

Vergelijking tussen IPTables (netfilter) en IPF(ilter)

Professionaliseringsproject Netwerken

Benjamin Carlier
2008 - 2009

Vergelijkende studie van de pakketverwerkingssnelheid tussen Linux en FreeBSD met elk hun eigen pakketfilter: respectievelijk iptables en ipfilter.

Inhoudsopgave

Inleiding.....	3
Testomgeving.....	4
Testmethode.....	5
Doel.....	6
Benchmark.....	7
Testsessies.....	8
Configuratie	9
Brug.....	9
Brug + Firewall.....	10
Router.....	11
Router + Firewall	12
Router + NAT	12
Router + NAT + Firewall	12
Resultaat	13
Conclusie	17
Bronnen.....	18
Bijlage 1: Regels	19
Ipfiler.....	19
Ipnat.....	20
Iptables.....	20

Inleiding

Een pakketfilter confronteert een pakket in doorgang met een set regels. Wanneer een overeenkomstige regel wordt gevonden, wordt de bijhorende actie op het pakket uitgevoerd. Dit betekent over het algemeen dat een pakket op basis daarvan wel of niet wordt doorgelaten. De verwerkingstijd die een pakketfilter nodig heeft per pakket hangt af van het aantal regels.

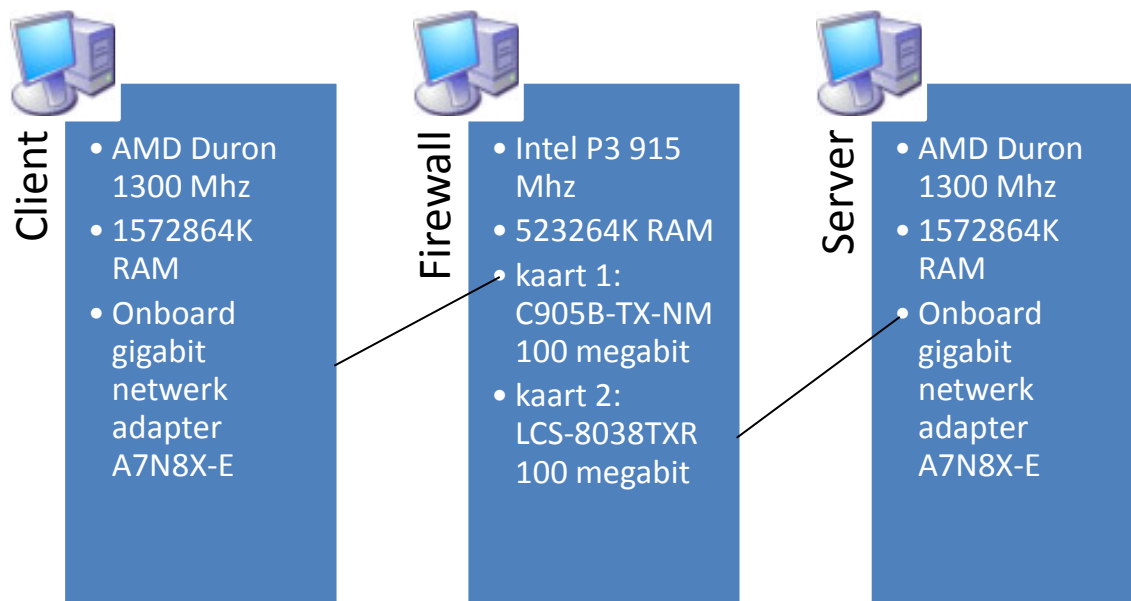
In dit professionaliseringsproject vergelijken we de verwerkingssnelheid tussen Iptables op Linux en Ipfilter op FreeBSD. Het hoofddoel is de pakketverwerkingssnelheid van beide methodes meten en bepalen hoe verschillende situaties de prestaties beïnvloeden. Daarom worden de metingen onder verschillende omstandigheden verricht.

Een firewall kan pakketten behandelen als een routing firewall (layer 3) of als een overbruggingsfirewall (layer 2). Firewalls gebaseerd op Linux en FreeBSD worden vaak als een routing firewall ingezet, maar bieden allebei de mogelijkheid om als een overbruggingsfirewall te functioneren, dus worden beiden ook onder deze omstandigheden getest.

Testomgeving

Onze configuratie bestaat uit 3 testcomputers, uitgerust met Fast-Ethernet netwerkkaarten en met elkaar verbonden via 2 CAT5 UTP crossover kabels (volgens RFC2544) [1].

Er zijn geen andere apparaten met de testcomputers verbonden, zodat niets anders hun gedrag kan beïnvloeden. De enige pakketten die over de kabels passeren zijn gegenereerd door de testcomputers.



