Linux - Friheden til sikkerhed på internettet

Version 2.9.20060510 - 2020-12-31

Hanne Munkholm

Peter Toft

Linux - Friheden til sikkerhed på internettetVersion 2.9.20060510 - 2020-12-31

af Hanne Munkholm og Peter Toft

Dette er en bog om Linux netværkssikkerhed. Emner som root-adgang, netværksservices, firewalls, sikker netværkskommunikation bliver behandlet.

Indholdsfortegnelse

Figurliste

Forord

Dette er en bog samlet ud fra en serie på 6 artikler om sikkerhed på Linux, som kan findes på http://www.sslug.dk/artikler/Linux_sikkerhed.

For få år siden var computersikkerhed et emne, de færreste behøvede at tænke på. Med den hastigt voksende udbredelse af internettet i dag er det et område, man ikke længere kan tillade sig at overse. Flere og flere computere kobles til internettet eller et lokalnet, og kan let blive ofre for ødelæggende indbrud, aflytning mm. Sikkerhed er for mange et emne, de ved, at de burde tænke på, men de finder det svært at komme i gang med at lære et så komplekst område. Da sikkerhed er et meget stort område, ved man ikke rigtig, hvor man skal starte, og det virker meget svært i begyndelsen. På internettet er der mange informationer at finde, men de findes spredt rundt omkring, og det er ikke altid let at forstå, hvor de enkelte "brikker" passer ind i det store puslespil.

Vi skriver disse artikler i håb om at hjælpe andre i gang med emnet, som er både spændende og til at forstå, når man først kommer i gang. Selvom du ikke lige nu har brug for at lære om sikkerhed, kan du alligevel være interesseret i at læse videre, idet du kommer til at lære hvor meget, din Linux-maskine kan på et netværk, og hvordan det kontrolleres.

1. Linux-bøgerne

Bogen er en del af en serie, som kan findes på http://www.linuxbog.dk/

- *Linux – Friheden til at vælge installation* – Om at installere Linux.
- *Linux – Friheden til at lære Unix* – Om hvordan man bruger Linux' (og Unix') kommandolinjeværktøjer.
- *Linux – Friheden til at vælge grafisk brugergrænseflade* – Om alle de grafiske brugergrænseflader, der findes til Linux.
- *Linux – Friheden til at vælge programmer* – Om de programmer du kan få til Linux.
- *Linux – Friheden til systemadministration* – Om at administre sit eget linuxsystem.
- *Linux – Friheden til at programmere* – Programmering på Linux
- *Linux – Friheden til at programmere i C* – Om at programmere i sproget "C".
- *Linux – Friheden til at programmere i Java* – Om at programmere i sproget "Java".
- *Linux – Friheden til sikkerhed på internettet* – Om at sikre dit Linuxsystem mod indbrud fra internettet.
- *Linux – Friheden til egen webserver* – Om at sætte en webserver med databaser, CGI-programmer og andet godt op.
- *Linux – Friheden til at skrive dokumentation* – Om at skrive dokumentation (og andet) i SGML/DocBook, LaTeX eller andre formater.

- *Linux – Friheden til at vælge kontorprogrammer* – Kontorfunktioner på et Linux/KDE/OpenOffice.org-system.
- *Linux – Friheden til at vælge IT-løsning* – Om muligheder, fordele og ulemper ved at bruge Linux i sin IT-løsning.
- *Linux – Friheden til at vælge OpenOffice.org* – Om at bruge OpenOffice.org, både på Linux og på andre styresystemer.
- *Linux – Friheden til at vælge digital signatur* – Digital signatur på Linux.

2. Ophavsret

Denne bog er skrevet af Linux-brugere til Linux-brugere. Store dele af bogen er skrevet eller redigeret af enkelte forfattere, hvilket er nævnt i revisions-historien til bogen.

Bogen kan findes i opdateret form på http://www.linuxbog.dk/, mens prøve-udgaver kan findes på http://cvs.linuxbog.dk/.

Figur 1. ÅDL

Kommentarer, ris og ros og specielt fejl og mangler bedes sendt til linuxbog@sslug.dk (mailto:linuxbog@sslug.dk), men er du medlem af SSLUG kan du i stedet for med fordel skrive til sslug-bog@sslug.dk (mailto:sslug-bog@sslug.dk).

3. Oversigt over bogen

- 1. Introduktion til sikkerhed (Kapitel 1).

I dette kapitel ser vi nærmere på, hvordan Linuxverdenen behandler sikkerhedsproblemer. Vi ser også på hvilke typer trusler, der findes, og nærmere på, hvor udsat dit system er for trusler imod sikkerheden.

- 2. Services - at slå services fra og begrænse adgang (Kapitel 2).

  Hvis du har installeret en standard Linuxmaskine, kan der allerede være mange services åbne for dit netværk. Dette kan give en form for adgang til maskinen fra nettet, f.eks. SMTP. Disse udgør en potientiel risiko. På de fleste systemer er det ikke alle disse services, som behøver at være åbne.

- 3. Root access - hvem, hvordan og hvorfor ikke? (Kapitel 3).

  På Linux findes en speciel bruger, "root", som har ubegrænset adgang til hele maskinen, herunder adgang til at ødelægge alt. Vi ser på hvordan og hvornår, du skal anvende root kontoen, nemlig kun når det er nødvendigt. Vi ser desuden på programmer, der kører med root rettigheder.

- 4. Remote login og netværksaflytning (Kapitel 4)

  En Linuxmaskine er født til at kommunikere via netværk. Problemet er bare, at de fleste af de programmer, man anvender, er designet med henblik på stabilitet og ikke sikkerhed. Derfor ser vi bl.a. på problemer med telnet og ftp, og hvordan du kan installere og anvende sikre alternativer såsom ssh (secure shell).

- 5. Systemovervågning, læsning af log-filer (Kapitel 5)

  Hvordan opdager du, hvis der har været nogen inde på Linuxmaskinen og lave slemme ting bag din ryg? Vi ser på måder at kunne læse mange af Linuxsystemets log-filer effektivt, og hvordan du sikrer dig, at systemfiler ikke er ændrede.

- 6. Firewall (Kapitel 6)

  Er dit netværk koblet til internettet, bør du beskytte det med en firewall (eng. firewall). Vi vil se på, hvad en firewall er, hvilke værktøjer du har til rådighed og hvordan firewallen kan sættes op.

4. Hvem er vi?

Forfatterene til denne bog er begge interesserede Linux-brugere. Vi har ikke beskæftiget os specielt med netværkssikkerhed før, men mente, at vi burde vide mere om det. Vi har sat os ind i emnet efter bedste evne i processen med at skrive disse artikler, og vi vil gerne have hjælp og rettelser fra læserne til at gøre

artiklerne endnu bedre. Vi håber derfor, at eventuelle fejl er blevet opdaget og rettet før du læser dette, og at vi på trods af manglende ekspertise og erfaring på området kan beskrive de udvalgte områder korrekt.

Figur 2. Peter Toft

Figur 3. Hanne Munkholm

Alle de emner, der beskrives i disse artikler, er noget vi har fra alment tilgængelige kilder: Bøger, internettet og linuxsystemet selv. Har du kommentarer, så skriv til os begge Hanne Munkholm <hanne@geekgirl.dk> (mailto:hanne@geekgirl.dk) og Peter Toft <pto@sslug.dk> (mailto:pto@sslug.dk).

5. Vi siger tak for hjælpen

Vi har haft stor glæde af mange SSLUG-medlemmers støtte, rettelser og forslag til forbedringer - bliv ved med dette. Specielt vil vi nævne:

- Torben Fjerdingstad for at finde en stribe fejl.
- The people from User Friendly allows us to use a number of their comic strips for the articles. We thank them.
- Jesper Laisen har rettet sprog og kommaer i hele bogen.
- Andre bidragydere er: Claus Nielsen, Rune Christiansen, Donald Axel, Frank Damgaard, Lars Sørensen, Johnny Ernst Nielsen, Henrik Christian Grove, Henrik Lund Kramshøj, Mayank Sarup, Henrik Størner, Tue Wennerberg, Ole Tange, Stefan Andersen, Kristian Vilmann, Ole Michaelsen,

Kristian Støchkel, Jacob Sparre Andersen, Tomas Bertelsen, Rolf Therkildsen, Thomas Hansen, Baldur Kristinsson og Torkil Zachariassen

Du kan i Appendiks A finde en liste over alle de revisioner, som bogen har været igennem.

6. Typografi

Vi vil afslutte indledningen med at nævne den anvendte typografi.

- Navne på filer og kataloger skrevet som `foo.bar`
- Kommandoer, du udfører ved at taste, skrives som **help**
- Der er flere steder i bogen, hvor vi viser, hvad brugeren taster, og hvad Linux svarer. Det vil se ud som:

  ```
  [daisy@hven daisy]$ Det her taster brugeren
  Det her svarer Linux
  ```

- Der er tilsvarende flere steder i bogen hvor vi viser hvad systemadministratoren (root) taster, og hvad Linux svarer. Det vil se ud som:

  ```
  hven# Dette taster systemadministratoren
  Dette svarer Linux.
  ```

  Det vigtige her er at kommandofortolkeren bruger nummertegnet (#) til at markere at man har systemadministratorrettigheder.

Kapitel 1. Introduktion til sikkerhed

1.1. Hvorfor er netværk usikkert?

Din maskine har ofte en del netværksservices åbne på forskellige "porte", når du har adgang til internettet. En port er ikke en fysisk port, men en logisk indgang til din maskine via netværksenheden (modem eller netværkskort). Disse "åbne porte" benyttes til at udføre netværkskommunikation, hvis maskinen tilbyder netværksservices som f.eks. ftp. Men de kan også misbruges, så uautoriserede personer kan komme ind på din computer via nettet. Dette kan ske, hvis der er fejl i netværksprogrammer (services, firewall og styresystem), eller hvis du lader dem stå helt åbne uden nogen form for adgangskontrol. Det kan også ske selvom du har adgangkontrol, hvis folk har mulighed for at opsnappe dit brugernavn og password via nettet. Ofte kan man direkte samle brugernavne og passwords op på nettet ved at bruge et "sniffer" program, som lytter til trafikken på et givent sted.

Dette er bare en forsmag på de emner, vi vil komme nærmere ind på i de kommende kapitler. Se også http://securityfocus.com/infocus/1539 om samme emne.

1.1.1. Er Linux et usikkert system?

Nogen har anklaget Linux for at være et meget usikkert system, idet alle kan læse kildeteksten. Når man kan læse kildeteksten til f.eks. netværksprogrammer og se, hvordan de fungerer, er det nemt finde eventuelle fejl. Disse kan bruges til at lave angreb på Linux-maskiner. Dette er delvis sandt! *Men* alligevel ikke. Vi ser på hvorfor.

For at knække sikkerheden på et UNIX-system, skal man enten *kende* en fejl eller *lede efter* fejl i systemet herunder systemopsætning og netværksprogrammer. For en kommerciel leverandør er en fejl ofte et prestigetab, og derfor er det måske ikke alle fejl, som bliver offentligt kendt. Hvis ingen kendte de fejl, der er i de lukkede systemer, ville alt være godt. Men ofte bliver fejl i lukkede systemer opdaget og kendt. Det kan skyldes at nogen udefra opdager fejlen ved et tilfælde, eller måske leder de bevidst efter fejl - eller at nogen indefra lækker information eller taler over sig. Så ender fejlen måske på en hjemmeside eller som en artikel i en avis. Så har du problemet: Hvis den, der vil bryde ind, kender fejlen, og du som systemadministrator ikke kender og tager højde for den, kan han komme ind på dit system.

Det er nok nærmest umuligt at gardere sig imod at en sikkerhedsbrist bliver offentligt kendt.

Ud over kendte sikkerhedshuller kan den, der vil bryde ind, selv lede efter fejl. I et lukket system er det langt sværere at lede efter fejl end i åbent system som Linux, fordi man ikke har kildeteksten. Men en måde at finde sikkerhedsfejl på, er at prøve kendte sikkerhedshuller af på nye maskinopsætninger - men med ændringer svarende til nye ideer. Nogle gange sker det så, at der er bid. Et system er ikke sikret imod indbrud, bare fordi kildeteksten er lukket.

Erfaring viser, at nogle af de helt store firmaer er meget ringe til at oplyse om sikkerhedsfejl - de skal helst negligeres eller skjules. Sikkerhedsbristerne bliver ofte hurtigt annonceret på internettet, mens firmaerne tit bruger lang tid på at lukke sikkerhedshullet ordentligt. I al den tid fra annoncering til rettelse er maskinerne åbne for mulige sikkerhedsangreb.

I Open Source-verdenen vender man alle disse problemstillinger på hovedet. For det første findes al relevant kode på internettet til offentlig beskuelse. Så nok er der mennesker, som kan læse og bryde sikkerheden for andre, men Open Source-verdenen er indrettet, så der er en meget stor prestige i at finde og specielt lappe sikkerhedshuller. Derfor vil folk, der finder fejl, oftest fortælle offentligt om de fundne fejl fremfor at gemme informationen. Det kan selvfølgelig også være, at man i stedet vælger at udnytte informationen til at bryde ind på andres systemer. Så kan det være, at det er et af de første "ofre", som rapporterer fejlen. Man vil så fokusere på hvilken service, der gav sikkerhedshullet, som man så lukker (eller giver speciel opmærksomhed gennem overvågning, hvis man ikke kan lukke servicen f.eks. webserveren for en internet-butik). I Open Source-verdenen indleder man så en klapjagt for at få fjernet problemet og udgive nye versioner af de berørte programmer. Det bliver en slags "trofæ" at komme først med den korrekte rettelse. Normalt er man nede på dage og ofte timer, fra en sikkerhedsfejl er rapporteret, til den er fundet og rettet, hvilket er markant mindre end på nogen af de andre lukkede systemer.

Alt dette lyder meget rosenrødt, men hvis din Linux-maskine har en kendt sikkerhedsbrist, og maskinen er på et usikkert netværk såsom internettet, så er der er en pris at betale. Du *skal* opdatere din maskine, når der er fundet sikkerhedsbrister, og den tilsvarende sikkerhedsrettelse er offentliggjort. Ellers har du reelt set nul sikkerhed! Alle kan jo se hvad, der er galt i din kode - det har jo været diskuteret på internettet, så man har kunnet lave rettelsen. På rootshell på freebsd.org (http://ftp2.de.freebsd.org/pub/misc/www.rootshell.com/) er der en lang række henvisninger til artikler og software. Det er ofte beskrivelse af problemets omfang og karakter og anvisninger på udnyttelse og reparation.

Det typiske forløb for en systemadministrator i forbindelse med opdagelsen af et sikkerhedsbrist vil være:

- Sikkerhedsbristen bliver annonceret
- Man lukker for den service, som bristen tilhører. Hvis det ikke er muligt iværksættes ekstra overvågning af servicen samt de filer, som kan påvirkes af sikkerhedsbristen.
- Annoncering af rettelse.
- Installation af rettelsen.
- Starte servicen igen eller stoppe den ekstra overvågning.

1.1.2. Hvilke angreb kan jeg blive udsat for

Der er flere forskellige typer angreb, der kan ramme din maskine. Der er forskellige metoder, en angriber kan benytte sig af, og som man kan gøre noget for at beskytte sig imod. Der er også forskel på, hvilken type adgang en angriber får til dit system, og hvor alvorlige konsekvenser et vellykket angreb har for dig.

Lad os først se på, hvordan en angriber kan bære sig ad med at få adgang til dit system.

- *Exploits* Det mest almindelige er at gå efter nogle netværksservice-programmer, der er fundet fejl i. Disse fejl udnyttes ofte for at få fuld adgang til maskinen (Se nedenfor og Kapitel 3). Som tidligere nævnt skal man holde øje med annoncerede sikkerhedsfejl, og få opdateret de anfægtede programmer. Vi vil i Kapitel 2 se på hvordan du minimerer disse problemer.
- *Password angreb* Den klassiske måde at komme ind på et system på er at få fat i et brugernavn og password (helst root passwordet).
  - *Packet sniffing* Man kan opsnappe brugernavne og passwords, hvis disse sendes i klar, ukrypteret form over nettet - se "man-in-the-middle angreb" herunder. Vi kommer desuden tilbage til emnet i Kapitel 4.
  - *Brute force* Man kan knække et svagt password med rå computerkraft. Se Kapitel 3.
  - *Social engineering angreb* Ofte kan en angriber slutte sig til ting ud fra sin viden om menneskelig adfærd. F.eks. at en bruger vil benytte det samme passwords til mange ting for at slippe for at huske så mange passwords. Så har man et password fra en telnet session, man har overvåget, kan det sikkert bruges i andre henseender.
- *Man-in-the-middle angreb* Her er tale om at angriberen befinder sig et sted på netværket, hvor dine data-pakker kommer forbi. F.eks. hos din internet-udbyder eller på dit lokalnetværk. Eller i princippet et vilkårligt sted på internettet, hvor dine pakker kommer forbi.
-
  - *Packet sniffing* Det mest almindelige man-in-the-middle angreb er, at angriberen kigger på dine data-pakker for at opsnappe information. Det kan f.eks.være brugernavne og passwords, som sendes i klar tekst over nettet, eller dine firmahemmeligheder, hvis du sender dem afsted ukrypteret.
  - *Session hijacking* En anden type man-in-the-middle angreb er, at en angriber går ind og overtager en igangværende session f.eks. en telnet-session. Dette er i praksis meget vanskeligt på internettet, da det enten kræver at target-maskinen bruger forudsigelige tcp-sekvensnumre, eller at man kan aflytte trafikken. Det første er ikke muligt i Linux, og hvis man kan det andet, er det meget nemmere blot at sniffe brugernavn og password.
- *IP spoofing* IP Spoofing går ud på at få systemet til at tro, at angriberens computer er en autoriseret computer, der har lovlig adgang. Dvs. at angriberens computer udgiver sig for at være en autoriseret host ved at bruge dennes IP-adresse. Dette kan bruges til at indsætte angriberens data i en datastrøm. Hvis angriberen ønsker at opnå tovejskommunikation, hvor han modtager de pakker, der er tiltænkt den, han udgiver sig for, er det sværere, idet han også skal ind og ændre routing-tabellerne. Uanset dette, så kan IP spoofing gøre analyse af angreb til en umulig opgave.
- *Trojanske heste* En "trojansk hest" er et program, som indeholder skjult funktionalitet. Et velkendt program på dit system kan se helt uskyldigt ud og udføre sin normale funktion. Programmet kan imidlertid være modificeret, så det ud over sine normale opgaver f.eks. sender navne og passwords til en fast modtager eller lader ukendte personer logge ind på maskinen uden, at dette skrives i systemets logfiler. "Trojanske heste" kan være programmer efterladt af en angriber, der har været inde på dit system. De kan skjule hans spor og lette hans adgang næste gang via bagdøre (*back-doors*). Det kan imidlertid også være programmer, du selv har hentet ned og installeret. Du tror, du installerer et almindeligt program, men i virkeligheden er det en modificeret udgave, du har fået fat i. Heldigvis medfører traditionen med Open Source (http://www.opensource.org) programmer, at vi har adgang til

kildeteksten, og at man i praksis ikke kan skjule disse problemer for os. Henter du binære programmer ned (såsom RPM-pakker), så bør dette kun ske fra betroede steder, hvor du kan regne med at der bliver holdt styr på sikkerheden - men det er altid en risiko, du må løbe. Ydermere findes der PGP-nøglesignaturer i alle RPM-pakker, som man kan bruge til at kontrollere, hvem der har lavet den enkelte pakke. Heldigvis er problemerne i Open Source-verdenen meget små i forhold til det niveau, som Windows-verdenen er udsat for med virus i programmer og selv i Word-dokumenter. Et velkendt eksempel på en trojansk hest fra Windows verdenen er "Back Orifice". Hvis det først er installeret, giver det folk udefra fuld kontrol over den inficerede maskine via internettet.

Vi har nu set på forskellige metoder, en angriber kan bruge til at få adgang til dit system. Men hvad kan han bruge det til? Lad os dele angrebene op efter resultatet - hvad opnår vedkommende, og hvor farligt er det for dig.

- *Læseadgang* Angriberen kan stjæle (kopiere) dine data. Dette er alvorligt, hvis du har hemmelige data, som kan udnyttes af andre på en måde, der skader dig. Læseadgang kan f.eks. opnås ved packet sniffing - hvis der er værdifuld information i pakkerne. Man kan beskytte sig ved at kryptere sine data. Læseadgang kan også opnås ved IP spoofing, hvis angriberen f.eks. giver din maskine besked om at videresende et vigtigt brev til ham. Endelig kan det opnås, hvis angriberen opnår login på dit system f.eks. ved at have skaffet sig et brugernavn og tilhørende password. Har man login på maskinen, kan man naturligvis også skrive men kun i de filer, der er adgang til for den bruger, man er logget ind som. Hvis man er logget ind som root, er det i praksis alle filer.
- *Skriveadgang* Ud over at dine data kan blive stjålet, kan du risikere, at de er ændret, og at dit system er modificeret. Dette er meget alvorligt. Dit system er kompromitteret. Er dette tilfældet, skal man være *ekstremt* forsigtig med systemet, da man typiske ikke ved hvor meget som er ændret.

  Skriveadgang kan opnås, hvis angriberen opnår login på systemet. Han kan modificere de data, som det anvendte brugernavn har adgang til. Hvis han er inde som root, har han fuld adgang til at ødelægge eller modificere dit system. Root-adgang kan opnås ved opsnappe eller knække root passwordet eller ved at udnytte et sikkerhedshul i en netværksservice f.eks. Sendmail. En angriber kan også opnå skriveadgang til dit system ved IP spoofing - hvis man kan ændre indholdet af pakker, der bliver sendt over nettet, uden at modtageren opdager det, så har man effektivt forfalskede data. Det kunne også være kommandoer til systemet. Har en angriber først haft skriveadgang til dit system, skal du være meget forsigtig. Du ved ikke hvilke data, der er ændret - måske er de programmer, du kontrollerer dit system med, selv modificeret, og du kan ikke regne med dem længere. Du ved måske ikke med sikkerhed, hvornår han første gang har været inde, og dine backups kan være inficeret et stykke tid tilbage. Vi kommer tilbage til, hvad du skal gøre for at opdage et indbrud i Kapitel 5. Har en angriber lokket dig til at installere en trojansk hest, er det også en slags skriveadgang - dit system er modificeret. Den eneste sikre kur er geninstallation af hele systemet.

1.1.2.1. Remote og local exploit

Når der findes en fejl i et program kan der ofte laves et lille program der demonstrerer effekten af fejlen. Disse programmer kaldes ofte for proof-of-concept kode. Crackeren kan bruge samme metode til at lave et exploit program, d.v.s. et program, som udnytter fejlen, sårbarheden.

Disse exploit programmer kan fungere via netværk eller afvikles lokalt. Ideen med lokale programmer er typisk at få flere rettigheder på systemet (privilege escalation). Derfor skal programmer som Apache webserveren afvikles med så få privilegier som muligt. Så skal der være to sårbarheder for at få magt over maskinen, - en for at få adgang til maskinen, og en lokal sårbarhed, som kan give root privilegier.

1.1.2.2. Denial of Service angreb

Ude-af-drift angreb (eng. Denial of Service - DoS) har til formål at få din maskine til at holde op med gøre det, den er sat til. Crackeren prøver at forhindre maskinen i at yde den service den skal ved at overbelaste nettet eller maskinens netsoftware. Ofte medfører den slags overbelastning at systemerne crasher. Dette er knap så alvorligt som et reelt indbrud på systemet, men kan alligevel godt koste meget tid og mange penge. Eksempler på DoS angreb er *Pentium F00F-fejlen*, *ping of death* og *teardrop*, som får maskinen til at låse. Disse fejl er rettet i nyere Linux-systemer.

Man kan ikke lokalt i en virksomhed beskytte sig mod almindelig, lovlig, men massivt overbelastende og unyttig traffik til en bestemt server og man ser derfor virksomheder, som opsætter alternative web-sites med nye IP-numre/navne som kan overtage arbejdet mens der køres DoS mod virksomhederne. Andet er der ikke at gøre.

1.1.2.3. Syn flooding og Ping flooding

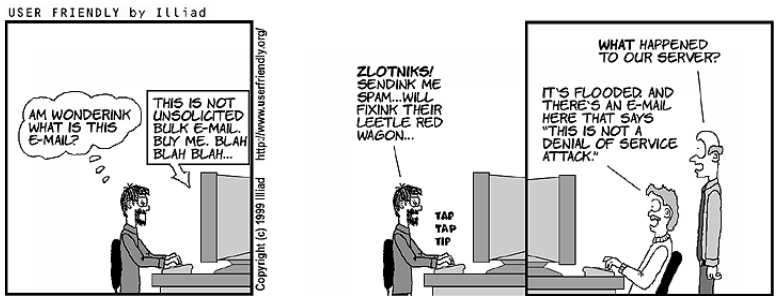
Ved *SYN flooding* og *Ping flooding* udsættes maskinen for en voldsom belastning fra en eller flere angribere med at svare på netværksforespørglser, og det er ikke muligt at bruge andre netværksservices på grund af overbelastning.

1.1.2.4. Smurfing

Ved *smurfing* er det netværket, der overbelastes. Disse problemer kan der dæmmes op for ved fornuftig router-opsætning. Der er også angreb, hvor der uploades meget store mængder data til en anonym ftp konto, hvor formålet er at få diske til at løbe fulde. Disse problemer kan der dæmmes op for med disk-kvoter og/eller smart partitionering (eller at upload af filer disables).

Fra den mere muntre ende kan vi lige vise lidt om DoS angreb fra User Friendly strippen den 7/8 99 http://www.userfriendly.org/cartoons/archives/99aug/19990807.html.

Figur 1-1. User Friendly



1.2. Har mit system sikkerhedsproblemer?

Nu skal vi se på, hvor meget du udsætter din maskine for, når du tilslutter den til internettet. Vi tager udgangspunkt i en Linux-maskine, som har en netværksopkobling enten via modem eller fast forbindelse. Det er reelt ikke så vigtigt, at det er en Linux-maskine - alle systemer med netværk har samme karakteristiske træk.

Lad os inddele verden efter

- "On and off" : Opkobling sker med modem og f.eks. PPP. Der hentes post og surfes minimalt på internettet.

  Du har ikke så meget at frygte, selvom du måske ikke har styr på din sikkerhed på maskinen. Når du laver en modem-opkobling til internettet, får du tildelt en IP-adresse, dvs. en adresse, som alle kan tilgå din maskine på. Men den adresse er forskellig fra gang til gang når du laver opkobling. Derfor bliver det i praksis umuligt at afsøge din maskines sikkerhedshuller fra en anden maskine, før du igen har lukket forbindelsen. Der er dog oplagt, at den risiko, du løber, svarer nøje til den tid du er koblet på internettet.

- "Seriøs surfing": Opkobling sker med modem og f.eks. PPP. Der hentes post jævnligt, og der surfes en del. Denne situation kan også gælde for en maskine, som via automatiske services laver internetopkobling for et helt lokalnet. Til Linux er det ofte "diald", som bruges til dette.

  Du er tilkoblet internettet i længere tid ad gangen men stadig med forskellig IP-adresse hver gang, så du er ikke helt nem at finde. Men i modsætning til "On and off" brugeren er du på så lang tid, at der godt kan laves skanning på din maskine. Hvis der findes en usikker service, kan den måske give oplysninger om svagheder i dit system, og dermed kan din netværkssikkerhed rammes. Du bør tænke

på hvilke åbninger, der er i dit system. Du bør lære noget om grundlæggende netværkssikkerhed. Læs f.eks. resten af artiklerne i denne serie.

- "Fast opkobling": Maskinen har fast opkobling til internettet via ethernet-, ISDN-, ADSL- eller anden forbindelse.

  Du kan skannes fra en enhver maskine på internettet medmindre der er firewalls eller såkaldte proxy-servere imellem. Du kan ikke overvåge alt 24 timer i døgnet. Maskinen efterlades. Du må tjekke dine log filer! Slå overflødige netværksservices fra og begræns adgangen til dem, der er tilbage. Du bør lære noget om sikkerhed. Læs f.eks. som en start de kommende artikler i denne serie. På internettet er der masser af yderligere information, og der findes gode bøger om emnet.

Lad os antage, at din maskine har direkte adgang til internettet uden firewall eller anden beskyttelse. Vi vil nu se på, hvorfor din maskine kan være udsat, og hvordan andre finder ud af, at du har en maskine på nettet. At det netop er din maskine, som er udset til et angreb, kan skyldes mange ting: Tilfældigheder, at du anses for at ligge inde med interessante data, eller at du ikke har vedligeholdt din Linux-maskine og fået lukket de kendte sikkerhedshuller.

En angriber har flere måder at finde din maskine på. En er f.eks. at studere brevhovederne i dine breve. En anden er at hente listen over hostnavnene i dit domæne vha. DNS (navneservere). En mere direkte måde er at bruge ping.

En angriber kan derefter spørge din maskine, hvilke services den tilbyder, og måske finde et sikkerhedshul. Som eksempel kan vi tage ssh, som tilbydes på port 22. Med telnet kan man logge ind på port 22 og få versionsnummeret på ssh. I det følgende eksempel finder vi ud af, at det er ssh version 1.2.27, der kører. Kan angriberen f.eks. på internettet finde en beskrivelse af en sikkerhedsfejl i den version af ssh, kan han udnytte dette til at angribe dit system med.

```
[robin@sherwood robin]$ telnet bohr 22
Trying 172.17.0.3...
Connected to bohr
Escape character is '^]'.
SSH-1.5-1.2.27
```

Normalt er det nødvendigt, at din maskine har nogle af de mange netværksservices kørende, da de bruges til at kommunikere med andre maskiner via netværk. Ofte kan de fleste af dem dog slås fra, se Kapitel 2.

I stedet for at en angriber manuelt skal gennemse alle dine porte for mulige huller, kan han gøre det nemmere og meget hurtigere ved at installere programmer såsom nmap. Nmap kan hentes på http://www.insecure.org/nmap. Med nmap kan man dels se hvilke maskiner, der er i live, men også hvilke porte, der er åbne. Nedenstående eksempel viser, at maskinen "bohr" har (alt for) mange porte åbne.

```
[robin@sherwood robin]$ nmap bohr
Starting nmap V. 2.12 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
```

```
Interesting ports on bohr.herne.dk (172.17.0.3):
Port    State       Protocol  Service
21      open        tcp        ftp
22      open        tcp        ssh
23      open        tcp        telnet
25      open        tcp        smtp
79      open        tcp        finger
80      open        tcp        http
98      open        tcp        linuxconf
111     open        tcp        sunrpc
113     open        tcp        auth
513     open        tcp        login
514     open        tcp        shell
515     open        tcp        printer
6000    open        tcp        X11

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

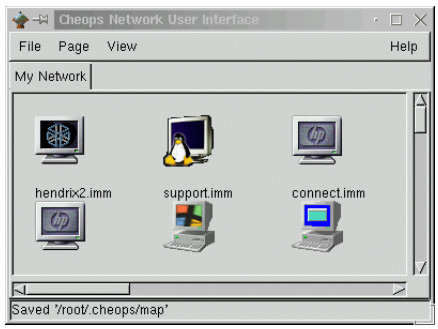
Eksemplet med nmap svarer til, hvad man udefra kan se om maskinen. Har man adgang til maskinen, så kan "netstat -a" også være interessant, idet kommandoen viser alle de netværksforbindelser, som er etableret, samt de ventende serverprogrammer. Et forkortet output af "netstat -a" kan være følgende, hvor man kan se, at en telnet-session fra sherwood til bohr er igang, og serverprogrammerne ssh, sendmail (smtp), telnet og ftp er klar.

```
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address   Foreign Address         State
tcp        0      0 bohr:telnet sherwood.herne.dk:1074 ESTABLISHED
tcp        0      0 *:ssh           *:*                   LISTEN
tcp        0      0 *:smtp          *:*                   LISTEN
tcp        0      0 *:telnet        *:*                   LISTEN
tcp        0      0 *:ftp           *:*                   LISTEN
```

Du kan læse mere om NMAP på http://www.itworld.com/Sec/2202/LWD010404vcontrol1/ og http://www.insecure.org/nmap/nmap_doc.html.

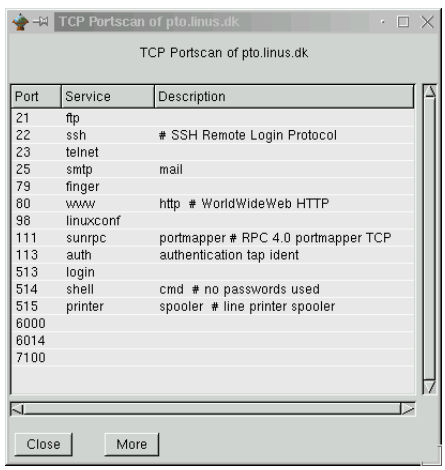
Man kan også have interesse i at bruge det grafiske program cheops, som findes på ftp://ftp.marko.net/pub/cheops/RPMS/. Programmet er skrevet sådan, at det grafisk viser hvilke maskiner, som er i live, og evt. overvåger, at de forbliver i live. Cheops er nok primært beregnet til at give systemadministratoren et hurtigt overblik over netværket, men det kan også bruges til at udse sig svage maskiner, som kan angribes. På følgende billede er vist, hvordan cheops ud fra et domænenavn selv finder ud af, at der er seks maskiner i live. Det vises grafisk hvilket operativsystem, der anvendes på hver af dem. Fra venstre mod højre: SGI IRIX, Linux, HP/UX, HP/UX, Windows og endelig en speciel maskine.

Figur 1-2. Cheops



Flytter man musen ned på ikonet for maskinen, får man flere oplysninger om den ofte også versionsnumre, og man kan yderligere direkte skanne maskinen for alle oplysninger. Man kan ofte se alt for meget - især når folk ikke har deres sikkerhed i orden.

Figur 1-3. Cheops



Det er altså nemt at trække informationer ud af en ubeskyttet maskine.

1.3. Hvordan sikrer jeg min maskine?

Hold din maskine ajour med sikkerhedsrettelser. Du kan følge med på diverse postlister/nyhedsgrupper om sikkerhed, og websider - se mere i Afsnit 1.4. Ofte kan man hurtigt efter en fejl er opdaget, hente

rettelsen fra internettet.

Mange peger på Linux, som et system der er meget tidskrævende at holde opdateret, fordi der kommer så mange opdateringer til Linux og de services, man ofte kører. Men mange af disse opdateringer er netop rettelser, som lukker sikkerhedshuller. Samme sikkerhedshuller findes ofte også til andre systemer, men de bliver ikke lukket ligeså hurtigt, hvis de overhovedet bliver det. Den ekstra tid du skal bruge på sikkerhedsopdatering er altså blot en sikring af dit system, som ikke er muligt i samme omfang for mange andre systemer. Desuden er det også lettere at finde sikkerhedsopdateringer til Linux og andre Open Source-systemer, da disse ganske bramfrit annonceres på internettet.

Når du henter en sikkerhedsopdatering, så tænk lige på om den kommer fra et sted, du stoler på - den kunne i princippet være falsk. Det er den sikkert ikke, men nøjes alligevel med at hente fra "officielle" steder. Den første lektie, du skal lære, hvis du vil have et sikkert computersystem, er at være en lille smule paranoid. :-) Der er intet på nettet, der er helt sikkert.

1.3.1. Honeypot - lær at forsvare dig før det koster

På http://www.honeynet.org/ forfølges en interessant tankegang. Man sætter en "honeypot", dvs. en krukke med lækker honning, op, og derefter studeres de bier som går i lag med honningen. I netværks-sammenhæng opsætter man en computer, som ikke har nogen data af værdi, men med det setup man har tænkt sig at køre sidenhen. I en periode lader man crackere angribe maskinen og holder meget øje med den, så man kan se, hvad der sker. Erfaringerne deles og diskuteres gerne med andre, og erfaringerne bruges til at lave endnu mere sikre maskiner.

1.4. Referencer

Gode steder at starte

- Linux Administrator’s Security Guide (LASG) by Kurt Seifried. Denne bog er guld værd - og kan findes på http://seifried.org/lasg/.
- Linux Security HOWTO - http://sunsite.auc.dk/ldp/HOWTO/Security-HOWTO.html

Annonceringer af fejl i flere af de kendte Linux-distributioner:

- *Debian* http://www.debian.org/security/ Lister sikkerhedsrelaterede fejl i Debian.
- *SuSE* http://www.suse.de/security/index.html har liste over sikkerhedsfejl i SuSE.

- *Red Hat* http://www.redhat.com/support/errata Generel indgang til fejllisterne for Red Hats produkter.

Steder som ellers bør eller kan følges

- Cert.org (http://www.cert.org/) er et center for internet sikkerheds-ekspertise, som drives af Carnegie Mellon universitetet. Deres alarm (eller alert) arkiv (http://www.us-cert.gov/cas/techalerts/index.html) (http://www.us-cert.gov/cas/techalerts/index.html) viser nyeste såvel som gamle sikkerhedshuller i gængse systemer.
- Bugtraq http://www.securityfocus.com bør du følge med på for at læse om de nyeste afsløringer af sikkerhedsbrister (exploits).
- Root Shell http://www.rootshell.com (http://ftp2.de.freebsd.org/pub/misc/www.rootshell.com/) har alt indenfor diskussion af sikkerhed.
- Linux Today http://linuxtoday.com bringer også sikkerhedsbrister frem sammen med andre nyheder. Det er muligt at tilpasse hvilke nyheder man får f.eks. nyheder om sikkerhed.
- http://www.securityportal.com/ er et andet interessant sted for folk, som interesserer sig for sikkerhed.
- Linux Security WWW - http://www.aoy.net/Linux/Security, med mange Linux-relaterede sikkerhedsannonceringer, FAQs og links.
- Linux Security Home Page - http://www.ecst.csuchico.edu/~jtmurphy/
- Reptile's Linux Security Page - http://www.reptile.net/linux
- Infilsec Vulnerability Engine - http://www.infilsec.com/vulnerabilities/ - generelt om sikkerhed.
- Munitions - http://munitions.polkaroo.net/ - stor samling af viden og programmer om kryptering af data og netværkstrafik til Linux.

Kapitel 2. Netværkssikkerhed - Services

2.1. TCP/IP-services

En Linux-maskine er bygget til netværksbrug. Hvis der ikke er noget netværk, laver den bare et for sig selv. Uanset om din Linux-maskine er på et netværk eller ej, tilbyder den en række netværkstjenester.

Er din computer koblet til netværk, er den udsat for misbrug udefra. Derfor er det vigtigt, at du forstår, hvilke services din maskine tilbyder, hvad disse services gør, hvem de tilbydes til, og at du i øvrigt har så få services kørende som muligt. Services, du ikke bruger, bør slås fra, idet de kan give andre adgang til maskinen. Hvis din Linux-maskine ikke er på et netværk, kan du egentlig være ligeglad med, hvilke services den tilbyder. Og dog - imorgen kommer den måske på lokalnet, og i overmorgen bliver lokalnettet sluttet til internettet. Denne kendsgerning kommer bag på de fleste, og det er en god idé at gøre sig nogle sikkerhedsovervejelser på forhånd. Desuden har IBM foretaget en undersøgelse der viser, at over 40% af alle sikkerhedsangreb sker internt, hvilket også skal med i overvejelserne ved sikring af Linux-maskine.

Hvad er en service? Det er et program, der kører på din computer, som er beregnet til at andre kan koble sig til din computer og udveksle data. Det kan f.eks. være ftp, telnet, finger, pop-2, pop-3, rpc (herunder NFS), nntp og talk. Det kan også være http, smtp eller linuxconf.

Det skal nævnes at man selv kan komme til at tilbyde en "exploit", ved at skrive hjemmesider, der anvender kode, som køres på f.eks. egen web-server, såsom CGI-scripts og PHP-kode. Vær særdeles varsom med dette, og i særdeleshed, hvis man har brugere på maskinen der frit kan opdatere web-server scripts.

2.2. Netværksprotokoller og porte

Før vi ser nærmere på, hvad Linux-maskinen kan servicere for et netværk, så lad os lige træde et skridt tilbage og se på, hvordan systemet er bygget op.

En Linux-maskine vil normalt foretage en masse ting samtidig, f.eks. håndtere e-post, login-brugere som kører programmer, samt have en web-server kørende. Derfor har man ikke bare kunnet nøjes med at hver enkelt computer har en IP-adresse. De forskellige services lytter på hver sin port, som har sit eget nummer. F.eks. bruges port 21 til ftp, port 22 til ssh (secure shell), port 23 til telnet, port 25 til smtp og port 80 til http. De mest kendte (well-known) portnumre, er tildelt af organisationen IANA (Internet Assigned Numbers Authority). I TCP/IP protokollen er det i TCP-datapakkerne, man finder information om afsender port og modtager port, imens IP-adressen på afsender og modtager er indeholdt i IP-transportlaget.

Porte er ikke fysiske men logiske konstruktioner, som anvendes i den måde, man sender data over netværket. Man skelner imellem TCP- og UDP-porte, som bruges til to forskellige typer dataoverførsler. TCP er forbindelsesorienteret. Den sender hele tiden kvitteringer frem og tilbage (handshakes), og når det går godt, behøver den ikke at få kvittering for sine data så tit. Hvis en datapakke går tabt, vil den blive gentransmitteret af protokollen. Der findes også en anden type IP-pakker med betegnelsen UDP. UDP anvendes oftest til hurtig letvægtsinformation, som kan accepteres at gå tabt i netværket såsom f.eks. NFS-forespørgsler. UDP-pakker, der går tabt vil kun blive retransmitteret, hvis applikationen sørger for det (typisk NFS).

Ofte tildeles en service både TCP og UDP porten med et givet nummer, selvom den kun bruger den ene.

User Friendly http://www.userfriendly.org/static ser på IP-pakker...

Figur 2-1. User Friendly

Porte vælges ikke tilfældigt, men mange er standardiserede, så maskiner kan koble sig til en given port og dermed vide hvilken service, der kan findes. Hvis du læser filen `/etc/services` på en vilkårlig UNIX-maskine såsom Linux, kan du se hvilke porte, der bruges til hvad. Udsnit af en `/etc/services`:

```
ftp-data        20/tcp
ftp             21/tcp
telnet          23/tcp
smtp            25/tcp          mail
```

Først står navnet på servicen, derefter portnummeret og om det er tcp eller udp. Det sidste felt er til et evt. alias for denne service - i eksemplet er `mail` et alias for `smtp`.

Portnumre under 1024 kaldes priviligerede porte, og man skal være root for at kunne lytte på dem. Dette er lavet, så en klient, der kontakter en fremmed maskine på en port under 1024, kan regne med at få fat i en standard service og ikke et eller andet tilfældigt bruger-program. Se **man services**.

IP (Internet Protocol) er den netværksprotokol, man bruger på internet-baserede netværk. Der findes mange andre. - Apple har deres egen AppleTalk-protokol, og Novell har IPX/SPX. UNIX-maskiner

bruger normalt IP, men har historisk også gjort brug af andre protokoller, f.eks. Digital's DECNET og Regnecentralens IMC. Linux-maskiner kan forstå mange af disse andre protokoller.

2.2.1. Daemons

En service kan enten køres som en selvstændig daemon, eller den kan styres af inetd. En daemon er en proces, som kører i baggrunden hele tiden, og venter på forespørgsler. Nogle services skal køre hele tiden og altid være klar, fordi det er nødvendigt med en hurtig responstid, eller at de ikke må være nede. Et eksempel er named, som laver navneoplag via DNS, og som skal være hurtigt. Andre eksempler på daemons er lpd (printer daemon), syslogd (logger system events) og sendmail. Vi kigger nærmere på sendmail i et senere afsnit. En anden daemon, som vi skal se på om lidt, er inetd, der bruges til at starte andre services efter behov. Prøv at køre kommandoen

```
[robin@sherwood robin]$ ps aux | more
```

Denne kommando viser alle de kørende processer på systemet, og du vil kunne genkende processer som lpd, syslogd og inetd.

2.2.2. Inetd

En del services har kun brug for at køre relativt sjældent. Disse services er i Linux styret af inetd-programmet. I stedet for at have f.eks. in.telnetd kørende hele tiden, sættes inetd daemonen til at lytte efter, om der er telnet forespørgsler, og starter in.telnetd efter behov.

Hvilke services, der skal startes, bestemmes i kataloget `/etc/xinetd.d`, og de filer der ligger der. Eks.:

En service kan slås fra ved at udkommentere den linje, der starter den pågældende service. Eks. er `telnet`

```
# default: on
# description: The telnet server serves telnet sessions; it uses \
#       unencrypted username/password pairs for authentication.
service telnet
{
        flags           = REUSE
        socket_type     = stream
        wait            = no
        user            = root
        server          = /usr/sbin/in.telnetd
        log_on_failure  += USERID
        disable         = yes
}
```

Det du skal se efter er om der står `disable=yes` i hver fil, for dermed startes de enkelte services *ikke*.

For at ændringerne i `/etc/xinetd.d` skal træde i kraft, er det nødvendigt at genstarte xinetd processen. Dette kan gøres på flere måder. Nogle distributioner har en struktur med startup scripts, der bruges til dette formål. Under Red Hat og Debian ligger de under `/etc/rc.d/init.d` hhv. `/etc/init.d`. Så kan man bruge startup scriptet til at genstarte inetd med.

Red Hat:

```
[root@sherwood robin]# /etc/init.d/xinetd reload
```

Der er dog også nogle Linux-distributioner, der ikke benytter denne struktur herunder Slackware, så vi har brug for en mere generel løsning. For en vilkårlig distribution kan man genstarte xinetd ved at skrive:

```
[root@sherwood robin]# killall -1 xinetd
```

Før Red Hat 7 anvendte man i stedet for **xinetd** forgængeren **inetd** og der skal man udkommentere linjerne i `/etc/inetd.conf`

Bemærk, at du i eksemplet kun slår din ftp-*server* fra. Du kan stadig bruge ftp fra maskinen til andre maskiner. Man kan bare ikke længere bruge ftp udefra og ind til din maskine.

Inetd har den sikring, at en service bliver lukket ned i 10 minutter, hvis der kommer mere end 20 forespørgsler i sekundet. For nogle services er dette ikke acceptabelt. Det er derfor, programmer som named og syslogd kører udenom inetd som daemons.

Gennemgå de andre services i `/etc/xinetd.d` på samme måde som vist ovenfor. Undersøg, hvad der kører på din maskine, og find ud af om det er nødvendigt at de enkelte services kører. Hvis ikke, så slå det fra. En gylden regel er, at du hellere må køre for få end for mange services. Vi kommer senere til en gennemgang af de services, der findes i `/etc/xinetd.d`.

Husk slutteligt, at hvis du ender med at du slet ikke vil køre services via `/etc/xinetd.d`, så kan du lige så godt lukke selve xinetd programmet ned.

2.2.3. TCP-wrappere

Adgangen til de services, der styres af inetd, kan på en nem måde konfigureres, så der er kontrol over hvem, der har adgang til at bruge de forskellige services, maskinen tilbyder.

Dette gøres ved at anvende en TCP-wrapper (tcpd), som er en agent, der bliver kørt i stedet for din normale server applikation. Den sørger for at requests om service bliver logget til syslogd, og den giver en stærk og fleksibel adgangskontrol. Man kan f.eks. også få den til at sende sig et e-brev, hvis nogen udefra forsøger at få adgang via telnet til maskinen.

2.2.4. Adgang til din maskine

Adgangskontrol sker via filerne `/etc/hosts.allow` og `/etc/hosts.deny`.

Når tcpd skal finde ud af, om der er adgang til en service, tjekker den først `/etc/hosts.allow`. Hvis der her er givet explicit adgang, godkendes forsøget, og servicen bliver startet. Hvis der ikke står noget om den pågældene service i `/etc/hosts.allow`, læses nu `/etc/hosts.deny`. Hvis `/etc/hosts.deny` ikke indeholder noget, der forbyder den forsøgte adgang, vil forsøget blive godkendt. Det kan altså være en god idé at sætte sin `/etc/hosts.deny` op til at slutte med at afvise alt og alle. På den måde er man sikker på, at hvis der er noget, man ikke explicit har givet adgang til, så er der ikke adgang. På den anden side må man så sikre sig, at alle services, der skal kunne anvendes, er sat op i `/etc/hosts.allow`.

Filerne har det følgende format. Eks. på en linje:

```
in.telnetd: 192.168.0.0/255.255.255.0
```

I `/etc/hosts.allow` ville denne linje tillade alle på lokalnettet 192.168.0.* at benytte telnet servicen. Stod linjen i `/etc/hosts.deny`, ville det betyde, at de ikke kunne.

Eksempel på `/etc/hosts.allow` fil:

```
in.ftpd: 0.0.0.0/0.0.0.0
ALL: 192.168.0.2
ipop3d: 192.168.0.0/255.255.255.0
```

Denne `/etc/hosts.allow` vil tillade hele verden adgang til ftp, og giver host 192.168.0.2 lov til alt. Hosts på netværket 192.168.0.* har desuden adgang til pop3.

Eksempel på `/etc/hosts.deny` fil:

```
in.telnetd: 0.0.0.0/0.0.0.0
```

Denne `/etc/hosts.deny` fil sørger for, at ingen har adgang til telnet servicen. Dette gælder dog ikke for 192.168.0.2, da den allerede i `/etc/hosts.allow` har fået lov til alt herunder telnet.

En mere sikker `/etc/hosts.deny` ville være denne:

```
ALL: 0.0.0.0/0.0.0.0
```

Denne linje nægter alle adgang til alt, medmindre der specifikt er givet lov i filen `/etc/hosts.allow`. Bemærk dog, at du kan blive overrasket over ting, der ikke virker, hvis du har sat den restriktive `/etc/hosts.deny`-fil op. Vær omhyggelig med at sætte de nødvendige tilladelser op i `/etc/hosts.allow`. Hvis du har glemt at give adgang til noget, opdager du det sikkert, når du skal bruge det, eller når nogen brokker sig. Har du derimod glemt at spærre for noget, opdager du det måske først den dag, det bliver brugt til at bryde ind på dit system. Med sikkerhed og paranoia i højsædet er det smart at have ovenstående linje i sin `/etc/hosts.deny`. Så bliver forespørgsler afvist, hvis du ikke har sat dem ind i `/etc/hosts.allow`. HUSK at en forespørgsel bliver godkendt, hvis du ikke har angivet andet.

Se i øvrigt **man tcpd**, og **man hosts.allow** eller **man hosts.deny** (de to sidste er samme fil).

2.3. Hvilke inetd services skal jeg slå fra?

Hvilke services, der kan undværes på dit system, afhænger af hvad det skal bruges til, og det er dig selv der må afgøre det. Men vi vil gennemgå de mest almindelige services, hvad de gør, om de er usikre, og hvad de evt. kan erstattes af.

2.3.1. Telnet

Telnet (telnetd) er en service der gør, at folk kan logge ind på maskinen ved hjælp af en telnet-klient. For at det er muligt at logge ind, skal man have et brugernavn og tilhørende password, som findes på maskinen. Når man er logget ind, kan man så udføre de tekstkommandoer, man plejer at kunne udføre, når man sidder foran maskinen. Telnet kører normalt på port 23, men kan sættes op til en anden port.

Telnet er en gammel sag, fra dengang i begyndelsen af internettets historie, hvor der kun var venner og ingen fjender på nettet. Det var mest forskere på universiteter, der havde adgang, og de ville ikke hinanden noget ondt. Alle kendte mere eller mindre hinanden, så hvis man lavede ballade var man hurtigt upopulær blandt ligesindede. Den slags moral kan vi ikke regne med mere.

Mange af de "gamle" services er usikre i dag, fordi de slet ikke er designet med sikkerhed i tankerne men alene stabilitet og fornuftig ydelse.

Telnet er først og fremmest usikker, fordi den sender brugernavn og password som klar ukrypteret tekst over nettet. Dvs., at det er lige til at læse for enhver, der måtte sidde og kigge på nettrafikken et sted imellem dig selv og den maskine, du logger ind på. Det er særdeles risikabelt at bruge telnet over internettet, hvor alle i princippet kan sidde og lytte med.

På en maskine, der har adgang til internettet, bør telnet slås fra, og erstattes med ssh, se Afsnit 4.2. En grund til at beholde telnet kan dog være, at der findes telnet-klienter til stort set alle systemer. Hvis man har brug for at logge ind fra et system, hvortil der ikke findes en ssh-klient, kan det være nødvendigt at bruge telnet. Men det er risikabelt at lade en telnet service køre, hvis maskinen er på internettet. Så overvej, om der ikke er andre muligheder for at løse dine behov.

Hvis telnet skal køre, kan man sikre sig bedre med adgangskontrol (`/etc/hosts.allow`), ved at skifte password hver gang og ved ikke at have det samme password på andre maskiner i lokalnettet, så skaden ved indbrud trods alt begrænses.

2.3.2. Ftp

FTP (File Transfer Protocol) bruges, når man vil kopiere filer fra en maskine til en anden. FTP er ligesom telnet en af de gamle services, som er meget stabil men slet ikke sikker, idet login navn og password sendes i klar tekst. FTP er imidlertid ofte mindre farlig end telnet, da det meste ftp-trafik foregår som "anonymous ftp".

Når man skal ftp'e til en maskine, skriver man ftp maskinnavn (evt portnavn). F.eks.:

```
ftp bohr 37067
```

Ovenstående kommando vil forsøge at åbne en ftp forbindelse til host "bohr" port 37067. Så bliver man spurgt om loginnavn og derefter password. Går det godt, er man inde, og man kan hente filer med kommandoen "get" og lægge dem ud med "put". Man kan desuden vælge, om det skal være binær eller tekstoverførsel, skrive "ls" for at se indhold af katalog eller skifte katalog med "cd" mm.

Har man en rigtig bruger konto på maskinen, og er ftp sat op til det, kan man logge ind med sit normale brugernavn og password. Dette kan være relevant på f.eks. et lokalnet, hvor brugerne har behov for at flytte filer. Faren er her den samme som ved brug af telnet, nemlig at brugernavn og password sendes i klar tekst over nettet. Foregår dette over det usikre internettet, må vi bestemt anbefale at man bruger scp secure copy fra ssh-pakken (se Afsnit 4.2), hvis det er muligt. Alternativet er en speciel krypteret udgave af ftp, men dette kræver, at klienten kan tale samme sprog. Ftp kører normalt på port 20 (data) og 21 (kommandoer).

Det er imidlertid meget almindeligt, at en ftp server er sat op til at i acceptere brugernavnet "anonymous" eller "ftp". En del ftp servere er beregnet til kun at understøtte denne type login. Det kaldes anonymous ftp, og bruges ved "offentlige" ftp servere, hvor enhver kan hente de filer der er lagt frem. Her bruger man sin e-postadresse som adgangskode. Ved anonym ftp kan alle logge ind, og det lyder måske farligt, men det er det kun, hvis man har sat sin ftp server forkert op.

Der sendes ikke brugernavne og passwords over nettet. Der sendes kun brugernavnet "anonymous", samt en e-postadresse, og det gør ingenting for serverens sikkerhed, at det bliver opsnappet.

Ved anonym FTP har folk ikke brug for at kunne ret meget. De har brug for at kunne hente og måske uploade. Og for at kunne gå hen i det katalog de skal uploade i og tilsvarende hente fra, men stort set har de ikke brug for mere. De har ikke brug for at kunne bevæge sig rundt på hele maskinen, og det bør de ikke i have lov til. Dette undgås ved, at man "chroot'er" dem, dvs., at man f.eks. får kataloget `/home/ftp` til at se ud som roden af filsystemet (dvs. /). Brugeren kan ikke se resten af filsystemet og kun få ordrer kan anvendes. Dermed er serveren ikke nær så udsat, som hvis alle og enhver kunne logge "rigtigt" ind. Det er kun lidt, der skal sættes op, før et chroot'et environment virker (så bla. "ls" og "cd" virker) - nogle filer skal kopieres til `/home/ftp/lib`, `/home/ftp/bin` og `/home/ftp/etc` filer. Dette er gjort for dig i alle de store Linuxdistributioner.

Endelig bør man huske, at `/home/ftp` ikke bør været ejet af den bruger, folk logger ind som, dvs. anonymous eller ftp. Så kan de ændre environment, lave .rhosts-filer og andet, som kan bringe net-sikkerheden i fare.

Der er et par ting mere, man bør tænke på. For det første bør der ikke være læse og skriverettigheder på det samme katalog. Folk skal uploade et sted og hente fra et andet, og serverens administrator skal så sørge for at uploadede ting evt. bliver tilgængelige til nedhentning. Dette skyldes, at hvis man bare lader sin server stå åben for uploads og nedhentning uden kontrol, vil folk bruge den til piratsoftware, porno etc. Derfor er det smart først at lade det folk uploader være tilgængeligt for andre, efter at man har kigget det igennem.

Der er en sidste ting, man skal være opmærksom på med en ftp server. Folk kan med vilje eller ved et uheld uploade så meget, at harddisken løber fuld. Det kan give systemet problemer, så det skal man undgå. Det kan styres ved at lade uploadkataloget ligge på en partition (eller disk) for sig selv, eller ved i opsætningen af ftp serveren at lave begrænsninger for størrelsen af den uploadede mængde. Hvis folk med vilje fylder ens disk op, for at systemet skal holde op med at fungere, kaldes det et "denial of service attack" ("ude af drift angreb", også kendt som DOS attack) - maskinen nægter brugerene den service, som skulle leveres, fordi den er sat ud af drift af angrebet. Der findes også sikkerhedshuller til andre slags DOS attacks i nogle ftp-servere. Hvis du ikke skal bruge ftp, bør du slå den fra i `/etc/xinetd.d`.

```
ftp    stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
```

bliver til

```
#ftp    stream  tcp     nowait  root    /usr/sbin/tcpd  in.ftpd -l -a
```

Husk at genindlæse `inetd`s opsætning.

2.3.3. Finger

Finger er et program, som viser hvem, der er logget på maskinen lige nu, hvornår de sidst var logget ind og hvor længe:

```
[robin@sherwood]$ finger robin
```

```
Login: robin                            Name:
Directory: /home/robin                  Shell: /bin/bash
On since Fri Jul  9 10:18 (CEST) on tty1    1 hour 14 minutes idle
     (messages off)
On since Sat Jul 10 01:13 (CEST) on tty2    1 hour 14 minutes idle
     (messages off)
On since Sat Jul 10 01:14 (CEST) on pts/0   1 hour 15 minutes idle
     (messages off)
On since Sat Jul 10 01:14 (CEST) on pts/2   26 minutes 11 seconds idle
     (messages off)
No mail.
No Plan.
```

Denne kommando kan køres fra andre maskiner mod din maskine, og kan anvendes af folk udefra til f.eks. at finde frem til en brugerkonto, der ikke har været brugt længe. Denne information bør du ikke give videre ud over maskinen selv. Derfor bør du i `/etc/xinetd.d/finger` rette

```
disable         = yes
```

Finger er for de fleste fuldstændig unødvendig og en dum sikkerhedsrisiko at lade stå åben. Hvis du kan finde en undskyldning for, at du har brug for finger, så skal den i hvert fald *kun* være tilgængelig internt. Dvs., at adgang fra andre maskiner skal være spærret f.eks. med tcp-wrappers.

2.3.4. POP

Hvis din maskine ikke skal være mailserver, så indsæt `disable=yes` i filerne med pop-2, pop-3 og imap i `/etc/xinetd.d`.

POP står for Post Office Protocol, og man bruger POP til at hente mail fra en mailserver og lægge i sin inbox på ens egen maskine. Bemærk, at selvom man slår POP-serveren fra lokalt, forhindrer det ikke, at man kan hente mail via sin udbyders POP-server. Da POP-daemonen skal skrive i brugernes hjemmekataloger, er den nødt til at køre som root. Dette er et sikkerhedsproblem. Ligesom med telnetd og ftpd er det desuden et stort problem, at POP kører med ukrypteret transmission af brugernavn og password.

Afhængig af hvad man skal bruge POP til, kan man sikre den. Hvis den kun skal bruges internt i ens firma, kan man enten

- sørge for, at der ikke kan tjekkes mail udefra ved opsætning TCP-wrappers og/eller firewall.
- bruge krypteret overførsel, hvis man kun har mailklienter, man selv bestemmer over.

Man kan kryptere POP, men igen er dette en udvidelse, som kræver at alle klienter understøtter det. Det er ikke altid tilfældet. Der findes også en metode, der hedder APOP, som de fleste POP-programmer understøtter. Med APOP sendes passwordet krypteret. Dette er baseret på MD5-kodning, hvor der også er et time stamp involveret. Mere information kan findes i RFC 1725.

IMAP er en udvidet version af POP. Den kan sætte sig selv til at køre med brugerens privilegier noget af tiden i stedet for root, og man kan meget mere med den end med POP. F.eks. kan man med IMAP være flere brugere om én account, nøjes med at hente headerene fra sine mails og lade resten ligge på serveren, etc.

For at slå POP og IMAP fra i `/etc/inetd` skal følgende linjer udkommenteres:

```
pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd ipop2d
pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd ipop3d
imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
```

bliver til

```
#pop-2   stream  tcp     nowait  root    /usr/sbin/tcpd ipop2d
#pop-3   stream  tcp     nowait  root    /usr/sbin/tcpd ipop3d
#imap    stream  tcp     nowait  root    /usr/sbin/tcpd  imapd
```

2.3.5. Linuxconf

Anvender du en nyere Linux distribution, kan det være, at du har programmet linuxconf kørende på din maskine på port 98. Du kan i `/etc/inetd.conf` se, om du har linuxconf kørende ved at lede efter en linje, der indeholder ordet **linuxconf**. Den kan f.eks. se sådan ud:

```
linuxconf stream tcp wait root /bin/linuxconf linuxconf --http
```

Ganske vist kan linuxconf umiddelbart kun tilgås fra den lokale maskine og ikke fra netværket, men programmets egen usikrede webserver bør man ikke stole blindt på. Luk det, hvis du har et usikkert net. Tilsvarende argumenter kan anvendes overfor programmer såsom webmin, som giver et webinterface til systemadministration. Webmin er normalt ikke installeret, men skal hentes som særskilt pakke. For at værktøjet til webadministration kan accepteres på et usikkert net, skal SSL eller bedre kryptering indbygges i server og klient.

2.4. RPC og portmap

Ud over services, som konfigureres via `/etc/xinetd.d`, er der en klasse af services, som anvender RPC - Remote Procedure call. Vi vil se lidt på RPC og de mest kendte RPC-baserede services.

RPC går ud på, at man har et netværk af computere og kan få dem til at arbejde sammen som et system. Altså en slags "distributed computing". Man kan driste sig til at kalde det en tidlig forgænger for CORBA (http://www.omg.org). Det overordnede princip er, at man kalder en funktion, som enten udføres af den lokale maskine eller en anden maskine i netværket.

I praksis betyder det, at der kan køre en række RPC services på din maskine. I stedet for at RPC services kører på et bestemt portnummer, får de tildelt et portnummer ved opstart. De registrerer sig hos portmapper daemonen, når de bliver startet. Portmapper daemon'en holder styr på hvilken RPC services der kører på hvilket portnummer, og videresender forespørgsler fra klienter til den rette service. RPC services behøver altså ikke køre på et bestemt portnummer, som klienten kender i forvejen. Klienten henvender sig til portmap på serveren og bliver "stillet igennem" til den rigtige service.

Dette er meget smart, men det komplicerer opsætningen af firewalls, da man ikke på forhånd ved, hvilken port den RPC service vil benytte. Prøv at skrive

```
[robin@sherwood robin]$ /usr/sbin/rpcinfo -p localhost
   program vers proto   port
    100000    2   tcp    111  rpcbind
    100000    2   udp    111  rpcbind
    100005    1   udp    635  mountd
    100005    2   udp    635  mountd
    100005    1   tcp    635  mountd
    100005    2   tcp    635  mountd
    100003    2   udp   2049  nfs
    100003    2   tcp   2049  nfs
```

(Dette er fra et Red Hat system. På Debian står der portmapper i stedet for rpcbind). Man kan se at NFS og mountd (NFS mount) kører, men ikke NIS.

Portmapperen kører normalt på port 111. Hvis man ikke har brug for, at nogle RPC services er tilgængelige udefra, kan man i sin firewall blokere port 111. Hvis ikke du ved, om du bruger RPC, så gør du det ikke. Fjern det, og stop portmapper servicen.

Det er i øvrigt ikke nok at stoppe portmapperen. Der findes værktøjer til at sniffe på alle åbne porte og dermed afsløre kendte RPC-services uden at spørge portmapperen.

Udefra kan man spørge maskinen hvilke netværksprocesser, som tilbydes. Et program, som er velegnet til dette, er "nmap", som kan hentes fra http://www.insecure.org/nmap. Prøv at køre nmap på din egen maskine:

```
[robin@sherwood robin]$ nmap localhost
Starting nmap V. 2.2-BETA4 by Fyodor (fyodor@dhp.com, www.insecure.org/nmap/)
Interesting ports on localhost (127.0.0.1):
Port    State       Protocol  Service
25      open        tcp       smtp
37      open        tcp       time
53      open        tcp       domain
70      open        tcp       gopher
80      open        tcp       http
98      open        tcp       linuxconf
111     open        tcp       sunrpc
^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^
113     open        tcp       auth
```

```
139     open        tcp       netbios-ssn
143     open        tcp       imap2
513     open        tcp       login
514     open        tcp       shell
515     open        tcp       printer
635     open        tcp       unknown
2049    open        tcp       nfs
6000    open        tcp       X11

Nmap run completed -- 1 IP address (1 host up) scanned in 0 seconds
```

Man kan således udefra se, at sherwood har sunrpc til at lytte på port 111 - sunrpc er det samme som portmapperen. Processen - det kørende program - hedder portmap, men servicen hedder sunrpc. Bliv ikke forvirret, kært barn har mange navne. Kør **ps -aux |grep portmap** på dit system, og du kan indefra se, at portmapper daemonen kører.

Portmapper daemonen er et program, der hedder portmap (på nogle systemer hedder det rpcbind). Den bliver startet op i dine startup scripts (Red Hat: `/etc/rc.d/init.d/portmap`, Debian: `/etc/init.d/netbase`).

RPC er rent sikkerhedsmæssigt noget skidt. RPC er, som alt muligt andet fra dengang, designet med henblik på funktionalitet og stabilitet fremfor sikkerhed. RPC blev i sin tid lavet af Sun, men det blev hurtigt efterlignet af andre og vandt stor udbredelse. Problemet med RPC er, at metoden til autentifikation er baseret alene på brugerens UID og GID (User/Gruppe ID). RPC serveren tror på, at man er den, man giver sig ud for at være - uden yderligere validering af identitet. Udgiver man sig for at være en bruger med adgang, så bliver man lukket ind.

Sun kom senere med en mere sikker version, secure RPC, som bruger en kryptering baseret på DES. Secure RPC vandt aldrig den samme udbredelse, fordi den ikke blev efterlignet af andre (noget med at den anvendte krypteringsmetode var patenteret, og man skulle købe en licens for at bruge den).

Det er en dårlig idé at slå portmapper daemonen fra, før du har undersøgt hvilke programmer, der bruger den. NFS kører f.eks. via RPC, hvilket vi allerede så, da vi kørte rpcinfo programmet. Vi så faktisk alle de services, der har registreret sig hos portmapperen. Portmap er selv en af dem (kan hedde rpcbind). Alt efter hvor mange services, du har startet op, så kan du se, at services som nfs, mountd, nis, automounteren amd m.fl.kører på dit system via RPC.

Du er nødt til at tage stilling til, om du har brug for disse services, hvis ikke skal de slås fra. Hvis **rpcinfo -p localhost** kun viser portmapperen selv, evt fordi du har slået de andre services fra, så kan du godt slå portmap fra også. Men bemærk, at den skal slås til igen, hvis du vil anvende en af de services, der benytter RPC.

2.4.1. NIS og NIS+

NIS (Network Information Service) er en måde, hvorpå man kan nøjes med at have sin `/etc/passwd` fil og andre opsætningsfiler (typisk en del filer fra `/etc/`) på én server. Klienterne får de nødvendige oplysninger af serveren via NIS. NIS er baseret på RPC. Med NIS slipper man for at vedligeholde sine opsætningsfiler på en masse maskiner. Det gør det lettere at holde styr på sine brugere, grupper og hvem, der har lov til hvad. NIS er mere komplekst end UNIX bruger og gruppe rettighedssystemet og gearet til at holde styr på et stort netværk, hvor en bruger ikke må det samme på alle maskiner. NIS+ er en mere sikker version med lettere administration af store systemer (lettere opdatering af serveren etc).

Klient daemonen til NIS hedder ypbind, server daemonen ypserv og kommandoerne starter med yp (et levn fra dengang NIS hed yellow pages). Se i øvrigt http://www.sunsite.auc.dk/ldp/HOWTO/NIS-HOWTO-6.html.

2.4.2. NFS

NFS (Network File System) er et netværksfilsystem, som muliggør af flere maskiner kan dele den samme disk via netværket. Med NFS kan man eksportere hele eller dele af sit lokale filsystem, så en bruger kan mounte det fra andre maskiner over nettet. Det er dumt og unødvendigt at eksportere hele sin systemdisk - man bør nøjes med at eksportere de dele af filsystemet, der er behov for NFS-adgang til. NFS kører normalt på port 2049.

NFS er en RPC baseret service og er afhængig af at portmapperen kører. NFS startes af et startupscript (Red Hat: `/etc/rc.d/init.d/nfs`, Debian: `/etc/init.d/nfs-server`. Daemonerne hedder **rpc.mountd** og **rpc.nfsd**. mountd daemonen tager sig af selve mount processen, imens NFS daemonen tager sig af at åbne, lukke, læse og skrive filer etc.

Filsystemer, der skal være tilgængelige via NFS, angives i `/etc/exports`, f.eks.

```
/home/robin     192.168.0.0/255.255.255.252(rw)
```

Denne linje eksporterer kataloget `/home/robin/` med adgang for maskinerne 192.168.0.1, 192.168.0.2 og 192.168.0.3. Den eneste option, der er sat i ovenstående linje, er "rw" (read write). Der findes desuden nogle sikkerhedsrelaterede options til `/etc/exports`, men angiver du ingen, er de sat til det sikreste. Du skal således manuelt slå dem til for at få den mere usikre version. F.eks. "secure" option, som betyder at en forespørgsel skal komme fra en port mindre end 1024. Den er default slået til - vil man slå den fra, må man specificere "insecure". Beskrivelse af yderligere options findes i **man exports**.

Lad ikke dette forlede dig til at tro, at NFS er sikkert. Det er grundlæggende usikkert, da det kører via RPC, som du ikke kan betragte som sikker. Selve datatrafikken er heller ikke krypteret. Begræns din filsystemexport til det, du rent faktisk har brug for, og mount kun de drev du har brug for. Lad være med at mounte alle eksporterede drev på alle klientmaskiner, hvis der ikke er behov for det. Dette er i øvrigt

klogt af hensyn til den almindelige driftstabilitet, idet du kun bliver afhængig af de maskiner, du skal bruge.

Hvis du ikke bruger NFS, så slå den fra. I Red Hat, Debian eller SuSE gøres det ved at fjerne symlinkene til startup scriptet i de forskellige runlevel kataloger `/etc/rc.d/rcN.d` eller `/etc/rcN.d`, hvor N er runlevel (0-6). I nogle distributioner er det i stedet nødvendigt at fjerne de linjer, der har med NFS opstart at gøre, i et større script, der køres, når maskinen startes, og som starter mange andre ting end NFS. Kører du Red hat, så kig på programmet **chkconfig**, som er et nemt og overskueligt værktøj til at styre, hvad der startes ved de forskellige runlevels.

2.5. Apache webserveren

Apache er verdens mest anvendte webserver. Den følger standard med Linux, og findes desuden til alle andre UNIX systemer. Ofte sker det under installationen af Linux, at Apache er installeret uden, at du har lagt mærke til det. Hvis din maskine ikke skal fungere som webserver, har du kun brug for Apache, hvis du vil lege med CGI-scripts, PHP eller andre smarte ting (hvor du for alvor skal passe på din sikkerhed). Tjek om du kører Apache eller en anden webserver på port 80 (kan være andre steder såsom port 8080) ved at skrive

```
[robin@sherwood robin]$ telnet MASKINNAVN 80
```

og så skriv f.eks. "?" og tryk retur. Ser du noget HTML kode, så er der en webserver. Apache er meget stabil og gennemtestet, men tænk over, om du behøver den.

2.6. Sendmail

Sendmail er en MTA (Mail Transfer Agent). Den bruges til at dirigere post rundt på systemet og lytter normalt også på smtp porten. Sendmail er et meget komplekst program at sætte op og meget komplekst at forstå. Hvis du er begynder, vil vi anbefale, at du ikke begynder at sætte dig dybdegående ind i sendmail lige nu, medmindre du har brug for det. Vi vil her kort forsøge at forklare, hvornår dit system anvender sendmail, hvorfor/hvornår det er farligt, og hvilke alternativer, der er.

2.6.1. Hvorfor er sendmail farlig?

Sendmail er en meget gammel sag, og den har et ret dårligt rygte, hvad sikkerhed angår, men er uhyre anvendt. Den har eksisteret i mange år og har i tidens løb haft mange slemme sikkerhedshuller. Se ftp://koobera.math.uic.edu/www/maildisasters/sendmail.html (ftp://koobera.math.uic.edu/www/maildisasters/sendmail.html). Disse er dog blevet rettet efterhånden - men der kan komme nye. Da sendmail er et meget komplekst program, kan der også godt være nogle fejl, som der går lang tid, før nogen opdager. Sendmail kører med root-privilegier, d.v.s., den har adgang

til hele systemet og kan gøre alt, hvad en administrator også kan gøre. Den fungerer på de fleste systemer som daemon, dvs., at den kører hele tiden og lytter på sin port (sendmail kører default på port 25), om der er noget mail, den skal tage imod. Hvis sendmail kører som daemon, og der er et sikkerhedshul i den, kan enhver ude fra nettet sende en forespørgsel til dens port og udnytte sikkerhedshullet. En af grundene til, at man ofte hører skrækhistorier om sendmail, er at den kører på langt de fleste maskiner, der håndterer post på internettet. Fordi den kører på så mange systemer, er det let nok at finde et system at udnytte, hvis man har fundet et sikkerhedshul. Det er dog ofte folk, der kører gamle versioner af sendmail, der er udsat. Hvis man hele tiden holder sin sendmail opdateret, er risikoen ikke så stor. Hent den nyeste version på http://www.sendmail.org/.

2.6.2. Skal jeg slå sendmail fra?

Det er desværre ikke så smart at slå sendmail helt fra - med mindre man i stedet installerer et sikrere alternativ som qmail eller postfix - da flere dele af systemet skal kunne levere mails internt på maskinen. Skal maskinen imidlertid ikke acceptere mail udefra, men kun fordele mail lokalt, kan man lade være med at lade sendmail køre i "daemon mode", hvor den lytter på en port. Man kan i stedet starte den i "queue mode". Sendmail kører stadig som en baggrundsproces, som med jævne mellemrum kommer frem og tømmer mailkøen. F.eks. en gang i timen, eller en gang hvert kvarter. Men den lytter ikke længere efter forespørgsler på port 25. Sendmail startes i "daemon mode", der tømmer mailkøen en gang i timen, ved at starte den med parametrene

```
sendmail -q1h
```

i stedet for

```
sendmail -bd -q1h
```

hvor `-q1h` betyder at køen tømmes en gang hver time, og **bd** betyder, at den startes i "daemon mode". Sendmail startes normalt fra et startup script. I Red Hat er det `/etc/rc.d/init.d/sendmail`. I SuSE skal man i `/etc/rc.config` ændre linjen

```
SENDMAIL_args="-bd -q30m -om"
```

til

```
SENDMAIL_args="-q30m -om"
```

Parameteren -q30m betyder, at mail-køen tømmes hver 30. minut. Hvis man ikke vil køre sendmail i "daemon mode", kan man ændre i startup scriptet. Hvis man slet ikke vil køre sendmail, skal man fjerne det symlink, der er til sendmail startup scriptet under de forskellige runlevels kataloger. I Red Hat er det `/etc/rc.d/rcN.d`, hvor N er runlevel nummeret. På Debian hedder det `/etc/rcN.d`. Red Hat eksempel: Prøv at skrive

```
[robin@sherwood robin]$ ls -l /etc/rc.d/rc3.d |grep sendmail
lrwxrwxrwx   1 root     root           18 Oct  3  1998 S80sendmail -> ../init.d/sendmail
```

Der kan man se at sendmail bliver startet i runlevel 3. Hvis sendmail ikke skal startes, skal dette symlink slettes. Ligeså under de andre runlevels, og husk også at slette shutdown symlinkene til sendmail (dem der starter med K).

Man kan også vælge at køre sendmail i "queue mode", selvom man har brug for, at maskinen tager imod mails udefra. Dette kræver, at man anvender en anden smpt-daemon, f.eks. optuse smtpd (http://www.optuse.com). Den tager sig af indgående smtp og lægger posten over i sendmails kø, hvor sendmail så kan overtage. Vi kan også nævne smap/smapd, som er en del af TIS Internet i Firewall Toolkit (http://www.tis.com/research/software/fwtk/fwtk_over.html).

2.6.3. Sikring af sendmail

Hvis du kører sendmail som i "dameon mode", kan du sikre den. Du kan bl.a enable nogle security options i `/etc/sendmail.cf` - som er en ret forvirrende fil ved første blik. Man kan f.eks. sætte linjen

```
O privacyoptions=authwarnings,needmailhelo, noexpn, novrfy
```

ind i sin `/etc/sendmail.cf`. `authwarnings` aktiverer "email authentication warnings", så man får "x-authentication-warning"-beskeder i sine brevhoveder, som f.eks.

```
X-Authentication-Warning: sherif.nottingham.dk: sherif owned process doing -bs.
```

`authwarnings` er pr. default slået til. `needmailhelo` gør, at den maskine, der vil sende mail til sendmail, skal identificere sig før den får lov. `novrfy` forhindrer `VRFY` kommandoen, som bruges til at bekræfte, om et bestemt brugernavn findes på maskinen. `noexpn` forhindrer `EXPN`, som er en kommando, der bruges til at finde ud af hvilken bruger/program på systemet en e-postadresse reelt peger på, dvs., hvem et mail alias dækker over, hvem der modtager root mails på systemet etc.

Prøv at skrive **telnet hostname 25**, hvor hostname er navnet på en maskine, der har sendmail (evt. din egen linux box - du behøver ikke 2 forskellige maskiner til dette). Sendmail bruger normalt port nummer 25. Så du kan faktisk kommunikere direkte med sendmail på maskinen ved at bruge telnet:

```
[robin@sherwood robin]$ telnet bohr 25
Trying 192.168.0.1...
Connected to bohr.herne.dk.
Escape character is '^]'.
220 sherwood.dk ESMTP Sendmail 8.9.3/8.9.3; Mon, 19 Jul 1999 22:47:59 +0200
VRFY robin
250 <robin@sherwood.dk>
EXPN robin
250 <robin@sherwood.dk>
```

Vi fik at vide at maskinen kører sendmail version 8.9.3, at robin fandtes på maskinen, og at mail til robin går til robin. Skal andre også have dette - og måske mere følsomme ting - at vide? Hvad nu hvis robin

bare var et mail alias for brugeren marion - kommer det andre ved? Skal andre kunne se, at roots mail også går til marion? Vi prøver samme kommando, når VRFY og EXPN er slået fra.

```
[robin@sherwood robin]$ telnet bohr 25
Trying 192.168.0.1...
Connected to bohr.herne.dk.
Escape character is '^]'.
220 sherwood.dk ESMTP Sendmail 8.9.3/8.9.3; Mon, 19 Jul 1999 22:53:17 +0200
VRFY robin
252 Cannot VRFY user; try RCPT to attempt delivery (or try finger)
EXPN robin
502 Sorry, we do not allow this operation
```

Der er flere muligheder. Man kan nøjes med at slå `expn` og `vrfy` fra, indtil den fremmede maskinen har identificeret sig selv. Se http://hoth.stsci.edu/man/man1m/sendmail.html.

2.6.4. Mail relaying

Derudover skal man slå mail relaying fra. Det gøres i accessfilen - `/etc/sendmail.cf` indeholder oplysningerne om, hvor den ligger. På Red Hat hedder den `/etc/mail/access`. Der må gerne stå f.eks.

```
localhost.localdomain           RELAY
localhost                       RELAY
```

- dette tillader relaying for den lokale maskine. Men der må ikke stå andre linjer med RELAY. (Man kan f.eks. også bruge IP-adressen på sit interne netværk og tillade intern mail relaying). Hvis man tillader extern mail relaying - dvs. at folk, der logger ind udefra, kan sende mails, så det ser ud som om, de er sendt indefra - så kan man risikere, at spammere bruger en som afsender for deres spam. Så selvom det til tider kunne være praktisk at logge ind på sin computer, når man er ude, og sende post med sin "hjemmeadresse" som afsender, så tillader internetsikkerheden i dag ikke dette. Man bliver *meget* upopulær, hvis nogle garvede internet folk opdager, at man har mail relaying slået til. Man kan komme på deres sortlister, så en masse folk ikke kan modtage mail fra ens domæne etc. Man kan dog også træffe andre foranstaltninger imod mail relaying end at slå det fra i sendmail.

Læs desuden **man sendmail**.

Der skal også nævnes, at der findes interessante alternativer til sendmail: qmail (http://www.qmail.org) og postfix (http://www.porcupine.org/postfix-mirror/start.html). Begge er fra begyndelsen designet med tanke på sikkerhed. De er bl.a.sikrere, fordi smtpd her er en selvstændig proces, som ikke kan skrive til nogen filer. Hver lille delopgave foretages af et separat program, som kun har en snæver, veldefineret funktion. Sendmail er derimod designet som et enkelt do-it-all program. Det er en grundlæggende i designfejl, som gør den håbløs i sikkerhedssammenhæng.

Vi kommer ikke her nærmere ind på qmail og postfix, men har man et i seriøst mailserver behov, kan det godt anbefales at kigge nærmere på disse to programmer.

2.7. Andre services

Udenfor de forrige kategorier falder XDM og DNS:

2.7.1. Xdm-lignende login

En service man ikke må overse, er XDM, som er en X-login daemon, der normalt kører på UDP port 177. Det er muligt at forbinde sig til en kørende XDM-server, dvs. direkte til X-systemet, ved at skrive "X -query SERVER". Alt efter Linux distribution, kan du have xdm, kdm (KDE's XDM) eller gdm (GNOME XDM) kørende. Hvis XDM kører, starter maskinen op med en grafisk login menu i stedet for den almindelige konsolmodus-login-prompt. Hvis du er i konsolmodus, kan du se, om du har en X-login daemon kørende, ved at trykke Alt-F7. Er der en menu, hvor du kan skrive loginnavn og password ind, så kører du en XDM-lignende service.

På din hjemmecomputer kan det grafiske login være meget rart, men hvis du administrerer en server, er det en uacceptabel sikkerhedsrisiko. Luk den ned og sørg for, at dette ikke automatisk startes op fra startupscriptet i `/etc/rc.d`:

- SuSE 6.1: slet /etc/rc.d/rc3.d/*xdm
- Red Hat 6.0: Udkommenter sidste linjen i `/etc/inittab`

  ```
  x:5:respawn:/etc/X11/prefdm -nodaemon
  ```

  så den ser ud som

  ```
  #x:5:respawn:/etc/X11/prefdm -nodaemon
  ```

I øvrigt bør man slet ikke køre X på en server, hvis det kan undgås. Kan du af en eller anden grund ikke undgå at have XDM kørende, så bør du som minimum læse http://metalab.unc.edu/LDP/HOWTO/mini/Remote-X-Apps.html. Dette sted beskriver, hvordan du med "magic cookies" kan gardere dig mod sådan noget som spoofingangreb, hvor andre på nettet prøver at tage din identitet.

Det kan nævnes, at vi i Kapitel 4 ser nærmere på, hvordan du kan sende X-programmer fuldt krypteret. Denne løsning er klart at foretrække, hvis man ser på netværks-sikkerheden.

2.7.2. DNS

DNS (Domain Name System) er navneservice, som anvendes til at få oversat et hostnavn f.eks. sunsite.auc.dk til dets IP-nummer. DNS er en af de store grundpiller i internettet og derfor også meget gennemprøvet men ret komplekst. Der er blevet fundet sikkerhedshuller i DNS-programmerne, f.eks. var der en fejl i Red Hat 4.2, hvor man kunne sende en meget stor datamængde til en nameserver, hvilket medførte, at den blev "overrasket" (buffer overflow problem) og fejlede. Man kunne derefter mirakuløst

udføre rootkommandoer på systemet! Resultatet var fatalt. Det er nok ikke sandsynligt, at du kører en nameserver (på port 53) uden at vide dette, men du kan verificere dette ved at skrive.

