
# Bruk av Honeypot-konseptet for å øke sikkerheten i nettverket

Hovedoppgave

ved

sivilingeniørutdanningen i

informasjons- og kommunikasjonsteknologi

Trond Høgsaas

Grimstad, mai 2000

# Forord

Denne rapporten inneholder resultatet av diplomarbeid utført i vårsemesteret 2000 av Trond Høgsaas, student ved sivilingeniørstudiet i informasjons- og kommunikasjonsteknologi ved Høgskolen i Agder. Diplomoppgaven er normert til 10 vekttall, som svarer til arbeidsmengden i ett semester.

Bakgrunnen for oppgaven er de siste årenes voldsomme vekst i antall organisasjoner og privatpersoner som er knyttet opp mot Internett. Etter hvert som det tilbys flere og flere tjenester over nettet, blir nettverkssikkerhet stadig viktigere. Ved å sette opp lokkesystemer, såkalte honeypots, kan en studere hvordan en inntrenger jobber, for siden å bruke denne kunnskapen til å beskytte de virkelige systemene til organisasjonen.

Oppgaven er gitt av firmaet System Sikker AS i Arendal. System Sikkerhet er et uavhengig konsulentfirma med spesialkompetanse innen informasjonssikkerhet..

Jeg ønsker å takke System Sikkerhet for oppgaven, og faglig veileder Per Kristian Johnsen for gode ideer og innspill underveis. Jeg vil også få takke ansvarlig veileder Vladimir Oleshchuk ved Høgskolen i Agder.

Grimstad, 29. mai 2000

Trond Høgsaas

# Sammendrag

De siste årenes kraftige vekst i nettverksbaserte tjenester, har ført til at datasikkerhet er blitt en stadig viktigere del av organisasjonenes IT-strategi. Et av de viktigste midlene i kampen mot datakriminalitet er kunnskap.

Ideen bak honeypot-konseptet er å lokke til seg fiendtlige angrep, for så å studere angrepet og tilegne seg ny kunnskap som kan brukes for å bedre sikkerheten i eget nettverk. I tillegg kan den virke som en distraksjon, og avlede oppmerksomheten fra organisasjonens produksjonsnett.

Det finnes to ulike typer honeypots. Den første er basert på programvare som simulerer nettverkstjenester. Den andre er bygd rundt et reelt operativsystem og kjører virkelige tjenester. Begge disse har sine sterke og svake sider.

Ved å benytte seg av den reelle typen honeypots, kan en bygge en kopi av organisasjonens produksjonsserver, og finne svakheter hos denne. Etter hvert som organisasjonens serverbehov økes, blir det naturlig å fordele oppgavene på ulike servere. Siden honeypoten har relativt liten arbeidsbyrde, kan det være ønskelig å la en maskin simulere flere servere. I denne rapporten er det vist hvordan en prototype på et slikt system kan være bygd opp.

# Innholdsfortegnelse

# 1 Innledning

## 1.1 Beskrivelse av oppgaven

Ideen bak honeypot-konseptet er å lokke til seg fiendtlige angrep, for så å studere angrepet og tilegne seg ny kunnskap som kan brukes for å bedre sikkerheten i eget nettverk. I tillegg kan den virke som en distraksjon, og avlede oppmerksomheten fra organisasjonens produksjonsnett.

Det finnes ulike måter å realisere en honeypot på. En mulig innfallsvinkel er å sette opp en maskin med en helt normal utgave av operativsystemet som en vil undersøke. Denne maskinen må kjøre de tjenestene som er aktuelle. Det er her meningen at en inntrenger skal kunne ta over maskinen, men alt som gjøres skal kunne logges. Samtidig må en forsikre seg om at ikke maskinen kan brukes i angrep videre. En annen mulig innfallsvinkel er å kjøre programvare som kan simulere ulike operativsystemer, og den responsen de gir på ulike angrep. Her er det meningen at en inntrenger tror han har tatt over maskinen.

Oppgaven har 4 deler:

- Gi oversikt over eksisterende produkter som implementerer honeypot-konseptet.
- Sammenligne muligheter, ulemper/fordeler i de to ulike innfallsvinklene skissert over.
- Kartlegge ulike tiltak som kan bidra til å gjøre systemet mer funksjonelt og realistisk. Det kan, for eksempel, dreie seg om å automatisere gjennomgang av ulike loggfiler for å trekke ut kunnskap om mulige angrep, eller å sette opp en maskin som kan simulere et helt nettverksegment.

- Videreutvikle honeypot-konseptet i retning skissert over og lage en demonstrasjons-prototype av det.

## 1.2 Motiv

System Sikkerhet er et uavhengig konsulentfirma med spisskompetanse innen informasjonssikkerhet. Firmaet har som målsetning å være ledende innen sitt felt, både nasjonalt og internasjonalt, og jobber derfor kontinuerlig med å utvide og videreutvikle sitt produktspekter. System Sikkerhet ønsker å la honeypot-konseptet inngå som del av en komplett sikkerhetsløsning, og ønsker derfor å styrke egen kompetanse innefor dette emnet.

## 1.3 Avgrensninger av oppgaven

Etter samtaler med System Sikkerhet har jeg valgt å konsentrere utviklingen mot en honeypot som kan simulere flere ulike servere.

## 1.4 Struktur

Rapporten har 3 hoveddeler. Kapittel 3 tar for seg ulike aspekter ved datasikkerhet. En fullstendig gjennomgang av dette emnet ligger over ambisjonsnivået til denne rapporten, men jeg har forsøkt å legge et grunnlag for resten av rapporten.

Kapittel 4 tar for seg honeypots. Jeg begynner med å gi en innføring i hva som ligger i begrepet honeypot, og hvordan de brukes. Videre følger en gjennomgang av de ulike konseptene, og deres egenskaper, fordeler og ulemper. Til slutt i dette kapittelet vil jeg se på ulike tiltak som kan bidra til å bedre funksjonaliteten til honeypoten.

Kapittel 5 viser utviklingen av en prototype på en honeypot som simulerer flere servere på en maskin. Kapittelet tar for seg de ulike operasjonene som må utføres ved oppsettet av denne maskinen, de ulike tjenestene som kjøres og

resultatene som oppnås ved å bruke rekognoseringsverktøy mot serveren. Til slutt kommer en vurdering av løsningen.

# 2 Metode

## 2.1 Teoretisk grunnlag

Arbeidet med oppgaven begynte med en litteraturstudie. For å kunne utføre oppgaven, måtte jeg sette meg inn i tre noe overlappende fagområder:

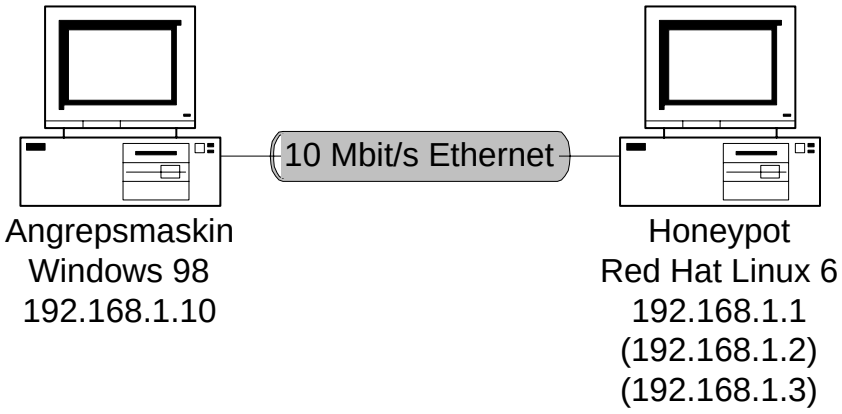
- Generell datasikkerhet
- Honeypotkonseptet
- Bruk og oppsett av Linux

Til hjelp med de to første punktene har jeg stort sett brukt kilder på Internett. Jeg er klar over at det kan være noe problematisk å bruke referanser fra Internett fordi disse kan være flyktige, men har allikevel valgt å bruke disse kildene.

Når det gjelder bruk og oppsett av Linux, har jeg i utgangspunktet brukt bøkene *Red Hat Linux Secrets*[1] og *Boken om Linux*[2]. I tillegg har jeg brukt dokumentasjonen som følger med Linux-distribusjonen.

## 2.2 Praktisk arbeid

For å sette opp prototypen og teste denne, satte jeg opp et testnettverk. Dette nettverket består av to maskiner. Figur 1 viser dette oppsettet.
Angrepsmaskinen kjører Windows 98, og honeypoten kjører Red Hat Linux 6.




**Figur 1 Testnettet som ble satt opp.**

# 3 Datasikkerhet

## 3.1 Hva er datasikkerhet?

Det er vanlig å bruke 3 kriterier for hva som utgjør datasikkerhet[3]:

- *Confidentiality*; tilgangen til konfidensiell informasjon skal forbeholdes dem som har autorisasjon til å benytte den. En trenger derfor sikkerhetsmekanismer som sjekker identiteten til brukeren før informasjonen utleveres.

- *Integrity*; for at informasjon skal kunne brukes til avgjørelser, må brukeren kunne stole på at den er riktig. En må derfor ha mekanismer som garanterer at informasjonen er korrekt til enhver tid, og som hindrer uautorisert endring.

- *Availability*; for at informasjon skal kunne brukes til avgjørelser, må den være tilgjengelig til enhver tid. Det er derfor påkrevd mekanismer som bidrar til en så høy oppetid som overhode mulig, og som får systemet raskt opp igjen etter uunngåelige brudd.

Med mekanismer i avsnittene over menes den blandingen av rutiner, hardware og software som er nødvendig for tilfredsstille de krav til datasikkerhet som organisasjonen har vedtatt i sin sikkerhetspolicy. Dette kan være krav om passord av en viss styrke, om data skal være kryptert, hvor ofte det skal taes backup av dataene, om serveren skal ha batteribackup i tilfelle strømbrudd eller om serveren skal være låst inne i et eget rom.

Med *angrep* eller *trussel* menes normalt en handling som truer gyldigheten av ett eller flere av kriteriene over. Også den rekognoseringen som ofte finner sted før selve hovedangrepet, regner jeg med under betegnelsen angrep. Det er verdt å merke seg at denne handlingen ikke nødvendigvis trenger å skje med

overlegg, også uvitenhet og skrivefeil i kommandoer kan føre til brudd på kriteriene.

## 3.2 Hvorfor beskytte seg

Det finnes mange grunner for at organisasjoner bør beskytte seg mot angrep. Det mest opplagte er at en beskytter seg for å opprettholde datasikkerheten. Det kan få store konsekvenser for organisasjonen dersom det forekommer brudd på de tre kriteriene skissert over. Organisasjonens troverdighet kan settes på spill dersom konfidensiell informasjon, spesielt om en tredjepart, kommer på avveie. I tillegg kan konkurrenter få innsyn i bedriftshemmeligheter, som forskningsresultater og viktige strategiske avgjørelser.