Figuur 1: Basis testconfiguratie

De firewall heeft de minder RAM geheugen toegewezen gekregen en is lager geklokt om er voor te zorgen dat de firewall een bottleneck vormt in de communicatie tussen de client en de server. Op deze manier komen de verschillen in de metingen beter tot uiting.

Testmethode

Om de efficiëntie van de pakketfilters te vergelijken, meten we de vertraging die optreedt in de communicatie tussen de client en de server via de firewall.

Deze vertraging is afhankelijk van het besturingssysteem op de firewall (Linux/FreeBSD), de manier waarop de pakketten worden geforward (router of brug) en het al dan niet filteren van pakketten op de firewall.

Er werden tests uitgevoerd om de doorvoersnelheid van UDP en TCP verkeer met verschillende framegroottes te meten met een verschillend aantal filterregels op de firewall. Er werd gekozen voor vier framegroottes die voor Ethernet communicatie worden aangeraden in RFC 2544 [1]: 64, 256, 512 en 1024 bytes. Voor elk van deze 4 framegroottes herhalen we de tests met een verschillend aantal filterregels (20, 100 en 500 regels).

Doel

1. Het eerste doel is om de verwerkingsnelheid van de pakketfilters te vergelijken tussen Ipfiler op FreeBSD en IPtables op Linux. We testen deze firewalls onder verschillende configuraties met een verschillend aantal filterregels.
2. De besturingssystemen die gebruikt worden voor deze testen, Linux en FreeBSD, kunnen zowel de rol van een router als die van een brug aan. Voor beide besturingssystemen wordt er gekeken als de brugmogelijkheid efficiënter is dan de routermogelijkheid.
3. Beide besturingssystemen kunnen ook Network Address Translation aan, per platform wordt de impact daarvan op de prestaties besproken.
4. Het vierde doel is het vergelijken van de doorvoersnelheid tussen TCP pakketten en UDP datagrammen wanneer de firewall zich enkel op één type pakket focust (TCP of UDP). De doorvoersnelheden van beide types worden vergeleken bij filterregels voor beide pakkettypes (Het TCP verkeer wordt dus ook gemeten met UDP regels, en omgekeerd) .

Benchmark

Om het verkeer te genereren en de doorvoersnelheid te meten wordt er gebruik gemaakt van Netperf 2, een netwerkprestatie benchmark van Hewlett-Packard, terug te vinden op netperf.org en ontworpen met een eenvoudige client-server opzet in gedachten.

Door het uitvoeren van de Netperf client op *nw-client*, wordt er een controleverbinding [2] gemaakt met de Netserver op *nw-server* om de configuratiegegevens en resultaten naar elkaar door te sturen.

Eenmaal de configuratiegegevens over de controleverbinding zijn verzonden, wordt een aparte verbinding opgezet voor de eigenlijke meting. Netperf verstuurt geen verkeer over de controleverbinding tijdens een meting [2].

Netperf wordt in dit professionaliseringsproject aangewend voor het meten van request-response prestaties.

Door het uitvoeren van Netperf vanaf de commandolijn kunnen we elke test een aantal parameters meegeven zoals het gebruikte protocol (TCP of UDP), de pakketgrootte voor request-responses, de lengte van de test (in seconden) en/of het aantal iteraties. Een transactie staat gelijk aan een enkele request-response.

Elke test wordt onder 4 verschillende omstandigheden uitgevoerd:

1. Request-response grootte = 1024 bytes; protocol = TCP; testtijd = 120 s.
2. Request-response grootte = 512 bytes; protocol = TCP; testtijd = 120 s.
3. Request-response grootte = 256 bytes; protocol = TCP; testtijd = 120 s.
4. Request-response grootte = 64 bytes; protocol = UDP; testtijd = 120 s.

Testsessies

We voeren 13 testsessies uit. Een enkele testsessie bestaat uit een serie tests in een gegeven configuratie.

In sessie 1 verbinden we *nw-client* en *nw-server* met een crossover UTP kabel. In deze sessie krijgt *nw-client* een 10.0.0.1/24 IP adres en *nw-server* krijgt 10.0.0.2/24 toegewezen.

Deze “directe configuratie” wordt zo genoemd doordat de communicatie tussen *nw-client* en *nw-server* plaatsvindt zonder tussenliggende apparaten (router, hub, brug, switch, ...).

De overige 12 sessies gebeuren met *nw-firewall* als verbinding tussen *nw-client* en *nw-server*. In deze configuratie worden transacties gegenereerd (en gemeten) door Netperf tussen *nw-client* en *nw-server* wanneer het netwerkverkeer door *nw-firewall* verloopt, die dienst doet als een knelpunt voor de verbinding en een vertraging veroorzaakt: door het vergelijken van deze configuraties en de “directe” configuratie, meten we de vertraging die *nw-firewall* veroorzaakt.

