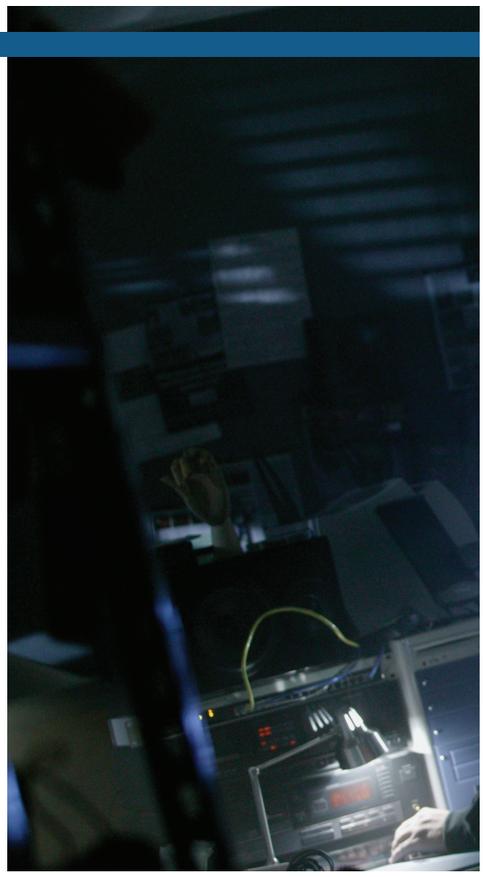
Næste generations højtydende firewall

I denne artikel træder vi et skridt ind i fremtiden – vi skal nemlig se nærmere på de kommende »high-performance« firewalls



Af Kristian Sørensen Foto: GettyImages

På Aalborg Universitet har to forskere udviklet en metode til at få firewalls til at yde meget, meget mere, end det hidtil har været muligt. Vi har talt med de to forskere Mikkel Christiansen og Emmanuel Fleury, som står bag projektet *Compact Filter*, som er en lille og særdeles højtydende firewall til Linux. Senere i artiklen vil vi kort gennemgå, hvordan Compact Filter tilføjes til Linux-kernen, og lave en testopsætning af firewallen.

Teknikken bag de revolutionerende ideer, som forskerne Mikkel Christiansen og Emmanuel Fleury har udviklet, er særdeles avanceret, og for at forstå detaljerne skal man have indgående kendskab til flere aspekter af datalogi. Vi vil dog alligevel i denne artikel på en overskuelig måde præsentere de nyskabende resultater og baggrunden for deres udvikling.

Hvis man skal forstå hele artiklen, kræves der dog en smule forkundskab til firewalls. I Alt om DATA 9/2003 har vi skrevet en introduktion til firewallen Iptables, som er indbygget i Linux 2.4 og 2.6. Har du ikke bladet, kan artiklen købes online: *itsvar.dk/articles/article.aspx?id=5058*.

Netværkstrafik

Det, en firewall beskæftiger sig med, er filtrering af netværkstrafik. Data, der overføres over et netværk, sendes af sted i små klumper, nemlig i såkaldte *pakker*. En pakke kan groft set sammenlignes med et normalt brev med en afsender og en modtager – i pakkerne er afsender og modtager IP-numre. Desuden er der også specificeret en port, der skal modtage pakken. Det svarer nogenlunde til at specificere én ud af flere personer, som bor på samme adresse. Eksempelvis er IP-adressen 129.142.226.101 en af Alt om DATAs servere, og når man taster dette nummer ind i en browser, bliver serveren kontaktet på port 80, som tilhører webserveren – og derfor kaldes hjemmesiden frem.

Ideen bag Compact Filter

Det hele startede tilbage i efteråret 2001, mens Emmanuel Fleury, der har baggrund inden for verifikation, og Mikkel Christiansen, som har baggrund inden for netværk og protokoller, delte kontor.

Iptables var lige begyndt at tilbyde *stateful filtrering*, og da dette var en meget efterspurgt feature, besluttede Mikkel og Emmanuel at afprøve det. En forklaring på »stateful filtrering« kan findes i boksen til venstre.

Stateful filtrering

Filtrering uden tilstande, eller »states«, for forbindelser er mindre komplicerede at implementere, men til gengæld svære at konfigurere og vil i nogle tilfælde være mindre sikre end firewalls, som er i stand til at holde styr på tilstandene for forbindelserne.