I tillegg til å beskytte organisasjonens informasjon fra innsyn, er det vel så viktig å beskytte den mot endring eller sletting. Endring av informasjon varierer fra det relativt harmløse, som å forandre på en internettside der organisasjonen presenterer seg, til det katastrofale, som å endre på pasientjournaler eller endre oppskrifter på medisiner.

Mange organisasjoner driver med tidskritiske aktiviteter, og for disse er det meget viktig at informasjon kan finnes til enhver tid. Det er derfor nødvendig å sørge for at organisasjonens datanett klarer å motstå angrep som har som hensikt å knekke dette. Også troverdigheten til organisasjonen som tjenesteleverandør synker dersom kunder stadig ikke oppnår kontakt med serveren.

Det er ikke bare organisasjoner med tidskritiske applikasjoner eller konfidensiell informasjon som må tenke datasikkerhet. Alle som har datamaskiner tilknyttet et offentlig nett kan risikere å bli angrepet. Selv om organisasjonen ikke selv blir angrepet, kan maskinene bli brukt som utgangspunkt for andre angrep. Dersom disse angrepene blir sporet tilbake til organisasjonen, kan denne risikere å bli holdt ansvarlig.

System Sikkerhet har i 1999 foretatt en sikkerhetsovervåking hos Kredittkassen[4]. Denne rapporten viser en klar økning av uønsket trafikk gjennom året. Det andre poenget som er verdt å ta med seg fra denne rapporten, er uønsket trafikk fordelt på land. De 3 øverste er USA, Ungarn og Korea. Norge følger først på 5 plass med kun 6 % av den totale trafikken. Dette viser at datakriminalitet er et globalt problem, og at angrep vel så gjerne kan komme fra Asia som fra Norge.

## 3.3 Hvem beskytter en seg mot

Dagens trusselbilde er ikke alltid like enkelt å forholde seg til. En skiller ofte mellom angrep fra utsiden og angrep fra innsiden. I begge gruppene finnes det ulike motiver for å utføre angrepene. Gruppen på innsiden består av personer som har tilknytning til organisasjonen. I en undersøkelse fra *Datapro Research* referert i [3] hevdes det at 81 % av angrepene skyldes personer ansatt i organisasjonen, mens 6 % skyldes tidligere ansatte. I samme undersøkelse hevdes det også at menneskelige feil er skyld i 52 % av tilfellene, mens bare 20 % skyldes uærlighet og teknisk sabotasje.

Den utenforstående gruppen er alle som ikke har tilknytning til organisasjonen.. Dette er en mer heterogen gruppe enn dem med tilknytning til organisasjonen. Her finner vi alle, fra såkalte scriptkiddies til profesjonelle crackere. Scriptkiddies er en betegnelse på folk som kjører ferdig lagde script som utnytter svakheter i systemet som angripes. Utrykket cracker blandes ofte med uttrykket hacker, spesielt i massemedia. Det råder ikke full enighet om hva som er forskjellen på disse to uttrykkene, men en fin definisjon er: "*The basic difference is, hackers build things; crackers break them.*"[5] I denne rapporten bruker jeg det norske uttrykket *inntrenger* om alle som prøver å komme seg inn på systemer de ikke har rettmessig adgang til. Jeg vil også omtale inntrengeren som *han*, selv om dette selvfølgelig ikke alltid er tilfelle.

I tillegg til å se på hvem som angriper, må en se på motivet til inntrengeren. Det finnes mange ulike motiver for å drive på med angrepene. For mange vil det å klare å bryte seg inn i et system være selve drivkraften. Når de så har klart

det, vil de enten skjule at de har vært der, eller skryte av det. I tillegg finnes også de som rapporterer svakheter til dem de har brutt seg inn hos. Det finnes flere eksempler på at websider til kjente organisasjoner og bedrifter blir endret.[6]

I den andre enden av skalaen finner vi dem som bevisst går inn for å sabotere eller stjele data. Denne gruppen er verre å forholde seg til, fordi motivene er så varierte. Gruppen kan inkludere folk som har interessekonflikter med organisasjonen, er ute etter hevn eller som er betalt av konkurrenter.

## 3.4 Ulike typer angrep

Det finnes mange ulike typer angrep, men de kan deles inn i noen grunnleggende typer.

### 3.4.1 Rekognosering

Noe av det første en eventuell inntrenger gjør, er å skaffe seg oversikt over hvilke maskiner som finnes og hvilke tjenester disse kjører. Nå kan det vel diskuteres om dette egentlig kan kalles angrep, og om lovligheten av å bruke disse metodene, men populært hos systemadministratorene er det ikke. En slik rekognosering er ofte et fortegn på at det kommer noe mer. Det finnes litt ulike definisjoner på hva de ulike metodene innebærer, men en brukbar definisjon følger her.

- *Forsøk på oppkobling* er forespørsel mot en port på en enkelt maskin.
- *Sweeps* er forespørsler mot samme port på minst to ulike maskiner. Dette brukes ofte til å finne maskiner som kjører en bestemt tjeneste, men kan også brukes til å finne maskiner som er i live.
- *Skanninger* er forespørsler mot minst 2 ulike porter på samme maskin. Det kjøres ofte skanninger mot flere maskiner samtidig. Dette brukes for å kartlegge hvilke tjenester som kjører på de ulike maskinene.

### 3.4.2 Denial of Service (DoS)

*Denial of Service* angrep har som mål å hindre tilgang til legitime tjenester hos en maskin. Denne maskinen blir typisk bombardert med falske forespørsler som opptar all dens kapasitet, og dermed hindrer reell bruk. Typiske eksempler på dette er SYN-flood angrep og Ping of Death.[7]

### 3.4.3 Trojanske hester

*Trojanske hester* er programvare skjult inne i annen programvare. Vertsprogramvaren ser som regel ganske uskyldig ut, men når denne kjøres installeres den trojanske hesten uten at det merkes. Den trojanske hesten kjøres i bakgrunnen, og åpner en bakdør til systemet. Trojanene kan ofte også ha annen uønsket funksjonalitet. NetBus og Back Orifice er eksempler på denne typen angrep.

### 3.4.4 Kjente tjenester

Mesteparten av all programvare inneholder svake punkter. Dette kan være dårlig design eller rene programmeringsfeil, også kalt bugs. De aller fleste programmer inneholder bugs av ulike slag. Noen av disse er av en slik art at de kan brukes til å skaffe inntrengeren en bakdør til maskinen. I tillegg så er mange av de protokollene som brukes designet før nettverkssikkerhet ble et hett tema, og er derfor lite egnet for bruk i et høyrisiko miljø. Metoder for å utnytte slike svakheter kalles ofte for *exploits*. Exploit World[8] inneholder en liste over exploits i forskjellige operativsystemer og applikasjoner.

### 3.4.5 Interne angrep

Angrepsmåtene nevnt over brukes ofte når angrepet kommer utenfra. Men angrepene kan også komme fra innsiden. Ved interne angrep har inntrengeren fysisk tilgang til maskinen, og dermed andre muligheter for å skaffe seg tilgang til data på maskinen. En av disse er å starte maskinen på nytt med en bootdisk, og benytte seg av andre innstillinger av sikkerhetsnivået. Inntrengere som jobber lokalt har også muligheten til å ødelegge eller fjerne hardware på maskinen.

## 3.5 Tegn på at en er angrepet

Hvordan kan en finne ut om et system er kompromittert? Dette kan være et vanskelig spørsmål. En inntrenger vil i utgangspunktet se ut som en ordinær bruker av systemet, og etter at han har rottilgang til maskinen, vil han mest sannsynlig skjule at han i det hele tatt er på maskinen. Men selv om det ofte er vanskelig, finnes det ting en kan se etter.

### 3.5.1 Trafikkmønster

Ved å studere trafikkmønsteret til en maskin kan en finne ut ganske mye. Plutselige endringer i bruken er et tegn på at det kan være noe som ikke stemmer. Et annet er trafikk på tider av døgnet det normalt skal være rolig. Er det mye trafikk på en maskin etter at alle brukerne er gått for dagen, er det et tegn på at noen utenfra kontakter maskinen. Dette kan selvfølgelig være legitime brukere men det er verdt å undersøke. En ting som er viktig å huske på, er at en inntrenger kan sitte hvor som helst på kloden, og at det derfor ikke er sikkert at angrepet skjer på natta. Angrepet kan like gjerne skje midt på dagen mens det er full aktivitet på maskinen.

En kan også se på tjenestene som kjøres. Dersom det er trafikk på tjenester som egentlig ikke skulle være aktive på den aktuelle maskinen, er det et tegn på at maskinen er kompromittert og at inntrengeren har startet nye tjenester.

### 3.5.2 Rootkits

Noe av det første en inntrenger gjør etter at han har fått tilgang, er å kamuflere seg. Dette gjøres ofte ved å bytte ut en del binærfiler med modifiserte utgaver. En samling av slike modifiserte filer kalles ofte for et *rootkit*. Disse binærfilene, som kan være både programmer og delte bibliotekfiler, modifiseres til å skjule visse brukere og deres filer, og ellers alt annet som kan være nyttig for inntrengeren.

For de ordinære brukerne av disse filene vil de ha nøyaktig samme funksjonalitet, så hvordan kan en oppdage at maskinen er infisert med et

rootkit? Alt etter hvor godt laget filene er, er det ulike måter å oppdage dem på. Det mest innlysende er filstørrelse. Ettersom funksjonaliteten er endret, vil filstørrelsen variere. Ved å vite hvor store de rene filene er, kan disse lett sammenliknes. Dersom de er bedre laget, og matcher på størrelsen, kan en bruke sjekksummer til å sammenlikne dem. Det finnes programmer som gjør dette, og et av dem er Tripwire. Denne lager en database over filer som skal beskyttes, og siden kan disse filene sammenliknes med verdiene i databasen.[9]

### 3.5.3 Loggfiler

De fleste operativsystemer har mulighet til å drive med logging av mye av det som skjer på serveren. Ved å bruke disse loggemulighetene kan en finne ut mye om hvilken bruk systemet utsettes for. Bruk av disse loggene kan vise om systemet utsettes for uautorisert bruk. Det som er verdt å merke seg, er at loggingen og loggfilene er noe av det første en inntrenger endrer på et kompromittert system. Ved å logge til flere ulike filer, gjerne på ulike maskiner, kan en se om disse filene er like, og dermed bedømme om systemet muligens kan være kompromittert.

## 3.6 Måter å beskytte seg på

Selv om en aldri kan beskytte seg helt, finnes det måter å redusere risikoen på. Generelt kan en si at jo bedre en beskytter seg, desto flere av de mindre kunnskapsrike inntrengerne vil bli stoppet, og en kan konsentrere seg om dem som virkelig er gode.