We herhalen elke testsessie 3 keer, waarbij er telkens zeer gelijkaardige resultaten worden behaald. We noteren alleen het gemiddelde van de betrouwbare resultaten.

Sessie	Besturingssysteem firewall	Configuratie
1		Direct
2	FreeBSD 7.1	Brug
3	FreeBSD 7.1	Brug + Firewall
4	FreeBSD 7.1	Router
5	FreeBSD 7.1	Router + Firewall
6	FreeBSD 7.1	Router + NAT
7	FreeBSD 7.1	Router + NAT + Firewall
8	Ubuntu Server 8.10	Brug
9	Ubuntu Server 8.10	Brug + Firewall
10	Ubuntu Server 8.10	Router
11	Ubuntu Server 8.10	Router + Firewall
12	Ubuntu Server 8.10	Router + NAT
13	Ubuntu Server 8.10	Router + NAT + Firewall

Figuur 2: Tabel met testsessies en configuraties

Configuratie

Brug

IP adres voor *nw-client* = 10.0.0.1/24.

IP adres voor *nw-server* = 10.0.0.2/24.

IP adres voor *nw-firewall:kaart1* = 0.0.0.0.

IP adres voor *nw-firewall:kaart2* = 0.0.0.0.

IP adres voor *nw-firewall:brug1* = 10.0.0.3/24.

Om FreeBSD als een brug te laten werken (we geven de brug interface een IP adres om deze alsnog te kunnen aanspreken van op het netwerk), voeren we de volgende regels uit:

```
ifconfig em0 down
ifconfig em1 down
ifconfig bridge0 create
ifconfig bridge0 addm em0 addm em1 up
ifconfig bridge0 inet 10.0.0.3 netmask 255.255.255.0 up
ifconfig em0 inet 0.0.0.0 up
ifconfig em1 inet 0.0.0.0 up
```

Waarbij em0 en em1 respectievelijk verbonden zijn met *nw-client* en *nw-server*.

De brug deactiveren op FreeBSD gebeurt door middel van:

```
ifconfig bridge0 destroy
```

Bij Linux installeren we eerst bridge tools door *bridge-utils-1.4* af te halen van <http://sourceforge.net/projects/bridge/>.

Nu transformeren we *nw-client* naar een brug door het uitvoeren van:

```
ifconfig eth0 down
ifconfig eth1 down
brctl addbr br0
brctl addif br0 eth0
brctl addif br0 eth1
ifconfig br0 10.0.0.3 netmask 255.255.255.0 up
ifconfig eth0 0.0.0.0 up
ifconfig eth1 0.0.0.0 up
```

Om de brug op Linux te deactiveren voer je het volgende uit:

```
ifconfig eth0 down
ifconfig eth1 down
ifconfig br0 down
brctl delif br0 eth1
brctl delif br0 eth0
brctl delbr br0
```

Brug + Firewall

De brug configuratie blijft ongewijzigd, we activeren de firewall mogelijkheden op *nw-firewall* door gebruik te maken van een pakketfilter (iptables op Linux en ipfilter op FreeBSD). Elk pakket wordt onderzocht door de pakketfilter, die beslist wat er met dat pakket moet gebeuren (tegenhouden of doorlaten).

Om ipfilter op FreeBSD te activeren, laad je de kernel module in:

```
kldload ipl
```

Regels die in `/etc/ipf.regels` staan laad je in door:

```
ipf -Fa -f /etc/ipf.regels
```

Om de actieve filterregels te wissen gebruik je:

```
ipf -Fa
```

Ipfilter deactiveren gebeurt met:

```
kldunload ipl
```

Iptables heeft geen configuratiebestand nodig om regels in te laden. Hoewel filterregels één voor één met de hand kunnen worden ingevoerd op de opdrachtregel, is het beter om deze in een script te verzamelen.

De iptables filter tabel gebruikt een lineair zoekalgoritme: de datastructuur is een lijst met regels, en een pakket wordt in volgorde vergeleken met elke regel totdat een regel is gevonden die overeenkomt met alle relevante velden. Ipfilter gebruikt een gelijkaardig zoekalgoritme, maar standaard beslist de laatste overeenkomende regel die van toepassing is (en niet de eerste) wat er met het pakket gebeurt.

Echter, als een regel in ipfilter de optie "quick" bevat, wordt deze regel als de laatste

overeenkomende regel beschouwd en worden de daarop volgende regels niet meer vergeleken.