Hvis man ser på stateless filtrering, vil enhver pakke, som ankommer til computeren, være en *ny* pakke uden nogen relationer til pakker, der kom før, eller som kommer efter. Det meste netværkstrafik foregår mellem to parter, og derfor skal stateless filtrering have en regel, der tillader både forespørgsler og svar i hver retning.

For at tillade at en klientcomputer på et internt netværk har adgang til at surfe på internettet, som er udenfor, skal man med stateless filtrering tillade udgående trafik på port 80, som er den officielle webserverport. Herudover skal webserverne have mulighed for at sende data til computerne på lokalnettet, hvilket ligeledes foregår over port 80.

Når man laver en webforespørgsel, som når man f.eks. besøger en hjemmeside, kommer forespørgslen fra en tilfældigt valgt port med højt nummer på computeren, og derfor skal trafikken have lov til at komme igennem netværket på alle høje porte. Dette udgør en sikkerhedsrisiko, da alle processer i princippet kan åbne en forbindelse på en høj port.

For at gøre det mere sikkert, tillader man svar til en IP-adresse og port, hvorfra en forespørgsel er sendt, samt fra IP-adressen og porten, som forespørgslen er sendt til. Denne proces kaldes *stateful filtrering*.

Holdet bag Compact Filter: Emmanuel Fleury (t.v.) og Mikkel Christiansen (t.h.).
Begge arbejder dog også på andet end Compact Filter under deres daglige gang på Aalborg Universitet. Emmanuel Fleury arbejder bl.a. med spilteorier, hvor man ud fra f.eks. tidsautomater kan finde optimale strategier. Mikkel Christiansen arbejder med realtidsafvikling af javaprogrammer i små indlejrede systemer.

Iptables blev afprøvet i en »testbed« (et lukket netværk udelukkende brugt til test). Her blev der simuleret et netværk med 3.000 brugere med normal internetaktivitet. Det tog ca. 15 sekunder, så løb maskinen tør for hukommelse og gik derefter ned. Og det var mildt sagt ikke særlig imponerende.

Et basalt filter

Filtrering af pakker ender altid i enten et *ja* eller et *nej* til spørgsmålet om, hvorvidt pakken må komme igennem filteret.

Der skulle startes helt fra grunden, og efter at have filosoferet over hvordan den helt basale pakkefiltrering kan laves, opstod ideen om at benytte det binære beslutningsdiagram. Flere andre havde allerede forsøgt dette, og forsøgene var altid slået fejl. Efter endnu flere grublerier fandt Mikkel og Emmanuel frem til, at den helt basale filtrering af pakker kan gøres med interval beslutningsdiagrammer.

Metoden har baggrund i verifikation, og det handler om at finde ud af, om en pakke er inden for et givet interval. Og ud fra det skal der således tages en beslutning, om pakken må passere eller ej.

Problemerne med Iptables

Der er mange punkter, hvor Iptables kan siges at være for besværligt og ulogisk, og samtidig er der også store problemer med hukommelsen.

Iptables tilbyder stateful filtrering, hvor en given forbindelsestilstand overvåges og bruges i filtreringen. Det bruger enorme mængder hukommelse, og der åbnes for et *Denial of Service*-angreb, hvis en angriber opretter mange åbne forbindelser. Læs mere om *DoS* i Alt om DATAs sikkerhedsguide på side 4.

Holdet bag Iptables er meget konservative, og da de har mange millioner brugere verden over, er de fanget af deres egen succes. Hvis de ændrer i syntaksen for filtreringsregler, er der milliarder af linjer, der skal ændres i firewalls verden over. Desuden skal alle hjælpeværktøjerne genudgives i nye versioner.

Generelt er alle firewalls i dag sat sammen af små hacks, som udelukkende er et produkt af programmørernes egne ideer. Der findes ingen officiel måde at lave en firewall på. Mikkel og Emmanuel har følgende taget elementer og teknikker fra verifikation og brugt dem til at lave regler for, hvordan man *bygger* firewalls. Compact Filter er et eksempel på, at byggestenene kan bruges i virkeligheden.

