Medlemsblad for

Dansk UNIX-system Bruger Gruppe

DKUUG-Nyt

Nummer 20, 6. marts, 1989

Indhold

Redaktionelt

DKUUG-Nyts redaktion består af Søren O. Jensen og René Seindal (ansvarshavende). DKUUG-Nyt nummer 21 udkommer d. 1. april 1989. Deadline er d. 13. marts 1989.

Vi er naturligvis altid interesserede i indlæg fra folk. Det behøver ikke være lange artikler, men kan også være annonceringer, opfølgninger af tidligere artikler, eller andet. Hvis I blot har ønsker eller gode ideer til artikler, er I også meget velkomne til at kontakte os.

Indlæg, foreslag, ønsker, etc. kan sendes med elektronisk post til redaktionen på adressen:

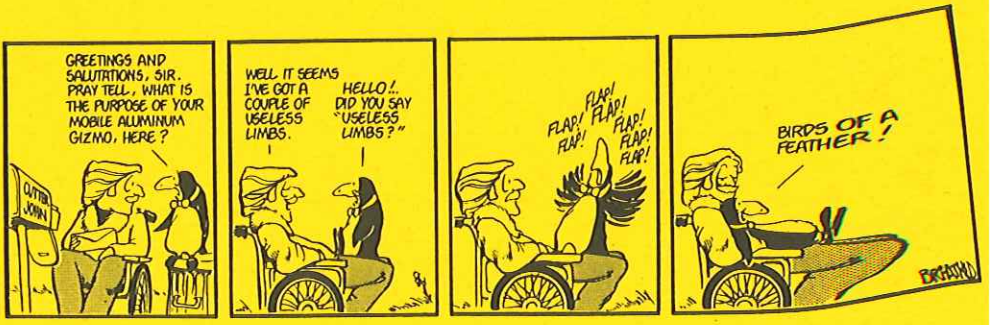
`dkuugnyt@dkuug.dk`

eller, hvis man foretrækker almindelig sneglepost, til:

René Seindal
Datalogisk Institut
Universitetsparken 1–3
2100 København Ø
Telefon: 01–39 64 66, lokal 221.

Udgivelsen af dette nummer er blevet skubbet lidt af grunde, som det vil tage for meget plads at komme ind på her. Vi satser på at udkomme til de fastsatte datoer for fremtiden.

Dette nummer indeholder to artikler af mere teknisk karakter. Det er lidt af et forsøg fra vores side, og vi vil meget gerne høre om folk er interesserede i flere artikler af denne art.



Internet-ormen

Af *René Seindal*

I sidste nummer af DKUUG-Nyt lovede vi at fortælle lidt om den famøse Internet-orm, som maltrakterede det amerikanske internet i november sidste år. Vi vil i denne artikel forsøge at beskrive hvad slags program ormen var, hvad det gjorde og ikke gjorde, hvordan det gjorde det og hvad konsekvenserne af det var.

internet-ormen er et selvreplikerende program, som blev sluppet løs på det amerikanske internet d. 2. november, 1988. Programmet kunne overføre sig selv til andre maskiner via internettet, og spredte sig således til tusinder af maskiner over hele USA. Programmet var ikke i sig selv ondsindet, men fejl i det gjorde at det ofte inficerede samme maskine flere gange, hvilket forøgede belastningen drastisk.

Denne artikel er baseret på to tekniske rapporter om internet-ormen, som blev skrevet kort efter dens fremkomst på internettet. Deres titler kan findes sidst i artiklen.

Lidt terminologi

Før vi kigger nærmere på selve ormen, er det vigtigt at gøre sig klart hvad forskellen på en orm og en virus er.

Den vigtigste forskel ligger i iagtagelsen af at en orm kan eksistere (leve?) uafhængigt af andre organismer, og reproducere sig uden hjælp udefra, mens en virus altid er bundet til en værtsorganisme, uden mulighed for at overleve uden værtsorganismen. I den forstand passer ordene ganske godt på deres edb-sidestykker.

De angreb som kendes fra PC-verdenen har alle været vira, som har inficeret værtssystemet (DOS), og spredt sig via inficerede disketter og lignende medier. Dette faktum har medvirket til at gøre virus-begrebet kendt, og delvis forårsaget den begrebsforvirring, der har været omkring internet-ormen.

En orm, derimod, er en organisme (et program) som kan leve uden at være bundet til andre specifikke organismer (programmer). (At internet-ormen kun fungerede på Vaxe og Sun-3'ere under Berkeley Unix ændrer ikke

noget. Der findes også orme, som kun kan leve i subtropiske omgivelser. Det gør dem ikke til vira af den grund).

Det vil fremgå af det følgende at Internet-ormen, som navnet siger, var en orm, og ikke en virus, selvom den ofte bliver omtalt som en sådan.

Hvad ormen ikke gjorde

En ikke uinteressant ting ved et ukontrollabelt program som Internet-ormen, er hvad det *ikke* gjorde.

Først, så angreb ormen ikke alle systemer, men kun Vaxe under Berkeley Unix, og Sun-3'ere under SunOS. Disse begrænsninger bidrog naturligvis i nogen grad til at hindre programmet i at sprede sig, men var dog ikke bare tilnærmelsesvis nok til at stoppe programmet.

Dertil kommer at ormen var afhængig af internettet for at kunne sprede sig. Det betyder specielt at systemer, som ikke var forbundet til internettet ikke kunne inficeres, hvilket på det pågældende tidspunkt inkluderede Danmark. Så vidt jeg ved, har der ikke været nogle angreb i Europa overhovedet.

Når ormen først havde inficeret et system, forsøgte den ikke på nogen måde at modificere værtssystemet. Den ændrede ikke nogle systemfiler, og den slettede ikke andre filer end sine egne temporære filer. Ormen forsøgte heller ikke at installere trojanske heste eller at registrere eller opsamle de passwords, som måtte komme den i hænde. Interessant nok, så forsøgte den heller ikke at opnå root-privilegier. Den udnyttede dem ikke engang, hvis den fik dem i hænde alligevel.

Hvad ormen gjorde

Når man tager i betragtning hvad ormen ikke gjorde, så er det ret klart, at dens eneste formål er at reproducere sig, og at sprede sig. Den gjorde begge dele meget effektivt.