Er wordt van start gegaan met een set van 20 filterregels, dewelke elk een TCP pakket blokkeren op een specifieke poort naar *nw-server* toe. Geen enkele van de pakketten die door Netperf wordt gegenereerd en uitgewisseld tussen *nw-client* en *nw-server* in de tests komen overeen met een filterregel (een volledig overzicht van de gebruikte regels is terug te vinden in Bijlage 1). We dwingen de pakketfilter om elk pakket met de volledige set regels te vergelijken om ze daarna door te laten. Op deze manier kunnen we de vertraging die de pakketfilter veroorzaakt meten, dewelke de volledige lijst met regels moet verwerken.

We herhalen vervolgens de tests met 100 en 500 regels voor TCP pakketten, en uiteindelijk met een lijst van 500 regels voor UDP datagrammen.

Router

IP adres voor *nw-client* = 10.0.1.2/24.

IP adres voor *nw-server* = 10.0.0.2/24.

IP adres voor *nw-firewall:kaart1* = 10.0.1.1/24.

IP adres voor *nw-firewall:kaart2* = 10.0.0.1/24.

De *nw-client* heeft een expliciete gateway nodig om alle verkeer die voor *nw-server* is bestemd via *nw-firewall* te sturen:

```
nw-client# route add 10.0.0.2 gw 10.0.1.1
```

Nw-server heeft een gelijkaardige opdracht nodig:

```
nw-server# route add 10.0.1.2 gw 10.0.0.1
```

Om *nw-server* als een router te laten werken, moeten we IP forwarding activeren.

Op FreeBSD kan dit als volgt worden gedaan:

```
nw-firewall# sysctl -w net.inet.ip.forwarding=1
```

Terwijl op Linux:

```
nw-firewall# sysctl -w net.ipv4.ip_forward=1
```

In deze “router” configuratie wordt geen pakketfilter geactiveerd.

Router + Firewall

Door het activeren van een pakketfilter en het inladen van filterregels (op dezelfde manier zoals in de brug + firewall sessie), wordt de router configuratie een router + firewall.

Router + NAT

Om Network Address Translation te activeren op FreeBSD, maken we gebruik van de ingebouwde ondersteuning bij ipfilter, namelijk ipnat.

NAT regels (terug te vinden in Bijlage 1) die in /etc/nat.regels staan laad je in door:

```
ipnat -F -f /nat/ipf.regels
```

Ipnat werkloos maken gebeurt met:

```
ipnat -F
```

Op Linux gebruik je iptables om NAT in te schakelen:

```
iptables -t nat -A POSTROUTING -o eth1 -j MASQUERADE
```

Ongedaan maken doe je met behulp van:

```
iptables -F nat
```

Router + NAT + Firewall

Door het activeren van een pakketfilter en het inladen van filterregels (op dezelfde manier zoals in de brug + firewall sessie), wordt de router + NAT configuratie een router + NAT + firewall.

Resultaat

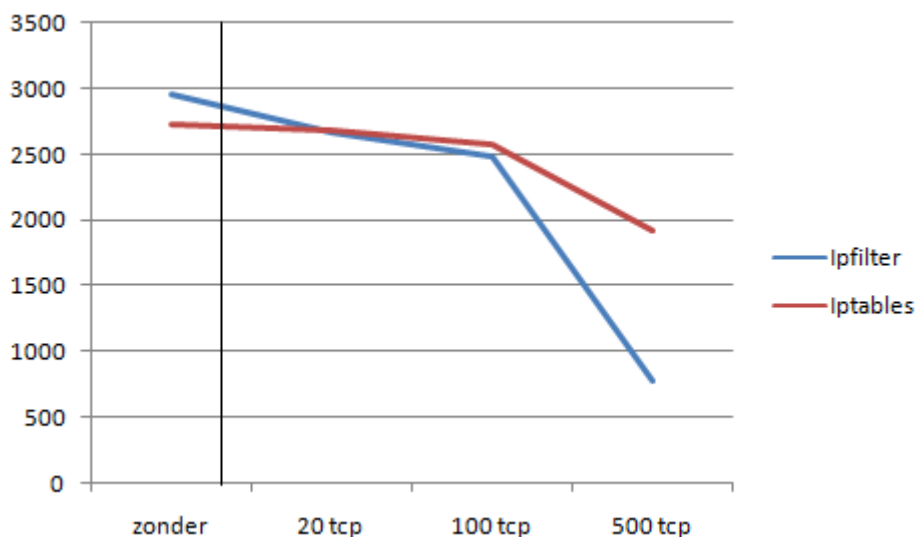
De resultaten zijn vermeld als het aantal transacties per seconde.

