HØGSKOLEN I SØR-TRØNDELAG

Avdeling for informatikk og e-læring

Bacheloroppgave 2013
Studium: *Dataingeniør*

Tittel – norsk: Dynamisk nettversbrannmur med rettferdig deling av linjekapasitet *Tittel - engelsk:Dynamic Network Firewall with fair distibution of bandwidth*		*Oppgave nr.:20 E*
Oppgavestiller: Svein Ove Undal		
	Kontaktperson: Svein Ove Undal	
	Telefon:481 91 703	*E-postadresse:sveinou@tihlde.org*
	Postadresse:	
Studenter:Espen Gjærde og Svein Ove Undal		
Veileder ved HiST: Helge Hafting		
Sammendrag:Gateway som tvinger pålogging via nettside, og med autmatisk justerande og ressursbalanserande brannmur. *Abstract in English: Gateway that forces authentication, and that automaicly adjusts the firewall and balances broadband capacity.*		
I henhold til kontrakt inngått mellom HiST/AITeL, oppgavestiller og studenter gjelder følgende for publisering av resultater: Normalsituasjonen er at rettigheter til å framstille kopier og å videreutvikle produktet og/eller metoder tilfaller HiST. Alle resultater er åpent tilgjengelig. Eventuelle avvik fra normalsituasjonen er markert med kryss nedenfor:		
☐	*Avvik fra normalsituasjonen: Oppdragsgiveren kan utnytte produktet kommersielt og videreutvikle produktet/metoden. Høgskolen vil ikke utnytte produktet kommersielt, men vil kunne arbeide videre med den grunnlagskompetansen som er vunnet gjennom prosjektet.*	
☐	*Avvik fra normalsituasjonen: Studenten(e) kan utnytte produktet kommersielt og videreutvikle produktet/metoden.*	
☐	*Avvik fra normalsituasjonen: Resultatene fra arbeidet er sperret og kun tilgjengelig etter avtale med oppdragsgiver.*	

Sluttrapport
Dynamisk Nettverksbrannmur

Espen Gjærde Svein Ove Undal

2013

Forord

Denne rapporten er ein del av bacheloroppgåva til forfattarane, og markerer avsluttinga av tre år på Dataingeniørlinja hjå Avdeling for Informatikk og e-Læring ved Høgskolen i Sør-Trøndelag.

Dette prosjektet er ei vidareføring av Svein Ove Undal sitt prosjekt «prosjektnamn» i faget «RMI med prosjekt i distibuerte systemer». Svein Ove Undal ønska då å vidareføre denne oppgåva som bachelorprosjekt, og det var no Espen Gjærde kom inn i bildet. Begge forfattarane går på studieretninga «Nettverksakitektur og -design», og er svært interessert i både nettverksteknologi og Linux. Det var derfor eit enkelt val når vi kunne bruke siste semester på å grave oss ned i desse fagfelta.

Sidan vi bestemte oss for å nytte Python som programmeringspråk i løysinga av oppgåva, og sidan ingen av forfattarane hadde noko erfaring med dette frå før, måtte vi bruke meir tid enn venta på å eiga opplæring. Sjølv om det i starten kunne sjå litt omfattande ut, henta vi inn mykje tid mot slutten når vi såg kor enkelt det var med eit programmeringspråk over heile fjøla, i staden for å blande bash og php.

Før vi tek til på sluttrapporten vil vi nytte høvet til å takke rettleiar for bacheloroppgåva, Helge Hafting, for gode råd og innspel undervegs. Vi vi også rette ein stor takk til TIHLDE-LAN og spesielt deltakarena for tolmod og gode tilbakemeldingar under testinga av systemet.

Trondheim, 15. mai 2013

Espen Gjærde

Svein Ove Undal

Del I

Oppgåva

Oppgåveteksten

0.1 Orginal oppgåvetekst

«Oppgava går ut på å lage ein streng gateway/brannmur der kun autentiserte brukarar kan logge seg på. Brannmuren skal dekke kjente problem med likandes system som mac-adresse spoofing og dns-tunnelering.»

Denne oppgåveteksten vart revidert for å utvide oppgåva til to personar. Aktuelle utvidingar var då automatisk justering av bandbredde til kvar enkelt brukar, administrasjonspanel og støtte for IPv6.

0.2 Revidert oppgåvetekst

Ein ny og revidert oppgåvetekst vart laga, og går som følgjer:

«Lage ein gateway med brannmur som tvingar brukarar til å logge seg på før dei har tilgang til internett. Gatewayen skal automatisk justere bandbredde til den enkelte brukaren. Administratorar skal ha tilgang til eit oversiktleg administrasjonsgrensesnitt.»

Samandrag

Treige nettverk er noko nesten alle har vore bort i, og irritasjonen over ei treig nettilkopling er noko mange kjenner til. Dette rammar oss ofte når vi nyttar oss av ymse offentlege – gjerne trådlause – nettverk. For oss studentar som gjerne bur i kollektiv og dermed deler nettlinja med fire-fem andre brukarar har også ofte kjent på korleis det er når nokon «stjel» nettlinja. Det er dette problemet vi vil til livs med denne bacheloroppgåva.

Ved å tvinge brukarane til å først logge seg inn, kan vi følje med på kor mykje av linja kvar enkelt legg beslag på, og ta affære om nokon øydelegg for andre. Slik kan vi sikre at alle får ein betre nettopplevelse og nettlinja vert rettferdig delt mellom brukarane. Det som skil dette systemet frå andre, er at vi ikkje sett avgrensar nettlinja til nokon av brukarane før bruken vert eit problem. Er du aleine på nettverket skal du sjølvsagt få heile linja – og brukar dei andre brukarane berre ein liten del av linja, er det ikkje noko problem at du tek resten av linja. Avgreninga skjer først når bruken er større enn kapasiteten. Da må vi sjå til at alle får lik utnytting av linja.

Det er også fleire ressursar enn berre bandbredde i ei nettlinje. Antall oppkoplingar kan fort bli eit problem for nokon rutarar. Derfor har vi tre «dimensjonar» vi ser på når vi avgrensar bruken. Dei to første bandbreidda opp og ned, men vi ser også på antall oppkoplingar. Ut i frå erfaringar vi fekk på «TIHLDE-LAN» gir antall oppkoplingar også ein god indikasjon på kven som nyttar mest av linja. Om nokon til dømes brukar torrent-teknologi, vil dei gjerne utmerke seg i statistikken med svært mange oppkoplingar.

Utviklingen av systemet er gjort etter rammeverk og utviklingsprosessar som vi er kjent med frå faget «Systemering med prosjekt», og vi har valt utviklingsprosessen UP. Vi fekk også testa delar av systemet under THILDE-LAN. Dette gav oss mange gode erfaringar, og vi fekk samla mykje trafikkdata som gjorde det enklare for oss å kjenne kvar skoen trykkjer. Vi fekk også

prøve oss på XP-utvikling, da det sjølvsagt oppstod nokre bugs. Da var det berre å kaste seg rundt å komme med ei løysing.

Innhald

Del II

Sluttrapporten

Kapittel 1

Introduksjon

1.1 Introduksjon

1.2 Definisjonar og forkortingar

1.2.1 Ordliste

Apache eller Apache HTTP Server er ein populær http-tjenar. Denne sørgjer for at nettsidene til systemet verte vist fram.

Django er eit rammeverk som gjer det enkelt å bygge dynamiske nettsider med python.

Python eit populært høgnivå kodespråk. Det er dette det meste av kodinga for prosjektet er gjort

Bash(-script) Bourne-Again-SHell. Er ein komandotolkar for unix-system. Det kan også setjast opp ei fil med komandoar som skal utførast av bash. Dette er eit bash-script.

NAT står for «Network Address Translation» og er ei fellesbenevning på oversetting av ip-addresser mellom ulike nettverk.

Port-Forward er ei form for NAT der vi oversett portnummer. Med Port-Forward kan vi rute om trafikk frå ein spesiell TCP- eller UDP-port til ein annan.

IPv4 er ei forkorting for «Internet Protocol version 4» og er i dag den IP-versjonen som er mest utbredt.

IPv6 er ein ny versjon av IP, med mange fleire tilgjenelege addresser enn IPv4.

Distro er ei forkorting/slang-form av distribusjon og blir brukt om ei avgreining eller ein type linux.

WebUI Web User Interface – eit nettbasert brukargrensesnitt.

DHCP *Dynamic Host Configuration Protocol* er ein protokoll for utdeling av nettverkskonfigurasjon i eit nettverk.

Linuxkjerna

CLI *Command Line Interface* eller «ledetekst» på norsk er eig grensesnitt for å sende kommandoar direkte til eit program. Det vert også ofte brukt om brukergrensesnitt som er basert i ein kommandoterminal.

Kapittel 2

Teori

I denne delen av sluttrapporten vil vi gå gjennom teknologien vi har tilgjeneleg for å løyse oppgåva, og gje ei oversikt over dei konsepta vi kan bruke.

2.1 Konsept

2.1.1 Captive-Portal

Captive-Portal er eit samleomgrep for nettverksløysingar der brukarar vert tvinga til å gå gjennom ei påloggingsprosedyre i nettlesaren. Det er ikkje nokon fast definisjon på ein Captive Portal. I følgje Cisco skal ein captive-portal videresende ein nettlesar til ein bestem plass, ta vare på orginal HTTP-informasjon og *kan* videresende brukaren basert på denne informasjonen. [1]

2.1.2 Behandling av trafikk

2.1.3 Tidsavbrot for sesjon

Når nokon har logga seg på eit system, er det ikkje sagt at vedkommande hugsar å logge seg ut. I eit høve der vi ikkje vil at uvedkommande skal kunne bruke eit system, eller der vi vil vere sikker på at vi har knytt rett brukarnamn til kvar sesjon, er det eit poeng at brukarar som ikkje er aktive vert logga ut.

2.2 Aktuell teknologi

Når vi no ser på dei forskjellige teknologiane vi kan nytte, er det naturleg å dele desse inn i operativsystem, kodespråk og programvare. Vi vel også å

skilje ut viktige modular som er inkludert i Linuxkjerna og forklare desse nærmare.

2.2.1 Operativsystem

Debian 6.0.6 «Squeeze»

Vi har valt å nytte oss av linux-systemet Debian. Vi nyttar versjon 6.0.6, som er den versjonen som var den siste stabile utgivinga av Debian då vi starta med bacheloroppgåva. Denne har kodenavnet «Squeeze». Det har i ettertid (3.mai) kome ut ein ny stable-versjon av Debian, «Debian 7 Wheezy». [2]

2.2.2 Programmeringsspråk

Python

Kodespråket Python er eit høgnivå kodespråk som kjem ferdiginstallert i dei fleste linux-distibusjonar. I Linuxversjonen vi nyttar er Python versjon 2.6.6 integrert. [3]. Python har gode moglegheiter for å samarbeide med linux-systemet, og er eit kodespråk som «plays well with others», altså er enkelt å setje opp mot andre systemer, slik som database og nettjenarar. [4, 5] I tillegg til dette er det også enkelt å gjere systemkall i linux frå Python. [6]

Django

Django er eit rammeverk for utvikling av nettsider i python. Å nytte dette rammeverket gjer at vi enkelt kan lage gode dynamiske nettsider. Django har gode modular for oppsett av nettsider, noko som gjer at vi slepp å lage mange nokså like modular. I staden kan vi bruke velkjente og gjennomtesta metodar og heller tilpasse desse til våra bruk. Dette sparar oss for mykje koding, blant anna av sesjonshandtering.[7, 8]

I dette prosjektet nyttar vi Django versjon 1.2.3, da det er denne som følgjer pakkebrønnen[1] til Debian Squeeze. [9]

Bash-scripting

Bourne-Again SHell er eit kommandoskall, og er det komandoskallet som blir nytta som standard i debian linux. [10, 11] Bash-scripting er ei liste med kommandoar som skal utførast av bash. Bash støttar også scripting språka

[1] Pakkebrønn: ein katalog av testa og godkjente programvare for eit system.
For Debian: `http://packages.debian.org`

til mange andre kjente og nytta unix-kommandoskall, deriblandt csh og ksh. [12].

