NTNU
Det skapende universitet

Informasjonssikkerhet i pasientrettede informasjonssystemer basert på åpen kildekode-komponenter

Ørjan Dowerdock Johansen

Master i datateknikk
Oppgaven levert: Juni 2010
Hovedveileder: Dag Svanæs, IDI

Norges teknisk-naturvitenskapelige universitet
Institutt for datateknikk og informasjonsvitenskap

Oppgavetekst

MOBESITY er et forskningsprosjekt ved NTNU i samarbeid med overvektsklinikken ved St. Olavs Hospital i Trondheim. Målet med prosjektet er å utvikle en nettside for egenomsorg blant overvektspasienter. Som del av oppgaven skal det utvikles sikkerhetsmekanismer for å sikre informasjonssikkerheten i nettsiden. Problemstillingen i oppgaven er knyttet til muligheter og begrensninger i integrasjonen mellom åpen kildekode og sikkerhetsmekanismene.

Oppgaven gitt: 15. januar 2010
Hovedveileder: Dag Svanæs, IDI

Sammendrag

Bruken av åpen kildekode i systemutviklingsprosjekter har blitt mer utbredt og utviklingstiden til slike prosjekter er ofte kortere enn prosjekter som ikke benytter seg av ferdigkomponenter. Det er stor uenighet i hvorvidt åpen kildekode er mer eller mindre sikkert enn lukket kildekode, eller om det er noen særlig forskjell i det hele tatt.

I dette prosjektet har jeg sett på hvilke krav som stilles til sikkerhet i systemer som behandler helse- og personopplysninger. Jeg har forsøkt å realisere disse kravene med utstrakt bruk av åpen kildekode-komponenter i utviklingen av et egenomsorgssystem for overvektspasienter. Jeg har sett på hvilke sikkerhetsutfordringer dette fører med seg, og hvilke tiltak som kan brukes for å redusere dem. Gjennom mine erfaringer i forbindelse med utviklingsprosessen i denne studien kan jeg se fordelene ved å bruke fri programvare i motsetning til tradisjonell systemutvikling, og det hadde ikke vært mulig å realisere systemet i samme grad med en tradisjonell systemutviklingsprosess hvor alt blir utviklet fra bunnen av.

De største utfordringene jeg møtte på var knyttet til integreringen av komponentene og utviklingen av en sømløs felles autentisering. For å redusere sikkerhetsutfordringene bør det gjøres en tidlig vurdering angående valg av komponenter og integrering av de valgte komponenter for å kartlegge muligheter og begrensninger som komponentene legger på systemet.

Forord

Denne masteroppgaven er utført i siste semester av sivilingeniørstudiet i datateknikk med studieretning Program og Informasjonssystemer ved Institutt for Datateknikk og Informasjonsvitenskap ved NTNU.

Oppgaven er et ledd i MOBESITY-prosjektet, som er et forskningsprosjekt i samarbeid med NSEP og Obesitasklinikken ved St.Olav, ledet av doktorgradsstipendiat Anita Das. Jeg vil takke Anita for et godt samarbeid, og jeg ønsker henne lykke til videre med prosjektet. Jeg ønsker å takke sikkerhetsansvarlig ved Lånekassen Lillian Røstad og Personvernombud ved St. Olavs Hospital Øyvind Røset som stilte opp til intervju og dermed ga meg en verdifull innsikt.

En stor takk vil jeg gjerne rette til min veileder Dag Svanæs for god veiledning og uvurderlig hjelp gjennom arbeidet med denne oppgaven. Samarbeidet med ham har gjort mitt siste år ved denne institusjonen til et spennende og lærerikt år.

Mest av alt vil jeg takke min beste venn og kjæreste Cecilie Therese Berntsen for hennes støtte og innspill gjennom dette siste semesteret, og spesielt hennes tålmodighet med meg når tidspresset mot slutten meldte sin ankomst. Uten deg ville det ikke vært det samme.

Til sist vil jeg takke mine gode venner og samarbeidspartnere Tarjei Eriksen Ormestøyl og Anders Kløverud Rognstad for fem fantastiske år på haugen, og for alle gode stunder vi har delt dette siste halve året på «læbben».

Trondheim, 24. juni 2010

Ørjan Dowerdock Johansen

Innhold

Tabeller

Figurer

Ordliste

2-faktor autentisering Autentisering med brukernavn og passord i tillegg til engangskode.

Aptitude Pakkehåndteringssystem for Linux.

Autentisering Identifisering av gyldig bruker.

Autorisering Tildeling av rettigheter til bruker.

Bakdør En metode for å unngå normal autentisering.

Binærkode Representasjon av kildekode ved bruk av totallssystemet.

Black box testing Testing basert på input og output.

Brute force Prøve alle mulige kombinasjoner av et passord eller en nøkkel til man finner den riktige.

Buffer overflow En prosess som lagrer mer data i minne enn det den har reservert, og skriver derfor over annen data i minnet.

Bug En feil i programvare.

CMS Content Management System.

Cookie En informasjonskapsel lagret på maskinen til en bruker.

COTS Commercial of the shelf.

DES Data Encryption Standard.

ebXML Electronic Business using eXtensible Markup Language.

FAD Fornyings-, administrasjons- og kirkedepartementet.

FTP File Transfer Protocol.

Hash Obfuskering av for eksempel et passord. Kan ikke reverseres til klartekst.

iFrame Definerer en ramme for visning av en nettside i en annen nettside.

Informert Samtykke Et samtykke hvor den som samtykker er tilstrekkelig informert.

Java Programmeringsspråk.

Journalføres Informasjon som er relevant for en behandlingen til en pasient og som skal knyttes opp mot pasientens logg.

KITH Kompetansesenter for IT i helse- og sosialsektoren AS.

Kryptering Omgjøring av data til uleselig kode.

Linux Åpen kildekode-operativsystem.

Netportal En nettside som gir tilgang til en rekke ressurser.

Patch En oppdatering av koden som retter en feil.

Peer review Kvalifiserte individer reviderer kildekoden til et system for å finne feil.

PhD Doktorgrad.

PHP Programmeringsspråk.

Ping of death Angrep hvor det sendes en for stor «ping» pakke som kræsjer operativsystemet.

PKI Public Key Infrastructure.

Proprietær programvare Systemer hvor bruker ikke har tilgang til kildekoden og kan ikke modifisere eller redistribuere systemet.

RSA Algoritme for kryptering.

Sessions Dialog mellom enheter hvor minst en av partene må lagre informasjon for å kunne kommunisere.

SSL/TLS Secure Sockets Layer/Transport Layer Security.

Stable Stable programvare er en versjon som anses som relativt feilfri med en kvalitet god nok for sluttbrukere.

TIS Trusted Information Systems.

White box testing Testing basert systemets indre struktur.

Åpen kildekode Systemer hvor bruker kan lese, modifisere og redistribuere kildekoden selv.

Kapittel 1

Introduksjon

Dette kapittelet beskriver motivasjonen bak dette prosjektet og konteksten til oppgaven. Jeg vil definere min problemstilling og mine forskningsspørsmål som jeg ønsker å besvare og beskrive avgrensningen til oppgaven. Til slutt vises en oversikt over kapittlene i denne rapporten.

1.1 Motivasjon

Fri programvare har eksistert svært lenge. På 1950 og 60 tallet ble IBM sine operativsystemer distribuert som kildekode og det var vanlig å gjøre modifikasjoner og dele disse med andre [1]. Avgjørelsen om å bruke begrepet åpen kildekode kom i 1998 som en respons på Netscape sin kunngjøring om at de ville utgi kildekoden til Navigator[1].

Idag er åpen kildekode meget utbredt både blant sluttbrukere og kommersielle virksomheter. Bare i Norge er det flere bedrifter som helt eller delvis fokuserer på åpen kildekode som for eksempel Redpill Linpro [2] som leverer produkter og tjenester basert på fri programvare. I Norge er det og opprettet et eget kompetansesenter for fri programvare. Friprogsenteret ble opprettet i 2007 og er et uavhengig, statlig finansiert kompetansesenter for fri programvare og ble opprettet for å bistå i bruk eller utvikling av fri programvare i IT-løsninger [3].

Angående informasjonssikkerhet og fri programvare er det ingen enighet i om fri programvare er mer eller mindre sikkert enn lukket kildekode, eller om det

[1] En populær nettleser på 90-tallet

er noen signifikant forskjell. Stort sett mener tilhengere av åpen kildekode at det er mer sikkert enn lukket, mens tilhengere av lukket kildekode mener det motsatte. Dette vil jeg se nærmere på i Seksjon 2.2

1.2 Kontekst

Bruk av åpen kildekode-komponenter gjør det mulig for et utviklingsprosjekt å dekke flere krav til funksjonalitet med mindre bruk av ressurser. Men sammen med nye muligheter kommer nye utfordringer. For å studere dette nærmere i praksis gikk jeg inn i det eksisterende forskningsprosjektet MOBESITY hvor det skulle utvikles en nettside.