		Zonder pakketfilter		20 TCP regels		100 TCP regels		500 TCP regels		500 UDP regels	
		FreeBSD	Ubuntu	IPFilter	Iptables	IPFilter	Iptables	IPFilter	Iptables	IPFilter	Iptables
Direct	UDP 64	11514,59		nvt							
	TCP 256	7477,77									
	TCP 512	5672,08									
	TCP 1024	3822,53									
Brug	UDP 64	6197,53	5562,13	5395,77	5486,65	4962,29	5201,21	1622,72	4140,78	930,59	3470,19
	TCP 256	4018,94	3662,72	3571,14	3601,67	3264,32	3395,42	855,63	2388,61	1414,18	2879,15
	TCP 512	2954,60	2714,67	2648,32	2675,90	2499,37	2568,21	791,45	1943,26	1247,95	2258,51
	TCP 1024	1882,78	1801,31	1788,37	1785,50	1696,72	1736,57	688,87	1422,19	1011,39	1580,47
Router	UDP 64	6111,19	5699,15	5495,79	5602,60	5494,20	5320,71	5481,66	4123,72	5490,43	3222,88
	TCP 256	3965,91	3719,26	3693,65	3649,87	3685,40	3453,55	3690,82	2602,16	3683,88	2986,08
	TCP 512	2818,29	2738,97	2758,97	2701,73	2757,05	2587,84	2754,20	2077,63	2758,29	2308,94
	TCP 1024	1848,22	1810,66	1828,46	1797,84	1825,13	1742,85	1826,34	1494,66	1826,20	1612,44
Router + NAT	UDP 64	6114,53	5692,34	5488,21	5595,74	5472,31	5285,55	5480,05	4166,49	5479,23	2647,47
	TCP 256	3947,72	3711,73	3684,19	3654,73	3680,93	3443,24	3683,68	2326,92	3680,13	2928,26
	TCP 512	2819,52	2743,97	2753,10	2706,63	2753,09	2595,18	2755,03	1899,33	2755,29	2293,39
	TCP 1024	1846,25	1810,15	1824,45	1794,31	1822,94	1741,96	1822,31	1396,74	1824,98	1603,95

Figuur 3: Resultaat

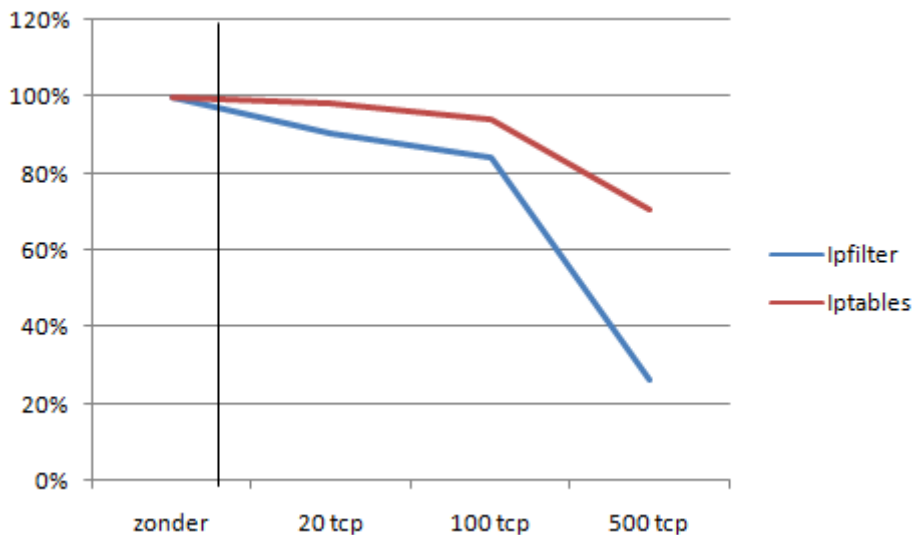
Zoals verwacht zorgt de aanwezigheid van *nw-firewall* tussen *nw-client* en *nw-server* voor een groot verlies aan doorvoersnelheid: het aantal transacties per seconde valt in de overige configuraties terug tot bijna de helft ten opzichte van directe communicatie.

Wanneer er gekeken wordt naar de resultaten voor iptables, zien we duidelijk dat de tijd om een pakket te classificeren groeit naarmate het aantal filterregels toeneemt, zowel voor TCP als UDP. Er is wel een snelheidstoename zichtbaar bij het vergelijken van UDP verkeer tegenover TCP filterregels, en omgekeerd.