I øjeblikket er en firewalls kompleksitet bestemt af antallet af regler, der skal filtreres på. De fleste firewalls bruger en pragmatisk tilgangsvinkel, hvor reglerne gennemgås én ad gangen, og når en regel matcher en pakke, udføres den (altså accepteres eller afvises). Det bevirker, at regler, der hyppigst vil blive matchet, *skal* stå øverst for at undgå et for stort overhead i firewallen. Dette kan man kalde for *håndoptimering*, og man er derfor fuldstændig afhængigt af, at administratoren er dygtig til det. Der er naturligvis lavet forskellige værktøjer til at hjælpe med optimeringen, men det hjælper ikke, hvis det helt basale design af firewallen ikke er godt nok. →

Links

Compact Filter
cs.aau.dk/~mixxel/cf

Iptables
netfilter.org

Linux-kernen
ftp.sunsite.dk/pub/os/linux/kernel.org/kernel/v2.6

Aalborg Universitet
aau.dk, aauni.dk

Datalogi ved Aalborg Universitet
cs.aau.dk

Mikkel Christiansens hjemmeside
cs.aau.dk/~mixxel

Emmenuel Fleurys hjemmeside
cs.aau.dk/~fleury

Mål for Compact Filter

Selve Compact Filter er implementeret for at teste, at teorien holder i praksis. Udvidelse af Compact Filter med alle de features, der findes i andre firewalls, er derfor ikke et opprioriteret mål.

Ved udviklingen af Compact Filter har det vist sig, at det faktisk er muligt at benytte klassisk algebra på firewallfiltre. Senere har det vist sig, at teknikken ligeledes kan benyttes på netværk til at opdatere IP-spoofing, som er en teknik, der bruges til teoretisk at undersøge, hvilken trafik der kan komme igennem et netværk.

Målet er, at ideerne implementeret i Compact Filter bliver videregivet og siden benyttet af andre, der lever af at bygge firewalls i f.eks. routere.

Status for Compact Filter

Selv om Mikkel og Emmanuel hovedsagelig fokuserer på forskningen frem for udviklingen af en firewall, er Compact Filter dog et særdeles spændende projekt.

I øjeblikket har de implementeret statiske filtre, og næste skridt på vejen er en implementering af stateful filtrering. Mikkel og Emmanuel regner med, at den viden, der allerede findes om blandt andet tidsautomater, kan bruges. Det vil gøre den grundlæggende datastruktur mere dynamisk og reducere load på runtime for firewallen. Tidsautomater er en model, der beskriver en tilstand på et program, hvor tilstanden kan skifte afhængigt af et fiktivt ur.

Fordelen ved tidsautomater er bl.a., at brugeren vil være klar over, hvilken tilstand en sådan maskine har på et bestemt tidspunkt, samt at der også vil være en klar beskrivelse tilgængelig, som ikke kun er skrevet i C. Teknikken for tidsautomater gør det også pludselig muligt at lave en formel verifikation af, at tidsautomaten ikke har en deadlock (en fejl, hvor alt går i stå) og ellers fungerer efter hensigten.

Det er lykkedes at finde selve kernen af, hvad en firewall i virkeligheden er. Herefter er der lavet en kompleks algoritme til at manipulere logikken. Når der her er tale om en fast matematisk logik, så er fordelen, at det kan oversættes til præcis det firewallsprog, man ønsker, og det er således helt generisk og fleksibelt. Konfigurationen, kommandoerne, firewall-linjerne kan simpelthen genereres ud fra en matematisk-logisk repræsentation.

Revolutionen

Revolutionen, der er frembragt og implementeret i Compact Filter, kan sammenlignes med overgangen fra fortolkede programmeringssprog, som f.eks. det gamle Basic, til sprog, der kan kompileres og følgende optimeres, inden det afvikles.

Denne optimering gør, at Compact Filter kan håndtere enormt store filtre på en smart måde, hvor andre firewalls må give op. For at presse Compact Filter bliver forsøgene udført på et lokalt gigabitnetværk – ellers

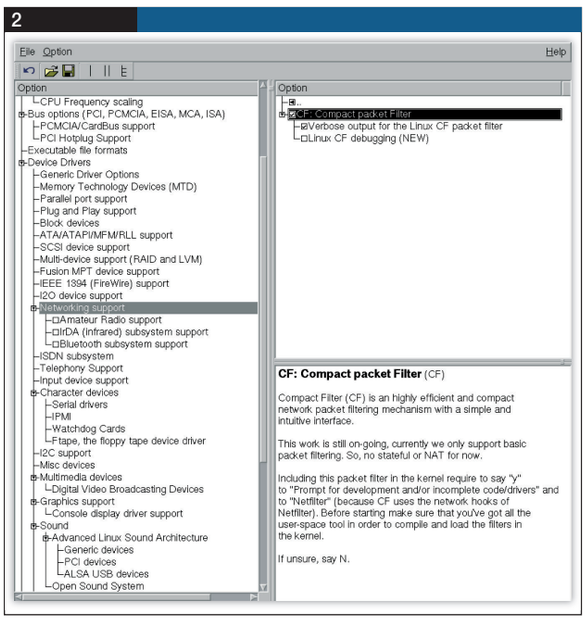
Patching af Linux 2.6.6 med Compact Filter.