Før ormen kunne inficerer en anden maskine, måtte den først finde ud af hvilke potentielle ofre, der var tilgængelige fra den maskine, den befandt sig på. Når den havde fundet et potentielt offer, var den nødt til at forbinde til den anden maskine over nettet, og få startet en shell i den anden ende. Hvis det lykkedes, kunne ormen overføre sig selv til den anden maskine,

starte den nye orm, og afbryde forbindelsen. Den ene orm var så blevet til to, som begge ville fortsætte med at lede efter maskiner at bryde ind på.

Lokalisering af potentielle ofre

En stor del af ormens succes i at brede sig, skyldes dens metoder til at finde andre maskiner den kunne inficerer. Ormen brugte forskellige metoder alt efter om den ledte efter maskiner på samme lokalnet eller på andre net.

Berkeley Unix giver mulighed for at erklære to maskiner for 'ekvivalente,' dvs. at man ikke skal angive password ved login mellem to ekvivalente maskiner. Det er på tilsvarende vis muligt for individuelle brugere at erklære at de stoler på givne brugere på andre maskiner, som da vil få adgang til den pågældende brugers konto uden at skulle bruge password. Det er et system, som kræver omtanke i brugen, men på trods af de åbenbare problemer det kan åbne op for, er det en fuldstændig uundværlig facilitet i et netværksbaseret miljø.

Ormen kunne bruge disse oplysninger til at gætte hvilke konti der måske fandtes på andre maskiner, ved at antage at det var et gensidigt tillidsforhold mellem to maskiner (hvad det som oftest også er). Logikken i det kan kort skitseres som følgende: Hvis en bruger ved navn 'foo' på maskinen 'A' erklærer at han stoler på brugeren 'bar' på maskinen 'B,' er der en vis sandsynlighed for at det rent faktisk er den samme person, der har begge konti, og der er således en god chance for at 'bar' på 'B' også stoler på 'foo' på 'A.' I så tilfælde kan man blot starte en shell på den anden maskine som den pågældende bruger, ved hjælp af standard procedure til udførsel af shell-kommandoer på andre maskiner.

Selv i det tilfælde hvor der ikke var adgang til andre maskiner uden password, kunne ormen stadig bruge en oplysning som at 'foo' på 'A' fik sin post videresendt til 'bar' på 'B,' til at bryde ind på den anden maskine. Blot det at kende et gyldigt brugernavn på en anden maskine, er det første skridt på vejen til en succesfuld angreb.

Hvis ormen kun havde angrebet lokale maskiner, var den aldrig kommet så langt omkring. Ormen brugte flere forskellige ikke-trivielle metoder til at finde ud af hvilke andre ikke-lokale maskiner den kunne komme i kontakt med. Den prioriterede specielt gateways højt (en gateway er en maskiner, der er forbundet til flere forskellige fysiske netværk, og den fungerer da som

en bro mellem de to net), da det ville give bedst mulig adgang til endnu flere maskiner.

Hvordan man får en shell på en fremmed maskine

Når ormen havde lokaliseret en potentielt offer, forsøgte den som det første at forbinde til den anden maskines *telnet* program. Hvis det lykkedes, afbrød ormen forbindelsen med det samme, og hvis det fejlede opgav den at angribe den anden maskine. På denne måde fik ormen filtreret 'dumme' bokse, som f.eks. routere, fra.

Efter at være kommet i kontakt med en maskine, var ormens første opgave at få startet en shell i den anden ende. Ormen havde tre måder den kunne starte en shell på den anden maskine på. De er, i den orden ormen prøvede dem på, (1) at benytte oplysninger om ekvivalente maskiner og konti, (2) at benytte et sikkerhedshul i *fingerd* programmet, og (3) at benytte et sikkerhedshul i *sendmail*. Hvad disse sikkerhedshuller dækker over vil vi gennemgå i det følgende.

Ormen forsøgte først at bryde ind på en almindelig konto på den anden maskine, ved at bruge brugernavne, som på den lokale maskine var erklæret ekvivalente med offeret, eller hvor den kendte det lokale password (en af grundene til at ormen forbrugte så mange ressourcer, var at den gjorde ihærdige forsøg på at bryde folks password, for på den måde at kunne komme videre). Hvis det lykkedes ormen at få adgang til en konto på den anden maskine, startede den ved hjælp af en standard biblioteksfunktion en shell på den anden maskine, som den kunne bruge til at overføre sig selv med.

Hvis det ikke lykkedes ormen at få adgang til den anden maskine via en bruger konto, forsøgte den at udnytte en sikkerhedshul i *fingerd* programmet. *Fingerd* er en facilitet, som tillader folk udefra at se hvilke konti der er på en maskine, og hvem der er logget ind. Det er absolut ikke noget essentielt program, men de fleste sites har det alligevel kørende.

Ormens angreb på *fingerd* er nok den mest opfindsomme del af ormen. Det foregik som følger: *fingerd* fungere ved at *finger* programmet fra en anden maskine forbinder over nettet til det. Det læser en linie fra netværksforbindelsen, giver den som argument til det lokale *finger* program, og sender uddata herfra tilbage over nettet. Det er et meget trivielt pro-

gram. Sikkerhedshullet var også trivielt, idet det bestod i at programmet læser sine inddata med biblioteksfunktionen *gets*() med en lokal variabel som den modtagende buffer! Oplagt, ikke?

Den buffer, som *fingerd* brugte, var på 512 tegn. Det ormen gjorde, var at den sendte en nøje konstrueret linie, på 536 tegn, som da overskrev return adressen fra *main*(), så den pegede ned i bufferen. Nå *fingerd* returnerede, udførte den koden i bufferen, som simpelthen startede en shell. Ormen havde da en netværksforbindelse til en shell på den nye maskine.

Hvis *fingerd* angrebet fejlede, faldt ormen tilbage på et sikerhedshul i *sendmail*. *Sendmail* er et systemprogram, som varetager forsendelse af elektronisk post over et internet, ved hjælp af SMTP protokollen. En af faciliteterne i *sendmail* er, at det kan udføre en given shell kommando som respons på modtagelsen af en brev. *Sendmail* har også nogle indbyggede 'debug' faciliteter. Hvis *sendmail* er konfigureret rigtigt (forkert!), er det muligt at aktivere 'debug' faciliteterne over nettet, og dermed starte programmer på den anden maskine, som får et brev som inddata.