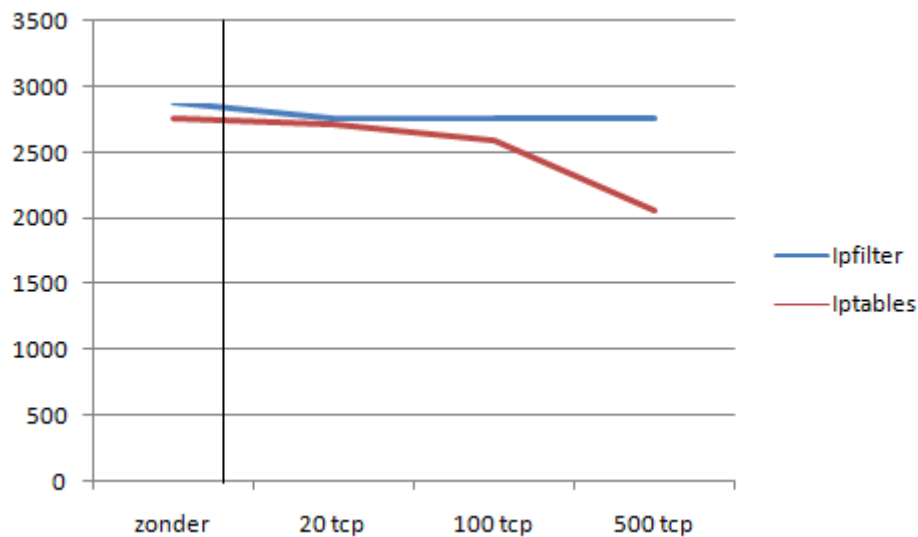
Figuur 4: Resultaat brug configuratie in absolute cijfers

Iptables vergelijkt een pakket met de filterregels, beginnend met de eerste, om vervolgens deze regels één voor één af te lopen tot er een overeenkomstige regel is gevonden. Wanneer een pakket overeenkomt met een filterregel, wordt er niet meer verder vergeleken maar wordt de actie verbonden met de huidige regel uitgevoerd. Aangezien geen enkel pakket die de firewall doorkruist overeenkomt met één van de filterregels, wordt in elke iptables sessie elk pakket vergeleken met elke regel.

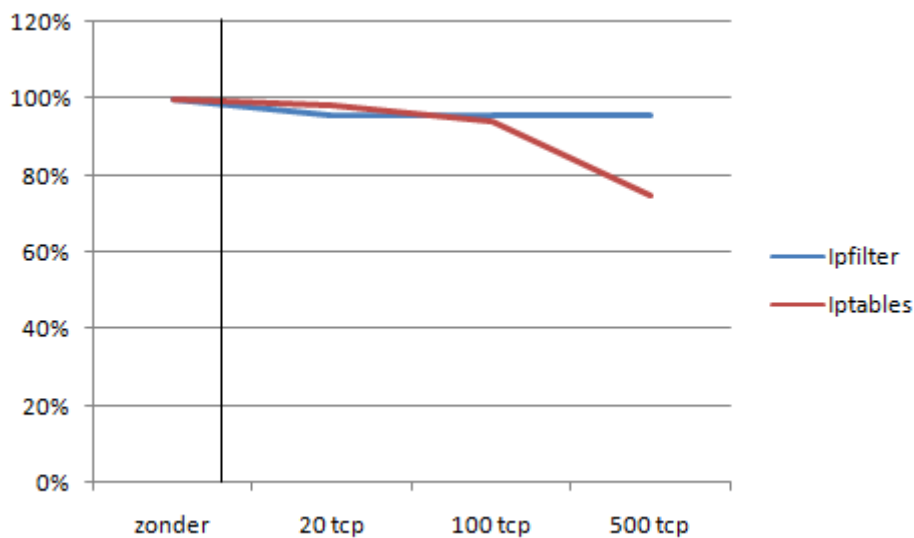


Figuur 5: Resultaat brug (TCP) configuratie in relatieve cijfers

Ipfilter gaat op een andere, efficiëntere manier te werk. Wanneer een lijst regels ingeladen wordt overloopt ipfilter deze lijst om de zogehete *skip-steps*, of overslaanbare regels, te berekenen. In elke regel, voor elke parameter, is er een verwijzing naar de volgende regel die een andere waarde herkent voor deze parameter. Tijdens het vergelijken van de pakketten met de filterregels wordt er, indien een pakket niet overeenstemt met één van de parameters, gekeken naar de bijhorende verwijzing om naar een volgende regel te gaan die zou kunnen overeenkomen, in plaats van de volgende regel in de lijst met filterregels uit te proberen. Wanneer de firewall echter als brug is geconfigureerd, blijkt ipfilter deze optimalisatie niet door te voeren en vertoont deze een gelijkaardig (lineair) gedrag als iptables (zie figuur 4).