```
[robin@sherwood robin]$ ps aux | grep named
```

Du skal kun køre et nameserverprogram, hvis maskinen reelt skal bruges som nameserver.

2.7.3. X-serveren og netværket

Vi så i Afsnit 2.4 at NMAP kunne se X-serveren udefra. Dette kan man undgå ved at rette sidste linje i `/etc/X11/xdm/Xservers` til følgende. Igen en opgradering af sikkerheden.

```
# $XConsortium: Xserv.ws.cpp,v 1.3 93/09/28 14:30:30 gildea Exp $
#
# Xservers file, workstation prototype
#
# This file should contain an entry to start the server on the
# local display; if you have more than one display (not screen),
# you can add entries to the list (one per line).  If you also
# have some X terminals connected which do not support XDMCP,
# you can add them here as well.  Each X terminal line should
# look like:
#       XTerminalName:0 foreign
#
:0 local /usr/X11R6/bin/X -nolisten tcp vt05
```

2.8. Epilog

Selvom vi har været inde på mange services her, har vi ikke været inde på alle. Programmer som **nmap** og kommandoer som **rpcinfo** og **netstat** kan vise dig hvilke porte, der benyttes, hvilke services, der kører, og hvilke netværksforbindelser, der er etableret. Brug også **ps aux** flittigt og se hvilke processer, der rent faktisk kører på dit system. Sørg for at alle unødvendige services i `/etc/xinetd.d` er disablet, og tjek dine runlevel kataloger. Læs manualsiderne til de services, du vælger at køre. Følg med på sikkerhedslister på internettet, og følg med i fejlrapportet for din Linuxdistribution. Tænk dig om, og hold øje med dine logfiler. Og læs de resterende artikler i denne serie.

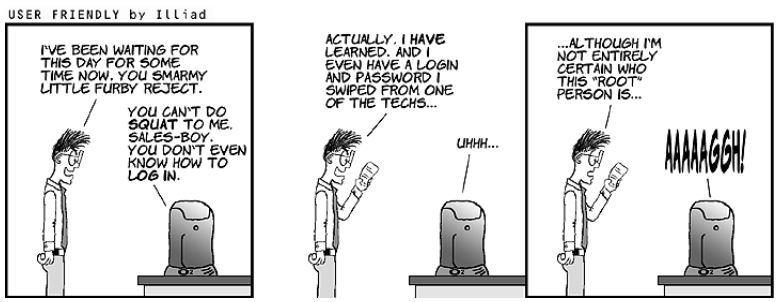
Kapitel 3. Root access - hvem, hvordan og hvorfor ikke?

Når en bruger for første gang skal til at anvende Linux (eller en anden UNIX), så det kan måske være svært at forstå, hvorfor man ikke skal være logget ind som systemadministrator - dvs. brugeren root - hele tiden. Dette vil vi starte dette kapitel med at diskutere. Derefter vil vi se på fornuftig håndtering af root rettigheder.

Der er flere sikkerhedsaspekter forbundet med root-adgang. Ud over risikoen for selv at komme til at ødelægge noget på sit system, er der spørgsmålet om hvilke personer ud over systemadministratoren, der skal have root privilegier - om nogen. I den forbindelse vil vi også se på nogle af de elementære forholdsregler, man bør tage for at forhindre, at uautoriserede personer får root adgang. Desuden vil vi se på valg af passwords. Det er meget vigtigt - specielt for root-kontoen - at der vælges passwords, der ikke kan gættes. Som det sidste i kapitlet ser vi på programmet **sudo** og suid programmer, dvs. programmer, der kører med rettigheder, som om de var startet af root. Dette kan naturligvis være et sikkerhedsmæssigt problem. Disse lokale problemer bliver til netværksproblemer i det øjeblik, nogen trænger ind på systemet via netværket. Er en cracker inde som almindelig bruger, er det endnu begrænset, hvor meget skade han kan gøre. Det er nok desværre sandsynligt, at crackeren vil gå efter at få root-rettigheder for at have fuld kontrol over din computer.

Det ville også være slemt, hvis nogen fra *salgsafdelingen* fik fat i dit root password... Fra http://www.userfriendly.org/static

Figur 3-1. User Friendly



3.1. Hvem er root?

På et Linuxsystem er der som regel et antal brugere med forskellige rettigheder. Linux er grundlæggende et fler-bruger system. Hvis man f.eks. bruger Linux i en virksomhed, eller på en skole, er der forskel på, hvad folk skal have lov til at gøre på maskinen. Hvor nogle brugere kun kan hente deres post fra

maskinen, kan andre betroede brugere have lov til at logge ind på maskinen og køre programmer. De forskellige brugere har et brugernavn og et password, som de logger ind på systemet med. Ud fra bruger-ID nummeret (fås ud fra brugernavnet) afgør systemet, hvilke handlinger man har lov til at udføre. Passwordet sikrer, at man er den man giver sig ud for at være. Ingen af de "almindelige" brugere kan ændre systemfiler, konfigurere ny hardware eller re-installere software. Derfor har Linux brugeren root, som kan alt dette. Er man root, så har man total kontrol over maskinen. At være root kan du blive ved at logge ind med brugernavnet root, og det dertil hørende password. Brugerkontoen root er en systemadministratorkonto, som du også kan skifte til og fra ved brug af **su** kommandoen.

Det skal nævnes, at det er bruger-ID 0, som giver root-rettigheder, egentlig ikke alene navnet root. Hvis du derfor finder flere brugere med bruger-ID 0 (nul), så skal du være meget opmærksom på, hvad der er sket. Det er med stor sikkerhed resultatet af et netværks-indbrud på maskinen.

Hvis du er systemadministrator for en Linux-maskine og kender root passwordet, er det stadig vigtigt, at du har en almindelig bruger konto, som du bruger til daglig. Man bør kun være root, når det er nødvendigt, for at udføre opgaver, hvor root rettigheder er nødvendige. Lad os illustrere hvorfor.

Antag som første eksempel, at du arbejder i et lokale, hvortil andre har adgang, og at du altid logger ind som root. Du skal i løbet af dagen forlade din maskine et antal gange f.eks. til frokost eller møder. Hvis du bare en gang glemmer at sætte en password-beskyttet screensaver på (denne må ikke kunne afbrydes med **Ctrl-Alt-Backspace**) eller logge helt ud, kan enhver, som kommer forbi dit kontor ødelægge hele din maskine. F.eks. ved at skrive **rm -rf /** som vil slette *alle* filer på din harddisk uden at spørge, om du mener det - og der er ingen fortrydelsesmulighed. Eksemplet er naturligvis grelt og ren sabotage, men det er faktisk sket, at andre *lige* skulle rette noget på et tidspunkt, hvor systemadministratoren lige var ude, og med root-login forvoldte stor skade, fordi vedkommende overvurderede sine evner som UNIX-administrator. Andre uheldige hændelser sker ved, at en root glemmer at logge ud, og andre ser dette. For at drille skiver de så **rm -rf /** men uden at trykke retur - dvs. ordren er ikke udført endnu. Hvis den rigtige root så kommer tilbage og ved et uheld trykker retur, er alt tabt, og han har selv udført handlingen. Det, som var en sjov spøg, blev pludselig til et sort uheld. Desværre er uheld, som disse set før.

Lad os som det næste eksempel se på de uheld, man selv kan komme til at lave, når man er logget ind som root. Den hyppigste grund til fejl og ulykker er nok slåfejl eller rettere sløseri. På en Linux-maskine vil du normalt altid få udført den ordre, du skriver, og du må selv garantere for fornuften i dette. Linux har den fordel, at du som almindelig bruger ikke kan slette systemfiler, såsom den meget vigtige fil `/etc/passwd`, der indeholder information om brugerne og deres passwords. Prøver du, som almindelig bruger at slette password-filen, vil blot få en besked med "Permission denied" - og det skal du faktisk være glad for. Sker det samme, imens du er root, vil du få lov til at slette filen. Styrer maskinen post for 500 brugere, så går der sikkert mellem 10 og 20 sekunder før at din telefon er rødglødende af sure folk, som ikke længere kan hente post eller logge ind på maskinen.

Selvom man bare har sin Linux-maskine derhjemme, og der ikke er andre, der bruger den, er det stadig vigtigt at have mindst én almindelig konto ud over root kontoen.

Når du skal lave systemarbejde, så kan følgende fremgangsmåde være anvendelig: Tag en speciel rød kasket på hovedet, som tegn på at du nu vil være root :-) Skift nu til root arbejde ved at skrive **su - root**. Bemærk at nogle gange udelades minustegnet. Det kommer vi tilbage til. Og før du skriver den mindste ordre, så placer begge hænder under din bagdel. I den tilstand tænker du dig så grundigt om, før du skriver og udfører ordrer - for du kan ALT som root. Det lyder nok lidt komisk, men der er alvor i noget af det. Skil dit normale arbejde på maskinen, såsom at læse din egne breve og programmere, fra root arbejdet. Vær kun root, når det er nødvendigt og brug root kontoen med stor varsomhed. Sørg bl.a. for at videresende breve til systemadministratoren (brugeren "root") til en almindelig brugerkonto (din egen), i det du *ikke* bør læse post, når du har systemadministratorrettigheder.

Hvis du hører til dem, der tror, at de sagtens kan administrere at være root hele tiden, så kan vi kun sige, at du ikke er den første. Det er noget man typisk hører fra begyndere, der endnu ikke har oplevet, hvor let det er som root at ødelægge meget med en lillebitte forkert kommando. Hvis du vil være root hele tiden, så kræver det stor disciplin. Du bør overveje, om du vil udsætte dig selv for de risici, det indebærer at være root hele tiden.

Vi vil gerne sige det en gang for meget, så det er helt klart: En god administrator giver altid så få rettigheder som muligt til brugerne og tilsvarende til sig selv. Kun når det er nødvendigt, skifter systemadministratoren fra sin egen personlige konto til root kontoen, og man låner ikke sit root password ud. I praksis vil man mange steder synes, at denne meget restriktive måde at administrere sikkerhed er unødigt streng og ofte fører til alt for langsomme ændringer af systemet, men det er den afvejning man altid skal lave mellem kontrol, sikkerhed og fleksibel brug af et computersystem.

3.1.1. su root

Så hvad er forskellen på **su root** og **su - root**? Minus-tegnet betyder at root brugerens environment (miljø-variable) bliver brugt. Hvis minus-tegnet udelades, beholder root de miljø-variable, som brugeren der skrev **su** kommandoen havde. Dette kan være et sikkerhedsaspekt. En brugers miljø-variable indeholder bl.a. hans personlige sti til de programmer, han vil køre, gemt i variablen PATH. Forrest i en brugers PATH kan man ofte finde ".", som betyder "det aktuelle katalog". Det betyder, at når man skriver en kommando, vil maskinen først lede efter programmet i det aktuelle katalog. Det kan være meget praktisk, men som root er det farligt. Vi antager, at en bruger har skrevet et program og kaldt det **ls**, og lagt det i sit hjemmekatalog. Root står tilfældigvis i denne brugers hjemmekatalog, og skriver **ls**. Hvis root har "." forrest i sin PATH, hvad sker der så? I stedet for at **ls** kommandoen bliver kørt, bliver brugerens eget program kørt - som root! Root bør *ikke* have "." i sin egen PATH - og slet ikke forrest. Dette er bare et eksempel. Brugeren kan også have aliaser i sine opstartsfiler, så kommandoer ikke gør det, man forventer, og der kan ske uheld - selvom man som regel bruger "su" fra sin egen bruger konto, hvor man kender opsætningen. Desuden har root ofte `/sbin` og måske `/usr/sbin` i sin PATH, hvor der ligger en række systemkommandoer. Det er en god ide altid at bruge minus-tegnet.

3.1.2. Uddeling af root rettigheder

Som systemadministrator vil du komme ud for, at nogle brugere har behov for at kunne lave noget

"specielt" systemarbejde, og derfor mener de skal have root passwordet. Det kunne f.eks. være selv at kunne genstarte en webserver, måske stoppe/genstarte maskinen eller dræbe processer efter programmer, som ikke fungerer. Disse ting kræver at root passwordet anvendes på tidspunkter, som ikke kan forudsiges. Du kan bevare root passwordet på få hænder og alligevel give nogle brugere de tilstrækkelige systemrettigheder ved at installere programmet "sudo".

Programmet sudo følger med de fleste Linux distributioner under navnet `sudo.*.rpm`, men kan er dette ikke tilfældet med din distribution, så kan hentes fra http://www.courtesan.com/sudo.

Derefter skal du lære at konfigurere sudo rigtigt. Dette afgør, hvilke programmer bestemte personer kan tilgå. Du bør også følge med i hvilke sikkerhedsfejl, der bliver fundet i sudo (følg med på deres hjemmeside). Der er på sudo-hjemmesiden en kortfattet eksempelfil. Læs desuden man-sider for **sudo**, **sudoers** og **visudo**.

Opsætningen startes som root ved at skrive

```
[root@hven tyge]#  /usr/local/sbin/visudo
```

Nu kommer editoren vi frem med den opsætningsfil, du skal udfylde. Som et eksempel lader vi brugeren "tyge" kunne genstarte NFS-serveren. Find linjen med

```
root    all=(ALL) ALL
```

Under denne tilføjer du, at brugeren "tyge" på maskinen "hven" må køre kommandoen **/etc/rc.d/init.d/nfs restart**.

```
tyge      hven=/etc/rc.d/init.d/nfs restart
```

Gem filen og prøv nu som brugeren "tyge" at genstarte nfs.

Det eneste trick er, at du, som almindelig bruger, skal skrive sudo foran den kommando, du skal kunne køre med root-rettigheder. Efter en lille formaning om at passe på skal "tyge" som almindelig bruger skrive sit *eget* passsword. Derefter udføres kommandoen, som om det var root, der gjorde det. Man skal skrive sit password som en sikkerhedsforanstaltning, der beskytter imod, at andre brugere kan udnytte ens sudo rettigheder. Hvis du f.eks. er gået til frokost uden at logge ud, så kan kollegaen ikke gå over og lave sudo kommandoer fra din maskine, da han stadig ikke kender dit password.

```
[tyge@hven ~]$ sudo /etc/rc.d/init.d/sendmail restart

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these two things:

        #1) Respect the privacy of others.
        #2) Think before you type.

Password:
```

```
Shutting down sendmail:                                    [  OK  ]
Starting sendmail:                                         [  OK  ]
```

Eksemplet er fra Red Hat, men kan være taget med en vilkårlig Linuxdistribution.

super er et andet program, som kan meget af det samme som sudo, og som kan hentes fra ftp://ftp.ucolick.org/pub/users/will. Umiddelbart har **super** flere muligheder, men det er sværere at anvende end **sudo**.

Endelig skal det siges, at programmet **sudo** også er smart, selvom du har spredt root password på flere hænder, idet det tvinger brugeren til at være lidt forsigtig med root-kontoen. Som afslutning vil vi dog påpege, at hvis du tildeler personer rettigheder via **sudo**, så bør det ikke være ene generelt blankocheck til at kunne køre alt som root. Istedet bør du anføre hver af de services, der skal kunne startes og stoppes via **sudo**. En god grund til dette er, at du så ikke havner i en sitation, hvor dine basale binære programmer eller bibioteker er bliver udskiftet pga. misforståelser eller destruktiv handling. Hvis en person virkelig skal være root ofte, så er det måske klogere, at personen også har root-password og et tilsvarende ansvar.

3.2. Valg af password

Det er vigtigt at hemmeligholde sine passwords - især sit root password. På en Linux-maskine logger man ind med et bruger-id såsom "tyge" og et tilhørende password. Vi vil se på, hvorfor det er vigtigt, at du beskytter dit password, og hvordan du kan beskytte det. Vi vil se nærmere på, hvad der sker med dit password på Linux-maskinen, og på nogle af de værktøjer, du kan bruge til at højne sikkerheden omkring passwords.

3.2.1. Skriv ikke dit password, hvor andre kan se det

Styrer du en Linux-maskine med et antal brugere, der har hver sin login-konto, så er det vigtigt, at de alle forstår, at de hver især er ansvarlige for deres password. Hvis de skriver det på en lap papir og sætter den på opslagstavlen eller på skærmen, er de med til at bryde sikkerheden. Man kan grine af dette, men det er langt mere udbredt, end man skulle tro. Sagen er, at en vilkårlig anden person, som kender din kombination af login-navn og password, kan logge ind som *dig* og arbejde som dig, misbruge, ødelægge eller spionere i dit navn. Det skal selvfølgelig undgås.

3.2.2. Hvordan vælger man passwords ?

Ud over at password ikke må skrives, hvor andre kan se det, så er det ikke lige meget, hvordan du vælger dit password og specielt ikke dit root password. Desuden bør man skifte password med passende mellemrum. Passwords kan knækkes ved en kombination af gode gæt og rå beregningskraft. Før vi ser på dette, så lad os se på, hvordan dit password er gemt på på din Linux-maskine.

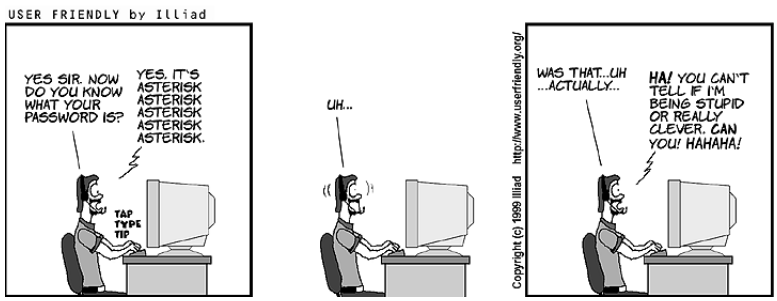
Dit password og information om dit login-navn og gruppe gemmes i filen `/etc/passwd`. Hvis du ikke anvender *shadow passwords* (hvad det er kommer vi ind på senere), så kan en linje i passwd-filen ligne følgende:

```
tyge:A1$HN1r2zxs$A2rmawluybzv8hkf4Hpzz0:501:501::/home/tyge:/bin/bash
```

De første to felter er dit login-navn og dit krypterede password. Når du ændrer password (med kommandoen passwd), så vil Linux-maskinen kryptere dit password til noget meget ukendeligt og gemme dette i `/etc/passwd`. Ideen bag dette er, at man har en algoritme, som hurtigt kan generere noget unikt krypteret tekst ud fra en anden tekst såsom dit password, men at det i praksis er umuligt at gå den anden vej. Det vil sige at der ikke findes en metode, hvor man finde det originale password ud fra det krypterede password. Som et lille matematisk eksempel kan vi nævne, at man nemt kan finde y hvis man kender x, ud fra y = x^3+2*x^2-3*x+2, mens det er langt mere besværligt at finde x ud fra y - i dette eksempel skal man løse en tredie grads ligning. De metoder, man anvender til at kryptere passwords, er meget smartere end bare en trediegradsligning. Metoderne er udviklet, således at man garanterer, at der kun er et password, som svarer til det krypterede password.

Fra den mere muntre afdeling kan vi lige igen vise en stribe fra http://www.userfriendly.org/static

Figur 3-2. User Friendly



3.2.3. DES kryptering af passwords

På det fleste UNIX systemer har det i mange år været standard, at et password måtte være op til otte tegn langt, og ud fra dette blev der gemt 13 krypterede tegn i din password-fil. På Linux har man længe anvendt crypt, som benytter sig af DES (Data Encryption Standard). Læs mere om DES på http://csrc.ncsl.nist.gov/cryptval/des/des.txt.

Funktionen crypt bruger det indtastede password som krypteringsnøgle. Den tekst, der krypteres, er blot en række nuller. Men det er ikke hele passwordet, der bruges som DES-nøgle, den sidste bit i hvert tegn smides væk. Desuden sættes 2 tilfældige tegn ind, det såkaldte "salt", der blot er beregnet til at gøre det sværere at rekonstruere passwordet. Se i øvrigt **man crypt**.

3.2.4. MD5

I gamle dage, hvor man ikke havde så store computere, var crypt en god løsning til passwords, da den er nærmest umulig at bryde på kort tid. Men i dag har man rå computerkraft nok til, at man ikke behøver at bryde krypteringen. Man kan i stedet sætte sine computere til at prøve sig frem fra en ende af. Derfor er der i dag brug for længere passwords, som er sværere at bryde.

Linux Red Hat 6.0 og senere har en langt mere veludviklet håndtering af passwords end tidligere UNIX systemer, hvis man har valgt at kryptere med MD5-tjeksummer. Man kan nu med MD5 vælge passwords, der har mere end otte tegn, og uanset længden af det man taster ind, gemmes der altid 34 krypterede tegn i password-filen. Det er langt sværere at knække de lange passwords.

Det, vi ønsker med vores avancerede kryptering, er, at der kun er en måde, hvorpå du kan få det ukrypterede password ud fra det krypterede - ved at indtaste det korrekte password. Er du cracker, må du prøve at gætte på passwordet, kryptere alle disse passwords og sammenligne det krypterede gæt med det rigtige krypterede password. Er de to krypterede passwords ens, så er gættet og det rigtige ukrypterede password ens - du kender nu passwordet og kan logge ind på systemet som den pågældende bruger. Derfor bør man vælge et password, der er svært at gætte. Ydermere kan vi igen anbefale, at du bruger shadow passwords, for så har man ikke adgang til det krypterede password.

3.2.5. Jeg knækker dit password

Der findes programmer, som kan bruges til at knække passwords effektivt. Man kan måske mene, at det er med til at svække sikkerhed generelt, at der ligger programmer frit tilgængeligt på internettet, som gør det nemt at cracke passwords. Vi mener, at det er for enkelt at tænke sådan. Enhver kan i princippet skrive et program til at knække passwords, og derfor bør man måske selv lade sit password komme igennem sådan et program. Hvis det ikke er knækket indenfor en måned, er det ikke så ringe... Men så er det alligevel på tide at skifte det ud. Et udbredt program til at knække passwords er *John the Ripper*, der kan findes på http://www.false.com/security/john/ (http://www.openwall.com/john/). Man kan finde programmer, som er en del hurtigere til blindt at knække passwords fra en ende af, men John the Ripper har nogle spændende aspekter. Nedenfor er vist, hvordan programmet knækker passwords for brugeren "tyge" i med tre forskellige passwords, hhv. "a", "ab" og "qsw". Udskriften stammer fra en 300 MHz PII, og det viser, at et bogstav knækkes på få sekunder, mens to og tre tilfældige bogstaver kan knækkes på 5 hhv. 8 minutter. Generelt så bliver det meget langsommere at knække et passwords for hver gang at der kommer et tegn mere i passwordet. Derfor bør man altid vælge passwords med mere end syv tegn.

```
Loaded 1 password (FreeBSD MD5 [32/32])
a                (tyge)
guesses: 1  time: 0:00:00:06 100% (2)  c/s: 746  trying: a

Loaded 1 password (FreeBSD MD5 [32/32])
ab               (tyge)
guesses: 1  time: 0:00:05:12 (3)  c/s: 665  trying: ab

Loaded 1 password (FreeBSD MD5 [32/32])
qsw              (tyge)
```

```
guesses: 1  time: 0:00:08:08 (3)  c/s: 669  trying: qsw
```

John the Ripper er lidt langsom til de "tilfældige" passwords ovenfor, fordi det er programmeret ud fra, hvordan mange brugere i praksis vælger password. Man vælger ofte kærestens navn, måske koblet med en fødselsdag, et sted man er glad for, eller andre ting man kan huske. Det er ikke klogt, fordi det er for nemt at gætte, hvilket følgende viser. Først har vi ladet brugeren "tyge" have password "abc" som står i alle ordbøger. Det koster kun 4 sekunder, før det er fundet. Dernæst er vist, at fem bogstaver som i ordet "apple" findes på 3 sekunder, og endelig gentagelsen "appleapple" med 10 bogstaver som kun tager 19 sekunder at knække - skræmmende, ikke sandt? Ordbøger findes til alle sprog, så vælg altid passwords som *ikke* står i en ordbog - bland tal ind i ord og lav et underligt system, andre ikke har en chance for at gætte. Brug f.eks. forbogstaver fra en sætning eller en sang, og flet specialtegn og numre med ind. Vær dog lidt varsom med specialtegn i passwords - specialtegnenes placering på tastaturet kan variere alt efter hvilket land, tastaturet er sat op til. Man kan f.eks. komme ud for et dansk tastatur, som er sat op som et amerikansk, hvor det kan det være ret svært at finde specialtegnene. Jo længere password du vælger des bedre - og altid på mere end syv tegn. Root passwordet skal helst være endnu længere og vælges med særlig omhu. I øvrigt bør du med jævne mellemrum ændre password, men sørg for at dette sker enten på selve maskinen eller via en krypteret adgang til maskinen såsom ssh (secure shell).

```
Loaded 1 password (FreeBSD MD5 [32/32])
abc              (tyge)
guesses: 1  time: 0:00:00:04 100% (2)  c/s: 837  trying: abc

Loaded 1 password (FreeBSD MD5 [32/32])
apple            (tyge)
guesses: 1  time: 0:00:00:03 100% (2)  c/s: 891  trying: apple

Loaded 1 password (FreeBSD MD5 [32/32])
appleapple       (tyge)
guesses: 1  time: 0:00:00:19 100% (2)  c/s: 710  trying: appleapple
```

3.2.6. Shadow files

Den almindelige password-fil, `/etc/passwd`, kan læses af alle. Dette er nødvendigt, da en del programmer bruger filen til at koble en brugers bruger-id (tredie felt i password-filen) med det tilhørende brugernavn. At alle kan læse filen betyder imidlertid også, at alle kan se dit krypterede password. Derfra kan man cracke dit password, og som vi har beskrevet, så kan det gøres hurtigt, hvis du har valgt et svagt password. Med flere Linux distributioner bl.a. Red Hat 6.2 bliver du ved installationen spurgt, om du vil anvende shadow passwords, hvilket du bør svare ja til. Når du har installeret dette, så vil du se, at der står et "x" i `/etc/passwd`, hvor dit krypterede password før ville have stået:

```
tyge:x:501:501::/home/tyge:/bin/bash
```

Dit krypterede password er nu flyttet til `/etc/shadow`, som kun kan læses af root - dvs., ingen almindelig bruger på maskinen nu kan læse dit krypterede password. Hvis du har installeret Linux, men ikke har shadow passwords slået til, kan det gøres med kommandoen **/usr/sbin/pwconv**, som skal køres som root. Den laver shadow filen ud fra password filen og tilsvarende laves en shadow gruppe-fil

`/etc/gshadow` ud fra `/etc/group` med programmet **/usr/sbin/grpconv**. For at dette virker, skal dine Linuxprogrammer være oversat til at kunne håndtere dette - tidligere var dette ikke altid tilfældet. Læs manualsiden for pwconv for detaljer.

I det ovenstående, hvor vi skriver, at alle kan læse password-filen, går vi ud fra, at det er brugere med lokal adgang. Hvordan får en cracker udefra adgang til min password-fil, så han kan se mit krypterede password? Ofte er det CGI-scripts på en web-server, som pga. simple programmeringsfejl eller pga. fejl i de anvendte programmeringsssprog kan lokkes til at vise de krypterede passwords fra password-filen. Dygtige crackere finder fra tid til anden nye metoder til at gøre dette. Normalt findes tilsvarende rettelser til disse huller - hold derfor altid din maskine opdateret.

3.3. SUID root programmer

SUID betyder, at et program kører "som sin ejer", og ikke "som" den bruger, der udfører det. Dvs. at det kører med ejerens rettigheder. Et program, som "tyge" ejer, og som er SUID, har f.eks. ret til at skrive i "tyge"s hjemmekatalog, selvom det er en anden bruger, der kører det. Et SGID program er det samme bare med gruppe rettigheder i stedet.

Hvorfor er det farligt? SUID er specielt farligt, når det er et SUID-program som root ejer. Så længe programmet opfører sig pænt, er det ikke noget problem. Men hvis der er fejl eller sikkerhedshuller i programmet, kan det være en trussel mod sikkerheden. Man kan forestille sig, at et program har et sikkerhedshul, som gør det muligt for en almindelig bruger at gå ind og overskrive noget af programmets hukommelse, imens det kører, og få det til at gøre noget andet, end det skal. Så har denne bruger faktisk root adgang til systemet. Man har set eksempler på dette med efterfølgende sikkerhedsopdateringer til følge.

Lad os nu se på hvilke programmer, der på en normal Linux-maskine er SUID programmer.

```
[tyge@hven ~]$ su - root
[root@hven /root]# find / -perm +4000
```

Sådan finder du alle SUID programmer. Men det er kun SUID root programmer, som vi vil være bange for i dag. Hvis andre brugere selv laver SUID programmer så lad os antage, at de ved, hvad de laver og selv tager eventuelle konsekvenser. Vi går nu efter de programmer, hvor root er ejeren og som er SUID. Så vi tager parametrene "-user root" med til find kommandoen:

```
[tyge@hven tyge]$ su - root
Password:
[root@hven tyge]#  find / -perm +4000 -user root
/bin/ping
/bin/mount
/bin/umount
/bin/su
/bin/login
/sbin/dump
```

```
/sbin/restore
/sbin/pwdb_chkpwd
/sbin/cardctl
/usr/bin/rcp
/usr/bin/rlogin
/usr/bin/rsh
/usr/bin/at
/usr/bin/dos
/usr/bin/chage
/usr/bin/lpq
/usr/bin/lpr
/usr/bin/lprm
/usr/bin/passwd
/usr/bin/suidperl
/usr/bin/procmail
/usr/bin/screen
/usr/bin/nwsfind
/usr/bin/chfn
/usr/bin/chsh
/usr/bin/newgrp
/usr/bin/crontab
/usr/bin/zgv
/usr/bin/gpasswd
/usr/bin/sperl5.00405
/usr/X11R6/bin/xterm
/usr/X11R6/bin/XConsole
/usr/X11R6/bin/nxterm
/usr/X11R6/bin/xscreensaver
/usr/X11R6/bin/Xwrapper
/usr/lib/news/bin/startinnfeed
/usr/local/bin/ssh1
/usr/sbin/usernetctl
/usr/sbin/inndstart
/usr/sbin/sendmail
/usr/sbin/traceroute
/usr/libexec/pt_chown
find: /proc/1144/fd: Permission denied
find: /proc/1145/fd: Permission denied
find: /proc/6286/fd/4: No such file or directory
/opt/kde/bin/kvt
/opt/kde/bin/kppp
```

Outputtet er de SUID root programmer, der findes på systemet. Der er også et par fejl fra find nede fra /proc kataloget. Det skal du ikke tage dig af, /proc er et interface til den kørende kerne, og der ligger ikke almindelige filer (f.eks. SUID root programmer) der. Det er *mange* programmer at passe på, måske er der et sikkerhedshul i nogle af dem. Har du brug for alle disse programmer? Er der mon nogle af dem, der kan køre uden SUID root eller helt fjernes? Tjek programmernes man page, tjek om andre programmer er afhængige af SUID-programmet, som det f.eks. er tilfældet med **/bin/su**. Det kan ofte undersøges med systemets pakkemanager. Er der sikkerhedsopdateringer til nogle af programmerne, du burde installere? Kan du overskue at følge med i sikkerhedsopdateringer for alle disse programmer? Konklusionen er, at du aldrig skal installere flere programmer, end der skal bruges.

SGID - set group ID - er også farligt, hvis gruppen er root. Det er ikke så almindeligt at anvende SGID

```
[root@hven tyge]#  find / -perm +2000 -group root
/sbin/netreport
/usr/sbin/sendmail
find: /proc/1144/fd: Permission denied
find: /proc/1145/fd: Permission denied
find: /proc/6296/fd/4: No such file or directory
```

Der var ikke så mange, men dem skal man også være opmærksom på.

Lad os som et eksperiment prøve at lade **/bin/ping** være ejet af "tyge" i stedet for "root", og lad os se om den stadig virker:

```
[root@hven tyge]# ls -l /bin/ping
-rwsr-xr-x   1 root     root        14116 Jun 18  1998 /bin/ping
[root@hven tyge]# chown tyge /bin/ping
[root@hven tyge]# ls -l /bin/ping
-rwsr-xr-x   1 tyge      root        14116 Jun 18  1998 /bin/ping
[root@hven tyge]# ping 10.10.10.3
ping: ping must run as root
```

Ups, det kunne man ikke. Vi må hellere skifte tilbage:

```
[root@hven tyge]# chown root /bin/ping
```

Programmet ping er svært at undvære og er nødt til at køre som SUID root. Du kan i øvrigt se, at det er SUID ved det "s" som kommer frem, når du kører **ls -al** på filen. Programmet **/usr/bin/passwd** er svært at undvære, og det er nødt til at køre som root for at kunne ændre i /etc/passwd filen. Et program som kppp kunne derimod afinstalleres, hvis du ikke bruger det. Kppp er et KDE program, der bruges til at koble sig til internettet via modem. Tilsvarende kan du afinstallere **/sbin/cardctl**, hvis du ikke har PCMCIA kort i din maskine. Anvender du en RPM baseret Linuxdistribution såsom Mandrake, SuSE eller Red Hat, kan du have glæde af at finde ud af fra hvilken pakke, et givent SUID program kommer fra.

```
[root@hven root]# rpm -qf /sbin/cardctl
kernel-pcmcia-cs-2.2.5-15
```

Så kan du tjekke hvilke filer, pakken indeholder. Når du frem til, at pakken ikke bruges, så afinstaller den:

```
[root@hven root]# rpm -ql kernel-pcmcia-cs-2.2.5-15
...klip mange linjer
[root@hven root]# rpm -e kernel-pcmcia-cs-2.2.5-15
```

SUID root programmer er en alvorlig sikkerhedsrisiko, og man bør i hvert fald ikke lave SUID root programmer selv for at løse en given opgave. Der er ting man ikke har fundet smartere løsninger på endnu, men de fleste ting kan gøres uden. I Linux har man som en sikkerhedsforanstaltning overfor SUID root programmer indbygget, at et script (tekstfil med kommandoer) ikke kan køres som SUID root.

Kapitel 4. Remote login og netværksaflytning

En af de store styrker ved et Linux-system (eller et andet Unix-system) er, at man kan administrere den fra en anden maskine via netværk. For at være kompatibel med alle andre UNIX varianter følger de gode gamle værktøjer **telnet** og ftp med i Linux-distributionerne, og de er meget anvendte til fjernadministration. Vi vil i dette kapitel se på, hvorfor værktøjer som **telnet** og ftp ikke bør anvendes, hvis maskinen er koblet til internettet eller et andet usikkert netværk. Vi skal se på, hvordan netværkstrafik kan aflyttes. Derefter skal vi se nærmere på alternativer til **telnet** og ftp, hvor datastrømmen bliver krypteret. Specielt fokuserer vi på OpenSsh (den frie version af secure shell) og viser installation og anvendelse. Meget af det, der beskrives i artiklen, gælder ikke blot for Linux, men også for andre UNIX'er.

4.1. Nem men usikker netværkstrafik

Fra en Linux-maskine kan man nemt logge ind på en anden Linux-maskine og køre programmer, endog grafiske programmer. Dermed kan man køre tunge programmer på en stærk server og få vist resultater på en anden (måske langsommere) maskine.

Antag, at vi har et lokalnet bestående af tre maskiner. Vi anvender maskinen "hven" (IP-adresse 192.168.0.1) med brugernavnet tyge, men vi vil køre programmer fra maskinen "bohr" (192.168.0.2). I netværket finder vi desuden maskinen "nottingham" (192.168.0.3), som vi leger er en ondsindet maskine, der ønsker at bryde vores sikkerhed. I praksis kunne det være tre maskiner på internettet, hvor netværkstrafik mellem "hven" og "bohr" tilfældigvis også kommer forbi "nottingham".

Resten af kapitlet vil handle om programmer til remote login, som **telnet**, **rlogin** og ssh, samt programmer til filoverførsel (ftp og scp). Men hvad bruges det til? Med remote login kan man udføre tekstkommandoer på den maskine, man er logget ind på, men man kan også køre X-programmer over nettet.

4.1.1. telnet og xhost

Hvis du på hven kører en X-baseret grafisk brugergrænseflade, og du vil køre X-programmer (grafiske programmer) fra maskinen "bohr", skal du starte med at fortælle maskinen "hven", at det er i orden, at maskinen "bohr" benytter dens display. Dette gøres ved at føje "bohr" til listen med godkendte X-klienter med kommandoen **xhost**:

```
[tyge@hven ~]$ xhost +bohr
bohr being added to access control list
```

Dermed vil grafiske programmer fra "bohr" blive accepteret af "hven". Udelades maskinnavnet, betyder det, at alle maskiner kan vise grafik på din skærm. Lad være med det, da det sikkerhedsmæssigt er en ekstremt dårlig ide.

Lad os nu logge ind på "bohr" med **telnet**,

```
[tyge@hven ~]$ telnet bohr
Trying 192.168.0.2...
Connected to bohr.herne.dk.
Escape character is '^]'.
Debian GNU/Linux 2.1 bohr.herne.dk

bohr login: tyge
Password:
```

Efter at have skrevet brugernavn og adgangskode på bohr-maskinen får du en kommandolinje, og du kan udføre programmer på maskinen, som om du var logget ind lokalt.

For at kunne få de grafiske programmer, du starter på bohr, til at vise sig på hvens skærm er det nødvendigt at sætte systemvariablen `DISPLAY`:

```
[tyge@hven ~]$ export display=hven:0.0
```

Mange X-programmer kan dog også kaldes med "-display" som option.

Nu kan du køre dit X-program, f.eks. "xload", som om du sad på bohr. Programmet kører på bohr, men alt grafik vises på hven, og programmet styres fra hven.