### 3.6.1 Firewalls

Generelt kan en si at firewalls er maskiner som filtrerer ut nettverkstrafikk etter gitte regler. Firewalls plasseres normalt ved en organisasjons tilkoblingspunkt til det offentlige nettet, eller ved interne nettverkssegmenter som skal beskyttes ekstra. All trafikk som skal slippe gjennom blir vurdert opp mot firewallens policy-fil, og pakker med protokoller eller adresser som ikke tillates blir slettet. Jo flere protokoller som kan stoppes, desto bedre er det, men mange organisasjoner opererer servere som allmennheten skal ha tilgang til, som for

eksempel webservere eller ftpservere. Dermed så må en slippe gjennom noe trafikk, og systemet er utsatt for en risiko. Dessuten så er selve firewallen også utsatt for angrep.

### 3.6.2 Dedikerte nettverk

For å hindre innbrudd i sensitivt materiale, er det mange bedrifter som kjører 2 parallelle nettverk, ett som er åpnet opp mot det offentlige nett, og ett skjermet nettverk som kun kjøres lokalt. Det åpne nettverket er like utsatt for angrep og følgene av det som før, mens det skjermede nettverket kun er utsatt for interne angrep. Ulempene med dette systemet er at de ansatte har to maskiner å forholde seg til, og at de materielle kostnadene i utgangspunktet blir dobbelt så høye. Dette er imidlertid lite i forhold til det et innbrudd kan koste, både i ekstra arbeid, tap av data og ikke minst tap av anseelse i markedet.

### 3.6.3 Kunnskap

Både firewalls og dedikerte nettverk er til liten hjelp dersom inntrengeren har tilknytning til organisasjonen. Firewalls fordi inntrengeren i utgangspunktet er innenfor, og dedikerte nett fordi inntrengeren jo også har tilgang på dette. At det i begge tilfellene vil være lettere å oppspore inntrengeren er en annen diskusjon.

En av de beste måtene en kan redusere risikoen for innbrudd i nettverket er kunnskap. Både kunnskap om eget nettverk og om inntrengerens arbeidsmåter er viktig. Den eksplosive veksten i antall datamaskiner de siste årene har ført til en generell nedgang i kompetanse hos brukerne. Dette har igjen ført til krav om økt brukervennlighet. Denne brukervennligheten har ofte kommet på bekostning av sikkerheten. Mulighet for å endre forholdet mellom sikkerhet og brukervennlighet kan godt være tilstede i produktet, men brukeren vet ikke at den eksisterer og bruker følgelig standardinnstillingen. Med tilstrekkelig kunnskap om nettverket og programvare som brukes, kan systemadministratoren finne den balansen mellom brukervennlighet og sikkerhet som gir det ønskelige sikkerhetsnivået.

Dersom en bruker en standard installasjon av et operativsystem, vil det ofte startes mange flere tjenester enn dem man trenger. Jo flere tjenester som kjøres, desto flere potensielle sikkerhetshull finnes det. Ved å kun kjøre tjenester en trenger eliminerer man en del av risikoen. I tillegg til å redusere antall tjenester, er det viktig å hele tiden kjøre den sikreste utgaven av de tjenestene en bruker. Det oppdages hele tiden sikkerhetshull i de ulike applikasjonene, og disse hullene tettes med midlertidige oppdateringer av programvaren. Disse oppdateringene kalles ofte for *patches*. Ved å følge med på buglister[10], får en vite om nye sikkerhetshull og svakheter som oppdages.

# 4 Honeypots

## 4.1 Generelt om Honeypots

Før en går nærmere inn på hvordan honeypots kan brukes til å bedre sikkerheten i nettverket, kan det være greit å komme med en forklaring på hva som ligger i begrepet honeypot. Honeypots er datamaskiner eller nettverk uten noen virkelig funksjon i produksjonsnettet til en organisasjon. På samme måte som honning og andre søtsaker tiltrekker seg insekter, er det meningen at honeypots skal tiltrekke seg uønskede elementer i datanettet. De går også under en rekke andre betegnelser som blant annet decoys eller lokkeduer.

## 4.2 Et typisk oppsett

I dette avsnittet vil jeg vise hvordan et typisk oppsett kan se ut. Det er verdt å merke seg at dette kun er en mulig løsning, og det finnes et utall andre måter å gjøre det på. Det sentrale i systemet er den maskinen som fungerer som selve honeypoten. Det er denne maskinen som vil være målet for en inntrenger. Resten av nettverket skal være så transparent som mulig. Honeypotmaskinen vil være sårbar for angrep, og vil sannsynligvis etter hvert bli kompromittert. Etter at maskinen er kompromittert, kan en ikke lenger stole på loggene fra denne maskinen. For å skjule sporene etter seg laster inntrengeren ofte ned modifiserte utgaver av programmene som kjøres på honeypoten. Et av disse programmene er systemloggeren. For at en skal kunne ha bedre kontroll med hva som foregår på denne maskinen kan det derfor være lurt å ha flere lag med logging enn kun internt på honeypoten.

Det første ekstra laget som kan være fornuftig å sette opp er en firewall som skjermer honeypoten fra omverdenen. Denne firewallen har flere funksjoner. For det første så skal den brukes til logge trafikken som kommer inn til honeypoten. Ved å sette opp honeypoten på et eget segment bak firewallen, vet en at all trafikk som logges er trafikk som er ment for honeypoten. Og siden

honeypoten ikke skal ha noen legitim trafikk, så er den trafikken som logges i utgangspunktet mistenkelig. Det skal også nevnes at selv om det registreres trafikk til honeypoten, så er det ikke sikkert at dette er forsøk på å kompromittere maskinen. Det kan også være noen som ganske enkelt har skrevet en IP-adresse feil og tilfeldigvis truffet honeypoten.

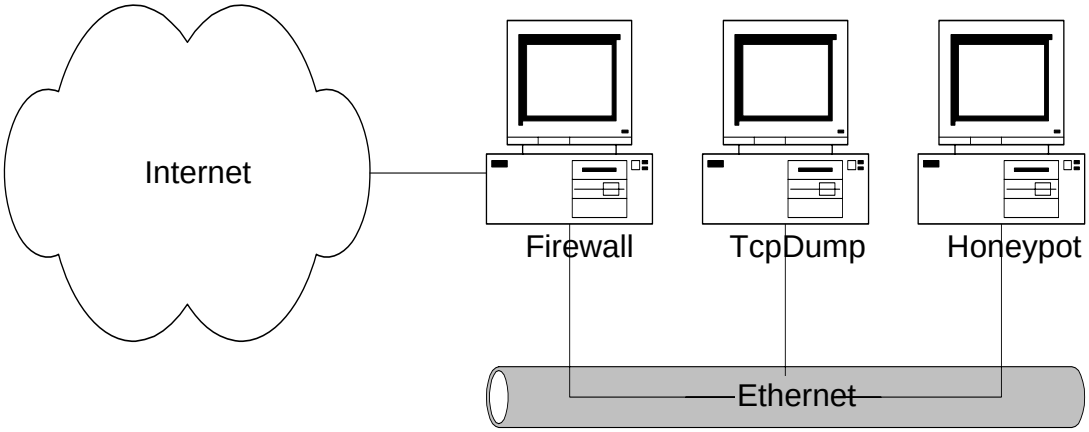
I tillegg til å logge innkommende trafikk til honeypoten, har firewallen også en sekundær oppgave. Dersom en inntrenger klarer å kompromittere honeypoten, vil han ofte bruke denne som utgangspunkt for angrep av andre systemer. Det kan få store følger for eieren av honeypoten dersom et angrep kan spores tilbake til denne. Ved å la all innkommende trafikk slippe gjennom og sperre for det meste av utgående trafikk kan en langt på vei hindre at honeypoten blir brukt som base for videre angrep. Ved å tillate f.eks. utgående ftp, vil inntrengeren ha tilgang til å hente sine rootkits og annet han måtte trenge for å kompromittere maskinen. Dersom all utgående trafikk sperres, vil det virke mistenkelig.

Selv om firewallen logger trafikken som går til honeypoten, sier den ingenting om innholdet i de pakkene som slipper gjennom. Dersom en ønsker å se i detalj på hvordan en inntrenger jobber ved å studere hvilken kommunikasjon som flyter mellom honeypoten og inntrengeren, kan det være lurt å installere en pakkedumper. Denne maskinen har et nettkort som opererer i promiscuous mode. Det vil si at den mottar all trafikk på nettverkssegmentet. Ved å bruke passende programvare, som for eksempel TcpDump , kan en lagre alle disse pakkene på harddisken, for siden å gjennomgå dem. Det skal bemerkes at dette ikke er til like stor hjelp dersom inntrengeren har mulighet til å benytte seg av krypterte tjenester.

Etter at en maskin er kompromittert er det vanlig at inntrengeren laster ned et rootkit som endrer på noen av de viktigste programmene. For å finne ut hvilke filer som er endret, kan det lønne seg å bruke en integritetssjekker. Denne bygger opp en database over egenskapene til de ulike filene. Etter at maskinen er kompromittert kjøres programmet en gang til og sammenlikner filene med

det den har i databasen. Eventuelle uoverensstemmelser blir rapportert, og en kan finne filene som er endret. Et slikt program er Tripwire.

Oppsettet skissert over er vist i figur 2. Dette er bare en av mange måter å sette opp et honeypotsystem på. Om en vil kalle hele dette systemet for en honeypot, eller la begrepet kun gjelde for selve lokkemaskinen, er vel en definisjonssak. Jeg vil i de følgende kapitlene bruke begrepet honeypot om kun selve lokkemaskinen, og det vil da være underforstått at resten av systemet rundt er med. Dersom jeg refererer til hele systemet som enhet, vil jeg bruke honeypotsystem.




**Figur 2 En mulig implementasjon av et honeypotsystem, her vist med firewall, pakkedumper og selve honeypoten på et ethernetsegment.**

I tillegg til oppsettet i figur 2, kan en også sette inn maskiner med andre oppgaver. Dette vil typisk være maskiner som kjører en eller annen form for Intrusion Detection System, eller flere honeypots som kjøres parallelt.

## 4.3 Hva kan de brukes til

Etter å ha sett på hvordan et typisk honeypotsystem ser ut, er det på tide å finne ut hva det kan brukes til. Bruksområdene til en honeypot kan være mange. Ved å sette opp en honeypot som tilsynelatende virker mer attraktiv enn produksjonssystemet til organisasjonen, kan en lokke en inntrenger til å bruke tid på honeypoten framfor produksjonssystemet. Det kan gi

systemadministratoren litt ekstra tid til å oppruste forsvaret, ved blant annet å sperre for den IP-adressen som inntrengeren opererer fra.

Når en først har en inntrenger på honeypoten, kan det være en god ide å samle bevis med tanke på å straffeforfølge inntrengeren. Nå skal ikke jeg her ta stilling til hva som trengs for å få noen tiltalt for datakriminalitet, men alt som inntrengeren foretar seg bør registreres i loggfiler. For å få best mulig logging skal en helst logge på flere ulike nivåer. På denne måten får en lettere med seg alt som skjer, og det er verre for inntrengeren å skjule noe.