De færreste leverandører har været opmærksomme på denne lille detaille, så mange systemer kørte *sendmail* med 'debug' aktiveret. Ormen benyttede det hul til starte en shell på det udvalge offer. Den lavede en forbindelse til *sendmail* på den anden maskine, aktiverede 'debug,' og sendte et lille shell-script med en `/bin/sh` som modtager.

Ironisk nok, så skyldtes denne 'debug' bagdør i *sendmail* en meget sikkerheds-orienteret systemadministrator på Universitetet i Berkeley, som insisterede på at benytte en beta-test udgave af *sendmail* (uden tilladelse, forøvrigt), men ikke ville forsyne *sendmails* forfatter med en konto til at teste den med. 'Debug' faciliteten blev derfor lagt ind for at kunne teste dette system, og den blev senere glemt [1].

Selvreplikationen

Nå ormen havde sikret sig at den anden maskine kunne inficeres, overførte den et lille 'vektor-program' til den anden maskine. Hvordan det blev gjort i praksis afhang af hvilken af de tre angrebsstrategier, der lykkedes. Vektorprogrammet blev oversat og startet på den anden maskine, og etablerede derefter en forbindelse tilbage til den maskine, den kom fra.

Når vektor-programmet og moder-ormen igen kom i kontakt med hinanden, blev der overført tre filer til den ny-inficerede maskine. Det drejer sig

om vektor-programmets kildetekst, og objekt filer til selve orme-programmet, i en vax udgave og en Sun-3 udgave. Der er her at begrænsningen til Vaxe og Sun-3'ere kommer ind i billedet. Ormen kan naturligvis ikke overføre sine egne kildetekster, da den jo så ville være triviel at stoppe.

Efter at filerne var overført, udførte vektor-programmet en shell, som læste kommandoer fra nettet. Moder-ormen sendte så de nødvendige kommandoer til at lænke orme-programmet sammen, både Vax og Sun-3 udgaven. Højst en af dem vile lykkes, og det resulterende program blev udført. Hermed var den nye maskine blevet inficeret.

Resultatet

Resultatet af den fremgangsmåde, der er blevet skitseret er naturligvis en eksponentiel vækst i antallet af orme, og at mange maskiner ville blive inficeret flere gange.

Der var lagt en del arbejde i at forhindre at en maskine blev inficeret for mange gange. Ormen ville med jævne mellemrum checke om der var andre udgaver af ormen på den samme maskine, og hvis der var det, ville de slå plat og krone (bogstavelig talt) om hvem der skulle dø. Dog var det kodet sådan at en ud af syv orme ville nægte at dø, sandsynligvis for at forhindre en falsk orm i at stoppe det hele, og selv om en orm tabte, ville den dog alligevel fortsætte i op til 15 minutter endnu. Resultatet var at mange maskiner blev oversvømmet med orme, og på et eller andet tidspunkt ville ormene ganske simpelt forbruge alle systemets ressourcer. Det gik specielt ud over foretrukne ofre som f.eks. gateways.

Ormen var meget omhyggelig med at beskytte sig selv og sine data mod opdagelse og undersøgelse. Den havde ingen filer i filsystemet, undtagen i en kort periode onkring inficeringen, og alle data og tekststrenge i programmet var kodet, og blev først afkodet når de skulle bruges. For at undgå at enkelte processer skulle tiltrække sig for meget opmærksomhed, rettede ormen i sin kommando-linie argumenter, så den lignede en almindelig shell, og den skiftede med jævne mellemrum sit process nummer ud, for ikke at akkumulere et for stort cpu-forbrug.

En ting som ormen ikke kunne kamouflere, var den meget store aktivitet som programmerne *telnet*, *sendmail* med 'debug' aktiveret, og *rexecd* (re-

mote execution daemon) udviste, og det var da også sådanne observationer, som ledte folk på sporet af ormen.

Lidt kronologi

De første kendte spor af ormen stammer fra den 2. november omkring klokken 18 (alle tider i amerikansk vestkyst tid), og omkring klokken 22 var flere systemer ubrugelige på grund af belastningen fra ormen. Den første dokumenterede erkendelse af at en orm/virus var løs på internettet er fra omkring klokken 23.30, i form af et elektronisk brev, som mange dog ikke fik, fordi de havde koblet sig af nettet i panik. Denne besked indholdt ellers meget vigtige oplysninger om hvordan ormen kunne begrænses i sin vækst.

Omkring klokken 3 om morgenen, den 3. november, har Berkeley et patch til *sendmail* klar, og poster det på nettet. Flere folk blive i løbet af morgenen opmærksomme på *finger*-angrebet, men af forskellige årsager bliver det ikke kendt før om eftermiddagen. Et officielt patch fra Berkeley bliver postet omkring klokken 19.

I løbet af natten mellem den 3. og 4. dekompilerer folkene i Berkeley og en gruppe på MIT ormen. De bytter kode et par gange i nattens løb. Klokken 6 om morgenen, den 4., er ormen stort set helt disassemblet, og næsten helt dekompileret. Omkring middagstid har både MIT gruppen og Berkeley gruppen dekompileret ormen til omkring 3200 linier C-kode.

Konsekvenserne

De kortsigtede konsekvenser af internet-ormen var naturligvis, at der gik panik i folk, da de så deres systemer opføre sig helt tosset. Mange lukkede deres systemer ned, og afbrød deres forbindelse til internettet. Meget store dele at internettet blev koblet af, og flere vigtige meddelser om hvordan ormen angreb blev forskinket i op til to dage.

De mere langsigtede konsekvenser er sværere at vurdere. Internettet kom meget hurtigt på benene igen, mest takket være den indsats som de to grupper i Berkeley og på MIT ydede, idet de meget hurtigt fandt ud af hvordan ormen virkede, og hvordan man kunne stoppe den.

En meget interessant observation er, at ormen sandsynligvis ville være blevet stoppet endnu hurtigere, hvis mange maskiner ikke var blevet lukket i

panik. Internettet var godt nok det middel, som ormen brugte til at sprede sig over, men det var også det våben, som stoppede den. Hurtigt kommunikation mellem geografisk spredte grupper var essentiel for den hurtige forståelse for hvad slags program ormen var, og hvordan den skulle stoppes. Mange vigtige informationer strandede, fordi vigtige gateways blev koblet af internettet.