MOBESITY er et forskningsprosjekt ved NTNU i samarbeid med overvektsklinikken ved St. Olavs Hospital i Trondheim. Prosjektet ble startet av Anita Das som hennes PhD-oppgave våren 2009 og skal gjennomføres hos Norsk senter for elektronisk pasientjournal (NSEP) som er et tverrfaglig forskningsmiljø ved NTNU.

Målet med prosjektet var opprinnelig å utvikle en mobil/Internett-applikasjon for egenomsorg blant overvektspasienter. Pasientene skal kunne overvåke egen helse ved for eksempel å observere/registrere inntak av mat og fysisk aktivitet. Helsepersonell skal kunne studere observasjonene pasientene gjør og dermed kunne gi råd og veiledning basert på dette. Det vil også være mulighet for å studere atferdsmønstre blant pasienter basert på denne dataen. Prosjektet har endret noe kurs underveis etter workshopper og diskusjon med pasienter og helsepersonell.

Min oppgave i prosjektet er å utvikle denne nettsiden for pasienter og helsepersonell med utstrakt bruk av åpen kildekode, for å kunne studere hvilke utfordringer knyttet til sikkerhet som eksisterer. Parallelt med mitt prosjekt vil Anders Rognstad og Tarjei Ormestøyl fokusere på brukbarhet ved bruk av fri programvare i sin hovedoppgave, ved å benytte seg av samme systemutviklingscase. Utviklingsprosessen vil være lik for disse to prosjektene, men fokuset underveis vil være spesifikt for de to prosjektene.

1.3 Problemstilling

Hovedmålet med dette prosjektet er å se på

Hvilke krav som stilles til systemer som behandler helse- og personopplysninger og hvilke utfordringer som eksisterer i forbindelse med å implementere kravene i prosjekter med utstrakt bruk av åpen kildekode.

For å besvare problemet vil vi gjennomføre en praktisk studie der vi forsøker å realisere flest mulig krav til et system ved hjelp av åpen kildekode-komponenter.

1.3.1 Forskningsspørsmål

På bakgrunn av problemstillingen har jeg formulert følgende forskningsspørsmål:

FS1 Hvilke krav stilles til informasjonssikkerhet i systemer som behandler helse- og personopplysninger?

FS2 Hvilke utfordringer er knyttet til å realisere disse kravene i prosjekter med utstrakt bruk av åpen kildekode, og hvilke tiltak kan gjøres for å redusere disse?

1.3.2 Avgrensning

For å kunne besvare forskningsspørsmålene vil jeg gjennomføre et systemutviklingscase der jeg utvikler et egenomsorgssystem for overvektspasienter basert på åpen kildekode-komponenter.

Oppgaven vil fokusere på de sikkerhetskravene som må implementeres i systemet, og vil ikke fokusere på sikkerhetskrav i forbindelse med infrastruktur og drifting. Hvis systemet skal testes i en større skala eller settes i drift antar jeg at slike rutiner og krav er tilfredsstilt hos den aktuelle databehandleren, være seg IDI, NSEP eller St. Olavs Hospital.

Oppgaven vil heller ikke være en direkte tolkning av lovverket, men vil basere seg på en litteraturstudie og intervju av eksperter på sikkerhet. Følgelig vil det ikke være noen garanti for at våre krav til sikkerhet fullt ut tilfredsstiller de kravene som er lovpålagt fra det offentlige.

1.4 Rapportutforming

Da dette prosjektet har vært et felles utviklingsprosjekt og en del av forskningsprosjektet MOBESITY, vil en liten del av rapporten være felles med rapporten til Tarjei Eriksen Ormestøyl og Anders Kløverud Rognstad. Dette gjelder hovedsakelig deler av Kapittel 6, som presenterer den endelige løsningen.

Kapittel 1 Introduksjon Dette kapittelet presenterer motivasjonen og konteksten for prosjektet. Problemstilling, forskningsspørsmål og avgrensning blir presentert.

Kapittel 2: Bakgrunn Dette kapittelet presenterer bakgrunnslitteratur som er relevant for å kunne besvare mine forskningsspørsmålene.

Kapittel 3 Forskningsdesign Dette kapittelet presenterer forskningsmetodene som blir benyttet for å besvare forskningsspørsmålene.

Kapittel 4: Intervjuer Dette kapittelet presenterer intervjuene med to sikkerhetseksperter i et ledd for å kartlegge kravene til sikkerhet.

Kapittel 5: Sikkerhetskrav Dette kapittelet presenterer iterasjonene mot den endelige listen med sikkerhetskrav og avsluttes med min endelige kravliste.

Kapittel 6: Endelig løsning Dette kapittelet presenterer den endelige løsningen som vi utviklet i dette prosjektet.

Kapittel 7: Utviklingsprosess Dette kapittelet beskriver utviklingsprosessen, og vektlegger implementeringen av tilfredsstillende sikkerhet i systemet.

Kapittel 8: Resultater Dette kapittelet presenterer mine resultater fra implementeringen av mine sikkerhetskrav og utfordringer jeg støtte på.

Kapittel 9: Diskusjon Dette kapittelet diskuterer jeg kvaliteten av mine resultater i forhold til mine forskningsspørsmål. Jeg ser også på metodene jeg brukte i prosjektet og diskuterer verdien av å bruke disse metodene.

Kapittel 10: Konklusjon Dette kapittelet oppsummerer mine svar på forskningsspørsmålene og problemstillingen samt ser på videre arbeid med dette temaet.

Kapittel 2

Bakgrunn

Dette kapittelet presenterer utgangspunktet for dette prosjektet, og plasserer det i en større kontekst. Relevant teori om sikkerhet og fri programvare vil også bli greid ut om.

2.1 Utgangspunkt

Dette prosjektet er en videreføring fra høstens fordypningsprosjekt, også dette i regi av MOBESITY-prosjektet. Gjennom det prosjektet utarbeidet vi krav og ønsker til en løsning for pasienter ved Obesitasklinikken ved St. Olav. Denne seksjonen vil presentere de relevante resultatene fra høstens prosjekt, som fungerte som grunnlag for utviklingsprosessen.

2.1.1 MOBESITY-prosjektet

MOBESITY er et forskningsprosjekt ved NTNU i samarbeid med overvektsklinikken ved St. Olavs Hospital i Trondheim. Prosjektet ble startet av Anita Das som hennes Phd-oppgave våren 2009 og skal gjennomføres hos Norsk senter for elektronisk pasientjournal (NSEP) som er et tverrfaglig forskningsmiljø ved NTNU.

Målet med prosjektet er å utvikle en mobil/Internett-applikasjon for egenomsorg blant overvektspasienter. Pasientene skal kunne overvåke egen helse ved for eksempel å observere/registrere inntak av mat og fysisk aktivitet. Helsepersonell skal kunne studere observasjonene pasientene gjør og dermed kunne

gi råd og veiledning basert på dette. Det vil også være mulighet for å studere atferdsmønstre blant pasienter basert på denne dataen.

Prosjektet skal forsøke å besvare fire forskningsspørsmål [4]:

- Hva trenger overvektspasienter for å opprettholde en sunn livsstil etter en overvektsoperasjon eller et livsstilskurs? Hvordan kan egenomsorg blant overvektpasienter bli forbedret ved hjelp av elektroniske hjelpemidler (PCer, mobiltelefoner)?
- Hvilke krav har helsepersonell til en mobilapplikasjon for egenomsorg blant overvektspasienter?
- Hvordan skiller pasientene sine krav til en E-helse-applikasjon seg fra helsepersonellets krav? Hvilke design-avgjørelser reflekterer disse forskjellene?
- Hva er resultatene av å bruke en mobilapplikasjon for egenomsorg blant overvektspasienter, sett fra et brukerperspektiv?

2.1.2 Original kravspesifikasjon

Kravspesifikasjonen [4] (versjon 1.0 (sist revidert 29.09.09)) er utarbeidet etter informasjon fra pasienter som har gjennomgått overvektsbehandling og fra helsepersonell som arbeider med denne pasientgruppen. Åtte personer med helsefaglig utdanning (medisin, sykepleie, ernæring) og tolv personer som har gjennomgått overvektsbehandling deltok i tre separate workshoper våren 2009. Seks personer hadde gjennomgått livsstilsbehandling og seks personer hadde gjennomgått kirurgisk behandling [4].

MOBESITY-kravspesifikasjonen ble utarbeidet i samarbeid med pasienter og helsepersonell med et felles mål [4]. Begge gruppene ønsket et system som kan støtte personer som har gjennomgått eller gjennomgår enten kirurgisk behandling eller livsstilskurs. Kravene er et produkt av flere workshoper der både pasienter og helsepersonell har deltatt. Begge parter hadde ønsker til et slikt system, og de ønskene som ble høyest prioritert ble tatt med i kravspesifikasjonen.

Det var ønske om ulike brukerroller i systemet; *Helsepersonell*, *Pasient før behandling*, *Pasient som gjennomgår kirurgisk behandling* og *Pasient som gjennomgår livsstilskurs*. Felles for alle rollene er at systemet skal være sikkert, og følge de lover og forskrifter som er påkrevd for et system som behandler pasientinformasjon.