PHP

Hypertext Preprocessor er eit kodespråk for å lage dynamiske nettsider. Dette er eit poplært kodespråk som er godt utbredt i open kjeldekode-miljø. PHP er laga for at utviklarar hurtig skal kunne lage dynamiske nettsider. [13] Versjonen som er aktuell for Debian Squeeze er 5.3.3 [14]

2.2.3 Programvare

Apache HTTP Server

Apache HTTP Server er den mest brukte http-tenaren på internett. [15]. Dette er ein robust vevstjenar som er basert på open kjeldekode. Dette har gjort at der er mange modular tilgjeneleg. Blandt desse har vi mod_wsgi for Python og php5_module for PHP. [16, 17]

MySQL Community Edition

MySQL er verdas mest populære opne relasjonsdatabase, og er spesielt vanlig å nytte til webapplikasjonar. Relasjonsdatabasen gir oss muligheten til å hurtig og effektivt lagre og hente data. [18]

SQLite

SQLite er ein filbasert relasjonsdatabase som er vanleg å nytte i frittståande applikasjonar. Den er vanleg å bruke til enkle nettsider og små applikasjonar. SQLite samlar all databaseinformasjonen i ei fil, og gjer det soleis enkelt når det kun er ein applikasjon som skal nytte databasen[19]. Bakdelane med denne programvara er at der er ein del begrensingar i SQL-funksjonalitet, heriblant JOIN og ALTER. [20]

Internet Systems Consortium DHCP Server

Internet Systems Consortium (ISC) DHCP Server er ein implementasjon av DHCP-systemet og står for utlevering av nettverksinformasjon til datamaskiner i eit nettverk. Noko av det mest vanlege er IP-addresser, DNS-informasjon og kva som er addressa til nærmaste router. Denne serveren har også moglegheit for å kjøre kommandoar når spesielle hendinga skjer [21, 22].

2.2.4 Integrerte linuxmodular

Netfilter og IPtables

Netfilter er et rammeverk for filtrering av nettverkstrafikk, og har vore ein del av linux-kjerna sidan linuxversjon 2.4 [23]. Gjennom netfilter og kommandoen «iptabels» gjer dette linux i stand til å mellom anna utføre NAT, pakkefiltrering for både IPv4 og IPv6, føre statistikk over oppkoplingar og utføre port forwarding.

Linux Advanced Routing and Traffic Control

Linux Advanded Routing and Traffic Control (lartc) er eit sett med verktøy for å manipulere pakkar som går gjennom linux, og er ein del av kjerna i linux si pakkehandtering. [24] Denne modulen gir mellom anna moglegheit for å merke pakkar slik at dei kan behandlast i andre verktøy seinare.

2.2.5 Protokollar

DHCP

DHCP er ein protokoll for utsending av nettverkskonfigurasjon til klientar i eit nettverk. Denne protokollen fungerar ved at ein klient sender ut ein førespurnad om DHCP-informasjon og ein server svarer på desse. Klienten vil deretter automatisk få IP-addresse og vanligvis også få tilsendt innstillingar for standardruter i nettverket og informasjon om navnetjenarar. [25] For å unngå at det «hopar seg opp» med inaktive klientar, har desse automatiske innstillingane eit tidsavbrot. Det er klienten sitt ansvar å seie frå om at den framleis er aktiv. Dersom dette ikkje vert gjort, kan tenaren sende ut desse innstillingane til nye klientar. [26]

2.3 Utviklingsprosess

2.3.1 Arbeidsmetodikk

Her vil vi raskt gå gjennom nokon av dei arbeidsmetodikkan som var aktuelle å bruke i prosjektet. Alle desse er smidige og iterative prosessar.

Scrum er ein arbeidsmetodikk som passar best for større multidisiplinære team. Prosessen involverer kunden etter hver iterasjon, eller «sprint» som det heter i Scrum-terminologien. Hver sprint varer typisk i to eller tre veker og ender med en fremvising av et ferdig produkt. [27]

eXtreme Programming (XP) er en utviklinsmetode som vil minimere tid brukt på design og planleggingsfasen, og heller fokusere på ofte og gode tilbakemeldingar frå kunde. Eit poeng er også å *«do the simplest thing that could possibly work»*, [28] og at ein skal programmere i lag, for å unngå misforståingar.

Unified Process (UP) består av fire fasar. *Oppstart* som er utviklinga av visjon og grove planskisser for prosjektet, *utgreiing* av krav og utvikling av kjernefunksjonar, *konstruksjon* av resterande funksjonalitet og start av implementasjon, og *overgangen* til betatesting og utrullinga av programvaren. [29] Det må understrekast at desse fire fasane ikkje er fire iterasjonar, men at kvar av desse fasane kan inneholde fleire iterasjonar. Det er også slik at vi ikkje er sekvensielt ferdig med prosessane (les: fossefall). Til dømes er ikkje visjonsdokumentet ferdig etter oppstartsfasen, men vert revidert gjennom heile prosessen.

2.3.2 Kodefilisofi

I unix-filisofien heiter det seg at program skal «gjere éin ting, og gjere den vel». Ein skal altså ha små men velfungerande modular, som skal fungere uavhengig av kvarandre [30]. På denne måten kan ein sikre at endringar i ein modul ikkje slår ut heile systemet, samstundes som at ein lagar eit fleksibelt og lett "hackbart"[2] system.

Frå objekt-orinentert programmering har vi også M-V-C prinsippet, som i grove trekk går ut på å dele koden i tre hovuddelar: [31]

Model Som tek seg av kommunikasjon med ytre system som database, filsystemet, kjernefunksjonar osv.

View er framvisinga av informasjon og innhenting av data frå brukaren

Controller gjer all kontroll og konvertering av data mellom framvisinga og det bakanforliggande systemet.

Ved å helde oss til desse prinsippa kan vi både lett halde oversikta når systemet verte komplekst, og enklare dele ansvarsområde for programmeringa mellom utviklarane.

[2] Hack som i å nytte systemet på alternativ måte, ikkje hacking som i datainnbrot.

Kapittel 3

Metode

I dette kapittelet vil vi gå gjennom og grunngi dei forskjellige vala av teknologi og greie ut om arbeidsprosessen fram mot det ferdige produktet.

3.1 Val av teknologi

3.1.1 Operativsystem

Valet av operativsystem var eigentleg meir eit val mellom linuxdistribusjonar. I Windowsverda får vi ikkje samme kontroll over kva modular vi vil ha med, og kan dermed ikkje på same måte skreddersy operativsystemet til det behovet vi har. I tillegg er dei fleste linuxdistibudjonar fritt tilgjeneleg, i motseting til Microsoft-system.

Debian er eit stabilt og anerkjent linuxsystem, og er «mor» til mange andre distroar, deriblant den brukarretta versjonen Ubuntu. Når vi vel å kjøre debian ar det to hovudgrunnar til dette. Ein av dei er at Debian har eit stort og svært aktivt utviklermiljø som prioriterer tryggleik og stabilitet. Dette gjer at vi kan stole på programvaren som følg med, men også at vi ikkje får siste skrik når det gjeld versjonar. Til dømes nyttar vi Python 2.6.6 sjølv om siste versjon av Python2 er 2.7.4, og Django 1.2.3 sjølv om siste versjon er 1.5. Den andre hovudgrunnen til at vi vel Debian er så enkel som at begge utviklarane har mykje erfaring med systemet. Det er også dette systemet som vert anbefalt for kurset «Linux Systemdrift» ved Høgskolen i Sør-Trøndelag.

Når dette er sagt, er der ikkje nokon grunn til at systemet vi har utvikla ikkje skal fungere på andre distribusjonar. Dei linuxmodulane vi nyttar er

felles for dei fleste system, og kommandoane er like.

3.1.2 Programmeringsspråk

Valet her har vore mellom å lage eit sett av bash-script med PHP som WebUI, og å kode eit system i Python. Vi har valt å nytte oss av høgnivå-språket Python med rammeverket Django for WebUI. Dette gjer at vi kan kode alt i eit programmeringspråk og gir ein enklare og tryggare måte å kommunisere mellom dei forskellige modulane. Å nytte Python gjer det også naturleg å skrive objektorientert kode og dele inn systemet etter MVC-prinsippet.

Det systemet som var utvikla gjennom Svein Ove Undal sitt haustprosjekt var laga med bash-script og PHP. Det hadde derfor vore eit enkelt val å fortsette med dette, og berre bygge på meir på dette systemet. Ettersom vi også såg etter måtar å utvide oppgåva på vart det naturleg å legge systemet om til Python. Dette gav også utfordringar i form av opplæring.

Ingen av utviklarane hadde frå før mykje erfaring med Python, noko som førte til at vi måtte bruke litt tid på å lære oss programmeringsspråket. Vi brukte noko tid på dette før vi endeleg bestemte oss for Python. Når vi etterkvart såg kor veleigna og enkelt dette språket var, bestemte vi oss også for å sjå nærmare på å nytte Django i staden for PHP. Det viste seg fort at å nytte Django gav mange gode effektar. Sidan Django har ein del ferdigutvikla modular for sesjonshandtering på nett, kunne vi konsentrere oss meir om kjernefunksjonane i systemet.

3.1.3 Programvare

For å få betre kontroll på brukarane av systemet og for å kunne bruke DHCP-hendingar som utløysar for aksjonar i koden, har vi valt å nytte ISC DHCP-server. Dette var også eit naturleg val da dette er den mest utbredete dhcp-serveren i *nix-verda. ISC DHCP-server sin funksjon for å utnytte DHCP-protokollen si mekanisme for fornying av TCP/IP-innstillingar gjer denne programvara spesielt egna. I tillegg er dette ei programvare som begge utviklarane har erfaring med frå kurset «Linux Systemdrift» ved Høgskolen i Sør-Trøndelag.

For å effektivt kunne lagre og hente ut data og statistikk frå systemet nyttar vi relasjonsdatabasen MySQL. Her var det også fleire andre alternativ, bl.a. har Django innebygd støtte for SQLite. Å bruke SQLite ville også gjort

at vi ikkje var avhengig av ein ekstern databaseserver, da SQLite lagrar alt i ei lokal fil. Når vi likevel valde å nytte MySQL var dette i hovudsak fordi vi har erfaring med denne frå før, og soleis var sikker på at alle databasefunksjonar vi hadde tenkt oss ville fungere.

Når vi også har valt å nytte Apache HTTP-server er det også tidlegare erfaingar som gjer at vi vel denne. Begge utviklarane har erfaring med konfigurasjon og oppsett, og har også her lært om denne i faget «Linux Systemdrift».

3.2 Systemarkitektur og strukturering av kode

Her vil vi gå gjennom de valgene som er gjort når det gjelder kodefilosofier og måter å bruke de forskjellige modeulene og programvarene på.

3.2.1 Kodestruktur

3.2.2 Tekniske løysingar

Tvungen pålogging til systemet blir gjort ved å lukke brannmuren for alle andre enn brukarar som alt er pålogga. Deretter brukar vi NAT for å avskjære trafikken og sende brukarane til ei påloggingside. Etter at brukarane her har logga på, åpnar vi brannmuren og fjernar avskjæringsfunkjonen for brukaren. Systemet vil no gå over i ein vanleg brannmur, gateway og ruter-funksjon for brukaren.

For å unngå at brukarar skal jukse seg gjennom brannmuren ved å stele andre sine IP-addresser krev vi at alle adresser er utdelt av DHCP-serveren. Ved innlogging vert adressa og mac-adressa registert og sjekka mot DHCP-serven sitt register. Det er kombinasjonen av brukar, passord og IP-addresse som vert sett på som ein unik brukar, men ved å stille inn konfigurasjonsfila for systemet kan administrator tillate fleire innloggingar pr brukar.

Automatisk utlogging ved tidsavbrot gjer vi ved å utnytte oppførselen til klientar av DHCP-protokollen. I DHCP-serveren si konfigurasjonsfil legg vil til nokre linje som gjer at CLI-kommandoen for utlogging av brukarar blir kjørt for IP-addresser som ikkje fornyar DHCP-leasen sin. På denne måten sikrar vi at pcar som ikkje er kopla til nettet automatisk vert logga ut.

Overvåking av linja gjer vi fortløpande, og for å spare på ressursar overvakar vi i utgangspunktet kun totaltrafikken, men loggar trafikkmengda for kvar brukar. Først når systemet oppfattar at linja ut har kapasitetsproblem ser vi på trafikkmengda for kvar brukar, og avgjer om nokon må avgrensast. Vi vil etter dette følge brukaren, og sleppe brukaren fri frå avgrensingane når trafikkmengda frå brukaren er på eit akseptabelt nivå.

Avgrensinga vert gjort ved at vi i brannmuren har laga eit sett med reglar som fungerar som eit «fengsel». Brukaren er ikkje lenger fri til å bruke kapasitet, og all trafikk frå brukaren vert først flagga når dei kjem inn til systemet, og trafikk som går ut over fastsette grenser blir forkasta. .. ICMP-TIMEOUT? eller berre dROP?

3.3 Prosessen

Som utviklingsprosess har vi valt å nytte *Unified Process* (UP). Som alternativ til denne var også *SCRUM* nevnt, men utviklarane såg ikkje på dette som ein føremålstenleg utviklingsprossess når det var tre personar – to utviklarara og ein rettleiar – involvert. Vi var også inne på tanken om å nytte *eXtreme Programming*, men sidan begge utviklarane har mest erfaring med UP, falt valet på denne utviklingsprosessen.

Vi laga ikkje noko tradisjonelt GANT-diagram for utviklingsprosessen, men laga i staden ein tabell med mål og tidsfristar for når forskjellige utviklingsteg skulle vere ferdig. Vi jobba deretter iterativt fram mot desse, og har testa kvar modul undervegs. Dette vart gjort for vi av erfaring veit at det kan bli lite tid til testing mot slutten, og da var dette ein måte å være sikker på at testing vart gjennomført for kvar modul.