```
[tyge@bohr tyge]$ xload -display hven:0.0 -geometry 60x60 -nolabel &
```

Figur 4-1. xload

Problemet er ikke at du nu kan kalde grafik op på en anden skærm end din egen. Problemet er, at den adgang du har etableret med **xhost** så at sige går begge veje. I og med du har lavet **xhost +bohr** har alle brugere på bohr adgang til at læse og skrive fra og til dit dislay på "hven". Er bohr et multibrugersystem, og har en cracker, eller blot en nysgerrig kollega konto på maskinen, kan vedkommende køre kommandoen **xwd -display hven:0.0 -root -out hven.dump** efterfulgt af **xwud -in hven.dump**. Den første giver vedkommende et skærmdump af hven, den næste viser dette billede lokalt (på bohr). Alt hvad du kan se på skærmen på hven kan enhver på bohr tage et dump af.

Selv hvis hven har **xhost -** kan en bruger, der har lokal shell-adgang på hven, i visse tilfælde stadigvæk køre xwd-kommandoen! Vedkommende skal så blot være logget ind på hven, når kommandoen udføres - det kan ikke ske over nettet som beskrevet ovenfor.

Det bliver værre: Med **xlswins** kan man vise hvilke vinduer (xterm, netscape osv.) der er åbne på den anden maskine, og så dump disse enkeltvis med **xwatchwin**. Begge programmer er enten med i standard X, eller fås kvit og frit på internettet.

Og værre: Programmet **xscan** skal efter sigende kunne lave et tilsvarende trick, men med tastetryk. Direkte til en logfil, alt hvad der tastes på maskinen, der har **xhost**: brugernavne, adgangskoder, pgp-koder.

xhost er generelt en meget dårlig ide, og bør i alle tilfælde erstattes med SSH, der automatisk og transparent sørger for at sætte display og X rettigheder, så man hele tiden "har sit display med sig".

4.1.2. Pas på adgangskoder sendt over netværk

For at aflytte netværkstrafikken kan man hente programmet sniffit til Linux. Når det startes op vises alle kommunikationslinjer, såsom telnet- og ftp-forbindelser, mellem maskinerne på netværket. Det kommer an på opsætning af routere, firewalls, switche og hubs, hvor meget man reelt får at se. Sniffit kan hentes fra http://reptile.rug.ac.be/~coder/sniffit/sniffit.html.

Det er en udbredt misforståelse at man ikke kan aflytte switchede netværk - men kun netværk med hubs. Dette er forkert og aflytningen kan typisk foretages ved hjælp af ARP spoofing - eksempelvis med arpspoof programmet der følger med dsniff. Link: http://www.monkey.org/~dugsong/dsniff/ (http://www.monkey.org/~dugsong/dsniff/). Dsniff kan opsamle og afkode traffik fra mange usikre protokoller. Sniffit og dsniff skal køres som root.

Lad os antage, at den ondsindede "nottingham" (192.168.0.3) lytter med på, hvad vi laver mellem "hven" (192.168.0.1) og "bohr" (192.168.0.2). Programmet sniffit startes i interaktiv tilstand med angivelse af hvilket netværks-interface, der skal lyttes på:

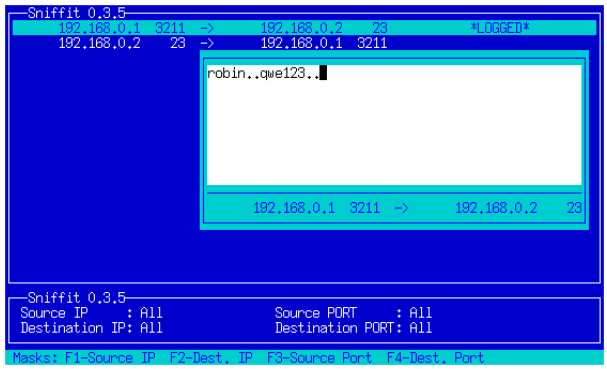
```
[root@nottingham root]#  sniffit -F eth0 -i
```

hvor eth0 betyder første ethernet kort i maskinen, og i betyder interaktiv tilstand. Ud kommer nedenstående billede, hvor man ser, at maskinen med netværksadresse 192.168.0.1 (hven) har oprettet en forbindelse til port 23 på maskinen med netværksadresse 192.168.0.2 (bohr). Port 23 er den port, telnet lytter på. Port 3211, som man sender fra, er valgt blandt maskinens ledige porte, dvs. de porte, der ikke er nogen service, der lytter på. Der er valgt et portnummer, som er større end 1024, dvs. en ikke priviligeret port.

Trykkes return på en linje vil den blive markeret med "*LOGGED*", og det mindre vindue mod højre vil vise trafikken på den forbindelse. Først kommer login navn: "tyge", og efter to punktummer kommer

adgangskoden i klar tekst: "qwe123". Der skal ikke stor fantasi til at forestille sig, at dette program nemt kan sættes til at dumpe samhørende brugernavne og adgangskoder ned i en fil over en periode, indtil der er gevinst.

Figur 4-2. Sniffit



Vi har vist, at man ikke skal bruge **telnet** til at kommunikere mellem to maskiner, hvis der er risiko for, at nogen lytter med, og slet ikke hvis man skal logge ind som root på maskinen. Det er simpelthen for nemt at lede efter loginnavn root og derefter få adgangskoden. Derfor har mange Linux-distributioner per default forhindret, at root logger ind via netværket. Red Hat 6.0, Debian 1.3 og SuSE 6.2 har filen "/etc/securetty", der indeholder de konsoller, hvor root må logge ind. Dette kan være "tty1" til "tty6", som er de tekst login konsoller, der normalt er på selve Linux-maskinen - dvs. ikke via netværket. Hvis man absolut vil tillade root login via netværk, kan man tilføje "0", "1" og opefter (for Linux-kerne 2.2.X) svarende til hvor mange logins, du forventer på maskinen fra netværket. Normalt er det *ikke* klogt at tillade direkte root login via netværk.

4.1.3. ftp

Hvis vi vil overføre filer fra "hven" (192.168.0.1) til "bohr" (192.168.0.2), er ftp et af de gennemprøvede og gamle værktøjer. Også her er der sikkerhedsproblemer. Vi sætter igen "nottingham" (192.168.0.3) til at køre sniffit som vist nedenfor. Denne gang sætter vi sniffit op til at vise alle pakker som ankommer til "bohr" (192.168.0.2), og man kan nemt se, hvad der sker.

```
[root@nottingham root]#  sniffit -t192.168.0.2 -a -F eth0

Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1299-192.168.0.2.21
 E . . 4 . . @ . @ . . . . . . . . . . . . . . . . . b . . . . . P . } x i !
 . . U S E R   r o b i n . .

Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1299-192.168.0.2.21
 E . . ( . . @ . @ . . . . . . . . . . . . . . . . . b ( . . . . P . } x . .
 . .
```

```
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1299-192.168.0.2.21
 E . . 5 . . @ . @ . . . . . . . . . . . . . . . . . b ( . . . . P . } x N D
 . . P A S S   q w e 1 2 3 . .
```

På den første af de tre data-linjer kan man se login navn (tyge), og som vist på den sidste data-linje, vil man direkte kunne læse adgangskoden (qwe123). Ser man lidt nøjere efter, vises også ftp-porten, port 21. Port 1299 er ligesom ved **telnet** en ledig port, der vælges til denne session.

Som vi ser, er ftp også et meget usikkert program, hvor andre kan lytte med. Så brug det med omtanke. For en systemadministrator er paranoia ikke en sygdom, men en kvalifikation...

Det skal nævnes, at også resten af den nettrafik, du laver med **telnet** og ftp, kan læses i klar tekst via f.eks. sniffit. Starter du andre programmer, kan man se det. Laver du "su - root" og skriver "root"s adgangskode, er sikkerheden på den maskine væk, for "root"s adgangskode er ude, hvis du bliver aflyttet.

4.1.4. Remote login, remote shell og remote copy

To andre velkendte programmer i samme kategori er "**rlogin**" og "rsh". Programmerne virker næsten ens. Begge giver interaktivt login på maskinen, men rsh kan udføre en kommando samtidig med, at man logger ind. F.eks. vil den følgende kommando logge ind direkte fra hven til bohr og køre kommandoen "df" (som viser hvor fyldte dine diske er).

```
[tyge@hven ~]$ rsh bohr df
```

Nu prøver vi at køre sniffit på en **rlogin** session for at se, hvad man kan se. På maskinen "nottingham" kører vi sniffit:

```
[root@nottingham root]# sniffit -x -s192.168.0.1 -a -F eth0
```

Vi logger ind fra hven til bohr med kommandoen:

```
[tyge@hven ~]$ rlogin bohr
Password: qwe123
$
```

hvor adgangskoden naturligvis ikke kan ses på skærmen, når man taster den ind. Udklip fra sniffit's output:

```
TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA41   ACK (hex): 105DA03F
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ? s . @ . @ . F Q . . . . . . . . . . . . . . . A . ] . ? P . } x C .
 . . r o b i n . r o b i n . x t e r m / 9 6 0 0 .
```

```
TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA58   ACK (hex): 105DA040
   FLAGS: -A----   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ( s . @ . @ . F g . . . . . . . . . . . . . . . X . ] . @ P . } x } A
 . .


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA58   ACK (hex): 105DA041
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . 4 s . @ . @ . F J . . . . . . . . . . . . . . . X . ] . A P . } x . .
 . . . . s s . . . P . . . l


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA64   ACK (hex): 105DA04B
   FLAGS: -A----   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ( s . @ . @ . F U . . . . . . . . . . . . . . . d . ] . K P . } x } *
 . .


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA64   ACK (hex): 105DA04B
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ) s . @ . @ . F S . . . . . . . . . . . . . . . d . ] . K P . } x . !
 . . q


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA65   ACK (hex): 105DA04B
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ) s . @ . @ . F R . . . . . . . . . . . . . . . e . ] . K P . } x .
 . . w


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA66   ACK (hex): 105DA04B
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ) s . @ . @ . F Q . . . . . . . . . . . . . . . f . ] . K P . } x . .
 . . e


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA67   ACK (hex): 105DA04B
   FLAGS: -AP---   Window: 7D78
```

```
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ) s . @ . @ . F P . . . . . . . . . . . . . . . g . ] . K P . } x L .
 . . 1


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA68   ACK (hex): 105DA04B
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ) s . @ . @ . F O . . . . . . . . . . . . . . . h . ] . K P . } x K .
 . . 2


TCP Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
   SEQ (hex): C2D0BA69   ACK (hex): 105DA04B
   FLAGS: -AP---   Window: 7D78
Packet ID (from_IP.port-to_IP.port): 192.168.0.1.1023-192.168.0.2.513
 E . . ) s . @ . @ . F N . . . . . . . . . . . . . . . i . ] . K P . } x J .
 . . 3
```

Igen ser man tydeligt brugernavn og adgangskode blive sendt ukrypteret over nettet (se sidste tegn på hver linje). Man kan se, at **rlogin** på bohr bruger port 513, og at der på hven sendes fra port 1023.

Skal du kopiere filer mellem to maskiner, så er ftp som tidligere skrevet meget anvendt. Alternativt kan remote copy **rcp** anvendes. Skal du kopiere filen `.emacs` fra hven til bohr, sker dette med

```
[tyge@hven ~]$ rcp .emacs bohr:
```

Sikkerhedsmæssigt er **rcp** på linje med **rlogin** og **rsh**. Vi skal senere i artiklen vende tilbage til et fuldt krypteret alternativ til **rcp**.

Med **rlogin**, rsh og rcp kan man vælge, at login kan ske uden adgangskode. Dette gøres ved at sætte sit hostnavn og evt. brugernavn ind i filen `~/.rhosts` i sit hjemmekatalog på den maskine, der skal logges ind på. Hvis du gerne vil kunne logge ind fra hven til bohr uden adgangskode, som brugeren tyge, kan `.rhosts`-filen i dit hjemmekatalog på bohr se sådan ud:

```
hven tyge
```

Der kan godt stå mange flere linjer med andre hostnavne. Brugernavnet kan udelades, idet brugeren sættes til den, hvis hjemmekatalog filen ligger i. Hvis filen er læsbar for alle, er det kun en bruger med samme brugernavn som sit eget, der kan logge ind fra en anden maskine. Er filen ikke læsbar for andre, kan man anføre linjer i `.rhosts`-filen med først maskinnavn og dernæst det brugernavn, som anvendes på den anden maskine. For at "peter" skal logge på ind på "hven" som "tyge", kan man således bruge **rlogin hven -l tyge**.

Den viste `.rhosts`-fil vil betyde, at du fra maskinen "hven" som brugeren "tyge", godt må komme ind på bohr som "tyge", uden adgangskode. Det kræver, at brugeren findes på begge maskiner, og ".rhosts"

filen skal ligge i brugerens hjemmekatalog på den maskine, man prøver at logge ind på. Det er nemt og giver lynhurtig adgang til, at man hopper fra maskine til maskine og laver meget effektive arbejdsgange. Det lyder smart, men medaljen har en bagside. En med root-adgang på maskinen "nottingham" (eller andre maskiner i netværket) kan aflytte rsh/rlogin netværkstrafik et stykke tid, og derved finde ud af fra hvilken maskine og med hvilket bruger navn, man kan logge ind direkte. Derefter kan man med lidt snilde sætte maskinen "nottingham" lidt anderledes op, så "bohr" tror, at den er "hven".

Der kan imidlertid være situationer, hvor det kan virke mere sikkert at lade en bruger logge ind uden adgangskode fra en kendt host end at lade brugeren have en adgangskode. Har man tillid til sit interne netværk, og er det nødvendigt, at nogle brugere kan logge på en udsat maskine, kan man vælge, at disse brugere ikke har nogen adgangskode på den udsatte maskine. I stedet får de en stjerne ("*") i `/etc/passwd`, der hvor adgangskoden normalt ville stå. Dette vil forhindre dem i at logge ind fra andre maskiner end dem, der er angivet i deres `~/.rhosts` fil i deres hjemmekatalog på den udsatte maskine. Når en bruger ikke har en adgangskode, kan man ikke bryde ind på den udefra ved at gætte eller knække adgangskoden. Men kan kun logge ind som denne bruger, hvis man kan overbevise maskinen om, at man logger ind fra en betroet maskine. Der kan i øvrigt læses mere om adgangskodebeskyttelse og håndtering i Kapitel 3.

Hvis man ikke tillader `.rhosts`-filer, vil der ved hver login sendes adgangskoder i klar tekst over nettet. Hvis der er `.rhosts`-filer, bliver der slet ikke sendt adgangskoder over nettet - i stedet for at give sin adgangskode skal man logge ind fra en betroet host. Ingen kan nu opsnappe adgangskoder - til gengæld kan de udgive sig for at være den betroede host. Stoler man ikke på sit interne netværk, eller er nogen *sluppet ind*, er begge dele nok lige dårligt.

Vi vil anbefale, at man i stedet for **rlogin** bruger et program, der krypterer login såvel som selve dataoverførslen, som f.eks. SSH giver mulighed for.

Som systemadministrator kan du se om dine brugere har `.rhosts` filer, og hvis det ikke er ønsket, kan du slette dem (og skælde brugeren ud). Hvis brugerne har hjemmekataloger under `/home/`, så kan dette gøres med

```
[root@hven root]# find /home/* -maxdepth 1 -name .rhosts -print;
```

Hvis man samtidigt vil slette de fundne .rhosts filer kan man anvende

```
[root@hven root]# find /home/* -maxdepth 1 -name .rhosts -print -exec rm {} \;
```

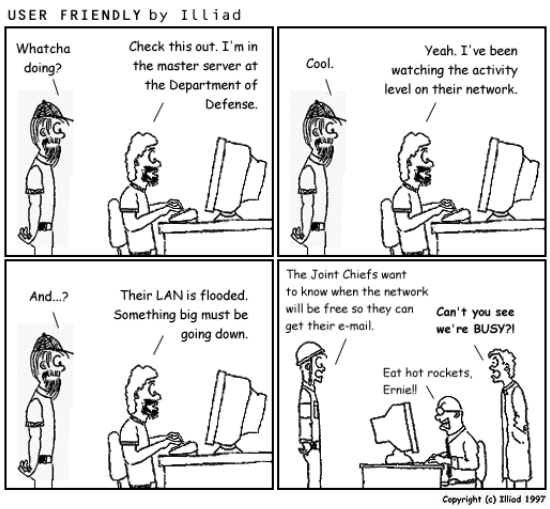
Det er enklere at disable brugernes egne `.rhosts`-filer ved, at man tilføjer parameteren `-l` efter `in.rshd` i filen `/etc/inetd.conf` og genstarter inetd-dæmonen.

Et andet argument imod at tillade ".rhosts"-filer er, at brugeren kan komme til at køre et program, som skriver til en fil uden hans viden. Filen kunne "~/.rhosts" og dermed give adgang til en ekstern bruger, som ikke behøver at have en konto på maskinen.

Et alternativ til "~/.rhosts" metoden er "hosts level equivalence" via filen "/etc/hosts.equiv". Dine brugere har ikke adgang til denne fil, og kun root kan rette i den. Root kan vælge at sætte en række hostnavne ind i denne fil eller evt. bare et plus. Det betyder, at disse hosts (+ betyder alle maskiner) har adgang til at udføre **rlogin** og rsh kommandoer uden adgangskode. Dette gælder alle brugere (dog ikke root) - brugeren skal dog findes på begge maskiner. Filen "/etc/hosts.equiv" kan være en bekvem løsning, men den er en bombe under systemsikkerheden - find en anden løsning, hvis du har et åbent eller halvåbent netværk.

Her i User Friendly fra den 12. december 1997 (http://www.userfriendly.org/cartoons/archives/97dec/19971212.html) kan Greg fra Columbia Internet aflytte netværkstrafikken i forsvarsministeriet...

Figur 4-3. User Friendly



Fortsættelsen kan findes på http://www.userfriendly.org/cartoons/archives/97dec/19971216.html :-)

4.2. Nem og sikker netværkstrafik

Hvis der er risiko for, at andre lytter med på netværkstrafikken, er der behov for erstatningsprogrammer for **telnet**, ftp og **rlogin**. Samtidig ønsker vi stadig at kunne afvikle programmer fra en maskine og se resultater på en anden, naturligvis også de grafiske programmer. Med andre ord ønsker vi samme funktionalitet - men med sikkerheden i orden.

Der findes flere sikre alternativer til de gamle, ukrypterede protokoller. Vi vil mest beskæftige os med SSH, men der findes også andre muligheder. Et alternativ til **telnet** er stelnet, som står for secure telnet.

Programmet baserer sig på SSL (Secure Sockets Layer), som er en måde at lave krypterering af datatrafikken. Kombinationen af stelnet og SSL er ikke så udbredt som SSH, og det er ikke så nemt at sætte op som SSH. Programmet stelnet kan findes på http://www.crufty.net/ftp/pub/sjg/ (http://www.crufty.net/ftp/pub/sjg/) og selve krypteringslaget SSL til Linux kan findes på http://www.psy.uq.oz.au/~ftp/Crypto/ (http://www.psy.uq.oz.au/~ftp/Crypto/). Denne sidste URL har også en hel del dokumentation.

Forhistorien er, at den gamle U.S. version af **telnet** allerede supporterede TELOPT_AUTHENTICATION og TELOPT_ENCRYPTION, men alle spor af krypteringskoden er fjernet fra den internationalt udbredte og anvendte **telnet**-kode.

Et andet alternativ til **telnet** og ftp, som er på vej, er SRP. Se mere om SRP på http://srp.stanford.edu/ (http://srp.stanford.edu/).

Det, der umiddelbart i dag er det bedste valg som erstatning for **telnet** og **rlogin**, er SSH (Secure SHell). Programmet er oprindeligt lavet af et finsk firma med navn SSH Communications Security med hjemmesiden http://www.ssh.fi. SSH har bl.a. den store fordel fremfor stelnet, at grafiske vinduer (XWindow) kan sendes over kryptererede linjer.

Det er et reelt problem, at den oprindelige kommercielle SSH ikke er Open Source (http://www.opensource.org/), og derfor er der startet et GNU-projekt under navnet PSST for at genskrive koden. Du kan finde mere information om dette på http://www.net.lut.ac.uk/psst. PSST's hjemmeside bør følges fra tid til anden, men status i øjeblikket (April 2003) er en fungerende version 1.5.1, og den siges at fungere sammen med OpenSSH. LSH kører kun ssh-protokol version 2, men roser sig af at have god kryptering og bedre random-generator for maskiner uden /dev/random. For at køre sammen med ssh kræves nogle scripts, fordi filformater for nøglefiler ikke er det samme.

En anden implementation af SSH-protokollen er OpenSSH, som er fri og er lavet af OpenBSD-folkene. Et separat hold programmører sørger for at lave en version for andre platforme, den portable version. OpenSSH er allerede lagt ind i OpenBSD-distributionen, der har ry for at være særdeles sikker. OpenSSH er ligeledes at finde i de fleste Linux-distributioner. Hjemmesiden for OpenSSH er http://www.openssh.com/. OpenSSH er baseret på OpenSSL, som skal oversættes først (hvis man vil bygge sin egen) Læs den medfølgende dokumentation, da der er nogle options for library generation, som ikke er helt trivielle. Den kan hentes under "support" samme sted, som OpenSSH findes, eller direkte fra http://www.openssl.org/. Man kan enten oversætte programmerne selv eller installere de allerede oversatte RPM-pakker.

OpenSSH er en godt projekt. Den omfatter en server, en klient og omfattende dokumentation af opsætningen. Den kan køre protokol version 2 og version 1. Hvis den er konfigureret til det, kan den endda starte rsh op, hvis alt andet fejler. Der er glimrende support fra OpenBSD teamet og det virker. Der er versioner af OpenSSH til alle Unix varianter også Linux, så i praksis er der ingen grund til at vælge den kommercielle SSH længere i forhold til OpenSSH.

Har du problemer med SSH, så læs http://sysadmin.oreilly.com/news/sshtips_0101.html. Der er et par

gode råd.

4.2.1. Installation af OpenSSH

Hvis dit system ikke kommer med SSH-server- og -klientprogrammer, kan du hente kildeteksten til OpenSSH kan hentes fra http://www.openssh.com/. Bemærk at OpenSSH-udgaver før 3.4 og OpenSSL-udgaver før 0.9.6g har kendte sikkerhedsfejl. [1] Her installationen vist med openssh-3.4 og det fungerer på samme måde med nyeste version.

Du kan hente tar-filer, som du selv kan oversætte, eller du kan få OpenSSH som RPM-filer. Som RPM-filer, skal du bruge openssh-*.i386.rpm, openssh-clients-*.i386.rpm og openssh-server-*.i386.rpm, hvor * betyder et versionsnummer, såsom 3.4.0. Du skal også bruge zlib, som sikkert allerede er installeret på din Linux-maskine. Er dette ikke tilfældet (se om du får fejl), så kan dette bibliotek hentes fra http://www.freesoftware.com/pub/infozip/zlib/. Igen kan du vælge mellem tar-filer og RPM.

OpenSSH anvender OpenSSL - et værktøj der implementerer Secure Socket Layer (SSL) til at lave stærk kryptering af transportlaget.

Fra RedHat-7.0 og senere er ssh en del af normal installation; det samme gælder de fleste andre Linux distributioner.

```
[tyge@hven ~]# su
[root@hven /home/tyge]# rpm -ivh zlib-1.1.3-i386.rpm
[root@hven /home/tyge]# rpm -ivh openssl-0.9.6g-i386.rpm
[root@hven /home/tyge]# rpm -ivh openssh-3.4-1.i386.rpm
[root@hven /home/tyge]# rpm -ivh openssh-clients-3.4-1.i386.rpm
[root@hven /home/tyge]# rpm -ivh openssh-server-3.4-1.i386.rpm
```

For at undgå de vanvittig lange indtastninger bruger man selvfølgelig tabulator tasten, når man har tastet de første par tegn ind, idet shell'en selv kan finde resten af filnavnet. Hvis der er flere filnavne der ligner, så taster man lidt flere tegn, og så igen tabulator.

Installerede du på denne måde, kan du springe det næste over og gå direkte frem til at generere bruger-nøgler med **ssh-keygen**. Vil du installere via tar-filer, så er det ret nemt, men der er et par skridt du skal igennem.

Først "zlib", hvis dette ikke er installeret (tjek om `/usr/lib/libz.so` findes).

```
[tyge@hven ~]# su
[root@hven /home/tyge]# tar xzvf zlib-1.1.3.tar.gz
[root@hven /home/tyge]# cd zlib-1.1.3
[root@hven zlib-1.3]# ./configure
[root@hven zlib-1.3]# make
[root@hven zlib-1.3]# make install
```

Dernæst skal OpenSSL installeres. Den har en variant af configure. Den er bl.a. god til at kryds-kompilere. Det nemmeste er at bruge kommandoen **./config** idet den foretager bestemmelse af maskintype og system, og derefter starter ./Configure.

```
[tyge@hven ~]# su
[root@hven /home/tyge]# tar xzvf openssl-0.9.6g.tar.gz
[root@hven /home/tyge]# cd openssl-0.9.6g
[root@hven openssl-0.9.6g]# ./config
[root@hven openssl-0.9.6g]# make
[root@hven openssl-0.9.6g]# make install
```

Og slutteligt skal OpenSSH oversættes og installeres

```
[tyge@hven ~]# su
[root@hven /home/tyge]# tar xzvf openssh-3.4.tar.gz
[root@hven /home/tyge]# cd openssh-3.4
[root@hven openssh-3.4]# ./configure --sysconfdir=/etc/ssh
[root@hven openssh-3.4]# make
[root@hven openssh-3.4]# make install
[root@hven openssh-3.4]# make host-key
```

Det er normalt at OpenSSH vil installere opsætning i `/usr/local/etc`, men Linux-folk vil oftest gemme opsætning til SSH i `/etc/ssh` - dette gøres med option **sysconfdir** (og to minusser foran).

Kommandoen **make host-key** installerer RSA og DSA nøgler for din maskine. RSA anvendes af SSH version 1, mens DSA er en nyere version, der anvendes i SSH2 protokollen.

Vi mangler stadig få ting, hvis du har oversat SSH selv (ikke ved RPM). Mange Linux-systemer anvender PAM til at styre login på maskinen. For Red Hat og SuSE skal du kopiere den generelle PAM-ssh fil til /etc/pam.d under navnet sshd. Samme idé anvendes nok af de andre store Linux-distributioner.

```
[tyge@hven ~]# su
[root@hven /home/tyge]# cp contrib/sshd.pam.generic /etc/pam.d/sshd
```

Har du oversat OpenSSH selv, mangler du stadig et start/stop script, som kan starte OpenSSH efter reboot. Som regel finder man et af de simple start/stop scripts i /etc/init.d og kopierer, retter til. I OpenSSH distributionen er der nogle eksempler på start/stop-scripts. Du kan for SuSE kopiere

```
[root@hven openssh-3.4]# cp contrib/rc.config.sshd /etc/rc.config.d/sshd.rc.config
```

og for Red Hat

```
[root@hven openssh-3.4]# cp contrib/rc.config.sshd /etc/init.d/sshd
[root@hven openssh-3.4]# chkconfig --level 234 sshd on
[root@hven openssh-3.4]#  # eller bare
[root@hven openssh-3.4]# chkconfig sshd on
```

For SuSE skal du nok rette i filen en sti fra `/usr/sbin` over til `/usr/local/sbin`.

Nu skal vi prøve at start SSH dæmonen "sshd", så man kan logge ind på maskinen udefra. For Red Hat køres

```
[root@hven openssh-3.4]# /etc/init.d/sshd start
```

eller

```
[root@hven openssh-3.4]# service sshd start
```

og tilsvarende for SuSE køres

```
[root@hven openssh-3.4]# /sbin/init.d/sshd start
```

Tjek at SSH dæmonen (dvs. serveren) kører ved at skrive **telnet localhost 22**

```
Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
SSH-1.99-OpenSSH_3.4
```

4.2.2. Brugeropsætning af OpenSSH

Hver bruger skal have sit eget sæt af privat/offentlig nøgle. Den private må aldrig sendes via netværk, idet den kan dekryptere datatrafik. Den offentlige nøgle anvendes til at kryptere data med, og den kan man så distribuere til andre maskiner, man skal kunne logge ind på.

Den fjerne maskine (remote-maskinen) kan nu kryptere trafikken på en måde, så kun os, der udleverede den offentlige nøgle, kan afkryptere datastrømmen. Dertil bruger vi naturligvis den private nøgle. Det er derfor, denne privat-nøgle ikke må gå over nettet.

Brugeren kan selv generere nøgler til login med løsen i stedet for at bruge en almindelig adgangskode til at logge ind. Det anses for mere sikkert, fordi man ikke kan logge ind på hvilken som helst konto selv om man skulle have skaffet sig den tilhørende adgangskode.

Man kører **ssh-keygen** som generer en identitetsnøgle, og undervejs kan man hertil selv vælge et løsen – svarende til en adganskode, men et, som kan være meget langt. Lav det tilpas kryptisk, dog på en måde, så du har nemt ved at huske det.

```
[tyge@hven ~]$ ssh-keygen -d
Generating DSA parameter and key.
Enter file in which to save the key (/home/tyge/.ssh/id_dsa):
Enter passphrase (empty for no passphrase): V1 hANDLER me spaghett1
```

```
Enter same passphrase again: V1 hANDLER me spaghett1
Your identification has been saved in /home/tyge/.ssh/id_dsa.
Your public key has been saved in /home/tyge/.ssh/id_dsa.pub.
The key fingerprint is:
ea:20:20:ae:b3:39:6f:d9:1b:a2:ef:02:0c:05:c5:fb tyge@hven
```

Når du kører **ssh-keygen** spørges du om du vil gemme i default stedet `/home/tyge/.ssh/id_dsa`. Dette giver dig mulighed for at have mere end et sæt nøgler. Din private nøgle gemmes i `$HOME/.ssh/id_dsa` (for DSA nøglen). Tilsvarende er den offentlige nøgle gemt med samme filnavn med et ".pub" tilføjet.

Lad os først se på et par af de filer, der blev installeret. Der er kommet filer tre steder. (1) /etc/ssh, (2) /etc/init.d/, (3) /usr/sbin/sshd og så også i hjemmekataloget, ~/.ssh .

```
[tyge@hven ~]$ ls -al /etc/ssh/*
-rw-r--r-- 1 root root  932 okt  6 00:09 ssh_config
-rw------- 1 root root  668 okt 11 01:33 ssh_host_dsa_key
-rw-r--r-- 1 root root  604 okt 11 01:33 ssh_host_dsa_key.pub
-rw------- 1 root root  529 okt 11 01:33 ssh_host_key
-rw-r--r-- 1 root root  333 okt 11 01:33 ssh_host_key.pub
-rw------- 1 root root 1194 okt  6 00:09 sshd_config
```

Filerne `/etc/ssh/*` er opsætningsfiler, der styrer ssh opsætningen. Det er kun `/etc/ssh/sshd_config`, du evt. skal rette i. [2] F.eks. kan du ændre "PermitRootLogin yes" til "PermitRootLogin no", hvis du mener, at root ikke må logge ind via ssh. Det kan være en god idé at forbyde root remote login i det hele taget. Tilsvarende kan du forbyde tomme adgangskoder ved at ændre "PermitEmptyPasswords yes" til "PermitEmptyPasswords no" - vent lige med at lave disse ændringer til du har fået ssh til at virke.

Vi skal også lige nævne, at ssh forbindelser normalt ikke styres via inetd-systemet, som du i øvrigt kan finde beskrevet tidligere i Kapitel 2. Grunden er, at det ville være for langsomt, idet der ved opstart af sshd skal genereres en server-nøgle. Dette kan tage flere sekunder for hver opstart.

4.2.3. Brug af SSH

Hvis sshd er startet op, er alt klar til at kommunikere sikkert, også over usikre netværk. Start med at skrive

```
[tyge@hven ~]$ ssh bohr
Host key not found from the list of known hosts.
Are you sure you want to continue connecting (yes/no)?
```

Første gang du kobler til en fremmed maskine, der ligeledes har fået installeret ssh, skal ssh acceptere at udveksle nøgler med en ukendt maskine. Dette spørgsmål skal du således acceptere, og næste gang du anvender samme fremmede maskine, skal du ikke igennem dette spørgsmål. Efter at have svaret "yes"

skal du aflevere din almindelige adgangskode, og du er så logget ind på maskinen. Dette kan du fortsætte med, men hvis du vil højne sikkerheden yderligere, bør du gemme din offentlige nøgle på fjernmaskinen. Har du denne nøgle gemt, kan man ikke logge ind med din adgangskode, men kun med dit lange og kryptiske løsen.

Hvis du ikke fik adgang til maskinen, så kan der være flere grunde. Enten kan det skyldes, at du bruger den forkerte protokol. Nyere versioner af OpenSSH kan både køre SSH version 1 og 2. Version 2 og DSA-nøglen hænger sammen som nævnt ovenfor.

```
[tyge@hven ~]$ ssh -2 tuck@bohr
```

I dette eksempel tvinges OpenSSH til at anvende SSH version 2, og brugeren tyge prøver at logge ind på maskinen bohr med brugernavn tuck.

Tip: Du bør udelukkende anvende SSH2 (SSH version 2); skriv følgende ind i din `~/.ssh/config`

```
# Tillad også grafiske programmer gennem SSH-forbindelse
ForwardX11 yes
# Anvend SSH2-identitet
IdentityFile ~/.ssh/id_dsa
# Anvend SSH2-protokol
Protocol 2
# Komprimer forbindelsen
compression yes
```

Log ud ved at skrive exit (eller trykke Ctrl-D) for at komme tilbage til din egen maskine. Kopier nu din public key fra din egen maskine (hven) til fjernmaskinen og gem den under `~/.ssh/authorized_keys` (for SSH1, dvs. RSA nøglen) og `~/.ssh/authorized_keys2` (for SSH2, dvs. DSA nøglen). Ingen andre end dig skal kunne læse den fil du laver. Denne kopiering laver vi med en ny kommando "scp" (secure copy) eller beder systemadministratoren at lægge filen ind hvis du ikke kan få adgang.

```
[tyge@hven ~]$ cd ~/.ssh
[tyge@hven .ssh]$ scp id_dsa.pub bohr:authorized_keys2
```

Nu skal du lave den sidste opsætning på bohr. Du laver kataloget `~/.ssh`, hvor du gemmer din offentlige nøgle, og sikrer, at andre ikke kan læse denne. Dit hjemmekatalog skal andre heller ikke have lov til at skrive i - denne foranstaltning er altid klog, men det er også nødvendig for at din offentlige nøgle "id_dsa2.pub" virker efter, at den er kopieret til fjernmaskinens "authorized_keys2". Det er en egenskab ved ssh.

```
[tyge@hven ~]$ ssh bohr
[tyge@bohr tyge]$ mkdir ~/.ssh
[tyge@bohr tyge]$ mv authorized_keys ~/.ssh
[tyge@bohr tyge]$ chmod go-w ~
[tyge@bohr tyge]$ chmod -R go-rwx .ssh
[tyge@bohr tyge]$ exit
```

4.2.4. Krypteret dataoverførsel

Nu kan du slappe af. Alt er sat op, og du kan uden at skulle frygte for netværkssikkerheden logge ind på bohr. F.eks. kan du starte et grafik program såsom "xload" ved at skrive

```
[tyge@hven ~]$ ssh bohr xload
Enter passphrase for RSA key 'tyge@bohr': V1 hANDLER me spaghett1
```

Du blev nu mødt af noget nyt igen, idet du skulle skrive dit løsen og ikke din adgangskode. Bemærk, at i virkeligheden vises dit løsen naturligvis ikke på skærmen. **xload** vil nu køre fra bohr og vises på din egen maskine (hven). Skal du logge ind på bohr, så skriver du blot "ssh bohr", og skal du have udført et program derfra, tilføjer du blot programnavnet til denne ordre.

Skal du kopiere en fil fra hven til bohr, skriver du

```
[tyge@hven ~]$ scp LOKALT_FILNAVN bohr:FJERN_FILNAVN
```

Tilsvarende erstattes ftp med **sftp**, og scp erstatter rcp (remote copy), som arbejder med samme syntaks.

Hvis du ikke frygter, hvem der kan tilgå din egen maskine, kan du få endnu nemmere adgang til de andre maskiner ved, at du en gang for alle i den X session, du har igang, giver dit løsen.

```
[tyge@hven ~]$ ssh-agent bash
[tyge@hven ~]$ ssh-add ~/.ssh/id_dsa
Enter passphrase for /home/pto/.ssh/id_dsa:
```

Derefter kan du med ssh fra den terminal logge ind og ud af "bohr" og andre ssh maskiner uden at skulle bruge løsen. Vil du have at alle terminal-vinduer skal kunne dette, skal du rette i din "~/.xsessionrc", "~/.xinitrc" eller lignende, hvor din window manager startes op. Er det f.eks. KDE, skal du ændre "startkde" til "ssh-agent startkde" og kun en enkelt gang køre "ssh-add". Derefter kan du slippe for at indtaste dit løsen i resten af den X session. Brug "ssh-agent" med omtanke.

Tip: Det anbefales at du flytter SSH til en anden port end port 22 - hvis det kun er dig som benytter denne. Den valgte port kan på klienten skrives ind i ~/.ssh/config således at det stadig er nemt at logge ind. Eksempel:

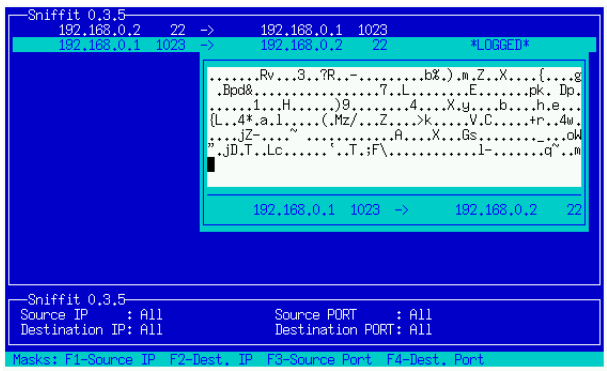
```
~/.ssh/config:

host tyge
     port 22123

host steno
     port 955
```

En af de (mange) meget elegante ting man kan med SSH (og OpenSSH) er at lave en sikker krypteret tunnel gennem en firewall og endda kunne transmittere grafiske programmer (X-programmer). Skal man administere servere (evt. via grafiske programmer) på et lukket netværk er en SSH-server i firewallen yderside en god måde at give adgang til netværket hjemmefra (husk dog at følge sikkerhedslisten for den SSH-variant der installeres). For at grafiske programmer kan transmitteres fra en maskine til en anden via SSH, kræves at man har programmet **xauth** installeret.

Anvender du ssh, kan andre med sniffit se, at der laves kommunikation på port 22, men prøver de at følge netværkstrafikken, kommer der ikke login navne, adgangskoder, eller efterfølgende kommandoer i klar tekst - alt er krypteret. Den lille boks i sniffit, som for **telnet** viste login sekvensen med adgangskoder osv., vil med ssh være fyldt med en bunke tilfældige tegn uden sammenhæng - kun de to SSH-programmer der har etableret forbindelsen kan i praksis afkode kommunikationen.

Lad os vende tilbage til sniffit og se, hvad der med ssh kommer over netværket under login via ssh. På næste billede kan vi se, at port 22 på bohr modtager tekst, der er krypteret og dermed ikke læseligt for andre. Igen er det "sniffit -F eth0 -i", som køres. Derefter har vi valgt at følge den linje fra 192.168.0.1, som kommer ind via port 22 til 192.168.0.2. I det lille vindue kan du se resultatet af vilkårlige og almindelige Linux-kommandoer - i dette tilfælde "ls -al /home".

Figur 4-4. Sniffit



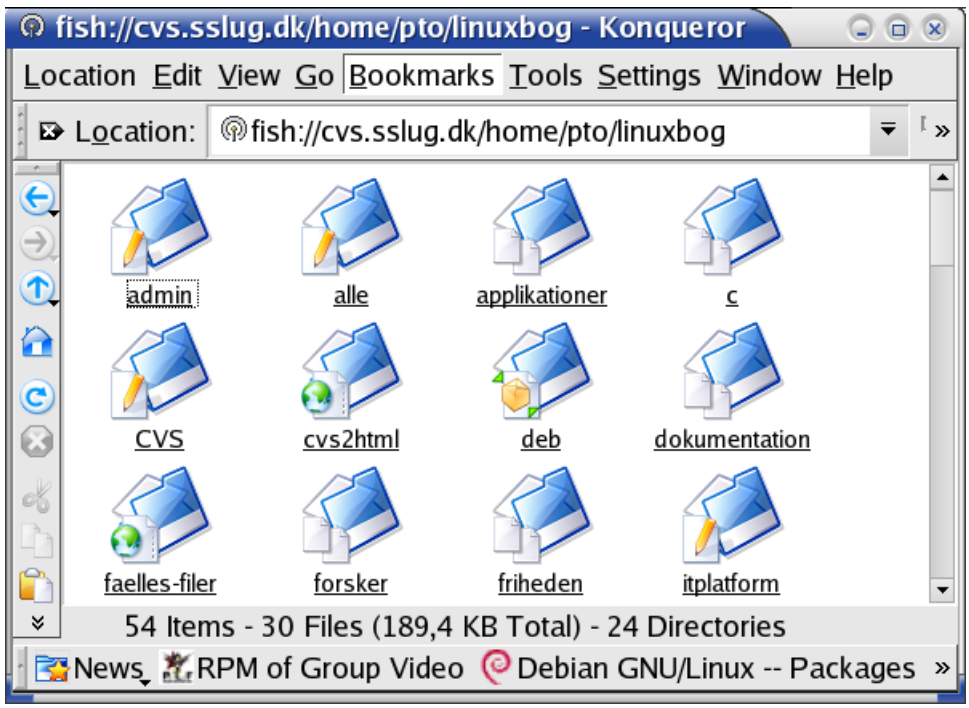
Endelig kan det nævnes at følgende link giver en god gennemgang af SSH http://www.redhat.com/docs/manuals/linux/RHL-8.0-Manual/custom-guide/s1-openssh-client-config.html.

4.2.5. Brugervenlighed

Som det er beskrevet ovenfor så bør kommunikation mellem maskiner ske over SSH-protokollen hvor det er muligt. Det er et tekst-baseret program hvor man kan logge ind på en fjernmaskine og udføre kommandoer - men det er ikke hele historien. Man kan også sætte OpenSSH op til at køre grafiske

programmer eller få andre programmer til at bruge OpenSSH til at danne en krypteret kanal mellem maskinerne og mellem maskinerne køre alle programmer transparent. Med Konqueror til KDE kan man f.eks. installere en lille udvidelse med navn `fish` http://ich.bin.kein.hoschi.de/fish/. Normalt angiver man i URL-feltet til Konqueror at man vil ser filerne i `/usr/` som `file:/usr/`. Vil man se filerne i kataloget "`/usr/`" på maskinen "`hven.sslug.dk`" med brugernavnet "`tyge`" skriver man i URL-feltet `fish://tyge@hven.sslug.dk/usr`. Har man SSH-adgang til maskinen vil filoversigten blive vist som om det var på egen maskine med fuld mulighed for træk-og-slip-kopiering, -flytning mv.

Figur 4-5. fish giver direkte adgang til filer på andre maskiner via sikker ssh-krypteret tunnel



4.2.6. Epilog

Der er mange forskellige smarte features i ssh, såsom at maskinerne skal acceptere hinandens identitet, brugeres skal accepteres via et løsen, og en gang hver time vil maskiner endda skifte nøgler, så en eventuel ondsindet person, som vil lytte med skal begynde forfra i dekodning af krypterings-nøgler.

Vi skal også nævne, at ssh kan anvendes til at lave VPN løsninger (Virtual Private network) mellem to lokale netværk, der forbindes via et usikkert net. Skal man køre revisionssystemer på Linux, kan vi anbefale CVS, som drager nytte af ssh til at skabe krypteret tilgang til ens server. Det er kun en systemvariabel (sæt $CVS_rsh=ssh), så kører det. Tilsvarende kan rsync kobles med ssh (sæt

$RSYNC_rsh=ssh) for at lave synkroniseret data mellem flere maskiner, hvor dataudveksling sker med secure shell.

Ud over Linux (UNIX) server og klient programmer, som er indeholdt i ssh-pakken, så kan du måske også have glæde af klienter til Windows.

Med en af disse kan du fra en Windows-maskine logge sikkert ind på din Linux-maskine. Du kan på hhv. http://www.chiark.greenend.org.uk/~sgtatham/putty/, http://www.mindbright.se/mindterm. Der findes også en kommerciel Windows ssh-version, som kan købes fra http://www.datafellows.com.

Slutbemærkning:

1. Hvis din distribution kører med ældre udgaver af OpenSSH og/eller OpenSSL er det muligt at de selv har lukket sikkerhedshullerne.
2. Hvis du har installeret fra source-tar-balls er disse filer default lagt i /usr/local/etc

Kapitel 5. Har du haft net-indbrud?

Selvom du er omhyggelig med, hvordan din Linux-maskine er sat op, og du håndterer adgangskoder med stor omhu, må du regne med at dit system aldrig er fuldstændig sikkert. Der kan stadig være en sikkerhedskritisk fejl i et af de programmer du bruger. En adgangskode kan blive opsnappet. Eller du kan have overset noget kritisk i opsætningen af din maskine. Det kan også være, at du af funktionalitetshensyn har valgt at benytte programmer med kendte fejl til trods for de sikkerhedsproblemer, det medfører. Derfor er det vigtigt, at du holder øje med, om der har været fremmede inde på dit system.

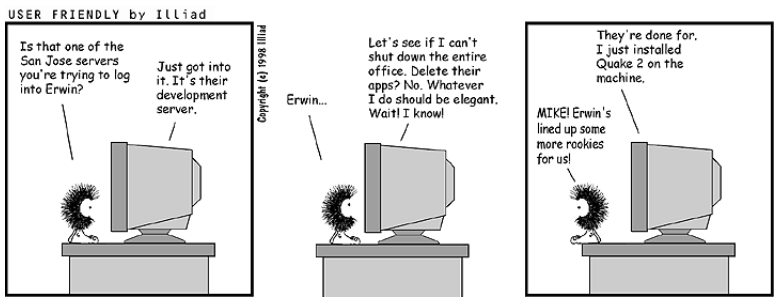
Lad os nu uden at tænke på hvordan antage, at en dygtig eller heldig cracker har fået root-adgang til din maskine. Denne cracker har med andre ord adgang til alt på din maskine, og har måske endda installeret nye programmer med hemmelige bagdøre på maskinen, der gør det nemmere at komme ind næste gang.

En person, der trænger ind på andres computere, har ofte et formål med det. Det kan være en som vil snuse rundt efter informationer, eller en som alene skal bruge maskinen til at angribe andres maskiner. Ofte vil han samtidig forsøge at sløre sit eget spor. Der er mange muligheder for adfærd for en cracker, som du skal kunne følge. I denne situation er det crackeren, der vælger angrebstidspunkt og metode, og du kan i praksis ikke overvåge alt 24 timer i døgnet på maskinen. Men ligesom du kan opdage en indbrudstyv i dit hjem, kan du opdage ham på din computer ved at lytte efter unormal adfærd og sætte tyverialarmer op, som vækker dig ved indbrud.

Det, vi skal se nærmere på i dette kapitel, er den interne overvågning af din maskine. Vi vil se på, hvordan du som systemadministrator kan finde ud af, at der har været indbrud, og hvordan du kan se hvad crackeren har lavet om på maskinen. Vi vil komme ind på de generelle forholdsregler, du som systemadministrator kan tage, samt en række værktøjer, som kan hjælpe dig.

En cracker kan finde på at installere skadelige programmer på din computer. Her en stribe fra User Friendly (http://www.userfriendly.org/)

Figur 5-1. User Friendly



5.1. Log-filer

Pas på med tillid til logfilerne. Husk, at du ikke kan stole på din maskine, hvis der har været indbrud. Hvis dine logfiler ikke afslører et indbrud, kan der godt have været et, som er blevet skjult. Hvis logfilen viser, at der har været indbrud, så kan du som regel godt stole på den.

Det er en god idé at bruge de retningslinier for oprydning, som CERT/CC har angivet på hjemmesiden http://www.cert.org/nav/recovering.html (http://www.cert.org/nav/recovering.html)

En Linux-maskine gemmer løbende en masse information om, hvad der sker på maskinen. Informationen gemmes i log-filer, som kan bl.a. bruges til at spore unormal anvendelse.

5.1.1. /var/log/messages

Den vigtigste log-fil er `/var/log/messages`, hvor mange af de vigtige systeminformationer gemmes, såsom

- hvem der loggede ind på maskinen hvornår
- hvem der skiftede identitet med su
- hvornår maskinen blev genstartet
- internetopkoblinger med ppp
- start og stop af tjenester, såsom NFS
- fejl i kernemoduler

og meget andet.

Lad os se på nogle eksempler som alle er brudstykker fra `/var/log/messages`. Vi ser, at det kræver lidt øvelse at overskue alt i messages-filen, men modsat er der mange nyttige informationer om systemets drift. Lad os først se på Eksempel 5-1 nedenfor, som viser de linjer, der kommer i messages-filen, når systemet startes.

Eksempel 5-1. `/var/log/messages` ved systemstart

