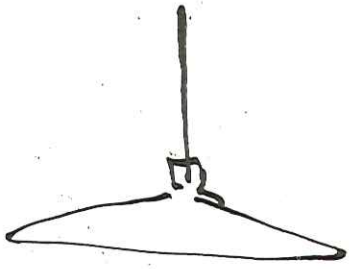
DKUUG

Vejen til viden om Åbne Systemer og Internet

nyt

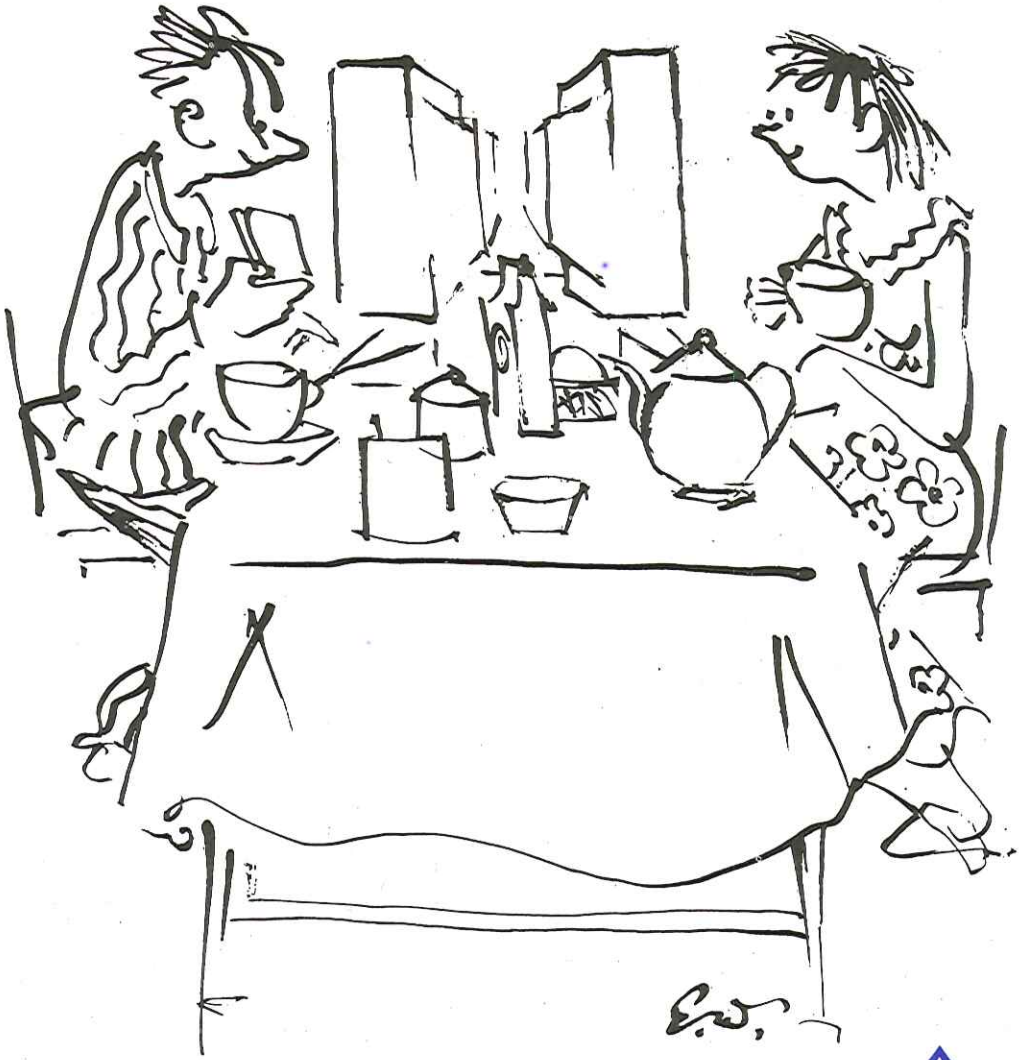
126/september 2000



Tema: Nyheder og journalistik på nettet

ICMP Fingerprinting

Sikkerhedseksperten Rik Farrow skriver om ICMP Fingerprinting



Software-patenter

En form for tyveri?

Katedralen eller basaren

Andet afsnit

Indhold

Tiden hvor journalister bare havde én deadline pr. dag er forbi. På Internettets netaviser lurer deadlines hele tiden, for læserne vil have nyheder hele tiden og så snart begivenheden sker. I denne måneds tema ser vi på nyheder på nettet, kildekritik af oplysninger på nettet og journalistik på nettet.

Leder

Nyhedsindustrien har kørt kanonerne i stilling. Et veritabelt bombardement af nyheder bliver sendt afsted døgnet rundt, og man behøver ikke være langt væk fra noget som helst for at modtage nyheder, om det er på den gammeldags papirmåde, via mobiltelefonen eller på Internettet. Internettet er den åbenlyse måde at få sine nyheder på. Det er hele tiden up-to-date på grund af teknologien, og både TV og aviser kan ikke konkurrere på det plan, men indtil videre er de medier bedre til den dybdegående og mere perspektiverende journalistik, vil nogen mene. Eller det mener de selv, for mange af de traditionelle medier står stadig tøvende og famlende overfor Internettet som medier, selvom de fleste aviser og TV-stationer har fået en hjemmeside at hægte på deres samlede udbud af nyheder. I dette nummer behandler vi i et tema nyheder på nettet, netjournalistik og nyhedstjenester. Flemming Bundgaard fortæller i sin artikel "Nyhedstjeneste bygget op fra bunden: Sådan kan det gøres" – ja, netop hvordan man kan bygge en nyhedstjeneste op. Han har startet BrugNettet.dk og fortæller hvilke faldgruber man skal passe på og hvad en forretningsplan er og hvorfor det er nødvendigt at have én. Per Andersen har interessante synspunkter om kildekritik på nettet, så "få fyldt på" inde i bladet.

Den amerikanske sikkerhedsekspert Rik Farrow har skrevet en artikel specielt til DKKUG-Nyt om ICMP Fingerprinting, som du kan læse tids nok til at være på forkant med udviklingen indenfor IT-sikkerhed.

Go'fornøjelse!

Ny nordatlantisk LUG har SSLUG som idol

Velkommen til verden siger vi til Linux-áhugafelagið, som er én af de senest tilkomne LUG'er (Linux User Group). Linux-áhugafelagið er en færøsk LUG, og har indtil videre 12 entusiastiske medlemmer. Gruppen har planer om at få en maskine kørende under navnet flug.fo og have månedlige møder den første onsdag i hver måned på gymnasiet i Hoydal (Tórshavn). Det første møde finder sted den 6. september med en "installfest", hvor interesserede kan få installeret Linux.

SSLUG fans

Det er en god idé at have idoler og den færøske LUG har dansk/svenske SSLUG som forbillede. En af initiativtagerne bag Linux-áhugafelagið, Jacob Sparre Andersen, udtaler at "det hele" ved SSLUG er værd at kopiere. "Både mødeaktiviteter og den elektroniske kommunikation. Arbejdsindsatsen bliver fordelt fornuftigt og folk får løst deres problemer - det første er vel en grundidé i „Open Source"-verdenen og det sidste er formålet med en LUG". En anden af medstifterne af den nordatlantiske LUG, Torkil Zachariassen, er ligeså begejstret for den dansk/skånske søsterorganisation: "SSLUG har overbevist os om, at der et behov for at Linux folket mødes, fysisk såvel som elektronisk i fora, hvor de nationale sprog kan anvendes uden videre. SSLUGs mail lister integreret med en website med arkiv og søgemaskine, hyggemøder, konferencer og for eksempel „Friheden til at vælge" - på dansk/svensk naturligvis - lavet af engagerede frivillige, der ønsker at dele sin begejstring, er nogle af de elementer, som vi vil forsøge os med - og formentligt skalere en faktor 100 (5M/5k) ned. TurboLinux cd´erne, som vi fik fra SSLUG, gik som varmt brød. Flug.fo maskinen er allerede doneret af en glad giver, og der er installeret RedHat på den, og den skal nu sættes på plads. Webmasterne har lavet pingviner med tophue og det nationale flag på maven. Maskinen forventes i luften på www.flug.fo, inden installfesten den 6.september". Som et kuriosum kan nævnes, at et „Áhugafelag" er en „interesseforening" - det vil sige en interesseorganisation eller en hobbyforening.

Indmeldelse kan ske til Torkil Zachariassen torkil@heima.als.fo

ICMP Fingerprinting

by Rik Farrow
rik@spirit.com

One of the first steps in successfully breaking into a computer is to identify the operating system. Obviously, if you do not want someone breaking in, you want to make it more difficult to discover information about your operating system. The easy ways to identify an operating system include banner grabbing, port scanning for characteristic ports, reading email headers and using a tool that does TCP fingerprinting. But there is a new technique surfacing that uses ICMP.

ICMP, Internet Control Message Protocol (RFC 792) provides a communication channel between different IP stacks. As such, the natural inclination is simply to permit all ICMP traffic into and out of your networks. But, this can be problematic, as ICMP packets are used in probes, denial of service attacks, and even for tunneling data. I am going to focus here on using some arcane tricks for probing. Most of this new information has surfaced just this year, based on research done by Ofir Arkin, an Israeli security consultant, who published a paper about tricks using ICMP: http://www.sys-security.com/archive/papers/ICMP_Scanning.pdf. In this paper, Arkin discusses some unusual uses of ICMP, for example, sending one portion of a fragmented packet but never sending its other part, eliciting an ICMP Time Exceeded packet. Note that the trick here is to send only part of a packet, and you will get an ICMP response, possibly through a firewall that blocks all incoming ICMP, but permits any outgoing messages. You can read more about these techniques in a magazine article I wrote for Network Magazine September issue, and will have posted on my Web site (http://www.spirit.com/Network/net0700.txt) after this issue is mailed.

Arkin did not stop his research, but continues to post interesting bits of information about other ICMP tricks to the Bugtraq mailing list (http://www.securityfocus.com/). For example, even though only routers are supposed to respond to ICMP Address Mask Requests, some Microsoft systems, Solaris, and ULTRIX systems will repond to these requests. You can tell if the target runs Solaris by sending the Address Mask Request in two fragments—the Solaris system responds with a netmask of all zeroes in this case. But if you set the DF (Don't Fragment bit), Microsoft 98 and 98SE systems will send you a reply with this bit reset.

Arkin also played around with setting the code bytes in ICMP types where there are not supposed to be any codes. Codes function as subtype indicators, for example there are 15 Destination Unreachable codes, but none are used with Echo Request (PING) or Timestamp Requests. But by setting the code field to some arbitrary value, Arkin noticed that Microsoft Windows 2000, NT4 with SP6a, and Windows 98SE would reset the value accompanying Echo Requests. And setting a code value with a Timestamp Request causes some systems, Microsoft 98, 98SE, and 2000, to stop responding to Timestamp Requests. Microsoft Windows machines that will not answer any Timestamp Requests are Microsoft Windows 95 and Microsoft Windows NT 4.0 Workstation with SP 6a (and below). All other operating systems answered the ICMP Timestamp request.

Linux systems can be discovered by sending a Timestamp Request with the DF bit set. While both the Windows systems that respond to this and Linux resets the DF bit, Linux has a time to live (TTL) of 255, while Windows stacks use 128 in ICMP packets. And, you can distinguish Windows 2000 from other Windows stacks by sending an Echo Reply with the ToS (Type of Service) set to an abnormal value (something other than zero but less than 15)—only Windows 2000 will set this value back to zero in the response.

While this business with strange ICMP packets all sounds a bit complicated (it is), sooner or later someone will write a tool to automate the process. Arkin uses snort http://www.clark.net/~roesch) for a lot of the packet sniffing, and mentions using a tool named sing (http://sourceforge.net/projects/sing) as a tool for generating some of the unusual ICMP packets.

For now, if you are concerned about people probing your networks, looking for interesting systems to "own" (takeover in hacker jargon), I suggest that you use firewalls if you can to block most incoming ICMP. The most important ICMP type to permit in is Destination Unreachable. Block all outgoing ICMP types. While this is pretty stringent, there are attack tools (some of the Distributed Denial of Service tools) that use ICMP Echo Relies to send commands to agents, and the LOKI tool also uses ICMP Echo Replies for tunneling through firewalls.

You can get a list of all ICMP types and codes by visiting ftp://ftp.isi.edu/innotes/iana/assignments/icmp-parameters, or check out http://www.spirit.com/Resources/icmp.html.

Hvad er netjournalistik?

I takt med udviklingen i de nye medier skal de gamle håndværk udvikle sig tilsvarende. Først skulle for eksempel journalistikken tilpasse sig radioen, derefter TV-mediet og nu de webbårne medier. Det har givet journalistikken helt nye muligheder for udfoldelse, men samtidig udfordrer det den tænkning, der ligger bag traditionel journalistik. Journalisten Nils Ulrik Pedersen har opfundet en definition på hvad netjournalistik er, og på det stadie net-journalistikken befinder sig på lige nu, dele han den ind i tre kategorier:

1. Netversion af allerede eksisterende indhold (shovelware)
2. Originalt indhold på nettet
3. Originalt indhold på nettet som inddrager de unikke muligheder for journalistisk forar-bejdning, fortælleteknik, multimedie, interaktivitet, personliggørelse, dialog med mere.

I dag ser man nyhedsformidlingen på alle tre niveauer, og der er mange danske aviser, der har været slemme til at holde sig på niveau 1, altså bare at smække artiklerne skrevet til den daglige avis op på deres hjemmeside. Hvis kreativiteten presses, drister webmasteren sig til at sætte ekstra linieskift ind i et misforstået forsøg på at hjælpe læseren. Alle undersøgelser viser netop, at tekster skal skrives til nettet og det ikke er en god idé bare ukritisk at overføre artikler til nettet uden forudgående omskrivning.