Det ligg ikkje føre noko forstudierapport for dette bachelorprosjektet, da det er ei viderføring av eit haustprosjekt, og forstuderapport dermed ikkje var påkrevd.

Første del av prosessen gjekk for det meste med til eiga opplæring, både i Python og deretter Django-rammeverket. Deretter brukte vi tida på å oversetje bash-script frå haustprosjektet til fungerande Python-program. Når dette var utført fekk systemet på mange måtar ein ilddåp, sidan vi skulle teste dette på TIHLDE-LAN . Dette var tidleg i prosessen og mykje av programvara var uferdig. Vi hadde CaptivePortal-funksjonaliteten på plass, og

konsentrerte oss dermed om å lage statistikk-funksjonalitet for å samle mest mogleg data.

For å få testa mest mogleg funksjonalitet – og rette nokon feil – nytta vi under TIHLDE-LAN element frå utviklingsprosessen eXtreme Programming. Under lanet vart mange av metodane for avgrensing av bruk og samling av trafikkdata utvikla og testa. Vi brukte her ei form for parprogrammering der ein programmerte, medan den andre las over koden før den deretter vart testa. Dette gav gode resultat om ein ser på antall utvikla modular og linjer, men gav også eit stort etterarbeid i for dokumentasjonsdelen.

Kapittel 4

Resultat

4.1 Faglige resultat

4.1.1 Måloppnåing

4.1.2 Kommentar til UseCase-ar

(UC)

UseCase 1: Logg inn

Innloggingа er fungerande og enkel for brukaren å utføre.
scpr. innlogging?

Use Case 2: Vis statistikk

Når kvar brukar har logga seg inn, vil nettoppslag mot innloggingstjenaren videresende brukaren til ei statistikkside. Her vi brukaren kunne sjå kor mykje han har lasta ned totalt, kor mange tilkoplingar han brukar og kva som er gjennomsnittleg linjebruk for økta. Brukaren vil også få informasjon her dersom det er sett restriksjonar for linja han har.

Use Case 3: Administrer Brannmur

Administrator vil få tilgang til eigne administrasjonsider der dei kan sjå på innstillingane, få oversikt over aktive brukarar og eventuelt legge til nye. Her er også ei eiga side for å legge til eller fjerne reglar i brannmuren. Noko som ikkje kjem fram i UCen, men vart naudsynt er at brukaren først må velje kva del av brannmuren han vil lage regel for. Vi har eit sett reglar for innkommande til server, og eit sett for trafikk som skal vidaresendast.

4.2 Administrative resultat

Kapittel 5

Drøfting

5.1 Kvifor dette resultatet?

5.2 Svakheiter ved produktet

5.3 Styrkar ved produktet

5.4 Vidareutvikling av produktet

Kapittel 6

Konklusjon

Kapittel 7

Kjelder og referansar

Mesteparten av litteraturen er på engelsk. Vi har derfor valt å merke det som er på norsk med [NORSK]. Dersom denne merkinga ikkje er i referansen, er kjelda engelsk.

Litteratur

[1] *Introduction to Captive Portal*, Cisco Systems, Inc, 07.05.2013. [Online]. Kopling: http://www.cisco.com/en/US/docs/net_mgmt/subscriber_edge_services_manager/3.3/administration/guide/config/ad_cfcp.html#wp1219317

[2] (07.05.2013) Debian 7.0 "wheezy" released. Software in the Public Interest. [Online]. Kopling: http://www.debian.org/News/2013/20130504

[3] *Package: python2.6 (2.6.6-8 and others)*, Debian Project, 07.05.2013. [Online]. Kopling: http://packages.debian.org/squeeze/python2.6

[4] (07.05.2013) About python. Python Software Foundation. [Online]. Kopling: http://www.python.org/about/

[5] (07.05.2013) Application domains. Python Software Foundatuion. [Online]. Kopling: http://www.python.org/about/apps/

[6] *17.1 subprocess – Subprocess management*, 2nd ed., Python Software Foundatuion, 2013. [Online]. Kopling: http://docs.python.org/2.6/library/subprocess.html

[7] (07.05.2013) Meet django. Django Software Foundation. [Online]. Kopling: https://www.djangoproject.com/

[8] *How to use sessions*, Django Software Foundation, Mai 2013. [Online]. Kopling: https://docs.djangoproject.com/en/1.2/topics/http/sessions/

[9] *python-django (1.2.3-3+squeeze5) [security]*, Debian Project, 07.05.2013. [Online]. Kopling: http://packages.debian.org/squeeze/python-django

[10] *Package: bash (4.1-3)*, Debian Project, Mai 2013. [Online]. Kopling: http://packages.debian.org/squeeze/bash

[11] C. Ramey. (2013, April) The gnu bourne-again shell. [Online]. Kopling: http://tiswww.case.edu/php/chet/bash/bashtop.html

[12] ——. (2013) Bash - the bourne-again shell. [Online]. Kopling: http://tiswww.case.edu/php/chet/bash/bash-intro.html

[13] (07.05.2013) General information [about php]. The PHP Group. [Online]. Kopling: http://no1.php.net/manual/en/faq.general.php

[14] (07.05.2013) Package: php5 (5.3.3-7+squeeze15). Debian Project. [Online]. Kopling: http://packages.debian.org/squeeze/php5

[15] (2013, May) May 2013 web server survey. Netcraft Ltd. [Online]. Kopling: http://news.netcraft.com/archives/2013/05/03/may-2013-web-server-survey.html

[16] G. Dumpleton, *Python WSGI adapter module for Apache.*, 07.05.2013. [Online]. Kopling: https://code.google.com/p/modwsgi/

[17] (Oktober 2006) Php 5.4.14 (current stable) complete source code. The PHP Group. [Online]. Kopling: http://www.php.net/downloads.php

[18] (2013, Mai) Mysql community edition. Oracle Corporation. [Online]. Kopling: http://www.mysql.com/products/community/

[19] (2013, Mai) Appropriate uses for sqlite. Hipp, Wyrick & Company, Inc. [Online]. Kopling: http://www.sqlite.org/whentouse.html

[20] (2013, Mai) Limits in sqlite. Hipp, Wyrick & Company, Inc. [Online]. Kopling: http://www.sqlite.org/limits.html

[21] T. Lemon, *dhcpd configuration file*, Internet Systems Consortium / Vixie Labs, Mai 2013. [Online]. Kopling: http://www.linuxmanpages.com/man5/dhcpd.conf.5.php

[22] ——, *Dynamic Host Configuration Protocol Server*, Internet Systems Consortium / Vixie Labs, Mai 2013. [Online]. Kopling: http://www.linuxmanpages.com/man8/dhcpd.8.php

[23] (2013, Mai) The netfilter.org project. Netfilter Core Team. [Online]. Kopling: http://netfilter.org/

[24] M. A. Brown. (2006, Oktober) Introduction to linux traffic control. [Online]. Kopling: http://www.linux.com/learn/docs/ldp/783-traffic-control-howto#intro

[25] O. S. Ø. Hallsteinsen, B. Klefstad, *Innføring i datakommunikasjon*, 2nd ed. Stiftelsen TISIP og Gyldendal Norsk Forlag, 2010, [NORSK].

[26] R. Droms, "Dynamic host configuration protocol," RFC 2131, Internet Engineering Task Force, Mars 1997, del 3.1, 4.1 - 4.4. [Online]. Kopling: http://tools.ietf.org/html/rfc2131#section-3.1

[27] G. o. B. C.Larman, P.Deemer, *The Scrum Primer*, 2nd ed. Good Agile, 2012. [Online]. Kopling: http://www.goodagile.com/scrumprimer/scrumprimer20.pdf

[28] D. M. Stephens, *Extreme Programming Refactored: The Case Against XP*. Apress, 2003.

[29] C. Larman, *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development*, 2nd ed. Prentice Hall PTR, 2002. [Online]. Kopling: http://books.google.no/books?id=r8i-4En_aa4C

[30] E. S. Raymond, *The Art of Unix Programming*. Addison-Wesley Professional, 2003. [Online]. Kopling: http://www.faqs.org/docs/artu/ch01s06.html

[31] P. Steve Burbeck. (1992) Applications programming in smalltalk-80(tm): How to use model-view-controller (mvc). [Online]. Kopling: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html

Del III

Vedlegg

Tillegg A

Testrapport frå TIHLDE-LAN

Tillegg B

Visjonsdokument

Nummereringa på sidetala følgjer no vedlagt dokument.

Visjonsdokument
Dynamisk Nettverksbrannmur

Espen Gjærde Svein Ove Undal

11.04.2013

Revisjonshistorie

Dato	Versjon	Forklaring	Forfattar
01.02.2013	1.0	Dokumentet oppretta	Espen, Svein
15.02.2013	1.0	Dokument klar for revisjon	Espen, Svein
09.04.2013	1.1	Tabellar oppdatert, små rettingar	Espen
11.04.2013	2.0	Oppdatert mtp. django	Espen
16.04.2013	2.1	Kost/Nytte-analyse oppdatert	Espen, Svein

Innhald

1 Innleiing

Dette dokumentet er ei analyse av prosjektet som skal gjennomførast, med mål om å avgjere om prosjektet bør startast. Vi vil i dokumentet gå gjennom bakgrunn og mål for prosjektet, greie ut om interessentar og utføre kost/nytte og risikoanalyse. Vi vil også gå gjennom teknologival, standardar for gjennomføring og suksessfaktorar.

2 Bakgrunn for prosjektet

Prosjektet er ei bacheloroppgåve for Espen Gjærde og Svein Ove Undal. Begge studentar ved Avdeling for Informatikk og e-Læring ved Høgskolen i Sør-Trøndelag. Prosjektet går ut på å lage ein smart brannmur basert på pålogging via web. Etter dette skal brukaren få tilpassa reglar for sin bruker.

2.1 Skildring av problem og behov

For administratorar av nettverk med mange brukarar – og mange forskjellige typar brukarar – kan det være vanskeleg å halde oversikt over nettilgang og effektivt og rettferdig fordele bandbredde og tilgangar mellom brukarane. Spesielt gjeld dette midlertidige nettverk - tildømes dataparty, eller trådlause gjestenett. Vi tek sikte på å lage eit system som sikrar rettferdig og dynamisk deling av bandbredde, samstundes som det gir rom for å spesialisere reglar for den enkelte brukar.

2.2 Skildring av dagens system og rutiner

Det finnast i dag døme på system der brukarar må logge inn for å nytte seg av ressursar. Dette er gjerne måten adgangskontroll for trådlausnett vert gjort på på flyplassar og hotell. Dette er det som blir kalla captive-portal[1]. Problemet med slike løysingar er at det enten er fritt fram etter du har logga inn, eller at kapasiteten blir delt etter eit ¡¡one-size-fits-all¿¿-prinsipp. Systema har og veikskapar som macadresse-spoofing [2] og DNS-tunnel som ein måte å unngå autentifiseringa heilt.

[1] Captive-Portal: System som tvingar brukarar via ei bestemt nettside eller gjennom eitt spesielt punkt.

[2] forfalsking av unik adresse eller identifikasjon

3 Prosjektmål

Overordna prosjektmål

- lage eit effektivt brannmursystem som er enkelt å nytte seg av
- tileigne oss meir erfaring og kompetanse
- lage eit solid og stabilt sluttprodukt

For å beskrive måla med prosjektoppgåva er det hensiktsmessig å dele måla inn i føljande kategoriar:

Effektmål — skildrar effekten sluttbrukar får av å bruke systemet

Resultatmål — skildrar kva som skal ligge føre når prosjektet er ferdig

Prossessmål — mål utviklarane har med å delta i prosessen

3.1 Effektmål

- Enklare administrasjon av brukarar i mellombels nett
- Dynamiske brannmurreglar og enkel administrering av desse

3.2 Resultatmål

- Eit fullverdig brannmursystem med pålogging
- Ein fungerande brannmur med individuelle reglar for kvar brukar eller gruppe
- Eit enkelt og oversiktleg administrasjonsystem for brannmuren.
- Eit system som sjekkar nettverkskonfigurasjon for pålogga maskiner.
- Ein pakke med enkel installasjon - gjerne .deb eller tar.gz.