```
Jun 30 20:17:45 hven syslogd 1.3-3: restart.
Jun 30 20:17:45 hven syslog: syslogd startup succeeded
Jun 30 20:17:45 hven syslog: klogd startup succeeded
Jun 30 20:17:45 hven kernel: klogd 1.3-3, log source = /proc/kmsg started.
Jun 30 20:17:45 hven kernel: Inspecting /boot/System.map-2.2.5-15
Jun 30 22:17:16 hven rc.sysinit: Loading default keymap succeeded
Jun 30 22:17:16 hven rc.sysinit: Setting default font succeeded
Jun 30 22:17:16 hven swapon: swapon: warning: /dev/hdb1 has insecure permissions 0660, 0600
Jun 30 22:17:16 hven rc.sysinit: Activating swap partitions succeeded
Jun 30 22:17:16 hven rc.sysinit: Setting hostname hven.galaxy.dk succeeded
Jun 30 22:17:16 hven fsck: /dev/hda3: clean, 56029/311296 files, 748112/1242864 blocks
```

```
Jun 30 22:17:16 hven rc.sysinit: Checking root filesystem succeeded
Jun 30 22:17:16 hven isapnp: Board 1 has Identity 70 01 00 00 00 01 00 93 05:  ALS0001 Seri
Jun 30 22:17:16 hven isapnp: ALS0001/16777216[0]{ALS100 Media Audio Controller}: Ports 0x22
Jun 30 22:17:16 hven isapnp: /etc/isapnp.conf:293 -- Fatal - resource conflict allocating 8
Jun 30 20:17:47 hven kernel: Loaded 6721 symbols from /boot/System.map-2.2.5-15.
Jun 30 20:17:47 hven kernel: Symbols match kernel version 2.2.5.
Jun 30 20:17:47 hven kernel: Loaded 146 symbols from 11 modules.
Jun 30 20:17:47 hven kernel: Linux version 2.2.5-15 (root@porky.devel.redhat.com) (gcc vers
Jun 30 20:17:47 hven kernel: Detected 120003903 Hz processor.
Jun 30 20:17:47 hven kernel: Console: colour VGA+ 80x25
Jun 30 20:17:47 hven kernel: Calibrating delay loop... 47.82 BogoMIPS
Jun 30 20:17:47 hven kernel: Memory: 63140k/65536k available (996k kernel code, 412k reserv
Jun 30 20:17:47 hven kernel: VFS: Diskquotas version dquot_6.4.0 initialized
Jun 30 20:17:47 hven kernel: CPU: Intel Pentium 75 - 200 stepping 06
Jun 30 20:17:47 hven kernel: Checking 386/387 coupling... OK, FPU using exception 16 error
Jun 30 20:17:47 hven kernel: Checking 'hlt' instruction... OK.
```

Ud fra den ottende linje kan vi se, at harddisk partitionen `/dev/hdb1` ikke har fornuftige læse/skriverettigheder. Vi kan også se, et ISA lydkort ALS100 bliver detekteret af isapnp-pakken, men at det ikke er konfigureret rigtigt. Derefter en del information om selve maskin-typen, CPU-hastighed (det er fra en Pentium 120 MHz PC). Du bør lige tjekke fra tid til anden, at der ikke logges fejl, såsom problemet med forkerte læse/skrive rettigheder eller hardwareproblemer. I øvrigt kan den opmærksomme læser se, at nogle tidspunkter er kl. 22 og andre 20. Dette skyldes, at nogle dele af systemet kører med lokaltid og andre med GMT. Vi kan også her nævne, at du med kommandoen "dmesg" kan se de fleste af de systemmeddelelser, som blev vist på skærmen under systemopstart. Tilsvarende information findes i `/var/log/dmesg`.

Eksempel 5-2 viser hvordan telnet, ftp og SSH logins gemmes i messages-filen. Den 3. juli kl. 10.06 kommer brugeren robin ind fra IP nummer 192.168.0.10 via secure shell (SSH). To minutter efter logges der ud igen. Dagen efter er det en telnet login, som går igennem PAM-adgangskodetjek (PAM_pwdb). Kun et minut er robin inde. Endelig er brugeren tuck kommet ind via ftp (ftpd) den 4. juli og kun blevet fire sekunder.

Eksempel 5-2. `/var/log/messages` ved remote logins

```
Jul  3 10:06:51 hven sshd[763]: log: Connection from 192.168.0.10 port 1023
Jul  3 10:06:57 hven sshd[763]: log: RSA authentication for robin accepted.
Jul  3 10:06:57 hven sshd[765]: log: executing remote command as user robin
Jul  3 10:08:58 hven sshd[763]: log: Closing connection to 192.168.0.10
Jul  4 23:07:07 hven PAM_pwdb[24298]: (login) session opened for user robin by (uid=0)
Jul  4 23:08:10 hven PAM_pwdb[24298]: (login) session closed for user robin
Jul  4 23:10:28 hven ftpd[24757]: FTP LOGIN FROM tuck @ vinglad.linus.dk [192.168.0.199],
Jul  4 23:10:32 hven ftpd[24757]: FTP session closed
```

Lad os dernæst se på en internetopkobling via modem (ppp). Vi kan i de følgende linjer fra messages-filen se, at brugeren har koblet op via "/dev/ppp0". De to linjer med identd skyldes secure shell. Der er lavet en SSH-login til www.sslug.dk under opkoblingen. De sidste linjer viser, at der er sendt 55395 bytes til maskinen og 80884 bytes fra maskinen under opkoblingen.

Eksempel 5-3. `/var/log/messages` ved internetopkobling

```
Aug 18 21:28:35 hven ifup-ppp: pppd started for ppp0 on /dev/modem at 115200
Aug 18 21:28:36 hven pppd[10800]: pppd 2.3.7 started by root, uid 0
Aug 18 21:29:03 hven pppd[10800]: Serial connection established.
Aug 18 21:29:03 hven pppd[10800]: Using interface ppp0
Aug 18 21:29:03 hven pppd[10800]: Connect: ppp0 <--> /dev/modem
Aug 18 21:29:11 hven pppd[10800]: Remote message: Login Succeeded
Aug 18 21:29:12 hven pppd[10800]: local  IP address 212.54.78.125
Aug 18 21:29:12 hven pppd[10800]: remote IP address 212.54.64.66
Aug 18 21:29:16 hven identd[10930]: Connection from www.sslug.dk
Aug 18 21:29:16 hven identd[10930]: from: 192.38.71.98 ( www.sslug.dk ) for: 1132, 25
Aug 18 21:35:14 hven pppd[10800]: Terminating on signal 15.
Aug 18 21:35:14 hven pppd[10800]: Connection terminated.
Aug 18 21:35:14 hven pppd[10800]: Connect time 6.2 minutes.
Aug 18 21:35:14 hven pppd[10800]: Sent 80884 bytes, received 55395 bytes.
Aug 18 21:35:14 hven pppd[10800]: Exit.
```

Det sidste eksempel fra messages-filen er et sted, hvor man kan se, at bruger nummer 500 (robin) skifter bruger-ID over til root for at kunne stoppe sendmail. Derefter logges det at su-root sessionen afsluttes.

Eksempel 5-4. `/var/log/messages` ved su og nedlukning af sendmail

```
jul 26 22:15:04 hven PAM_pwdb[4220]: (su) session opened for user root by (uid=500)
jul 26 22:15:25 hven sendmail: sendmail shutdown succeeded
jul 26 22:15:41 hven PAM_pwdb[4220]: (su) session closed for user root
```

`/var/log/messages` kan blive meget stor. Da alle systemhændelser bliver logget i mellem hinanden, kan det således være svært umiddelbart at skelne de interessante beskeder i mængden.

5.1.2. Rotering af log-filer

Du kan opleve at finde filer i `/var/log`, der f.eks. hedder `messages.1` og `messages.2`, eller `messages.1.gz`. Dette skyldes, at man på Linux-systemer ofte har et cron-job sat op til at rotere logfilerne, herunder messages-filen. Dette sker f.eks. en gang i døgnet eller en gang om ugen. I så fald indeholder `/var/log/messages` kun det nyeste log-information, som er sket siden sidste rotation.

Under Red Hat styres rotering med af logfiler med logrotate-pakken, som konfigureres i filen `/etc/logrotate.conf`

5.1.3. /var/log/secure

Fordi vigtig information så nemt kan drukne i anden information i messages-filen, findes filen `/var/log/secure`, som kun indeholder sikkerhedsrelateret information. Eksempel 5-5 viser `/var/log/secure` filen. Brugerne robin og john logget ind via terminal tty1 nogle gange, og en telnet

og en ftp login findes i bunden af filen. `/var/log/secure` kan være en meget vigtig fil at se igennem jævnligt.

Eksempel 5-5. `/var/log/secure`

```
Aug 15 14:45:16 hven login: LOGIN ON tty1 BY robin
Aug 16 15:13:53 hven login: LOGIN ON tty1 BY john
Aug 16 21:00:55 hven login: LOGIN ON tty1 BY robin
Aug 17 11:22:41 hven login: LOGIN ON tty1 BY john
Aug 17 17:45:16 hven login: LOGIN ON tty1 BY robin
Aug 18 07:11:29 hven login: LOGIN ON tty5 BY robin
Aug 18 12:49:56 hven login: LOGIN ON tty1 BY john
Aug 18 20:17:07 hven login: LOGIN ON tty1 BY robin
Aug 18 23:07:03 hven in.telnetd[24297]: connect from 192.168.0.1
Aug 18 23:07:07 hven login: LOGIN ON 3 BY robin FROM linus
Aug 18 23:10:24 hven in.ftpd[24757]: connect from 192.168.0.1
```

5.1.4. /var/log/maillog

En anden vigtig fil er `/var/log/maillog`, hvor information om post til og fra maskinen gemmes. Vi kan i Eksempel 5-5 se, at robin sender en besked fra hven.galaxy.dk til <sslug-teknik@sslug.dk>. Postloggen kan nemt blive ekstremt stor og svær at overskue.

Eksempel 5-6. `/var/log/maillog`

```
Jun 18 23:00:42 robin sendmail[1966]: XAA01966:
from=<robin@hven.dk>, size=496, class=0, pri=30496,
nrcpts=1,
msgid=<Pine.LNX.4.10.9906182300010.662-100000@hven.dk>,
proto=ESMTP, relay=robin@localhost
Jun 18 23:00:44 hven sendmail[1968]: XAA01966:
to=<sslug-teknik@sslug.dk>, delay=00:00:02, xdelay=00:00:02,
mailer=esmtp, relay=mail.sslug.dk. [192.38.71.98],
stat=Sent (ok 929739581 qp 27375)
```

5.1.5. Rapporting af log-meddelelser

I stedet for at læse logfiler direkte kan man installere programmer, der kan sortere i logfilerne og fremhæve vigtige ting. Vi vil nu se nærmere på LogWatch, der kan hentes fra http://www.kaybee.org/~kirk/html/linux.html.

LogWatch sættes normalt op til at køre en gang i døgnet via Linux maskinens crontab system. LogWatch skanner alle log-filerne og sender en samlet rapport til systemadministratoren som et e-brev. I filen `/etc/log.d/logwatch.conf` er det muligt at konfigurere LogWatch. Det kan f.eks. anbefales at sætte parameteren "Detail" fra "Low" til "High", så alle login og "su" hændelser rapporteres.

LogWatch kan f.eks. give følgende e-brev tilbage for en dags trafik for en maskine, der ikke kører telnet, men som tillader folk at hente filer over FTP og at logge ind med SSH. Man kan se, at ftp dæmonen har overført 5 Mb, og filerne er vist sammen med navnet på modtagermaskinen. På maskinen har der i PAM_pwdb (dvs. adgangskodehåndteringen) været en række skift af bruger-id. Desuden er alle SSH opkoblinger til maskinen vist, derefter cron kørsler og endelig de problemer, som navneserveren har haft igennem dagen. Let og overskueligt at se igennem. Har man pludselig logins fra uventede domæner, er det noget, der skal kigges nærmere på.

Eksempel 5-7. logwatch

```
 --------------------- ftpd-messages Begin ------------------------

Anonymous FTP Logins:
   laptop.etsted.dk (192.168.0.10) : IE40user@ - 4 Time(s)
   dbserver.etandetsted.dk (192.168.0.98) :  mozilla@ - 1 Time(s)
   dbserver.etandetsted.dk (192.168.0.99) :  getright@ - 2 Time(s)

 ---------------------- ftpd-messages End -------------------------



 --------------------- ftpd-xferlog Begin ------------------------
TOTAL KB OUT: 5339KB (5MB)
TOTAL KB IN: 0KB (0MB)

Outgoing Anonymous FTP Transfers:
   /pub/linuxbog.pdf.gz -> dbserver.etandetsted.dk
   /pub/utils/cvsstat -> 192.168.1.10
   /pub/linuxbog.pdf.gz -> 192.168.10.12
   /pub/videoclips/Linux1601.mpg -> ns.etandetsted.dk
   /pub/utils/cvs2html -> cvs.etandetsted.dk

 ---------------------- ftpd-xferlog End -------------------------



 --------------------- Identd Begin ------------------------
Identd Lookups:


 ---------------------- Identd End -------------------------



 --------------------- PAM_pwdb Begin ------------------------

SU Sessions:
      robin(uid=71) -> john - 1 Time(s)
      john(uid=0) -> root - 3 Time(s)
      robin(uid=0) -> root - 5 Time(s)
      sherif(uid=0) -> root - 1 Time(s)
      john(uid=5192) -> charles - 1 Time(s)
```

```
      robin(uid=5192) -> charles - 5 Time(s)

Opened Sessions:
   Service: su
      User nobody - 1 Time(s)
      User news - 1 Time(s)

 ---------------------- PAM_pwdb End -------------------------



 --------------------- SSHD Begin ------------------------

Connections:
   laptop.etsted.dk (192.168.0.10) : 1 Connection(s)
   dbserver.etandetsted.dk (192.168.0.98) : 27 Connection(s)

 ---------------------- SSHD End -------------------------



 --------------------- Cron Begin ------------------------

Commands Run:
   User root:
      /usr/bin/mrtg /etc/mrtg/mrtg.cfg >/dev/null 2>&1: 288 Time(s)
      /usr/local/bin/daily_backup: 1 Time(s)
      run-parts /etc/cron.daily: 1 Time(s)
      run-parts /etc/cron.hourly: 24 Time(s)


 ---------------------- Cron End -------------------------


 --------------------- Named Begin ------------------------

**Unmatched Entries**
   Response from unexpected source ([157.151.95.204].53): 1 Time(s)
   bad referral (US !< RESTON.VA.US): 6 Time(s)
   bad referral (US !< SF.CA.US): 6 Time(s)

 ---------------------- Named End -------------------------
```

Et alternativ til LogWatch er **logcheck**, som kan hentes fra http://sourceforge.net/projects/sentrytools/ (http://sourceforge.net/projects/sentrytools/) Ideen er den samme, og virkemåden er umiddelbart også ens. Blot er formatteringen af log-rapporter ikke sorteret så elegant som med LogWatch. Prøv begge programmer, og vælg selv dit foretrukne.

Problemet med programmer som LogWatch og **logcheck** er bl.a., at programmet normalt køres en gang per døgn (dette kan ændres). Ergo kan et eventuelt indbrud sløres, f.eks. ved at personen blot sletter linjer i log-filerne svarende til egen adfærd. Det eneste alternativ er nok, at væsentlig log-information skrives

til en enhed som kun accepterer tilføjelser - ikke redigering af data. Nogle bruger f.eks. en almindelig gammel linje-printer, som udskriver log-meddelelser straks efter hændelsen.

5.1.6. Alarmer

Du kan lave dine egne hjemmelavede alarmer på dit Linux-system. Har du en maskine med meget få logins, kan det være interessant, at der afsendes et e-brev til en fast ekstern modtager, hver gang der laves login på din maskine. Brevet kan f.eks. sendes til en mobiltelefon eller pager, for at du med det samme kan få at vide, når der er gæster. Der er mange muligheder for, hvordan dette kan gøres. Den enkle måde er at tilføje følgende til filerne `/etc/csh.login` og `/etc/profile`.

```
mail < /dev/null > /dev/null -s "login at `date`" robin@hven.herne.dk
```

Derved får "robin@hven.herne.dk" en besked med tidspunkt for login, uden at den, der logger ind, kan se det. Personen kan dog bagefter selv læse `/etc/csh.login` og `/etc/profile` og se, at der er lagt en fælde, og evt. vælge at forsvinde. En interessant beskrivelse af indbrud, alarmer og fælder kan læses på http://www.ja.net/CERT/Cheswick/berferd.txt

En anden mulighed er at bruge en almindelig brugerkonto på systemet til at have et crontab job, der køres hvert minut, og ser hvem der er logget ind (via "who") og sender resultatet til din eksterne maskine, hvis der er brugere logget ind. Man vælger selv hvordan den type og niveau af alarm der sættes op, alt efter hvor paranoid man er :-)

Man skal dog passe på, at de alarmsystemer, der sættes op, ikke åbner et nyt sikkerhedshul. Som eksempel kan det være, at du har ændret kildeteksten til **/bin/login**, så der logges mere information til f.eks. `/var/log/messages`, men du kom måske ved et uheld til at få en buffer overflow fejl, som efterfølgende udnyttes af en ihærdig cracker. Det er heldigvis sådan, at den person, som bryder ind på din maskine, ikke i forvejen ved, hvor du lægger fælder, og forhåbentlig ikke har en chance for at vide, hvordan de virker, før du har opdaget indbruddet.

Husk også de følgende tre ting: Backup, backup og backup. Lav jævnligt backup af din maskine. Har du haft besøg af en ondsindet cracker, kan du have mistet alt. Sørg også for at have ældre backups. Hvis dit system har været inficeret igennem et stykke tid, er dine nyeste backups også inficerede. En sund strategi er, at have en system backup, som du laver før systemet sættes i drift - alle ændringer skrives ned, så de kan geninstalleres fra din sikre backup. Derudover skal du lave jævnlige separate backups af dine brugeres data.

5.1.7. Ændringer af filsystemet

Du bør holde øje med hvilke filer på dit system, der ændrer sig. Forestil dig, at en person er kommet ind på din maskine og har haft held til at erstatte **passwd** kommandoen med en ny binær fil, som dels laver det den skal men samtidig sender login-navn og password til supercracker@passwordcracker.net. Det er ikke urealistisk svært at lave sådan et program, idet alle har adgang til kildeteksten til Linux-systemet, og

dermed kan lave ændringerne uden at behøve at skrive hele programmet forfra. Er crackeren først inde på dit system, og har han lavet programændringerne på forhånd, skal han blot erstatte de rigtige programmer med de nye. Det er normalt de binære filer i `/usr/bin`, `/bin`, `/usr/sbin/`, `/usr/X11R6/bin` og `/sbin`, man skal passe på, samt biblioteker, som normalt findes i `/lib`, `/usr/X11R6/lib` og `/usr/lib`.

Generelt leder man efter ændringer af filsystemet ved at tjekke fire ting:

- Om ejerskab/gruppeejerskab er ændret
- Om filens rettigheder (permissions) er ændret
- Om filstørrelse er ændret
- Om indholdet er ændret

Hvis vi kan stole på output af kommandoen "ls", så er de tre første nemme at undersøge. Man kan f.eks. sammenligne output fra "ls -l FILNAVN" før og efter ændringer.

Det er dog straks sværere at tjekke, om indholdet af filen er intakt. For et filsystem med en stor mængde data er det ikke muligt at have en kopi af hele filsystemet og tjekke byte for byte, om de er ens. Det tager for meget diskplads og alt for lang tid. Derfor er en del matematisk forskning gået på at udtænke metoder til at generere en funktion, som tager f.eks. indholdet af en fil og ud fra dette genererer et langt tal, som i princippet skal være unikt. Denne hash-funktion må således kun give det samme tal for to filer, hvis de er helt ens. Dermed kan man kontrollere, om en fil er ændret, blot ved at generere et tal for filen med hash-funktionen og sammenligne med det tal, filen gav sidste gang. Giver hash-funktionen samme output, antager vi, at filen er urørt.

En simpel hashfunktion kunne være summen af alle de tal, som filen består af. Den er blot alt for nem at forfalske. I litteraturen vil man ofte støde på CRC-tjek, som er relativt nemme funktioner at beregne. Mange komprimeringsprogrammer anvender CRC til at kontrollere, om det, der pakkes ud, er i orden. Til at opdage ændringer af filer med tanke på sikkerhed, er CRC-tjek ikke gode nok. Et hyppigt anvendt alternativ er MD5, som genererer en 128 bit kode ud fra en fil. På de fleste Linux-systemer findes programmet **md5sum**, som køres med et filnavn som argument.

```
[tyge@hven ~]$ md5sum /bin/ls
dc2ac9d1c1658d5b4381ca2c280425ee  /bin/ls
```

Et karakteristisk træk ved MD5 algoritmen er at selv små ændringer i input giver radikale forskelle i checksummen. Det er en almindelig antagelse, at det er umuligt at ændre filen **/bin/ls** uden at det kan afsløres ved hjælp af MD5-sum af den oprindelige version. Lad os lave et eksempel med en lille fil, der får indholdet ændret, men hvor man kan ikke se udefra, at den er ændret.

```
[robin@hven robin]$ echo "robin og tuck" > testfil
[robin@hven robin]$ touch -d 19990719 testfil
[robin@hven robin]$ ls -al testfil
-rw-r--r--   1 robin      robin            14 Jul 19 00:00 testfil
[robin@hven robin]$ md5sum testfil
e829f144f6e43d55daa442baf1462544  testfil
```

```
[robin@hven robin]$ echo "tuck og robin" > testfil
[robin@hven robin]$ touch -d 19990719 testfil
[robin@hven robin]$ ls -al testfil
-rw-r--r--   1 robin      robin            14 Jul 19 00:00 testfil
[robin@hven robin]$ md5sum testfil
18b8426e71e781dc68f8f4ee415963cc  testfil
```

Vi ser, at filen i de to tilfælde indeholder de samme bogstaver, har samme længde (14 tegn), har samme datostempel (efter lidt snyd med touch-kommandoen) - men MD5-tjeksummen er helt forskellig. Derfor ved vi, at indholdet ikke det samme. Prøv selv at eksperimentere med dette.

Der er andre hash-funktioner end MD5, såsom MD4 og Snefru. Det er ikke så vigtigt, hvilken algoritme man anvender (selv om en inkarneret kryptoanalytiker vil natuligvis protestere her!). Det vigtige er at kontrollere. Men selvfølgelig er MD5 den bedste, og nemmeste, da den jo kan kontrollere ud fra en tidligere genereret filliste.

Vi vil nu omtale tre forskellige programmer til at hjælpe dig med at lave tjek af ændringer i filsystemet; RPM, Tripwire og Aide.

5.2. rpm -Va

Med Linux-distributionerne SuSE, Caldera Open Linux, Mandrake og Red Hat følger programmet "rpm", som er et pakkestyringsprogram. Rpm holder styr på, hvad der er installeret. Samtidig er der mulighed for at holde styr på ændringer af filerne. Du kan tjekke alle RPM-styrede filer med kommandoen "rpm -Va". Denne kommando vil tjekke for følgende ændringer siden installation:

- S - Ændringer af filstørrelse
- M - Mode - Ændringer i permissions og filtype
- 5 - Tjek af MD5-sum
- ? - Normalt problemer med at læse filen
- L - Symlink
- D - Device ændringer
- U - User - Ændringer i ejerskab
- G - Group - Ændringer i gruppeejerskab
- T - Mtime - Modifikationstidspunkt
- missing - Hvis en fil er forsvundet
- c - Opsætningsfil.
- . - Testen gik fint.

Bogstaverne til venstre er dem, der vises på skærmen, når man kører **rpm -Va**. Alle de filer, der er ændret i siden installationen, bliver vist. Hver af de ændrede filer vises efter formatet "SM5?LDUGT c filnavn", hvor hver af bogstaverne er forklaret ovenfor. Hvis den enkelte test gik godt, vises et punktum, ellers vises det bogstav for testen, der gik galt. Et typisk output kan være (i forkortet form)

Eksempel 5-8. rpm -Va

```
..?..... c /etc/securetty
S.5....T c /etc/services
....L...   /usr/local
.M......   /usr/local/bin
.M......   /usr/local/lib
..?.....   /usr/share/afterstep/ascp/help/animate.hlp
..?.....   /usr/share/afterstep/ascp/help/audio.hlp
..?.....   /usr/share/afterstep/ascp/help/autoexec.hlp
..?.....   /usr/share/afterstep/ascp/help/pager.hlp
missing    /usr/share/afterstep/ascp/icons/wharf2.xpm
missing    /usr/share/afterstep/ascp/icons/woptions.xpm
....L... c /etc/localtime
......G.   /etc/aliases.db
S.5....T c /etc/rc.d/init.d/sendmail
S.5....T c /etc/sendmail.cf
S.5....T c /etc/sendmail.cw
..5....T c /etc/sysconfig/sendmail
S.5....T   /var/log/sendmail.st
```

Nu er det op til dig, som systemadministrator, at vurdere hvilke ændringer, der er i orden, og hvilke, der kan være farlige. Der vil altid være ændringer, siden pakkerne er installeret, men nogle af ændringerne må gerne være der. Du ved forhåbentlig nogenlunde, hvad du selv har ændret. Ændringer, du ikke selv kan huske at have foretaget, kan vurderes ud fra, hvor filerne ligger og hvilken type filer, det drejer sig om. F.eks. virker filerne i `/usr/share/afterstep/` ikke særlig farlige. Det er nok bare noget opsætning af Window manageren Afterstep. Tegnet "?" tyder på at problemet i skyldes, at brugeren, der har udført kommandoen, ikke har rettigheder til at læse `/etc/securetty` og de fire hlp-filer. Vi kan se, at dette faktisk er tilfældet

```
[robin@hven robin]$ ls -al /etc/securetty
-rw-------  1 root  root  40 Sep 4  1995 /etc/securetty
```

Om filerne faktisk er ændrede, kan vi således først se, hvis vi kører "rpm -Va" som root. Vi kan også se, at der åbenbart mangler to xpm-filer (billeder) i "/usr/share/afterstep/ascp/icons".

Næste punkt er at kataloget `/usr/local` efter installation faktisk blev flyttet til en anden partition, så det er fint, at der er meldt fejl der. Filen `/etc/localtime` er et symbolsk link og en opsætningsfil ("L" og det lille "c"). Da Linux blev installeret, skulle man vælge hvilken tidszone man var i - og så ændrede filen sig naturligvis. Kun ved at se hvor linket peger hen, kan man se, at det nok _nu_ er i orden. Bemærk, at linket senere kan ændres til at pege på noget andet, måske noget "farligt", men vi vil stadig kun se samme fejl-meddelelse.

```
lrwxrwxrwx  1 root  root  39 May 15 16:27 /etc/localtime -> ../usr/share/zoneinfo/Europe/Co
```

Det ses også, at en stribe sendmail-filer er blevet ændret. Det passer med, hvad jeg som systemadministrator på maskinen ved, at der er lavet - men bemærk dog, at der kan være senere ændringer af de samme filer. På denne måde kan vi fortsætte. Viden om systemet er med andre ord nødvendig, men rpm er er godt sted at starte.

Ovenstående gennemgang med "rpm -Va" afslørede et grimt problem. Efter systeminstallation og systemtilpasning er der allerede sket en del ændringer. Man kan måske ikke overskue, hvad der er farlige ændringer, og hvilke der er banale. Et andet problem er, at "rpm -Va" ikke kan kontrollere programmer på systemet, der er installeret udenom rpm, såsom WordPerfect og StarOffice. Hvad værre er, så kan man teoretisk tænke sig, at en cracker afinstallerede en pakke, og installerede en farlig rpm-pakke i stedet for, hvor alle tjek med rpm gik glat. Eller installerede en falsk udgave af selve rpm programmet! Derfor er det ønskeligt at få et godt supplement til rpm.

Desuden er der Linux-distributioner, der ikke bruger rpm, f.eks. Slackware og Debian.

5.3. Tripwire

Tripwire fra 1992, skabt af Eugene Spafford og Gene Kim, og kildeteksten stod fra begyndelsen til rådighed for brugerne. Siden dengang har en kvart million brugere downloadet *Tripwire Academic Source.* Siden da er der kommet en kommerciel version med lidt flere features og support. (se http://www.tripwire.com). Til Linux findes der stadig en Open Source udgave, som er udgivet under GNU General Public Licence. Open Source udgaven kan hentes fra http://www.tripwire.org. Der er flere svar på den slags spørgsmål på dette site. (http://www.tripwire.org/qanda/index.php)

Tripwire er et program, der kan tjekke filsystemet for ændringer. Der genereres en database for alle filer i filtræet. For hver fil gemmes informationer om sidste ændringsdato, størrelse, indhold, ejer osv. Når først databasen er genereret, er det nemt at tjekke og opdatere for ændringer.

Den database, der genereres med Tripwire sker ud fra en opsætningsfil `/etc/tripwire/twpol.txt`, som beskriver hvilke dele af filsystemet, som er med i overvågningen og den opførsel de enkelte filer må tillades. Til eksempel må filen `/etc/shadow` godt ændre sig - da dette sker hver gang en bruger ændrer password. Modsat så ville det være højst underligt hvis filen `/bin/ps` har ændret sig over en nat. Det program bruges til at vise hvilke processer, der kører på systemet. Ergo skal opsætningen af overvågningen være skræddersyet til det enkelte system. I praksis betyder det at den Red Hat-opsætningsfil, der kommer med Tripwire skal sandsynligvis modificeres, hvis man vil køre med Debian eller lign.

Tankegangen med Tripwire er at man sætter programmet op *før* maskinen kommer på netværket og før brugere kommer på - dvs. før maskinen er i risiko-zonen. På dette tidlige tidspunkt genereres en "snapshot" af hvordan det oprindelige system ser ud. For hver af de vigtige filer på systemet gemmes information om fil-størrelse, ændringsdatoer og tjeksummer på filerne, så man kan afgøre om filerne er ændret.

Når maskinen så kommer i drift skal man køre Tripwire hver dag, time eller hvad der nu passer systemadministratoren bedst. Hver eneste gang genereres en rapport typisk per e-post til systemadministratoren over systemets status i forhold til systemets oprindelige "snapshot" - denne rapport skal der være tid til at læse roligt igennem!

Hvis rapporten viser at der er ændringer, så må systemadministratoren tage stilling til om det er i orden eller om der er tegn på indbrud - hvis det sidste er tilfældet, bør maskinen tages af netværket med det samme.

Det kan også ske at der er ændringer på systemet, som er i orden. Hvis man f.eks. installerer en ny RPM-pakke (fra en betroet kilde) så vil Tripwire melde at der er nye programmer typisk i `/usr/bin`. I dette tilfælde skal man så opdatere Tripwire-databasen med ens "snapshot".

5.3.1. Tripwire installation og opsætning

Tripwire installeres på Red Hat ved at man henter en `.tar.gz`-fil, der indeholder en RPM-pakke, som kan installeres. Tjek at den fil du henter har den rigtige tjeksum - jfr. http://www.tripwire.org/downloads/index.php. Kør **md5sum** på `.tar.gz`-filen:

```
[root@hven /root]# md5sum tripwire-2.3-47.i386.tar.gz
661a54a6429d4ecb0d756de5046da48f  tripwire-2.3-47.i386.tar.gz
[root@hven /root]# tar xzvf tripwire-2.3-47.i386.tar.gz
[root@hven /root]# rpm -ivh tripwire-2.3-47.i386.rpm
```

Derefter kommer der tekst på skærmen om hvad der skal gøres følgende: Først skal filen `/etc/tripwire/twpol.txt` ses igennem og rettes til. Det vigtigste at starte med i denne fil er følgende sektion (som startes med @@). Her er kommentarerne oversat til dansk.

```
@@section FS
SEC_CRIT      # Kritiske filer, der ikke må ændre sig
SEC_SUID      # Binære SUID eller SGID-filer, der ikke må ændre sig
SEC_BIN       # Binære filer, der ikke må ændre sig
SEC_CONFIG    # Opsætningfiler, der tilgås ofte og ændres sjældent
SEC_LOG       # Filer, der vokser i størrelse - ændrer ikke ejerskab
SEC_INVARIANT # Kataloger, som aldrig ændrer ejerskab eller rettigheder
```

I slutningen af afsnittet er der også inddelinger efter risiko-niveau.

```
SIG_LOW       # Ikke-kritiske filer med lav sikkerheds-risiko
SIG_MED       # Ikke-kritiske filer med væsentlig sikkerheds-risiko
SIG_HI        # Kritiske filer med væsentlig sikkerheds-risiko
```

Ideen er at man kan betegne filer og kataloger med de ovennævnte risiko-niveau. Til eksempel kan vi fortsætte ned i opsætningsfilen til "OS-Utilities":

```
  ################
 #              ##
################ #
#              # #
# OS Utilities # #
#              ##
################
(
  rulename = "Operating System Utilities",
  severity = $(SIG_HI)
)
{
  /bin/cat                              -> $(SEC_CRIT) ;
  /bin/date                             -> $(SEC_CRIT) ;
  /bin/dd                               -> $(SEC_CRIT) ;
...
}
```

Afsnittet er forkortet her. Forståelsen er at filer såsom `/bin/cat` er meget kritiske og må aldrig ændre sig i indhold, ejerskab eller rettigheder. Den lille tekst box over "rulename" er kun til pynt.

Opsætningsfilen `/etc/tripwire/twpol.txt` bør redigeres svarende til det enkelte system - dette kræver systemkendskab. Alternativet er at bruge den originale fil og måske opnå en falsk sikkerhed. Så tag lige tiden til at se filen igennem. Bemærk at der flere steder står anført linjer med hash-tegn foran, f.eks.

```
# /dev/printer -> $(SEC_CONFIG) ; # Uncomment if you have a printer device
```

Ideen er at man bør se disse linjer igennem specifikt for at se om du har de filer eller ej. Har man filen, så skal linjen typisk med ved at fjerne udkommenteringen.

Filer såsom `/proc/rtc`, `/proc/mdstat` og `/usr/sbin/fixrmtab` findes ikke på Red Hat 7.1 i standard installation - det er eksempler på filer, dermed linjer i opsætningsfilen som skal fjernes.

5.3.1.1. Opstart af Tripwire databasen

Efter opsætningsfilen er rettet til, så skal Tripwire opsætningen krypteres.

```
[root@hven /root]# /etc/tripwire/twinstall.sh
```

Det første der nu skal vælges er et løsen (eng. "passphrase") som anvendes til at kryptere databasen så andre ikke kan ændre den.

Nu skal der vælges fire løsener. De fire løsener bør principielt vælges forskellige, men en doven systemadministrator vil nok vælge dem ens (læs; dette er langt det nemmeste). Først for ens *site*, dvs. for hele maskin-parken. Dernæst for den enkelte maskine. Dernæst skal opsætningsfilen også signeres med

et løsen – ideen er at man heller ikke skal have lov til at kunne ændre den opsætningfil, som databasen er laves til `/etc/tripwire/tw.cfg` (Gemmes også i klar tekst i `/etc/tripwire/twcfg.txt`). Denne fil gemmer kun på systemparametre, såsom hvilet program der anvendes når tripwire-rapportering skal sendes til systemadministratoren. Til slut skal der vælges et løsen for resten af opsætningen, dvs. "policy"-delen, dvs. hvilke filer skal skannes og hvordan. Filen der gemmes er `/etc/tripwire/tw.pol` (Gemmes også i klar tekst under `/etc/tripwire/twpol.txt`). De to filer i klar tekst, mens `/etc/tripwire/tw.pol` og `/etc/tripwire/tw.cfg` endelig ikke må ændres manuelt - de er binære og krypterede.

For at resultater af skanning af filsystemet kan sendes til systemadministatoren per e-post så skal man indsætte e-postadressen for hver af de afsnit, der er defineret i filen. Bemærk at der er et komma til at afslutte linjen før `emailto = MODTAGERENS_ADRESSE`. Hvis man har tænkt sig at køre tripwire tjek via crontab, så er dette ikke nødvendigt, da man får output i et brev alligevel.