Hvis man så aspirerer til niveau 3, definerer Nils Ulrik Pedersen netjournalistik som journalistik der er urepræsenterbart i noget andet medium. Han mener, at netjournalistikkens karakteristika blandt andet er nyheder her og nu uden snærende deadlines inklusiv tryk- eller sendetider. Netjournalistikken er den sande globale kommunikationsform, hvor alle med en internetforbindelse kan følge med i de samme nyheder. Der er ingen pladsbegrænsninger. Hvor rapporten om den amerikanske præsidents eskapader med en praktikant fra det hvide hus af pladshensyn ikke kunne bringes på TV eller radio (og ej heller er egnet), kunne rapporten lægges på nettet, hvor millioner af mennesker kunne klikke sig ind. Netjournalistik giver unikke muligheder for interaktivitet og dialog, for eksempel afstemninger om aktuelle emner, video- og audioklip.

(kilde: www.netjournalist.dk)

FAKTABOKS

Fremtidsudsigter

Skræddersyet til dig

Internet-teknologien har allerede revolutioneret den måde vi får informationer på i dag. Mange eksperter kigger i kaffegrumset og forudsiger, at mediebilledet vil fortsætte med at ændre sig i de næste fem år. Én af eksperterne er Preben Mejer, Udviklingsdirektør hos Tele Danmark Erhverv, som i en artikel på www.netjournalistik.dk kommenterer fremtidens mediebillede.

Fremtidens medier vil være fragmenterede, skræddersyede og individuelle. Kort sagt personaliseret. Den information, som man som bruger vælger at modtage kommer til en computer, mobiltelefon eller fjernsyn. Preben Mejer forudser af mindst én af de store landsdækkende aviser afgår ved døden, på grund af Internettets fremadstormen, mens de glittede magasiner stadig har en rolle at spille, men ofte med et website som samarbejdspartner a la Yahoos papirmagasin.

Borgerjournalistik

Et andet begreb, der kommer til at indgå i fremtidens mediebillede, er borgerjournalistik, hvor Internettet er den helt naturlige partner. Begrebet blev for alvor kendt under Kosovo-krisen, hvor borgerne i Beograd på allernærmeste hold kunne kommuni-kere on-line til omverdenen, som dermed kunne følge med i rædslerne. Borgerjournalistik ser man mest på newsgroups, hvor interesserede samler sig i et fællesskab om et bestemt emne. Det ses også på forskellige websites, hvor man ytrer sig om et centralt emne.

Teksten er konge

De seneste Internetundersøgelser viser, at flertallet af netbrugere er tiltrukket af teksten først, og ikke af billederne, som man havde troet.

Af Lotte Kristiansen
Lk@dkuug.dk

Fotograferne og webdesignerne græd, mens journalisterne lo. I en undersøgelse foretaget af det toneangivende amerikanske institut for journalistik The Poynter Institute, fastslås det, at alle øjne vender sig mod teksten når man læser på nettet, mens tidligere tiders undersøgelser af "rigtige" papiraviser viser, at læsernes øjne først vandrer hen imod billeder og grafik.

Resultaterne, som er fortaget i USA i samarbejde med kommunikationsinstituttet på det amerikanske universitet Stanford, er revolutionerende fordi det vender den traditionelle tænkning på hovedet. En af forskerne bag undersøgelsen, Andrew Vigal, knytter på Poynters hjemmeside (www.poynter.org) et par kommentarer til undersøgelsen: "Et par billedjournalister havde fået lov til at smugkigge i undersøgelsens resultater før offentliggørelsen, og de var mildest talt skeptiske. Undersøgelserne gik imod alt hvad de tidligere var gået ud fra, nemlig at billeder og farvestrålende grafik var det der trak". Forskerne fastslår, at de ikke ville bevise, at netlæseres vaner er anderledes end avislæsernes vaner. Men det var netop hvad de var – meget til Poynter-forskernes egen overraskelse. Men Internettet og adfærden og læsning på Internettet er en ny videnskab. Og Poynters undersøgelse fra 1990 om avislæseres adfærd kunne altså ikke bare overføres til en computerskærm.

Hovedkonklusionerne

On-line læsere kigger først på overskrifter eller nyhedsresuméer, mens forskerne bemærkede, at læsernes øjne derefter vandrede hen på billederne og grafikken på skærmen. Men det skete oftest efter, at læserne havde læst en hel artikel og derefter klikkede tilbage til forsiden. Og som om det ikke var slemt nok for billederne, så var det endnu værre for grafikken. Kun 22% af brugerne så på grafikken, hvorimod 64% fik set på billederne.

Én af konklusionerne eller forklaringen på tekst som førerhunden på nyhedssider er, at når man surfer på nettet, så er man mere fokuseret fordi man søger noget specifikt. Det er de færreste, der surfer på nettet for underholdningens skyld og uden mål med søgningen. En anden konklusion er, at netlæserne foretrækker ligefremme overskrifter frem for sjove eller kunstneriske overskrifter, og her fremhævede brugerne, at on-line nyhedsmedier var bedre til at komme til sagen end aviserne. På den måde kunne læserne nemmere fornemme om artiklen var værd at ofre et klik – og dermed tid – på. Så til webredaktører og nyhedsformidlere på nettet anbefales det at være ligefrem og effektiv med overskrifterne, så selvom man fristes til at udfolde sine kunstneriske evner, så betaler det sig at vende tilbage til gode gamle journalistiske dyder. Læserne understregede også i de efterfølgende interviews, hvor vigtig det var, at artiklen var godt skrevet. Det kan gøre hele forskellen på, om man vælger den ene nyhedssite fremfor den anden.

Rulletekster

En anden interessant slutning på undersøgelsen er, at on-line læserne ikke havde noget imod at "scrolle", det vil sige at rulle ned på siden for at læse lange artikler. For det første fordi on-linelæseren har haft besværet med at klikke sig ind på artiklen og derfor er der større chance for at artiklen faktisk bliver læst. På en almindelig avisside er læserens øjne svigefulde, og bliver nemt tiltrukket af andet læse- og billedstof, hvis artiklen ikke lever op til forventningerne. Ca. 75% af artiklerne blev læst i on-line udgaven, hvorimod kun ca. 20-25% læser hele avisartikler. Som et eksempel på et nyhedssite, der får glimrende anmeldelser hele tiden, og som er alenlang, er CNN.com. Hvis bare stoffet er inddelt i tilstrækkelig mange logiske kategorier og underkategorier, viser undersøgelsen, at det er nemt for læseren at finde det stof han søger. Det samme gælder den norske netavis www.nettavisen.no. som også får glimrende anmeldelser og er født til nettet.

Deltagerne i undersøgelsen fik monteret letvægtsudstyr på hovedet, som skulle tælle øjenbevægelser. Udstyret er oprindeligt udviklet af militæret og er videreudviklet af eksperter, specifikt til dette formål.

Ikke billedfjendtlig

Skal siderne så være fuldstændig rensede for alt hvad der kan ligne et billede eller et stykke grafik? Skal alle nyhedssider ligne Jakob Nielsens useit.com? Ja, vil Jakob Nielsen selvfølgelig sige, men Poynter maner til besindighed. Den fejl mange on-line nyhedsredaktører har gjort er, at de har overført en avisside til Internettet. Og sådan fungerer det altså ikke. Altså må man tage billedbehandling og brugen af billeder og grafik mere alvorligt. Poynter foreslår, at billederne skal have større gennemslagskraft, for eksempel ved at man skærer billederne tættere på det de skal fortælle, og ikke er rutinebilleder med bred vinkel. Og det hjælper ikke at sætte større billeder på siden, understreger undersøgelsen. Billedredaktører skal også passe på ikke at lave fantasifulde overskrifter der løber ind i billederne, for brugerne kan ikke lide dem.

Kritikken kom

Undersøgelsen som er én af de første på området, var ikke mange timer gammel før kritiske røster begyndte at melde sig. Blandt er undersøgelsen blevet klandret for at have brugt for få deltagere. 67 respondenter blev brugt i undersøgelsen, og de var alle erfarne og regelmæssige brugere af Internettet, som læser on-line nyheder mindst tre gange om ugen. Kritikken går på, at antallet er for lavt til at kunne komme med hårdtslående konklusioner, og i øvrigt mener kritikerne, at Poynter burde have brugt almindelige Internetbrugere, fremfor erfarne og regelmæssige brugere. Kort sagt: deltagergruppen var for snæver. Desuden fik deltagerne lov til at kigge på nyhedssites, der allerede var kendte for dem. En af kritikerne af undersøgelsen, Alan Jacobson, direktør i et Internet designfirma der arbejder med avisers online udgaver, pointerer at mange af de sider ikke ændrer udseende og deltagerne derfor ikke kigger på billederne, men på teksten som fylder mere end 50% af siden. Men Alan Jacobson byder undersøgelsen velkommen, fordi den tager hul på en diskussion om hvordan Internet brugere bruger og læser netnyheder. Og al begyndelse er som bekendt svær.

Kend din læser

Undersøgelsen kan også bruges som et opråb til redaktører, webmastere og andre der vil lægge information og nyheder på Internettet: Kend din målgruppe. Man skal kunne fange sine læsere som har andre vaner på nettet end når de læser aviser eller ser TV. On-line læsere skal have deres nyheder mellem besvarelser af e-mails og samtaler med kolleger og før de skal tage telefonen. Adgangen til nyhederne skal være hurtig og overskuelig.

Facts om undersøgelsen

Deltagerne blev udstyret med avanceret hovedudstyr sammen med specialudviklet software, der skulle måle øjenbevægelser (hvis et øje hviler på et punkt i mindst en tiendedel af et sekund er det en øjenbevægelse), antal skærmbilleder og keyboard aktiviteten. I alt blev der målt 608.063 øjenbevægelser og 24.530 museklik. De 67 deltagende brugte sammenlagt 40 timer på at surfe på deres favorit nyhedsportaler.

Nyhedstjeneste bygget op fra bunden: Sådan kan det gøres

Af Flemming Bundgaard
Info@brugnettet.dk

Flemming Bundgaard, som står bag IT nyhedstjenesten BrugNettet, har bygget tjenesten op fra bunden. Her giver han opskriften på hvordan han udviklede tjenesten BrugNettet: sit erklærede hjertebarn.

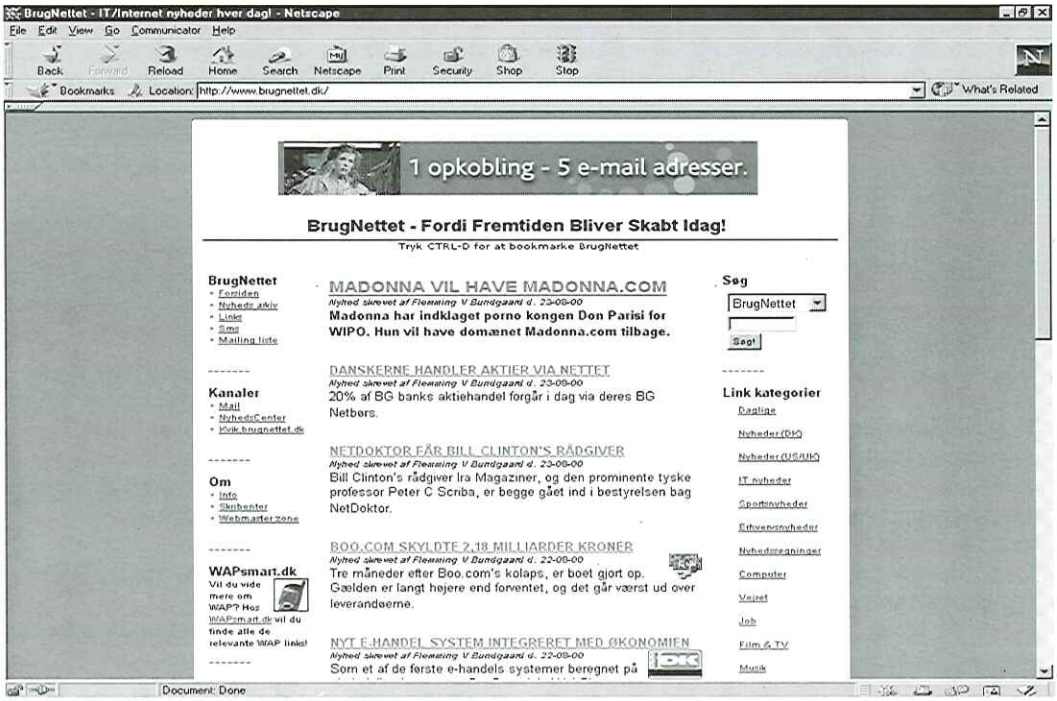
Historien bag BrugNettet

Det hele startede vinteren 1998. Da dukkede BrugNettet op som en portal. I april 1999 dukker der nyheder op på BrugNettet, og omkring sommeren samme år blev BrugNettet til en ren IT nyhedstjeneste. Ideen med nyhederne kom sig af, at ingen andre danske nyhedstjenester levede op til mine ønsker. For eksempel var der ingen links i nyhederne, eller noget der bare lignende en udnyttelse af nettet som medie. Så må man jo selv gøre det. Og som sagt så gjort. I starten havde jeg kun vage ideer om hvad jeg ville med websitet og hvem jeg henvendte mig til med det og så videre. Og det var min første indledningsvise fejl.

Forretningsplanen skal først udvikles

Nu kan jeg så passende fortælle hvad jeg burde have gjort fra starten, fremfor at vente til nu. Først og fremmest, så skal forretningsplanen for ens nyhedstjeneste udvikles. Det sikrer fokus fra starten, og man udgår at løbe panden mod en mur eller forvirre sig ind i blindgyder og lignende. Din forretningsplan skal fortælle hvorfor din idé er bedre end alle andres. Kort sagt. Det første man skal få styr på er formålet. Hvorfor bruge tid og penge på at udvikle og drive en nyhedstjeneste? Jeg bruger ca. 1-3 timer dagligt på BrugNettet, og det indskud giver et afkast jeg finder acceptabelt. Derfor gør jeg det. Men hvorfor vil *du* gøre det?

Formålet kan være mange ting. Idéen med tjenesten kan være, at den skal skabe en indtægt via bannerreklamer, eller at den skal skabe opmærksomhed omkring et specielt emne. I BrugNettets tilfælde er formålet dog ganske anderledes: Tjenesten skal skabe opmærksomhed omkring mig, samt hvad jeg kan berige virksomheder med i kraft af mine programmeringsevner, og det virker. Jeg har aldrig haft mere travlt end netop nu.