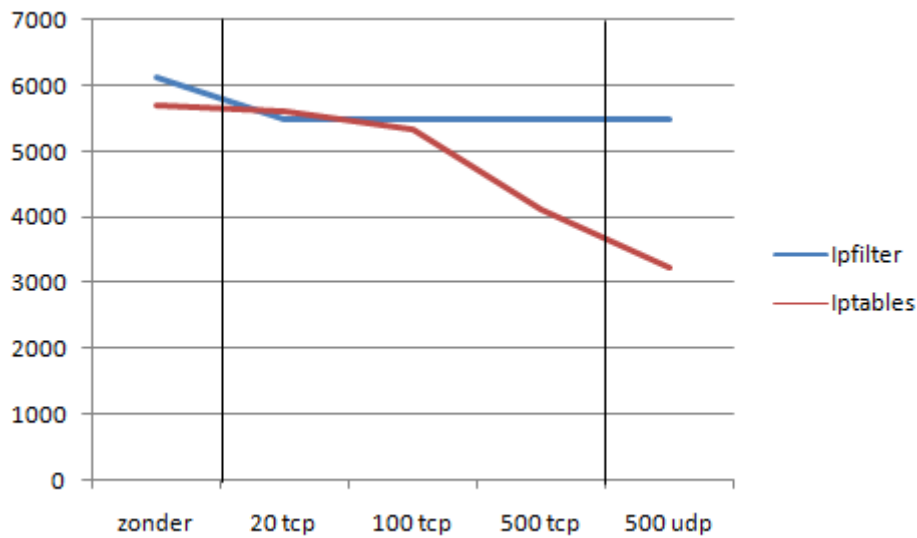


Figuur 6: Resultaat (TCP) router in absolute cijfers



Figuur 7: Resultaat (TCP) router in relatieve cijfers

Als we naar de resultaten van ipfilter kijken, zien we dat wanneer het verkeer enkel uit UDP datagrammen bestaat en de filterregels zich toeleggen op TCP verkeer, we een constante hoeveelheid transacties zien ongeacht het aantal regels (20, 100 en 500): het aantal TCP filterregels heeft geen effect op het aantal UDP transacties tussen *nw-client* en *nw-server*. Hetzelfde geldt ook voor het aantal TCP transacties wanneer deze uitsluitend met UDP regels worden vergeleken.



Figuur 8: Resultaat (UDP) router configuratie in absolute cijfers

Algemeen beschouwd zijn de netwerkprestaties van FreeBSD beter dan die van Linux. Merk op dat FreeBSD + Firewall minder efficiënt is in een brug configuratie dan in een router configuratie, voor Linux is er geen belangrijk verschil in doorvoersnelheid bij beide configuraties.

De impact van NAT op de transacties die door de firewall verlopen is zeer gering.

Conclusie

Ipfiler wordt het best in een router omgeving neergezet en komt het best tot zijn recht in een omgeving met veel filterregels, waar iptables op dit ogenblik de beste keuze is voor een brug configuratie.

De mogelijkheid van ipfilter om transport-protocollen te onderscheiden dankzij de automatische optimalisatie van de filterregels is met name interessant om bijvoorbeeld op UDP-gebaseerde aanvallen van zich af te slaan – hoewel het merendeel van het verkeer van en naar een beschermd netwerk meestal uit TCP zal bestaan (protocollen zoals HTTP, SMTP en FTP). In een dergelijk geval kan ipfilter doordat hij geen UDP datagrammen met TCP filterregels zal vergelijken eventuele vertragingen van legitieme pakketten voorkomen.

In tegenstelling tot iptables zal ipfilter dus automatisch z'n filterlijst optimaliseren, en het verwerken in meermaals gelinkte lijsten [7, 8]. Echter kan je in iptables het zoekproces naar een overeenkomstige regel versnellen door gebruik te maken van een "jump" parameter [4] om naar een subset van de regels te springen (een *chain*) gereserveerd voor TCP of UDP pakketten, afhankelijk van het protocol type.

Bronnen

[1] S. Bradner and J. McQuaid, "Benchmarking Methodology for Network Interconnect Devices, RFC 2544," 1999.

[2] Information Network Division of the Hewlett-Packard Company, "Netperf: A Network Performance Benchmark," Revisie 2.0, Februari 1995:
<http://www.netperf.org/netperf/training/Netperf.html>.

[3] Brendan Conoboy, Erik Fichtner, IP Filter-Based Firewalls Howto, 2002:
<http://www.obfuscation.org/ipf/ipf-howto.txt>.

[4] Linux documentatie, man iptables(8).

[5] Linux documentatie, man brctl(8).