Oppsummert bestod den funksjonelle kravspesifikasjonen av følgende hovedpunkter:

- Krav om tilgang til informasjonsmateriell og ofte stilte spørsmål og svar.
- Krav om kommunikasjon mellom pasienter og helsepersonell.
- Krav om fadderordning for å gi bedre støtte og oppfølging for nye pasienter.
- Krav om en egen plan der pasientene kan holde oversikt over egne aktiviteter og oppgaver.
- Krav om informasjon om kosthold og måltidsrytme, og mulighet for å få påminnelser om å for eksempel spise.
- Krav om kvalitetssikring av informasjon (utført av helsepersonell).

2.1.3 Workshop med prototype

Basert på kravene i MOBESITY-kravspesifikasjonen [4] ble det sett på ulike måter å realisere kravene på. Vi avholdt derfor en workshop med pasienter og helsepersonell 3. November 2009. Som input til workshopen ble det utarbeidet en tidlig-fase prototype for å kvalitetssikre kravene og få i gang en diskusjon med brukergruppen hvor ønskene til funksjonaliteten kunne bli tilstrekkelig kartlagt.

Under workshopen ble deltakerne delt opp i forskjellige grupper og satt til å prioritere krav til systemet. De ulike gruppene hadde noe ulike prioriteringer, men det var noen punkter som gikk igjen som spesielt viktige. Nettbasert kommunikasjon mellom pasienter og helsepersonell ble sett på som essensielt for et ønsket system. Dette var ønsket av begge brukergrupper; Helsepersonellet følte de ville spare tid på å svare elektronisk, samt at pasient til pasient-kommunikasjon (gjennom for eksempel forum) ville ta av litt av lasten på helsepersonellet. Pasientene så også muligheten til å oppnå flere effekter. De var interessert i et sosialt nettverk der de kunne søke svar på spørsmål og diskutere utfordringer og muligheter knyttet til behandlingsopplegget.

«Min plan» ble også sett på som en viktig del av løsningen. Det var mange som hadde behov for å bli minnet på å spise måltider, trene, ta vitaminer og medisiner og lignende. Vi så en mulighet for at dette kunne realiseres på lik linje som Google har gjort det i Google Calendar som vi brukte som eksempel på workshopen. «Min Plan» kunne erstatte eksisterende individuelle rutiner

å gjøre dette på, slik som mobilpåminnelser, egne kalendere og alarmklokker. I tillegg ville det forhåpentligvis gi en bedre helhet ved å inkludere denne funksjonaliteten i systemet.

2.2 Åpen kildekode og sikkerhet

Tilhengere av åpen kildekode mener at å gjøre koden tilgjengelig for alle fører til mer sikker programvare. Hovedgrunnlaget for dette argumentet summeres fint med «Linus's Law» formulert av Eric S. Raymond [5]:

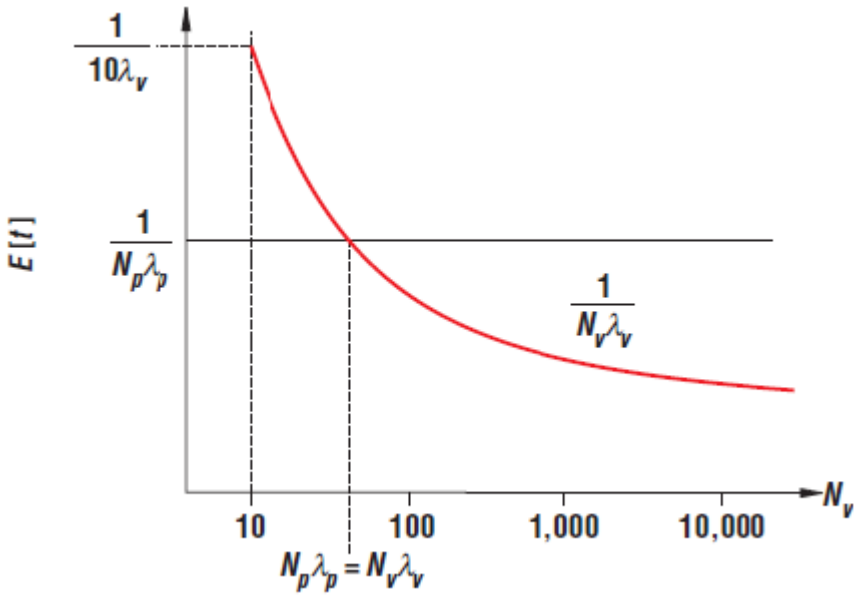
> «To many eyes, all bugs are shallow»

Her vil jeg ta for meg argumentene for hvorfor åpen kildekode fører til økt sikkerhet i systemer.

2.2.1 Peer review

Hovedargumentet bak påstanden om at åpen kildekode er sikrere enn lukket, proprietær kildekode ligger i den antatte styrken til «peer review» prosessen [6, 7, 8]. siden kildekoden er tilgjengelig for alle, så kan alle gjøre sin egen analyse av koden og avdekke sårbarheter, og så rapportere om sårbarheten til utviklerne. Ofte vil personen som finner sårbarheten også legge ved et forslag for å eliminere sårbarheten, som igjen fører til at feilrettingsprosessen går fortere.

Et eksempel på effektiviteten og styrken av denne prosessen er vanskeligheten med å injisere bakdører[1] inn i koden. Når en angriper fikk tilgang til FTP serveren hvor kildekoden til Wietse Venema's TCP Wrapper var lagret ble det injisert en bakdør i koden. Det samme skjedde med Borland/Inprise database software i 1992, som var lukket kildekode. Forskjellen på disse to er at i den siste ble bakdøren satt inn av programmerere som jobbet i selskapet, og bakdøren var ikke ment for å være skadelig selv om den ville vært det hvis angripere fikk vite om den. En annen forskjell er at bakdøren som ble injisert i den åpne kildekoden ble funnet og fikset etter kun én dag, mens bakdøren i den lukkede kildekoden ble funnet og fikset etter 9 år. Dette var rett etter at prosjektet gikk over til å være åpen kildekode [6].



Figur 2.1: Y-aksen viser forventet tid til å finne neste sikkerhetsfeil vs. antall frivillige kontrollører i åpen kildekode miljøet

Selskaper som gir ut lukket, proprietær software gjør selvsagt også sin egen koderevidering, og man kan argumentere for at personene som reviderer her i gjennomsnitt er mer eksperter på det de gjør enn de som reviderer i åpen kildekode-miljøet. Men i det åpne kildekode-miljøet er det potensiale for mange flere personer som kan revidere koden. Brian Witten [8] har satt opp en graf, se Figur 2.1, hvor det illustreres når det åpne kildekode-miljøet finner flere feil per tid enn revisjon i lukket kildekode.

[1] En bakdør er en metode for å unngå normal autentisering

Som man kan se av Figur 2.1 representerer λ_p raten som en betalt kontrollør finner feil og λ_v representerer raten som en frivillig kontrollør finner feil. N_p antas å være et gitt antall betalte kontrollører og N_v representerer antallet frivillige kontrollører. Ut fra dette ser man at forventet tid til man finner neste feil for betalte kontrollører er $\frac{1}{N_p\lambda_p}$ og tilsvarende for frivillige er den $\frac{1}{N_v\lambda_v}$. Dette betyr at forbi punktet når $N_p\lambda_p = N_v\lambda_v$ vil åpen kildekode metoden finne feil raskere.

Denne modellen tar ikke tar hensyn til forandring i effektivitet i forbindelse med gruppesamarbeid eller sentral styring, og heller ikke at raten for å finne feil vil synke etter hvert som det blir færre feil i systemet. Men som Witten [8] også sier så gir grafen en idé om effekten av en større gruppe som reviderer sikkerheten i et system, selv om de ikke nødvendigvis har den samme ekspertisen som den betalte motparten.

2.2.2 Utviklingsprosessen

Selve måten å produsere åpen kildekode legger hindringer i veien for at ondsinnet kode blir lagt inn. Utstrakt bruk av versjonskontroll som kontrollerer og logger alle endringer som gjøres i prosjektet, kombinert med antall øyne som kikker, gjør det vanskelig å legge inn ondsinnet kode som vi så av eksemplet i Kapittel 2.2.1. På spesielt sårbare komponenter som for eksempel Linux kernel er det også vanlig at endringer først analyseres og godkjennes av utviklere med lang fartstid i miljøet før endringen inkorporeres [6].

En annen effekt av at koden er åpent tilgjengelig for alle er at utviklere tvinges til å skrive ryddig og forståelig kode. Ryddig kode er generelt sikrere enn uryddig kode siden det er lettere å overse en feil om koden er veldig vanskelig å lese og forstå.

2.2.3 Egen sikkerhetsrevisjon

Når kildekoden er tilgjengelig kan personen eller firmaet som skal ta i bruk systemet velge å gjøre sin egen sikkerhetsrevisjon av koden, eller leie inn noen for å gjøre det for dem. På denne måten kan brukeren selv vurdere hvor sikkert systemet er og hvor utsatt det er for angrep. Å kunne gjøre sin egen revisjon er med på å lukke avstanden mellom den oppfattede og den faktiske sikkerheten til systemet [7].