1

```
qbox linux-2.6.6 # ls
COPYING         MAINTAINERS  REPORTING-BUGS  drivers  init    lib  patches.txt  sound
CREDITS         Makefile     arch            fs       ipc     mm   scripts      usr
Documentation   README       crypto          include  kernel  net  security
qbox linux-2.6.6 # bzcat /tmp/cf/linux-2.6.6-cf-0.2.patch.bz2 | patch -p1
patching file include/linux/cf/cf.h
patching file include/linux/cf/cf_netlink.h
patching file include/linux/cf/filter_candidate.h
patching file include/linux/cf/filters.h
patching file include/linux/cf/idd.h
patching file include/linux/cf/protocols.h
patching file net/Kconfig
patching file net/Kconfig.orig
patching file net/ipv4/Makefile
patching file net/ipv4/cf/Kconfig
patching file net/ipv4/cf/Makefile
patching file net/ipv4/cf/cf_core.c
patching file net/ipv4/cf/cf_netlink.c
patching file net/ipv4/cf/filter_candidate.c
patching file net/ipv4/cf/filters.c
patching file net/ipv4/cf/protocols.c
qbox linux-2.6.6 #
```

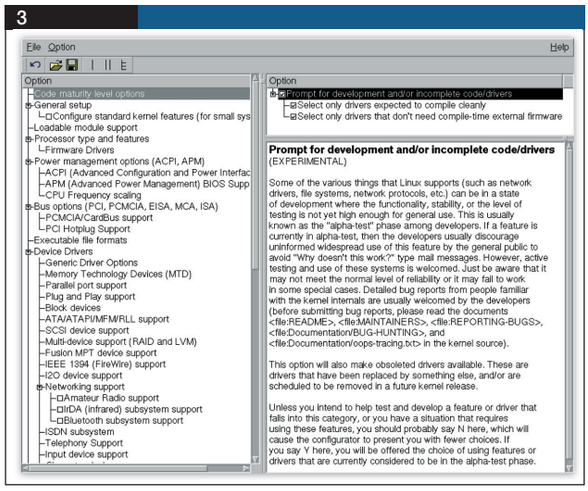
Konfiguration af Compact Filter i Linux-kernen.

2

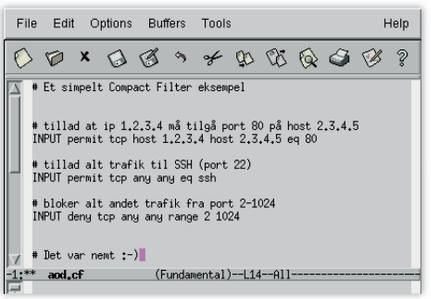


Konfigurer Linux-kernen til at vise nye drivere.

3



4



Simpelt firewallfilter til Compact Filter.

er der simpelthen ikke udfordring nok i opgaven.

Aalborg Universitet har investeret 100.000 kr. i udstyr, som benyttes til forskellige netværkssimuleringer for at afprøve teorien i praksis gennem Compact Filter.

Compact Filter i praksis

Det er nemt at afprøve Compact Filter, men det kan ikke anbefales, hvis man hverken har prøvet at bruge en firewall eller kompileret sin egen kerne. Følgende giver vi en lille introduktion til læserne, der vil være med på »bleeding edge« af næste generation af high-performance firewalls.

At installere Compact Filter er, når det kommer til stykket, ikke så svært – lav en kerne, installer administrationsprogrammerne og lav et filter.

Først skal man have en Linux 2.6.6 kernekode, links hertil findes i boksen *Links*. Udpak kildekoden i /usr/src, og lav et symbolsk link dertil, som hedder *linux*: ln -sf linux-2.6.6 linux.

Kernen skal *patches*, hvilket vil sige, at Compact Filter-koden tilføjes. Når patchen er hentet, stiller man sig i /usr/src/linux og patcher kernen: *bzcat linux-2.6.6-cf-0.2.patch.bz2 | patch -p1*. Er patchingen gået godt, ser du resultatet, som vist på figur 1.