```
# Tripwire Binaries
(
  rulename = "Tripwire Binaries",
  severity = $(SIG_HI),
  emailto = sysadm@hven.dk
)
{
  $(TWBIN)/siggen                       -> $(SEC_BIN) ;
  $(TWBIN)/tripwire                     -> $(SEC_BIN) ;
  $(TWBIN)/twadmin                      -> $(SEC_BIN) ;
  $(TWBIN)/twprint                      -> $(SEC_BIN) ;
}
```

Nu skal Tripwire databasen genereres og du skal bruge dit løsen til den lokale maskine, som blev sat lige før dette. Det kan let tage 5-10 minutter alt efter maskintype og diskkapacitet.

```
[root@hven /root]# /usr/sbin/tripwire --init
```

Det mest sandsynlige er at man får ca. 50 fejl af filer, der ikke findes på systemet. De filer bør man nu udkommentere i `/etc/tripwire/twpol.txt` og lave en ny database. Dette kunne man have undgået ved at gennemgå `/etc/tripwire/twpol.txt` meget forsigtigt.

```
[root@hven /root]# /usr/sbin/twadmin --create-polfile /etc/tripwire/twpol.txt
Please enter your site passphrase:
Wrote policy file: /etc/tripwire/tw.pol
[root@hven /root]# rm -f /var/lib/tripwire/*.twd
[root@hven /root]# /usr/sbin/tripwire --init
Please enter your local passphrase:
Parsing policy file: /etc/tripwire/tw.pol
Generating the database...
*** Processing Unix File System ***
```

5.3.1.2. Tjek af maskinen med Tripwire

Nu er man klar til at tjeke maskinen - dette kan køres manuelt (dette kan man gøre uden brug af løsen - men man skal dog være root).

```
[root@hven /root]# /usr/sbin/tripwire --check
```

Det er lidt bøvlet at se gamle tripwire rapporter, da disse også er krypterede. Kommandoen er som følger, hvor XX skal erstattes af det filnavn, man har på disken. De er navngivet efter maskinnavn og dato.

```
[root@hven /root]# /usr/sbin/twprint -m r --twrfile /var/lib/tripwire/report/XX.twr
```

Lad os nu prøve at lave om på maskinen. Vi opgraderer procmail RPM-pakken og se hvad Tripwire finder. Bemærk de 5 modificerede "user binaries", som dernæst vises nedenfor.

```
Note: Report is not encrypted.
Tripwire(R) 2.3.0 Integrity Check Report

Report generated by:          root
Report created on:            Wed Oct 31 00:29:46 2001
Database last updated on:     Never

===============================================================================
Report Summary:
===============================================================================

Host name:                    k6.sslug
Host IP address:              192.168.1.3
Host ID:                      None
Policy file used:             /etc/tripwire/tw.pol
Configuration file used:      /etc/tripwire/tw.cfg
Database file used:           /var/lib/tripwire/k6.sslug.twd
Command line used:            /usr/sbin/tripwire --check

===============================================================================
Rule Summary:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

  Rule Name                     Severity Level    Added    Removed  Modified
  ---------                     --------------    -----    -------  --------
  Invariant Directories         66                0        0        0
  Temporary directories         33                0        0        0
* Tripwire Data Files           100               1        0        0
  Critical devices              100               0        0        0
* User binaries                 66                0        0        5
  Tripwire Binaries             100               0        0        0
  Libraries                     66                0        0        0
```

```
  File System and Disk Administraton Programs
                                100               0        0        0
  Kernel Administration Programs  100               0        0        0
  Networking Programs             100               0        0        0
  System Administration Programs  100               0        0        0
  Hardware and Device Control Programs
                                100               0        0        0
  System Information Programs     100               0        0        0
  Application Information Programs
                                100               0        0        0
  Shell Releated Programs         100               0        0        0
  (/sbin/getkey)
  Critical Utility Sym-Links      100               0        0        0
  Critical system boot files      100               0        0        0
  Critical configuration files    100               0        0        0
  System boot changes             100               0        0        0
  OS executables and libraries    100               0        0        0
  Security Control                100               0        0        0
  Login Scripts                   100               0        0        0
  Operating System Utilities      100               0        0        0
  Shell Binaries                  100               0        0        0
  Root config files               100               0        0        0

Total objects scanned:  19398
Total violations found:  6

===============================================================================
Object Detail:
===============================================================================

-------------------------------------------------------------------------------
  Section: Unix File System
-------------------------------------------------------------------------------

-------------------------------------------------------------------------------
Rule Name: User binaries (/usr/bin)
Severity Level: 66
-------------------------------------------------------------------------------
  ---------------------------------------
  Modified Objects: 5
  ---------------------------------------

Modified object name:  /usr/bin

  Property:            Expected                   Observed
  -------------        -----------                -----------
* Modify Time          Sat Oct 20 17:15:27 2001   Wed Oct 31 00:29:44 2001


Modified object name:  /usr/bin/formail

  Property:            Expected                   Observed
  -------------        -----------                -----------
```

```
* Inode Number         443138                    442309
* Size                 26736                     26900
* Modify Time          Sat Jan  6 23:13:11 2001  Wed Jul  4 00:55:12 2001
* CRC32                CB/Sfy                    DJdys6
* MD5                  A4Lbe9dcQUPMqvFeZQ/qnF    CazhNDHWCoxKh46uXHElej


Modified object name:  /usr/bin/lockfile

  Property:            Expected                  Observed
  -------------        -----------               -----------
* Inode Number         443139                    442310
* Size                 11124                     12440
* Modify Time          Sat Jan  6 23:13:11 2001  Wed Jul  4 00:55:12 2001
* Blocks               24                        32
* CRC32                CWnVmO                    A6qdab
* MD5                  C2Lvt5A+Kx3cjOYgGCdHd9    BQpFs92xGV/D+G+aPGbG8T


Modified object name:  /usr/bin/mailstat

  Property:            Expected                  Observed
  -------------        -----------               -----------
* Inode Number         443140                    442311
* Modify Time          Sat Jan  6 23:13:10 2001  Wed Jul  4 00:55:11 2001


Modified object name:  /usr/bin/procmail

  Property:            Expected                  Observed
  -------------        -----------               -----------
* Inode Number         443141                    442312
* Size                 63484                     75420
* Modify Time          Sat Jan  6 23:13:11 2001  Wed Jul  4 00:55:12 2001
* Blocks               136                       160
* CRC32                A4Xs3q                    D/ZVXm
* MD5                  Af2+FBsxRo0LTeWeLim2iB    Bez1qNLwdxqPMYR4GPq8EB



--------------------------------------------------------------------------------
Rule Name: Tripwire Data Files (/var/lib/tripwire)
Severity Level: 100
--------------------------------------------------------------------------------
  ---------------------------------------
  Added Objects: 1
  ---------------------------------------

Added object name:  /var/lib/tripwire/k6.sslug.twd

================================================================================
Error Report:
================================================================================
```

```
No Errors

-------------------------------------------------------------------------------
*** End of report ***

Tripwire 2.3 Portions copyright 2000 Tripwire, Inc. Tripwire is a registered
trademark of Tripwire, Inc. This software comes with ABSOLUTELY NO WARRANTY;
for details use --version. This is free software which may be redistributed
or modified only under certain conditions; see COPYING for details.
All rights reserved.
```

5.3.1.3. Opdatering af Tripwire-databasen

Nu er der selvfølgelig problemer - er det rimeligt at de filer at ændret? I dette tilfælde er det, da filerne kommer fra den nye RPM-pakke. Generelt skal man køre Tripwire-skanningen før man installerer en RPM-pakke for at være sikker på at der ikke er andre ændringer. Dernæst skal man så opdatere databasen med systemets snapshot, så man ikke længere ser de opgraderede pakker med i listen. Igen skal XX erstattes med den seneste .twr-fil for maskinen.

```
[root@hven /root]# /usr/sbin/tripwire --update --twrfile /var/lib/tripwire/report/XX.twr
```

Op kommer en fil startet i **vi**-editoren, hvor en bid af den er som følger:

```
Remove the "x" from the adjacent box to prevent updating the database
with the new values for this object.

Modified:
[x] "/usr/bin"
[x] "/usr/bin/formail"
[x] "/usr/bin/lockfile"
[x] "/usr/bin/mailstat"
[x] "/usr/bin/procmail"
```

Tripwire foreslår, at man opdaterer alle de ændrede filer, så Tripwire ikke brokke sig over dem igen. Hvis du ikke vil ændre så fjern det kryds (dvs. x) som er i tekst-boksen. Når du forlader editoren skal man igen angive løsen for at opdatere. Herefter vil Tripwire ikke brokke sig over den opdaterede procmail pakke.

Tripwire bør indsættes i `/etc/cron.daily`, dvs. det bliver kørt en gang hver dag. Resultatet bliver sendt til root på maskinen per e-post.

```
[root@hven /root]# cat /etc/cron.daily/t
#!/bin/sh

/usr/sbin/tripwire --check
[root@hven /root]# chmod +x /etc/cron.daily/t
```

Bemærk at det her ikke at valgt at kalde filen `tripwire`, som ville være naturligt. Men det giver jo kun en angriber nemmere ved at se at tripwire er sat op på maskinen. Det er vigtigt at holde øje med at brevet fra tripwire kommer på ca. det tidspunkt som de *bør* komme på jfr. `/etc/crontab`.

Du kan finde en anden vejledning til at installere og bruge Tripwire på http://www.net-security.org/text/articles/tripwire.shtml.

5.4. Aide

Et interessant alternativ til Tripwire var Aide (Advanced Intrusion Detection Environment), der kan findes på http://www.cs.tut.fi/~rammer/aide.html. Versionen er pr. oktober 2003 aide-0.9, hvilket indikerer, at Aide ikke er helt færdigt. Det ser ud som om Aide ikke har udviklet sig meget siden Tripwire blev Open Source, nyeste file i aide-0.9 er fra Juni 2002.

Aide er Open Source (udgivet under GPL), det vil sige, at det kommer med komplet kildetekst. Aide oversættes med **./configure** og **make**, og installeres som systemadministrator (root) med **make install**.

Efter oversættelsen og installationen ligger der en test-opsætningsfil i doc-kataloget under det katalog, hvor du oversatte aide. Du kan med **man aide.conf** se lidt om syntaksen i opsætningsfilen. Selv aide-programmet er installeret under `/usr/local/bin` (medmindre du har ændret i opsætningen). For at generere en database over dine filer, skrives

```
[root@hven /root]# aide --init
```

Hvor databasen kommer til at ligge, afgøres af opsætningsfilen, ligesom den afgør hvilke filer, der skal med i databasen. Ligeledes er de regler, der afgør hvilke tjek, der skal udføres, defineret i aide.conf. Som standard leder aide efter opsætningsfilen i det katalog, hvor du står, når du kører programmet. Opdatering af databasen sker med

```
[root@hven /root]# aide --update
```

og tjek af ændringer med

```
[root@hven /root]# aide --check
```

Se i øvrigt **man aide**.

Vi ville gerne anbefale dette lovende program, men det er ikke færdigt endnu, og hvis du bruger det, er det på eget ansvar. Vi tør imidlertid godt spå en lysende fremtid for aide og vil opfordre til, at man følger med i udviklingen og evt. bidrager til projektet.

5.5. Generelt om skanning af filsystemet

Ofte benyttede programmer til at skanne filsystemet kan findes på:

- F.I.R.E http://biatchux.dmzs.com/ (http://biatchux.dmzs.com/).
- http://www.mycert.mimos.my/resource/fic.htm.
- http://www.knoppix.org/, (http://www.knoppix.org/) som jo kan bruges til at boote fra, så man er sikker på en kerne, der ikke er kompromitteret.
- http://www.knoppix-std.org/tools.html (http://www.knoppix-std.org/tools.html), Knoppix Security Tool Distribution, har mange værktøjer til sikkerhedscheck.
- Radmind er en anden file-checker. http://rsug.itd.umich.edu/software/radmind/.

Når du skal tjekke filsystemet, bør du boote maskinen via en "sikker" Linux-kerne. Det kan være fra en diskette eller cd-rom. Ligeledes bør du bruge en cd-rom eller andet read only medie, til at gemme program og database over filsystem på. Ellers kan du ikke tjekke filtræet mod tidligere skanninger og være sikker på, at du får de reelle forskelle, da en cracker så vil kunne ændre i din gemte database, filtjekprogrammet eller i selve Linux-kernen.

Kapitel 6. Firewall på Linux

6.1. Hvad er en firewall?

At forbinde sin virksomheds eller sit kollegiums lokale computernetværk direkte til internettet kan være en risikabel ting at gøre, for man åbner jo for trafik begge veje, altså også for trafik ind på det lokale netværk. Hvis man uden videre forbinder sit lokale netværk til internettet, kan man blive udsat for, at andre trænger ind på firmaets eller kollegiets interne computere, læser og ændrer følsomme data, får servere til at gå i stå osv. Ønsker man at give adgang for beboere og medarbejdere til e-post og den store mængde information, man kan finde på internettet uden at åbne for trafik den anden vej, kræver det en firewall.

Firewall er det engelske ord for en brandvæg eller en brandmur. I bygninger bruges en brandmur til at dele en bygning i sektioner. Hvis der opstår en brand i den ene sektion, kan brandmuren forhindre at branden breder sig (hurtigt) til andre sektioner.

En computer som firewall bruges til på lignende vis at opdele et netværk i sektioner, typisk i det, der er uden for lokalnetværket, og det, som er indenfor. Firewallen er først og fremmest et pakkefilter, som forhindrer pakker af bestemte typer i at slippe igennem fra det ene til det andet netværk. Linux firewallen kan imidlertid meget mere end det, og i denne sammenhæng er det vigtigt, at den kan lade trafik, som har oprindelse på lokalnettet, gå til nettet udenfor, det offentlige net, og vel at mærke tillade svarpakkerne at passere på tilbagevejen.

Man kan opnå en fremragende sikkerhed med en Linux box som firewall, men det er imidlertid bedst at kombinere firewallen med de forholdsregler for sikker opsætning af en computer, som er omtalt tidligere i bogen.

En firewall kan selvfølgelig også bruges til at adskille to eller flere lokalnetværk fra hinanden.

Der findes forskellige måder at opbygge en firewall, næste afsnit lister de vigtigste.

6.1.1. Firewall typer

- Pakkefiltrering
- Proxy, en applikation som optræder som mellemmand
- Stateful Multi-Layer Inspection, SMLI, en kombination af pakkefilter, tilstands-betingelser og indholds-filtrering

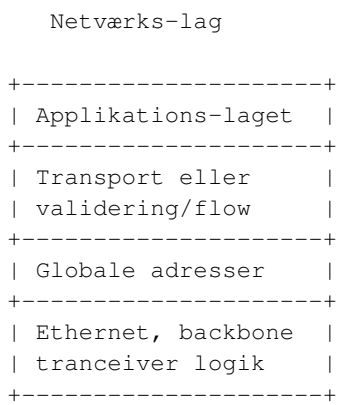
Et pakkefilter ser på pakkens IP-adresse, den protokol-type, som den hører til, og eventuelt flere ting.

En proxy, som egentlig blot betyder en stedfortræder, var oprindeligt et program på en gateway, som man inde på lokalnettet henvendte sig til, og så kunne dette program sende forespørgsler videre, som om de kom fra proxy-maskinen. Det er en meget sikker, men også ret besværlig måde at skille lokalnet fra offentlige net, og det kræver en proxy for hver protokol, man ønsker at åbne for, http, ftp, ssh, og hvad det nu kunne være.

En mere moderne proxy kan fungere uden at man henvender sig specielt til den. Det kaldes en transparent proxy. Den lytter efter al trafik og videresender de pakker, som er tilladte, ligesom et pakkefilter, men i modsætning til pakkefilteret kan proxyen altså sende dem videre, som om de kom fra den selv, og desuden inspicere dem på indholdsniveau eller applikationslag.

I forbindelse med firewalls taler man ofte om netværkslag, fordi det gør det muligt at beskrive mere præcist, hvad der sker. Den forsimplede Department of Defence-model, DoD 4-lags modellen, er en god hjælp til de fleste formål:

6.1.1.1. Netværks-lag



Nogle vil måske foretrække at få en forklaring på, hvorfor man i det hele taget har lag, og så er det bedst at tage forklaringerne fra ledningerne, det fysiske netværk og opefter.

- Fysisk link-lag, det vil sige ethernetkort og ledninger (der er egentlig også et par abstraktionslag her); ofte kaldes det for det nederste lag; det skyldes måske, at ledningerne som regel liger på gulvet og føres ud gennem kælderen - hvis man stadig brugte telefonstolper, ville man nok kalde det for det øverste lag;-))
- netværkslaget, her finder man IP-numre, ping programmet og ICMP; dette lag har ansvaret for at en pakke kan komme fra Jylland til Australien, det vil sige ansvaret for at forbinde forskellige net med hinanden, herunder også fragmentering af pakker, som er for store til det underliggende fysiske net;
- transportlaget, TCP, som er forskelligt fra IP-laget derved at det sørger for at data er valide, det vil sige at der oprettes handshake; nyere TCP implementationer kan også håndtere flow-kontrol: parametre på fysisk niveau kan styres indirekte af TCP oplysninger om, hvor meget modtageren kan klare, førend man får data galt i halsen; man kunne også kalde TCP laget for validerings- og flow control lag;

- applikationslaget, higher level, for eksempel FTP og HTTP.

6.1.1.2. Filtrering på applikationslag

En proxy kan inspicere på applikationslaget. Den kan derfor kaldes for et censur-program, netværks-barnepige eller spion.

Stateful Multi Layer Inspection (SMLI) er den mest avancerede form for firewall. Det er faktisk muligt at lave den slags filtrering med Linux-kernen. Udvalgte pakker kan med iptables kommandoen sendes til et program, som inspicerer dem og som kan afgøre, om de skal sendes videre eller ej. En forudsætning for det er, at Linux' indbyggede pakkefilter har mulighed for at holde styr på det, man kalder en forbindelse (connection), det vil sige en session, hvor man indleder en konversation med en server, udveksler data og ender med at sige "nu er jeg færdig".

TCP benytter sådanne connection-meddelelser. Linux-kernen kan holde styr på status for en forbindelse. Derfor kaldes Linux Netfilter for et State Packet Filter, SPF.

Derved kan man bygge en SMLI firewall baseret på Linux Netfilter. Endvidere kan en user-applikation inkorporeres i kernen (for at effektivisere den), så man kan skam bygge tiptop professionelle SMLI-firewalls på basis af Linux.

6.1.2. Grundlæggende pakkefiltrering eller politiske overvejelser

Opsætningen af en firewall kan enten bygge på den grundlæggende antagelse, at alt, hvad der kommer udefra, er mistænkeligt og bør stoppes - eller, omvendt, at vi tror på det gode i netværket og kun spærrer af for de ting, som notorisk er nemme at misbruge.

6.1.2.1. Luk alt

Hvis man lukker for alt, er det selvfølgelig overflødigt med en netforbindelse ud af huset. Men det er nu alligevel den bedste måde at sikre systemet på (;-) For at få lidt glæde af netforbindelsen kan man så åbne for web-protokollen HTTP. En TCP/IP-pakke har ud over IP-nummeret et port-nummer med sig, som blot er et tal (16 bit) i TCP-pakkens header. På modtagermaskinen ser TCP-softwaren i kernen efter, hvilket portnummer der er, og om der er noget program, som har meldt sig som interesseret i pakker til dette portnummer. Det kalder man, at et program har registreret på den pågældende port. Et program, som lytter efter indkommende forbindelser, kaldes en server (eller somme tider et serverprogram.)

HTTP-protokollen er karakteristisk ved, at serveren som regel lytter på port 80. Vi kan derfor lave en firewall regel, som siger, at al trafik, som er til eller fra port 80 på en computer *udenfor lokalnetværket*

skal have lov at passere. Det er en løsning, men en fattig løsning, idet man kommer til at mangle Domain Name Service, DNS, og endvidere kommer til at lide under, at nogle web-sites forventer, at man kan bruge FTP til download, eller ligefrem flytter brugerens forespørgsel til en server på et andet portnummer.

En netværksforbindelse mellem to applikationer defineres af IP-adresse plus portnummer. En sådan identificeret forbindelse kaldes en socket.

```
SOCKET-illustration

Afsender (browser)                      Modtager (http-server)

IP-adresse  + portnummer                IP-adresse   + portnummer

201.2.3.4     51001          <----->    193.88.44.22   80

En anden afsender (wget)                Modtager (samme http-server)

201.2.3.4     47011          <----->    193.88.44.22   80


En socket definerer en forbindelse,
typisk fra applikation til applikation.
```

6.1.2.2. Lad alt være åbent - undtagen de farlige ting

For at undgå problemer kunne man så beslutte, at man lader det meste stå åbent og - selvfølgelig - lukker for de farlige services som telnet, remote shell og så videre.

Et eksempel kunne være, at man lokalt gerne må bruge telnet, men at det er sket at folk udefra har forsøgt at få forbindelse via telnet. Derfor kan det se ud som en løsning at lukke for port 23. Den løsning er dog heller ikke særlig god. Intrusion på en computer kan sagtens benytte sig af andre portnumre (og andre protokoller) og ved snedige, ulovlige datapakker forsøge at sætte en kæp i hjulet på de programmer, der lytter. Hvis vi var flittige og lappede alle hullerne i serverprogrammerne og i øvrigt overvågede maskinen døgnet rundt, så ville denne løsning være god nok, men den er dyr i arbejdstimer.

6.1.3. Den lukkede model bør vælges som udgangspunkt

Hverken den åbne eller helt lukkede model er særligt nyttige. Der skal mere til. Som udgangspunkt er den lukkede model imidlertid den sikreste, men den skal kombineres med connection-tracking.

Forhåbentlig fremgår det af ovenstående, at en firewall egentlig er et ret kompliceret stykke software. Man skal nu ikke lade sig skræmme væk af den grund. Det er i grunden ret nemt at skrive de nødvendige kommandoer og så blot kontrollere en gang i mellem, at det kører som det skal.

Man bruger **iptables** kommandoen til at opsætte firewall reglerne i kernen. Men man behøver ikke at kunne **iptables** kommando syntaksen, idet SuSE, Red Hat og andre lavet GUI-administration af firewall reglerne. Så hvis man holder sig til de gode, kendte distributioner eller på anden måde sørger for, at de nødvendige ting er til stede, så kan man blot krydse af, hvilke servere andre udefra skal have lov at bruge, om nogen. Er man af gør det selv typen, følger her en opskrift på, hvordan man kan lave en lille komplet firewall-opsætning.

6.1.3.1. Lynkursus

Senere bliver kommandoerne til en firewall beskrevet og forklaret grundigt, men for den utålmodige, som bare vil i gang, kan vi lige vise "Rusty's Really Quick Guide To Packet Filtering" fra packet-filtering-HOWTO. Vi antager, at maskinen har en modemforbindelse til en ISP og ikke ønsker, at man skal kunne tilgå maskinen udefra:

```
## Insert connection-tracking modules (not needed if built into kernel).
# insmod ip_conntrack
# insmod ip_conntrack_ftp

## Create chain which blocks new connections, except if coming from inside.
# iptables -N block
# iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT
# iptables -A block -m state --state NEW -i ! ppp0 -j ACCEPT
# iptables -A block -j DROP

## Jump to that chain from INPUT and FORWARD chains.
# iptables -A INPUT -j block
# iptables -A FORWARD -j block
```

De første kommandoer er nødvendige med de fleste distributionskerner. Det er ikke spor svært at lave sin egen superkerne, hvis man interesserer sig for, hvad en kerne er, og man har lært noget om, hvordan et operativsystem fungerer. Har man en kerne hvor alt er kompileret ind i kernen, vil systemet være en anelse hurtigere[1].

iptables -N block laver en kæde (en skuffe eller hylde) for et regelsæt. Regelsættene kaldes chains, fordi man bruger reglerne i den rækkefølge, de er sat ind i kæden. Kæden her hedder "block", blokér.

iptables -A block -m state --state ESTABLISHED,RELATED -j ACCEPT

-A block indsætter en regel i block-kæden, som siger, at man skal bruge tilstands-modulet, **-m state** , og undersøge, om pakken tilhører en forbindelse (connection), som allerede er startet, eller som er i slægt

med en connection **--state ESTABLISHED,RELATED**, hvis det er tilfældet, skal pakken accepteres, d.v.s. regelfortolkeren jumper (-j) hopper til feltet for videresendelse af pakker, "ACCEPT".

Næste linje siger, at alle pakker, som *ikke* kommer udefra (i dette tilfælde fra et modem som bruger Point-to-Point-Protokollen, PPP), skal have lov at passere.

Alt andet smides simpelt hen væk! DROP!

Og de sidste par linjer bevirker, som kommentaren siger, at både pakker, som er INPUT og pakker og FORWARD pakker kommer igennem den regelkæde, som vi har dannet. Dette er, som nævnt, et eksempel for den utålmodige, men for den tålmodige er der en nærmere forklaring af kommandoen og mulighederne med netfilter i Afsnit 6.5.3.

Det er vigtigt at forstå, at en firewall ikke er spor sikker, hvis den er sat forkert op. Når firewallen er sat op, bør man grundigt teste, at den nu også gør, som man forventer. En vigtig del af sikkerheden omkring en firewall er desuden, at man registrerer og overvåger den trafik, der passerer igennem den. Hvis man glemmer at holde øje med de log-filer, firewallen genererer, er ens sikkerhed reelt væk (Se også Kapitel 5). Man kan så risikere, at man ikke opdager, at der har været indbrud. En firewall er aldrig helt sikker, og man bør kun sætte en firewall op, hvis man har i sinde at vedligeholde den.

En forkert opsat firewall eller en firewall, ingen holder øje med, kan være værre end ingen firewall! Tilstedeværelsen af en firewall giver let en falsk fornemmelse af tryghed, så man ikke er omhyggelig nok med at beskytte hver enkelt computer. Det er også meget vigtigt, at man holder sig ajour med nyheder om fejl i den software, der anvendes på firewallen (Se Afsnit 1.4).

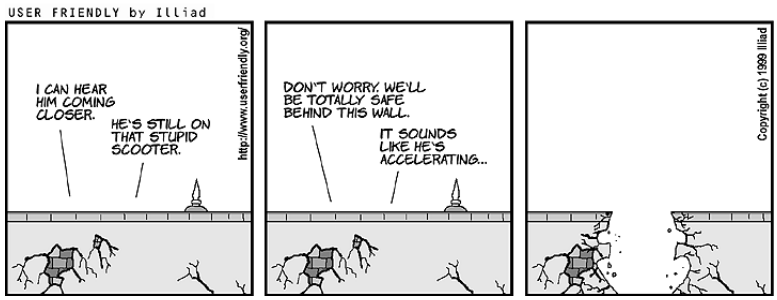
Selvom man har sat en firewall op for at beskytte et netværk imod folk på internettet, bør man stadig være omhyggelig med sikkerheden i det lokale netværk. For det første kan det være, at nogen bryder igennem ens firewall, og dermed er der kun lokalnetværkets sikkerhed tilbage. For det andet beskytter en firewall ikke imod internt misbrug af ens netværk. I en virksomhed eller på et kollegium eller lignende kan man ikke altid stole på de lokale brugere. Opgørelser fra IBM peger på, at der ca. er ligeså mange interne netværksindbrud som eksterne.

Et andet problem inden for firewallen er lokale modems. Hvis en bruger har et modem, hvormed han kobler sig direkte til sin hjemmearbejdsplads, der ikke er sikret, så er der via denne hjemmeopkobling fri bane til virksomhedens net. Der er mange variationer på dette sikkerhedshul, for eksempel også at nogen medbringer en transportabel maskine og sætter den på kollegiets eller virksomhedens net. På Linux/GNU/BSD platforme sikrer man den enkelte maskine bedst ved at den er firewall for sig selv.

Så det er vigtigt at bemærke, at en firewall ikke er en universalløsning på sikkerhed, som mange tror i dag. Den er god at have, hvis man er indstillet på at gøre det ordentligt. Men den er ikke en erstatning for alle de andre sikkerhedsforanstaltninger på et netværk.

Man er aldrig helt sikker, selv med en firewall ...

Figur 6-1. User Friendly



6.2. Links

I dette kapitel henvises der til en række HOWTO's og andre dokumenter rundt om i teksten. Her er en hurtig oversigt:

HOWTO's:

- Linux IPCHAINS-HOWTO http://sunsite.dk/ldp/HOWTO/IPCHAINS-HOWTO.html
- Firewall and Proxy Server HOWTO http://sunsite.dk/ldp/HOWTO/Firewall-HOWTO.html
- Transparent Proxy with Squid mini-HOWTO http://sunsite.dk/ldp/HOWTO/mini/TransparentProxy.html
- Linux 2.4 Packet Filtering HOWTO: http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc.html (http://www.netfilter.org/unreliable-guides/packet-filtering-HOWTO/packet-filtering-HOWTO.linuxdoc.html)
- Linux 2.4 NAT HOWTO http://netfilter.samba.org/unreliable-guides/NAT-HOWTO/index.html
- (Linux IP Masquerade HOWTO http://sunsite.dk/ldp/HOWTO/IP-Masquerade-HOWTO.html
- Oversigt over Linux distroer specielt til firewalls: http://distrowatch.com/dwres.php?resource=firewalls (http://distrowatch.com/dwres.php?resource=firewalls)

  Der findes mange forskellige distro'er inden for det emne. F.eks. http://www.ipcop.org/

  Eller måske en kombineret firewall og web/mail/db/ftp server: http://www.e-smith.org/ (Gert Lavsen bruger den på en 266MHz PII)

Andet:

- Sikkerhed på Linux http://www.sslug.dk/artikler/Linux_sikkerhed/
- Linux - Friheden til at vælge www.linuxbog.dk/bog/ (http://www.linuxbog.dk/bog/)
- IPChains i Linux 2.2 http://www.sslug.dk/sikkerhed/ipchains.html
- Netfilter i Linux 2.4 http://www.sslug.dk/sikkerhed/netfilter.html
- The Netfilter Project HomePage http://netfilter.samba.org/
- Squid Web Proxy Cache http://www.squid-cache.org/

6.3. Firewallens funktioner

Linux firewallen vil på et mindre lokalnet som regel have flere funktioner. Den skal kontrollere, hvilken trafik, der kan tillades, den skal sikre os mod angreb, og endelig kan den overvåge trafikmængder og for eksempel fortælle os, om et eller andet program begynder at udspy pakker på internettet. Det sker somme tider. Og måske er man bare almindeligt nysgerrig og ønsker at se, hvad der sker på maskinen. Endvidere vil vi også gerne kunne servicere maskiner, som ikke har et offentligt IP-nummer.

- Kontrol af trafik-typer
- Sikring mod indtrængen
- Overvågning, logning
- Servicering af maskiner uden offentligt IP-nummer

6.3.1. Kontrol af trafik-typer.

Data sendes igennem et netværk i form af datapakker. På internettet bruges IP-pakker. En IP-pakke består af en header med information om, hvor den skal hen, hvor den kommer fra, hvad den er for en slags samt eventuelt nogle data. Ved pakkefiltrering kigger man på informationen i headeren og bestemmer, hvad der skal ske med pakken: Om den skal afvises eller slippes igennem.

Ideen er så, at en firewall stopper alle de pakker, som man ikke har "inviteret". Kører man f. eks. en web-server, slipper man kun web-pakker igennem. En privat maskine med internetadgang kan godt stoppe alle udefra kommende forespørgsler samtidigt med, at alle de netværks-sessions, der startes af brugeren får lov til at køre som normalt, d.v.s. at *svar-pakker* fra det eksterne net får lov at komme igennem.

Man bør blokere:

- Pakker fra services, der ikke skal bruges. Hvis en server f.eks. ikke tilbyder FTP, er det klogt at afvise alle forsøg på at etablere en FTP-forbindelse med det samme.
- Pakker med mistænkelige bits sat som IP source routing (kan bruges til spoofing, hvilket betyder at udgive pakkerne for at komme fra et andet sted, end de egentlig kommer fra).
- Pakker, der er til en lokal host, som ikke burde modtage pakker udefra. Det vil sige pakker, som er direkte addreseret til en maskine, som ikke har lov til at snakke med fremmede maskiner på internettet.

Hvis for eksempel Boligforeningen Xyzzy-42 ikke ønsker, at der skal være adgang til FTP-serveren fra maskiner udefra, men gerne vil se, hvor mange, der prøver at komme i forbindelse med serveren, så kan nedenstående kommando bruges.

```
[root@hven /]# iptables -A INPUT -p tcp --destination-port 21 -j DROP
```

For mere information om IP, TCP og netværk generelt se f.eks. "Introduktion til Netværk" af Geir Steen-Olsen & Arne Stalheim (IDG-bog til 69 kr.) eller "Linux - Friheden til til systemadministration" på www.linuxbog.dk (http://www.linuxbog.dk/). Den grundige gennemgang af IPtables begynder Afsnit 6.5.2

6.3.2. Sikring mod indtrængen

Firewallens vigtigste opgave er selvfølgelig at hindre uvedkommende at trænge ind på systemet. Det bedste er at spærre for alt og åbne for en positivliste, som kan bestå af de tilladte brugeres IP-numre. Hvis man derimod vil tillade bruger XYZ at komme ind gennem vores firewall, uanset hvilken maskine XYZ benytter, ses (let) at det ikke kan lade sig gøre ved hjælp af firewall-regler. Man kan ikke på netværksniveau eller transportniveau åbne for en bestemt loginkonto, men her kommer SSH, secure shell, protokollen heldigvis til undsætning og løser problemet elegant og sikkert på session-niveau.

6.3.3. Overvågning og logning

Når man installerer en regel i kernens pakkefilter, holder kernen automatisk øje med, hvor mange pakker der falder ind under denne regel, og med -L (list) kommandoen få iptables til at vise tællerne (counters).

Et eksempel:

```
[root@hven /]# iptables -L INPUT -v
```

6.3.4. Servicering af maskiner uden offentligt IP-nummer

En router for et lokalnetværk har som regel flere opgaver end at sende pakker videre og stoppe

uvedkommende pakker. I de fleste netværk bruger man adresser, som er reserveret til testformål, de såkaldte private adresser. De skal ompakkes til en rigtig IP-adresse, og hertil bruger routeren NAT, Network Address Translation.

NAT er en teknik, som minder om en transparent proxy. NAT-ing har vist sig at være en halv løsning på mangelen på IP-numre. IP-numre er 32 bit. Hvis man kunne udnytte alle 4,294,967,295 numre (4 mia.), så ville det stadig ikke være nok til at dække behovet. Der er ca. 0.5 mia adresser, som bliver brugt til specielle formål, men selv hvis man kunne inddrage dem til "almindelige" adresser, så ville der ikke være nok numre til at dække behovet for at hvert menneske havde et IP-nummer, for slet ikke at tale om, hvis vi skulle have IP-numre til arbejde og hjem, køleskab, cykel og fjernsyn.

Hvis to maskiner på et net udgiver sig for at have samme IP-nummer, så vil kommunikationen blive upålidelig for begge maskiner.

Da man jo ikke behøver at koble et lokalnet på det globale internet, behøver man heller ikke at få en adresse på det globale net fra InterNIC eller deres lokale repræsentant. Man kunne sådan set godt bruge de numre, man synes så pænest ud. Imidlertid viser det sig ofte, at folk alligevel gerne vil have deres separate TCP/IP-net tilsluttet det globale internet på et eller andet tidspunkt, og derfor indskærpes det overfor netværksadministratorer, at man skal bruge de "private adresser", selv om man ikke lige nu har tænkt sig at slutte maskinerne til det offentlige net. De private adresser får ikke lov til at passere en router, det er indbygget i routeren. På den måde kan man undgå uskyldige fejltagelser, hvor en maskine har fået et IP-nummer, som tilhører en anden.

De mest benyttede private adresser er intervallet 192.168.1.0 - 192.168.255.255. Man kan også bruge alle adresser, som begynder med 10, og adresser, som begynder med 172.16

Det er så routerens opgave at foretage en omskrivning af pakkerne og sende dem videre, som om de kom fra routerens eksterne, offentligt synlige IP-adresse. Lige så vigtigt er det, at den kan identificere svar-pakkerne, og sende dem til den rigtige maskine. Dertil bruges portnumrene som en forlængelse af IP-adressen. Denne funktion kaldes *N*etwork *A*ddress *T*ranslation eller NAT. En router, som kan det, kaldes somme tider en NATting router.

Den utålmodige læser kan nu springe frem til Afsnit 6.5.2, mens den grundige læser i de næste afsnit kan få meget mere at vide om, hvordan Linux fungerer.

6.4. Firewallens virkemåder

Som sagt kan en Linux-pc fungere som firewall og router. En firewall-router er dog ofte en dedikeret, flad boks, som egentlig er et diskløst system, der kan fjernstyres fra en anden computer. Der fås billige routere til mindre virksomheder og private net, og der fås også store routere med mange stik, som kan håndtere meget store trafikmængder. Der er en Linux-kerne inde i mange af den slags dedikerede routere. TODO: eksempler, billede.

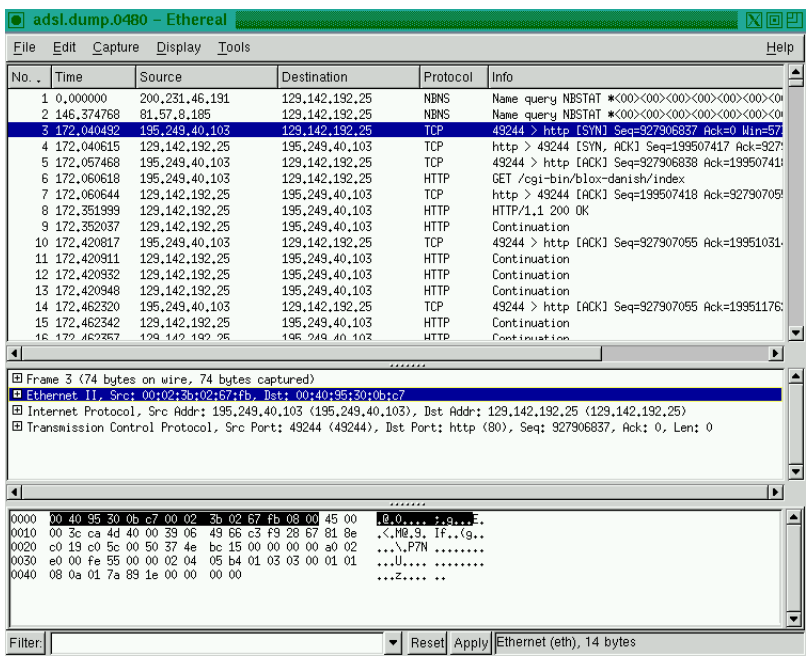
Inde i firewallen inspiceres pakkerne. Hvordan man gør, og hvor meget det koster i tid, og hvilke problemer, der kan opstå, er emnet for dette afsnit.

Netværksmodellen (Afsnit 6.1.1.1) viser to ting. Dels viser den abstraktionslagene, som giver de elektriske signaler mening. Dels viser modellen, at en applikation, som ønsker at sende data (typisk tekst) til en anden applikation på en remote værtsmaskine, skal sende sin pakke gennem adskillige instanser (lag svarer her til programafsnit eller kernefunktioner). Hver gang pakken passerer et lag, bliver det hele lagt i en slags "konvolut". Først sendes applikationens information til transportlaget, og her lægges TCP-oplysninger ind i nogle felter forrest i pakken. Pakken (eller rettere en pointer til den) sendes videre til IP-laget, som forsyner den med en IP-header, "konvolut", foran.

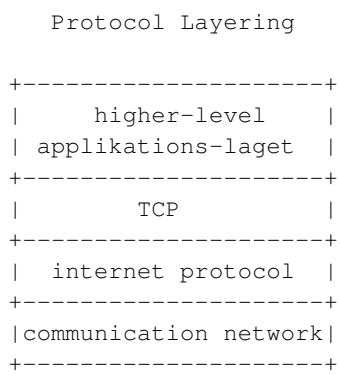
```
<-IP-header----|--TCP-header----|----FTP-header-|-FTP-data----|->
|                                                             |
<-----------------------IP-pakke--------------------------------->
               |                                              |
               <--------------TCP-segment-eller-pakke----------->
                                 |                            |
                                 |                            |
                                 <-------------FTP-pakke--------->
```

Skematisk opstilling af IP-pakke som konvolut for en TCP-pakke

Hver konvolut svarer til et lag i netværksmodellen. Uden om IP-pakken er der på et ethernet en ethernet pakke. Det er en god ide at se på den slags pakker med programmet ethereal. Når ethernet-headeren er selected i mellemste rude, så er de tilsvarende bytes highlighted i hexdump-ruden nederst.



Lagene fra den simple DoD model med kommentarer ses her:



- Applikationslaget, f.eks. FTP og HTTP. Applikationen kan selv lave nogle regler for, hvordan pakkerne skal håndteres og se ud. Der er *mange* forskellige applikationsprotokoller, og det er let at lave nogle selv eller afprøve teknikken ved at lave f.eks. en **finger** klient (TODO: Reference til gawk-finger-server).
- Transportlaget, TCP står for Transport Control Protocol. Her tilføjes TCP-portnummer, TCP-type, tjeksum m.v.
- Netværkslaget, her arbejdes med IP-numre, SNMP, ICMP og router-protokoller; IP, eller Inter Net Protokollen, er reglerne for, hvordan man kommer fra ét lokalnet til et andet.
- Fysisk link-lag, d.v.s. MAC-adresser, ethernetkort og ledninger. Mac-adresser er knyttet til ethernet-pakker. Der er egentlig også et par abstraktionslag her.

Et par forklaringer: Finger protokollen er en af de simplere, morsomme protokoller. Det er nemt at prøve fingerprotokollen: **finger quake@geophys.washington.edu**

SNMP, Simple Network Management Protocol, bruges til fjernkontrol af netværks enheder som f.eks routere. ICMP, Internet Control Message Protocol, er den, som ping programmet benytter, og den, som gør det muligt at sende en besked om, at vores maskine ikke har tænkt sig at svare på for eksempel ftp. MAC-adresser kan listes med **arp -a** programmet og er et tal, som er brændt ind i netkortets ROM, så hardwaren altid kan skelne to netkort fra hinanden.

Disse netværkslag er defineret med henblik på at kunne transportere så meget som muligt så hurtigt som muligt. Derfor behøver IP-laget ikke at interessere sig for, om data er korrekte eller om der lyttes på modtagerens portnummer, og heller ikke for applikationens datakodning og så videre.

Det er derfor meget enkelt og "billigt", målt i tid, at filtrere indkommende pakker på netværkslaget. Bare se efter, om de er til vores maskine, og om de kommer inde- eller udefra.

Hvis firewallen imidlertid er nødt til også at udpakke transportpakken (d.v.s. se i TCP-konvolutten for at få fat i portnummeret) så kræver det selvfølgelig lidt mere tid. Hvis pakken yderligere skal om-adresseres og en ny tjeksum skal beregnes, så bliver det rigtig dyrt. Som tidligere nævnt er det for primitivt kun at filtrere på netværkslaget.

De omtalte netværkslag svarer i kernen til programafsnit, som bearbejder henholdsvis IP-adresse, TCP-oplysninger og, i sidste instans, flytter pakken til den applikation, som skal bruge data.

6.4.1. NAT-ing giver også mere sikkerhed

Som tidligere nævnt skal en NAT-ing router omsætte de interne eller private IP-adresser til den offentlige (lovligt købte) IP-adresse, som man har fået af sin internetudbyder (eng. »Internet Service Provider« eller »ISP«). Man kan for eksempel bruge adresserne 192.168.1.1-255 som lokale adresser, denne nummerserie er reserveret. Det er en del af de såkaldt "private adresser", som ikke bliver sendt rundt på det offentlige net. De er reserverede for at gøre det lettere at etablere et lokalnet uden at skulle søge og betale for "rigtige IP-adresser".

NAT-teknikken er med til at gøre firewallen mere uigennemtrængelig. Det ses, hvis man forestiller sig hvad der sker, hvis man prøver at "snakke" med en maskine med et 192.168.x.y - nummer? Pakkerne kommer ingen vegne. Hvis nogen prøver at trænge igennem en firewall ved at bryde ind i en kommunkations sekvens (trediemandsprincippet, spoofing) så skal pakkerne kunne snyde firewallen *og* modtagemaskinen inden for muren. Det er svært, og det sker heldigvis meget sjældent.

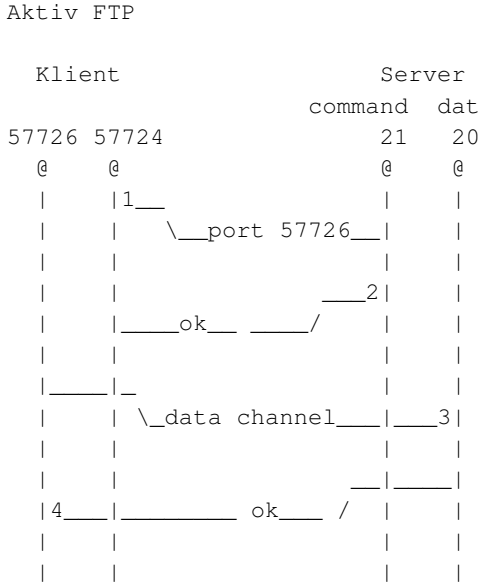
6.4.2. Kontrol med oprettelse af forbindelser

At åbne for alle forbindelser, som startes indefra, kræver, at firewallen holder styr på, hvilke forbindelser der er oprettet; for at gøre det, skal firewallen have inkluderet et modul, der kaldes "connection tracking". Derved kan Linux-kernen styre hvilke forbindelser, der må oprettes imellem de to adskilte netværk. Den kan også logge, hvad der sker på disse forbindelser.

Der er her tale om filtrering på session- eller forbindelses-niveau (se Afsnit 6.1.1.1). FTP er en protokol oven på TCP-laget, og når denne applikation går i gang, opretter den mange "forbindelser". Det giver en ekstra vanskelighed for en firewall. Man skal huske, at FTP protokollen kom til verden i en tid, hvor der ikke var noget, der hed firewalls.

6.4.3. Ftp-problemet

Traditionel FTP, også kaldet aktiv FTP, er lidt et problem i forbindelse med en firewall, da det laver indgående forbindelser. Ikke alene laver det indgående forbindelser, men det er ikke muligt på forhånd at vide hvilken port, forbindelsen laves på. Det vil sige, at for at tillade sine brugere at benytte aktiv FTP, er man nødt til at lade en masse porte stå åbne. Lad os se, hvad der sker, når man bruger traditionel FTP:

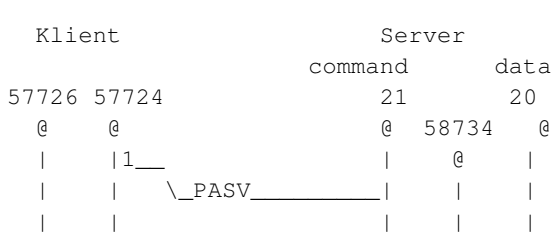


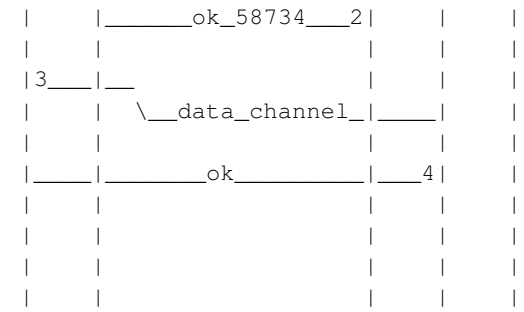
- 1) Først oprettes en forbindelse fra klienten på en tilfældig port (her port 57724) til port 21, FTP-kommandoporten, på serveren. Klienten overfører et portnummer, som den vil benytte til dataoverførslen (her port 57726).
- 2) Serveren siger ok
- 3) Serveren opretter en dataforbindelse fra sin FTP-dataport - port 20 - til den angivne port på klienten (57726).
- 4) Klienten siger ok

Man kan se, at serveren opretter en forbindelse til klienten på port 57726. Da man ikke på forhånd kan vide hvilken port, der vil blive brugt på klienten - FTP-programmet vælger bare en ledig port - er det nødvendigt at lade et portinterval på klienten stå åbent for FTP-forbindelser, så FTP-klienten kan virke.

Problemet kan løses med nyere udgaver af FTP, som kan køre "passiv FTP" i stedet for. Ved passiv FTP er det kun klienten, der starter forbindelser op.

Eksempel 6-1. Passiv FTP





- 1) Først oprettes en forbindelse fra klienten på en tilfældig port (57724) til port 21, ftp-kommandoporten, på serveren. Klienten overfører en kommando, der fortæller serveren, at der skal bruges passiv ftp.
- 2) Serveren siger ok og fortæller hvilket portnummer, den vil have, at klienten connecter til.
- 3) Klienten opretter en dataforbindelse fra sin ønskede dataport - her port 57726 - til den angivne port på serveren (58734).
- 4) Serveren siger ok

Lad os se på, hvad der er vundet. Hvis man vedligeholder en firewall bag hvilken, der er klienter, der gerne vil hente via ftp, er man ved traditionel ftp nødt til at lade et portinterval stå åbent til indgående forbindelser, som de forskellige ftp-servere, folk henter fra, skal oprette. Bruges der derimod passiv ftp, skal man kun tillade udgående forbindelser - man behøver ikke længere lade et portinterval stå åbent til de indgående forbindelser fra diverse ftp-servere.

Ftp-serveren skal til gengæld lade en port stå åben til data, men den bestemmer selv hvilken. Den oplysning sender den med over, før dataforbindelsen oprettes. Alligevel betragtes det nogle steder som et problem på serversiden. Selvom de fleste ftp-programmer - herunder web-browsere - understøtter passiv ftp, er det derfor ikke alle ftp-servere, der understøtter det.

6.5. Linux som firewall

I dette afsnit ser vi på Linux-kernens netfilter modul og hvad man kan gøre med det. Det er en forudsætning, at man kender til begreberne IP-adresse, netværksadresse, netmaske, routing og DNS. Hvis du ikke gør, så læs i det mindste indledningen til denne bog, eller se på http://www.netfilter.org/documentation/index.html#HOWTO (http://www.netfilter.org/documentation/index.html#HOWTO) efter Networking Concepts HOWTO.

De mere avancerede anvendelser af Network Address Translation, NAT, omtales sidst i denne del (Afsnit 6.5.5). Derefter omtales proxy opsætning med squid.

6.5.1. GUI: Anvendelse af grafisk-interface ved installation af firewall

Findes der ikke noget nemmere? hører man ofte en Linux begynder spørge, med rette, for det er i grunden meget få oplysninger der skal til at lave en meget fornuftig firewall, som kan bruges af de fleste. Derfor har distributioner som Red Hat, Fedora, SuSE og Mandrake alle et spørgsmål om man ønsker at maskinen skal være åben eller om der skal lukkes for services. Benytter man en anden distribution (og dem er der jo mange af) kan det hænde, at man selv skal konfigurere firewall.

Det kan også ske, at man selv vil raffinere opsætningen af firewall. Nogen vil derfor gå på jagt efter et værktøj til opsætning og selv installere dette. Her er et par eksempler: **Firestarter** og **Lokkit**.

Figur 6-2. Firestarter giver ikke ligefrem indtryk af at være en sikker firewall, tværtimod står den vildt udseende pingvin med en tændstik her. Men det *er* linux-humor og ikke en trussel!

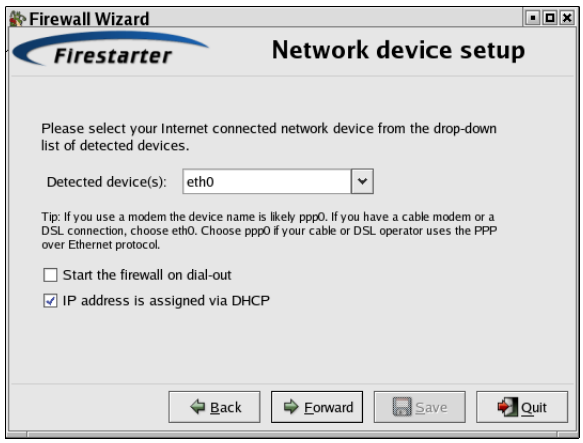
Der mange andre gode front ends til iptables kommandoen. Allerede under installationen kan man på mange distributioner indsætte et firewall filter af medium styrke. Derefter kan man begynde med en GUI som for eksempel Red Hats systemopsætningsværktøj, **setup** eller **/usr/sbin/lokkit**. Disse nemme programmer er front-ends til den bagvedliggende mekanisme og fungerer bravt. De er til stede efter en normal installation. Firestarter kan man selv hente i forskellige grydeklare versioner, så den er også nem at komme igang med.

Hvis man vil noget særligt eller bare vil prøve kommandoerne direkte, så står der mere om det i de efterfølgende afsnit, som beskriver **iptables** anvendelse[2].

Firestarter er en generel front-end, som er meget nem at bruge, fordi den ligesom Red Hats installations program har nogle meget fornuftige default værdier, hvis man bruger troldmand/hjælpe systemet til opsætning af sin første firewall.

Grafikken er rigtig pæn, og let at læse. I første vindue står en pingvin med en tændstik, og sådan noget kan godt gøre mig lidt urolog. Pingvinen ser dog ikke umiddelbart ud til at være i gang med ildspåsættelse!

I andet vindue skal man fortælle, hvilke netkort (d.v.s. interfaces) reglerne skal gælde for. De kan også gælde for modem opkobling med Point-to-Point Protocol (PPP), hvilket også er forklaret. Man lades ikke i stikken!

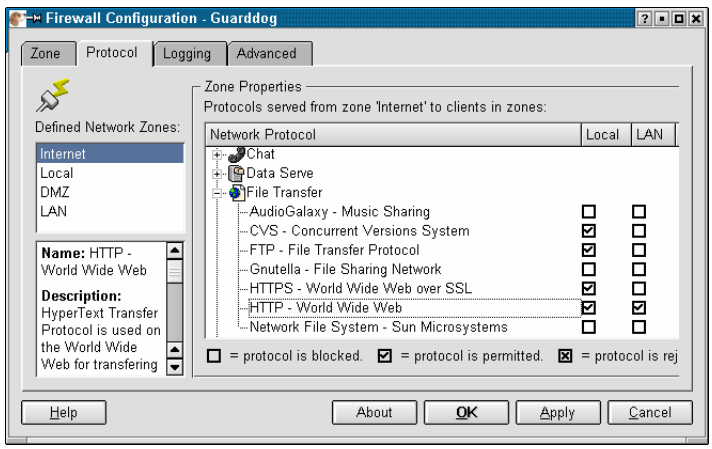


Det er meget fine forklaringer, også på det næste skærmbillede, hvor man rådes til i første omgang at lukke for alle "offentlige" services (disable public access to all services), så intet er tilgængeligt udefra. Så kan man åbne for dem senere, hvis man får brug for dem. Der burde stå, at man ikke bør åbne for telnet, men derimod kan klare de fleste opgaver med ssh.

Figur 6-3. Man kan åbne for de services, man har brug for

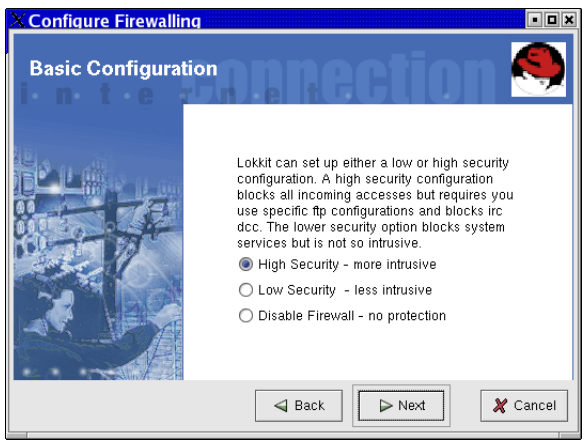
Der er et par skærmbilleder mere, som på en overskuelig måde tilbyder at justere på trafikken af forskellige typer. Denne form for justering vil blive meget mere almindelig i fremtiden, fordi mange typer service kræver en jævn datastrøm, men som udgangspunkt kan man blot acceptere default indstillingerne.

Ønsker man dansk tekst i sin GUI er guarddog (GPL) det klare valg. Gunner Poulsen har oversat til dansk, og man kan hente denne fra http://www.simonzone.com/software/guarddog/#download (http://www.simonzone.com/software/guarddog/#download) Guarddog kan skelne mellem lokalnet og *demilitariseret zone,* som er et område udenfor lokalnettet, hvor man har anbragt sine servere - tilbørligt sikrede, men der skal jo være en åben dør, for at folk kan komme ind og se på éns dejlige web-sider. Man kan selv definere flere zoner og zonetyper.



Læg mærke til, at guarddog styrer protokollerne pr. zone. Det vil sige, at man kan lukke for alt fra for eksempel externe numre og åbne for sine lokale 4 maskiner. Man kan have flere zoner, for eksempel kan en anden afdeling huset få adgang til noget specielt, som de har brug for.

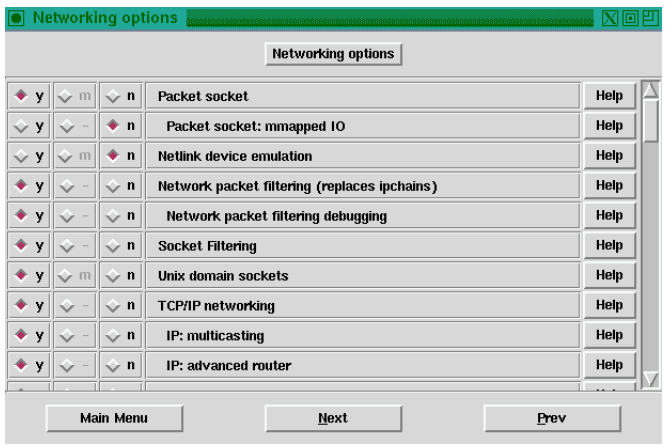
Et andet fremragende GUI firewall-protection program er lokkit, som er bruges ved installation af Red Hat og ved kørsel af setup programmet.



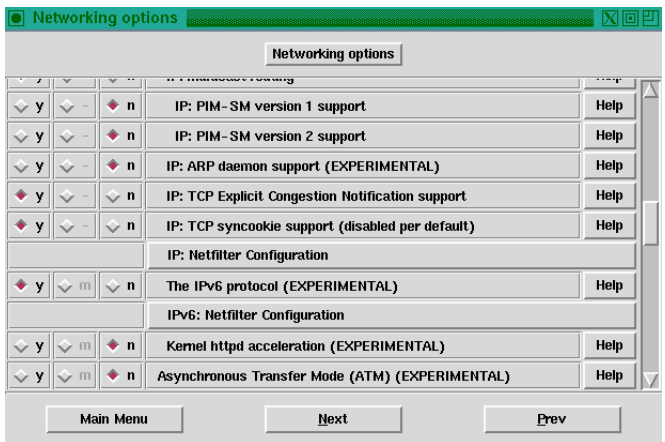
Her skal man blot angive for eksempel maximum sikkerhed og så åbne for den eller de services, som skal være tilgængelige udefra. (De skal jo så være sikret på anden måde, pas især på CGI scripting i Apache hvis den kører som root.)

Hvis du selv vil opdatere din kerne med source fra www.kernel.org (http://www.kernel.org/) så kommer nedenfor nogle screenshots som svarer til **make xconfig** . Her kan du se, hvor i menuerne du skal klikke; husk at læse help teksterne, det er gode forklaringer.

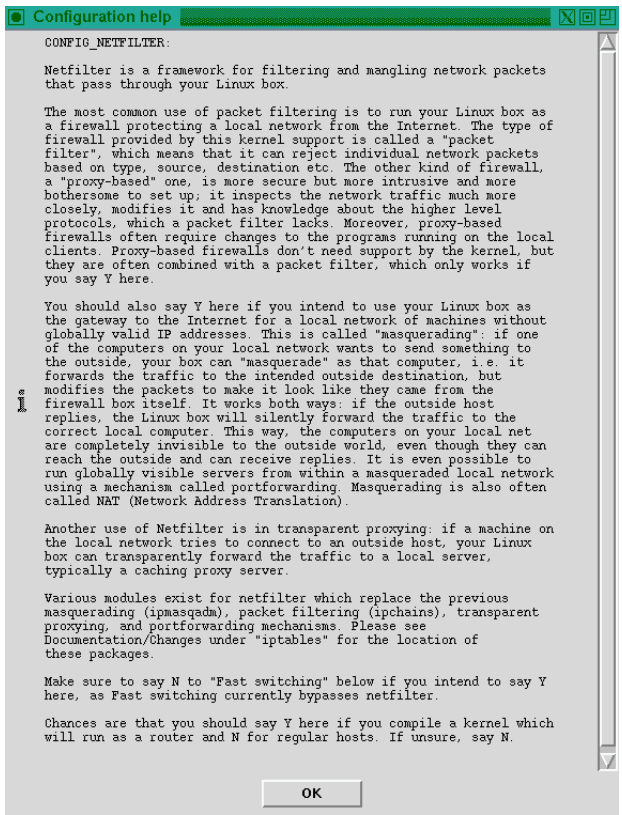
Figur 6-4. make xconfig, vælg network packet filtering



Figur 6-5. Længere nede vælges sub-menu for netfilter configuration



Figur 6-6. Den engelske hjælpetekst er meget fyldestgørende, se appendix Afsnit 6.8



iptables kommandoen skal bygges, så den passer til den kerne, man bruger, eller rettere, så den passer til den version af netfilter, som er i kernen. Også det er gjort i alle distributioner.

iptables indsætter som nævnt regler i kernens netfilter tabeller, hvilket betyder, at reglerne går tabt ved reboot. Du kan prøve at bruge kommandoerne iptables-save og iptables-restore.

Hvis man har en opsætning, som man gerne vil gemme, så kan det lade sig gøre uanset om det er en "håndlavet opsætning" eller om den kommer fra et af de nævnte GUI-tools. Man bruger simpelthen iptables-save.

Resultat-filen gemmer man, fx. i /etc/init.d eller /etc/rc.d og så sætter man en overordnet kommando ind i et start/stop script, som køres af rc kommandoen ved opstart. Det er måske lidt for vanskeligt for en GUI bruger, men med lidt hjælp udefra skal det nok kunne lade sig gøre.

Man skal i så fald huske at få scriptet til at foretage sig noget fornuftigt, hvis en af iptables

kommandoerne fejler; normalt vil det fornuftigste være at starte **/sbin/sulogin** for så kan system-administrator se, at der er noget galt, og systemet går ikke videre (på nettet) uden at denne fejl er løst. For mere avancerede anvendelser, hvor ubemandet opstart er en nødvendighed, må man finde en anden måde at systemet kommer videre på, det kunne fx. være at tilbyde login til en overvågningsmaskine på lokalnettet.

6.5.2. Iptables og netfilter i kerne 2.4 og 2.6

I Linux er pakkefiltrering i en avanceret version indbygget i kernen; kernedelen kaldes netfilter, og programmet **iptables** kommunikerer vores ønsker om hvad kernen skal foretage sig fra user-space til kerne-space.

For at kunne bruge kernens netfilter og **iptables** kommandoen, skal kernen være bygget med netfilter, og, hvis man planlægger at sikre en maskine, må det stærkt anbefales at man også inkluderer connection tracking. Det koster lidt belastning, men giver mange fordele. De fleste standard kerner fra distributioner med kerne 2.4.18 og frem har den slags. Det kan kontrolleres med **ls -l /proc/net/ip_conntrack**

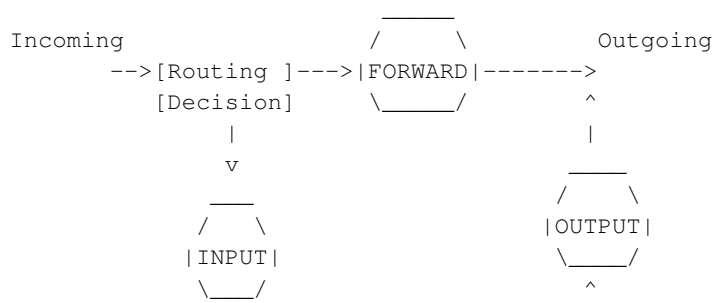
6.5.2.1. Hvordan pakker rejser igennem filtrene

Navnet iptables hentyder til, at kernen har tre tabeller:

- Filter tabellen, som bruges, hvis man udelukkende skal stoppe uvedkommende trafik.
- NAT-tabellen, som bruges, hvis man skal foretage mapning / spejling af interne adresser, obligatorisk, hvis der er tale om private adresser som for eksempel 192.168.x.x.
- Mangle-tabellen, som bruges, hvis man vil pille ved bits i pakkerne for at kunne klare vanskelige situationer.

Kernen begynder med at have 3 lister med regler i "filter" tabellen. Disse lister kaldes firewall kæder eller blot kæder, chains. De tre kæder kaldes INPUT, OUTPUT og FORWARD (pakker-ind, pakker-ud og pakker-til-videresendelse).

Her er en ASCII-kunstnerisk gengivelse af kædernes organisation (med tak til Rusty Russell's packet-filtering-HOWTO). Det er væsentligt forskelligt fra arrangementet i kerne 2.0 og 2.2.



```
                |                         |
                 ----> Local Process ----
```

Det ses, at en pakke kan komme fra to principielt forskellige kilder. Enten kommer pakkerne fra et netkort (rettere interface - men prøv at oversætte det til dansk) eller også kommer de fra et program, en lokal proces.

De tre "cirkler" repræsenterer de tre kæder nævnt ovenfor. Når en pakke ankommer til en cirkel i diagrammet, undersøges reglerne i denne kæde for at afgøre pakkens skæbne. Hvis kæden siger DROP pakken bliver den elimineret på stedet, men hvis kæden siger ACCEPT vil den fortsætte sin rejse gennem diagrammet.

En kæde er en tjekliste med regler. Hver regel siger "hvis pakke-headeren ser ud på denne-hersens-måde, så se her hvad der skal gøres med pakken". Hvis en regels pakke-beskrivelse passer på pakken, kalder man det et match. Det svarer selvfølgelig til en masse betingelser, som er opfyldt. Hvis pakken ikke matcher reglen fortsætter man til næste regel. Endelig når vi til et punkt, hvor der ikke er flere regler at undersøge. Nu ser kernen på kæde-policy for at afgøre, hvad der skal ske. I et sikkerheds-bevidst system siger denne policy som regel at kernen skal DROPpe pakken.

1. Når en pakke kommer ind, lad os sige fra ethernet-kortet ser kernen på pakkens destination: Dette kaldes routing (rutning eller bedre, ruting?)

2. Hvis den er til denne box bliver pakken sendt videre ned i diagrammet til INPUT kæden. Hvis den passerer denne, så vil alle processer, som venter på denne pakke, få den.

3. I modsat fald, hvis kernen ikke har videresendelse aktiveret, eller den ikke ved, hvordan pakken skal sendes videre, vil pakken blive droppet. Hvis videresendelse er aktiveret, og pakken er tiltænkt til et andet netværks-interface (hvis du har sådan et), så vil pakken gå fremad til FORWARD kæden. Hvis den accepteres her, bliver den sendt ud.

4. Sluttelig kan et program, som kører på boxen, sende netværkspakker. Disse passerer gennem OUTPUT kæden straks: Hvis den siger ACCEPT, så fortsætter pakken til hvad nu end det måtte være for et interface, som den er bestemt for.

6.5.3. Anvendelse af iptables kommandoen

iptables kommandoen har en ret detailleret manual side (**man iptables**) hvis du har brug for oplysninger om specielle ting.

Der er mange ting, man kan gøre med **iptables** kommandoen. Man begynder med at have de tre indbyggede kæder, INPUT, FORWARD og OUTPUT, som man ikke kan slette. Lad os først se på

hvordan man administrerer hele kæder.

- **-N** : Opretter en ny kæde.
- **-X** : Sletter en tom kæde.
- **-P** : Ændrer policy for en kæde.
- **-L** : Lister regler i en kæde.
- **-F** : Flusher, sletter reglerne i en kæde.
- **-Z** : Nulstiller tællerne for en kæde.

Brugerdefinerede kæder bruges til at strukturere filtreringen og demonstreres senere. Her er et simpelt eksempel på en kæde-kommando, som nulstiller tællerne i INPUT kæden. Prøv at liste før og efter med **iptables -L INPUT -v** :

```
[root@hven /]# iptables -Z INPUT
```

Der er fire kommandoer til at manipulere reglerne i en kæde:

- **-A <kæde>** Tilføj (append) regel
- **-D <kæde> [nummer]** Slet (delete) regel
- **-R <kæde> [nummer]** Erstat (replace) regel
- **-I <kæde> [nummer]** Indsæt (insert) regel

En regel er en beskrivelse plus en handling.

Beskrivelsen er et match-udtryk og kan være sammensat af mange del-udtryk: *Hvis pakken kommer fra adresse 44.55.66.77 og skal videre gennem interface eth1 ...* Det ville man skrive sådan: **--source 44.55.66.77 --out-interface eth1** .

En handling er et "jump", og det svarer til at tage en bestemmelse om, hvad der skal ske med pakken. Det, man hopper til, kaldes et mål eller target, og to af de indbyggede targes er de samme som de policy, vi tidligere har set, DROP, ACCEPT. At tage beslutning om at droppe en pakke skrives **-j DROP**

Man kan sætte flere match udtryk sammen. Nedenfor ses efter pakker, som kommer fra host 44.55.66.77 og som skal ud gennem interface eth1. Alle pakker, som har de to egenskaber, droppes:

```
[root@hven /]# iptables -A OUTPUT --source 44.55.66.77 --out-interface eth1 -j DROP
```

Endvidere kan man hoppe til en anden, brugeroprettet kæde, eller man kan returnere fra en kæde (med **-j RETURN**). Der er et REJECT target, som dog kræver at kernen er kompileret med support for det. Der er andre inbyggede targets, som kræver tilvalg i kernen: QUEUE beder kernen om at sende pakkerne til en pseudo-device, som et program i userspace kan lytte på, og så er der bl.a. også REDIRECT, SNAT, DNAT og LOG targets. (Lad være med at studere listen i detailler! Bladr frem.)

1. ACCEPT

   Pakken sendes videre uden yderligere test og dermed uden yderligere tidsforbrug.

2. DROP

   Pakken ignoreres ("smides væk").

3. RETURN

   Der hoppes tilbage til den kæde, man kom fra eller for indbyggede kæder til default policy.

4. REJECT

   Pakken droppes men der sendes en ICMP pakke til modtageren med besked.

5. QUEUE

   Pakken lægges i kø til pseudodevice der tillader data at blive hentet til userspace.

6. REDIRECT

   En -t nat funktion. Modtageradressen ændres, så pakken sendes til maskinen selv (lokalhost); for tcp/udp kan portnummer ændres. Bruges til proxy-funktionalitet.

7. SNAT

   Kun for nat-tabellen, POSTROUTING kæden, specificerer at afsenderadresse skal ændres for denne og alle fremtidige pakker i denne forbindelse.

8. DNAT

   Modtager adressen ændres for denne og følgende pakker i samme forbindelse.

9. LOG

   Begynd kernel-logging for matchende pakker. Reglen behøver ikke at have noget target. Netfilter fortsætter med næste regel. Bruges til at analysere.

10. MARK

    Bruges i mangle tabellen i forbindelse med iproute2.

11. MASQUERADE

    Bruges til NAT-ing med dynamiske adresser.

12. TCPMSS

    En speciel funktionalitet for fragmentering af ICMP pakker, som ens ISP ellers kasserer.

13. TOS

    For mangle tabellen, ændring af Type Of Service (TOS) feltet i TCP headeren.

14. ULOG

    Pakken multicastes til userspace for logging/analyse.

15. DSCP

    Nyere version TOS mangling.

16. ECN

    For løsning af problem med Explicit Congestion Notification.

17. MIRROR

    Kun for forsøg, bytter om på afsender/modtager adresserne.

Listen er ordnet efter hvad jeg synes er vigtigst. Se manual page for nærmere detailler om targets. Lige nu er de vigtigste ACCEPT, DROP og SNAT, som kun kan bruges i nat-tabellens POSTROUTING chain.

Opsummerende kan vi sige, at en regel består af består af en eller flere kommandoparametre, som beskriver en egenskab ved en netværkspakke, samt (som regel) en besked om, hvad der skal ske med pakker, som beskrivelsen passer på. Hvis en beskrivelse passer, siger man at pakken matcher.

De vigtigste kommandoparametre til match er

- **--protocol [icmp|tcp|udp|all]**

  Man kan også bruge et navn fra `/etc/protocols` . Hvis man ønsker at matche alle pakker, som ikke tilhører en bestemt protokol, negeres med udråbstegn. Fx. **iptables -A INPUT --protocol ! tcp -j ACCEPT** accepterer alt undtagen TCP-pakker. Kan forkortes til -p.

- **--source [!] adresse[/netmaske]**

  Angiver oprindelsesadressen. Kan forkortes til -s. Eksempel: **iptables -A INPUT --source 123.45.67.89 -j ACCEPT** vil bevirke at vi accepterer alt, hvad der kommer fra 123.45.67.89 - og formentlig dropper alt andet.

- **--destination [!] address[/mask]**

  matcher på slutadressen, destinationen. Forkortes -d.

- **-i, --in-interface [!] name**

  hvor *name* kan være eth0, eth1, ppp0 m.v. kan bruges til at selektere alle pakker, som kommer fra ydersiden eller indersiden af firewallen.

- **-o, --out-interface [!] name**

  som -i, bruges til at selektere på alle pakker, der skal sendes ud via et bestemt interface.

- **-m state**

  bruges til at fortælle kernen at tilstandsmodulet (connection tracking) skal bruges. Derefter kan man specificere, at pakker, som er ESTABLISHED eller RELATED til en forbindelse, som er i gang, skal have lov at passere. Det er den måde, man åbner for al trafik indefra.

Næsten alle options har en kort form, for eksempel -s i stedet for --source.

Adresser til --source og --destination kan specificeres på fire måder:

- Som domaine-navn, www.kernel.org, linuxbog.dk, www.netfilter.org
- Som IP-adresse: 127.0.0.1, 193.88.44.22
- Som netværksadresse med skråstreg + antal signifikante bits: 10.11.128.0/17
- Netværksadresse med skråstreg+bitmaske, 10.11.128.0/255.255.128.0

Man kan selektere alle pakker, som *ikke* kommer fra en adresse ved at negere adresse udtrykket, hertil bruges et udråbstegn, som skal have space før og efter: **-s ! localhost** vil matche alle pakker, som ikke kommer fra localhost.

6.5.3.1. Nogle praktiske eksempler

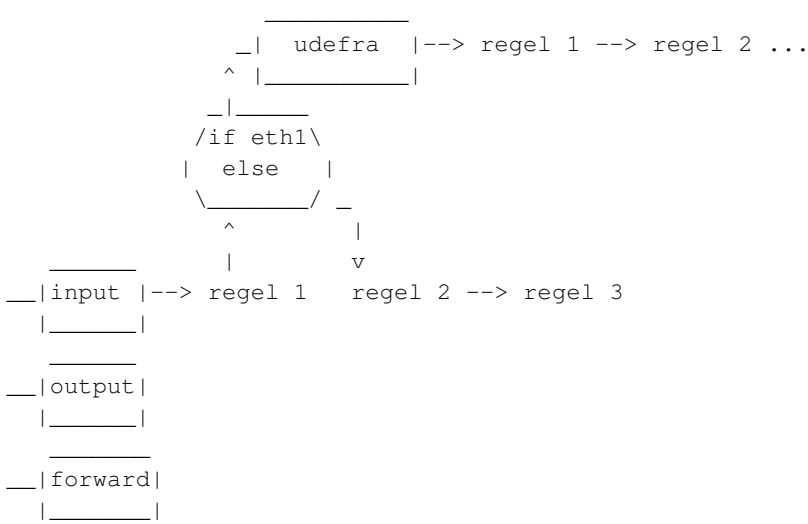
Nu til hands on! Vi antager, at vi har fem maskiner i huset og et en ADSL forbindelse på en af maskinerne. ADSL er på et separat netkort, som har betegnelsen eth1. De andre maskiner er forbundet via eth0 og vil gerne have lov at få del i ADSL forbindelsen.

Lad os lave en kæde af regler, som skal inspicere al trafik, der kommer ind på vores box, men kun, hvis det er fra ADSL interfacet, altså eth1.

```
[root@hven /]# iptables -N udefra
```

Man kan liste den nye kæde sammen med de andre: **iptables -L** Kæden er ikke taget i brug, så der står "0 references". For at bruge den til noget må vi instruere kernen yderligere med en jump kommando. Vi laver en regel 1 for INPUT kæden.

```
[root@hven /]# iptables -A INPUT -i eth1 -j udefra
```



Det svarer til en if-sætning. Vi tester for om det var fra eth1 og hvis det er det, så fortsæt med regelkæden *udefra*.

```
if (pakke_source == device_eth1) {
    tjek_reglerne_i_kaede_eth1();
}
else {
    fortsaet_med rule2();
}
```

Lad os inspicere pakkerne udefra for, om de er til vores http server. Hvis de er det, så slip dem igennem, ellers skal de droppes.

```
[root@hven /]# iptables -A udefra -p tcp --dport 80 -j ACCEPT
```

Når man specificerer -p tcp (eller --protocol tcp) så loader iptables et extension modul, og vi kan nu kigge ind i tcp-headeren, hvis vi vil - og det vil vi, vi undersøger, om pakkerne er bestemt for port 80, den velkendte port, som benyttes af http serveren. Alle pakker til serveren accepteres. Der er forskellige protocol extensions, som tillader avancerede ting: Læs manual-page hvis du vil vide mere.

```
[root@hven /]# iptables -A udefra -j DROP
```

Derefter appender vi med **-A** endnu en regel, som siger, at alt andet skal droppes. Svarpakker til brugerne inde på lokalnettet vil imidlertid også blive stoppet af denne stop-regel, så nu vil vi benytte connection tracking og indsætte en regel, som siger, at alle pakker, som er beslægtede med en forbindelse, som allerede er oprettet, skal accepteres.

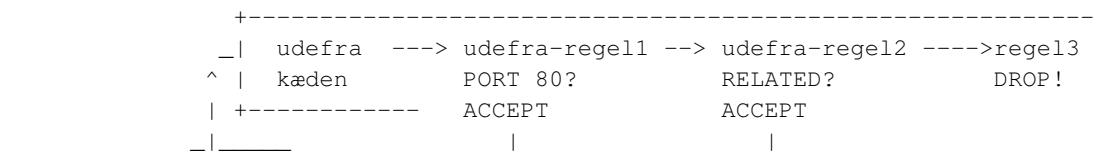
```
[root@hven /]# iptables -I udefra -m state --state ESTABLISHED,RELATED -j ACCEPT
```

Denne regel bliver nu nummer 1, og de andre rykker ned som 2 og tre. Hvis vi hellere vil sætte denne regel ind som nummer 2, skal kommandoen bruge **-I udefra 2** ; det er en meget bedre løsning, for connection tracking koster trods alt mere tid, så man bør acceptere så mange pakker som muligt med simple regler inden man bruger connection-regler.

```
[root@hven /]# iptables -I udefra 2 -m state --state ESTABLISHED,RELATED -j ACCEPT
```

-m state fortæller kernen, at den skal benytte tilstandsmodulet, ip_conntrack. Derefter er det nødvendigt at bruge --state parameteren til at fortælle, hvilke tilstande, der skal accepteres. Ud over ESTABLISHED og RELATED er der bl.a. NEW og INVALID, som betyder at en pakke ikke tilhører nogen kendt forbindelse.

Læg mærke til, at der ikke er mellemrum efter kommaet mellem de to tilstande, det må der ikke være.



```
 /if eth1\              |                 |
|  else   |             |                 |
 \_______/------------> +-----------------+----------> Sendes videre
   |
   |
INPUT-chain

Lidt forsimplet af pladshensyn, man skal spørge både på
ESTABLISHED og RELATED i regel 1 i  "udefra"  regel kæden.
```

Lad os nu antage, at vi slet ikke kan lide telnet protokollen, heller ikke internt på det lille net, og derfor vil blokere for al telnet. Nu er telnet jo så en applikationsprotokol, så vi kan ikke bruge **--protocol** optionen, idet den kun kender protokollerne tcp, udp, icmp eller "all", eller også et nummer fra filen `/etc/protocols` . Med andre ord, telnet står ikke i `/etc/protocols` og derfor må vi "sigte efter telnet" på en anden måde, for eksempel ved at stoppe al trafik, som har destination port 23.

```
[root@hven /]# iptables -I INPUT -p tcp --dport 23 -j DROP
```

Der benyttes -I, insert, fordi reglen skal komme før eth1-reglerne, hvis det skal nytte noget. Der står jo en accept inde i eth1-reglerne, som kunne forstyrre billedet (lidt). Nyere versioner af iptables vil kunne matche generelt på --ports 23, så det både er afsender/modtager pakker, som stoppes.

Eksemplet er ikke fuldkomment endnu, hvis maskinerne på lokalnettet har "private IP-adresser" - så næste opgave går ud på at opsætte NAT for lokalmaskiner.

6.5.4. Opsætning af NAT

NAT og Masquerading sker i nat-tabellens FORWARD kæde, når en datapakke skal til at forlade maskinen. Det kan kun ske her. Source adressen, som skal NATtes, er den interne eller private adresse, men man kan også angive netværket ved at skrive 0 for den sidste byte i en almindelig 192.168 - adresse og også angive netmaske, typisk 192.168.1.0/24, så vil alle de lokale adresser blive NAT-ede.

iptables har mange måder at angive adresser på, navne, numre, netværk, og for NAT også intervaller m.v., så hvis der opstår specielle behov, så kig lige en gang ekstra på manual page for iptables. Det er bedre at nøjes med én regel, som dækker flere adresser, end at have flere regler med hver deres adresse.

```
[root@hven /]# iptables -t nat -A POSTROUTING --src 192.168.1.0/24  -o eth1 -j SNAT --to x.y.w.z
```

Hvis man får en dynamisk IP adresse fra sin internet leverandør, skal kommandoen se lidt anderledes ud, og så bruger man betegnelsen *masquerading* om metoden. Dynamiske adresser benyttes typisk med modemforbindelser, eller når man har en Point-to-Point Protocol (PPP) forbindelse til sin ISP. Man får en ny adresse hver gang. Linux kan tage denne adresse og bruge som den officielle adresse for NAT-ing af interne maskiner. Masquerading er oftest dynamisk NAT, med kun en ekstern adresse.

MASQUERADE target er kun muligt i *nat* tabellen, og kun i POSTROUTING kæden. Dette target bør kun benyttes med dynamiske IP-adresser.