Dette er en mulighet man ikke har med lukket kildekode og man må da stole

på at systemet er så sikkert som utgiver hevder at det er. For selskaper med ekstreme og/eller spesifikke krav til sikkerhet vil muligheten for å kunne gjøre sin egen sikkerhetsrevisjon nesten være obligatorisk [6].

For eksempel ville neppe det amerikanske forsvaret tatt i bruk et system utviklet av Kina uten muligheten for først å gjennomgå koden og gjøre en sikkerhetsrevisjon, ikke bare etter feil, men også for å finne eventuelle bakdører og bevisste sikkerhetshull. Det samme gjelder selvsagt motsatt vei og det finnes flere eksempler på dette. I 2001 uttalte Tyskland at de beveget seg vekk fra Microsoft som operativsystem på maskiner som behandlet sensitive opplysninger grunnet mistanke om at NSA hadde tilgang på bakdører i Microsoft sin kildekode [9].

Også Kina har delt disse bekymringene og sjef for Red Flag Linux, Sun Yufang, uttalte at

> «Vi er hovedsakelig bekymret for at utenlandsk programvare, inklusive Microsoft, inneholder bakdører». [10]
>
> — Sun Yufang

2.2.4 Modifisering

Med åpen kildekode har ikke brukere bare tilgang til å lese og revidere kildekoden som nevnt i Seksjon 2.2.3, men også lov til å modifisere og redistribuere koden så lenge man overholder vilkårene gitt i lisensen som er brukt. Vilkårene det er snakk om er ofte om den modifiserte koden må utgis med samme lisens, om programmet kan linkes til annen proprietær programvare og så videre. Eksempler på vanlige lisenser er GNU General Public License [11] og Eclipse Public License [12].

Ved å ha denne friheten står man fritt til å modifisere systemet etter egne behov og tilpasse det mot andre systemer det skal interaktere med. En annen stor fordel med å kunne modifisere systemet er for å kunne legge inn ekstra sikkerhetstiltak for å tilfredsstille ekstra høye sikkerhetskrav man eventuelt har. Ved gjennomføring av en sikkerhetsrevisjon, Seksjon 2.2.3, kan det avdekkes mangler ved sikkerheten i systemet og man har da mulighet til å legge inn egne sikkerhetstiltak man føler mangler, eller er mangelfulle, i systemet.

2.2.5 Tid som brukes på å fikse sårbarheter

Tiden som brukes på å fikse sårbarheter i systemer har mye å si for det potensielle skadeomfanget til en sårbarhet. Ifølge Witten [8] er det beviser for at sårbarheter i åpen kildekode fikses dobbelt så fort som sårbarheter i lukket kildekode, og har dermed et mye mindre vindu for potensielle angrep. En bidragende faktor til dette kan være som nevnt i Seksjon 2.2.4 at brukere selv kan modifisere og også utgi fikser i form av en «patch».

Brukere av lukket kildekode har kun tilgang til binærkoden av systemet og er derfor nødt å vente på at utgiveren skal lage og gi ut en oppdatering som fikser sårbarheten. Det er flere grunner til at oppdateringer for sårbarheter i kommersielle lukkede systemer kan ta lenger tid. En årsak kan være at utgivelse av oppdateringer kan styres mer av økonomiske motiv enn av tekniske, siden offisiell anerkjennelse av en sårbarhet kan påvirke aksjene negativt [6]. Utgivere kan også måtte informere sine store kunder om at problemet blir offentliggjort og dette fører til at «vanlige» kunder er sårbare lenger[6].

I noen tilfeller kan også utgiveren bestemme seg for at det ikke er verdt å fikse et problem. I slike tilfeller har brukere kun to valg, aksepterer at feilen ikke blir fikset eller slutte å bruke systemet.

Et godt eksempel på at det å fikse en sårbarhet tar tid er illustrert med den kjente «Ping Of Death» sårbarheten som påvirket en mengde operativsystemer, både åpen og lukket kildekode, ved å sende en for stor pingpakke som igjen forårsaket en «buffer overflow». En «patch» for Linux ble gitt ut noen timer etter sårbarheten var oppdaget, mens brukere av kommersielle, lukkede systemer måtte vente mye lenger før en fiks ble gitt ut [13].

En mer kontroversiell, kanskje konspiratorisk, forskjell på åpen og lukket kildekode er ifølge Anderson at amerikanske myndigheter foretrekker at sårbarheter i visse systemer rapporteres til dem først slik at de kan utnyttes av politi og sikkerhetstjenester en stund. Utgivere oppfordres til å fikse sårbarheten når andre oppdager og begynner å utnytte feilen [14]

2.2.6 Utvikling er ikke drevet av kommersielle krefter

Det er en fundamental forskjell på utvikling av åpen kildekode kontra lukket kildekode. Åpen kildekode kan utvikles i henhold til rene tekniske krav, mens lukket kildekode ofte har press på seg for å få produktet ut på markedet, enten for å tjene penger på produktet og/eller for å få en fordel over konkurrenter. I en slik setting blir ofte krav som brukere stiller til systemet prioritert og

sikkerhet, som ikke er en «funksjon» som brukere benytter, kan derfor bli nedprioritert. Siden åpen kildekode ikke er styrt av markedet står utviklere fritt til å implementere det som trengs, ergo hvis sikkerhet er nødvendig så vil det implementeres [6].

2.2.7 Vanskelig å finne sårbarheter

Mens tilhengere av åpen kildekode mener at peer review, beskrevet i Seksjon 2.2.1, fører til at flere sårbarheter blir funnet og fikset og som en konsekvens av dette at systemer blir sikrere, mener tilhengere av lukket kildekode at hvis en bug aldri oppdages, så er det ikke et sikkerhetsproblem og derfor er det det samme som at den ikke eksisterer [6]. Og fordi sikkerhetshull er lettere å finne hvis man har åpen tilgang til kildekoden så vil det å lukke kildekoden for innsyn føre til at færre sårbarheter blir funnet. Og siden sårbarheter som ikke blir funnet ikke utgjør noen trussel så vil systemet være sikkert.

Ved å åpne kildekoden gir man eventuelle angripere en stor fordel til sammenligning med utviklerne som skal beskytte mot angrep. En angriper trenger kun å finne en sårbarhet og utnytte denne for å kunne gjøre skade på systemet i en eller annen form. Den som beskytter derimot, må avdekke og fikse alle sårbarheter for å kunne beskytte seg mot angriperne.

2.2.8 Effektiviteten av peer review

Argumentasjonen som både tilhengere av lukket og åpen kildekode bruker for å fremheve at det ene er bedre en det andre er i bunn og grunn svært lik. Begge sider hevder at ved å åpne kildekoden til systemer så vil man finne flere sårbarheter. Forskjellen i synspunkter ligger i om dette er en bra eller dårlig ting.

Nylig sluppet åpen kildekode får ofte tillit med en gang og blir raskt tatt i bruk, ofte før noen har rukket å teste systemet ordentlig. Dette er i sterk kontrast til for eksempel nye publiserte vitenskapelige resultater, som utsettes for grundig analyse og kritikk [6]. Peer review prosessen er ikke særlig effektiv før nok kvalifiserte brukere har tatt den første «risikoen» og tatt i bruk systemet og testet det. Tilhengerne av lukket kildekode hevder de er mer kvalifiserte til å gjøre testing og revidering før systemet lanseres enn det åpen kildekode-miljøet er.

Et annet argument mot effektiviteten til peer review er at selv om kildekoden

er tilgjengelig for alle er det ikke sikkert alle ser på den. Og hvis det kun er noen få som faktisk kikker på kildekoden mens resten bare tar i bruk systemet og antar at andre reviderer kildekoden, så er det begrenset hvor effektiv peer review prosessen er. Og av de som faktisk kikker på kildekoden vil det være varierende grad av kvalifikasjoner for å finne sårbarheter. Sikkerhetsselskapet TIS[2] gjorde kildekoden til sin brannmur tilgjengelig for kundene slik at de selv kunne revidere koden og rapportere feil i koden. Det de så var at svært få personer som ikke jobbet i TIS sendte inn rapporter [15].

Dersom størstedelen av brukerne til åpen kildekode kun tar i bruk systemer og overlater revidering av koden til andre, så vil ikke denne prosessen være særlig effektiv. I senere tid har det også blitt mer og mer populært å distribuere åpen kildekode som binærkode, mens kildekoden enten følger med eller må lastes ned separat. Dette er en faktor som ytterligere bidrar til at færre faktisk leser og reviderer kildekoden.

2.2.9 Forholdet mellom kildekoden og binærkoden

Som nevnt i Kapittel 2.2.8 er det blitt mer og mer vanlig at åpen kildekode distribueres i binærform og at kildekoden er et valgfritt vedlegg. Før var det vanligere at brukere som installerte åpen kildekode-systemer selv kompilerte koden og installerte systemet. Når man laster ned og installerer binærkoden har man ingen garantier for at kildekoden man leser og reviderer faktisk tilsvarer det som er innholdet i binærkoden [15]. En angriper kan kompilere systemet med for eksempel en bakdør innlagt, mens kildekoden som ligger ved ikke inneholder denne bakdøren. Hvis man da installerer binærkoden vil man ha en bakdør installert på maskinen sin, uansett hvor mange ganger du har revidert kildekoden.