En tredje ting honeypots kan brukes til er å studere hvordan en inntrenger jobber. Ved å studere loggfiler, eller følge med live kan en se hvordan inntrengeren går fram for å knekke et system. Kunnskap om dette kan i etterkant brukes for å sikre de virkelige systemene for samme type angrep ved å tette sikkerhetshull som blir avdekket.

## 4.4 Forskjellige kategorier

Det finnes slik jeg ser det to ulike kategorier av honeypots. Den første kategorien består av programvare som kan simulere ulike tjenester på en maskin. Det finnes flere programmer av denne typen, men felles for dem er at de overtar styringen av enten deler av eller hele vertsmaskinens nettverksvirksomhet. Når disse maskinene blir forsøkt kontaktet, returnerer de et svar som skal lure inntrengeren til å tro at de simulerte tjenestene virkelig eksisterer på maskinen. Disse svarene kan enten være statiske, eller genereres dynamisk ut fra et script. Dette konseptet vil gå under betegnelsen programvarebasert honeypot.

Den andre typen er den som bruker et ordinært operativsystem, som Windows NT eller Linux. I disse systemene er faktisk honeypoten en reell server på lik linje med organisasjonens produksjonsservere, og har de samme svakhetene som disse. Forskjellen er at honeypoten ikke bidrar med reelle tjenester. Jeg vil i de følgende kapitlene omtale dette som reelle honeypots.

## 4.5 Systemer som implementerer honeypots

Det finnes noen kommersielle systemer som implementerer en honeypot-funksjon. Et utvalg av disse er Cyber Cop Sting fra Network Associates, Deception Toolkit fra Fred Cohen & Associates og Back Officer Friendly fra Network Flight Recorder. Alle disse er programvareimplementasjoner som simulerer ulike servere. Beskrivelsen av disse er basert på informasjon fra produsentenes internettsider, og ikke på personlig bruk.

### 4.5.1 Cyber Cop Sting

Cyber Cop Sting[11] er et produkt fra Network Associates. Produktet er en del av Network Associates Cyber Cop Monitor Intrusion Detection system. Det har muligheten til å simulere et virtuelt nettverk som består av Windows NT og Solaris servere og Cisco rutere. Cyber Cop Sting vil være skjult for normale brukere av systemet, og bare oppdages dersom en kjører rekognoseringsverktøy mot nettverkssegmentet. På denne måten vil det ikke forstyrre den daglige driften. Ved å simulere flere lag i IP-stakken, kan Cyber Cop Sting imitere såkalte OS fingerprints som benyttes av inntrengere til å finne ut hvilket operativsystem som brukes. Cyber Cop Sting kjører på Windows NT-plattformen.

### 4.5.2 Deception Toolkit (Dtk)

Deception Toolkit[12] fra Fred Cohen And Associates er en samling små C-programmer og Perl-script. Et av disse Perl-scriptene lytter på portene som skal brukes. Dersom de blir forsøkt oppkoblet, vil dette Perl-scriptet gi respons ut fra oppsatte endelige tilstandsmaskiner. På denne måten kan Dtk programmeres til å gi en hvilken som helst respons. For en angriper vil det se ut som systemet har en masse kjente svakheter. Fra versjon 1999-08-18 har Dtk mulighet til simulere ulike IP-adresser fra samme maskin.

Folkene bak Dtk har en ide om at port 365 skal bli kjent som Deception porten, og at det skal kunne kjøres en tjeneste her som forteller at Dtk kjøres. Håpet er da at dette skal kunne skremme vekk potensielle angripere, ved at de først

sjekker på port 365. Selvfølgelig skal dette være mulig å slå av slik at en kan lure angriperen

Dtk kommer som kildekode, og kan kjøres på maskiner der det finnes et Perl-miljø.

### 4.5.3 Back Officer Friendly

Back Officer Friendly[13] er et produkt fra Network Flight Recorder, og har som hovedmål å simulere den trojanske hesten Back Orifice. Back Orifice er et program som gir inntrengere full kontroll over den maskinen som er infisert. Ved å utgi seg for å være en Back Orifice server, kan Back Officer Friendly varsle og logge forsøk på å ta seg inn i maskinen. Inntrengeren får hele tiden sannsynlige svar i retur. I tillegg kan også Back Officer Friendly simulere andre nettverkstjenester, som FTP, HTTP og SMTP.

Back Officer Friendly er tilgjengelig for Windows, og det finnes også en noe redusert utgave for Unix.

### 4.5.4 Operativsystemer

De fleste operativsystemer kan brukes til honeypots, men noen er bedre egnet enn andre. For at en skal kunne følge en inntrengers bevegelser på maskinen er det viktig at operativsystemet har mulighet til å utføre grundig logging av det som skjer. Typiske serveroperativsystemer som Unix, Linux og Windows NT gjør seg derfor bedre enn operativsystemer ment til hjemmebruk og andre mindre kritiske bruksområder. Windows 95/98 havner i den siste kategorien.

## 4.6 Sammenlikning av konseptene

Som omtalt i kapittel 4.4 finnes det to hovedtyper av honeypots, de reelle og de som benytter seg av programvareimplementasjoner. Begge konseptene har sine fordeler og ulemper. Jeg vil her komme med noen sammenlikninger av disse to konseptene innenfor noen bruksområder.

### 4.6.1 Programvare

Programvareimplementasjoner av honeypots er programmer som simulerer nettverksfunksjonalitet. Ideen bak disse er at en eventuell inntrenger skal tro han jobber inne på maskinen, mens han i realiteten blir stoppet ved døra og blir servert løgner. En av fordelene med dette er at maskinen ikke blir virkelig kompromittert, og en har kontroll på hva som skjer. Inntrengeren får i utgangspunktet ikke gjort annet enn det honeypoten er i stand til å gjenkjenne av forespørsler

For at denne kategorien av honeypots skal kunne presentere en sannsynlig respons på en forespørsel, må den aktuelle forespørselen på en eller annen måte være programmert på forhånd. Denne programmeringen kan enten hardkodes inn i programvaren, eller den kan lages som et eksternt script. Ved å hardkode oppførselen er en avhengig av å gi ut ny versjon av programmet hver gang en skal legge inn ny oppførsel. Den script-baserte honeypoten er lettere å oppdatere. Her kan en bare legge ut nye script og endre konfigurasjonsfilene til å ta hensyn til den nye oppførselen.

I tillegg til å kjøres som rene honeypots, kan disse programmene også kjøres parallelt med de reelle tjenestene i organisasjonens produksjonsnett. En server kan da tilsynelatende kjøre mange tjenester, men kun et fåtall er reelle. Poenget er da at inntrengeren statistisk sett skal bruke lengre tid på å knekke maskinen. Dette kan slå til hvis inntrengeren velger en av de falske tjenestene først. Velger han derimot en av de reelle tjenestene først, er det ikke noe hjelp å få av de falske.

### 4.6.2 Reelle

Det jeg har kalt reelle honeypots, er honeypots som består av et ordinært operativsystem som kjører reelle tjenester. En av de største fordelene med disse er at de kan konstrueres slik at de nøyaktig imiterer organisasjonens produksjonssystem, og at de derfor kan være med å avsløre svakheter og sikkerhetshull i produksjonssystemet.

For å benytte reelle tjenester kreves det en del forhåndskunnskap om oppsett av nettverksservere. Dette er normalt ikke noe problem, fordi folk som har interesse av honeypots ofte er systemadministratorer eller andre med interesse for nettverket. Når en først har denne kunnskapen er det lett å få tjenestene til å fremstå som troverdige, i og med at det faktisk er de reelle tjenestene som kjøres.

Det faktum at det er reelle tjenester som kjører, gjør at maskinen er mottakelig for misbruk. Hele dette konseptet legger egentlig opp til at maskinen skal kunne bli overtatt av inntrengeren. Derfor må en være 100 % sikker på at tjenestene ikke kan misbrukes videre. Dette krever en god kontroll av miljøet rundt honeypoten, med blant annet et sikkert oppsett av firewallen.

### 4.6.3 Læring

Et av formålene med å bruke honeypots, er å lære inntrengernes arbeidsmetoder. Til dette brukes loggfilene som systemet genererer. De reelle honeypotene vil gi systemloggfiler som gjenspeiler det som har skjedd på maskinen, mens de programvarebaserte logger det som inntrengeren prøver å gjøre. Dersom ikke den programvarebaserte honeypoten er programmert til å takle en gitt exploit, vil en ikke nødvendigvis få en realistisk log av resultatet. Det vil man derimot med den reelle honeypoten.

Med de programvarebaserte honeypotene må en på forhånd vite hvordan et program må oppføre seg ut fra en gitt exploit, mens med de reelle finner en det ut etter at angrepet er utført. Nå vil jo leverandøren av en programvarebasert honeypot komme med nye script og oppdateringer slik at man slipper å lage alle selv. Det betyr at man kan godt ha en brukbar honeypot selv om man ikke ligger i front når det gjelder kunnskap om nye expolits.

Både med reelle og programvarebaserte honeypots har en muligheten til å begynne i det små, med bare en eller noen få tjenester, for så gradvis bygge opp kunnskapen. Jeg mener allikevel at de reelle honeypotene er bedre egnet til å tilegne seg kunnskap ifra, fordi det er lettere å sette opp et hva-om scenario: prøve seg fram med ulike, og ofte unike, konfigurasjoner, og så se hvordan

systemet reagerer på angrep. I programvarebaserte løsninger vil reaksjonen allerede være programert inn i scriptene.

### 4.6.4 Oppsett

De to ulike konseptene blir satt opp på to vidt forskjellige måter. Når det gjelder de programvarebaserte, så består disse ofte av en enkelt pakke som skal installeres. Ved å hente ned nye script fra leverandøren, kan en enkelt oppgradere funksjonaliteten til honeypoten.

De reelle honeypotene krever en del mer arbeid for å fungere. I tillegg til at de skal settes opp til å fungere på samme måte som en ordinær server, må en også passe på at mest mulig blir logget. Videre må en ha en gjennomgang av maskinen hver gang maskinen blir kompromittert. For å oppgradere funksjonaliteten, må en sette opp nye programmer, eller nye versjoner av de eksisterende programmene. Dette medfører ofte å få et utall ulike konfigurasjonsfiler til å samarbeide, noe som ikke alltid er like enkelt.

### 4.6.5 Realisme

En programvarebasert honeypot vil være så realistisk som det kvaliteten på scriptet tilsier. Er det gjort en god jobb med scriptet, vil funksjonaliteten ligge nært opp til originalen. Problemene oppstår derimot når honeypoten utsettes for angrep det ikke er tatt høyde for i scriptet. Disse problemene unngår en ved å bruke en reell honeypot. Siden det er den virkelige tjenesten som kjører, vil enhver respons være realistisk.