Litteratur

[1] Eric Allman, *Worming my way...*, *UNIX Review*, vol. 7, no. 1, Januar 1989.

[2] Donn Seeley, *A Tour of the Worm*, Department of Computer Science, University of Utah.

[3] Eugene H. Spafford, *The Internet Worm Program: An Analysis*, Department of Computer Science, Purdue University, Purdue Technical Report CSD-TR-823.

EUUG-konferencen

Som I sikkert har læst i EUUG Newsletter, er det igen tid for EUUGs halvårlige konference. Denne gang foregår det i Bruxelles i dagene fra d. 3. til 5. april (tutorials d. 6. og 7. april). I forbindelse med konferencen er der også en udstilling.

Emnerne for konferencen er: Operativsystem-udvikling (bl.a. udvikling af distribuerede operativsystemer), netværk, sikkerhed, grafik, Unix standarder, værktøjer og sprog.

Konferencen bliver holdt på mere eller mindre korrekt engelsk og er en fantastisk mulighed for at mødes med andre Unix-folk.

Der er tilmeldingsblanketter i EUUG Newsletter.

DKnet status februar 1989

Af *Kim Fabricius Storm*
Texas Instruments A/S
storm@texas.dk

DKnet har radikalt forbedret post-tjenesten, login-tjenesten er sat i drift og vi skal have ny backbone maskine.

Post-tjenesten

Den internationale post sendes nu for næsten 100% vedkommende via faste internationale linier mellem de nordiske lande, til Europa via Holland og til USA.

Hvis en modtagers maskine kan nås direkte via disse linier – og det kan bl.a. de fleste amerikanske maskiner – så er vi nu i stand til at aflevere et brev mindre end 1 minut efter at det er ankommet til vores backbone maskine.

Lars Povlsen fra DIKU har ydet en stor indsats for at få dette til at fungere, hvilket vi er ham meget taknemmelig for.

Nyheds-tjenesten

Vi har besluttet at yde en introduktionsrabat på nyheds-tjenesten ved at gøre første kvartal efter tilmeldingen gratis. Dette er også tænkt som et incitament til at få programmelt installeret og sat i drift hurtigt efter tilmeldingen. Der er også tale om en stramning, idet der derefter vil blive opkrævet fast kvartalsabonnement uanset om programmellet er i drift eller ej.

Login-tjenesten

Login-tjenesten er nu sat i drift, men pga. forsinkelsen har vi forlænget prøveperioden til 1. april; indtil da er det gratis at anvende tjenesten (med et moderat forbrug og uden garanti).

Det er endnu for tidligt at drage nogen konklusioner mht. login-tjenesten, men interessen for at deltage i prøveperioden har været tilfredsstillende. Interesserede kan fortsat nå at melde sig til afprøvningen af login-tjenesten (kontakt DKUUGs sekretariat).

Arkiv-tjenesten

Pga. problemer med installation af en disk er arkiv-tjenesten endnu ikke etableret i fuldt omfang. De store pakker (f.eks. GNU og X) bør under alle omstændigheder bestilles via DKUUGs båndtjeneste, da de ellers vil belaste vore modem linier i et urimeligt omfang.

Nye Modemer

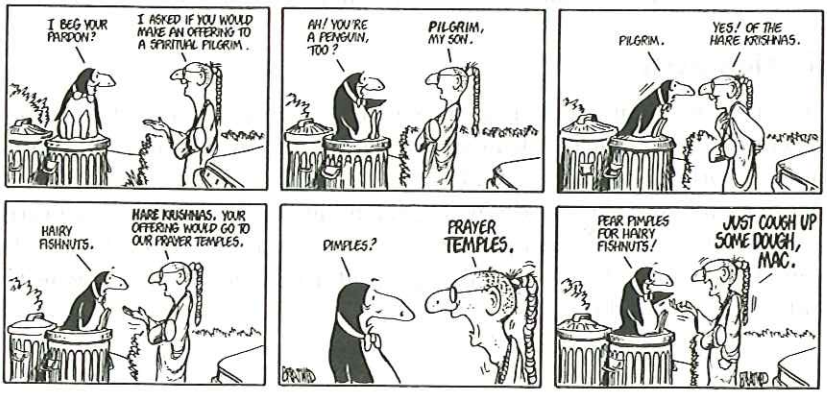
Som følge af den voksende brugerskare og dermed voksende trafik, er der desværre også voksende problemer med at komme igennem til backbone maskinen via de eksisterende modem-linier. Vi har besluttet at udvide antallet af modem-linier så hurtigt som KTAS kan levere dem.

Vi kan i denne forbindelse anbefale at anskaffe et TrailBlazer modem (se omtalen i DKUUG-Nyt nr. 19), da det er nemmere at komme igennem til backbone maskinen via dette modem (de har separat telefonnummer) og der er stadig rigelig fri kapacitet på disse linier.

Ny backbone

Den nuværende SUN-2 backbone har tjent os godt, men med den stærkt stigende trafik har vi nået grænsen for dens ydeevne.

Vi er derfor ved at forberede en flytning af backbone opgaven til en SUN-3 som vi har lejet på favorable vilkår hos IT-Center. Der ikke sat dato på overgangen, men vi håber det sker meget snart.



Stormedlemsskab

Af *Mogens Buhelt*
DKUUG
mogens@dkuug.dk

Som det er nogle af jer bekendt, blev det på generalforsamlingen d. 23. november i år vedtaget at etablere en ny medlemskategori: stormedlem.

Den rettede vedtægt siger herom:

> Et stormedlem er et organisationsmedlem, som ønsker at støtte foreningens aktiviteter yderligere, og som ønsker foreningens services til denne medlemskategori.

Et stormedlem er altså et særligt organisationsmedlem, og en organisation (f.eks. en virksomhed) kan selv vælge, om den vil være stormedlem eller almindeligt organisationsmedlem. Den kan derimod (naturligvis) ikke være individuelt medlem.

Årskontingenterne for stormedlemsskab og almindeligt organisationsmedlemsskab er henholdsvis 3600 kr. og 1800 kr.