3.3 Prosessmål

- Vidareutvikle kunnskapar om linux og nettverstryggleik
- Erfaring med utvikling i Python
- Utvikling av eit open source distribuert system

3.4 Omfang

- Det skal vere alternativ å lage informasjon som filer, eller i ein database
- Det skal implementerast eit oversiktlig administrasjonspanel
- Prosjektet skal være ferdigstilt innan 25.mai 2013.
- Systemet blir utvikla og testa for Debian 6.0.6 Squeeze"

3.5 Prosjektets milepælar og hovudaktivitetar

3.5.1 Dokumentasjon

Dato	Milepæl
04.02.2013	Oppstart av prosjektet
15.02.2013	Visjonsdokument til revisjon
10.04.2013	Kravdokument til revisjon
01.04.2013	Arkitekturdokument til revisjon
25.05.2013	Sluttrapport ferdig og levert

3.5.2 Utviklingsteg / Mål

Dato	Milepæl
15.02.2013	Captive-Portal i funksjon
20.02.2013	Brannmur og pålogging styrt av Python
08.03.2013	Prototype klar til test på TIHLDE-LAN
01.04.2013	Administrasjonspanel i testversjon
01.05.2013	Release Candidate 1 klar

3.6 Teknologi

Vi vil basere prosjektet på eksisterande teknologi og programmer med open kjeldekode. Vi vil utvikle systemet i Debian Linux, men systemet skal kunne kjøre på alle linuxplattformer. Mykje av dei programma vi nyttar finnast også til andre *nix-system, og burde med enkle steg kunne nyttast her også. Nedanfor vil vi kort grunngi forskjellige val av teknologi.

Debian Linux Wheezy Debian er den Linuxdistibusjonen med open kjeldekode som er sikrast og mest utbreidd. Det gjere Debian som eit lett val til vårt prosjekt. I tillegg til dette er Debian den linuxversjonen vi har mest erfaring med.

Python Som kodespråk mot systemet nyttar vi oss av Python. Slik får vi eit abstraksjonsnivå mellom kode og system, som vi ikkje får med bash-script. Dette gjer det også enklare å portere systemet over til andre plattformer enn Debian Linux.

Django Som brukargrensesnitt vil vert rammeverket Django nytta. Dette er eit web-rammeverk som er bygd i Python, og vi kan dermed få direkte tilgang til den koden vi utviklar i sjølve brannmursystemet. Dette gjer det meir effektivt og enklare å knytte saman brukergrensesnittet og det bakanforliggande systemet.

ISC DHCP-server Dette er ein av dei mest utbredte dhcp-tjenarane i marknaden, og er den klart mest brukte i linux-verda. Vi finn det naturlig å nytte denne både fordi kjeldekoden er open, og fordi det gjer eventuell interaksjon mellom systemet vi utviklar og eventuelle eksisterande system enklare.

IPtables For sjølve brannmuren vil vi nytte iptables. Dette er en enkel pakkefiltrerande brannmur som er støtta i dei fleste *nix-systemer.

Brukardatabase - PAM Vi vil kjøre autentisering gjennom Linux PAM[3]. Dette gjer at sluttbrukar står fritt til å velje brukardatabase. For demonstrasjon og testing vil vi bruke OpenLDAP og FreeRadius som brukardatabasar.

[3] Pluggable Authentiction Modules: Eit fleksibelt authentiseringsystem. Gir moglegheit for mange forskjellige typar brukardatabasar.

4 Interessentar og rammevilkår

4.1 Interessentanalyse

Interessent	Suksesskriterie	Bidrag til prosjektet
Eksterne interessentar		
Høgskolen i Sør-Trøndelag	Dokumentasjon og kode levert innan frist	Oppdaragsgiver
TIHLDE	Programvare stabil under testing	Tilgang til å teste programmet under TIHLDE sitt dataparty.
Interne interessentar		
Studentane	Strukturert jobbing	Systemutviklarar og prosjekteigarar
Rettleiar	God kommunikasjon, hyppig oppdatering	Kunnskap, rettleiing

4.2 Rammevilkår

- Produkt og dokumentasjon ferdig innan 25.mai 2013
- Produktet skal kunne installerast og brukast av ein brukar utan store krav til linux/nettverks kunnskap.
- Systemet er utvikla og testa i Debian 6.0.6 "Wheezy"
- Utviklingsmetoden Unified Process vert nytta.
- Systemet vert basert på og utgitt som open kjeldekode.

5 Kritiske suksessfaktorar

5.1 Suksessfaktorar

Saumlaust system, som fungerer uten at sluttbrukar merker at systemet er der. Systemet skal effektivt gjere nettverket betre å bruke med tanke på ping og yting under stress.

Systemet skal vere særdeles lett å settast opp, det skal ikkje krevst høge kunnskapar innan linux og nettverksadministrasjon for å få systemet til å fungere.

5.2 Informasjonsbehov

Rettleiar skal haldast oppdatert om framgangen i prosjektet

6 Risikoanalyse

A Langvarig sjukdom/fråvær av gruppemedlemmer

B Samarbeidsvanskar i teamet

C Feil på programvare vi er avhengig av

D Produkt ikkje i kjørbar versjon ved testtid

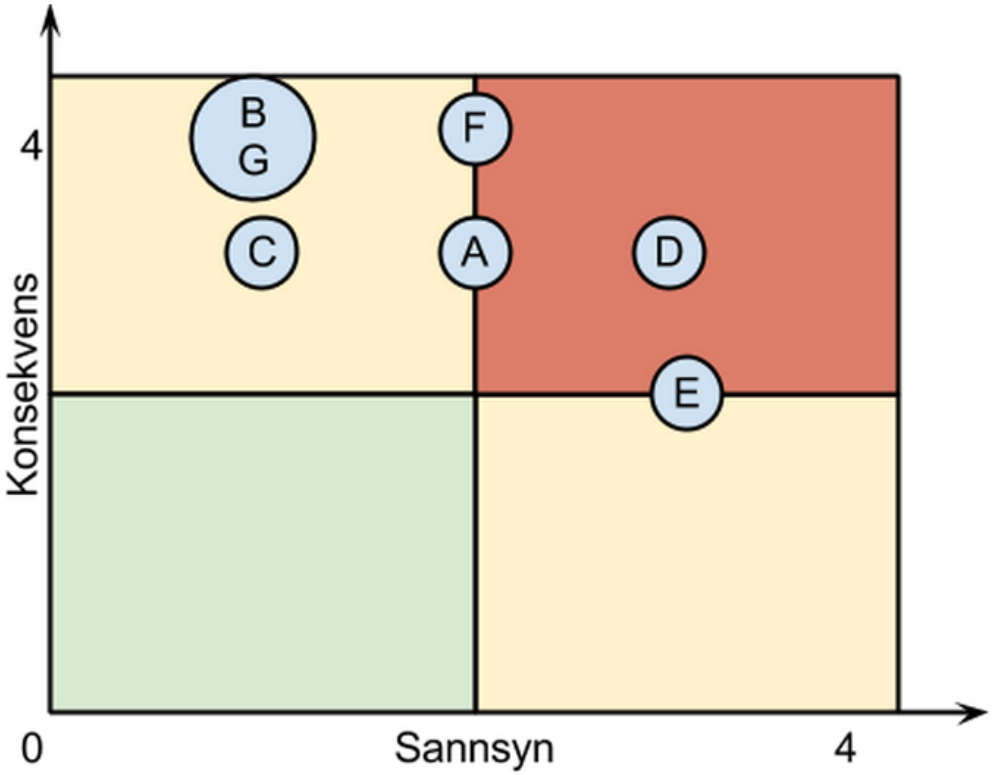
E Programvare oppfører seg ikkje som forutsett

F Vanskeleg brukargrensesnitt

G For høge systemkrav

H Tap av viktig data

Sjå Vedlegg A for risikoscore og utrekning av sannsyn og konsekvens



Figur 1: Identifiserte risikoar plassert i koordinatsystem.

7 Kost/nytte-analyse

Dette er eit bachelorprosjekt som baserer seg på bruk av open kjeldekode og alle nytta verktøy er opne og fritt tilgjengeleg. All kjeldekode vil også verte publisert under ein open lisens. Det er derfor vanskeleg å sette opp gode kostnadsvurderingar; arbeidskraft, materia og verkty er gratis. Likevell vert det sett opp ei kost/nytte-analyse der ein tek utgangspunkt i sal av nettilgang.

7.0.1 Kvantifiserbar nytte

Mindre bruk av eksterne konsulentar Systemet let seg lett administrere via eit nettbasert brukargrensesnitt, og vi kan derfor medrekne mykje mindre bruk av konsulentar. Å vedlikehalde og justere ein brannmur og eit tilgangsystem er mykje arbeid. Vi reknar med 10 timar i månaden, altså 120 timar pr år. Ved å gå over til dette systemet, som er enkelt å administrere kan vi spare inn halvparten av dette, då dei avanserte innstillingane i brannmuren justerer seg sjølv – ut i frå tilgjengeleg kapasitet.

Betre utnytting av nettlinje Ved at dette systemet automatisk lastbalanserer den ledige kapasiteten mellom brukarane i nettverket, kan fleire brukarar bruke same linja. Systemet sikrar at ingen kan stelelinja og brukaropplevinga vil bli betre. Brukartalet skal kunne aukast med 25% på same nettlinje.

Auka sal av nettilgang I tillegg til dette vert det i kost/nytte-analysa teke utgangspunkt i at systemet vert nytta ein stad der ein skal selje internetttilgang. Timepris for tilgang vert da sett til 60kr pr brukar. Vi tek utgangspunkt i at det vert i snitt seld 60 brukartimar kvar dag, noko som gir ei årsomsetning på (60*365)*60 = 1 314 000 kr. Dette kan vi potensielt auke med 25% utan å måtte ha dyrare nettlinje.

7.0.2 Ikkje-kvantifiserbar nytte

- God kompatibilitet med andre brukardatabasar
- Lykkelege brukarar

7.1 Estimerte kostnadar

Utviklingskostnader Prosjektgruppa består av to utviklarar som kvar har 450 timar til disposisjon. Som utgangspunkt for kostnaden vert det rekna

med ein timekostnad på 450kr pr utviklar. Utviklingskostnadene er altså 2*450*450 = 405 000 kr

Maskinvare Systemet har låge systemkrav, og skal kunne kjørast på ein minipc. Einaste krav er to nettverksportar. Vi har testa systemet på ARM-pcen RaspberryPI[4] med eit ekstra USB-nettverkskort. Kostnaden for dette utstyret er berre 350kr, men ein må og rekne med litt ekstrautstyr.

Utstyr	NOK
Raspberry PI	350
Min 4GB Lagring	100
Kabinett	100
Straumforsyning	200
Ekstra nettverkskort	200
I/O-utstyr	400
Total	1450

Tabell 1: Estimerte prisar på maskinvareutstyr

7.2 Samanstilling av kostnadar og nytte

Forklaring	1.år	2.år	Total
Lågare vedlikehaldskostnad	30 000	30 000	60 000
Auka salsinntekter	328 500	328 500	657 000
Maskinvare	- 1 450		- 1 450
Utviklingskostnader	- 404 550		- 405 000
Total			**252 550**

Tabell 2: Samenstilling av kost vs. nytte

[4]http://raspberrypi.org/faqs Sist vitja 16.04.2013

8 Retningslinjer og standardar

8.1 Krav til dokumentasjon

Endeleg versjon av følgjande dokumentasjon skal være klar 5.mai:

- Visjonsdokument
- Kravdokument
- Arkitekturdokument

Vidare skal følgjande dokumentasjon foreligge ved prosjektinnlevering:

- Brukardokumentasjon
- Installasjonsrettleiing
- Sluttrapport

8.1.1 Utforming og digitale format

Følgjande krav vert sett til levert dokumentasjon

- Dokumentasjonen vert utforma i LaTeX
- Dokumentasjon vert tilgjengeleg i PDF-format
- Framsida av sluttrapporten skal være etter mal frå HiST Avdeling for Informatikk og e-Læring

8.2 Krav til kvalitetskontroll

Kvalitetsgjennomgang blir tildels dekka av dei daglege utviklingsmøta i prosjektgruppa. Vi får her tilbakemeldingar om eventuelle endringa som burde vore utført, og om det må føyast til noko.

Prosjektgruppa må også ha en gjennom gang av følgjande:

- Pythonkode
- Django / HTML-kode
- Brukarvennligheit på nettsida
- Databasestruktur og tryggleik

I tillegg til å kvalitetsikre, testar vi programvara undervegs og etter kvar endring. Testinga undervegs blir ein peikepinne på om systemet fungerer som det skal, samt at vi utbetrar alle problema når dei oppstår. Vi vil òg ha ei utbredt testing av ein prototype på TIHLDE-lan, som vil effektivt vise veikskapar og styrkar i systemet.

8.3 Krav til standardar og metodar

Systemet skal bruke kjent teknologi og basere seg på innførte standardar. Dette fordi det for sluttbrukar sine klientar ikkje skal være behov for spesialutstyr eller eigne klientprogram.

8.3.1 Programmeringstandardar

- Programmering skjer i språka HTML og Python.
- Alle metodar skal ha engelske navn, og følge kjente namnekonvensjonar i sine respektive språk.
- Kommentarar og forklaringar i kodefilene skal skrivast på engelsk.

