tcp --dport 22 -j ACCEPT
```

Sådär, nu kan vi SSH:a till maskinen och köra rsync och scp och annat nyttigt!

Kommandot förklarat

```
# iptables -A INPUT -p tcp --dport 80 -j ACCEPT
```

`-A INPUT`	Lägg till regel i input-kedjan
`-p tcp`	Protokollet måste vara TCP/IP
`--dport 80`	Endast paket med destinationsport 80
`-j ACCEPT`	Hoppa till slutet av kedjan och accept

Inga andra sorters paket matchar denna regel.

ACCEPT, DROP, DENY

- Dessa är targets, alla paket får en target.
- De som inte matchar en regel får target från POLICY
- När en regel matchar som hoppar till target slutar man processa, inga fler regler kommer synas för just detta paket.
- Reglerna tillämpas i den ordning de lagts till i kedjan
- Paket måste antingen hitta en target eller routas till en annan kedja

Aktiv blockering

Stoppa t.ex. trafik från vissa nät

```
# iptables -A INPUT -s 192.168.0.0/16 -j DROP
```

Regeln kastar alla paket med source från

192.168.0.0 - 192.168.255.255

Kan användas för att stoppa “source routing”

Blockera vissa länder

Med så kallade CIDR-listor kan vi lätt blocka oönskade länder från våra servrar. Listorna finns på sajter på nätet.

```
for f in /etc/iptables/banned-hosts/*
do
  while read p
  do
    iptables -A INPUT -s $p -j DROP
  done < $f
done
```

Logga paket

- Loggning är viktigt och iptables har det i sig
- Logga till syslog om möjligt, skapa gärna egen fil
- Du skapar enkelt egna kedjor som loggar
- Oftast skapas två stycken
  - Logga och kasta paketet
  - Logga och acceptera paketet

Logga paket

Skapa ny kedja LOGDROP

```
# iptables -N LOGDROP
```

Lägg till loggregeln

```
# iptables -A LOGDROP -j LOG --log-prefix \
    "IPT: DROP " --log-level 7
```

Därefter kastar vi paketet

```
# iptables -A LOGDROP -j DROP
```

Logga paket som är felaktiga

Paket med trasiga checksummor, signalbitar som inte får vara satta samtidigt, felaktiga längder mm

```
# iptables -A INPUT -m state --state INVALID -j LOGDROP
```

Du kan också logga alla paket som knackar på SSH

```
# iptables -A INPUT -p tcp --dport 22 -m state \
    --state NEW -j LOGACCEPT
```

Stateful

`-m state`	Använd modulen stateful inspec.
`--state x`	Vilken state är förbindelsen i

`NEW`	Nyöppnad (SYN)
`ESTABLISHED`	Pågående
`RELATED`	Relaterad annan trafik

Bättre säkerhet med stateful

```
# iptables -A INPUT -p tcp --dport 22 -m state \
    --state NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 22 -m state \
    --state ESTABLISHED -j ACCEPT
```

Vi kan alltså begränsa vilka states förbindelser får vara i om vi skall tillåta paket in eller ut. I det här fallet tillåter vi bara utgående paket som svar på SSH om det finns ett etablerat koppel in.

Detta kan göras mycket avancerat. Det är också lätt att göra fel men det är väldigt effektivt.

Logga allt som inte matchar

Ibland vill man logga allt som inte matchar

Avsluta då kedjorna INPUT, OUTPUT och FORWARD med följande regel EFTER alla andra regler:

```
# iptables -A INPUT        -j LOGDROP
# iptables -A OUTPUT    -j LOGDROP
# iptables -A FORWARD   -j LOGDROP
```

Nu spelar din policy inte så stor roll längre eftersom du explicit kastar allt som inte matchar en tidigare regel i kedjan.

Spara brandväggen

- Normalt lägger man regler i ett skript
- Det kan ta lång tid köra skriptet
- Inte effektivt under boot

Spara brandväggen

- Därför finns `iptables-save >filnamn`
- Sparar alla reglerna i en snabbläst fil
- Återställs snabbt med `iptables-restore <filnamn`

Routing

- Iptables kan så mycket mer
- Routing
- NAT
- Policy based routing
- Quality of Service
- Quenching…

Routing

Jag tänker bara gå igenom ett av många exempel på hur man routar paket

Slå på routingen i kerneln

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

Routing

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
# iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

Vi har här visat några nya targets och chains som uppstår vid behov när man använder dem men som vi inte i denna presentation kommer gå in på mer noga

Ser ni vad som händer här?

Routing

Externt nät eth0

Internt nät eth1

```
# iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

Sätter upp en forwardregel med trafik från eth0 till eth1 som är etablerad och relaterad till existerande trafik.

Trafik tillåts bara routas in om den tillhör etablerad trafik som initierats från insidan. Precis som en NAT router skall göra.

Routing

```
# iptables -t nat -A POSTROUTING -o etho -j MASQUERADE
```

-t nat Inbyggd tabell med regler för NAT

Tar hand om adressöversättningen så att det som ramlar ut från routern har en adress i det lokala nätet. Postrouting innebär att vi kör igenom den här regeln innan den hamnar i INPUT och blir föremål för några andra regler.
Masquerade är en särskild target som används vid NAT.

Routing

```
# iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

Nästa regel är enklare. All trafik från insidan tillåts ut. Kolla första raden igen:

```
# iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Trafiken från insiden maskeras med routerns adress så den ser ut som den kommer från routern. Det är det MASQUERADE gör. All trafik som lämnar på externa interfacet har routerns address och inte vår interna 192.168…whatever.

Ingen floodping här

Skydda dig mot floodpings?

```
# iptables -A INPUT -p icmp -m icmp -m limit --limit 1/second -j ACCEPT
```

En per sekund är väl ganska lagom?

Limit-modulen är sjukt användbar för sådant!

Synflood attack (DDOS)

```
# iptables -A INPUT -m state --state NEW -p tcp -m tcp --syn -m recent --name synflood --set
```

```
# iptables -A INPUT -m state --state NEW -p tcp -m tcp --syn -m recent --name synflood \ --update --seconds 1 --hitcount 60 -j DROP
```

Tagga paket först, räkna och kasta om det kommer förmånga per sekund.

Som ni ser kan man göra en del fräna saker.

Om du blockar många CIDR

- Använd i stället “ipset”, mer effektivt!
- Behöver ofta installeras separat
- Ipset kan skapa hashtabeller med många CIDR för att öka effektiviteten
- Du kan namnge tabellen med “china” eller vad du nu vill
- Därefter skjuter du in den med iptables

IPv6

Of course!

ip6tables ...

Det var en början i alla fall...

Som sagt, det var en crasch course i hur man konfar en brandvägg med iptables.

Som ni säkert förstått finns det tusen fler saker man kan göra med den.

Var inte rädd…

…men konfa inte brandväggar på remotemaskiner om du inte är HELT SÄKER på vad du gör.

Provkör hemma först.

TACK FÖR ÅHÖRANDET

Dags för en bärs