Foreningens bestyrelse vil gerne opfordre alle organisationsmedlemmer – især naturligvis større og helt store virksomheder – til at blive stormedlemmer. Et ekstra driftstilskud vil give foreningen mulighed for at iværksætte flere aktiviteter til gavn for foreningens medlemmer. Men det skal ikke være lutter filantropi. Vi lægger ud med følgende ekstra tilbud til stormedlemmerne:

- Stormedlemmer kan tilmelde op til 10 medarbejdere - gerne med forskellige adresser – som *ekstra-modtagere*, dvs modtagere af DKUUG-Nyt og andet materiale, der udsendes af DKUUG. Almindelige organisationsmedlemmer kan højst tilmelde 3 ekstramodtagere, og denne regel vil for eftertiden blive håndhævet strengt. DKUUG-Nyt har i øvrigt nu fået en halvprofessionel redaktion, et nyt layout og kommer 8 gange om året.

  Vi minder i den forbindelse om, at et organisationsmedlem (og altså også et stormedlem) kan tegne vilkårligt mange *EUUG-abonnementer*,

dvs. ekstraabonnementer på EUUG Newsletter. Disse skal dog alle lyde på medlemmets adresse. Et EUUG-abonnement inkluderer, at abonnenten også bliver ekstra-modtager uden at blive talt med i ovennævnte begrænsning. Et EUUG-abonnement koster 250 kr/kalenderår (4 numre).

- Ekstra-modtagere hos stormedlemmer modtager desuden et gratis eksemplar af den danske UNIX-markedsoversigt.
- Stormedlemmer kan tilmelde et ubegrænset antal medarbejdere til foreningens arrangementer (medlemsmøder). Almindelige organisationsmedlemmer kan højst tilmelde 3, og også denne regel vil nu blive håndhævet strengt.
- Stormedlemmer har ret til at blive nævnt som sponsorer på DKUUGs stand på kommende udstillinger.
- De første 25, der melder sig som stormedlem, får et gratis eksemplar af /usr/group products catalog 1988 (medlemspris 300 kr. inkl. moms).

Vil I være stormedlem?

I kan altid konvertere et almindeligt organisationsmedlemsskab til et stormedlemsskab mod betaling af differencen på 1800 kr. (900 kr. ved konvertering efter 30. juni). Men jo før I gør det, jo før kan I udnytte de særlige fordele. Rekvirer derfor—hvis I er interesserede—en tilmeldingsblanket til stormedlemsskab i foreningens sekretariat, telefon 01–60 66 80, Email `mogens@dkuug.dk`. I sekretariatet kan I også få oplyst, hvem i organisationen, der p.t. er registreret som ekstra-modtagere og EUUG-abonnenter.

Mogens Buhelt

DKUUG Dansk UNIX-system Bruger Gruppe

Systemudvikling under Unix

Computer Aided Software Engineering (CASE) er et *hot* emne i dagens Unix verden. Der spås også om en stor fremtid for Objekt Orienteret System Udvikling. Kom og hør om disse to spændende emner og deres fællesnævnere.

Seminaret afholdes d. 30. marts på Hotel Scandinavia i København. Seminaret har særlig interesse for system-analytikere og -udviklere, metodefolk, softwarechefer mv.

9.00–9.15 Registrering.

9.15–9.30 Velkomst
v/ Hans Kierulff, Merkur Data A/S

9.30–10.30 Om CASE
Metodisk tilgang til systemarbejdet giver resultater, edb-støtte (CASE) kan gøre det lettere.

10.30–11.30 CASE—en brugers erfaring
v/ P&T LISDA Projekt
Bliver systemarbejdet forenklet, når man bruger CASE?

11.30–12.30 Frokost.

12.30–13.30 Objekt Orienteret Programmering

DKUUG - Dansk UNIX-system Bruger Gruppe
Bestillingsliste vedr. medlemstilbud

Listen sendes til:

DKUUG
Sekretariatet
Kabbelejevej 27 B
2700 Brønshøj

(telefon 01 60 66 80)

Afsender:
Medlemsnr.:

att:

Medlems-
navn og
adresse:

Medlemsnavn og adresse tages normalt fra vor database, men bedes angivet her (gerne stempel) af hensyn til kontrol.

Prissatte medlemstilbud (priser i kr inkl. moms)	Antal	Medl.pris	Beløb
UNIX-bogen (dansk udgave af "UNIX - the book")		175,00	
Dansk UNIX markedsoversigt, 2. udgave 1988-09-23 ...		50,00	
/usr/group products catalog 1988		300,00	
Kompendium DKUUG netseminar 1988-11-22		60,00	
Beløbet er: [ ] ikke betalt. Send faktura. [ ] vedlagt i check. [ ] betalt på giro 1 37 86 00.		I alt:	

Øvrige medlemstilbud, der fremsendes gratis	Antal

- DKUUG's DKnet Tjeneste	
- Ekstra abonnement på DKUUG udsendelser (abonnementet er gratis, højst 2 stk pr. organisationsmedlem) ...	
- Ekstra abonnement på EUUG Newsletter og DKUUG udsendelser (250,- DKK pr. år, kun organisationsmedlemmer og studerende)	
- Medlemsskab af /usr/group (70,-/120,- USD pr år for associeret/generelt medlemsskab)	
- Abonnement på PC World og/eller Computerworld (100,- DKK + moms pr. stk for første år)	
DKUUG's vedtægter pr. 1988-11-23	
DKUUG's medlemsliste (overdriv ikke)	
DKUUG's pjece (vær ikke for tilbageholdende)	
"Hvordan kommer jeg på UNIX-nettet?"	
"USENET NEWS beskrivelse" på dansk	
Vejledning vedr. DKUUG bånddistribution (public domain software)	

Dato:

Underskrift:

Forbeholdt DKUUG:
Modt. d.

Eksp. d.

Medltilb 88-12-16

En oversigt over de populæreste objekt orienterede programmeringssprog, og de applikationsområder hvor de kan udnyttes.

13.30–14.30 Objektorientering i praksis
Skaber udvikling af edbsystemer med objekt orienterede programmeringssprog behov for nye arbejdsformer i det øvrige systemarbejde.

14.30–14.50 Kaffe og te.

14.50–16.00 Sammenfald
v/Søren Lyngsø, Lyngsø Information Industry.
Hvilke sammenfald er der mellem de to nye systemudviklingsformer, CASE og Objekt Orientering.