```
# Eksempel på anvendelse af MASQUERADE target:

        # Masquerade out ppp0
        iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

        # Disallow NEW and INVALID incoming or forwarded packets from ppp0.
        iptables -A INPUT -i ppp0 -m state --state NEW,INVALID -j DROP
        iptables -A FORWARD -i ppp0 -m state --state NEW,INVALID -j DROP

        # Turn on IP forwarding
        echo 1 > /proc/sys/net/ipv4/ip_forward
```

Man kan godt bestemme, hvilken port-range, som må bruges til *nat* tabellen. Hvorfor skulle man gøre det? I nogle tilfælde laver man huller i firewallen for at sikre folk adgang til en bestemt protokol. Det er klart, at dette er en sikkerhedsfaktor, men hvis man skal have glæde af en server, så må der jo åbnes. Det kan evt. aftales med kendte brugere, at man kun holder et mindre port-interval åbent for bestemte protokoller.

DNS forespørgsler kan bruges som eksempel. Vi vil sætte en intern DNS server op, en hjælpeserver, som husker nyligt brugte adresser, så alle maskinerne på vores lille netværk ikke behøver at spørge igen og igen ud over det overbelastede ISP link. Alle på det interne net kan spørge denne maskine, men den skal være fuldstændig lukket udadtil, bortset fra, at den skal kunne spørge en remote DNS server. Det kaldes en DNS caching server. Til DNS benyttes User Datagram Protokol, UDP, som let kan udsættes for spoofing. Det ønsker vi at gøre alt for at forhindre. Det forbedrer selvfølgelig sikkerheden at køre en DNS caching server på en maskine for sig selv.

Når hjælpeserveren skal foretage en forespørgsel, bruger den som afsenderport et nummer som den får tildelt lidt tilfældigt, men det vil ligge i intervallet 1024 - 65535. Det kaldes de ikke-priviligerede porte. Netfilter skelner mellem 1-511, som er såkaldt privilegerede, velkendte portnumre, som kun kræver root privilegier, og som typisk bruges af specielle serverprogrammer som for eksempel httpd, 512-1023, som også er priviligerede, og så 1024 - 65535, der kan anvendes til alt. Netfilter mapper ikke portnumre fra en gruppe til en anden. (At mappe = at spejle eller sige a svarer til X, b svarer til Y, c svarer til Z og så fremdeles, en af de få ting, en computer er god til.) Man vil kunne begrænse DNS-klienten til at bruge ganske få portnumre for forespørgsler, og derved yderligere skærpe sikkerheden, men det er ikke netfilters område.

De eksterne DNS servere sender svar til det portnummer, som klienten brugte til at spørge.

Her kommer kommandoen, som tillader alle remote DNS servere at svare til vores maskines eksterne netkort på en ikke-priviligeret port:

```
[root@hven /]#  iptables -A INPUT -p udp -s 0/0 --source-port 53 -d 66.14.136.144/32 --destination-po
```

Det ville være væsentligt bedre, hvis vi kun brugte en enkelt server udenfor firewallen, men så begynder det at blive et stort setup. Med den foreslåede lokale caching server kan anvendelsen af udp på alle lokale maskiner begrænses til, at de kun må lave UDP til denne specifikke IP-adresse. Her kommer kommandoerne, som kan gøre det, idet vi antager, at caching serveren har lokalnummeret 192.168.1.2:

```
[root@hven /]#  iptables -A INPUT -p udp -s 192.168.0.0/16 --destination-port 53 -d 192.168.136.144 -
[root@hven /]#  iptables -A INPUT -p udp -s 192.168.1.2/16 --source-port 53 -d 192.168.0.0/16 1024:65
[root@hven /]#  iptables -A INPUT -p udp -j DROP
```

Som man kan se er det lidt mere besværligt at kontrollere UDP, fordi man ikke kan regne med connections - alle pakker er som en enkeltstående begivenhed. Det er en statusløs protokol. Rækkefølgen er kritisk. Det ville være en god idé at bruge -i for interface specifikation, så firewallen styrer på IP nummer og hardware.

Ligesom for tcp protokollen loades der et extension modul, når vi specificerer -p udp. De ekstra kriterier, som dette modul giver mulighed for, er dog kun portnumre for hhv. source og destination. Det ovenstående eksempel specificerer, at alle inputporte over 1023 må bruges. Det ville være meget bedre at sige til DNS-forwarder programmet at det kun må bruge for eksempel port 32109. Ved at bruge fast portnummer for hjælpeserverens udgående trafik kan antallet af åbne porte begrænses meget.

```
[root@hven /]#  iptables -A INPUT -p tcp ! --syn --source-port 20 --destination-port 1024:65535 -j AC
```

Her vises en måde at åbne for alle FTP-pakker udefra, som ikke forsøger at indlede en konversation. Optionen --syn betyder, at pakker, som ikkehar syn-flaget (synkroniserings-request) sat, skal have lov at passere. Det er en anden måde at specificere, at maskiner indenfor firewallen skal have lov til at gøre alt, men alligevel beskyttes de såkaldt priviligerede porte i intervallet 1 - 1024. En angriber kan godt sende en pakke, som ikke har syn flaget sat, men det er jo sværere at komme igennem til noget sårbart, når man nu ikke kan sende til portnumre under 1024.

En måske mere avanceret anvendelse af port-ranges er at åbne for et mindre område ved source NAT-ing; man reserverer et interval til et bestemt formål og kan på remote maskinen filtrere alle andre afsenderporte fra. Mange netværksprogrammer har en mulighed for at brugeren specificerer, hvilke porte, der må bruges.

```
[root@hven /]#  iptables -t nat -A
POSTROUTING -j SNAT --to 129.142.141.140:47000-49000
```

Et eksempel på anvendelse af multiport extension, hvor der åbnes for (kendte) serverporte i et hug (mere effektivt, mindre load på firewall):

```
[root@hven /]#  iptables -A INPUT -p tcp -m multiport --dport 21,25,80,110,143,443 -j ACCEPT
```

I det efterfølgende eksempel bliver port specifikationer brugt til at omdirigere trafikken, en stærk feature, når man vil lave en HTML proxy.

6.5.5. Source/Destination NAT og omdirigering

I det følgende ses på henholdsvis source NAT og destination NAT. I source NAT er det afsenderens adresse+port, som oversættes til gateway-adresse + en anden port. I Destination NAT sendes pakker, som egentlig er til a.b.c.d i stedet til x.y.z.q, hvilket jo strengt taget blot er en omdirigering.

Destination NAT rummer imidlertid mange muligheder, og er nødvendigt, når for eksempel 1000 klienter i sekundet spørger på Google, der i snit måske er 1/100 sekund om at svare. Man kan hurtigt regne ud, at de har mange servere kørende!

6.5.5.1. Source NAT

Source NAT er det almindeligste, og er det, som vi hidtil har set på i forbindelse med i192.168-numre klienter bag en firewall. Man kan sige, at vi spejler, (mapper, afbilder) source adressen plus portnummer som en global (offentlig) IP adresse med et andet portnummer. Derved kan mange lokale maskiner deles om et enkelt IP nummer. Teoretisk set kan man løbe tør for portnumre på NAT maskinen, og det er der skam også taget højde for, idet maskinen simpelthen dropper alle pakker som ikke kan mappes til dens range af IP-numre+portnumre.

6.5.5.2. Destination NAT

Man kan også mappe (afbilde) destinationsadressen - Destination NAT - så folks forespørgsler f.eks. bliver omdirigeret hen til den lokale webserver eller til en server ud af en server pool.

For TCP og UDP kan man lave portmapping med iptables, hvor man skifter destinationsporten ud. F.eks. kan en klient oprette en forbindelse til en web-server på port 80, og NAT-serveren kan afbilde pakkerne til sin egen port 3128, hvor der kører en squid. Dette kaldes redirect, men er også en form for destination NAT. Resultatet er transparent proxying.

Lav destination NAT til ip 1.2.3.4, 1.2.3.5 eller 1.2.3.6

```
# iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 1.2.3.4-1.2.3.6
```

Fordelingen til serverne nr. 4 - 6 sker efter tur. Det kaldes ofte round-robin princippet. Derved kan man få fordelt belastningen til flere servere. Det egner sig ikke til CGI eller .php løsninger, hvor man skal fastholde en klient, hertil kræves en router, som kan holde styr på connections og samtidig lave round robin.

Lav destinations NAT fra port 80 til den lokale port 3128, hvor squid kører

```
# iptables -t nat -A PREROUTING -i eth1 -p tcp --dport 80 \
-j REDIRECT --to-port 3128
```

Eksemplerne er taget mere eller mindre direkte fra Linux 2.4 NAT HOWTO, og er blot ment som illustration af, hvad man kan med iptables NAT.

For nærmere beskrivelse, se Linux 2.4 NAT HOWTO på
http://www.netfilter.org/unreliable-guides/NAT-HOWTO/index.html
(http://www.netfilter.org/unreliable-guides/NAT-HOWTO/index.html) Se desuden man-siden for iptables.

6.5.6. En komplet firewall

Selv om der allerede er givet eksempler på en firewall Afsnit 6.1.3.1 så vises her en lidt større opsætning af en firewall. Eksemplet er valgt ud fra det kriterium, at det skal være en opsætning, som har fungeret og har vist sig at være stabil.

Setuppet er, at vi har en maskine, som har forbindelse til vores ISP med fast (eller næsten fast) adresse, 130.3.2.99. Den skal være gateway og firewall. Desuden står der et antal maskiner i vores lille computerworkshop, og de må gerne snakke med internettet gennem vores gateway/firewall, men da de ikke har "private adresser" må vi også sørge for, at firewallen laver NAT. Desuden vil vi gerne have, at firewallen er web-server. Dette er ikke realistisk for en virksomhed, som ønsker high-speed, ultimativ sikkerhed, her ville man sørge for at fordele belastningen på flere maskiner, men realistisk nok for at demonstrere mange pointer.

Firewall maskinen har 2 netkort, eth0 og eth1. På eth0 er den forbundet til en hub på det interne net, og på eth1 er der hul til ISP'ens gateway, der har nummer 130.3.2.1, en såkaldt klasse B adresse, som har netmaske 255.255.255.0 så den fungerer som en klasse C adresse, med max 255 maskiner i netværket.

Det interne net har adresser i området 192.168.67.0 - 255. Det er for at være lidt på tværs, at jeg altid anbefaler andet end 192.168.1.1-254 om ikke andet, så kan de værste root-kit til intrusion på lokalnet herved vanskeligere få adgang ved spoofing.

Vi starter med at konstatere, at man ikke kan lave en default policy som kun gælder for det ene netkort. Derfor adskilles trafikken, således at alt input fra eth1 ledes igennem en kæde, som kaldes door1.

Door1 skal ende med at droppe al trafik, som vi ikke ønsker ind i vores netværk. De ting, som vi gerne vil have med, skal accepteres med en kommando -j ACCEPT.

Det aller første, scriptet gør, er at sikre, at netfilter delen i kernen er i en kendt tilstand, d.v.s. vi flusher alle regler i alle kæder og sætter alle default policies. Hvis man prøver at optimere ved kun at foretage de nødvendige ting for ens opsætning efter et boot, så skal man lige sørge for at have en version liggende, som kan klare *alle* situationer.

Det næste, der sker, er at kæden door1 slettes. Det sker selvfølgelig for at sikre, at der ikke ligger rester af andre regler i door1.

Det kan gøres på mange mange andre måder, men denne her metode er valgt efter lidt eksperimenteren. Man får også nulstillet tællerne for door1, og fejlene er ikke nogen, som får systemet til at hænge, hvilket vistnok har kunnet forekomme med nogle af de iptables options, som giver fejl, hvis man kører dem selv om de ikke er nødvendige. Læseren kan eksperimentere med at flushe kæden door1 i stedet for (også hvis den ikke findes).

Næste skridt er så at oprette kæden door1. Nu er den tom - ingen regler, ingen forbindelse med andre kæder. Lister man med -L kommandoen, får man at vide "no references", svarende til død kode eller ubrugt variabel.

Nu installeres regel efter regel i firewallen. De vigtigste er stadig dem, som tillader brugere indefra at kommunikere frit, men man skal lige huske, at denne trafik altid starter med input fra eth0, som vi slet ikke ser i denne door1 kæde. Indtil videre er der fri passage indefra og ud. Det er først til sidst, at der etableres en regel som siger, at al indadgående trafik fra eth1 skal via door1. Det kunne lige så godt have været først, og under alle omstændigheder må man kun aktivere eth1, når alle reglerne i firewallen er opsat. Man kan godt opsætte regler for et interface, som er "down", **ifconfig eth1 down** .

Der åbnes for al trafik til ssh-porten, både den indefra-ud og udefra-ind. Ligeledes åbnes der for al trafik til http porten.

Næste regel åbner for indkommende trafik, der stammer fra port 53, nameservere.

Nu kommer en regel, som kun skal benyttes, hvis ISP benytter DHCP til kontrol af IP, nameservices og routing. Det er en god ting og forhindrer slet ikke, at man selv styrer opsætningen.

Der åbnes også for ping. Det er ikke et krav for en firewall, men er ok at gøre på en maskine, der står udenfor en firewall men inden en gateway. I eksemplet her er gateway og firewall den samme, men det er ok at åbne for ping/ICMP, hvis man holder øje med trafikken.

Endelig kommer de spændende linier, hvor der åbnes for indgang for alt, hvad der er svar på forbindelser etableret indefra.

Dernæst er der kun opsætning af NAT-ing og aktivering af forwarding.

```
#/bin/ksh

iptables -t filter -F
iptables -t nat    -F
iptables -t mangle -F
iptables -X door1 2> /dev/null 1>&2
```

```
iptables -N door1

iptables -A door1 -m state --state INVALID -j DROP
# åbne for al ssh og http
iptables -A door1 -p tcp --sport ssh  -j ACCEPT
iptables -A door1 -p tcp --dport ssh  -j ACCEPT
iptables -A door1 -p tcp --sport http -j ACCEPT
iptables -A door1 -p tcp --dport http -j ACCEPT
# åbne for svar fra nameserere, DNS.
iptables -A door1 -p udp --sport 53   -j ACCEPT
# der åbnes for konversation med dhcp serveren på gatewayen.
iptables -A door1 -p udp --source 130.3.2.1 --dport bootps -j ACCEPT
# åbne for ping (mange vil foretrække at lukke i stedet!)
iptables -A door1 -p icmp             -j ACCEPT
# Alt, hvad der er etableret indefra, accepteres.
iptables -A door1 -m state --state ESTABLISHED,RELATED  -j ACCEPT
# alt andet smides væk.
iptables -A door1 -j DROP

# Hop til door1 hvis input er fra eth1
iptables -A INPUT -i eth1 -j door1

# NAT-ing sættes op, så alt, hvad der kommer fra lokalnettet,
# bliver videresendt som om det kommer fra firewallen.
iptables -t nat -A POSTROUTING --src 192.168.67.0/24 \
         -o eth1 -j SNAT --to 130.3.2.99

# for at det skal fungere, må vi aktivere forwarding:
echo 1 >> /proc/sys/net/ipv4/ip_forward
```

Der er mange andre ting, man kan foretage sig, for at maskinen bliver sikret: http://www.bastille-linux.org/ (http://www.bastille-linux.org/). Det kaldes også "hardening", hærdning, af et system. For eksempel anbefales det på udsatte systemer at fjerne de tools, som en systemadministrator bruger til sletning, kopiering og omdøbning af filer, mount, diskformatering, oprettelse af filsystemer mv. Hvis en cracker opnår root-privilegier, vil aktionsmulighederne derved være begrænset. Men det vil for de fleste private medføre irritationsmomenter i det daglige arbejde. Man kunne selvfølgelig have en supplements-server stående, som man lige kunne hooke op når man havde brug for det ...

Sådan er der så mange ting, man kunne gøre. De fleste af disse tiltag beskrives i sidste afsnit

Kommentarer til ovenstående firewall er velkomne, særligt hvis der er konkrete ting, som er afprøvet, men også kommentarer, som bevæger sig i yderkanten af emnet.

6.5.7. Linux som proxy server

I det følgende beskrives den situation, at en virksomhed med 40 ihærdige brugere på en 2 Megabit forbindelse ønsker at forbedre browser-response, således, at når en person har hentet en web-side, så kan de andre hente den samme web-information lokalt fra en server med buffer, en såkaldt caching proxy.

6.5.7.1. Proxy med Squid

Squid er en god proxy server bl.a. til Linux, der kan findes på http://www.squid-cache.org/. Squid er en agent, der henter de hjemmesider, brugeren beder om, og videresender resultatet til brugeren. Samtidig kan Squid fungere som proxy-cache - den gemmer hjemmesiden i cache, og er der nu en anden bruger, der vil se den samme hjemmeside, hentes den direkte fra den lokale hukommelses- eller disk-cache på proxy-maskinen - det vil sige, uden at man skal via internettet. Squid tester, om en hjemmeside på nettet er nyere end den i cachen, og kun hvis cachen er forældet, vil en ny version blive hentet. Fordelen ved en proxy-cache er således, at man ofte kan spare måske 50% på båndbredden.

Det er naturligvis ikke alt, som kan gemmes i en cache, f.eks. skal cgi-kald netop ikke køre fra cache. Disse hindringer håndterer Squid dog transparent. Ud over hjemmesider (HTTP) er FTP, GOPHER, SSL og WAIS protokollen understøttet. Squid kan dog ikke klare POP-mail, NNTP (News grupper) og RealAudio.

6.5.7.2. Installation af Squid-serveren

Man bør ikke installere Squid, så den kører som root. Ofte vælger man at lade Squid køre som brugeren **squid** i sin egen gruppe **squid**, og kun med få rettigheder.

Vi har installeret `squid-2.2.STABLE5-1.i386.rpm`, som kan hentes fra http://www.squid-cache.org/. Squid startes op via `/etc/rc.d/init.d/squid`. Squids opsætningsfil hedder `/etc/squid/squid.conf`. Der er *mange* opsætningsmuligheder. Basalt set skal man fjerne kommentartegnene fra nogle linjer i `/etc/squid/squid.conf` og udkommentere andre, og så starte Squid. Den originale `/etc/squid/squid.conf` er stor, og vi vil nu se nærmere på en simpel opsætning af squid.

6.5.7.2.1. Basal opsætning af squid.conf

Man kan i `/usr/doc/squid-2.2.STABLE5/QUICKSTART` finde en minimal beskrivelse af parametre for `squid.conf`. En meget bedre gennemgang fås ved at læse brugermanualen til Squid, som kan findes på http://www.squid-cache.org/Doc/Users-Guide/. En anden god start er også at læse (og gemme) den originale `/etc/squid/squid.conf`, som fulgte med RPM-pakken.

Vi skal nu vise en kort `squid.conf`, hvor vi lader squid køre som brugeren `squid`.

```
#squid.conf -  Basal opsætning

#Laveste niveau af logging
debug_options ALL,1

#Gruppe af IP numre, som kan tilgå Squid (Access Control List)
#Vil man kun give access til netværket 192.168.0.0/255.255.255.0
#så brug følgende linje
acl all src 192.168.0.0/255.255.255.0

#Skal alle kunne bruge Squid så udkommenter følgende linje.
#acl all src 0.0.0.0/0.0.0.0

#Port, man anvender til opsætning af Netscape klienter
http_port 3128

#Lad alle i ACL bruge Squid til HTTP
http_access allow  all

#test følgende sites for at tjeke, om maskinen er koblet til internettet
dns_testnames internic.net usc.edu cs.colorado.edu mit.edu yale.edu

#Kør som effektiv bruger squid og gruppe squid
cache_effective_user squid squid

# Squid vil oftest bruge to-tre gange denne RAM størrelse. Vil man max
# bruge 24 Mb RAM til Squid, så sæt cache_ram til 8 Mb.
# Jo mere cache_mem desto hurtige er cachen (mindre diskaccess).
cache_mem  8 MB

#Maximal størrelse på object i cache
maximum_object_size 4096 KB

# Næste parameter-opsætning er disk cache struktur. Parametre er
#  Dirname - hvor på disken er disk cache, dvs. spool dir
#  Mbytes under spool dir - hvor mange Mb må gemmes i disk cache.
#  Level-1 dir antal - Antal underkataloger under Dirname
#  Level2 dir antal - Antal underkataloger for hver Level-1 kataloger
# Sæt ikke produktet mellem de to sidste vildt højt!
cache_dir /var/spool/squid 100 16 256
```

Lad os se nærmere på, hvad der installeres, og hvordan squid kører. Installer squid med

```
#rpm -ivh squid-2.2.STABLE5-1.i386.rpm
```

Ret `/etc/squid/squid.conf` til som vist ovenfor. Husk dog at gemme den originale `/etc/squid/squid.conf` før ovenstående eksempel anvendes.

- `/etc/squid/` indeholder opsætningsfiler - specielt er `/etc/squid/squid.conf` vigtig.
- `/var/spool/squid` indeholder den dynamiske database. (se parameteren `cache_dir` i `squid.conf`). Over 4000 filer laves med ovenstående opsætning.

- `/var/log/squid` indeholder log-filer.
-
  - `access.log` - Hvad blev hentet fra nettet. Her kan systemadministratoren overvåge trafik.
  - `cache.log` - Opstartsmeddelelser.
  - `store.log` - Oversigt over, hvad der findes i proxy-cache nu.

Med installation af squid blev `squid` brugeren og tilsvarende gruppe oprettet. Kataloget `/var/spool/squid` er tomt men ejet af brugeren squid. Selve databasestrukturen skal man en gang for alle sætte op - hertil bruges parameteren `cache_dir`.

```
[root@sherwood root]# /usr/sbin/squid -z
```

Anvender man 16 og 256 som de sidste to parametre til `cache_dir`, skal man ikke blive bange, når man initialiserer databasen. Der køres hårdt på harddisken, og det tager måske flere minutter - databasen bliver stor!

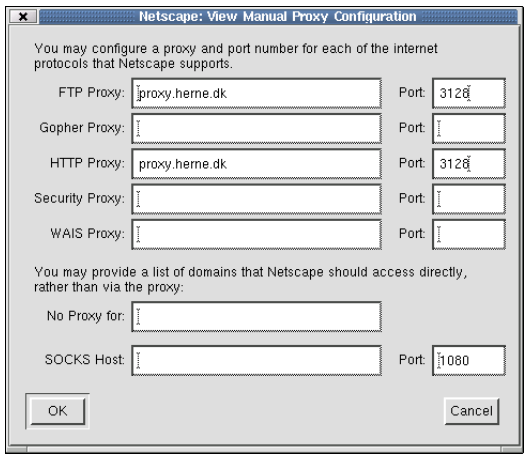
6.5.7.2.2. Brugerens opsætning

For at man kan få squid til at virke sammen med Lynx og Wget, kan man sætte følgende systemvariable (her hedder vores proxy-server "proxy.herne.dk")

```
setenv http_proxy http://proxy.herne.dk:3128/
setenv gopher_proxy http://proxy.herne.dk:3128/
setenv ftp_proxy http://proxy.herne.dk:3128/
```

I Netscape skal brugerne under `Edit->Preferences->Advanced->Proxies` vælge `Manual proxy configuration` og tryk på `View`. Som det kan ses på det næste billede, skal man skive navnet på proxy maskinen (her anvender vi `proxy.herne.dk`), og port 3128 svarer til parameteren "http_port" i `squid.conf`.

Figur 6-7. Proxy opsætning i Netscape



Squid er i dagligdagen et meget driftsikkert program og kan anbefales. Vi skal dog lige huske at nævne, at hvis der ikke er blokeret for det via en firewall, så kan brugeren af Netscape faktisk godt køre forbi proxy cachen ved enten at udelade proxy-opsætningen eller at holde SHIFT nede, før der trykkes på et link. Derfor skal firewallopsætningen være lavet, så proxy-programmet godt kan hente data fra internettet, men at brugere ikke kan - hvis man virkelig ønsker, at folk skal bruge proxy-serveren.

6.5.7.3. Transparent proxy med squid

Man kan lave transparent proxy med squid sammen med iptables. Kernen skal være oversat med "Transparent proxying". Se Afsnit 6.5.5.2

Der findes også en mini-HOWTO på http://sunsite.dk/ldp/HOWTO/mini/TransparentProxy.html (http://sunsite.dk/ldp/HOWTO/mini/TransparentProxy.html).

6.5.8. Masquerading med ipchains (kerne 2.2)

I Linux-kerne 2.2 hedder firewall kommandoen **ipchains** , og denne kommando bruges også til opsætning af masquerading.

Forwardkæden sættes op med en default target MASQ eller, hvis default politik (policy). Den kan stadig væk smide væk, men i **ipchains** hedder det DENY.

Den simple og hurtige måde at sætte IP masquerading op på er:

```
[root@myhost /root]# ipchains -A forward -i ppp0 -j MASQ
[root@myhost /root]# ipchains -A forward -s 0.0.0.0/0 -d 0.0.0.0/0 -l -j REJECT
[root@myhost /root]# echo 1 > /proc/sys/net/ipv4/ip_forward
```

hvor ppp0 er det interface, der peger ud imod verden, imod internettet, imod det net, man vil give sine maskiner adgang til via masquerading.

Det kan også give mening at sætte timeouts for masquerading forbindelserne:

```
[root@myhost /root]# ipchains -M -S 7200 10 160
```

Det første tal er timeout for en tcp forbindelse, det næste er hvor længe en tcp forbindelse skal kunne eksistere efter modtagelsen af en FIN pakke, og det sidste er timeout for UDP. Timeout tiderne er i sekunder. Default timeout er 15 minutter, og det kan være lidt kort for en tcp session.

Nu sættes de andre maskiner i lokalnetværket til at bruge myhost som default gateway, og voila! Man kan komme på internettet fra de andre maskiner.

6.5.8.1. Moduler til ipchains

Og dog. Http mm virker nu, men der er et par protokoller, der kræver særlige hensyn. Det er bla. ftp, realaudio, irc og forskellige spil.

For at kunne bruge ftp, real audio, irc, spille quake mm via masquerading, er vi nødt til gøre det lidt mere indviklet. Der findes moduler til **ipchains** , der kan håndtere en del forskellige applikationer: ip_masq_ftp, ip_masq_raudio, ip_masq_irc, ip_masq_quake etc. For at kunne bruge disse moduler, skal de være oversat med, da man lavede kernen. Det er inde under networking options, hvor man skal slå "IP masquerading special module support" og "ipportfw masq support" til. I kerne 2.2.12 er det nødvendigt at slå "prompt for development and/or incomplete code/drivers" til for at få lov til at vælge ipportfw.

Modulerne indlæses nu med:

```
[root@myhost /root]# /sbin/modprobe ip_masq_ftp
[root@myhost /root]# /sbin/modprobe ip_masq_raudio
```

Nu virker ftp og realaudio fra de andre maskiner. Gør det samme med de andre moduler, der skal bruges. Der findes også et værktøj, der hedder IPMASQADM, man kan bruge. Der findes en slags beskrivelse af emnet i IP-Masquerade-HOWTO'en på
http://sunsite.dk/ldp/HOWTO/IP-Masquerade-HOWTO-6.html#ss6.8

6.5.8.2. Opsætning af de andre maskiner i nettet

Default gateway sættes med kommandoen **route**:

```
route add default gw myhost
```

Hvis det skal være permanent, er der lidt forskel på distributionerne. Nogle har et fint opsætningsværktøj, det vil vi ikke komme ind på her. I SuSE er det en `/etc/route.conf`, man kan sætte det ind i. I debian er det direkte i `/etc/init.d/` i et netværks startup script. I Red Hat er det filen `/etc/sysconfig/network`, man sætter sin default gateway ind i.

For yderligere information om IP masquerading kan man læse IP-Masquerade-HOWTO'en http://sunsite.dk/ldp/HOWTO/IP-Masquerade-HOWTO . HOWTO'en indeholder også referencer til en række gode ressourcer.

6.6. Afprøvning af en firewall

Når man har sat sin firewall op, skal man altid undersøge, at den virker som forventet.

- *Funktionalitet:* Test at de services, der skal kunne benyttes, virker. Både indefra og udefra. Der er dog stor chance for, at brugerne nok skal fortælle meget hurtigt, hvis det ikke er tilfældet.
- *Sikkerhed:* Test at firewallen rent faktisk lukker for de ting, den burde. Prøv at bruge services, der skulle være lukket for, ved at prøve at oprette forbindelser udefra. Portscan firewallen for at se, hvad folk kan få at vide om den udefra. Der findes også mere omfattende tests. Man kan forsøge sig med diverse cracker-programmer, eller man kan betale andre for at teste firewallen.
- *Vedligeholdelse:* Hold altid softwaren på firewallen opdateret med de nyeste sikkerhedsrettelser. Læs altid logfiler.

6.7. Epilog

Firewallløsninger kan ikke beskrives fuldstændigt her. Vi har taget en del af Linux-mulighederne, men der er meget, vi ikke har dækket:

- Trusted Information Systems Firewall Toolkit (FWTK): Se http://www.fwtk.org/fwtk/
- T.REX: Open Source firewall: Se http://www.opensourcefirewall.com/
- En side med mange links om firewalle: Se http://www.linux-firewall-tools.com/linux/
- Firewall-1 er en fuld kommerciel firewall: Se http://www.checkpoint.com/products/firewall-1/
- Socks5 er en proxy server, der anvendes mange steder. Se http://www.socks.nec.com
- Mange gode netværkssikkerhedsværktøjer og ressourcer findes på http://www.freefire.org

- En opskrift på test af firewall kan læses på http://linuxtoday.com/news_story.php3?ltsn=2001-05-04-015-20-SC.

6.8. Oversættelse af xconfig netfilter configuration help tekst.

Netfilter er en række faciliteter til at filtrere og berarbejde de pakker, som kommer igennem din Linux box.

Den mest almindelige anvendelse af netfilter er at bruge din Linux box som firewall, der beskytter et lokalt netværk mod internettet. Den type firewall, som denne kerne supporterer, kaldes et pakkefilter, hvilket betyder, at den kan afvise individuelle pakker baseret på type, afsenderadresse, modtager etc. Den anden slags firewall er "proxy-baseret", den er mere sikker men mere besværlig at sætte op. Den inspicerer netværkstrafikken meget mere i detailler, modificerer den og har viden om de højere protokol-lag, som et pakkefilter ikke ved noget om. Endvidere kræver proxy'er ofte ændringer i de programmer, som kører på den lokale klient. Proxy baserede firewalls har ikke brug for support i kernen, men de er ofte kombineret med et pakkefilter, som kun virker, hvis du siger Y (Yes, ja) her.

Du bør sige Y her, hvis du har tænkt dig at bruge Linux som en gateway til internettet for et lokalt netværk af maskiner uden globalt gyldige IP-adresser. Dette kaldes masquerading: Hvis en af dine lokale maskiner ønsker at sende noget til verden udenfor, kan din box "forklæde sig" (maskere sig) som den maskine, d.v.s. den videresender pakkerne til den angivne ydre destination, men modificerer pakkerne, så de ser ud, som om de kommer fra firewall-boxen selv. Det fungerer begge veje: Hvis hosten udenfor svarer, vil linuxboxen uden yderligere kommentarer sende pakkerne videre til den rigtige lokale computer. På denne måde er maskinerne på indersiden af firewallen fuldstændig usynlige for verden udenfor, selv om de kan nå udenfor og får svar. Det er endog muligt at køre globalt synlige servere inde fra et masqueraded lokal netværk ved at anvende en mekanisme, som kaldes port-forwarding. Masquerading kaldes også ofte NAT (Network Address Translation)[3].

En anden anvendelse er transparent proxy-ing. Hvis en maskine på lokalnettet prøver at få skabe forbindelse til en maskine udenfor, kan Linux-boxen uden at det mærkes sende pakkerne til en lokal host, som typisk vil være en caching server.

Der er forskellige moduler, som erstatter tidligere masquerading (ipmasquadm), packet filtering, transparent proxying og portforwarding mekanismer. Se venligst ~kernel/Documentation/Changes for placeringen af af disse pakker.

Vær sikker på at svare N til "Fast switching" nedenfor, hvis du siger ja til netfilter her, eftersom "Fast switching" i den aktuelle version laver bypass på netfilter.

Du bør nok sige Y her, hvis du oversætter en kerne, som skal fungere som router, og N, hvis det er til en

almindelig host. Hvis du er usikker, så svar Y.

[Oversætterens anmærkning: Det er i dag bedst at sige Y til netfilter, idet man også på et lokalnet kan have fordel af at kunne spærre af for uønskede pakker udefra. De store distributioner har det altid med.]

6.9. System hærdning

Hærdning af kernen er relevant på en firewall. Man kan for eksempel tilføje flg. til sit firewall script:

```
# Svar ikke på ping
/bin/echo "1" > /proc/sys/net/ipv4/icmp_echo_ignore_all

# undgå at være med til smurf angreb
/bin/echo "0" > /proc/sys/net/ipv4/icmp_echo_ignore_broadcasts
# Tillad ikke source routing
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_source_route
/bin/echo "0" > /proc/sys/net/ipv4/conf/all/accept_redirects
/bin/echo "1" > /proc/sys/net/ipv4/icmp_ignore_bogus_error_responses
for i in /proc/sys/net/ipv4/conf/*; do
    /bin/echo "1" > $i/rp_filter
done
# Log forsøg på source routing samt redirect
/bin/echo "1" > /proc/sys/net/ipv4/conf/all/log_martians
/bin/echo "0" > /proc/sys/net/ipv4/ip_forward
```

Det kræver en speciel kerne, hvor de forskellige nødvendige features er aktiveret. Man kan for eksempel godt køre en firewall på en kerne, som ikke har logging af fremmede (spoofede) pakker, også kaldet mars-boere, "Martians".

Slutbemærkning:

1. Summen af denne og andre optimeringer kan tydeligt mærkes også på dagens hurtigste maskiner.
2. Der kan være mange grunde til, at man selv vil skrive iptables kommandoerne. Ved afprøvningen af Firestarter konstaterede jeg, at den laver om på filer i /etc/ uden at aflevere en liste over hvilke filer, den har ændret. Firestarter kan ikke save sine resultat filer, firewall scriptet, uden at køre iptables kommandoen. Derfor kan man ikke "se den over skulderen" inden man vælger at acceptere den. Jeg kørte den i en 'chroot'ed partition og den stoppede min firewall uden at spørge.

   For en begynder er det nok en fordel, at den opfører sig så håndfast uden at spørge.

   Ældre versioner af Red Hats firewall-config bruger ipchains og forventer, at man har en modulariseret (distributions)kerne. Det gør ikke så meget, at den råber FEJL når man kører med optimeret non-modul kerne, hvis den i øvrigt bare fungerer. Den kan heller ikke vise, hvad den vil gøre *uden* at gøre det.

   I de senere afsnit vises, hvordan man skriver et script med sine egne firewall regler, og hvis man har gjort det, så kan man jo tage en komplet kopi af /etc til for eksempel /etc-bkup og på den måde sikre sine ændringer mod overskrivning.

Red Hats setup er et xterm - menuprogram, og starter forskellige subprogrammer til opsætning og opsætning. Bortset fra små problemer med samtidig opsætning fra det ene/andet GUI er jeg glad for Red Hat setup. Dens firewall configurator hedder /usr/sbin/lokkit og fungerer. Det er også den, som bruges under installation af Red Hat, og den fås også som GUI version, /usr/sbin/redhat-config-network.

3. NAT bruges dog oftest om en gateway, som har flere IP adresser, som den kan bruge for mapning af de lokale maskiner. Det er som regel nok med 2 - 3 stykker, idet man sjældent løber tør for portnumre som jo er en 16-bit størrelse. Masquerading har yderligere den egenskab, at det oftest kan sættes op til at ske automatisk på et dynamisk tildelt IP-nummer for en klient, som fx. benytter PPP på en modem-linje.

Appendiks A. Revisionshistorie for bogen

Denne bog er skabt ud fra de seks artikler om introduktion til sikkerhed skrevet af Hanne Munkholm og Peter Toft, der kan findes på www.linuxbog.dk/sikkerhed (http://www.linuxbog.dk/sikkerhed).

Vedligeholdelse af bogen koordineres via SSLUG postlisten <sslug-bog@sslug.dk> Alle kan bidrage - til om tilmelding på http://www.sslug.dk/tilmeld.

- Version 2.9.20060510 - 10. maj 2006: Donald Axel, link til Freebsd.org mirror af rootshell. Link til cert.org's alert arkiv.
- Version 2.9.20040518 - 18. maj 2004: Jacob Sparre Andersen: Retter sprog og opmærkning. Rettelser fra dip.nerd.dk og Henrik Kramshøj.
- Version 2.9 - 25. januar 2004: Gert Lavsen har bidraget med links til firewall-distroer. Donald Axel har skrevet Firewall afsnittet om. Iptables kommandoen beskrives grundigt, og der er kommet mange GUI-firewall tools med, tak til Torkil Zachariassen <torkil@flug.fo> for Firewall screenshots og tekst. Har fået ideer fra mange Linuxbog-læsere, blandt andet fra en, som hedder "Kim" (husk altid efternavn), tak for det. Tak til Kenneth Ahn Jensen for korrektur. Jacob Sparre Andersen: Retter sprog.
- Version 2.8 - 7. oktober 2003: Peter Toft tilføjer link til NMAP-artikel. Jacob Sparre Andersen retter små sproglige fejl. Afsnit om fish overflyttes fra itplatform. Johnny Ernst Nielsen sletter døde links.
- Version 2.7 - 6. april 2003: Navngivning (bruger/maskine) til alle eksempler lavet om. Donald Axel og Peter Toft reviderer tekst om SSH på opfordring fra Lars Sørensen.
- Version 2.6 - 1. december 2002: Thomas Hjorth og Peter Toft retter et par småfejl. Peter Toft opdaterer tekst om sudo.
- Version 2.6 - 1. december 2002: Thomas Hjorth og Peter Toft retter et par småfejl. Peter Toft opdaterer tekst om sudo.
- Version 2.5 - 1. september 2002: Peter Toft: URL til Putty ændret og afsnit om honeypot startet. Jacob Sparre Andersen - Harmoniseret brugen af **tar**. Udbedret logiske fejl i SGML-koden. Prøvet at sørge for at standardnavne for brugere og maskiner svarer til resten af bøgerne (tyge@hven).
- Version 2.4 - 14. juni 2002: Casper S finder en slåfejl. Donald Axel retter et par trykfejl. Jacob Sparre Andersen: Rettet sproglige småfejl.
- Version 2.3 - 10. marts 2002: Casper S finder en slåfejl.
- Version 2.2 - 29. december 2001: Peter Toft: Nyt stort afsnit om Tripwire tilføjet. Har også skrevet mere om OpenSSH (efter inspiration fra Henrik Størner). Lille ændring i stikord. Christian Andersen har lavet en opdatering mht. at ipchains nu er standard med i nyere Red Hat versioner.
- Version 2.1 - 21. oktober 2001: Peter Toft: Link til artikler om NMAP, Tripwire og firewall-test. Henrik Lund Kramshøj har kommet med en tilføjelse til at få X-serveren skjult for netværket. Jacob Sparre Andersen: Rettet enheder til (MHz = megahertz, Mbit = megabit, Mb = megabyte).
- Version 2.0 - 11. august 2001: Peter Toft: Mindre SGML-fejl rettet. Rolf Therkildsen: Peter Toft retter overgang fra inetd til xinetd.
- Version 1.9 - 9. juli 2001: Peter Toft: En stribe indeks-referencer indsat (Tak til Claus Sørensen for at nævne dette). Desuden masser af stil-ændringer, så det er langt tydeligere hvad brugeren skriver i

eksemplerne. Har også tilføjet en henvisning til ny bog om Office. Ole Michaelsen: Har tilføjet en langt bedre forklaring på hvorfor man aldrig skal bruge xhost-kommandoen. Claus Nielsen har rettet op på status for Tripwire. Henrik Christian Grove fandt serie døde URL'er som er rettet op. Ole Tange fangede en stribe format-fejl.

- Version 1.8 - 24. maj 2001: Peter Toft: Lille SGML-fejl rettet.
- Version 1.7 - 12. marts 2001: Hanne Munkholm har opdateret firewall-kapitlet så det er ajour med kerne 2.4. Peter Toft: Link til sbscan væk, projektet er vist dødt. Henrik Christian Grove fandt en dum trykfejl i oversigt over bøger. Jacob Sparre Andersen har lavet sproglige rettelser.
- Version 1.6 - 4. februar 2001: Peter Toft: Ny bog om Docbook nævnt i serien. Rettet tre SGML-fejl i henvisninger. Tilføjet link til god artikel om SSH. Tommy Mogensen fandt en sprogbøf. Jacob Sparre Andersen og Hans Schou: Rettet nogle sproglige fejl. Kristian Støchkel fandt en URL, som ikke virkede længere.
- Version 1.5 - 29. december 2000: SGML fejl rettet, så henvisninger er i orden igen. Peter Toft har skrevet afsnit om SSH helt om, så OpenSSH er det foretrukne valg. Der er anvendt input fra Mayank Sarup. Claus Sørensen om SSH-klienter til Windows - slettet en henvisning. Link til den nye bog "Linux - friheden til at programmere i C".
- Version 1.4 - 3. december 2000: Lille skønhedsfejl rettet (manglede en prompt) - tak til Ole Michaelsen og Jesper Laisen.
- Version 1.3 - 10. november 2000: Langt større stikordsregister er nu med.
- Version 1.2 - 26. oktober 2000: Masser af rettelser fra Claus Sørensen.
- Version 1.1 - 30. august 2000: Tomas Bertelsen fandt en dårlig henvisning. Mads Sejersen fandt URL fejl, sikkerhedsproblem ved screensaver + et par andre SGML fejl.
- Version 1.0 - 30. juli 2000: Første offentlige bogversion.

Stikordsregister

Symboler

A

B

C

D

E

F

H

I

K

L

M

N

O

P

R