Men allerede i 1983 viste Ken Thompson et angrep for å injisere en bakdør i systemer som fremdeles er relevant idag og hvor det ikke var noen spor i kildekoden [16]. Han modifiserte C kompilatoren i UNIX slik at den kjente igjen når login programmet ble kompilert og satte inn en bakdør slik at han kunne logge inn som hvilken som helst bruker med et valgt passord. Dette var selvsagt ikke et angrep som var spesielt vanskelig å finne, siden hvem som helst kunne lese kildekoden og se koden som la inn bakdøren. Men for å ta angrepet et steg videre modifiserte Thompson kompilatoren slik at når kompilatoren kompilerte seg selv så la den inn både koden for å injisere bakdøren og koden for å modifisere kompilatoren. Med denne endringen kunne han

[2]Trusted Information Systems

fjerne modifiseringene fra kildekoden og hadde dermed en trojansk kompilator som la inn en bakdør i innloggingssystemet og som «smittet» andre kompilatorer den kompilerte og det var ingen spor av dette i kildekoden.

2.3 Sikkerhetsnivåer i offentlig sektor

Som et ledd i å definere sikkerheten i offentlige tjenester har FAD utviklet et rammeverk for autentisering og uavviselighet. Rammeverket definerer fire nivåer av sikkerhet hvor nivå fire er best, og gir et sett med teknologinøytrale krav som må oppfylles for at en løsning skal kvalifisere for et gitt nivå. Kravnene, hentet fra FAD [17], som stilles til de forskjellige nivåene er:

Krav til autentiseringsfaktor(er):
Beskriver kravet til antall autentiseringsfaktorer og om faktoren skal være statisk eller dynamisk. Statiske autentiseringsfaktorer er data som ikke endres fra gang til gang, for eksempel faste passord på et kodeark eller biometriske data. Dynamiske autentiseringsfaktorer er data som endres fra gang til gang, for eksempel passordkalkulator eller engangskode på SMS.

Utlevering til bruker:
Beskriver krav om hvordan en bruker kan opprette en konto. For eksempel om brukeren må møte opp i person og legitimere seg, om brukeren kan få tilsendt informasjon i posten på folkeregistrert adresse eller om brukeren kan opprette konto over Internett.

Sikring av autentiseringsfaktor ved lagring:
Dette kravet beskriver hvordan autentiseringsfaktorer skal lagres lokalt og hvordan de skal sikres. Med dette menes om faktorene er kopierbare eller ei. For eksempel dersom en passordliste er skrevet på et ark, så er de kopierbare, men hvis passordene er beskyttet som et skrapelodd vil de ikke være kopierbare uten at brukeren oppdager det.

Krav til offentlig godkjenning:
Beskriver om det finnes en offentlig kravspesifikasjon for denne løsningen og at løsningen er deklarert ved en offentlig ordning.

Krav til uavviselighet:
Dette kravet sier noe om hvilken grad det skal være mulig å dokumentere at en bruker står bak en utført handling.

Nedenfor følger en oversikt over kravene som stilles for å kvalifisere til de fire sikkerhetsnivåene hentet fra FAD [17]. Kravet til systemer som behandler helse- og personopplysninger, slik som vårt prosjekt potensielt vil, er sikkerhetsnivå 4. I skrivende stund er det derimot ingen systemer som fult ut støtter sikkerhetsnivå 4, men det jobbes med å få på plass slike løsninger.

- **Nivå 1**

  Krav til autentiseringsfaktor(er)
  Ingen krav.

  Utlevering til fysiske personer
  Ingen krav.

  Utlevering til juridiske personer
  Ingen krav.

  Sikring av autentiseringsfaktorer ved lagring
  Ingen krav.

  Krav til offentlig godkjenning
  Ingen krav.

  Krav til uavisslighet
  Ingen krav.

- **Nivå 2**

  Krav til autentiseringsfaktor(er)
  Enfaktor

  Utlevering til fysiske personer
  Post til folkeregistrert adresse.

  Utlevering til juridiske personer
  Post til enhetsregisterets registrerte adresse. Navnet til den fysiske personen som kan tegne for den juridiske personen, skal stå først på forsendelsen. Alternativt kan det sendes til tegnerens folkeregistrerte adresse.

  Sikring av autentiseringsfaktorer ved lagring
  Både statiske og dynamiske kan være kopierbare.

  Krav til offentlig godkjenning
  Ingen krav.

  Krav til uavisslighet
  Det skal foreligge rutiner og logger, som gjør at det er rimelig

sikkert at kommunikasjonsparten står bak en handling eller et informasjonselement.

- **Nivå 3**

  Krav til autentiseringsfaktor(er)
  Tofaktor, hvorav en er dynamisk

  Utlevering til fysiske personer
  Samme krav som i 2, men med ett tilleggskrav om at utsendelsesprosedyren skal ha integrert tilleggssikring som sørger for at sannsynligheten for at feil person tar løsningen i bruk minimaliseres. Det er ikke krav om personlig oppmøte.

  Utlevering til juridiske personer
  Samme krav som i 2, men med ett tilleggskrav om at utsendelsesprosedyren skal ha integrert tilleggssikring som sørger for at sannsynligheten for at feil person tar løsningen i bruk minimaliseres. Det er ikke krav om personlig oppmøte.

  Sikring av autentiseringsfaktorer ved lagring
  Dynamiske kan være kopierbare. Statiske kan ikke være kopierbare.

  Krav til offentlig godkjenning
  Ingen krav.

  Krav til uavisslighet
  Det skal foreligge rutiner og logger, som gjør at det er rimelig sikkert at kommunikasjonsparten står bak en handling eller et informasjonselement.

- **Nivå 4**

  Krav til autentiseringsfaktor(er)
  Tofaktor, hvorav en er dynamisk

  Utlevering til fysiske personer
  Kravene til registrering og utleveringsprosedyrer er tilsvarende kravspesifikasjon for PKI , Person Høyt. Personlig oppmøte med legitimering, minst en gang.

  Utlevering til juridiske personer
  For juridiske personer skal den fysiske personen som tegner den juridiske, enten møte opp personlig, eller gi fullmakt til en annen som kan møte personlig på personens vegne. Det skal fremlegges

legitimasjon for begge, samt sjekkes mot enhetsregisteret. Krav tilsvarende Kravspesifikasjon for PKI nivå Virksomhet

Sikring av autentiseringsfaktorer ved lagring
Ikke-kopierbare.

Krav til offentlig godkjenning
Løsningen skal være deklarert i henhold til offentlige krav.

Krav til uavisslighet
En kommunikasjonspart skal kunne verifisere at den andre part står bak en handling eller et informasjonselement. Den skal ikke selv kunne produsere eller endre på et slikt bevis i etterkant.

2.4 Sikkerhetskrav i helsesektoren

Informasjonssikkerhet er definert av Datatilsynet [18] til å omfatte:

- Sikring av konfidensialitet; forhindre at uvedkommende får adgang til opplysninger.
- Sikring av integritet; beskyttelse av utilsiktet endring av data.
- Sikring av tilgjengelighet; sikre at data er tilgjengelig når og der det skal være.

I dette kapittelet skal jeg se på noen av de sikkerhetskravene som kan være relevante for et system som MOBESITY.

2.4.1 Meldingsoverføring

Man kan ikke bruke vanlig e-post for utveksling av helse og personopplysninger fordi disse opplysningene må kunne journalføres og informasjonen må kunne sikres av systemer for tilgangsstyring, samt at e-post vanligvis ikke er kryptert. Det må implementeres strukturerte meldinger for kommunikasjon av helse- og personopplysninger til pasienter og mellom helsepersonell [19]. Det stilles krav fra det offentlige om at alle overføringer av slike opplysninger skal implementere følgende:

- Sikre at overføringen er kryptert, enten ved bruk av SSL/TLS eller PKI.

- Kan kun sende til forhåndsdefinerte mottagere som er lagret i en katalogtjeneste.
- Må ha beskyttelse mot virus og uønsket post (spam)

Bruk av vanlig e-post kan kun brukes for overføring av opplysninger og informasjon som ikke innebærer personsensitive opplysninger. Det stilles krav til at en vanlig e-post-løsning ikke må eksponere interne systemer og helse- og personopplysninger. Det er derfor viktig å ha klare retningslinjer for hvilke systemer som kan brukes til hva, i tillegg til for eksempel å forhindre at man kan klippe og lime sensitive opplysninger fra en applikasjon til en annen [20].

2.4.1.1 ebXML

Electronic business XML er en internasjonal standard (ISO 15000) [21] som definerer et XML rammeverk for utveksling av informasjon på en trygg og konsistent måte uavhengig av plattform. Denne standarden erstatter det tidligere systemet for meldingsutveksling i Helse-Norge kalt Trygd-helsepostkassen som ble annonsert avviklet 1. juni 2009 [22]. ebXML er bygd opp i moduler og og er et omfattende rammeverk, men det er bare delen for trygg meldingsutveksling, ebXML Messaging Service, som benyttes [23].