Bijlage 1: Regels

Ipfilter

De set van 20 filterregels die worden gebruikt in de tests:

```
pass in from any to any
pass out from any to any
block in proto tcp from any to 10.0.0.2 port = 3001
block in proto tcp from any to 10.0.0.2 port = 3002
block in proto tcp from any to 10.0.0.2 port = 3003
block in proto tcp from any to 10.0.0.2 port = 3004
block in proto tcp from any to 10.0.0.2 port = 3005
block in proto tcp from any to 10.0.0.2 port = 3006
block in proto tcp from any to 10.0.0.2 port = 3007
block in proto tcp from any to 10.0.0.2 port = 3008
block in proto tcp from any to 10.0.0.2 port = 3009
block in proto tcp from any to 10.0.0.2 port = 3010
block in proto tcp from any to 10.0.0.2 port = 3011
block in proto tcp from any to 10.0.0.2 port = 3012
block in proto tcp from any to 10.0.0.2 port = 3013
block in proto tcp from any to 10.0.0.2 port = 3014
block in proto tcp from any to 10.0.0.2 port = 3015
block in proto tcp from any to 10.0.0.2 port = 3016
block in proto tcp from any to 10.0.0.2 port = 3017
block in proto tcp from any to 10.0.0.2 port = 3018
block in proto tcp from any to 10.0.0.2 port = 3019
block in proto tcp from any to 10.0.0.2 port = 3020
```

De eerste twee regels (*from any to any*) laten elk pakket door (*pass*).

De andere twintig regels blokkeren (*block*) alle binnenkomende (*in*) pakketten komende van elke bron (*from any*) en bestemd voor een bepaalde poort (*port = x*) van de *nw-server* (*to 10.0.0.2*).

Ipfilter vergelijkt elk passerend pakket met bovenstaande 20 regels: wanneer een regel overeenkomt met een pakket, wordt de bijhorende actie (*block*) toegepast.

Zoniet wordt het pakket doorgelaten.

De sets van 100 (en 500) filterregels bevatten gelijkaardige regels zoals hierboven, maar dan 100 (500) keer voor 100 (500) verschillende poorten bestemd.

Volgende set van 500 “UDP regels” filtert enkel UDP datagrammen bestemd voor *nw-firewall* op 500 verschillende poorten.

```
pass in from any to any
pass out from any to any
block in proto udp from any to 10.0.0.2 port = 3001
...
block in proto udp from any to 10.0.0.2 port = 3500
```

Ipnat

De volgende ipnat regels worden gebruikt in de test:

```
map em1 10.0.1.0/24 -> 10.0.0.0/24 portmap tcp/udp auto
map em1 10.0.1.0/24 -> 10.0.0.0/24
```

De eerste regel vertaalt (*map*) alle verkeer die naar een bepaalde interface wordt gestuurd (*em1*) en afkomstig is van een specifiek intern netwerk (*10.0.1.0/24*) naar het beoogde externe doelnetwerk (*10.0.0.0/24*), en vertaalt daarbij de poorten van het interne netwerk (*portmap*) enkel voor het TCP en UDP protocol (*tcp/udp*) en kiest automatisch de beschikbare poorten daarvoor (*auto*).

Op de tweede regel worden de overige protocollen behandeld zonder poortmapping.

Iptables

De volgende regels implementeren op de Linux firewall een gelijkaardige set filterregels zoals op FreeBSD het geval is:

```
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3001 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3002 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3003 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3004 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3005 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3006 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3007 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3008 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3009 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3010 -j DROP
```

```
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3011 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3012 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3013 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3014 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3015 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3016 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3017 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3018 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3019 -j DROP
iptables -A FORWARD -d 10.0.0.2 -p tcp --dport 3020 -j DROP
iptables -A FORWARD -j ACCEPT
```

Deze set regels dropt (*-j DROP*) alle doorgaande (*-A FORWARD*) TCP pakketten (*-p tcp*) gericht aan *nw-server* (*-d 10.0.0.2*) bestemd voor bepaalde poorten (*--dport x*).

De laatste regel heeft als doel om de overige pakketten, die niet geblokkeerd worden door de 20 voorafgaande regels, door te laten (*-j ACCEPT*).

Door deze filterregels 100 (500) keer te herhalen, bekomen we een pakketfilter die 100 (500) poorten blokkeert voor *nw-firewall*.

Hieronder volgt een pakketfilter toegespitst op 500 UDP datagrammen:

```
iptables -A FORWARD -d 10.0.0.2 -p udp --dport 3001 -j DROP
...
iptables -A FORWARD -d 10.0.0.2 -p udp --dport 3500 -j DROP
iptables -A FORWARD -j ACCEPT
```

Merk op dat beide soorten regels, voor zowel ipfilter als iptables, zodanig zijn gekozen dat de filter ze allemaal moet vergelijken met elk passerend pakket alvorens te beslissen om deze te aanvaarden.

Ipfilter loopt zelfs standaard alle regels af (volgens de volgorde van verwerking) en baseert de actie aan de hand van de laatst overeenkomende regel. Geen enkele van de blokkeerregels komt overeen met een passerend pakket gegenereerd door de benchmark (Netperf gebruikt de poorten van 3000 tot 3500 niet), behalve de eerste twee regels, dus het pakket wordt doorgelaten.

Iptables past standaard de eerst overeenkomende regel toe, maar geen enkele van de drop regels komt overeen, behalve de laatste, die het pakket laat accepteren.