16.00-16.45 Debat.
Foredragsholderne vil være tilstede.

Der vil i forbindelse med seminaret blive udleveret proceedings.

Pris for deltagelse er kr. 800 for medlemmer, og kr. 1200 for ikke-medlemmer, og tilmelding skal ske til sekretariatet inden d. 22/3 1989. Sekretariatets adresse er:

DKUUG
Kabbelejevej 27B
2700 Brønshøj
Tel: 01–60 66 80

Referat af Generalforsamlingen

Af *Peter Cordsen*

Her er det endelige referat af DKUUGs generalforsamling i november. Der har tidligere, i DKUUG-Nyt nr. 18, været bragt en reportage fra generalforsamlingen.

Referat af ordinær generalforsamling i DKUUG, afholdt d. 23. november 1988 kl. 13.30 på Hotel Scandinavia, København.

Dagsorden:

Valg af dirigent og referent
Lovlighed af indkaldelse konstateres
Bestyrelsens beretning
Regnskab
Indkomne foreslag
Valg af Bestyrelse
Valg af revisor og -suppleant
Budget og fastsættelse af kontigent
Eventuelt

Valg af dirigent og referent

Bestyrelsen indstillede Gert Illemann, NCR som dirigent og Peter Enø Cordsen som referent, hvilket forsamlingen tilsluttede sig.

Lovlighed af indkaldelse konstateres

Dirigenten konstaterede, at generalforsamlingen var rettidigt indkaldt.

Bestyrelsens beretning

Keld Simonsen gennemgik den skriftlige beretning fra DKUUG-Nyt nr. 17. Den medfølgende diskussion drejede sig især om medlemsmøderne, DKUUG-Nyt og leverandørgruppen. Omkring medlemsmøderne blev der udtrykt ønske om bla. kort referat, deltagerliste og især proceedings (OH-er) fra møderne og arkivering af disse i DKUUG. Oftere afholdelse i Odense og

benyttelse af billigere lokaliteter som KU var der også et vist ønske om, ligesom der blev givet udtryk for at frokosten havde mindre betydning.

Diskussionen om DKUUG-Nyt drejede sig udover forslaget om to ansatte redaktører om:

- tilsendelse af andres nyhedsblade.
- ønsket om en biblioteksfunktion.
- hellere vurderinger end klip fra andre kilder.
- anmeldelser af maskinel på det danske marked.

At der skulle være et problem i, at begge de foreslåede redaktører er involveret i nettet blev afvist af bestyrelsen. Leverandørgruppen efterlyser arbejdsemner, og der blev fra forsamlingen foreslået, at produktkataloget kom på nettet. Endvidere var der foreslag om særlige bruger- eller interessegrupper.

Regnskab

Regnskabet, som er blevet udsendt sammen med indkaldelsen, blev gennemgået af Mogens Buhelt. Der er en fejl i det udsendte regnskab, idet budgetforslagkolonnen ud for nettoaktiver primo skal stå i 132 st. f. 147, og derfor 677 i st. f. 682 udfor balance. Regnskabet blev godkendt.

Indkomne foreslag

Det blev vedtaget, at følge bestyrelsens indstilling, om at ansætte René Seindal og Søren O. Jensen, begge DIKUf som redaktører af DKUUG-Nyt for det næste års tid.

Bestyrelsen foreslag om en såkaldt leverandør medlemskatagori blev vedtaget efter, at der var foretaget væsentlige ændringer, så formuleringen blev:

> Et stor-medlem er et organisationsmedlem, som ønsker at støtte foreningens aktiviteter yderligere, og som ønsker foreningens services til denne medlemskategori.

Spørgsmålene til bestyrelsen gik især på hvilke konkrete tjenester, det drejede sig om, og hvor mange potentielle leverandørmedlemmer der er. Bestyrelsen nævnte som eksempel, at man kunne få flere eksemplarer af DKUUG-Nyt, og potentialet lå på 80 til 240 leverandørmedlemmer.

De indvendinger, som det vedtagne forslag løste op for, handlende især om manglende frivillighed og at små firmaer kommer i klemme.

Valg af Bestyrelse

Genvalgt blev:

Keld Simonsen — Center for Anvendt Datalogi
Isak Korn — IT Center
Kim Biel-Nielsen (næstformand) — UNIWARE
Erik Wismann (sekretær) — SDC
Kim F. Storm — Texas Instruments

I stedet for de tre, som ikke ønskede at opstille, blev følgende nyvalgt:

Tonny Andersen (kasserer) — Advanced Computer Software
Hans Kierulff — Merkur Data-Service
Steen K. Larsen — Ambrasoft
Poul-Henning Kamp — Kuwait Petroleum

Antallet af bestyrelsesmedlemmer blev hermed med et snuptag udvidet fra otte til ni, bl.a. fordi der nu netop var ni frivillige.

Valg af revisor og -suppleant

Lars Thorsen, IBM blev revisor, og Dennis Olsson suppleant.

Budget og fastsættelse af kontigent

På grund af princippet om frivillighed i det vedtagne forslag om stormedlemmer, havde bestyrelsens ingen garanti for, at budgettet kunne holde med de foreslåede kontigentsatser. Resultatet blev herefter vedtagelsen af de nedenfornævnte satser:

	1988	1. foreslag	Vedtaget
Individuelle medlemmer	300	600	600
Organisationsmedlemmer	900	1.500	1.800
Stor-medlemmer	–	2.500	3.600

Budgettet blev vedtaget.

Eventuelt

Efter spørgeskema-undersøgelsen om det månedlige møde på Ambrosius blev det oplyst, at det nu vil foregå på AW på Gammel Kongevej, og at det nye sted skulle annonceres i DKUUG-Nyt og resultere i flere deltagere.

Netværk på Mikrodata?

I de senere år har der på mange udstillinger i udlandet været installeret et fælles ethernet, for at demonstrere at systemerne virkelig kan tale sammen. Vi har tænkt på at det samme kunne lade sig gøre på Mikro-data 89.

Ideen er altså kort og godt at nedlægge et ethernet mellem de stande, der vil være med, slutte en masse maskiner på, og demonstrere networking for alvor.

Man kunne for eksempel have to logins på de tilsluttede maskiner, guest1 og guest2, som var offentligt tilgængelige, men passende beskyttede, et par filer til ftp, email, news, ideerne er mange.