### 4.6.6 Distraksjon

I tillegg til læring, kan honeypoten brukes til å lede oppmerksomheten vekk fra produksjonssystemet. Til dette formålet mener jeg programvarebaserte løsninger er det beste. Det krever minst kunnskap, er lettest å sette opp og krever minst vedlikehold.

## 4.7 Videreutvikling

Et av delmålene med denne oppgaven er å se på hvordan en honeypot kan videreutvikles for å bli mer funksjonell. Jeg vil ta utgangspunkt i det honeypotsystemet som er skissert i kapittel 4

### 4.7.1 Forbedre scenariet

Selve scenariet som honeypoten opererer under må gjøres så realistisk som mulig. Dette kan fort bli en ressurskrevende arbeidsoppgave, alt etter hvor mye en vil gjøre ut av det. Det finnes to ulike retninger en kan følge. Den første er å fylle tjenestene med innhold. En ftpserver fylt med tilsynelatende hemmelige dokumenter, vil ofte virke mer tiltrekkende enn en server med for eksempel kakeoppskrifter. Ved å bruke litt tid på troverdige, falske dokumenter, og samtidig ikke bruke altfor harde passord, kan en muligens holde en litt for nysgjerrig konkurrent på avstand.

Den andre retningen går ut på å la honeypoten se ut som en gjenglemt maskin med mange sikkerhetshull. Dette vil ofte tiltrekke seg inntrengere på jakt etter et lett bytte som de kan bruke som mellomstasjon for andre angrep.

### 4.7.2 En maskin som simulerer et helt segment

Ettersom serverbehovene til en organisasjon vokser, er det normalt å fordele oppgavene på flere servere, som hver får sine spesielle oppgaver. Dette krever at antallet servere økes. Dersom en har den ambisjonen at honeypoten skal gjenspeile produksjonssystemet, er det også ofte nødvendig å utvide antall maskiner i honeypotsystemet. Men i og med at hver maskin i honeypotsystemet normalt har veldig lite med reellt arbeid, kan det virke som om det er unødvendig å ha flere maskiner til å gjøre dette. Det ideelle er å ha en maskin som simulerer hele nettverkssegmentet. Ved å gjøre det på denne måten kan en spare både plass og penger.

# 5 Prototype

I dette kapittelet vil jeg ta for meg hvordan en kan gå fram for å sette opp en reell honeypot som simulerer flere servere på et nettverkssegment.

## 5.1 Plan

Før en går i gang med å sette opp en honeypot, må en ha en plan over hvordan denne skal bli seende ut. I denne prototypen har jeg valgt å sette opp et system som består av 3 servere med hver sin hovedfunksjon. Disse tre funksjonene er telnet, ftp og http. Tabell 1 viser hvordan honeypoten er tenkt oppsatt.

| | IP-adresse | Tjeneste | Port | Protokoll |
|---|---|---|---|---|
| Server 1 | 192.168.1.1 | Telnet | 23 | TCP |
| Server 2 | 192.168.1.2 | Ftp | 21 | TCP |
| Server 3 | 192.168.1.3 | Http | 80 | TCP |

**Tabell 1 Oversikt over hvilke tjenester som skal kjøres på honeypoten.**

## 5.2 Linux

Til prototypen har jeg valgt å bruke operativsystemet Linux. Det er flere grunner for dette valget. Linux er et gratis operativsystem som stadig blir mer utbredt, også blant bedrifter. Linux kommer med åpen kildekode, noe som gjør det forholdsvis enkelt å tilpasse det til spesielle bruksområder. Det finnes også mye tilgjengelig informasjon om hvordan Linux kan settes opp for å oppnå ulike ting.

Den versjonen av Linux som brukes er Red Hat Linux 6 med kernel 2.2.5-15. Linux er satt opp med det meste av funksjonalitet, bortsett fra X-Windows. For å ikke bli distrahert av andre tjenester enn dem som er aktuelle etter planen, blir alle andre nettverkstjenester stoppet.

## 5.3 Maskinoppsett

Maskinen som er brukt i prototypen er en Pentium 166MMX med 32 MB ram. Den er utstyrt med 2 harddisker på henholdsvis 10 GB og 2GB. Tabell 2 viser hvordan partisjonene er satt opp. En må merke seg at dette kun er et forslag, og at andre maskinoppsett like gjerne kan brukes.

| Partisjon | Enhet | Størrelse | Kommentar |
|---|---|---|---|
| / | /dev/hda1 | 256 MB | Rotkatalogen, det logiske utgangspunktet for resten av filstrukturen. Inneholder også de katalogene som ikke har egen partisjon. |
| /opt | /dev/hda5 | 1000 MB | Katalog som brukes til programpakker. |
| /home | /dev/hda6 | 512 MB | Hjemmeområde for brukere. |
| Swap | /dev/hda7 | 128 Mb | Linux sitt virtuelle minne. |
| /tmp | /dev/hda8 | 100 MB | Katalog for temporær lagring. |
| /var | /dev/hda9 | 100 MB | Inneholder blant annet loggfiler. |
| /usr | /dev/hda10 | 2000 MB | Inneholder blant annet nyttige programmer og biblioteksfiler. |
| /fake | /dev/hda11 | 5060 MB | Område som skal romme de virtuelle serverene. |
| /div | /dev/hdb1 | 2164 MB | Område som skal romme programmer og konfigurasjonsfiler for prototypen. |

**Tabell 2 Oversikt over partisjonene på prototypemaskinen.**

## 5.4 IP-aliasing

For å sette opp en maskin som skal simulere flere servere, er en avhengig av at systemet kan lytte på flere ulike IP-adresser. Dette går ut over nettverkskortenes normale operasjon, og er ofte omtalt som *promiscuous mode*. I Linux kalles denne funksjonaliteten IP-aliasing. IP-aliasing kan enten brukes som en modul som lastes inn i kernelen etter behov, eller kompilert inn som en del av kernelen. Når IP-aliasing er definert så kan det settes opp virtuelle IP-adresser med denne syntaksen:

```
/sbin/ifconfig eth0:x yyy.yyy.yyy.yyy
```

Ifconfig er det programmet Linux bruker for å sette opp nettverkskortet på maskinen. Nettverkskortet har betegnelsen *eth0*. Dersom det er flere nettverkskort blir de betegnet med *eth1, eth2* osv. X står for nummeret til det virtuelle nettverksadapteret og har verdier fra 0 og oppover. Den reelle IP-adressen settes opp uten dette tallet. Den siste delen er den IP-adressen som skal brukes. Linux har i utgangspunktet mulighet for 256 virtuelle adresser på hvert nettverkskort.

Oppsettet av de virtuelle adressene gjøres i filen */etc/rc.d/rc.local*. Vedlegg A viser denne filen. Her legges innstillinger som kun kjøres en gang uavhengig av hvilket kjørenivå Linux opererer på, og vekslingene mellom disse. Prototypen er satt opp med:

```
/sbin/ifconfig eth0:0 192.168.1.2

/sbin/ifconfig eth0:1 192.168.1.3
```

I tillegg til de virtuelle adressene, har også prototypen den reelle adressen 192.168.1.1. Disse adressene er valgt ut fra RFC 1918 som omhandler reserverte IP-adresser til private nettverk.[14] Før systemet tas i bruk i en reell setting må disse adressene byttes ut med organisasjonens offentlige adresser.

## 5.5 Virtuelle servere

### 5.5.1 Generelt

Etter at maskinen er satt opp med IP-aliasing, er neste trinn å se på selve filstrukturen på maskinen. Siden denne ene maskinen skal kunne simulere flere ulike servere, må det bygges opp flere filsystemer slik at hver virtuelle maskin får sitt område. For at hver av de virtuelle maskinene skal ha samme funksjonalitet som den reelle, må alle viktige kataloger kopieres over i filsystemet til den virtuelle serveren. Dette gjelder både kataloger som inneholder kjørbare programmer, som */bin* og */sbin*, og kataloger med felles biblioteker og andre ressurser, som */etc* og */usr*. I tillegg til disse nødvendige

katalogene, kan en ta med kataloger slik at de nye filstrukturene får et mer realistisk utseende. Dette gjelder kataloger som */lost+found*, */boot* og */home*.

For å automatisere opprettingen av nye filstrukturer har jeg laget et script, */div/bin/newdomain*, som kopierer over de filene som den virtuelle serveren har bruk for. Vedlegg B viser dette scriptet. Når en kopierer filer i Linux har man to ulike måter å gjøre dette på. Det ene er å kun lage en link til den filen man vil kopiere. Ved å gjøre det på denne måten sparer en plass på harddisken fordi selve filen kun lagres ett sted. Dette har imidlertid den ulempen at dersom filen endres, så vil det påvirke begge instansene av den. Dette er ikke ønskelig i dette tilfellet. Derfor bruker scriptet den andre metoden, nemlig ren kopiering, og dermed lage 2 separate sett med filer. På denne måten kan man også selv endre ulike filer for å tilpasse dem til formålet. På prototypen er det satt opp 2 virtuelle servere med:

```
/div/bin/newdomain dummy1

/div/bin/newdomain dummy2
```

Etter at de nye filstrukturene er ferdige og de nødvendige filene er kopiert over, er det tid for å sette opp tjenester og brukere.

### 5.5.2 Server 1: Telnet 192.168.1.1

Grunnen til at den reelle serveren blir stående som telnetserver, er at telnet er en tjeneste som er vanskelig å lure. Ved å kjøre den på den reelle serveren, slipper en unna disse problemstillingene. Det som gjør dette litt risikabelt er at en inntrenger nå kan komme inn på det virtuelle serverene via filsystemet til den reelle serveren. Ved å kjøre telnet på denne må en finne en måte å kamuflere disse katalogene på. Ved å sette alle rettighetene til 0 kommer vi et stykke på vei. Det vi oppnår med dette er at */fake* katalogen ikke kan aksesseres, og at innholdet ikke kan listes opp. Men selve katalognavnet vil vises om en lister opp rotkatalogen. For å gjøre det på denne måten anbefales det å kalle katalogen noe mindre fristende enn */fake*.

Om en velger å endre rettighetene i henhold til avsnittet over, skal en være klar over at dette ikke hjelper dersom inntrengeren klarer å skaffe seg rottilgang. Da får han full tilgang til maskinen uansett. Så for å skjule katalogen da, kan en

lage seg egne rootkits, der visse kataloger ikke blir listet opp. Dette kan hjelpe helt til inntrengeren laster ned sine egne rootkits, noe som de jo ofte gjør. Det beste en kan gjøre er å ikke tillate telnet tilgang til maskinen i det hele tatt. Telnet er tatt med i prototypen for å vise hvordan det er mulig å gjøre det.

### 5.5.3 Server 2: Ftp 192.168.1.2

Server 2 er den første av de virtuelle serverene, og finnes på */fake/dummy1*. Den er satt opp som ftpserver med anonym tilgang. Ingen modifikasjoner er gjort på ftpserveren etter installasjonen.