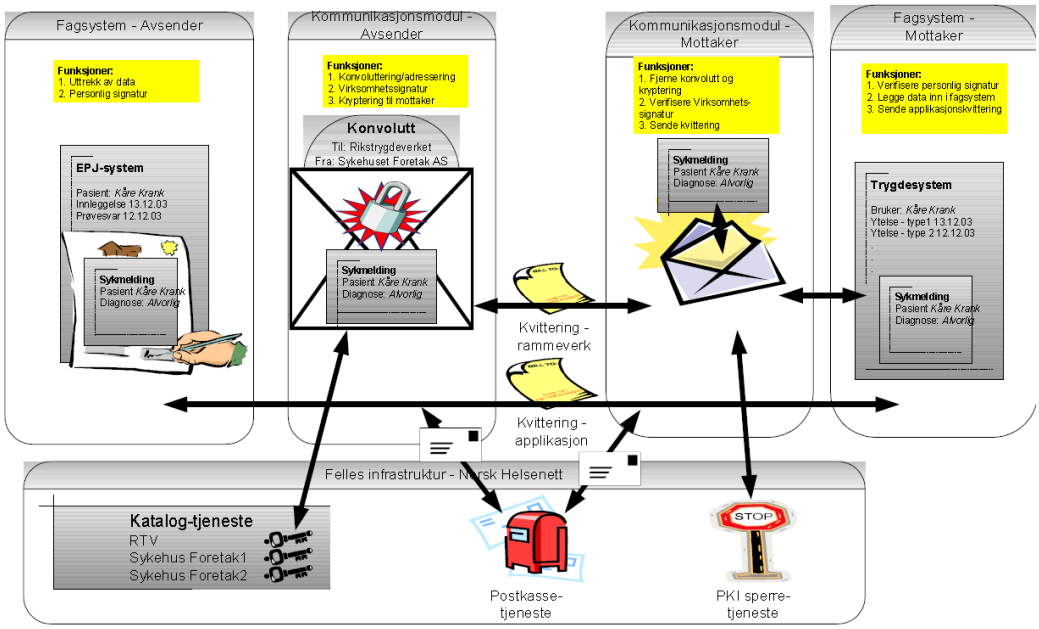
Delen av ebXML som innføres kan ses på som «konvolutten» til meldinger som sendes. Dette betyr at meldingene kan beholdes som før, men sendes i en standardisert konvolutt; ebXML.

2.4.1.2 PKI

PKI står for Public Key Infrastructure og er en løsning for å sikre kommunikasjon med kryptering, autentisering og signatur. Et nøkkelpar dannes ved bruk av samme algoritme, som for eksempel RSA[3]. Det vil da dannes en privat nøkkel som kun skal gjøres tilgjengelig for virksomheten eller personen som eier nøkkelen og en offentlig nøkkel som skal være tilgjengelig for alle.

Et slikt nøkkelpar fungerer slik at en melding krypteres med mottagerens offentlige nøkkel. For å dekryptere meldingen er man da nødt til å benytte den korresponderende private nøkkelen som kun er tilgjengelig for personen meldingen er ment for. For å signere en melding krypterer senderen sertifikatet med sin private nøkkel som da kan dekrypteres av hans offentlige nøkkel

[3] http://en.wikipedia.org/wiki/Rsa



Figur 2.2: Informasjonsflyt ved ebXML og PKI.

for å verifisere avsender. Siden avsender er den eneste som har tilgang til den private nøkkelen kan man dermed med sikkerhet autentisere avsender.

2.4.1.3 Norsk Helsenett

Norsk Helsenett[4] er et lukket nettverk som benyttes for elektronisk kommunikasjon i helse- og sosialsektoren i Norge. De tilbyr postkassetjeneste for meldingskommunikasjon. De tilbyr også adressebok og PKI-katalog hvor offentlige nøkler kan lagres slik at de er lette å få tak i når meldinger skal krypteres og sendes til mottager.

2.4.1.4 Bruk av ebXML og PKI

Ved å bruke teknologiene i Seksjon 2.4.1.1, 2.4.1.2 og 2.4.1.3 vil man sikre en standardisert plattform for sikker utveksling av meldinger som kan håndtere alle typer meldinger og vedlegg på en ensartet måte. Hvordan meldingsflyten vil fungere er godt illustrert i Figur 2.2 som er hentet fra KITH sin veiledning for innføring av ebXML og PKI [23].

[4] `http://www.norsk-helsenett.no`

Som man kan se av Figur 2.2 overføres informasjonen til ebXML-kommunikasjonsmodulen. Denne modulen bruker Norsk Helsenett for å hente ned adresseinformasjonen til mottager og signere meldingen med virksomhetens private nøkkel. Modulen henter så ned mottagerens offentlige nøkkel og krypterer konvolutten med denne. Konvolutten blir overført til riktig postkasse, for eksempel en SMTP/POP - postkasse hos Norsk Helsenett. Mottagerens ebXML kommunikasjonsmodul henter ned konvolutten og bruker deres private nøkkel for å dekryptere konvolutten. Etter at den er blitt dekryptert brukes senderens offentlige nøkkel for å dekryptere signaturen og den verifiseres opp mot en PKI sperretjeneste. Meldingen overføres til systemet og en kvittering på at melding er mottatt sendes til avsender. Hvis ikke avsender mottar kvittering for mottagelse etter rimelig tid sendes meldingen på nytt for å sikre at alle meldinger kommer frem.

2.4.2 Kryptering

All kommunikasjon over Internett er sårbar for avlytting, og spesielt kommunikasjon over trådløse nett hvor en angriper bare kan snappe opp kommunikasjonen som går over det trådløse nettverket ved å være i rekkevidde av trådløssignalene. For å beskytte sensitive opplysninger som sendes over internett er det viktig at informasjonen krypteres slik at kun den tiltenkte mottager kan lese informasjonen.

Hvor sterk kryptering man bruker for å sikre kommunikasjon og data er en viktig faktor for hvor sikkert det faktisk er. Tidligere var det vanlig å bruke DES56 eller tilsvarende kryptering, men grunnet utviklingen av maskinkraft kan kryptering med 56 bits nøkkelrom «brute-forces» innen rimelig tid. Datatilsynet anbefaler derfor bruk av DES128 kryptering eller tilsvarende for tilstrekkelig sikkerhet [24].

Antall operasjoner som trengs for å «brute-force» en kryptering stiger eksponentielt med størrelsen på nøkkelrommet. Illustrert betyr dette at for å knekke en 128 bits kryptering må alle 2^{128} mulige kombinasjoner sjekkes. Hvis man kunne sjekke en trillion (10^{18}) nøkler i sekundet ville det fremdeles ta 10^{13} år før man hadde prøvd alle nøklene. Dette er ca 1000 ganger lenger enn alderen til universet[5] [25].

[5]Dette er en teoretisk antagelse og under antagelsen om at hele nøkkelrommet benyttes.

2.4.3 Autorisering og autentisering av brukere

Autentisering er prosessen hvor brukeren identifiserer seg som en gyldig bruker av systemet. Autorisering er å gi en bruker rettigheter til å bruke bestemte ressurser, det være seg filer, prosesser eller applikasjoner.

Det stilles forskjellige krav til autentiseringsstyrke som avhenger av bruksområdet. For autentisering i interne nett er minimumskravet pålogging med brukernavn og passord [26]. Det anbefales at det stilles krav til passordstyrke, som minimum 8 tegn og bruk av store og små bokstaver. For pålogging fra eksterne nett som hjemmekontor eller en mobil enhet som gir tilgang til sensitive opplysninger stilles det krav om 2-faktor autentisering [26]. En 2-faktor autentiseringsmekanisme betyr at man først må logge inn med brukernavn og passord, og så taste inn en engangskode fra et kodeark, kodekalkulator eller en tilsendt sms.

Vestad og Alsaker [27] trekker også frem viktigheten ved å sikre hele autentiseringskjeden. Generelt sett vil en bruker måtte registreres og identifisere seg med for eksempel identitetspapirer, og så få tildelt brukernavn og passord og dermed mulighet for å logge seg inn. Alle ledd i denne prosessen må sikres, siden et kjempesikkert passord ikke vil være til mye nytte hvis hvem som helst kan be om en konto og få en.

> «En kjede er ikke sterkere enn sitt svakeste ledd»
>
> — Vestad og Alsaker [27]

I følge Huston [28] er det feil å forestille seg at alle angrep mot informasjonssystemer utføres av eksterne. Det er ofte disse angrepene man får høre om, men størsteparten av sikkerhetsbrudd utføres internt av medarbeidere. Disse sikkerhetsbruddene kan variere fra utilsiktet utlevering av sensitiv informasjon til bruk av sensitiv informasjon for hevn eller profitt. Det er derfor viktig at det kun gis tilgang til nødvendig informasjon slik at behandling kan utføres. Normen for informasjonssikkerhet i helsesektoren [26] nevner to metoder for tilgangskontroll:

1. Rollebasert tilgang:
   Alle ansatte er gruppert etter klart definerte roller, og hvilken rolle man har har betydning for hvilken informasjon man har tilgang til.
2. Beslutningsbasert tilgang:
   I beslutningsbasert tilgang må det foreligge en beslutning for gjennomføring av en behandling før tilgang til informasjon angående den spesifikke sak gis.