Et andet interessant formål finder vi hos Movie Media Online (www.moviemedia.dk). Navnet dækker over en online butik der sælger DVD film. Deres nyhedstjenestes primære formål er at få fat på DVD interesserede og fastholde dem. Dermed er chancen for at de vil købe DVD film samme sted, langt større. Og det gør de også med stor succes.

Fælles for disse to eksempler er, at hvor trist det end lyder, så er bannerreklamerne ikke det der giver mad på bordet. For at man som website er interessant at annoncere på, skal man have en massiv trafik fra en veldefineret brugergruppe, og det er de færreste der kan det i et omfang der er stort nok. Hele internet markedet er lagt an på potentialer, men det hjælper ikke ret meget, hvis potentialet ikke kan omsættes til noget brugbart. Derfor vil vi til stadighed se mange forsøge at bygge en virksomhed op, men deres grundlag er som oftest for spinkelt. Derfor skal du hverken placere din økonomiske basis i banner-reklamerne eller gøre den indtægt til dit primære formål med nyhedstjenesten. For det er svært, og giver ikke penge nok til at være grundlaget for en helt ny tjeneste eller virksomhed.

Men formålet med din nyhedstjeneste kan som sagt - og illustreret - være mange ting. Flere primære og sekundære formål kan også sagtens tænkes, det er alt sammen helt op til dig og dine ønsker. Forretningsplanen skal så beskrive, hvordan du vil opfylde dit mål.

Berig målgruppen

I forhold til formålet kan man så finde midlet. Hvad vil du med din nyhedstjeneste, overfor hvem vil du gøre det, og hvordan vil du gøre det? Jo mere du ved om din målgruppe, jo lettere bliver det at kommunikere med dem. Du vil for eksempel logisk nok aldrig få fat på e-entreprenører, hvis din tjeneste handler om lystfiskeri. Men hvordan kan du så få din målgruppes opmærksomhed? Logisk nok, så skal man lave en tjeneste der omhandler emner de interessere sig for, eller på anden måde kan berige dem. For som alt andet her i livet, så handler det om at investere noget og få et afkast, der er mere værd end indskuddet. Det beviser brugernes adfærd ganske effektivt. Så alt

hvad du fremover laver, skal laves 100% målrettet imod din målgruppe. Navn, design, funktioner, markedsføring og valget af nyheder skal alt sammen sigte på din målgruppe. Hidtil har BrugNettet båret meget præg af, hvad jeg fandt interessant. Den noget uovervejede strategi er nu lagt om, så alle historier vi nu bringer, kan bruges til et eller andet af vores målgruppe, nemlig e-entreprenørerne. Derfor vil man ikke længere kunne finde nyheder om Intel og AMD's krig, for det er dybest set uinteressant i forhold til brugerne.

Undersøg konkurrenterne

En anden ting der med succes kan undersøges, er markedet. Hvilke andre tjenester findes der i sammen branche? Hvis du ikke kan differentiere dig i forhold til dine konkurrenter på en eller anden måde, så er chancen for at få brugernes opmærksomhed meget lille. Men differencen kan til gengæld være placeret mange forskellige steder. Det kan være emnerne du tager op, din tjenestes troværdighed eller noget tredje. Nyhedsflowet er ikke det eneste der tæller. En af de nyeste trends er at udbygge tjenesterne med funktionalitet. Senest har ComON (www.comon.dk) introduceret deres NetSvar, og ComputerWorld (www.computerworld.dk) holder fast i deres DatoWorld, JobWorld og så videre. Et endnu bedre eksempel er Børsen Online (www.borsen.dk), hvor nyhederne næsten er det mindste. Deres website er stoppet til randen med funktionalitet for den travle forretningsmand. På BrugNettet har vi – udover vores stærke informationsstruktur - også sådanne funktionelt orienterede tiltag på vej. Det indebærer blandt andet en såkaldt domæne browser, hvor vores bruger vil få en grafisk brugerflade ovenpå RIPE’s (www.ripe.net) domæne database, og vi er også i gang med at indgå aftaler med forskellige websites om en job sektion. Så undersøg markedet, og lad din idé blive inspireret.

Kodningen af din tjeneste

Efterhånden er din idé blevet gennemarbejdet. Du har et formål, du har en målgruppe og du ved hvad du skal gøre for at få fat på den. Så er det værste overstået. Nu er der kun koden tilbage. Om du vælger det ene sprog fremfor det andet er i og for sig underordnet. Det, det hele kommer an på er udformningen af den database du har under motorhjelmen. ASP, Perl og PHP kan - dybest set - det samme. Det samme er tilfældet med databaser som Access, SQL Server og mySQL. Det, der gør de produkter forskellige, er prisen og hvor lang tid der går inden du skal opgradere til et eventuel dyrere produkt. Så der er det alene pengepungen og personlige præferencer der tæller. Men som sagt, så er udformningen af databasen, hvilke tabeller der er, og hvilke attributter de har, det der tæller. For holder din struktur, så er den skalerbar og kan nemt udbygges med nye features. Kan den ikke det, så vil du hurtigt drukne i arbejde, når kravet melder sig fra din eller dine brugeres side.

Markedsføringen er alfa og omega

Markedsføringen af tjenesten kan være svær. Specielt hvis man ikke har flere millioner på kontoen. Har man det, så kan man i princippet springe alt det her over, og bare markedsføre sig selv aggressivt. Så skal man bare passe på ikke at lave samme stunt som Boo.com (www.boo.com), der brugte omkring en milliard kroner, og alligevel gik fallit. Det er dog de færreste der har det. Heldigvis.

Analyser har vist, at links mellem tjenester har en bedre effekt end for eksempel banner reklamer. Så der er der i hvert fald en metode. Ligeledes er „word of mouth" heller ikke at foragte. En tredje - og endnu bedre metode - er det som kaldes for „integrated marketing". Idéen går ud på, at man integrerer forskellige medier, og udnytter dem hver især, hvorved man fastholder brugerens opmærksomhed. Vi ser det blandt andet i fjernsynet og i aviserne, hvor de hele tiden henviser til deres respektive hjemmesider. Og på deres hjemmesider kan man tilmelde sig nyhedsbreve og så videre. Som tjeneste forbliver man hele tiden i brugerens opmærksomhed, og det er, hvad markedsføringen handler om. For endnu engang at henvise til BrugNettet, så kan man hos os modtage vores nyhedsbrev som både e-mail, SMS eller ICQ modtager. Hvert nyhedsbrev er specielt tilpasset. Og formålet er som sådan ikke at gøre dig som bruger glad, men at gøre dig opmærksom på, at vi stadig eksisterer. Som udgangspunkt kan man aldrig få markedsføring nok. Selv dårlig markedsføring er - trods alt - markedsføring.

Summa summarum

Konklusionen er, at hvis du virkelig vil satse på en nyhedstjeneste, skal du tænke dig godt om, og ellers afsætte en masse tid. Og uanset hvad, så tænk på dine brugere, og hvad du kan gøre for at få deres opmærksomhed og gøre dem glade. Kun derved kan du få dit formål opfyldt.

BrugNettet kan findes på www.brugnettet.dk. Der kan du også melde dig på vores daglige nyhedsbrev.

Siden sidst

25timer.dk forsinket

Første danske avis født på Internettet – 25timer.dk – ser ikke dagens lys før midten af september. Den originale lanceringsdato i juli, er derfor for længst overskredet. Manden bag 25timer.dk, Poul Høgsted siger, at problemet har været manglende investorer til projektet. Selv mener han, ifølge MediaWatch, at investorerne holder sig tilbage på grund af de mange konkurser på dot.com-markedet.

Opkald din baby efter dot.com og bliv rig

Utroligt, men sandt! En amerikansk dot.com-virksomhed, Internet Underground Music Archive (IUMA), har udskrevet en konkurrence, hvor man får 5000 amerikanske, grønne, højkurs-dollars, hvis man opkalder sin baby efter firmaet, der specialiserer sig i musiktjenester i stil med MP3.com. Den lille nyfødte dreng kom altså til at hedde Iuma Dylan-Lucas Thornhill, og hans forældre kunne gå smilende hen i banken og åbne en polstret børneopsparing hurtigt efter nedkomsten, fortæller BrugNettet.dk.

IBM-praktikanter udvikler værktøj til Linux

Det nye værktøj hedder Sash Weblications, og gør det muligt for webudviklere at skrive software til Linux uden erfaring med programmering, skriver CNET. Den nye teknologi understøtter KDE og Gnome, og frigives om kort tid. Læs mere på:
http://sash.alphaworks.ibm.com/overview.html

Døren lukket for netjournalister til OL

Dørene til efterårets ellers så festlige olympiske lege, bliver knap så festlige for journalister der arbejder for on-line aviser. Den Internationale Olympiske Komité (IOC) har lavet en aftale med det amerikanske TV-selskab NBC, og mener derfor ikke, at de kan give netjournalister de samme privilegier som TV-journalister. Der bliver derfor ikke nogen on-line dækning af legene på Internettet.

Microsoft nedgør Linux – igen!

Den norske online tjeneste Digi.no skriver, at den seneste tids rygter om at en version af Microsofts Office-pakke til Linux skulle være på trapperne, er fulde af løgn. I pressemeddelelsen fra Microsoft ikke bare dementerer de Linux-udgaven, men siger også, at Linux ikke kan tilbyde den infrastruktur, der er nødvendig for at kunne køre en arbejdspakke som Office og kalder Linux for umodent til at kunne møde kundernes forventninger.

På forkant med arrangementerne

På DKUUGs hjemmeside www.dkuug.dk kan du ved at klikke dig ind og skrive din e-mail adresse, modtage ugentlige forvarsler om de gratis arrangementer DKUUG tilbyder, og de betalingsarrangementer der også foregår i DKUUG-regi. I øvrigt kan du også framelde dig tjenesten på hjemmesiden, hvis du ikke længere ønsker at få påmindelser.

DualHead på Linux

Matrox Graphics Inc. siger i en pressemeddelelse, at de nu er det første grafikkortfirma, der producerer "single-slot DualHead" til Linux. Betaversionen understøtter tre DualHead indstillinger: Multi-Display, Clone and TV-Output – og op til otte display konfigurationer på Matrox Millennium G400 series. Man kan downloade betaversionen af Linux XFree86 driveren på http://www.matrox.com/mga/drivers/latest_drivers/home.htm

On-line køleskaber

Som led i et forsøg skal 50 danske hjem nu have installeret Electrolux Screenfridge. Køleskabet har en skærm monteret fast på døren og skærmen er tændt 24 timer i døgnet og giver hurtig adgang til Internettet. De heldige test-deltagere kan også hente indkøbssedler frem fra on-line skabet via en Wap-telefon, der hører med i forsøget, skriver Mediawatch.

Software-patenter - nej tak

Af medlemmer af Skåne Sjælland Linux User Group.

Kronikken blev bragt i dagbladet Information, fredag den 1. september 2000.

Et patent giver en opfinder monopol på anvendelsen af en opfindelse. Et softwarepatent giver monopol på anvendelsen af en idé eller metode, der bruges i software.
Der er tale om monopol i den forstand, at man kan forhindre andre i erhvervsmæssig brug af ideen/metoden, hvad enten det drejer sig om komplicerede, tekniske metoder eller om enkle ideer - for eksempel patentet på at en webside bruger et billede af en indkøbsvogn til at vise kunden hvilke varer vedkommende har valgt i en handel via Internettet. At opnå patent på software er meget udbredt i USA, men har hidtil kun været muligt i et meget begrænset omfang i Europa. Dette er tilsyneladende ved at ændre sig. I juni i år har Det europæiske Patentkontor foreslået indførelse af softwarepatenter i form af et ændringsforslag [1] til den europæiske patentkonvention, som Danmark er tilsluttet. Formålet med softwarepatenter (og patenter i det hele taget) er at fremme den teknologiske innovation. Patenter skal beskytte innovative firmaer mod efterligninger af deres ideer, således at de får større incitament til at investere i forskning og udvikling. Samfundet tilgodeses via kravet om at patenter offentliggøres, hvorefter andre firmaer har lov til at bruge ideen efter udløbet af de 20 år som et patent gælder. Vi vil argumentere for, at i praksis virker softwarepatenter ikke fremmende for innovationen.

Patentret versus ophavsret

I Danmark og det øvrige Europa har software hidtil været beskyttet af ophavsret, ikke patentret. Patentloven indeholder simpelthen en undtagelse netop for software.

Indehaveren af ophavsretten til et program kan forbyde kopiering af programmet, ligesom en forfatter af litteratur kan forbyde andre at kopiere hans eller hendes værk. Indehaveren af et softwarepatent vil have den langt mere vidtgående ret: retten til at forhindre andre i at skrive et program, der anvender den patenterede metode eller idé.

Selv om programmer ikke er kunstneriske produkter, giver det god mening at anvende ophavsretten, for programmer er dybest set tekst. Kildeteksten til et program er en tekst bestående af instruktioner til computeren. Og det er faktisk sådan, at kildeteksten til mange programmer indledes med at programmøren/forfatteren bruger det lille (c) til at gøre opmærksom på at ophavsretten tilhører vedkommende personligt, en organisation eller et firma.

Det nye forslag fra Det europæiske Patentkontor fjerner undtagelsen af computerprogrammer. Hvis dette vedtages vil den ophavsretligt baserede beskyttelse blive suppleret med den langt mere vidtgående beskyttelse via patenter.

Amazons trivielle patent

Amazons 1-klik-patent er et af mange eksempler på softwarepatenter uddelt i USA uden at der foreligger en teknologisk innovation: Internetboghandlen Amazon har i USA fået patent på bestilling af en bog via Internettet med et enkelt muse-klik [2]. I oktober 99 anlagde Amazon sag mod sin største konkurrent, Barnes & Noble, for at krænke dets patentrettigheder, og en domstol i Seattle gav i december Amazon medhold i en foreløbig kendelse. Barnes & Noble måtte så ændre deres websider med online-handel, og nu er det blevet mere besværligt for kunder at bestille en bog. Der var vel at mærke ikke tale om at Barnes & Noble havde kopieret Amazons programmel; Barnes & Noble havde selv udviklet et nyt program. Amazons 1-klik-teknik er imidlertid en fuldstændig indlysende anvendelse af en teknik (baseret på de såkaldte cookies), som var både kendt og vidt udbredt i '97 da Amazon indgav patentansøgning. Kort fortalt er en cookie et stykke ekstra information, der sendes mellem webbrowser (kunde) og webserver (boghandel). I dette tilfælde bruges cookien til at identificere kunden overfor Amazon, som derefter genbruger kundens kreditkortsdata, i stedet for at spørge om dem igen.