DKUUG vil i den forbindelse gerne koordinere tildeling af Internetadresser, og generel bistand med at få det til at virke.

Hvis der er nogen der synes at dette er alletiders ide, så kontakt `netpasser@dkuug.dk`, eller på telefon 01–397322, hvor der som regel er en telefon-svarer.

Poul-Henning Kamp

Profilering af C-programmer

Af *Poul-Henning Kamp*

Jeg har ladet mig fortælle, at årsagen til de gamle ægyptere altid lod sig afbillede i profil, var ønsket om at få sjælen med. Dette er delvist sammenligneligt med formålet med *prof*(1)—Hvad bruger programmet sin tid til?

Der er selvfølgelig mange programmer, der ikke er den anstrengelse værd, som det vitterligt er at lave en profilering, for derefter at optimere programmet. Men personligt er jeg en del fornærmet over at få leveret væsentlige dele af mit OS som uoptimeret kode, ikke mindst kernen. Jeg kender maskiner, hvor System/User-time forholdet ikke kan bringes under 0.2 for typiske programmer.

Jeg vil i det følgende vise hvilke data man kan hente ud af et UNIX-systems profilerings mekanismer, og derefter fortælle hvordan man selv kan implementere en meget stærk profilering.

Jeg har brugt et program ved navn *hgrep* som eksempel, det kom på nettet for nogle måneder siden.

prof(1) er baseret på kernens *monitor*(2) facilitet, der ca. 60 gange i sekundet noterer sig hvor i objekt-koden programmet "opholder" sig. Derudover indsætter C-compileren kode i starten af hver procdure, der tæller antallet at gange hver procedure startes.

Som det ses af ovenstående eksempel, er det ikke meget information man får ud. Hvis man skal have et fornuftigt resultat skal programmet, der er på tællebrædtet, køre i lang tid, og helst ikke lave for meget I/O. Hvis programmet kun kører kort tid, får man for få data til en pålidelig analyse, og hvis programmet er meget I/O afhængigt, kan man komme ud for, at der ikke går 1/60 sekund mellem hver systemkald, og dermed falder kvaliteten af de data, man får fra *prof*(1), ved stikprøver bør prøverne jo ikke tages systematisk.

Jeg synes selv, at *prof*(1) er som en abe i havsnød: ikke meget værd.

gprof(1) - en lidt bedre løsning

gprof(1) virker på samme måde som *prof*(1), men giver lidt mere information end *prof*(1), nemlig hvor mange gange procedure X kalder procedure Y.

Name	%Time	Seconds	Cumsecs	#Calls	msec/call
_pipe	33.3	0.02	0.02	1	17
_flsbuf	33.3	0.02	0.03	439	0.04
_write	33.3	0.02	0.05	10	1.7
_putch	0.0	0.00	0.05	18	0.0
_fgets	0.0	0.00	0.05	10	0.0
_monitor	0.0	0.00	0.05	2	0
_creat	0.0	0.00	0.05	1	0
_popen	0.0	0.00	0.05	1	0

Figur 1: Eksempel på output fra *prof*(1)

Desværre er de grundliggende data stadig for dårlige, man ved ikke rigtig hvad der sker i koden.

icnt(1) — Det total overblik, for selvbyggere

Ideen fik jeg fra en artikel af P.J.Weinberger: "Cheap Dynamic Instruction Counting" fra Bell Lab's tech. journal.

icnt(1) virker ved at tælle hver gang koden for en kilde-tekst linie udføres. Det giver en tids-uafhængig optælling, der ikke er baseret på stikprøver, men en total optælling.

Det er nemt at se, at vi ikke længere ved hvor meget tid, der anvendes i en bestemt procedure, men til gengæld ved vi, hvor ofte bestemte løkker og forgreninger blev udført.

icnt(1) et virkeligt godt værktøj, der kræves ingen ting af programmet på tællebrædtet. Man kan sågar summere resultateter fra flere kørsler af samme program, evt med forskellige ind-data. Men bedst af alt: man behøver end ikke et Super-user password for at implementere Icnt, kun lidt kendskab til assembler på den CPU, man vil gøre det på (det er dog ikke helt trivielt).

Implementering af Icnt(1) – Gør Det Selv

Jeg har implementeret Icnt på flere maskiner, og det er egentligt nemt, alt taget i betragtning:

granularity: each sample hit covers 4 byte(s) for 25.00% of 0.07 seconds

%time	cumsecs	seconds	calls	name
25.0	0.02	0.02	9	_write
25.0	0.03	0.02	3	_read
25.0	0.05	0.02	1	_syscall
25.0	0.07	0.02		_main
0.0	0.07	0.00	439	__flsbuf
0.0	0.07	0.00	421	_strncmp

index	%time	self	descendents	called/total called+self called/total	parents name children	index
					<spontaneous>	
[1]	90.1	0.02	0.04		_main	[1]
		0.00	0.02	421/439	__flsbuf [2]	
		0.00	0.02	10/10	_fgets [3]	
		0.00	0.00	421/421	_strncmp [16]	
		0.00	0.00	4/4	_strcat [20]	
		0.00	0.00	18/439	_putch [11]	
		0.00	0.02	421/439	_main [1]	
[2]	37.5	0.00	0.03	439	__flsbuf	[2]
		0.02	0.00	9/9	_write	[9]
		0.00	0.01	1/2	_fstat	[6]
		0.00	0.00	1/2	_isatty	[24]

Figur 2: Eksempel på output fra *gprof*(1)

```
    |       else
  421|               {
  421|               if ( strncmp( &(bigbuf[i]), grepword,
    9|                   for ( j = i; j < i + grepwordlen; j++ )
   27|                       somap[j] = 1;
  421|               }
    |
    |       /* And now do the highlighting. */
    9|       for ( i = 0; bigbuf[i] != ' '; i++ )
  421|           {
  421|           if ( somap[i] )
   27|               {
   27|                     if ( ! soon )
```

Figur 3: Eksempel på output fra *icnt*(1)

Man kan enten få C-compileren til at gøre det, det kræver mod og kildetekst. Alternativt kan man få C-compileren til at markere hver sourcelinie i den assembler-kode, den laver.

Det kan være, at den gør det som default, ellers vil en eller anden option få den til det. kig efter sdb/xdb/*db relaterede options.