8.3.2 Konfigurasjonsfiler

- Konfigurasjonsfiler og programvare skal i størst mulig grad følge standardar og ligge der det er naturlig i eit linux / Debian-system.
- Kommentarar og forklaringar i filene skal skrivast på engelsk

8.3.3 Bruk av verktøy

- Versjonskontroll skjer med versjonshandteringssystemet GIT.
- Systemet blir testa på ei tjenermaskin med operativsystemet Debian 6.0.6 ¡¡Squeeze¿¿.

8.3.4 Andre standardar

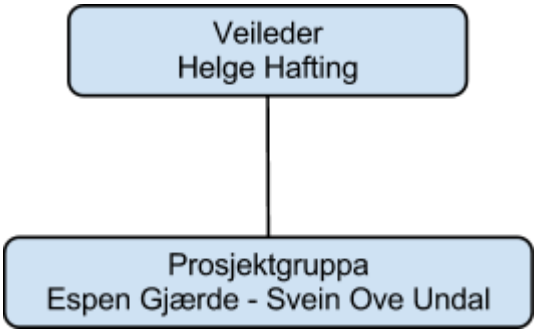
- Systemet skal baserast på kjente og de-facto standardprotokollar som IP, TCP, DHCP og HTTP.
- Språk for administrasjonsystemet skal være norsk - nynorsk.

8.4 Endringshandtering

1. Eventuell mistanke om problem meldast tidleg til dei andre i prosjektgruppa
2. Det skal arbeidast med å få oversikt over endringar og ringverknadar.
3. Endringa dokumenterast og avvik rapporterast
4. Tidsplanar justerast
5. Endringa blir gjennomført
6. Evaluering av endring og endringsprosessen vert gjennomført

9 Prosjektorganisering

Prosjektorganisasjonen består av ei prosjektgruppe, studentane, og rettleiar som også utgjer styringsgruppa.



Figur 2: Prosjektorganisasjonen

10 Tilråding om vidare arbeid

Med bakgrunn i kost/nytte-analysa og dette visjonsdokumentet elles, vert det tilrådd at prosjektet vert utvikla vidare.

Risikoanalyse

ID	Kategori	Hending	Sannsyn	Konsekvens	Risiko	Tiltak
A	Personell	Langvarig sjukdom eller anna fråvær	2	3	6	Deling av all dokumentasjon, informasjon, kode og notat
B	Personell	Samarbeidsvanskar i team	1	4	4	Teambuilding, aktivitetar og god kommunikasjon
C	Avhenigheter	Feil på programvare vi er avhengig av	1	3	3	Så langt som mogleg nytte programvare som er i stable-versjon.
D	Test	Produkt ikkje i kjørbar versjon til avtalttestid	3	3	9	Utvikle i små steg, alltid utføre enkle testar etter omprogrammering
E	Programvare	Programvare oppfører seg ikkje som forutsett	3	2	6	Gjere god research, finne eventuelle alternativ
F	Sluttprodukt	Vanskeleg brukergrensesnitt	2	4	8	Testing og prototyping. Få tilbakemeldingar fra ikkje-teknologar
G	Utvikling	Tap av viktig data	1	4	4	Hyppig opplasting av kode, backup.

Sannsyn	Forklaring
1	Lite truleg at hendinga skjer. Sjeldnare enn kvart 5. år.
2	Hendinga kan inntreffe. Skjer omlag annankvart år.
3	Det er truleg at hendinga inntreff. Årleg hending.
4	Det er vanleg at hendinga inntreff. Skjer meir enn ein gong i året.

Konsekvens	Forklaring
1	Ubetydleg. Skade kan lett utbetrast.
2	Liten konsekvens, kan påvirke tidsfristar.
3	Store konsekvensar, vil påverke gjennomføring og sluttprodukt
4	Katastrofale konsekvensar. Vil vå betydlige konsekvensar for sluttprodukt.

Utrekning av risiko Risikoen blir utrekna som eit produkt av sannsynet for ei hending og konsekvensen av hendinga. Dette gir eit tal mellom 1 og 16, der 1 er lav risiko og 16 er ei katastrofe som kjem til å skje.

Tillegg C

Kravdokument

Nummereringa på sidetala følgjer no vedlagt dokument.

Kravdokument
Dynamisk Nettverksbrannmur

Espen Gjærde Svein Ove Undal

09.04.2013

Revisjonshistorie

Dato	Versjon	Forklaring	Forfattar
18.02.2013	1.0	Dokumentet opprettet	Espen
04.04.2013	1.1	Kapittel 1 og 3 på plass	Espen
08.04.2013	1.2	Tabellar UC og SSD	Espen

Innhald

1 Innleiing

1.1 Hensikta med dokumentet

Hensikta med dette dokumentet er å gi ei oversikt over krava som vert stilt til bruk systemet, og eventuelle tilleggsfunksjonar som er ønska. Dokumentet sin viktigaste funksjon er å skildre systemet og systemet sitt grensesnitt. Dette blir gjort ved bruk av standardiserte logiske modellar.

1.2 Avgrensingar

Innhaldet i dette dokumentet skildrar den funksjonaliteten som produktet gir, samt det som vi har programmert, men ikkje funksjonaliteten til dei linux-modulane som produktet nyttar seg av.

1.3 Definisjonar og forkortingar

Som definert i kapittel 8.3.4 i Visjonsdokumentet nyttar vi nynorsk i skriftleg dokumentasjon, men engelsk i all kode og alle kodekommentarar. Det kan derfor skje at vi refererer til det engelske navnet «Dynamic Network Firewall» eller «DNF» også i den norske dokumentasjonen.

1.4 Referansar

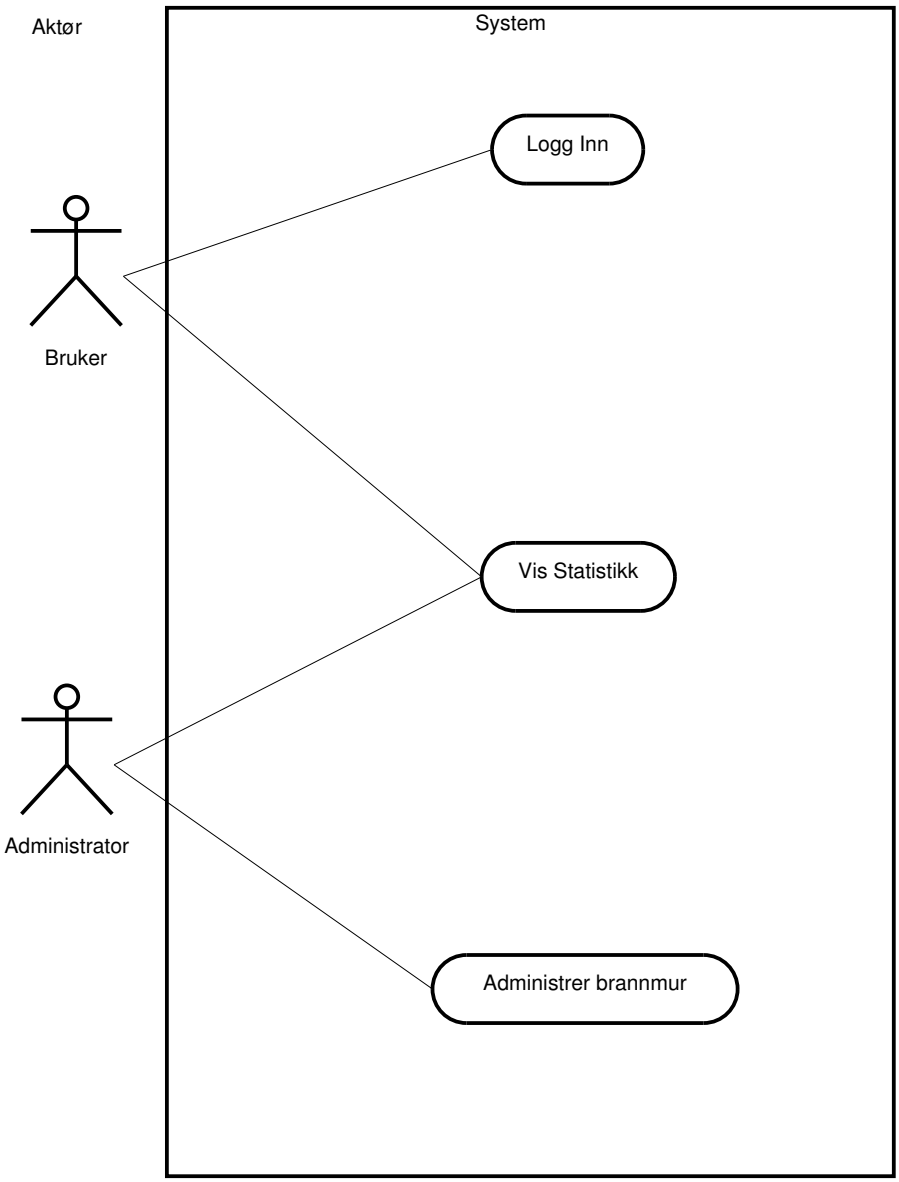
Visjonsdokument versjon 1.2, side 15

1.5 Oversikt over innhald

Dette dokumentet har i alt fem kapittel. I første kapittel vert bakgrunn og grunnlag for sjølve dokumentet forklart. Vidare vil kapittel to og tre ta for seg use caser der dei først blir skildra med logiske modellar, og deretter skriftlig forklart. I kapittel fem går vi gjennom dei same usecasane frå systemet sin ståstad og fokuserer meir på det tekniske.

2 Bakgrunn og oversikt

2.1 Use Case – UML-diagram



3 Detaljerte krav

3.1 Use Case: Logg inn

Namn	Logg inn
Mål	Authentisere brukarar og gi tilgang gjennom brannmuren
Aktør	Brukar, administrator
Utløysar	Brukar koplar seg til internett før pålogging
Føresetnad	Brukaren er ikkje pålogga
Effekt	Brukaren får tilgang til ressursar bak brannmuren, vert innlogga.
Hovudløp	

1. LOGG INN

Relaterte løp	Ingen
Unntak	Ingen
Tilleggsinformasjon	Ingen

3.2 Use Case: Vis statistikk

Namn	Vis statistikk
Mål	Vise brukaren statistikk over sin nettbruk, informere om eventuelle avgrensingar sett for brukaren.
Aktør	Brukar, administrator
Utløysar	Ingen
Føresetnad	Brukaren er logga inn
Effekt	Ingen
Hovudløp	

1. VIS STATISTIKK

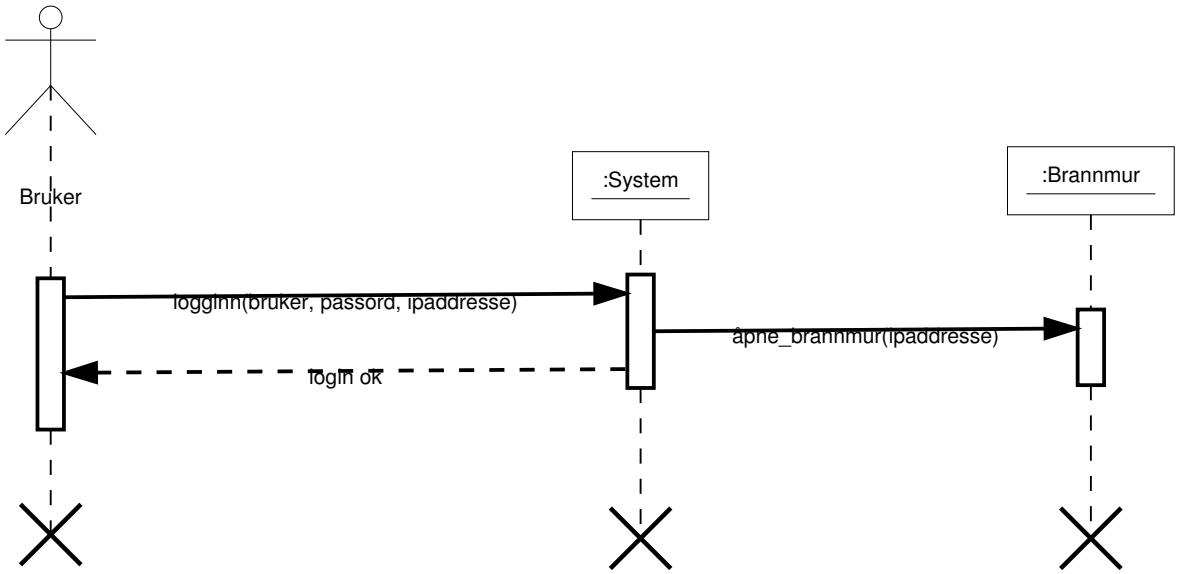
Relaterte løp	Ingen
Unntak	Ingen
Tilleggsinformasjon	Ingen