Reelle og påståede opfindelser

Hvis en patenteret idé - som i Amazon-tilfældet - er indlysende, er der ikke tale om en langvarig indsats med forskning og udvikling, som der er en risiko for firmaet ved at investere i, og en samfundsinteresse i at beskytte med et patent. Ydermere er patentering af allerede kendte idéer fuldstændig urimelig - det har karakter af tyveri fra de virkelige ophavsmænd og -kvinder.

I dette tilfælde er det tyveri fra samfundet, fordi opfinderne af de basale webteknologier, der ligger til grund for Amazons 1-klik-teknik, har stillet dem gratis til rådighed for alle.

Dette argument går på patent-praksis i USA, ikke selve idéen om software-patenter, men praksis viser, at de amerikanske patentmyndigheder ikke håndhæver patentlovens krav om at opfindelser skal være originale og ikke-indlysende.

En del af forklaringen kan være, at med over 20.000 softwarepatenter udstedt alene i '99 har de amerikanske patentmyndigheder ikke den fornødne kapacitet til kritisk efterprøvning.

På grund af den ekstremt hurtige teknologiske udvikling på IT-området er det tvivlsomt om Det europæiske Patentkontor vil kunne gøre det bedre end i USA. Og uanset om man kan etablere en mere kritisk praksis i Europa kan man frygte at konsekvensen af en harmonisering med USA vil være at junglen af trivielle, amerikanske softwarepatenter får europæisk gyldighed.

Den lette adgang til softwarepatenter i USA forsvares af og til med et argument om at man kan anfægte et patent og få det omstødt - før man selv bliver anklaget for at krænke patentet. I Danmark kan man ved at betale nogle tusinde kroner få patentmyndighederne til at revurdere et patent, hvis man f.eks. mener man kan dokumentere at den patenterede idé ikke er original. I vore øjne er det imidlertid uacceptabelt med en retstilstand, hvor man som firma eller privatperson er nødt til at rejse sager ved domstole eller patentmyndigheder for at omstøde uretmæssige patenter.

Hvem betaler for patentholders monopol?

Der er altid en anden part, der betaler for patenthavernes fordel, også selv om opfindelsen er reel. Hvis patenthaveren har indtægter på at licensere en patenteret idé, ja så har andre firmaer den tilsvarende udgift. Udgifterne vil især skulle afholdes af mindre og nystartede softwarefirmaer, som ikke selv har patenter, hvorimod mange større og etablerede softwarefirmaer i USA indgår aftaler om fri brug af hinandens patenter. Softwarepatenter vil således have tendens til at beskytte etablerede firmaer mod nystartede firmaer, og dermed begrænse konkurrencen og tilgangen af firmaer med nye ideer.

Patenthaveren kan også vælge at forhindre konkurrenterne i at bruge idéen overhovedet, hvorved innovationsprocessen svækkes i den forstand at teknologien ikke spredes. Det er særlig vigtigt med netværks-relateret teknologi, hvor spredning af en metode er en forudsætning for at teknologien realiseres.

Tim Berners-Lee, der opfandt selve Word Wide Web, det vil sige den grafiske, klikbare grænseflade til Internettet, er blevet spurgt om det ikke har ærgret ham, at han ikke har kunnet „score kassen" (cash in) på web'en. Han svarede: „Hvis teknologien havde været privatejet (proprietary), var den aldrig gået i luften. Beslutningen om at gøre web'en til et åbent system var nødvendig for at den kunne blive universel" [3]. En af de faktorer, der har skabt web'ens udvikling, er at browsere og servere har kunnet fås gratis. Hvis alle brugere af Berners-Lees idéer skulle betale licens, havde web'en simpelthen ikke eksisteret i dag. Unægtelig det modsatte af at fremme innovation. Uden web intet forretningsgrundlag for bl.a. Amazon.

Teknologisk forspring uden patenter

Der er stort set ingen samfundsmæssig fordel ved frigivelsen af software-patenter efter den 20-årige gyldighedsperiode. I IT-branchen vil en 20 år gammel teknologi normalt være forældet. Selv en periode på to-tre år er innovationsmæssigt meget lang tid. Ja, alene den tid det tager at udvikle en efterligning udgør en slags beskyttelse af innovative firmaer.

En central konkurrence-parameter for IT-virksomheder er intellektuel kapital i form af teknologisk og markedsmæssigt kendskab. Hvis et firma har intellektuel kapital til selv at udvikle de teknologisk og markedsmæssigt rigtige ideer, og et andet firma kun evner at efterligne, så vil sidstnævnte som oftest alligevel være håbløst bagefter. Man skal huske at ophavsretten tvinger efterligneren til at udvikle sin egen version af programmet, hvilket kan være uhyre tidskrævende. De almindelige regler om forretningshemmeligheder, industrispionage osv. gælder selvfølgelig også på IT-området.

Open Source er innovativt

Softwarepatenter kan blive et særdeles stort problem for udviklere og brugere af Open Source-software – eksempelvis operativsystemet Linux, som vores forening SSLUG er en brugergruppe for.

Linux og andre Open Source-programmer kan frit kopieres, ligesom enhver har ret til at foretage ændringer i kildeteksten, forudsat at de stiller ændringerne til rådighed for andre på de samme betingelser. Linux er hovedsagelig skabt af privatpersoner, der har arbejdet gratis i deres fritid.

Udviklerne af Open Source-programmer har reelt ikke mulighed for at sikre sig, at programmerne ikke krænker et eller andet patent. Det vil betyde, at den, der anvender programmerne i erhvervsmæssigt regi, uforvarende kan krænke et patent.

Hvis en patentindehaver får medhold i at et patent er krænket af et program, kan det ikke anvendes erhvervsmæssigt uden tilladelse fra patentindehaver, hvilket kan kræve at der betales en licens til vedkommende. Og da Open Source-programmer er gratis, skabes der ikke salgsindtægter til at dække betalingen af en sådan licens. Der er heller ikke penge til at rejse sager for at omstøde et patent.

I dag er Open Source-bevægelsen en af de rigeste kilder til innovation indenfor IT, og vi har allerede set mange succeshistorier. For eksempel anvender over 60% af alle webservere på Internettet Open Source-webserveren Apache [4].

Lovgivning på forkant med udviklingen

SSLUG ser det som afgørende, at de europæiske lande ikke indfører softwarepatenter. Undtagelsen af computerprogrammer i europæisk patentlovgivning skal fastholdes. De europæiske lande bør arbejde for at USA og Japan overtager den nugældende europæiske holdning på patentområdet. Endelig bør man i Europa stoppe den seneste tendens til at uddele softwarepatenter på trods af den stadig gældende undtagelse. En fastholdelse af undtagelsen vil give den størst mulige valgfrihed for virksomheder og private forbrugere til at vælge det bedste og billigste software til den givne opgave - herunder også til at vælge gratis open source software.

Det eneste helt sikre ved at følge i USAs fodspor synes at være udgifterne til selve patentsystemet: ansøgninger, retsager, skattekroner til patentmyndighederne.

Ovenikøbet er de hidtidige retsager om softwarepatenter i USA måske kun forpostfægtninger, eftersom de fleste patenter endnu ikke er søgt håndhævet. Indehaverne af de tusindvis af amerikanske softwarepatenter opretholder en slags væbnet neutralitet, som kan udvikle sig til et ragnarok af vanvittige, ressourcekrævende og lammende retsager.

Amerikanerne løber en alvorlig risiko med tilladelsen af softwarepatenter - vi behøver ikke gøre det i Danmark og det øvrige Europa.

Artiklen er skrevet af Anne Østergaard, Carsten Svaneborg, Hanne Munkholm, Keld Jørn Simonsen og Niels Jørgensen fra Skåne Sjælland Linux User Group (SSLUG).

Kilder:

[1] The European Patent Office: Basic proposal for the revision of the European Patent Convention. http://www.freepatents.org/law/epc-nolimits.pdf

[2] http://www.oreilly.com/ask_tim/amazon_patent.html

[3] Tim Berners-Lee FAQ (Frequently Asked Questions). http://www.w3.org/People/Berners-Lee/FAQ.html.

[4] Ifølge en undersøgelse foretaget af firmaet Netcraft og offentliggjort på http://www.netcraft.com/survey/.

[5] EPO-forslaget s. 37.

Yderligere referencer

Kronik til Information om Open Source 21/1-2000: http://www.sslug.dk/artikler/informationkronik.html.

Software-patenter - Skal Europa følge efter USA? af SSLUGs bestyrelse: http://www.sslug.dk/bestyrelsen/swpat.html

James Bessen og Eric Maskin, Massachusetts Institute of Technology (MIT): Sequential Innovation, Patents, and Innovation: http://www.researchoninnovation.org/patent.pdf.

En masse information om softwarepatenter: http://www.freepatents.org.

Om Open Source generelt. http://release1.edventure.com/Issues/1198.pdf.

SEMINAR:

Kan UNIX vinde markedsandele på Mainframe-området?

Deltag i seminaret den 12. oktober om UNIX og mainframes. Mainframen overlevede truslen fra distribuerede systemer på trods af mange profetier om det modsatte, men er der ved at ske et skred nu, hvor flere store finansielle institutioner og store virksomheder vælger UNIX til deres vigtigste applikationer? De store leverandører af UNIX servere, er i løbet af de sidste to år blevet anvendt inden for områder og til opgaver, der tidligere udelukkende var forbeholdt mainframes. En direkte sammenligning viser, at UNIX servere tilbyder mere computerkraft til færre penge. Mange danske virksomheder vil kunne spare penge ved at anvende UNIX servere, ikke nødvendigvis som erstatning for mainframes, men som et godt og stabilt supplement til nye områder, som f.eks. Internet relaterede opgaver. Denne tendens er en vigtig faktor når firmaets IT-strategi lægges fast. UNIX er ikke længere et styresystem med børnesygdomme og sikkerhedsproblemer, men et voksent og/eller sikkert alternativ til mainframes.

Hvem bør deltage? Beslutningstagere, IT-chefer, økonomichefer og driftsansvarlige.

Læs mere om arrangementet og tilmeld dig på www.dkuug.dk eller kontakt DKUUG's sekretariat på tlf. 39 17 99 44.

Tilmelding kan også ske via www.ehuset.com, faxes på 44572001 eller sendes til eHuset A/S, Vesterlundvej 14, 2730 Herlev.

Linux som Firewall

af Hanne Munkholm
<hanne@aub.dk>

og Peter Toft
<pto@sslug.dk>

De forskellige versioner af Linux kernen bruger forskellige pakkefiltreringssystemer. Dette er ikke for at gøre livet besværligt for os andre, men fordi firewalling er et område, som udvikler sig med stor hast. De nyere programmer er mere fleksible, og det er gjort nemmere at sætte sine firewall regler (policies) op.

I denne artikel er kerne 2.2 i fokus. Vi vil ikke beskæftige os med kerne 2.0, men vi vil kaste et blik på næste generation af firewall værktøjer til Linux kerne 2.3, som endnu er på et udviklingsstadie, og som kommer til at hedde 2.4, når den er færdig.

Desuden vil vi kigge nærmere på proxy opsætning med squid, samt på IP masquerading i kerne 2.2, og snuse lidt til Network Address Translation i kerne 2.3 (2.4).

Pakkefiltrering

Ipchains - kerne 2.2

I kerne 2.2, som er den Linux-kerne, man hovedsagelig anvender i dag, hedder firewall værktøjet ipchains. Ipchains er en omskrivning af ipfwadm, som man anvendte i kerne 2.0.

En mere grundig introduktion til ipchains kan findes i IPCHAINS-HOWTO http://sunsite.auc.dk/ldp/HOWTO/IPCHAINS-HOWTO.html.

Er ipchains installeret på min maskine?

Ipchains er en del af selve Linux kernen. Man skal altså bruge en kerne, som ipchains er kompileret ind i. Undersøg om kernen er oversat med support for ipchains:

```
[root@sherwood /root]# ls /proc/net/ip_fwchains
```

Hvis filen eksisterer, er ipchains understøttet i den kerne, der er installeret. Hvis den ikke gør, skal der oversættes (kompileres) en ny kerne. Vi vil ikke her komme ind på, hvordan man oversætter en ny kerne. Der findes en del beskrivelser på Internet om dette, f.eks. http://www.sslug.dk/linuxbog/bog/kernelcompile.html. Man skal under konfiguration af den nye kerne vælge "Network Firewalls" og "IP firewalling" under "network options". Hvis maskinen også skal bruge IP masquerading, så vælg dette med ind i kernen med det samme - se afsnittet om Masquerading med ipchains.

Opsætning af rules

Pakkefiltreringen styres af en række rules (regler), som systemadministratoren selv sætter op. Disse regler styrer, hvilke pakker, der slippes ind og ud, baseret på afsender, modtager og typeinformation i pakke-headeren.