2.4.4 Bruk av ekstern databehandler

Virksomheter, eller som her forskningsprosjekter, kan ha behov for å benytte seg av utenforstående miljøer i forbindelse med behandling eller drifting av personsensitiv data. Det stilles strenge krav til sikkerhet for eksterne databehandlere og virksomheten som er ansvarlig for dataen [29].

Det må utformes en databehandleravtale mellom partene hvor den eksterne databehandleren forplikter seg til kun å behandle dataene slik som det er avtalt og å tilfredsstille kravene stilt i Norm for informasjonssikkerhet i helsesektoren [30]. Det stilles også krav til at databehandleransvarlig i virksomheten skal ha innsyn i prosedyrer og praksis som følges for informasjonssikkerhet hos den eksterne databehandleren.

For en full oversikt over krav som stilles til eksterne databehandlere se faktaark 10 utgitt av Helsedirektoratet [29].

2.4.5 Fysisk sikring

Sikringstiltak som er nevnt i Seksjon 2.4 er viktig for å hindre uautorisert tilgang, men like viktig er fysisk sikring av områder og utstyr. Områder og utstyr som inneholder helse- og personopplysninger må være sikret slik at kun autorisert personell har tilgang, for eksempel i rom med kodelås [31]. Det bør være en oversikt over hvem som har tilgang og gjøres en revisjon av listen for å fjerne personer som ikke lenger skal ha tilgang.

Gode rutiner for opplæring i retningslinjer og sikkerhetsprosedyrer blant medarbeidere er viktig slik at den enkelte ivaretar informasjonssikkerheten. Eksempler på dette kan være at utskrift av sensitive opplysninger kun skal gjøres på skrivere som er lokalisert i sikrede områder, sensitive papirer skal oppbevares i låst skuff eller skap, PC-er som ikke er i bruk skal være låst og installasjon av programvare skal være godkjent av IT-avdelingen [31].

Sikkerhetstiltakene nevnt i Seksjon 2.4.5 anses ikke som kritiske i dette prosjektet da en eventuell applikasjon vil benytte eksisterende rutiner og sikring ved St. Olavs hospital eller NTNU.

Kapittel 3

Forskningsdesign

Dette kapittelet vil ta for seg forskningsmetodene jeg har benyttet for å besvare problemstillingen og forskningsspørsmålene presentert i Seksjon 1.3. De forskjellige metodene vil bli presentert og jeg vil så redgjøre for valg av metoder og forskningsdesign.

3.1 Forskningsmetoder

I denne seksjonen vil jeg beskrive metoder som kan være aktuelle for dette prosjektet.

3.1.1 Aksjonsforskning og eksperiment

Aksjonsforskning er ifølge Oates [32] en forskningsmetode hvor forskeren går inn i en ekte instans av en organisasjon, et prosjekt eller lignende og utfører en endring eller gjør et arbeid som er av praktsik nyttte for denne instansen. Målet med dette er å generere ny kunnskap og erfaring rundt det som er gjort. I aksjonsforskning samarbeider forskeren med de som er involvert i studien, og teori og praksis kombineres for å oppnå verdi for alle parter.

I et eksperiment vil det være en større kontroll over variabler enn det er i aksjonsforskning og følgelig en mindre grad av realisme. Eksperimenter vil utforske årsak-virkning-forhold og bevise eller motbevise hypoteser under mer kontrollerte forhold enn i aksjonsforskning [32].

3.1.2 Utvikling

I kombinasjon med for eksempel et eksperiment eller en case-studie er det vanlig, for IT relaterte forskningsprosjekter, å inkludere design og implementasjon som en del av forskningsmetoden for å kunne svare på forskningsspørsmål. For å undersøke en ny utviklingsmetode kan det utvikles et system med denne teknikken for å generere ny kunnskap om bruk av denne metoden.

Ved å benytte seg av utviklingsprosessen som en forskningsmetode vil man kunne få ny kunnskap som kan analyseres og brukes til å besvare forskningsspørsmål og hypoteser. Ofte er det problemer som oppstår og uventede resultater som bidrar til kunnskapsgenerering, og det er derfor utfordringene man støter på som er viktige for utvikling som forskningsmetode, i tillegg til det endelige resultatet.

Det er vanlig å gjøre en evaluering av systemet som er blitt utviklet. Det finnes mange kriterier som kan brukes, og man må velge de som passer i forhold til hensikten med systemet. Disse kriteriene kan være funksjonalitet, ytelse, pålitelighet, reliabilitet og så videre. Systemet kan også evalueres på et mer teknisk grunnlag, med bruk av black box- og white box-testing [32].

3.1.3 Intervju

Intervju er en metode for å produsere og samle inn data om et spesifikt emne og kan ses på som en dialog mellom to eller flere personer. Det ligger derimot mer bak et intervju enn en vanlig samtale: Det er gjerne planlagt på forhånd og har som hovedmål å hente ut informasjon fra respondenten; det være seg respondentens synspunkter eller informasjon han har om et tema.

Det finnes tre forskjellige former for intervju: strukturert, semi-strukturert og ustrukturert [32].

Strukturert:
Et strukturert intervju bruker standardiserte og identiske spørsmål som stilles likt til alle respondenter. Intervjueren skal ikke komme med egne kommentarer eller stille oppfølgingsspørsmål til svarene respondenten gir. På så måte kan man se på strukturert intervju som en form for spørreundersøkelse i verbal form.

Semi-strukturert:
I et semi-strukturert intervju har intervjuer forberedt noen temaer som

det skal snakkes om, og noen spørsmål innenfor disse temaene. Intervjuer er derimot mindre låst til spørsmålene og har mulighet til å tilpasse spørsmål og stille oppfølgingsspørsmål avhengig av hvordan samtalen utvikler seg.

Ustrukturert:
I et ustrukturert intervju har intervjueren mindre kontroll og introduserer bare et emne som respondenten kan ytre sine meninger og tanker om. I denne formen skal intervjuer fokusere på å la respondenten snakke fritt og forstyrre ham så lite som mulig.

Uansett valg av intervjuform så er det viktig å ta notater av det som blir sagt. Om det er greit for respondenten, er det også lurt å bruke lydopptak slik at man i ettertid kan høre nøyaktig hva som ble sagt.

3.1.4 Penetrasjonstesting

Penetrasjonstesting er en metode for å finne sårbarheter i systemer. Dette er en prosess som kan gjøres automatisk av programvare eller manuelt av profesjonelle [33]. Metoden går ut på å samle informasjon om systemet man skal angripe, og så bruke denne informasjonen til å finne sårbarheter.

Ofte kalles slik testing for «white hat-attacks» fordi det er de «snille» som prøver å bryte seg inn i systemet.

Det er flere måter å utføre en penetrasjonstest: Det kan gjøres «inhouse» eller av et eksternt firma, det kan testes for eksterne og interne angrep og det kan testes med forskjellige grader av informasjon om systemet.

3.2 Valg av metode

I denne seksjonen vil jeg se på hvilke forskningsmetoder jeg har valgt for å hjelpe meg å svare på den overordnede problemstillingen presentert i Seksjon 1.3 og forskningsspørsmålene presentert i Seksjon 1.3.1.

3.2.1 Systemutviklingscase

For å besvare min problemstilling og mine forskningsspørsmål gjennomfører jeg et systemutviklingscase hvor jeg går inn i forskningsprosjektet MOBESITY.

Utviklingen av systemet gjøres i samarbeid med Anders Rognstad og Tarjei Ormestøyl, men jeg vil fokusere på informasjonssikkerhet og de vil fokusere på brukbarhet.

Systemet som skal utvikles er et egenomsorgssystem for overvektspasienter basert på åpen kilekode. Utgangspunktet for utviklingen er høstens fordypningsprosjekt hvor krav og behov fra brukergruppen ble identifisert. Dette er nærmere forklart i Seksjon 2.1.

Mine erfaringer og kunnskapen jeg genererer gjennom dette prosjektet og utviklingsprosessen av systemet vil være fundamentet for å svare på min problemstilling og mine forskningsspørsmål. I tillegg vil jeg basere meg på relevant litteratur presentert i Kapittel 2

Systemutviklingscaset kan ses på både som aksjonsforskning og som et systemutviklingseksperiment. Jeg undersøker årsak-virkning-forhold ved å se på sikkerhetsutfordringer som oppstår med utvikling basert på fri programvarekomponenter. Graden av kontroll over variabler er likevel ikke tilstede slik den er i vanlige eksperimenter. Det er tydelig at utviklingen er en form for aksjonsforskning da det gir verdi til MOBESITY-prosjektet samtidig som jeg får innsikt og ny kunnskap om sikkerhetsutfordringer i et slikt prosjekt.

3.2.2 Intervju

For å få en større innsikt i krav som stilles til informasjonssystemer som behandler helse- og personopplysninger vil jeg intervjue en eller flere eksperter på området og få et innblikk i deres synspunkter og en klarlegging av kravene som stilles.

Jeg vil benytte meg av semi-strukturerte intervjuer slik at jeg kan få svar på mine spørsmål innenfor de ulike temaer, men også åpne for å tilpasse intervjuet etter hva intervjuobjektet har kunnskap om og hvordan samtalen utvikler seg.