3.3 Use Case: Administrer brannmur

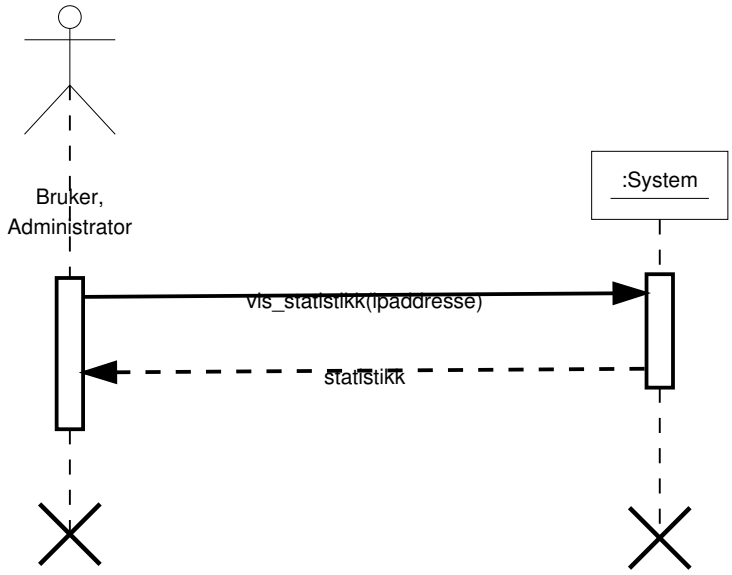
Namn	Administrere brannmur
Mål	Manuelt sette eller oppdatere avgrensing og reglar i brannmuren
Aktør	Administrator
Utløysar	Ingen
Føresetnad	Pålogga som administrator
Effekt	Brannmuren vert oppdatert
Hovudløp	1. HENT REGLAR 2. GJENTA 3-5 SÅ LENGE BRUKAREN ØNSKER 3. ELLER: ENDRE REGEL 4. ELLER: LAG REGEL 5. ELLER: SLETT REGEL 6. OPPDATER BRANNMUR
Relaterte løp	Ingen
Unntak	Ingen
Tilleggsinformasjon	Ingen

4 Systemsekvensdiagram

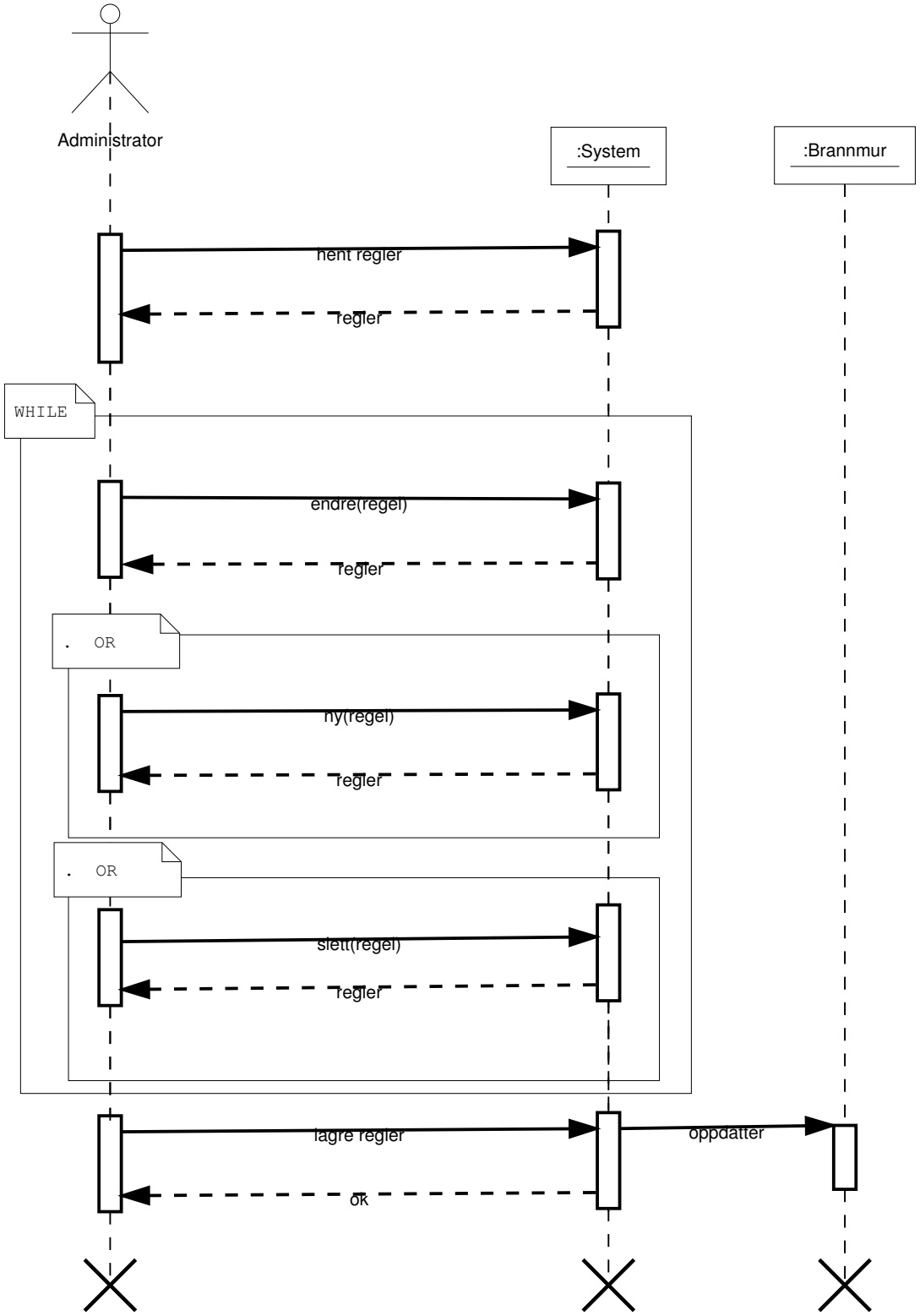
4.1 Use Case: Logg inn



4.2 Use Case: Vis statistikk

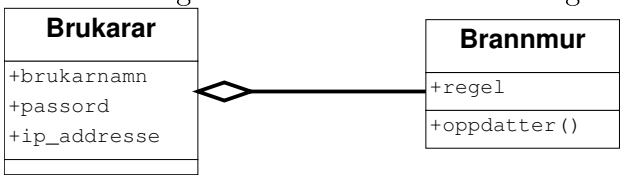


4.3 Use Case: Administrere brannmur



5 Problemdomenemodell

Reglar kan enten være knytta til ein brukar, eller gjelde heile systemet. Ein brannmurregel har altså ein null til mange relasjon med brukarane.



Tillegg D

Arkitekturdokument

Nummereringa på sidetala følgjer no vedlagt dokument.

Arkitekturdokument
Dynamisk Nettverksbrannmur

Espen Gjærde Svein Ove Undal

11.04.2013

Revisjonshistorie

Dato	Versjon	Forklaring	Forfattar
15.04.2013	1.0	Dokumentet oppretta	Espen
16.04.2013	1.0	Diagram og illustrasjonar oppdatert	Espen

Innhald

1 Innleiing

1.1 Hensikta med dokumentet

Dette dokumentet skal beskrive arkitekturen i systemet som blir utvikla. Dokumentet vil gå djupare inn i dei forskjellige systema som vert nytta og korleis desse heng saman. Også her vert det nytta standardiserte logiske modellar for å gje ei betre oversikt over systemet.

1.2 Avgrensingar

Dokumentet skildrar systemet som vert utvikla, og relasjonar til andre system. Det vil ikkje skildre eksterne system som vert nytta.

1.3 Definisjonar og forkortingar

Ord	Forklaring / Definisjon
IPtables	Pakkefilter /-manipulator tilgjengeleg i dei fleste linuxdistribusjonar
Linux-distibusjon	Variant av operativsystemet linux
WebUI	Nettside som kontrollerer eit system/program.
DNF	Systemet som vert utvikla (*Eng: Dynamic Network Firewall*)
Ruting	Sending/Vidaresending av pakkar i eit nettverk
Prerouting	(*Nor: Før-ruting*) behandling av pakkar før dei blir ruta
Daemon	Program som kjører i bakgrunn og utfører oppgaver automatisk.

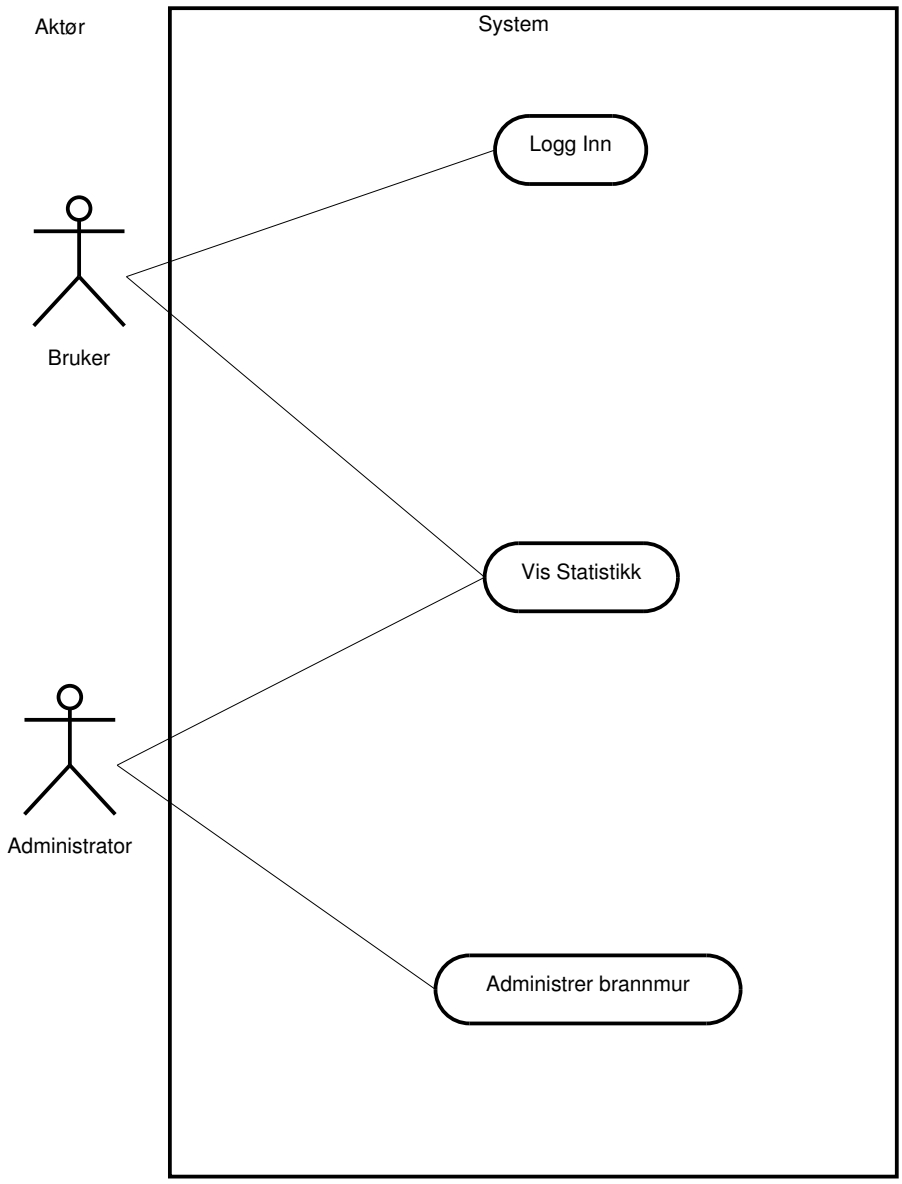
1.4 Oversikt over innhald

Dette dokumentet er av litt meir teknisk art, og inneheld ein modellar for å forklare korleis komponentane i systemet heng saman, og korleis dei kommuniserer med andre komponentar i operativsystemet. Her vil være både UML-standardmodellar, og figurar som bryt noko med UML-standarden. Figurar som bryt litt med standarden vil verte forklart nærmare.

2 Bakgrunn og oversikt

2.1 UseCase modell

UseCase-modellen i figur 1 gir eit bilete av kva funksjonar brukarane av systemet skal ha tilgjeneleg.



Figur 1: UseCase-modell

2.2 Ikkje-funksjonelle krav

Systemet vert utvikla under ein open lisens, og skal være tilgjengeleg for allmennheita. Det vil ikkje følgje nokon garantiar eller krav til service og støtte til systemet.

