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

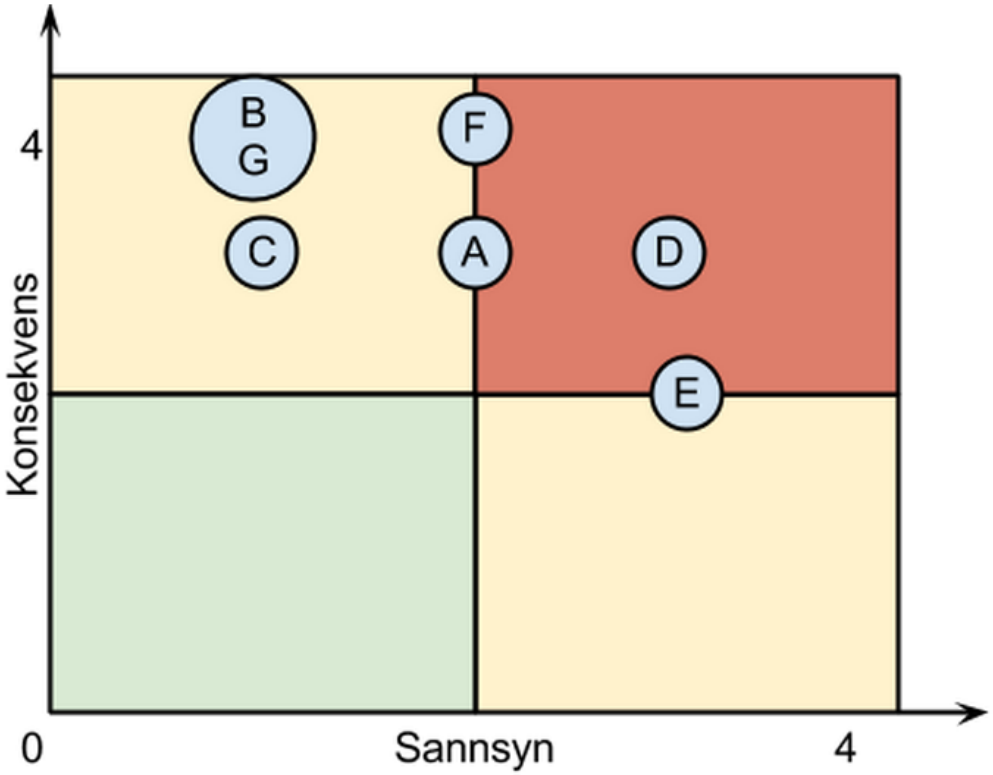
E Programvare oppfører seg ikkje som forutsett

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

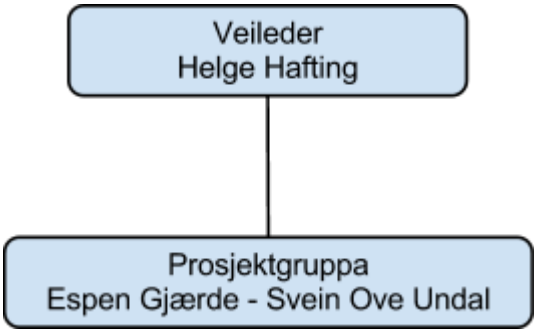
- Systemet skal baserast på kjente og de-facto standardprotokollar som IP, TCP, DHCP og HTTP.
- Språk for administrasjonsystemet skal være norsk - nynorsk.

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.

A ROS-analyse

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.