Nu skal kernen konfigureres som normalt, men uden Iptables og med Compact Filter slået *til* (se figur 2). For at man kan se Compact Filter i kerne-konfigurationen, skal *Prompt for development and/or incomplete code/drivers* under *Code maturity level options* være slået til (se figur 3).

Næste skridt er at oversætte og installere administrationsprogrammerne. Det foregår på traditionel vis med *./configure*, *make* og *make install*.

Endelig skal filtret specificeres. Sproget ligner meget Ciscos syntaks og er meget behageligere at specificere end at skulle skrive en række mere eller mindre uforståelige kommandoer til Iptables. Reglerne kan, på grund af oversættelsen, sorteres, præcis som man har lyst – alt, hvad der hedder håndoptimering, er fortid! For detaljer her, må vi henvise til dokumentationen på hjemmesiden. På figur 4 kan du se en kort eksempel-konfiguration.

Compact Filter i test

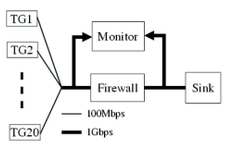
Det setup, der er brugt til at teste Compact Filter med, er illustreret i figur 5, hvor der er 20 klienter, som genererer trafik. Firewallen overvåges af en Cisco SPAN portmonitor sammen med en monitormaskine. Data, der går igennem firewallen, bliver blot smidt væk.

På figur 6 og 7 er Compact Filter, som er markeret med blå, sammenlignet med Iptables, der er markeret med rød. På figur 6 ses det tydeligt, at Iptables falder drastisk i ydelsen allerede ved 300 regler, mens Compact Filter ikke har ændret sig mærkbart. Figur 7 viser, at Compact Filter skal op på ca. 6.000 regler, førend ydelsen ændres betydeligt – bemærk den røde graf for Iptables i bunden. Den er sat *helt* ud af spillet ved den kraftige belastning, hvilket allerede skete ved omkring de 2.000 regler. Ud over Compact Filters imponerende ydelse, bruger den tilmed også mindre cpu-tid end Iptables.

Fremtidens firewalls

Der er meget arbejde forbundet med at vedligeholde koden til både Linux-kernen og de medfølgende administrationsprogrammer. Udviklingsmæssigt er målet at have en lille, men god og solid implementation med få features frem for mange ikke-optimerede features.

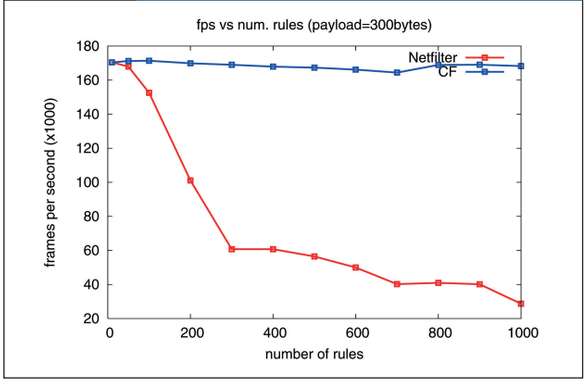
Figur 5. Et testsetup for Compact Filter.



Såfremt der skulle være seriøse programmører, som har lyst til at hjælpe med at udvikle og vedligeholde fremtidens firewall – *Compact Filter* – vil det i høj grad blive støttet og hjulpet af Mikkel og Emmanuel.

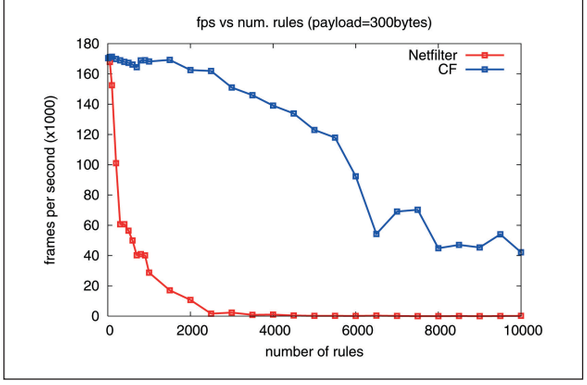
Ideerne bag Compact Filter er helt enestående og revolutionerende, og de fantastiske testresultater taler deres tydelige sprog. Og både Mikkels og Emmanuels ideer vil helt klart blive brugt til at bygge næste generations high-performance firewall verden over. ■

6



Iptables vs. Compact Filter med op til 1.000 regler.

7



Iptables vs. Compact Filter med op til 10.000 regler.