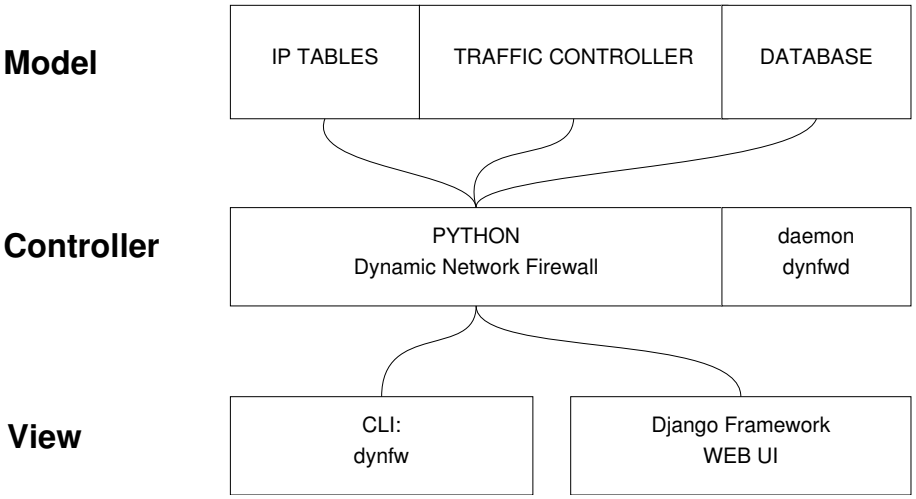
2.3 Vilkår og avhengigheiter

Tjenarside Systemet vert utviklar for Debian Linux, og er avhengig av programmeringsspråket Python, brannmursystemet Iptables og Linux Traffic Control. Alle desse skal være implementert i dei fleste linux-distribusjonar. Vi har også programmert systemet mot ein mySQL-database, men systemet sine grunnfunksjonar kan fungere utan databasen. For authentisering vert linux sitt PAM-system nytta.

Klientside Systemet nyttar webteknologi for å authentisere klientar opp mot systemet. Klientar som skal nytte systemet må derfor ha ein nettlesar. Vi anbefala ein nettlesar av nyare dato.

3 Arkitekturperspektiv

3.1 Logisk

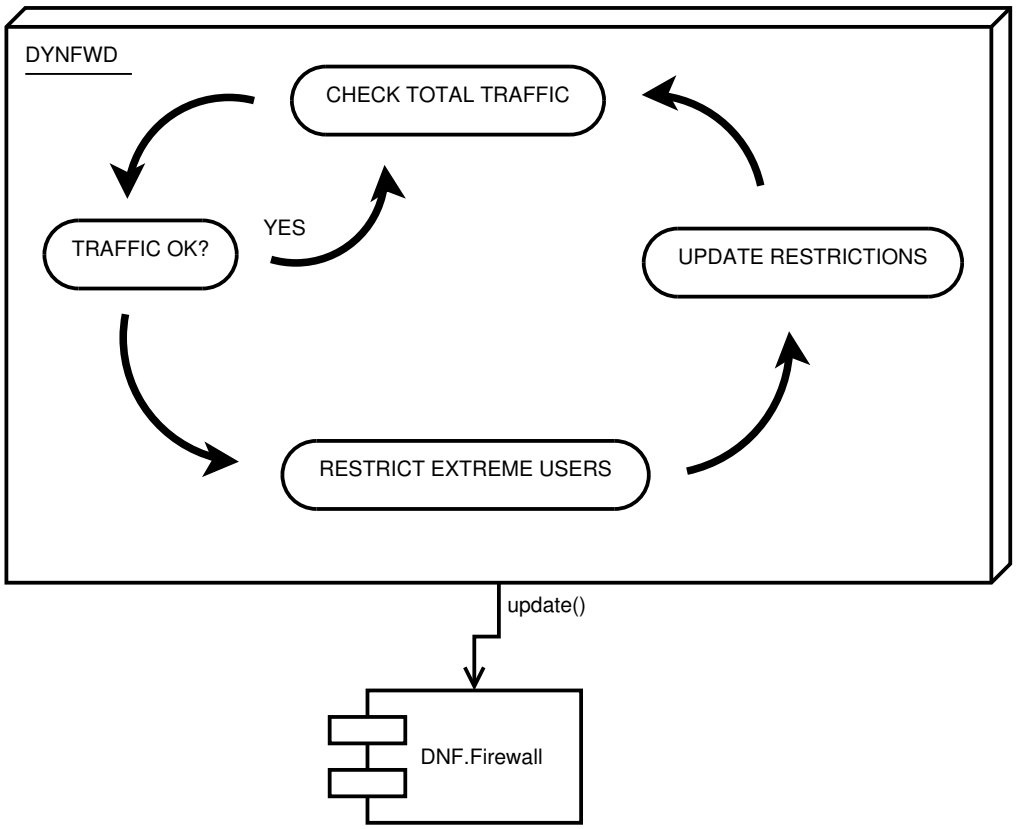


Figur 2: MVC-oppsett av systemet.

3.2 Prosess

3.2.1 Dynfw daemon

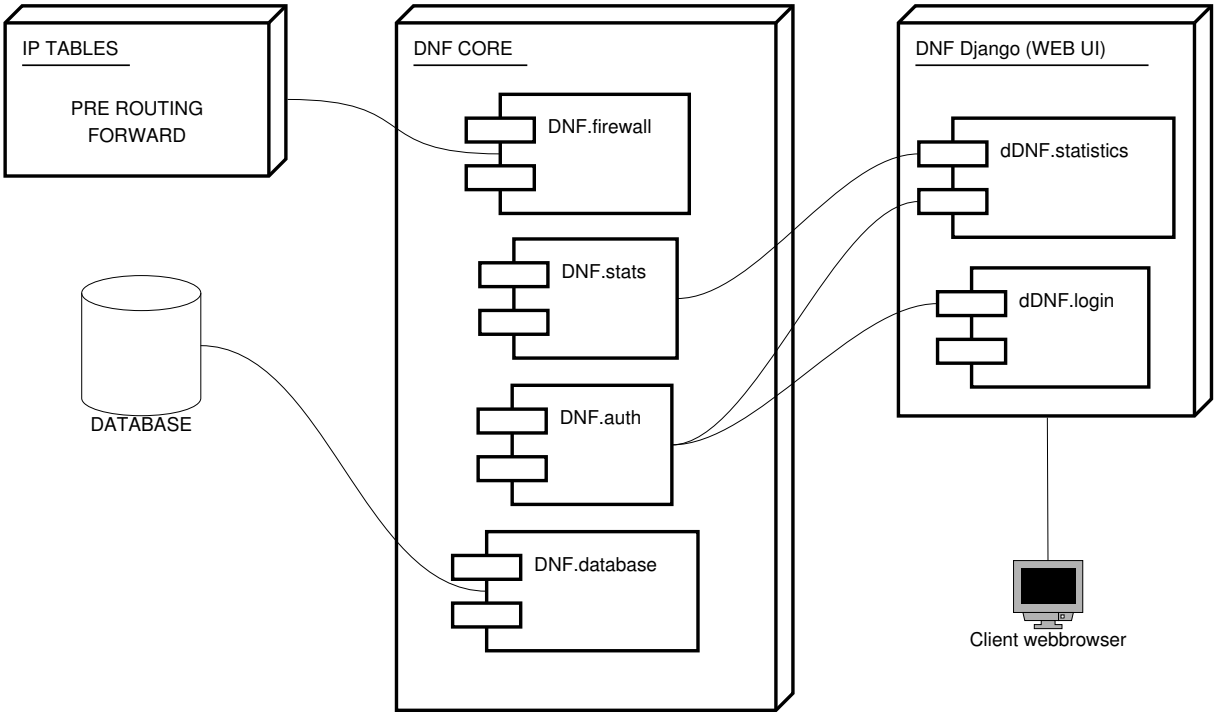
Dynfwd er ein tjeneste (deamon) som kjører i bakgrunnen og automatisk justerer brannmuren og vurderer om nokon av brukarane må avgrensast.



Figur 3: grov skisse over dynfwd si framferd

3.3 Implemetasjon

Vi kan seie at systemet har tre hovuddelar (sjå også figur 2). Figur 4 viser korleis programmvara vi har koda snakkar med dei andre komponentane når nokon nyttar webgrensesnittet. Det gir også ei oversikt over kva pakkar som har ansvar for dei forskjellige linux-komponentane som systemet samarbeider med. Det er verd å merke seg at den automatiske justeringa ikkje kjem til syne i figur 3.

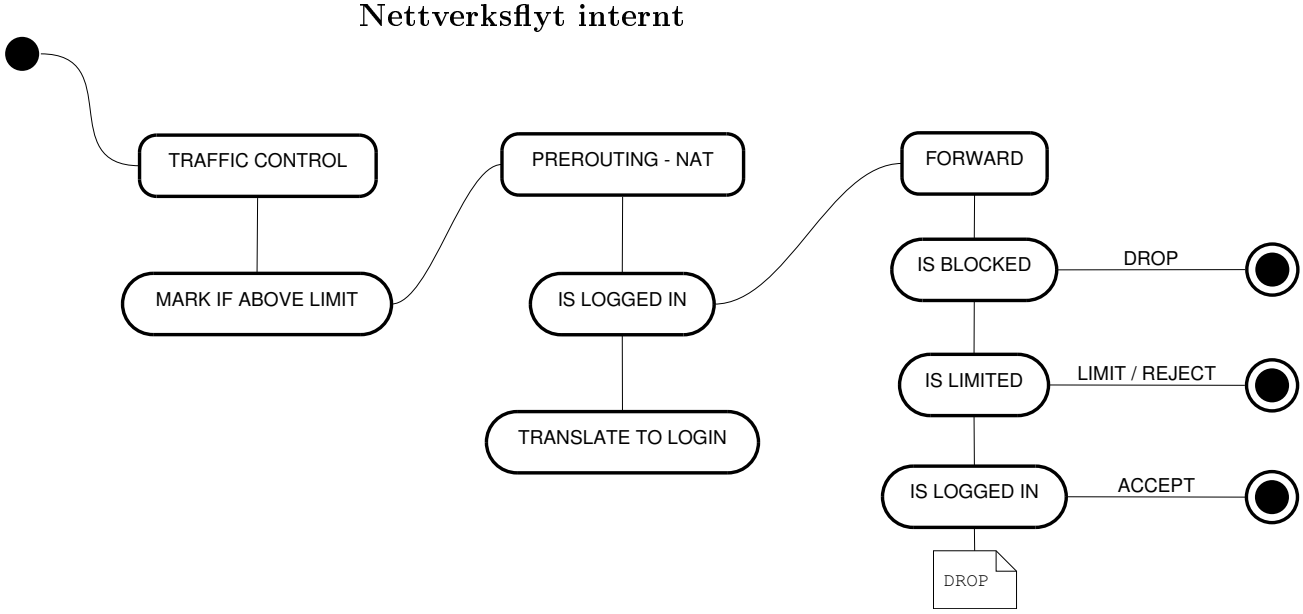


Figur 4: Implementasjonsperspektivet

3.4 Andre perspektiv

3.4.1 Nettverksbehandling

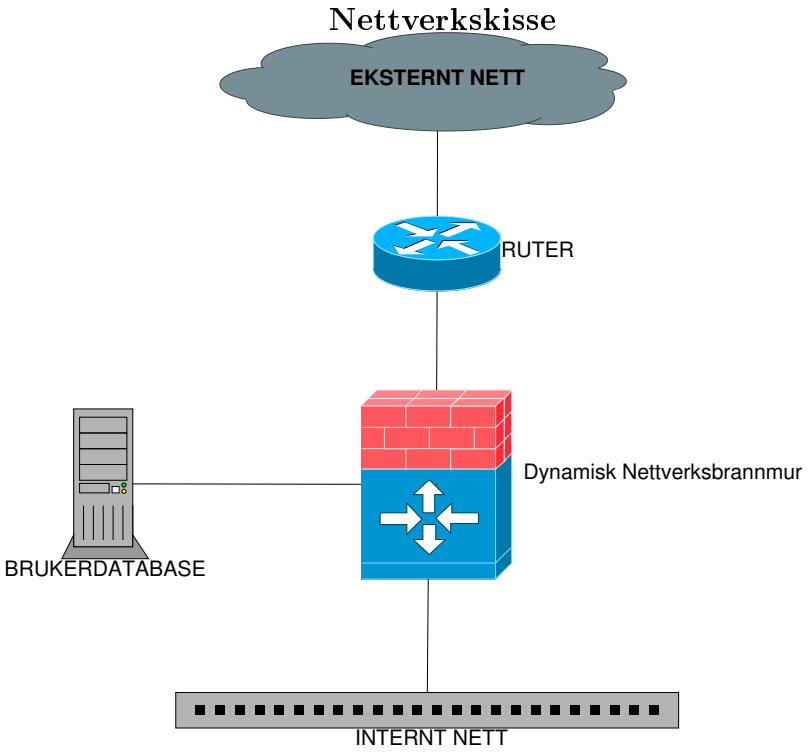
Figur 5 viser korleis trafikken blir behandla internt i systemet. Trafikken går først til linux si mekanisme «Traffic Control», der pakkar som bryt med eit gitt regelverk blir merka. Pakkar går så vidare inn i førruting-regelsettet i Iptables. Her blir det sjekka om ipadressa alt er registrert og logga inn. Om ipadressa ikkje er logga inn, blir trafikken fanga opp og omruta til ei påloggingside. Trafikken blir deretter sendt gjennom ei regelsettet for vidaresending, der det blir sjekka om nokon er svartelista, om trafikken er merka og om personen framleis er pålogga og aktiv.