Lav et program (i c/awk/perl eller lign.) der laver et funktionskald istedet for disse linier. Alternativt kan man lave tælleriet direkte i koden, men man står mere frit med et funktionskald.

Den væsentligste egenskab ved den kode man sætter ind er selvfølgelig, at den *ikke* ændrer på korrektheden af programmet, hold udkig efter flag og registre, der ændres eller post-jump-execution (late-jumps).

Skriv noget kode, der samler alle tælletabeller, så deres positioner er kendt, ellers kan tælletallene ikke genfindes.

Skriv en *exit*(2) rutine der skriver tælletallene i en fil.

Der er intet til hinder for at bruge metoden på en unix-kerne, man må så bare hente sine tælletal fra /dev/kmem.

Eksempel på implementering: Pyramid 9825 versionen

Jeg inkluderer her som et bevis på metodens implementabilitet [1] de fire filer, der er relevante for min implementering på vores Pyramid 9825. Jeg vil ikke kommentere disse i detaljer, der er vist ikke så mange af den slags maskiner i omløb herhjemme.

Hvis der er spørgsmål er man velkommen til at kontakte mig.

```
# Makefile for hgrep

# Hvis c-compileren faar en -g option producerer den kassevis
# af information for diverse debuggere.

        cc -S -g $<
        mv $*.s $*.ss
        /work/system/phk/ICnt/incnt $< < $*.ss > $*.s
        cc -c $*.s

hgrep:  hgrep.o Makefile
        cc hgrep.o -ltermcap /work/system/phk/ICnt/ctr.c -o hgrep
        tst

hgrep.o: hgrep.c
```

[1] Nyt ord?

```
/* source for incnt program */
main(argc,argv)
int argc;
char **argv;
{
    char s[200];
    int i,k,km,y;

    km=0;
    while(gets(s))
    {
        if(strncmp(s,"\t.stab",6))
        {
            puts(s) ;
            continue;
        }
        if(strncmp(s,"\t.stabd",7))
            continue;
        y=sscanf(s,"%*s 0x%x,0x%*x,0x%x",&i,&k);
        if(y != 2 || i != 0x44)
            continue;
        /* Aha ! en ny sourcelinie */
        if(k > km)
            km=k;
        /* Lav et kald til __ICNT:
           __ICNT(LARRY,__LINE__,__FILE__); */
        printf("#line %d\n",k);
        printf("\tmovw\t$LARRAY,tr0\n");
        printf("\tmovw\t$0x%x,tr1\n",k);
        printf("\tmovw\t$LFILENM,tr2\n");
        printf("\tcall\t___ICNT\n");
    }
    /* Producer de data-omraader der skal til */
    printf("\t.data\t1\n");
    /* source-filename */
    printf("\t.align\t5\n");
    printf("LFILENM:\n");
    printf("\t.string\t%c%s%c\n",34,argv[1],34);
    /* taelle-array */
    printf("\t.align\t5\n");
    printf("\t.lcomm\tLARRAY,0x%x\n",4*km+4);
    return 0;
}
```

```
/* source for ctr.c */
#include <stdio.h>
#include <syscall.h>
#include <sys/types.h>
#include <sys/times.h>
#ifndef __ICNTFILES
#define __ICNTFILES 50
#endif
struct __ICNT_S
{
    char *s;
    int *m;
};

static struct __ICNT_S __ICNT_ST[__ICNTFILES];

char * __ICNT_FILE;

__ICNT(m,n,s)
char *s;
int *m;
int n;
{
    struct __ICNT_S *p;
    ++m[n];
    /* Hvis m[0] == 0 skal vi
    notere os taelle-arrayets adresse og filnavn */
    if(!*m)
    {
        for(p=__ICNT_ST;p->s;p++)
            ;
        p->s=s;
        p->m=m;
    }
    /* Husk hoejeste linienr */
    if(n> *m)
        *m=n;
}
/* fortsaettes */
```

```
/* source for ctr.c  -- fortsat */
/* Her skal man vaere MEGET forsigtig ! */
/* Hvis man proever at exit(x) herinde fra gaar det MEGET galt */
exit(n)
int n;
{
    FILE *f;

    if(!__ICNT_FILE)
        __ICNT_FILE="icnt.out";
    f=fopen(__ICNT_FILE,"w");
    if(f)
    {
        struct __ICNT_S *p;
        struct tms t;
        int i;
        for(p=__ICNT_ST;p->s;p++)
        {
            for(i=1;i<p->m[0];i++)
                if(p->m[i])
                    fprintf(f,"%s\t%d\t%d\n",p->s,i,p->m[i]);
        }
        times(&t);
        fprintf(f,"U_TIME\t0\t%d\n",t.tms_utime);
        fprintf(f,"S_TIME\t0\t%d\n",t.tms_stime);
        fclose(f);
    }
    _cleanup();
    return(syscall(SYS_exit,n));
}
```

```
: 'cntprt'
i=/tmp/icnt$$
set -x
cat $* |
awk '
        { a[$1" "$2] += $3 }
END     {
        for(i in a)
          print i"      "a[i]
        }'  > $i
cut -f1 $i | sort -u |
while read a
do
  (
  grep "^$a     " $i |
    awk '
        { printf("%4d!%7d||n",$2,$3) }
    '
  pr -n:4 -t $a
  ) | sort -n |
  sed '
  /^....:/s//       |/
  /^....!/{
  N
  s/^....!\(........\)\n....:/\1/
  }
  ' | pr -e -f -h "ICNT: $a/$i"
done
```

Oversigt over medlemsmøder i 1989

Dato	Sted	Tema
30/3	København	Systemudviklingsværktøjer (CASE).
19/4 †	København	UNIX i kontormiljøer.
30/5	Odense	Datasikkerhed.
31/5	Odense	Netværk.
14/6 †	København	Migration til UNIX.
28/9	København	Distribuerede systemer.
11/10 †	Provinsen	Industrielle systemer.
27–28/11	København	Årsmøde med netindslag.

De med † markerede møder er eftermiddagsarrangementer, som typisk er af ca. 2 timers varighed, placeret efter normal arbejdstid. Disse møder er gratis. De øvrige møder er heldagsarrangementer.

Tid, sted og program for hver enkelt møder vil blive annonceret i DKUUG-Nyt forud for mødets afholdelse.