### 5.5.4 Server 3: Http 192.168.1.3

Server 3 er den andre av de virtuelle serverene. Den finnes på */fake/dummy2*. Den er satt opp som webserver, og kjører Apache. Den eneste endringen som er gjort på denne er å endre ServerType fra standalone til inetd i filen */etc/httpd/conf/httpd.conf.*

## 5.6 Daemons

Linux har en del programmer som kjører i bakgrunnen, og tilbyr tjenester til andre programmer og brukere. Slike programmer kalles ofte for *daemons*. I dette oppsettet må en ta hensyn til 4 slike daemons.

### 5.6.1 Inetd

Inetd er den daemonen som har ansvaret for å starte andre nettverkstjenester. Den lytter på nettverket, og starter opp de andre tjenestene ettersom det blir bruk for dem. Ved å bare ha en daemon som overvåker nettverket, istedenfor en for hver tjeneste, sparer en ressurser. Inetd bruker en konfigurasjonsfil, /etc/*inetd.conf*, for å bestemme hvilket program som skal startes for den enkelte tjenesten. Vedlegg C viser den aktuelle inetd.conf-filen.

### 5.6.2 Virtuald

Etter at tjenesten er blitt identifisert av inetd, skal den kobles mot riktig server. Ved å undersøke forespørselen, kan en finne ut hvilken IP-adresse det spørres etter. Denne oppgaven ordnes av virtuald[15]. Vedlegg D viser kildekoden til en utgave av virtuald som også tar hensyn til tcpd. Hver tjeneste kan ha sin egen konfigurasjonsfil som viser sammenhengen mellom IP-adresse og virtuell server. Disse konfigurasjonsfilene finnes i vedlegg E, F og G. Det er verdt å merke seg at tjenestene kan kjøre på flere av serverene samtidig, ved å sette opp dette i konfigurasjonsfilene. Formatet for disse konfigurasjonsfilene er:

```
IP-address <space> virtual domain
```

Hver av de virtuelle serverene kjøres i et såkalt *chroot-miljø*. Det vil si at rotkatalogen blir omdefinert til å være rotpunktet for de virtuelle serverene. Dette settes opp som virtual domain i konfigurasjonsfilene. På prototypen er det henholdsvis */fake/dummy1/* og */fake/dummy2/*. De programmene som skal kjøres på de virtuelle serverene må da startes med:

```
chroot newdir command
```

*Newdir* er den nye rotkatalogen og *command* er det programmet som skal kjøres. Ved å omdefinere rotkatalogen på denne måten, vil hver av de virtuelle serverene bli sperret inne i sin egen filstruktur, uten mulighet til å påvirke de andre. Denne inndelingen gjør at det er mulig å kjøre separate konfigurasjonsfiler som brukes av programmer som startes opp etter at chroot er utført. Dette gjelder også for filen */etc/passwd* som inneholder oversikten over brukerene på systemet. Som vi skal se i avsnitt 5.6.4 kan denne atskillelsen også skape problemer.

### 5.6.3 Tcpd

Etter at forespørselen har havnet på riktig virtuelle server, blir den overlevert til *tcpd*. Tcpd, eller TCP Wrapper, er en daemon som legger seg rundt tjenesten. Ved hjelp av filene */etc/hosts.allow* og */etc/hosts.deny* kan tcpd utføre tilgangskontroll. /etc/hosts.allow inneholder en liste over maskiner som skal ha tilgang, mens maskinene som er listet i /etc/hosts.deny blir nektet tilgang.

Siden tcpd kjøres etter overgangen til de virtuelle serverene, kan hver server operere med separat tilgangskontroll.

### 5.6.4 Syslogd

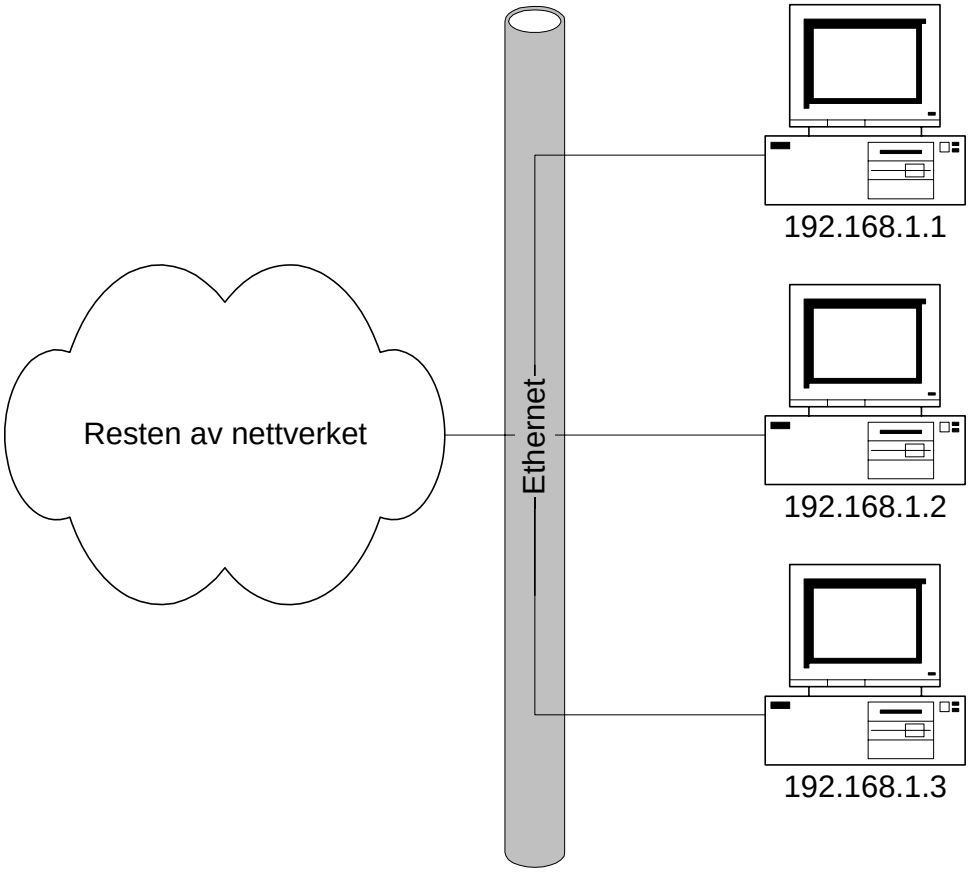
Syslogd er den daemonen som har ansvaret for loggingen av systemet. De ulike programmene som skal logge noe, benytter seg av systemkall som skriver til et FIFO-buffer, som så syslogd leser fra og skriver til loggfilen. Dette FIFO-bufferet, ofte kalt socket, er normalt */dev/log*. Ettersom de virtuelle serverene bruker et chroot-miljø, har ikke disse tilgang til denne socketen. Men siden syslogd kjører på det reelle systemet, har den tilgang til hele filsystemet, inkludert de to virtuelle serverene. Nyere utgaver av syslogd har mulighet for å opprette flere socketer. Ved å opprette en socket i hver av de virtuelle serverene, kan tjenestene her fortsette å logge til systemloggen uten å merke forskjell. Syntaksen for å legge til flere socketer er:

```
daemon syslogd –a newsocket
```

Ved å gjøre denne endringen i filen */etc/rc.d/init.d/syslog* vil de nye socketene bli aktivert hver gang syslogd startes. Vedlegg H viser denne konfigurasjonsfilen.

## 5.7 Virkemåte

Da skulle alle delene være på plass. For en inntrenger vil det se ut som om honeypotsegmentet har 3 ulike servere. Figur 3 viser dette.



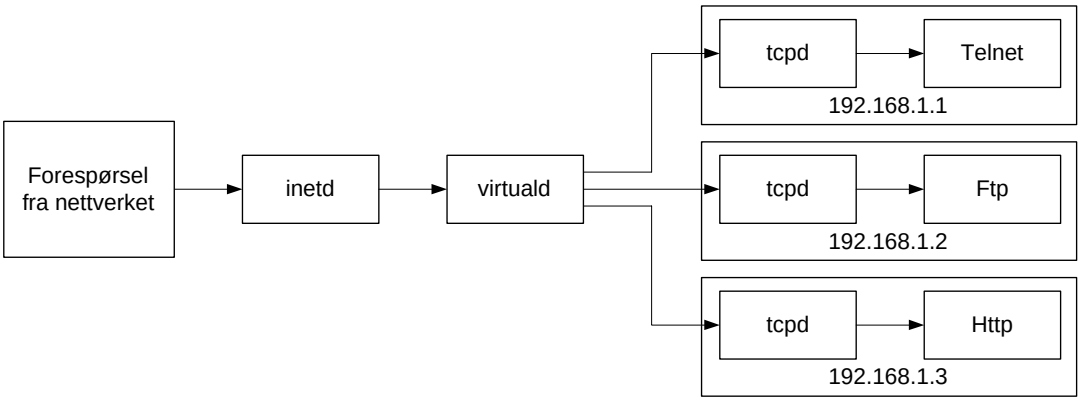

**Figur 3 Hvordan honeypotsegmentet skal se ut fra utsiden.**

Når det kommer en forespørsel vil denne bli fordelt til riktig server gjennom en trinnvis prosess. Denne prosessen er vist i figur 4. Forespørselen kan bli avslått i hvert av trinnene hvis den ikke matcher kriteriet for å slippe videre.

- En forespørsel kommer inn til den reelle maskinen. Denne maskinen kjører inetd. Inetd sjekker med /etc/inetd.conf om tjenesten det spørres etter tilbys. Hvis tjenesten tilbys, blir forespørselen sendt videre til virtuald.

- Virtuald undersøker hvilken IP-adresse det spørres etter, og sjekker med /div/config/conf.* om tilhørende server tilbyr denne tjenesten. Hvis serveren tilbyr tjenesten, blir forespørselen sendt videre til tcpd.

- Tcpd utøver vanlig tilgangskontroll ut fra /etc/hosts.deny og /etc/hosts.allow som i et standard oppsett med en enkelt server. Hvis forespørselen ikke blir sperret av tcpd, blir den overlevert til det programmet som styrer tjenesten.




**Figur 4 Trinnvis prosess fra forespørsel til respons. Forespørselen må gjennom 3 ulike programmer for den kommer fram til serverprogrammet.**

## 5.8 Resultater

### 5.8.1 Portskanninger

Portskanninger mot de tre serverene viser at alle tilbyr ftp, telnet og http tjenester. Vedlegg I viser resultatet av en skanning foretatt med IPTools[16]. IPTools er en samling med nettverksverktøy som blant annet inneholder en portskanner. Det er valgt ut en del porter som det testes mot. Denne listen er på ingen måte komplett, men skanning av alle 65535 portene er ikke hensiktsmessig og vil ikke gi noe utslag for resultatet.