En rule kan f.eks. være:

```
[root@sherwood /root]# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

som betyder, at ICMP pakker, der kommer fra 127.0.0.1 (loopback interfacet) skal afvises. Dette er nok ikke en god regel at køre med til daglig, da det betyder, at man ikke kan pinge sin localhost, men den er glimrende som eksempel.

Vi kommer nærmere ind på de forskellige dele af den ovenstående kommando i resten af dette afsnit.

Chains Rules puttes ind i kæder (chains). Pakkerne passerer igennem disse kæder, og checkes imod hver rule, der er i kæden. Der er tre indbyggede kæder:

- **input**: Input-kæden bruges på alt, hvad der kommer ind til maskinen udefra, lige når det kommer ind. Det kan være pakker, der kommer på netkortet (ethx), modemet (pppx) eller loopback (lo) interfacet.
- **output**: Output-kæden bruges lige inden, en pakke forlader maskinen.
- **forward**: Forward-kæden benyttes på pakker, der kommer ind, som har en anden maskine som destination. Dvs. pakker der skal videresendes.

I eksemplet ovenfor er det input kæden, vi arbejder med.

Chain relaterede options til ipchains:

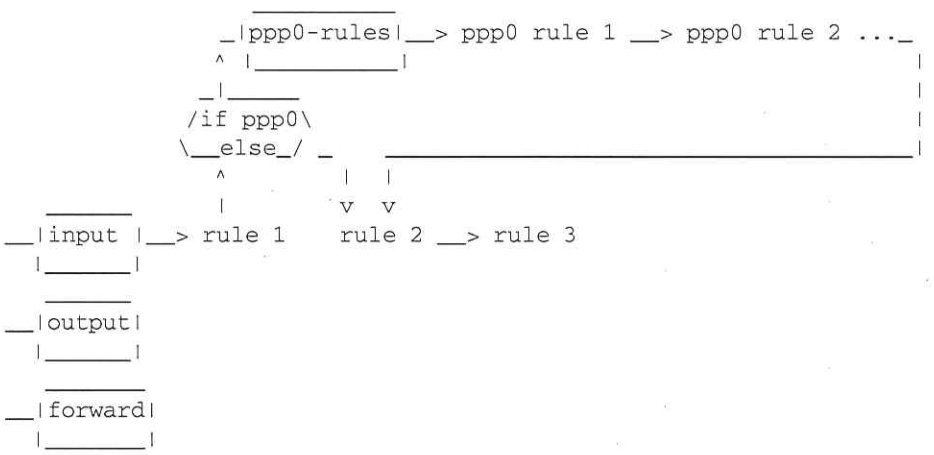
- -N Opret ny kæde
- -X Slet tom kæde
- -F Flush (tøm) kæde
- -L Vis de rules, der er i en kæde
- -Z Nulstil (zero) pakke- og byte-tællere på alle rules i en kæde

Pakke- og byte-tælleren på en rule tæller hvor mange pakker hhv bytes, der har været forbi, som har "opfyldt" den rule.

Private chains: Ud over de tre indbyggede kæder kan man lave sine egne. Det kan være smart, da det bl.a gør ens firewall-regler mere overskuelige.

Hvis man f.eks. har et system, hvor ppp0 (modemet) er det interface, der er forbundet til internet, vil man nok gerne sætte en masse firewall-regler op i input-kæden for ppp0-interfacet. Disse kan samles i en hjemmelavet kæde.

Lad os kalde vores hjemmelavede kæde for ppp0-rules. Vi sætter så en regel op i input-kæden, der kalder ppp0-rules kæden, hvis en pakke kommer ind på ppp0. På den måde undgår man dels, at input-kæden bliver uoverskuelig pga. alt for mange regler, dels opnår man, at reglerne for ppp0 kun gennemløbes, hvis pakken rent faktisk er kommet ind på ppp0.



En kæde oprettes med kommandoen:

```
[root@sherwood /root]# ipchains -N ppp0-rules
```

og kan slettes igen med

```
[root@sherwood /root]# ipchains -X ppp0-rules.
```

Man kan kun slette en tom kæde. En kæde kan flushes (tømmes) med en "-F" option til ipchains kommandoen. Det betyder, at alle regler i kæden slettes.

Targets: En rule har et target (mål), som fortæller hvad der skal ske med pakken. Logikken i navnet "target" kan diskuteres, men sådan hedder det. Pakkens skæbne kan være

- ACCEPT betyder, at pakken accepteres.
- REJECT betyder, at pakken afvises, og at afsenderen orienteres om dette.
- DENY betyder at pakken afvises, uden at afsenderen får det at vide.
- MASQ er relateret til masquerading og bruges kun i forward kæden. Se om masquerading senere i artiklen.
- REDIRECT betyder, at pakken sendes til en lokal port i stedet for at gå videre til destinationsmaskinen. REDIRECT kan kun bruges i input kæden og kun med pakker til UPD eller TCP protokollerne.
- RETURN betyder, at man straks springer til enden af kæden.

I eksemplet

```
[root@sherwood /root]# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

er target sat til DENY. Option "-j", som bruges til at definere target, kan huskes ved at tænke "jump to",

hvilket måske siger mere om betydningen end "target" gør.

Default policy er det, der sker, hvis en pakke når enden af en af de tre standardkæder uden at være blevet afvist eller accepteret. Default policy kan være et af de fire første targets: ACCEPT, REJECT, DENY eller MASQ - MASQ bruges dog stadig kun i forward kæden.

Default policy for en kæde sættes med en "-P" options til ipchains:

```
[root@sherwood /root]# ipchains -P input ACCEPT
```

En rule behøver ikke at have et target. En pakke, som matcher en rule, der ikke har noget target, går bare videre til næste rule. En rule uden target kan f.eks. bruges til at tælle antallet af pakker, der opfylder bestemte kriterier.

Rules

Rule relaterede options til ipchains:

- -A Tilføj (append) rule
- -D Slet (delete) rule
- -R Erstat (replace) rule
- -I Indsæt (insert) rule

Lad os beholde vores rule-eksempel fra ovenfor.

```
[root@sherwood /root]# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

"-A" betyder at vi tilføjer en rule (append). En regel kan indeholde forskellige betingelser, som gør, at pakkerne matcher eller ikke matcher.

Betingelser

- -p Protokol
- -s Source IP adresse
- -d Destination IP adresse
- -i Interface
- -y SYN pakker (kun TCP)
- -f Fragmenter af en pakke

Desuden kan der ved TCP eller UPD specificeres en port eller et port interval efter source eller destination IP adresse. Ved ICMP kan der specificeres en ICMP type og kode efter source eller destination adressen.

En oversigt over protokollerne findes i /etc/protocols. ICMP typerne kan ses med kommandoen

```
[root@sherwood /root]# ipchains -h icmp | more
```

Lad os prøve at lave nogle rules i praksis. Vi ser igen på vores eksempel fra tidligere:

```
[root@sherwood /root]# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
```

Den betyder, at der ikke må komme ICMP pakker ind på lo (loopback interfacet), og derfor kan man ikke kan pinge sin localhost:

```
[root@sherwood /root]# ping -c 1 localhost
PING localhost (127.0.0.1): 56 data bytes
```

Før eller siden vil man få timeout på sin ping kommando. For at fjerne den nye rule igen skrives

```
[root@sherwood /root]# ipchains -D input -s 127.0.0.1 -p icmp -j DENY
```

og prøv at pinge localhost igen:

```
[root@sherwood /root]# ping -c 1 localhost
PING localhost (127.0.0.1): 56 data bytes
64 bytes from 127.0.0.1: icmp_seq=0 ttl=255 time=0.1 ms

— localhost ping statistics —
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.1/0.1/0.1 ms
```

En rule kan også fjernes ved blot at angive dens nummer i stedet for hele i dens indhold. Da vores rule fra før havde nummer 1, kunne den være fjernet med

```
[root@sherwood /root]# ipchains -D input 1
```

Rules får nummer i den rækkefølge, man laver dem: Næste rule vi laver, får nummer 2.

```
[root@sherwood /root]# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
[root@sherwood /root]# ipchains -A input -s 192.168.1.1
```

Den første kommando danner rule nummer et, den næste rule nummer 2. Slettes nummer et, rykker nummer 2 frem og bliver nummer 1.

Man kan dog indsætte rules inde i en kæde med -I optionen. Dette gøres ved at angive det ønskede rule nummer:

```
[root@sherwood /root]# ipchains -A input -s 127.0.0.1 -p icmp -j DENY
[root@sherwood /root]# ipchains -I input 1 -s 192.168.1.1
```

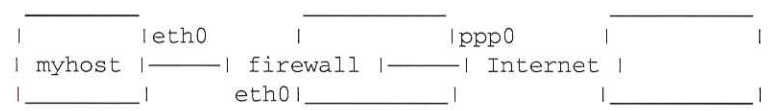
Her vil den sidste rule blive nummer 1, pga -I optionen og ettallet, i der er angivet efter kædenavnet (input).

Eksempel: hjemmebruger

Nu har vi lært lidt om at oprette og slette rules. Lad os kigge på hvilken opsætning af kæder og regler, man kunne ønske sig i virkeligheden.

Det eksempel, vi skal se på, er en hjemmebruger, der har en eller flere computere og en opkobling til Internet. Vores hjemmebruger kører sin egen webserver. Han kører også en auth server til validering af hvilken bruger, der kører en bestemt service. Desuden bruger han en række klienter, der kræver forbindelser (connections): ftp, DNS og Real Audio.

Figuren viser et netværk der består af to computere, en på det lokale LAN og en, der skal fungere som firewall. Der kunne selvfølgelig godt være mange flere maskiner indenfor firewall'en. Men selvom man kun har én computer, kan det godt give mening at installere ipchains på den alligevel, for at sortere i, hvilke pakker der skal slippe igennem til maskinens egne lokale services.



Først skal vi sætte vores default policy op. I input og output kæden kan man vælge ACCEPT, DENY eller REJECT. Vælger man ACCEPT, skal man sætte rules op for at spærre for alt det, der ikke må komme igennem. Vælger man DENY eller REJECT, skal man sætte rules op for alt det, der gerne må komme igennem. Det er nemt at regne ud, at det sidste er det mest sikre. Især på input kæden kan det frarådes at køre ACCEPT som default policy, med mindre man er helt sikker på, at man ved, hvad man gør, eller sikkerheden ikke er så vigtig i det pågældende netværk I forward kæden kan man desuden vælge MASQ, dette kommer vi ind på i afsnittet om masquerading.

Vores hjemmebruger bag firewall'en ønsker adgang til Internet igennem firewall'en. Derimod vil han gerne skærme sit lokale netværk imod de indgående forbindelser fra fremmede maskiner.

Først sætter vi "default policies" op.

```
ipchains -P input DENY
ipchains -P output ACCEPT
ipchains -P forward DENY
```

Rules for vores hjemmebruger

Tillad alt på loopback interfacet

```
ipchains -A input -p all -j ACCEPT -i lo
```

Tillad alt på LAN

```
ipchains -A input -p all -j ACCEPT -i eth0
```

Nu er der kun indgående trafik fra Internet (ppp0) tilbage. Vi starter med TCP pakker. Tillad alt, hvad der *ikke* forsøger at oprette en forbindelse, det vil sige alt, hvad der ikke har SYN bitten sat:

```
ipchains -A input -p tcp -j ACCEPT \! -y
```

Tillad forbindelser til vores http, https og auth server:

```
ipchains -A input -p tcp -j ACCEPT -s 0/0 -d 0/0 http -y
ipchains -A input -p tcp -j ACCEPT -s 0/0 -d 0/0 https -y
ipchains -A input -p tcp -j ACCEPT -s 0/0 -d 0/0 auth -y
```

Tillad ftp dataforbindelser (når vores hjemmebruger vil lave ftp downloads og ftp dir kommandoer på ftp-servere ude i verden)

```
ipchains -A input -p tcp -j ACCEPT -s 0/0 ftp-data -d 0/0 56000:65096 -y
```

UDP pakker:
Tillad DNS replies:

```
ipchains -A input -p udp -j ACCEPT -s 0/0 domain -d 0/0 56000:65096
```

Tillad Real Audio / Real video i bedste kvalitet:

```
ipchains -A input -p udp -j ACCEPT -d 0/0 32768:37769
```

ICMP
Tillad ikke redirect (kan principielt bruges til at manipulere routing)

```
ipchains -A input -p icmp -s 0/0 redirect -j DENY –log
ipchains -A input -p icmp -s 0/0 timestamp-request -j DENY –log
ipchains -A input -p icmp -s 0/0 address-mask-request -j DENY –log
ipchains -A input -p icmp -j ACCEPT
```

Log resten

```
ipchains -A input –log
```

Alt, hvad der kommer på inputkæden, og som ikke er blevet accepteret af en af de andre regler, matcher denne sidste regel. Reglen gør, at der logges information om disse pakker i kerne-logfilen (ofte /var/log/messages).
En log entry kan f.eks. se sådan ud:

```
input - ppp0 PROTO=17 4.3.2.1:4000 1.2.3.4:1872 L=53 S=0x00
I=60456 F=4000 T=240 (#14)
```

hvor **1.2.3.4** er vores hjemmebrugers egen ip-adresse på ppp0. Man kan se, at det er kommet ind på **input**kæden, interface **ppp0**. **PROTO=17** betyder, at det er udp (se /etc/protocols). **4.3.2.1:4000** er afsender-adressen, port 4000. Pakken er forsøgt sendt til port **1872** hos vores bruger. **L=53** betyder, at pakken var 53 bytes lang. **S** betyder type of service. **I** betyder IP ID, **F** betyder fragment offset, og **T** betyder Time to live. **#14** betyder, at det var vores rule nummer 14, der loggede pakken. Se ipchains HOWTO'en for yderligere detaljer.
Hvis ftp ikke virker med det angivne portinterval (56000:65096), er det fordi, maskinen er sat op til at bruge nogle andre porte til at oprette forbindelser på. Det kan systemadministratoren selv sætte med kommandoen

```
echo "56000 60999" >/proc/sys/net/ipv4/ip_local_port_range
```

Se i øvrigt et glimerende eksempel på ipchains opsætning på http://www.sslug.dk/sikkerhed/ipchains.html

At aktivere sine rules ved opstart

Ipchains styres af en række rules. Selve pakkefiltreringen sker i kernen, så det eneste, man skal gøre, er, at fortælle kernen om sine ipchain rules.

Da rules gemmes i kernen, går de imidlertid tabt ved reboot. Derfor er det en god ide at gemme dem i en fil. Man kan så lave et startup script, som læser filen og sætter kernen op til at bruge de rules ved genstart af maskinen. Ipchains rules gemmes i en fil med følgende kommando:

```
[root@sherwood root]# ipchains-save > /etc/ipchains.rules
```

Det følgende startup script kan hvert fald bruges med Red Hat, SuSE og Debian:

```
#!/bin/sh
#ipchains startup script
#To be run before starting network on startup
#and to be shut down after network on shutdown