Figur 5: Skisse over flyt gjennom brannmur

3.4.2 Systemet si plassering i nettverket

Nettverkskissa i figur 6 viser korleis nettverket *kan* settast opp. Det er ikkje naudsynt med ein ekstern brukardatabase, sjølv om det vil være naturleg å ha. Det er også fullt mogleg å ha all rutingfunksjon i same system som dette systemet.



Figur 6: Nettverkskisse

Tillegg E

Prosjekthandbok

Nummereringa på sidetala følgjer no vedlagt dokument.

HER KJEM ET DOKUMENT!

Nummereringa på sidetala følgjer no hovuddokumentet

Tillegg F

Dokumentasjon frå haustprosjektet

Nummereringa på sidetala følgjer no vedlagt dokument.

Høstprosjekt

Svein Ove Undal

1.0 Innholdsfortegnelse

2.0 Innledning

Bakrunnen til oppgava er simpelt den enkle interessa og erfaringa rundt å sette opp og utvikle eit gateway/kaptiveportal system, som har god oversikt over alle brukerane. Det finst mangen former for gateway/brannmur/kaptiveportals, men det finst ikkje system som gjere det på den måten som er gjort i det prosjektet her.

2.1 Oppgåvebeskrivelse

Oppgåva er ein del av eit større bachelor prosjekt, som forprosjekt vil fokuset ligge på å lage eit simpelt gateway/kaptiveportal system, systemet skall ha kontroll på kvar enkelt brukar. IP, mac-addresse og brukernavn blir knytta opp mot kvarandre. Det vil og vere kontroll av duplikater, eller forandringer av ip / mac-addresser. som forhindrer mangen sikkerhets problem som er kjent med liknande system.

Innlogginga, som på alle andre kaptiveportal system skall gå via web loggin. Der før ein bruker blir autentisert vil alle forespørlar på port 443 eller port 80 bli routa til innloggings sida. Når du logger inn, skall det bli oppretta ein brannmur regel til å tillate den klienten på nett.

Begrenser systemet til ipv4, vil og kunn vere mulig å administrere systemet via terminal.

3.0 Teknologi, Arkitektur

Alt er gjort under, linux debian 6. Av utvikling er det for det meste benytta bash scripting, og litt php. Systemet vil forøvrig fungere på alle nye linux distrubusjoner bassert på debian, gjitt att du har rette pakker.

Det var ikkje eit vanskelig valg, det er ingen operativsystem som er like gode på nettverksadministrasjon som linux. Bash scripting er ikkje nødvendig vis det beste, men til den bruken her av mangen små script fungerer det utmerket. PHP er og i små skala eit svært godt verktøy til kjøring av script på server-sia aktivert over web.

3.1 Linux pakker

iptables: er det som kontrollerar all nettverkstraffikk inn og ut. Iptables er eit sett med regler du definerer sjølv, du bestemmer kriteriene på kva typer nettverks pakker den skall sjå etter. så

bestemmer du kva den skall gjere med den pakken. forkaste, acceptere eller endre destinasjon.

apache2: apache2 er i dag den mest brukte og ikkje minst dokumenterte opensource webserveren, i prosjektet er det den serveren som hoster login-sida.

isc-dhcp-server: dhcp serveren som deler ut ip, dns og gateway info til alle klienter. Her er lease tabellen (oversikt over iper som er delt ut) tatt i bruk til å bestemme att dei som faktisk logger på har ein ip ifra dhcp serveren. Vist ikkje, har dei ein statisk ip-addresse som medører att dei blir nekta nett til dei har aktivert dhcp.

ldap-utils: Ein pakke som gjir blant anna ldapsearch, som blir brukt til å autentisere brukere oppimott oppgjitt open-ldap server.

4.0 Systemets virkemåte

Systemet fungerer som dhcp-server, gateway, kaptive portal og brannmur. Når du kopler deg til nettverket, fysisk eller på eit trådløst nett. får du utdelt nettverks informasjon via dhcp-server, isc-dhcp-server. Den gjir deg ein eigen ip, ipen til gateway og dns server.

Brannmuren er konfigurert sånn att den dropper all traffik utenom, dns, http og https (port 53, 80, 443) all traffikk som går på http og https blir redirekta til systemet sin webserver, som er loginsida index.php (10.0.0.1:80). Som betyr att du alltid, uansett kva addresse du skriver i ein nettleser vill du få opp login sida.

Under innloggінga blir det tatt eindel tester. Du kan ikkje ha statisk ip-addresse, dhcp-serveren lager ei oversikt over alle leases som er delt ut, er din klient ikkje i lista vil du bli nekta å logge inn. Den vil og teste om din ip, mac og brukernavn ligger i systemet, isåfall blir du nekta å logge inn. Etter det blir brukernavnet og passordet sjekka, stemmer det vil det bli oppretta ein brannmuregel som tillater din pc sin ip til å kopple til nettet. Det blir og oppretta ei fil med ditt brukernavn i clients mappa, som innehalder din ip, og mac-addresse.

Det er og lagt inn eit liten ekstra ting som gjere det lettare å feilsøke. Det er 3 forskjellige bilder som blir lasta når du skall lese login sida. Vist du ikkje er innlogga vil du få opp ein ipfire-tux, er du innlogga skall du få opp ein world-tux, er du banna får du opp ein shocked-tux. Hensikten er å raskt finne ut av om det er noko gale med clienten, eller serveren.

Når klienten er logga inn er det ein test som blir kjørt på alle klientane, er det ikkje mulig å pinge klienten blir den tvungen til å logge seg inn igjen. Den tester og om det finst duplikater av mac, klienten vil då og få fjærna dens nett tilgang, som tvinger klienten til å logge seg på igjen. Vist alt

går bra vil det bli loggført antall tilkoplinger på nettet klienten har, og kor masse data klienten har lasta opp og ned.

4.1 Scripts

4.1.1 brannmur.sh

Scriptet setter nødvendige brannmuregler, det fjærner alle tidligare regler, tillater å forwarde ipv4. den setter og oppgjit ip til eth1, og restarter dhcp serveren.
brannmur regler:

- tillater 1 til mangen NAT
- redirekter all traffikk på port 80 og 443 til oppgjit ip
- tillater forwarding av traffik på port 443 og port 80
- tillater forwarding av trafikk på port 53
- dropper all anna forwarding traffik

Scriptet må bli kjørt etter restart av server.

4.1.2 accept.sh

Er scriptet som blir kjørt kvar gong nokon logger seg på, scriptet tar 3 variabler. ip, bruker og passord. Scriptet sjekker, om du har ein dhcp-lease, om det er andre klienter med det brukernavnet eller samme ip-addresse. så om du har rett bruker / passord. Om alt stemmer lager scriptet ein brannmur regel for din ip, og kjører scriptet lease.sh. klienten skall no ha full tilgang til internett.

4.1.3 drop.sh

Script som blir kjørt får å slette tilgangen av den gjitte bruker. drop.sh tar 1 eller 2 variabler. ip og ein eventuell grunn. først finner scriptet brukernavnet til tilhøyrandes ip, så fjærner scriptet rekrusivt alle regler i iptables som innehelder ip-addressa. Om scriptet får 2 variabler, vil den i tilleg til å droppe tilgangen til brukaren, og legge ei fil i error mappa med teksten i den andre variabelen.

4.1.4 ping.sh

Script som er meint å kjøre ein gang i blant, som for eksempel kvart 2 minnut. Scriptet sjekker om alle klienter går å pinge, eller om det finst duplikater av mac-addresser. vist det er tilfelle vil klienten bli tvungen til å logge seg inn på nytt igjen. Scriptet driver og logg føring. skriver til ei fil antall connections og kor masse data klienten har sendt og motatt.

4.1.5 findUser.sh

simpelt script som returnerer brukernavnet til den ipen, ved å gå igjøna alle filer i clients til den finner ein ip som stemmer overens med klienten som kopler til.

4.1.6 lease.sh

Scriptet tar to variabler, brukernavn og ip. søker igjøna lease fila for å finne mac-addressa til gjitt ip. scriptet lagrer så ei fil med ip og mac-addrese i client mappa. filnavnet blir brukernavnet.

4.1.7 Index.php

php fila som kontrollerer alt. Ved hjelp av ip-addressa til klienten sjekker scriptet om du allerede har nett, om du er banna. eller om du ikkje endå er logga inn. Det blir demonstrert med 3 forkjellige bilder som blir lasta. Poenget her er å gjere det lettare å feilsøke eventuelle problemer med klienten.

Når knappen på sida blir aktivert vil den prøve å autentisere deg, med å kjøre accept.sh. om alt går bra får du nett. Du vil sjå att bildet på sida forandrer seg til ein “world-tux” og du blir snart redirekta til tihlde.org si side. og har no muligheten til å bruke nettet fritt.

om du allerede var logga inn og trykker på knappen vil den logge deg ut, du ser då att bilde på sida forandrer seg til ein ipfire-tux.

5.0 Konkusjon

Prosjektet har våre ei forprosjekt til ein større bachelor oppgave, hensikten var å sjå om det er mulig å lage eit sånt system på ein effektiv måte. Som det heilt klart er, systemet fungerer svært bra i praksis og på grunn av att det er modulbassert er det lett å vidare legge til funksjonalitet.

Det er selfølgelig litt funksjonalitet eg gjerne skulle ha fått til, der tida ikkje strekte til. Manglar som ein måte å forhindre dns-tunnelering, og automatiske brannmuregler som fordeler båndbredden på brukerane om det blir mangel på kapasitet.

Eg er nøgd med sluttresultatet, det fungerer smertefritt og har stort potensiale i vidare utvikling.

6.0 Installasjon

Ein nyare versjon av debian/ubuntu er nødvendig for å få det her til å fungere, det er og

nødvendig å ha følgandes pakker installert.
ldap-utils, apache2, sudo, dhcp3-server, git

```
sudo apt-get install ldap-utils apache2 dhcp3-server git
```

6.1 dhcp3-server:

endre fila /etc/default/isc-dhcp-server så nederstelinja ser sånn ut

```
INTERFACES=”eth1”
```

(viktig att du har 2 nettverkskort, og att eth0 er til koppla nett. og eth1 er nettverket som du skall gji nett.)

endre fila /etc/dhcp3/dhcpd.conf til følgandes

```
ddns-update-style none;

option domain-name "dittdomene.org";
option domain-name-servers 8.8.8.8, 8.8.4.4;

default-lease-time 600;
max-lease-time 7200;
authoritative;

log-facility local7;


subnet 10.0.0.0 netmask 255.255.255.0 {
range 10.0.0.10 10.0.0.20;
option broadcast-address 10.0.0.255;
option subnet-mask 255.255.255.0;
option routers 10.0.0.1;
}
```

legge til ip til eth1

```
sudo ifconfig eth1 10.0.0.1 netmask 255.255.255.0
```

Restarte dhcp serveren

```
sudo /etc/init.d/isc-dhcp-server restart
```

6.2 Apache2, sudoers

apache-2 trenger i utgangspunktet ingen endring. Men du er nødt til å tillate www-data til å bruke sudo på enkelte script.

Så legg til følgandes nederst i fila /etc/sudoers.d

(Anbefaler å aktivere ssl modulen på apache2, så det er mulig å ha innloggingа via https. Det innebærer å produsere sjølvsignerte sertifikat, for å halde installasjonen simpelt blir det ikkje tatt med her)

```
www-data ALL = (ALL) NOPASSWD: /var/www/accept.sh, /var/www/drop.sh,
/var/www/index.php, /var/www/tv.php, /var/www/findUser.sh
```

6.3 scripta

clone eit prosjektet med git, og legge alt i rett mappe

```
cd /var/www
sudo git clone https://github.com/sveinou/project.git
cd project && sudo mv * /var/www && cd .. && sudo rm -r project
```

6.4 Crontab

legge til ping script som skall kjørast kvart 2 min.

```
sudo crontab -e
```

legg til nederst i fila.

```
*/2 * * * * /var/www/ping.sh
```

6.5 Brannmur / testing

Kjør brannmur fila.

```
sudo ./brannmur.sh
```

Då skall alt fungere! får å teste det trengs det ein switch tilkopla eth1 til server maskina, klienter på den vil då få nett via systemet.

Bruker forøvrig tihlde sin ldap-server, så brukeren testsau med passord 0188 skall fungere.

Nummereringa på sidetala følgjer no hovuddokumentet

Etterord

F.1 Takk!

Helge Hafting for god rettleiing og gode fagkunnskapar.

TIHLDE og spesielt TIHLDE-LAN for at vi fekk nytte dei som testkaninar lenge før systemet var produksjonsklart. Dette gav oss mykje gode erfaringar.

Deltakarane på TIHLDE-LAN som var hyggelege og ga oss verdifulle tilbakemeldingar.