Resultatet er ikke helt som forventet. Alle 3 serverene rapporterer de samme tjenestene. Dette skyldes trolig det faktum at den typen portskanning som benyttes, ikke venter på en fullstendig oppkobling før den rapporterer om tjenesten finnes. Forespørselen kommer kun til inetd, som ser at tjenesten finnes på systemet. Dermed så opprettes det en socket som sendes videre inn i systemet. Portskanneren får vite at det er blitt opprettet en socket, kobler ned igjen forbindelsen og rapporterer at honeypoten svarer på den aktuelle porten.

### 5.8.2 Telnet

Forsøk på oppkobling via telnet viser at telnet kun kjører på server 1. På de andre serverene blir oppkoblingen stoppet i virtuald, som ikke finner IP-

adressen i sine konfigurasjonsfiler. Vedlegg J viser utdrag av filen */var/log/messages* for denne testsesjonen.

### 5.8.3 Ftp

Forsøk på oppkobling via ftp viser at ftp kun kjører på server 2. På de andre serverene blir oppkoblingen stoppet i virtuald, som ikke finner IP-adressene i sine konfigurasjonsfiler. Vedlegg K viser utdrag av filen */var/log/messages* for denne testsesjonen.

### 5.8.4 Http

Forsøk på oppkobling via http viser at http kun kjører på server 3. På de andre serverene blir oppkoblingen stoppet i virtuald, som ikke finner IP-adressene i sine konfigurasjonsfiler. Vedlegg L viser utdrag av filen */var/log/messages* for denne testsesjonen.

## 5.9 Vurdering av løsning

### 5.9.1 Testresultater

Portskanningen viser en klar svakhet ved prototypen. Alle 3 serverene rapporterer de samme 3 tjenestene, og på en portskanning vil serverene se identiske ut. Det er ikke usannsynlig at 3 servere på samme segment kjører de samme tjenestene, men når det kun er snakk om 3 tjenester blir det veldig påfallende. Dersom en også hadde kjørt andre tjenester, ville et slikt sammentreff blitt noe mer kamuflert. For å komme rundt dette problemet, er en mulig løsning å modifisere inetd til både å sjekke hvilken tjeneste som det spørres etter, og samtidig om tjenesten kjører på den aktuelle serveren.

Når det gjelder forsøkene på oppkobling mot de ulike tjenestene, så viser disse at prototypen virker etter sin hensikt.

### 5.9.2 Ressursbruk

Ved å la alle tjenestene starte gjennom inetd og virtuald, har en god kontroll på hvilke tjenester som starter. Men denne trinnvise oppstarten krever en del overhead. For en honeypot har ikke dette så mye å si, fordi operasjonen av denne ikke er tidskritisk, og at den normalt ikke har så mye trafikk som en produksjonsserver

# 6 Konklusjon

Som et resultat av den kraftige økningen av nettverksbaserte tjenester de siste årene, blir kravet om datasikkerhet bare viktigere og viktigere. Et av de viktigste verktøyene for å opprettholde et akseptabelt sikkerhetsnivå, er kunnskap. Både kunnskap om eget nettverk, og om eventuelle inntrengeres verktøy og metoder. I denne sammenhengen kan såkalte honeypots hjelpe med å øke kunnskapsnivået hos den nettverksansvarlige.

Jeg har i denne rapporten sett på ulike aspekter ved honeypots, og vist hvordan en kan sette opp en honeypot som kan simulere et helt nettverkssegment, og tilby flere servere med ulike tjenester. I sin nåværende form har den noen svakheter, men testresultatene viser at maskinen tilbyr 3 ulike tjenester på 3 ulike IP-adresser.

# Referanser


[1] Barkakati, Naba. *Redhat Linux Secret, Third Edition*, IDG Books Worldwide, Foster City, 1999

[2] Elboth, David. *Boken om Linux, 2 utg.*, IDG Norge Books, Oslo, 1999

[3] Boran, Sean. *The IT Security Cookbook*, 1999
URL: http://secinf.net/info/misc/boran/

[4] System Sikkerhet AS, *Årsrapport for sikkerhetsovervåking*
URL: http://www.system.sikkerhet.no/

[5] Happy Hacker. *What is a hacker*
URL: http://www.happyhacker.org/define.shtml

[6] 2600 Magazine. *Hacked Sites*
URL: http://www.2600.com/hacked_pages/

[7] Graham, Robert. *FAQ: Network Intrusion Detection Systems*, 1999
URL: http://www.ticm.com/kb/faq/idsfaq.html

[8] Fyodor. *Exploit World*
URL: http://www.insecure.org/sploits.html

[9] Tripwire, Inc.
URL: http://www.tripwire.com

[10] Cooper, Russ. *NTBugTraq*
URL: http://www.ntbugtraq.com/

[11] Network Associates. *CyberCop Sting*
URL: http://www.pgp.com/asp_set/products/tns/ccsting_intro.asp

[12] Fred Cohen & Associates. *Deception Toolkit*
URL: http://all.net/dtk

[13] Network Flight Recorder. *Back Officer Friendly*
URL: http://www.nfr.net/products/bof/

[14] Rekhter, Y., Moskowitz, B., Karrenberg, D., deGroot, G. J., Lear, E. *RFC 1918 Address Allocation for Private Internets*, 1996
URL: http://www.ietf.org/rfc/rfc1918.txt?number=1918

[15] Ackerman, Brian. *Virtual Services Howto*, 1998
URL: http://howto.tucows.com/LDP/HOWTO/Virtual-Services-HOWTO.html

[16] Kozloc, Alexander. *IP-Tools*, 1999
URL: http://ks-soft.mastak.com/users/kssoft/ip-tools.eng/index.htm

# Vedlegg A /etc/rc.d/rc.local

```
#!/bin/sh
# This script will be executed *after* all the other init scripts.
# You can put your own initialization stuff in here if you don't
# want to do the full Sys V style init stuff.
if [ -f /etc/redhat-release ]; then
   R=$(cat /etc/redhat-release)
   arch=$(uname -m)
   a="a"
   case "_$arch" in
         _a*) a="an";;
         _i*) a="an";;
   esac
   # This will overwrite /etc/issue at every boot.  So, make any changes you
   # want to make to /etc/issue here or you will lose them when you reboot.
   echo "" > /etc/issue
   echo "$R" >> /etc/issue
   echo "Kernel $(uname -r) on $a $(uname -m)" >> /etc/issue
   cp -f /etc/issue /etc/issue.net
   echo >> /etc/issue
fi
# Setting up IP aliases
echo "Setting up IP aliases"
echo "Main IP already set up"
/sbin/ifconfig eth0:0 192.168.1.2
/sbin/ifconfig eth0:1 192.168.1.3
#Setting up routes
/sbin/route add -host 192.168.1.2 dev eth0:0
/sbin/route add -host 192.168.1.3 dev eth0:1
```

# Vedlegg B /div/bin/newdomain

```
#!/bin/sh
#constants
basedir=/fake
#echo "New domain: "
#read newdomain
if [ $# != 1 ]
then
        echo Use: newdomain dirname
        echo where dirname = name of new directory
        exit 0
fi
newdir=$basedir/$1
#Test if new directory already exists
if [ -d $newdir ]
then
        echo Directory already exists
        exit 0
fi
#Make directory
echo Make new directory: $newdir
echo Press any key
read ans
mkdir $newdir
#Make directory structure
echo Make directory structure
# Notes:
# p: preserves owner, perm. and timestamp
# R: copies recursively
# d: copies link
#Make /bin
echo Make bin
cp -pR /bin $newdir
```

```
#Make /root
echo Make root
cp -pR /root $newdir
#Make /sbin
echo Make sbin
cp -pR /sbin $newdir
#Make /dev
echo Make dev
cp -dR /dev $newdir
#Make /usr/*
echo Make usr
mkdir $newdir/usr
echo "     bin"
cp -pR /usr/bin $newdir/usr
echo "     sbin"
cp -pR /usr/sbin $newdir/usr
echo "     lib"
cp -pR /usr/lib $newdir/usr
#Make /etc
echo Make etc
cp -pR /etc $newdir
#Make /lib
echo Make lib
cp -pR /lib $newdir
#Make /tmp
echo Make tmp
mkdir -m 0777 $newdir/tmp
#Make /var
echo Make var
cp -pR /var $newdir
#Make /proc
#echo Make proc
#cp -pR /proc $newdir
#Make /lost+found
```

```
echo make lost+found
mkdir $newdir/lost+found
#Make home
echo Make home
mkdir $newdir/home
echo "     ftp"
cp -Rp /home/ftp $newdir/home
#Finished
echo Thats it
```

# Vedlegg C /etc/inetd.conf

```
# Virtual inetd.conf
# Format:
# <service_name> <sock_type> <proto> <flags> <user> <server_path> <args>
#
# These are standard services.
#
ftp     stream tcp      nowait root     /div/bin/virtuald \
        /div/config/conf.ftp tcpd in.ftpd -l –a
telnet stream  tcp      nowait root     /div/bin/virtuald \
        /div/config/conf.telnet tcpd in.telnetd
www stream tcp      nowait root     /div/bin/virtuald \
        /div/config/conf.www            tcpd     httpd -f \
        /etc/httpd/conf/httpd.conf
```