# If no rules, do nothing.
[ -f /etc/ipchains.rules ] || exit 0

case "$1" in
    start)
        echo -n "Turning on packet filtering:"
        /sbin/ipchains-restore < /etc/ipchains.rules || exit 1
        echo 1 > /proc/sys/net/ipv4/ip_forward
        echo "."
        ;;
    stop)
        echo -n "Turning off packet filtering:"
        echo 0 > /proc/sys/net/ipv4/ip_forward
        /sbin/ipchains -X
        /sbin/ipchains -F
        /sbin/ipchains -P input ACCEPT
        /sbin/ipchains -P output ACCEPT
        /sbin/ipchains -P forward ACCEPT
        echo "."
        ;;
    *)
        echo "Usage: /etc/init.d/packetfilter {start|stop}"
        exit 1
        ;;
esac
```

Lav et symlink til det i /etc/rc.d/rc*N*.d (N=runlevel) directory'erne (Red Hat) eller /etc/rc*N*.d (Debian/SuSE), så det bliver startet op før netværket. Hvis netværket f.eks. startes op med et symlink, der hedder S10network, skal ipchains symlinket hedde et lavere tal. F.eks. S9ipchainsrules. Så er ipchains altid kørende før netværket startes op. Ligeledes bør netværket lukkes ned før ipchains lukkes ned. Selvom der er grænser for, hvor meget skade en cracker kan nå at gøre på systemet, før det når at lukke ned. Det skal specielt bemærkes, at hvis man med det viste script anvender stop-kommandoen, efterlader man sit system i en særdeles usikker tilstand med alt nettraffik tilladt. Lad være med at gøre det, mens systemet er i drift. Man lukker simpelthen ned for firewall'en.
Se i øvrigt en udemærket HOWTO på http://sunsite.auc.dk/ldp/HOWTO/IPCHAINS-HOWTO.html.

Iptables (Netfilter) - kerne 2.3 (2.4)

Kerne 2.3-serien er en ustabil udviklingsgren af Linux-kernen, som forhåbentlig snart bliver til kerne 2.4, den næste stabile kerne-serie. Vi vil ikke gå så meget i dybden med kerne 2.3 her, da vi antager, at denne artikels primære målgruppe ikke bruger en udviklingskerne, og da tingene kan nå at være lavet om igen, inden den endelige 2.4 kerne frigives. Afsnittene om kerne 2.3 skal derfor blot forstås som en forsmag på, hvad der venter i næste version. Vi vil ikke komme med eksempler på opsætning af iptables.

I kerne 2.3 (2.4) er firewalling systemet igen lavet om. Kommandoen, der før hed ipchains, hedder nu iptables. Systemet består af to dele: Pakkefiltrering og NAT (**N**etwork **A**ddress **T**ranslation). NAT, som er lidt mere avanceret end masquerading, vender vi tilbage til i et senere afsnit. Hele systemet hedder netfilter, og er grundlæggende et framework i kernen, som iptables bygger på.

Der er nogle grundlæggende ændringer i designet, men kommandoerne og navnene ligner næsten sig selv fra ipchains. Man skal dog ikke lade sig narre - iptables kan meget mere.

Hvad skal jeg bruge?

For at komme igang med netfilter skal man bruge en 2.3 kerne med netfilter kompileret ind samt user space værktøjet iptables. Iptables kan downloades fra Netfilter hjemmesiden: http://netfilter.kernelnotes.org/ .

Kommer man fra ipchains er det ikke svært at følge med i kommandoerne til iptables - der er dog ændringer.

Chains

Ligesom i ipchains er der i iptables 3 indbyggede kæder: INPUT, OUTPUT og FORWARD. Chain relaterede options til iptables er uændret fra ipchains

Targets

Der er lavet lidt om i targets:

- DROP betyder, at pakken smides væk
- ACCEPT betyder, at pakken accepteres
- RETURN betyder, "gå til enden af kæden"
- QUEUE betyder, at pakken skal sættes i kø til user mode processering

Når vi sammenligner med ipchains, kan vi se, at REJECT er forsvundet. Den findes dog som modul (extension):

- REJECT
- LOG

er de to extensions, der standard følger med iptables. Det er muligt at skrive sine egne extension og derved lave andre "targets".

Desuden er MASQ og REDIRECT forsvundet - de hører til NAT, og deres funktionalitet er kommet over i NAT delen af iptables (se afsnittet om Network Address Translation med iptables).

Default policy

Default policy er som i ipchains

Rules

Rule relaterede options til iptables er som i ipchains

Betingelser

En regel kan indeholde forskellige betingelser, som gør, at pakkerne matcher eller ikke matcher.

Betingelser

- -p Protokol
- -s Source IP adresse
- -d Destination IP adresse
- -i Input interface
- -o Output interface
- -f Fragmenter af en pakke

Vi ser at **-i** nu ikke betyder interface, men input interface, og **-o** er kommet til for output interface. Desuden er **-y** (SYN) forsvundet, men bare rolig, den kommer om lidt.

Ud over standardbetingelserne er der en del protokolspecifikke udvidelser:

TCP

- —tcp-flags (SYN,ACK,FIN,RST,URG,PSH)
- —syn (SYN,RST,ACK SYN)
- —source-port
- —destination-port
- —tcp-option

UDP

- —source-port
- —destination-port

ICMP

- —icmp-type

Desuden er der skrevet en række extensions, som følger med iptables:

Andre extensions

- mac
  - —mac-source
- limit
  - —limit
  - —limit-burst
- owner
  - —uid-owner userid
  - —uid-owner groupid
  - —pid-owner processid
  - —sid-owner processid
- unclean
- state
  - NEW
  - ESTABLISHED
  - RELATED
  - INVALID

Extension modulerne loades med **-m**, f.eks.

```
# iptables -A INPUT -m mac —mac-source 192.168.0.1
```

Det er også muligt at skrive extensions selv.

State

State er værd at kigge lidt nærmere på, da det bruges til at checke på connection tracking. Det vil sige, at man kan sortere på, om en pakke er en del af en eksisterende forbindelse (ESTABLISHED), om den vil etablere en ny forbindelse (NEW), om den er relateret til en eksisterende forbindelse, f.eks. en ICMP fejl (RELATED), eller om den ikke kan genkendes (INVALID).

BEMÆRK: iptable er på udviklingsstadiet endnu, og man kan altså ikke regne med, at det er stabilt.

For mere information om iptables og netfilter anbefaler vi at læse Linux 2.4 Packet Filtering HOWTO: http://netfilter.kernelnotes.org/unreliable-guides/packet-filtering-HOWTO.html

Katedralen og basaren

af Eric S. Raymond
esr@thyrsus.com

Dansk oversættelse

Ole Michaelsen
omic@fys.ku.dk

Jesper Laisen
post@laisen.dk

4. Frigiv tidligt, frigiv hyppigt

Tidlige og hyppige frigivelser er en vigtig del af Linux' udviklingsmodel. De fleste udviklere (mig selv inklusive) troede, at det var en dårlig politik for større projekter, da tidlige versioner per definition altid er fulde af fejl, og da du ikke ønsker at opbruge dine brugeres tålmodighed.

Denne tro var forstærket af den generelle tilslutning til katedralen som udviklingsstil. Hvis den overvældende målsætning var, at brugerne skulle se så få fejl som muligt, så burde du kun frigive en version hver sjette måned (eller sjældnere) og arbejde som en hest på at finde fejl mellem frigivelserne. Emacs' C-kerne blev udviklet på denne måde. Lisp-biblioteket blev det faktisk ikke — fordi der var aktive Lisp-arkiver uden for FSF's kontrol, hvor du kunne finde nye versioner af kode under udvikling uafhængig af Emacs' frivgivelsesmønster **[QR]**.

Det vigtigste af disse, the Ohio State Elisp Archive, foregreb ånden og mange af mulighederne i nutidens store Linux-arkiver. Men få af os tænkte særlig meget over, hvad vi gjorde, eller over hvad eksistensen af det arkiv antydede af problemer med FSF's udviklingsmodel med katedral-bygning. Jeg gjorde et seriøst forsøg i 1992 på formelt at få en stor del af Ohio-koden ind i det officielle Emacs Lisp-bibliotek. Jeg løb ind i politiske problemer og havde ingen synderlig succes.

Men et år senere, da Linux blev bredt kendt, blev det klart, at noget anderledes og meget sundere foregik der. Linus' åbne udviklingspolitik var modsætningen til katedral-bygningen. Sunsite- og tsx-11-arkiverne bugnede og adskillige distributioner kom frem. Og alt dette var drevet af en uhørt frekvens af frigivelse af nye Linux-kerner.

Linus behandlede sine brugere som medudviklere på den mest effektive måde:

7. Frigiv tidligt. Frigiv hyppigt. Og lyt til dine kunder.

Linus' fornyelse bestod ikke så meget i, at han ekspederede hurtige frigivelser, hvor han indkorporerede masser af bruger feedback (noget lignende havde været en del af Unix-traditionen længe), men at han skalerede princippet op til et intensitetsniveau, der matchede kompleksiteten af det, som han udviklede. Dengang (omkring 1991) var det ikke ualmindeligt, at han frigav en kerne hyppigere end en gang om dagen! Det virkede, fordi han dyrkede sin base af medudviklere og brugte Internet til samarbejde meget mere end nogen anden. Men hvordan virkede det? Og var det noget, jeg kunne gøre efter, eller skyldtes det Linus Torvalds' unikke geni? Det troede jeg ikke. Jeg medgiver gerne, at Linus er en pokkers god hacker (hvor mange af os kunne konstruere en fuldstændig, produktionsklar kerne til et Operativsystem fra bunden?) Men Linux repræsenterede ikke nogen frygtindgydende begrebsmæssigt spring fremad. Linus er ikke (eller i det mindste ikke endnu) et innovativt design-geni, ligesom Richard Stallman eller James Gosling (NeWS og Java) er det. Snarere synes Linus at være et geni til udvikling, med en sjette sans til at undgå fejl, udvikling i forkert retning og sand evne til at finde vejen mellem A og B med den minimale indsats. Hele Linux' opbygning stråler i sandhed af den kvalitet, der afspejler Linus' i grunden konservative og forenklede udviklingsstil.

Så hvis hurtig frigivelse og fuldt brug af Internet som medie ikke var et tilfælde men integrerede dele af Linus' udviklings-genis forståelse for vejen med den minimale indsats, hvad var det så han maksimerede? Hvad var det han hev ud af maskineriet? Når det udtrykkes sådan, besvarer spørgsmålet sig selv. Linus holdt sine hackere/brugere konstant stimulerede og belønnede — stimulerede af udsigten til en del af æren til at tilfredsstille egoet og med udsigt til konstante (ja daglige) forbedringer af deres eget arbejde. Linus forsøgte direkte at maksimere antallet af mandetimer, der blev brugt til at finde fejl og til at udvikle, selv om det betød ustabil kode og udbrændte brugere, hvis en alvorlig fejl viste sig at være umedgørlig. Linus opførte sig som om, han troede på sådan noget som dette:

8. Med en stor nok base af betatestere og medudviklere, vil næsten ethvert problem blive hurtigt beskrevet og rettelsen være åbenlys for en eller anden.

Eller mindre formelt, ‚Med nok øjne er alle fejl banale'. Jeg kalder det: Linus' lov.

Min originale formulering var, at ethvert problem ‚vil være gennemskueligt for en eller anden'. Linus indvender, at den person, som først forstår og retter problemet, ikke nødvendigvis —

eller ikke engang som regel — er den, som først beskrevet det. ‚En eller anden finder problemet', siger han, ‚og en helt anden forstår det. Og jeg vil gerne citeres for at sige, at det at finde problemet er den største udfordring'. Men pointen er, at det har en tendens til at ske hurtigt.

Jeg tror, at her er den fundamentale forskel på katedral-stilen og basar-stilen. Når en katedralbygger programmerer, er fejl og udvikling vanskelige, lumske, grundlæggende fænomener. Det tager måneder med grundig overvejelse for de pligttro at tro på, at de har fundet alle fejlene. Derfor er der lange intervaller mellem frigivelser, og den uundgåelige skuffelse, når den længe ventede frigivelse ikke er perfekt. På den anden side er holdningen i basaren, at du går ud fra, at fejl i al almindelighed er banale — eller i det mindste bliver de hurtig banale, når de eksponeres til tusinder af villige medudviklere, der hamrer løs på enhver ny frigivelse. Følgelig frigiver du hyppigere for at få flere rettelser, og som en god sidegevinst er der mindre at miste, hvis et lejlighedsvis makværk slipper ud af døren.

Og det er det. Det er nok. Hvis ‚Linus' lov' er falsk, så burde ethvert system så komplekst som Linux-kernen, der er blevet hacket af så mange, på et eller andet tidspunkt være brudt sammen under vægten af uforudset dårligt sammenspil og ‚grundlæggende' fejl, der ikke er fundet. På den anden side — hvis den er sand, er det tilstrækkeligt til at forklare Linux' relative mangel på fejl og dens fortsatte høje oppetid, der strækker sig over måneder og år.

Og måske burde det ikke have været sådan en overraskelse. Sociologer opdagede for år siden, at den gennemsnitlige mening hos en gruppe af lige dygtige (eller lige ignorante) iagttagere, er en del mere pålidelig end en enkelt tilfældigt udvalgt af iagttagerne. De kaldte det ‚delfi-effekten'. Det virker som om, at det Linus har vist, er at det også gælder om det at finde fejl i et operativsystem — at delfi-effekten kan tæmme udviklingens kompleksitet selv ved et kompleksitetsniveau som med en operativsystem-kerne.

En særlig egenskab ved Linux-situationen, som klart hjælper sammen med delfi-effekten, er det faktum, at bidragsyderne til et hvilket som helst projekt er selv-valgte. En tidlig kritiker gjorde opmærksom på, at bidrag ikke modtages fra en tilfældig stikprøve, men fra folk, som er tilstrækkeligt interesseret til at bruge softwaren, lære hvordan den virker, forsøge at finde løsninger på de problemer, som de støder på, og faktisk producere en tilsyneladende fornuftig løsning. Det er overordentlig sandsynligt, at enhver der kommer igennem disse filtre, kan bidrage med noget brugbart.

Jeg står i gæld til Jeff Dutky (dutky@wam.umd.edu), som gjorde opmærksom på, at Linus' lov kan omformuleres til ‚Fejlfinding kan paralleliseres'. Jeff observerer, at selvom

fejlfinding kræver, at fejlfinderne kommunikerer med en eller anden koordinerende udvikler, så kræver det ikke nogen betydelig koordinering mellem fejlfinderne. Det falder ikke som bytte for den samme kvadratiske kompleksitet og de ledelsesudgifter, som gør det problematisk at øge antallet af udviklere.

I praksis synes det teoretiske tab af effektivitet, som skyldes duplikeringen af fejlfindernes arbejde, aldrig at være et problem i Linux-verdenen. En effekt af ‚politikken om at frigive tidligt og hyppigt' er at minimere sådan duplikering ved at hurtigt sprede rettelser, der stammer fra feedback **[JH]**.

Brooks (forfatteren af "The Mythical Man-Month) kom endda med en henkastet bemærkning om Jeff: ‚Den totale omkostning ved at vedligeholde et bredt anvendt program er typisk 40 procent eller mere af omkostningen ved at udvikle det. Det er overraskende, at de omkostninger er påvirket af antallet af brugere. Flere brugere finder flere fejl' (min fremhævning).

Flere brugere finder flere fejl, fordi flere brugere betyder flere måder at stresse programmet på. Den effekt er forstærket, når brugerne er medudviklere. Enhver af dem griber opgaven med at beskrive fejl an med et lidt anderledes begrebssæt og andre analytiske værktøjer, en anden vinkel på problemet. Delfi-effekten synes

at virke præcis på grund af denne variation. Specifikt med hensyn til at finde fejl har variationen også tendens til at reducere duplikeringen af indsatsen. At have flere betatestere reducerer således ikke nødvendigvis kompleksiteten af forhåndenværende grundlæggende fejl, men det øger sandsynligheden for, at en eller andens arbejdsmetode bliver stillet overfor problemet på en sådan måde, at fejlen bliver banal for denne person. Linus helgarderer også sit væddemål. Hvis der er alvorlige fejl, er en version af Linux-kernen nummereret på en sådan måde, at en potentiel bruger kan vælge enten at køre den sidste version, som er ‚stabil', eller afprøve det nyeste nye og risikere fejl for at få nye muligheder. Denne taktik er endnu ikke formelt efterlignet af de fleste Linux-hackere, men måske skulle den være det; faktum er, at det at begge valg er mulige, gør begge valg mere attraktive **[HBS]**.

5. Hvornår er en rose ikke en rose?

Da jeg havde studeret Linus' adfærd og formuleret en teori om, hvorfor den var succesfuld, tog jeg en bevidst beslutning om at afprøve hans teori på mit nye projekt (som jeg indrømmer er meget mindre komplekst og ambitiøst). Men det første jeg gjorde var at reorganisere og forenkle Popclient en hel del. Carl Harris' implementeringer var meget solide, men bar præg af en unødvendig kompleksitet, som er kendetegnende for mange C-programmører. Han behandlede koden som om, den var det centrale i sig selv, og datastrukturerne som understøttelse for koden. Resultatet var, at koden var smuk, men designet af datastrukturerne var tilfældigt og temmelig grimt (i det mindste efter den gamle LISP-hackers høje standarder).

Udover at forbedre koden og datastrukturerne havde jeg dog en anden grund til at omskrive koden. Det var at udvikle det til noget, som jeg fuldstændigt forstod. Det er ikke særligt morsomt at rette fejl i et program, som du ikke forstår fuldt ud.

I den første måned, cirka, fulgte jeg simpelthen betingelserne i Carls grundlæggende design. Den første alvorlige ændring, jeg lavede, var at tilføje understøttelse af IMAP. Jeg gjorde dette ved at reorganisere indretningen af protokollerne til en standard-driver og tre metode-tabeller (til POP2, POP3 og IMAP). Dette og de forrige ændringer illustrerer et generelt princip, som programmører gør klogt i at huske, specielt med hensyn til sprog som C, der ikke naturligt arbejder med dynamisk skrivning:

9. Smarte datastrukturer og dum kode virker meget bedre end det modsatte.

Brooks, kapitel 9: ‚Vis mig din [kode] skjul dine [data strukturer], og jeg vil fortsat være mystificeret. Vis mig dine [data strukturer], og jeg vil sandsynligvis ikke behøve din [kode]; den vil være indlysende'.

Faktisk sagde han ‚oversigts-diagrammer' og ‚tabeller'. Men hvis man tager højde for tredive års skift i terminologi/kultur, er det næsten samme pointe. På dette tidspunkt (tidligt i september 1996, omkring seks uger efter år nul) begyndte jeg at tænke på, at et navneskift ville være passende — når alt kommer til alt, var det jo ikke mere bare en POP klient. Men jeg tøvede, fordi der endnu ikke var noget virkelig nyt i designet. Min version af Popclient manglede stadig at udvikle en selvstændig identitet. Det ændredes radikalt, da Fetchmail lærte at videresende post til SMTP-porten. Jeg kommer til det om et øjeblik. Men først: jeg sagde ovenfor, at jeg bestemte mig til at bruge dette projekt til at teste min teori om, hvad Linus Torvalds havde gjort rigtigt. Hvordan (kan du sagtens spørge) gjorde jeg det? På disse måder:

1. Jeg frigav tidligt og hyppigt (næsten aldrig mindre end hver tiende dag; en gang om dagen i perioder med intens udvikling).

2. Jeg udvidede min beta-liste ved at tilføje enhver, som kontakte mig angående Fetchmail.

3. Jeg sendte sludrende meldinger til beta-listen hver gang, jeg frigav, hvor jeg opfordrede folk til at deltage.

4. Og jeg lyttede til mine betatestere, spurgte dem om beslutninger om design og roste dem hver gang, de sendte rettelser og feedback ind.

Gevinsten fra disse simple forholdsregler var øjeblikkelig. Fra projektets begyndelse fik jeg fejlrapporter af en kvalitet, som de fleste udviklere ville dræbe for, ofte vedhæftet rettelser. Jeg fik opmærksom kritik, jeg fik fanpost, jeg fik intelligente forslag til faciliteter. Hvilket fører til:

10. Hvis du behandler dine betatestere, som om de er din mest værdifulde ressource, vil de reagere ved at blive din mest værdifulde ressource.

Et interessant mål for Fetchmails succes er bare størrelsen af beta-listen, vennerne af Fetchmail. I skrivende stund har den 249 medlemmer, og der kommer to eller tre til hver uge.

Faktisk er der en interessant grund til, at listen er ved at miste medlemmer fra højdepunktet tæt på 300, da jeg reviderer i maj 1997. Adskillige folk har bedt mig slette dem, fordi Fetchmail virker så godt for dem, at de ikke længere har behov for at være på listen! Måske er det den normale livscyklus for et modent projekt af basar-stilen.

6 Popclient bliver til Fetchmail

Det virkelige vendepunkt i projektet var, da Harry Hochheiser sendte mig sin skrabede kode til at videresende post til klientmaskinens SMTP-port. Jeg blev straks klar over, at en pålidelig implementering af denne facilitet ville gøre alle de andre leveringsmåder praktisk talt forældede. I mange uge havde jeg kun rykket gradvist frem med Fetchmail, selv om jeg syntes, at designet af brugergrænsefladen var brugbart men groft — uelegant og med for mange ubetydelige muligheder stikkende frem alle vegne. Muligheden for at smide den hentede post ned i en postkasse-fil eller til standard uddata generede mig især, men jeg kunne ikke finde ud af hvorfor. (Hvis du ikke er interesseret i den tekniske side af hvordan elektronisk post på Internet fungerer, kan du uden problemer springe de næste to afsnit over)

Det jeg så, når jeg tænkte over videresendelse med SMTP var, at Popclient havde forsøgt at gøre for mange ting på en gang. Den var blevet designet til både at være en posttransportagent (MTA) og en lokal leverings-agent (MDA). Med videresendelse med SMTP kunne den komme væk fra MDA-området og blive en ren MTA, der afleverer posten til andre programmer, som så håndterer den lokale levering — ligesom Sendmail gør. Hvorfor besvære sig med kompleksiteten i konfigurering af MDA eller med at sætte en lås-og-tilføj på en postkasse, når port 25 til at begynde med næsten med sikkerhed findes på enhver platform med understøttelse af TCP/IP? Især når det betyder, at hentet post med sikkerhed vil se ud som SMTP-post fra en normal afsender, hvilket i virkeligheden var det, som jeg alligevel ville.

(Tilbage til et højere niveau...)

Selvom du ikke fulgte den foregående tekniske jargon, så er der adskillige lektioner her. For det første var denne ide om videresendelse med SMTP den største enkelte gevinst, jeg fik ved bevidst at efterligne Linus' metoder. En bruger gav mig denne fremragende ide — det eneste jeg behøvede at gøre var at forstå implikationerne.

11. Det næstbedste efter at få gode ideer er at genkende gode ideer fra dine brugere. Nogle gange er det sidste bedre.

Interessant nok finder du hurtigt ud af, at hvis du er hudløst ærlig med hensyn til, hvor meget du skylder andre mennesker, så vil verdenen som helhed behandle dig som om, du selv har opfundet hver en detalje, og som om du bare er passende ydmyg med hensyn til dit medfødte geni. Vi kan alle se, hvor godt det virkede for Linus! (Da jeg holdt dette foredrag på Perl-konference i august 1997, sad Larry Wall på forreste række. Da jeg kom til den sidste af ovenstående linier, råbte han i religiøs vækkelsesstil, ‚Sig det, sig det, broder'! Hele publikummet grinede, fordi de vidste, at det også havde virket for opfinderen af Perl)