# Vedlegg D /div/source/virtuald.c

```
#include <netinet/in.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <stdarg.h>
#include <unistd.h>
#include <string.h>
#include <syslog.h>
#include <stdio.h>
#define VERBOSELOG
#define BUFSIZE 8192

int getipaddr(char **ipaddr)
{
        struct sockaddr_in virtual_addr;
        static char ipaddrbuf[BUFSIZE];
        int virtual_len;
        char *ipptr;
        virtual_len=sizeof(virtual_addr);
        if ( getsockname(0,(struct sockaddr *)&virtual_addr,&virtual_len)>0)
        {
                syslog(LOG_ERR,"getipaddr: getsockname failed: %m");
                return -1;
        }
        if (!(ipptr=inet_ntoa(virtual_addr.sin_addr)))
        {
                syslog(LOG_ERR,"getipaddr: inet_ntoa failed: %m");
                return -1;
        }
        strncpy(ipaddrbuf,ipptr,sizeof(ipaddrbuf)-1);
        *ipaddr=ipaddrbuf;
        return 0;
}
```

```
int iptodir(char **dir, char *ipaddr, char *filename)
{
	char buffer[BUFSIZE], *bufptr;
	static char dirbuf[BUFSIZE];
	FILE *fp;
	if(!(fp=fopen(filename,"r")))
	{
		syslog(LOG_ERR,"iptodir: fopen failed: %m");
		return -1;
	}
	*dir=NULL;
	while(fgets(buffer,BUFSIZE,fp))
	{
		buffer[strlen(buffer)-1]=0;
		if(*buffer=='#' || *buffer==0)
			continue;
		if(!(bufptr=strchr(buffer,' ')))

		{
			syslog(LOG_ERR,"iptodir: strchr failed");
			return -1;
		}
		*bufptr++=0;
		if(!strcmp(buffer,ipaddr))
		{
			strncpy(dirbuf,bufptr,sizeof(dirbuf)-1);
			*dir=dirbuf;
			break;
		}
		if(!strcmp(buffer,"default"))
		{
			strncpy(dirbuf,bufptr,sizeof(dirbuf)-1);
			*dir=dirbuf;
			break;
```

```
                }
        }
        if(fclose(fp)==EOF)
        {
                syslog(LOG_ERR,"iptodir: fclose failed: %m");
                return -1;
        }
        if(!*dir)
        {
                syslog(LOG_ERR,"iptodir: ip not found in conf file");
                return -1;
        }

        return 0;
}

int main(int argc,char **argv)
{
        char *ipaddr, *dir;
        openlog("virtuald",LOG_PID,LOG_DAEMON);
#ifdef VERBOSELOG
        syslog(LOG_ERR,"Virtuald started");
#endif
        if(!argv[1])
        {
                syslog(LOG_ERR,"invalid arguments: no conf file");
                exit(0);
        }
        if(!argv[2])
        {
                syslog(LOG_ERR,"invalid arguments: no program to run");
                exit(0);
        }
        if(!argv[3])
```

```
        {
                syslog(LOG_ERR,"invalid arguments: no program to run");
                exit(0);
        }
        if(getipaddr(&ipaddr))
        {
                syslog(LOG_ERR,"getipaddr failed");
                exit(0);
        }
#ifdef VERBOSELOG
        syslog(LOG_ERR,"Incoming ip: %s",ipaddr);
#endif
        if(iptodir(&dir,ipaddr,argv[1]))
        {
                syslog(LOG_ERR,"iptodir failed");
                exit(0);
        }
        if(chroot(dir)<0)
        {
                syslog(LOG_ERR,"chroot failed: %m");
                exit(0);
        }
#ifdef VERBOSELOG
        syslog(LOG_ERR,"Chroot dir: %s",dir);
#endif
        if(chdir("/")<0)
        {
                syslog(LOG_ERR,"chdir failed: %m");
                exit(0);
        }
        if(execvp(argv[2],argv+3)<0)
        {
                syslog(LOG_ERR,"execvp failed: %m");
                exit(0);
```

```
        }
        closelog();
        exit(0);
}
```

# Vedlegg E /div/config/conf.telnet

```
# Config file for telnet services
#
# Format: IP <space> dir
192.168.1.1 /
#192.168.1.2 /fake/dummy1
#192.168.1.3 /fake/dummy2
#default /
```

# Vedlegg F /div/config/conf.ftp

```
# Config file for ftp services
#
# Format: IP <space> dir
#192.168.1.1 /
192.168.1.2 /fake/dummy1
#192.168.1.3 /fake/dummy2
#default /
```

# Vedlegg G /div/config/conf.www

```
# Config file for http-services
#
# Format: IP <space> dir
#192.168.1.1 /
#192.168.1.2 /fake/dummy1
192.168.1.3 /fake/dummy2
#default /
```

# Vedlegg H /etc/rc.d/init.d/syslog

```
#!/bin/sh
#
# syslog        Starts syslogd/klogd.
#
#
# chkconfig: 2345 30 99
# description: Syslog is the facility by which many daemons use to log \
# messages to various system log files.  It is a good idea to always \
# run syslog.
# Source function library.
. /etc/rc.d/init.d/functions
[ -f /usr/sbin/syslogd ] || exit 0
[ -f /usr/sbin/klogd ] || exit 0
# See how we were called.
case "$1" in
  start)
   echo -n "Starting system logger: "
   # we don't want the MARK ticks
   #
   # added the extra sockets to make it available
   # from the chroot environment
   #
   daemon syslogd -m 0 -a /fake/dummy1/dev/log -a /fake/dummy2/dev/log
   echo
   echo -n "Starting kernel logger: "
   daemon klogd
   echo
   touch /var/lock/subsys/syslog
   ;;
  stop)
   echo -n "Shutting down kernel logger: "
   killproc klogd
```

```
  echo
  echo -n "Shutting down system logger: "
  killproc syslogd
  echo
  rm -f /var/lock/subsys/syslog
  ;;
 status)
  status syslogd
  status klogd
  ;;
 restart)
  $0 stop
  $0 start
  ;;
 *)
  echo "Usage: syslog {start|stop|status|restart}"
  exit 1
esac
exit 0
```

# Vedlegg I Resultat av portskanning

Address : 192.168.1.1

Ping .... Ok, Time : 1

Port  21 ... Ok !

Port  23 ... Ok !

Port  80 ... Ok !

Port   9 ... Connection refused

Port   1 ... Connection refused

Port  13 ... Connection refused

Port   7 ... Connection refused

Port  11 ... Connection refused

Port  20 ... Connection refused

Port 220 ... Connection refused

Port  25 ... Connection refused

Port  37 ... Connection refused

Port  69 ... Connection refused

Port 109 ... Connection refused

Port 113 ... Connection refused

Port 161 ... Connection refused

Port  22 ... Connection refused

Port  70 ... Connection refused

Port  82 ... Connection refused

Port 110 ... Connection refused

Port 194 ... Connection refused

Port  47 ... Connection refused

Port  67 ... Connection refused

Port  79 ... Connection refused

Port 107 ... Connection refused

Port 111 ... Connection refused

Port 115 ... Connection refused

Port 119 ... Connection refused

Port 143 ... Connection refused

Port  24 ... Connection refused

Port  68 ... Connection refused

Port  92 ... Connection refused

Address : 192.168.1.2

Ping .... Ok, Time : 1

Port  21 ... Ok !

Port  23 ... Ok !

Port  80 ... Ok !

Port 194 ... Connection refused

Port   7 ... Connection refused

Port  11 ... Connection refused

Port  47 ... Connection refused

Port  67 ... Connection refused

Port  79 ... Connection refused

Port 115 ... Connection refused

Port 119 ... Connection refused

Port 107 ... Connection refused

Port 111 ... Connection refused

Port 143 ... Connection refused

Port  20 ... Connection refused

Port  24 ... Connection refused

Port  68 ... Connection refused

Port  92 ... Connection refused

Port 220 ... Connection refused

Port   1 ... Connection refused

Port   9 ... Connection refused

Port  13 ... Connection refused

Port  37 ... Connection refused

Port  25 ... Connection refused

Port  69 ... Connection refused

Port 109 ... Connection refused

Port 113 ... Connection refused

Port 161 ... Connection refused

Port  22 ... Connection refused

Port  70 ... Connection refused
Port  82 ... Connection refused
Port 110 ... Connection refused

Address : 192.168.1.3
Ping .... Ok, Time : 1
Port  21 ... Ok !
Port  23 ... Ok !
Port  80 ... Ok !
Port   1 ... Connection refused
Port  13 ... Connection refused
Port   9 ... Connection refused
Port  37 ... Connection refused
Port  25 ... Connection refused
Port  69 ... Connection refused
Port  22 ... Connection refused
Port  70 ... Connection refused
Port  82 ... Connection refused
Port  11 ... Connection refused
Port   7 ... Connection refused
Port  47 ... Connection refused
Port  67 ... Connection refused
Port  79 ... Connection refused
Port  20 ... Connection refused
Port  24 ... Connection refused
Port  68 ... Connection refused
Port  92 ... Connection refused
Port 220 ... Connection refused
Port 113 ... Connection refused
Port 109 ... Connection refused
Port 161 ... Connection refused
Port 110 ... Connection refused
Port 194 ... Connection refused
Port 111 ... Connection refused

Port 115 ... Connection refused

Port 107 ... Connection refused

Port 119 ... Connection refused

Port 143 ... Connection refused

Done

# Vedlegg J Utdrag av /var/log/messages for telnet

May 28 15:12:57 test virtuald[441]: Virtuald started

May 28 15:12:57 test virtuald[441]: Incoming ip: 192.168.1.1

May 28 15:12:57 test virtuald[441]: Chroot dir: /

May 28 15:13:09 test PAM_pwdb[442]: (login) session opened for user trond by (uid=0)

May 28 15:13:40 test PAM_pwdb[442]: (login) session closed for user trond

May 28 15:13:51 test virtuald[455]: Virtuald started

May 28 15:13:51 test virtuald[455]: Incoming ip: 192.168.1.2

May 28 15:13:51 test virtuald[455]: iptodir: ip not found in conf file

May 28 15:13:51 test virtuald[455]: iptodir failed

May 28 15:14:00 test virtuald[456]: Virtuald started

May 28 15:14:00 test virtuald[456]: Incoming ip: 192.168.1.3

May 28 15:14:00 test virtuald[456]: iptodir: ip not found in conf file

May 28 15:14:00 test virtuald[456]: iptodir failed

# Vedlegg K Utdrag av /var/log/messages for ftp

May 28 15:32:56 test virtuald[464]: iptodir failed

May 28 15:42:42 test virtuald[486]: Virtuald started

May 28 15:42:42 test virtuald[486]: Incoming ip: 192.168.1.1

May 28 15:42:42 test virtuald[486]: iptodir: ip not found in conf file

May 28 15:42:42 test virtuald[486]: iptodir failed

May 28 15:42:51 test virtuald[487]: Virtuald started

May 28 15:42:51 test virtuald[487]: Incoming ip: 192.168.1.2

May 28 15:42:51 test virtuald[487]: Chroot dir: /fake/dummy1

May 28 13:43:05 test ftpd[487]: ANONYMOUS FTP LOGIN FROM 192.168.1.10 [192.168.1.10],

May 28 13:43:11 test ftpd[487]: FTP session closed

May 28 15:43:22 test virtuald[488]: Virtuald started

May 28 15:43:22 test virtuald[488]: Incoming ip: 192.168.1.3

May 28 15:43:22 test virtuald[488]: iptodir: ip not found in conf file

May 28 15:43:22 test virtuald[488]: iptodir failed

# Vedlegg L Utdrag av /var/log/messages for http

May 28 16:26:46 test virtuald[417]: Chroot dir: /

May 28 16:34:15 test virtuald[425]: Virtuald started

May 28 16:34:15 test virtuald[425]: Incoming ip: 192.168.1.1

May 28 16:34:15 test virtuald[425]: iptodir: ip not found in conf file

May 28 16:34:15 test virtuald[425]: iptodir failed

May 28 16:34:24 test virtuald[426]: Virtuald started

May 28 16:34:24 test virtuald[426]: Incoming ip: 192.168.1.2

May 28 16:34:24 test virtuald[426]: iptodir: ip not found in conf file

May 28 16:34:24 test virtuald[426]: iptodir failed

May 28 16:34:33 test virtuald[427]: Virtuald started

May 28 16:34:33 test virtuald[427]: Incoming ip: 192.168.1.3

May 28 16:34:33 test virtuald[427]: Chroot dir: /fake/dummy2