Efter i nogle få uger at have kørt projektet i den samme ånd, begyndte jeg at få lignende ros ikke bare fra mine brugere men fra andre folk, som havde hørt rygtet. Jeg har gemt nogle af de emails; jeg vil tage dem frem igen, hvis jeg nogen sinde begynder at tvivle på, om mit liv har været noget værd :-)

Men der er yderligere to fundamentale, upolitiske lektioner her, som gælder for alle former for design.

12. Ofte stammer de mest slående og nyskabende løsninger fra en erkendelse af, at din forståelse af problemet var forkert.

Jeg havde forsøgt at løse det forkerte problem ved at forsætte med at udvikle Popclient som en kombineret MTA/MDA med alle slags smarte lokale leveringsmåder. Fetchmails design trængte til at blive genovervejet fra grunden som en rendyrket MTA, som en del af den normale postvej på Internet med SMTP. Når du rammer muren under udvikling — når du har problemer med at komme gennem næste rettelse — er det ofte på tide at spørge, ikke om du har det rigtige svar, men om du stiller det rigtig spørgsmål. Måske trænger spørgsmålet til at blive omformuleret. Ok, jeg havde omformuleret mit problem. Tydeligvis var det rigtige at gøre, (1) at hacke understøttelse af videresendelse med SMTP ind i standard-driveren, (2) at gøre det til standardmåden og (3) i sidste ende smide alle de andre leveringsmåder væk, især mulighederne for levering som fil og levering som standard uddata.

Jeg tøvede nogen tid med hensyn til skridt 3, da jeg var bange for at gøre gamle brugere af Popclient, som var afhængige af vekslende mekanismer til postleveringer, vrede. I teorien kunne de med det samme skifte til .forward-filer eller deres tilsvarende non-sendmail for at få samme effekt (for andre postservere end Sendmail) for at have samme mulighed. I praksis kunne overgangen blive noget skidt.

Men da jeg gjorde det, viste fordelene sig at være enorme. Den tungeste del af koden til driveren forsvandt. Konfigurationen blev voldsomt enklere — ikke flere problemer med MDA-systemet og brugerens postkasse, ikke flere bekymringer om hvorvidt det underliggende OS understøttede fillåsning.

Desuden forsvandt den eneste måde, man kunne miste post. Hvis du valgte aflevering som fil, og din disk løb fuld, mistede du posten. Det kan ikke ske ved videresendelse med SMTP, da postserveren ikke returnerer et OK med mindre beskeden kan blive leveret eller i det mindste sendt til en midlertidig kø for senere levering. Derudover blev ydelsen forbedret (dog ikke sådan, at du ville bemærke det i første omgang). En anden ikke uvæsentlig fordel af denne

ændring var, at den manuelle side blev enklere. Senere måtte jeg genindføre levering via brugerspecificeret MDA for at tillade håndtering af nogle obskure situationer med dynamisk SLIP. Men jeg fandt en meget simplere måde at gøre det på.

Moralen? Tøv ikke med at kaste over-udviklede funktioner bort, når du kan gøre det uden at miste effektivitet. Antoine de Saint-Expéry (som var pilot og flydesigner, når han ikke forfattede klassiske børnebøger) sagde:

13.„Perfektion (med hensyn til design) opnås ikke, når der ikke er mere at tilføje, men snarere når der ikke er mere at tage bort'.

Når din kode er ved at blive bedre og enklere, så ved du, at det er rigtigt. Og i processen fik Fetchmails design sin egen identitet, som var forskellig fra forgængeren Popclient.

Det var tid til en navneforandring. Det nye design lignede Sendmail meget mere end den gamle Popclient; begge er MTA'er, men hvor Sendmail sender før levering, så henter den nye Popclient før levering. Så to måneder ude af starthullerne omdøbte jeg den til Fetchmail.

Der er en mere almindelig lektie af lære fra denne historie om hvordan, SMTP-levering blev indarbejdet i fetchmail. Det er ikke kun fejlfinding, der kan paralleliseres; det kan udvikling og (i en måske overraskende grad) udforskning af designrummet også. Når din udviklingsmetode hurtigt gentages, kan udvikling og forøgelse blive særlige tilfælde af fejlfinding — at rette ‚udeladelsesfejl i softwarens originale anvendelighed og koncept. Selv på et højere designniveau, kan det være meget værdifuldt, at masser af tænkende medudvikleres tilfældigt gennemgår designrummet omkring dit produkt. Se til eksempel den måde en vandpyt finder et afløb, eller endnu bedre hvordan myrer finder mad: i grunden udforskning gennem spredning efterfulgt af udforskning formidlet gennem en skalerbar kommunikationsmekanisme. Det virker rigtig godt; som med Harry Hochheiser og jeg er det muligt, at en af dine udforskere finder et stort bytte i nærheden, som du var bare lidt for snævert fokuseret til at se.

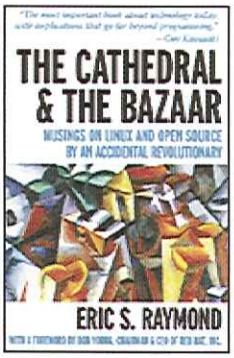
Noter:

[QR] Der har været eksempler på open source, basar-udvikling, der er fra før Internet-eksplosionen, og som ikke er relateret til traditionerne omkring Unix og Internet. Udviklingen af info-Zip-komprimeringsværktøjet omkring 1990-1992 især til DOS-maskiner var sådan et eksempel. Et andet var RBBS bulletin board-systemet (igen til DOS), som begyndte i 1983 og udviklede et tilstrækkeligt stærkt miljø, og der har været temmeligt regelmæssige frigivelser op til nu (midten af 1999) på trods af vældige teknologiske fremskridt med email og frem for alt fildeling på lokale BBS'er. Mens info-Zip-miljøet i nogen grad brugte email, var udviklerkulturen omkring RBBS faktisk i stand til at basere et væsentligt online-miljø på på RBBS, som var fuldstændigt uafhængig af TCP/IP-infrastrukturen.

[JH] John Hasler har foreslået en interessant forklaring på det faktum, at fordobling af indsats ikke synes at bremse open source-udvikling. Han foreslår, hvad jeg vil kalde ‚Haslers lov': omkostningerne ved dobbelt arbejde har tendens til at vokse med mindre end kvadratet på teamstørrelsen — det vil sige langsommere end de faste omkostninger forbundet med den planlægning og ledelse, som ville være nødvendig for at eliminere dem.

Denne påstand er ikke i modstrid med Brooks' lov. Det er måske sådan, at de samlede faste omkostninger ved kompleksitet og sårbarhed overfor fejl vokser med kvadratet på teamstørrelsen, mens omkostningerne ved dobbelt arbejde ikke desto mindre er et særligt tilfælde, der vokser langsommere. Det er ikke svært at udvikle plausible grunde til dette, startende med det utvivlsomme faktum, at det er meget lettere at blive enige om funktionelle grænser mellem forskellige udvikleres kode, som vil forhindre dobbelt indsats, end det er at forhindre de forskellige uplanlagte, dårlige samspil over hele systemet, der ligger til grund for de fleste fejl.

Kombinationen af Linus' lov og Haslers lov antyder, at der faktisk er tre kritiske/vigtige størrelsesforhold i software-projekter. I mindre projekter (med mellem en og højst tre udviklere) behøver ledelsesstrukturen ikke at være mere kompliceret end at udnævne en hoved-programmør. Og der er en mellemgruppe derover, hvor omkostningerne ved traditionel ledelse er relativt lave, så fordelene ved at undgå dobbelt arbejde, fejlfinding og overvågning, så detaljer ikke overses, er en netto fordel.



This journal re-publication of "The Cathedral and the Bazaar" has been authorized by Eric S. Raymond and O'Reilly & Associates. The paper and its sequels are collected in the book "The Cathedral and the Bazaar", (ISBN 1-56592-724-9) published by O'Reilly & Associates in October 1999. If you enjoy what you read here, please buy a copy and give it to whoever you think most needs to read it.

Men derover betyder kombinationen af Linus' lov og Haslers lov, at omkostningerne og problemerne i traditionelt ledet store projekt-grupper stiger meget hurtigere end den forventede omkostning ved dobbelt indsats. Ikke mindst består en del af omkostningen af en strukturelt bestemt manglende evne til at udnytte effekten af mange øjne, der (som vi har set) meget bedre end traditionel ledelse sikrer, at fejl og detaljer ikke overses. I store projekter er det således kombinationen af disse love, der effektivt får nettogevinsten ved traditionel ledelse til at gå mod nul.

[HBS] Opsplitningen i Linux' eksperimentelle og stabile versioner har en anden funktion, som er beslægtet med, men alligevel forskellig fra sikring mod risici. Opsplitningen bekæmper et andet problem: de farlige tidsfrister/deadlines. Når programmører skal opfylde en uforanderlig facilitets-liste og nå en fastsat definitiv deadline, går kvaliteten tabt, og der er sandsynligvis et stort rod på vej. Jeg er taknemmelig overfor Marco Iansiti og Allan MacCormack fra Harvard Business School, som gjorde mig opmærksom på, at planlægningen kan lykkes, hvis der slækkes på et af disse krav.

En måde at gøre dette på er at fastholde fristen/deadline men lade facilitets-listen være fleksibel og fjerne faciliteter, hvis de ikke er færdige inden fristen udløber. Det er i grunden strategien for den ‚stabile' udviklingsgren af kernen; Alan Cox (der vedligeholder den stabile kerne) sender frigivelser ud med ret jævnlige intervaller men garanterer ikke, hvornår specifikke fejl vil være rettet, eller hvornår faciliteter fra den eksperimentelle kerne implementeres i den stabile.

Den anden måde at gøre dette er at fastsætte en ønskelig facilitets-liste og først levere, når det er lavet. Det er i grunden strategien for den ‚eksperimentelle' udviklingsgren. De Marco og Lister citerede forskning, der viste, at denne planlægningspolitik (‚væk mig, når arbejdet er udført') ikke bare giver den bedste kvalitet men i gennemsnit kortere afleveringstider end enten ‚realistisk' eller ‚aggressiv' planlægning.

Jeg er begyndt at tro (i starten af 2000), at jeg i tidligere udgaver af dette dokument alvorligt undervurderede vigtigheden af ‚væk mig, når arbejdet er udført' anti frist-politikken for open source-miljøets produktivitet og kvalitet. Den generelle erfaring fra den forcerede GNOME 1.0 i 1999 antyder, at pres for en for tidlig frigivelse kan neutralisere mange af kvalitetsfordelene, som open source normalt giver.

Det kan meget vel vise sig, at den gennemskuelige proces inden for open source er en af tre ligeværdige drivkræfter bag kvaliteten sammen med ‚væk mig, når arbejdet er udført'-planlægning og selvudvælgelse blandt udviklere.

Rettelse:

I noten [JB] i sidste måneds afsnit af Katedralen og Basaren er titlen på bogen forkert. Det skal være Programming Pearls i stedet for Programming Perl. Vi beklager fejlen.

DKUUG-Nyt er medlemsbladet for DKUUG, foreningen for Åbne Systemer og Internet

Udgiver:
DKUUG
Fruebjergvej 3,
2100 København Ø.
Tlf: 39 17 99 44
Fax: 39 20 89 48
email: sek@dkuug.dk
Sekretariatet er åbent:
Mandag-fredag
kl. 9.00-17.00

Direktør:
Bo Folkmann

Redaktion:
Lotte Kristiansen
(ansvarshavende)
Gitte D'Arcy
Oskar Jensen
Hans Arne Niclassen
Jacob Bække
Peter Holm
Bo Folkmann

Tryk:
Palino Print

Annoncer:
Kontakt DKUUGs sekretariat

Oplag:
1500 eksemplarer

Artikler m.v. i DKUUG-Nyt er ikke nødvendigvis i overensstemmelse med redaktionens eller DKUUGs bestyrelses synspunkter. Eftertryk i uddrag med kildeangivelse er tilladt.

Deadline:
Deadline for næste nummer nr. 127 er onsdag d. 27. september 2000

Medlem af Dansk Fagpresse

DKUUG-Nyt
ISSN 1395-1440

Aktivitetskalender

September:

12. SSLUG Symbion
- Stordrift af Linux-kasser

16. SSLUG - Linux Demoday 2000

17. SSLUG - Linux´s 9 års fødselsdag

18.-19. DKUUG - Sikkerhedskursus "Intrusion Techniques and Countermeasures"

19. DKUUG/SILD - PHP

21. SSLUG Niels Bohr Instituttet
- Hyggemøde

26. DKUUG/Klub København

27. KLID
- Linux-klynger - supercomputere mm.

Oktober:

05. SSLUG
- Hyggemöte på Malmö Högskola
- gcc og embedded Linux

07 SSLUG - Alan Cox i Lund

10. SSLUG
- Softwarepatentering i EPO
- DKUUG/Symbion/M1

12. DKUUG seminar
- Kan Unix vinde markedsandele på Mainframe-området?

16. DKUUG/SILD - Perl

19. SSLUG
- ZSH - Z-shell på Niels Bohr Instituttet

24. FLUG - Linux Boot Proces

25. KLID - XML server

27. NBD - NetWare Perspective

November:

02. DKUUG Konference
- Kvinder og IT

02. SSLUG
- Hyggemöte på Pauliskolan i Malmö

14. SSLUG
- Hyggemøde på Niels Bohr Instituttet

16. DKUUG/SILD - DNS/BIND

25. SSLUG
- Årsmøde (generalforsamling)

28.-29. DKUUG konference
- ASP - Application Service Provider

29. KLID
- Internationalisering indenfor Linux

December:

09. SSLUG - LinuxFrokost

18. DKUUG/SILD - Julehygge

Se www.dkuug.dk for nærmere oplysninger

Pers hjørne

Man starter som barn

Igennem omkring 20 år har jeg haft fornøjelsen at kunne følge med i elever på danske folkeskoler, og hvordan de udarbejdede opgaver inden for et ganske bestemt emne. Emnet er såmænd UFOer, men egentligt er det konkrete emne uden betydning her.

Indtil for omkring 5 år siden gik det egentligt ganske godt med deres opgaver – når man altså lige ser bort fra bagateller som stavning og tegnsætning. Et intens arbejde gennem 1980erne og 1990erne med at opdrage skoleeleverne til en skeptisk holdning over for emnet begyndte at bære frugt, og skoleeleverne begyndte at forholde sig mere og mere kildekritisk til emnet UFOer. Og det var godt – for i guder, hvor der udgives meget bavl om et sådant emne.

Men pludselig for få år siden begyndte opgaverne at ændre fuldstændig karakter. I stedet for saglige oplysninger om UFOer og en vis kritisk holdning til de mange skøre påstande, der findes på området, kunne man nu begynde at læse om "interessante" teorier som at UFOer skulle komme fra jordens indre, at der er hemmelige ufobaser i USA, at der finde døde rumfolk på en base osv. osv.

Hvad var der sket? Jo, det var såmænd Internettet, der havde holdt sit indtog på skolernes dagsorden og i elevernes informationsindsamling. I stedet for at ringe til hjemlige ufo-entusiaster og interviewe dem, surfede eleverne nu rundt på webbet og fandt oplysninger om ufoer i alle verdenshjørner. Og med et slag indeholdt elevernes opgaver nu alskens påstande fra fantasiens overdrev – og stavningen og tegnsætningen var i øvrigt heller ikke blevet bedre.

Hvad er problemet, kan man spørge? Internettet er blot en nemmere adgang til denne verdens information, som eleverne alligevel på en eller anden måde kunne få fat på? Det er rigtigt, men der er to afgørende nye ting:

- For det første findes der millioner gange mere gak-gak på Internettet, for informationsleverandøren behøver ikke at finde nogen, der gider udgive ens materiale. Man poster det bare på en hjemmeside.

- For det andet fungerer nogle af de traditionelle vurderingskriterier for informationens lødighed ikke længere. Fordi en hjemmeside tilsyneladende er flot, omfattende og har mange referencer, behøver den ikke at være lødig.

Skoleeleverne bliver med andre ord ladt TOTALT i stikken hvad angår de gængse måder at vurdere information på.

Der er – på godt og ondt – kommet et nyt medie i denne verden. Mediet hedder Internettet, og det bliver uden tvivl en af de største forandringer i vores generation. Men det er vigtigt at lære vores unge mennesker, hvordan man anvender Internettet til at indhente information – og hvordan man kan vurdere denne information. Til dette er der brug for nogle helt nye værktøjer, værktøjer som vel dårligt er opfundet endnu.

Men med Internettet og information er to ting helt sikkert: Kildekritik er mere vigtig i dag end nogensinde. Og Internettet kræver en helt ny måde at udføre kildekritik på. Jeg frem til, at skolerne – gerne anført af undervisningsministeren – tager fat på denne udfordring.

SuperUsers a/s
Karlebogaard · Karlebovej 91 · DK-3400 Hillerød
Tel.: +45 48 28 07 06 · Fax: +45 48 28 07 05
Giro 458-2764 · E-mail: super@superusers.dk
URL http://www.superusers.dk

Brian Eberhardt, Direktør

SuperUsers a/s

SuperUsers a/s, en 100% dansk virksomhed med ca. 35 medarbejdere, har mange års erfaring inden for åbne netværk, operativsystemer og programmeringssprog:

- UNIX, Windows NT/ 98/CE, NetWare
- Internet/Intranet baseret på TCP/IP
- C/C++ /Java/Perl/ActiveX/HTML/CGI
- ORACLE og andre åbne databaser

SuperUsers a/s leverer viden og løsninger i form af undervisning og konsulentydelser inden for systemnære områder:

- System Drift
- System Support
- System Management
- System Integration
- System Udvikling

Her ses SuperUsers anno 1999 i rokoko stemning på gamle Karlebogaard.

Kurser

Åbne kurser: SuperUsers a/s afholder løbende ca. 115 forskellige kurser inden for internet, åbne netværk, operativsystemer og programmeringssprog.

Specialkurser: Derudover tilbyder vi at afholde kurser tilpasset efter kundens individuelle ønsker. Ved at plukke dele af eksisterende kurser og sammensætte disse, kan næsten ethvert behov opfyldes.

Kursusforløb: Vi hjælper gerne med at vurdere og sammensætte flere kurser, således at der opnås et sammenhængende forløb.

SuperUsers a/s er:

- Sylvan Prometric Testcenter og tilbyder/afholder tests, som fører frem til følgende certificeringer:
  Microsoft: MCP, MCSE og MSCD
  Novell: CNA, CNE og Master CNE.
- Microsoft Certified Technical Education Center (CTEC)
- Novell Authorized Education Center (NAEC).

Konsulentydelser

SuperUsers a/s har konsulenter indenfor:

- Drift: Support og konfiguration
- Udvikling: Analyse, design, programmering og test

Faste opgaver: Konsulenter til udførelse og styring af drift i større installationer.

Tilkald: Et af specialerne er udrykning med sekunders varsel til hasteopgaver - ofte opgaver, hvor andre har givet op.

Telefonsupport: Endelig tilbyder vi pakkeløsninger inden for "online support".