Et slikt møte vil også åpne for muligheten til å diskutere løsningen vår, og valg vi har tatt for å realisere systemet.

Dette vil kunne gi meg verdifull informasjon i forbindelse med design og implementasjon av systemet, og viktig kunnskap ved besvaring av mine forskningsspørsmål.

3.2.3 Penetrasjonstesting

En penetrasjonstesting for å avdekke sårbarheter i systemet er absolutt nødvendig for alle systemer som er åpent tilgjengelige på Internett. I dette prosjektet vil jeg derimot ikke utføre noen penetrasjonstesting av systemet basert på to grunner. Jeg har ikke erfaringen og kunnskapen om hvordan man gjennomfører en penetrasjonstesting og, kanskje viktigere; tidsbegrensninger gjør at jeg ikke har tid til å gjennomføre det på en tilfredsstillende måte.

Kapittel 4

Intervjuer

I dette kapittelet presenteres intervjuene som ble gjennomført med to sikkerhetseksperter; Lillian Røstad og Øyvind Røset. Ved å gjennomføre disse intervjuene håpet jeg å få et større innblikk i de kravene som stilles til systemer som behandler helse- og personopplysninger. Med dette ville jeg forsøke å klargjøre hvilke krav til sikkerhet som eksisterer for vårt system.

4.1 Møte med Lilian Røstad

Underveis i utviklingsprosessen ønsket vi å få tilbakemeldinger fra en sikkerhetsekspert på valgene vi hadde tatt i utviklingen vår og hvilke sikkerhetskrav som var gjeldende for det systemet vi skulle utvikle. Det var også greit å kunne ha muligheten til å diskutere hva som burde implementeres for å sikre systemet vårt. Målet med møtet var å få tilbakemeldinger på vår løsning og hvordan den forholdt seg til gjeldende sikkerhetskrav for systemer av denne typen.

Lilian Røstad er sikkerhetsansvarlig for Lånekassen og førsteamunensis II ved NTNU og har lang fartstid innen informasjonssikkerhet. Vi avtalte å møte henne 16. Mars på kontorene til Lånekassen for å ta en prat. Med på møtet var meg selv, Tarjei Ormestøyl, Anders Rognstad, Anita Das og Dag Svanæs.

Vi hadde på forhånd satt opp noen temaer vi ønsket å ta opp, sammen med noen spørsmål til hvert tema. Vi hadde også fortalt henne om hva møtet skulle handle om. Vi ønsket et semi-strukturert intervju slik at det var rom for å tilpasse spørsmål og temaer etter hvordan samtalen utviklet seg. Temaene vi hadde forberedt var følgende:

- Sikkerhetsnivåer
- Helse og personsensitive opplysninger
- Sikring av informasjon
- Bruk av SMS
- Åpen kildekode
- Utviklingen av vår autentiseringskomponent

For å gi henne et innblikk i prosjektet og hva vi skulle gjøre forklarte vi kort prosjektets kontekst og mål, og ga en kort presentasjon av hvilke komponenter vi hadde valgt å bruke i systemet.

Vi forklarte hvordan vi hadde tenkt å lage et system for overvektspasienter hvor de kunne kommunisere seg imellom og med helsepersonell, og at vi ønsket å realisere systemet ved hjelp av åpen kildekode-komponenter der det var mulig. Videre beskrev vi hvordan vi tenkte å lage vår egen 2-faktor-autentiseringskomponent for å sikre siden mot uvedkommende.

Sikkerhetsnivåer

Som nevnt i Seksjon 2.3 er det definert fire sikkerhetsnivåer for autentisering og uavviselighet i offentlige systemer, og vi spurte Røstad hvilket nivå eksisterende systemer lå på og hva som var kravet til systemer som vårt. Hun forklarte at det som egentlig skilte de forskjellige nivåene var hvor sterkt systemet identifiserte personen som brukte det. Altså med hvor stor sikkerhet man visste at personen som brukte tjenesten var den han utga seg for å være. På nivå 1 var det ingen garanti for at personen var den han utga seg for å være, mens på nivå 4 hadde man sikker identifisering av personen.

Det kom frem at systemer som behandler helse- og personopplysninger skulle tilfredsstille kravene satt til nivå 4, men per dags dato var det ingen som oppfylte disse kravene og de fleste befant seg på nivå 3. Hun fortalte at det ble arbeidet med å utvikle løsninger som tilfredsstilte kravene til nivå 4, og per dags dato var det Norsk Tipping sin autentiseringsløsning med kortleser som var nærmest nivå 4.

> «For å kommunisere sensitive personopplysninger over nett, så skal man være på nivå 4. Og det er det egentlig ingen som er»
>
> — Lillian Røstad

Hun oppsummerer at det er to viktige forskjeller på nivå 3 og 4:

1. Selv om man møter opp personlig for å få opprettet en konto så blir koder sendt i posten. På nivå 4 må man også møte opp personlig for å få utlevert dette, eller få det tilsendt rekommandert i posten.
2. Hvordan nøkklene er lagret. Om de er lagret på et smartkort som er beskyttet mot å bli fiklet med så er det sikrere enn nøkler lagret i software på maskinen.

Helse- og personsensitive opplysninger

Siden vår løsning innebærte et forum hvor brukere kunne utveksle erfaringer og stille spørsmål var det en mulighet for at brukere kunne dele informasjon med andre som kunne være personsensitivt. Vi ønsket derfor å få mer informasjon om problemstillingen rundt dette, og hvilke tiltak som eventuelt måtte på plass for at en slik løsning kunne tas i bruk. Vi hadde vært inne på tanken om informert samtykke fra brukere, der de blir opplyst og sier seg enig i at informasjon de deler på forumet vil kunne ses av andre pasienter i samme situasjon som dem.

Røstad informerte om at bruk av informert samtykke kan brukes for å samtykke til registrering av personsensitive opplysninger, men en pasient kan ikke samtykke til brudd på personvernsloven. Siden det er en lukket gruppe med personer som kommer til å ha tilgang til informasjonen som er lagt ut på nettsiden sier hun at det er mulig man kan få tillatelse med bruk av informert samtykke slik at pasienter er klar over at det som skrives kan sees av andre. Hun presiserer at det er ikke noe endelig svar og sikkerheten rundt lagring av informasjon må uansett ivaretas.

Hun trekker frem at det ville vært lettere å få til et slikt system om det kun var tilgjengelig på et lukket nett, og om det kun kunne aksesseres fra visse terminaler utplassert på sykehuset eller klinikken. Kravene til nivå 4-sikkerhet er gitt når ressursen skal kunne nåes over Internett og være tilgjengelig for brukere fra en hvilken som helst pc. Men det er klart at en slik løsning vil stride imot hele idéen med systemet, siden brukere da måtte møtt opp på sykehus for å bruke det.

Sikring av informasjon

Vi spurte om kravene til kryptering av innhold i databasen og hvorvidt all informasjon som ble lagret der måtte krypteres. Røstad forklarte at ifølge datatilsynets kommentarer til personopplysningsforskriften [34] så må det være to uavhengige sikkerhetsbarrierer som må penetreres før uvedkommende får tilgang til personsensitive opplysninger. Hun forklarer at oftest så realiseres dette ved å ha en ekstern sone og en sikker sone i nettverket, men presiserer at dette bare er en vanlig tolkning av datatilsynets kommentarer og at det også vil være tilstrekkelig med andre sikkerhetsbarrierer som for eksempel kryptering av innhold i database.

Bruk av SMS

En annen problemstilling vi hadde var utsending av informasjon til brukere over SMS. Systemet skulle støtte kalendervarsling via SMS, og dette betydde at det brukeren hadde skrevet i kalenderen ville kunne bli tilsendt via SMS. Siden SMS er en usikret kanal betyr dette at man ikke kan overføre personsensitiv informasjon via SMS. Vi så dette som et problem siden brukeren kunne skrive inn sensitiv data som så ble sendt på SMS, og spurte om hennes synspunkter på dette. Det ble raskt klart at muligheten for å sende personsensitiv informasjon over SMS ikke er bra og ikke burde være mulig. Hun viser til personopplysningsloven som sier at «helsemessige forhold» er definert som sensitivt, altså informasjon som sier noe om din sykdom eller helse. Hun utdyper videre at datatilsynet sin tolkning av dette er at ordet «sykdom» ikke er sensitivt, men alt mulig mer er det. Hun viser til et eksempel hvor en time hos en fastlege ikke regnes som sensitiv informasjon fordi det ikke nødvendigvis betyr at du er syk, mens en time hos en spesialist på sykehuset regnes som sensitivt siden det avslører så mye mer.

Røstad foreslår at i stedet for at en bruker selv skriver det som skal stå i påminnelsene som sendes på SMS, burde det være forhåndsdefinerte påminnelser som brukeren kan velge å sende ut og som ikke inneholder sensitive opplysninger, som for eksempel «Husk å ta vitaminene dine.»

Åpen kildekode

Vi spurte også om Røstads´ synspunkt på det å bruke åpen kildekode-komponenter kontra lukket kildekode-komponenter eller egenutviklede kom-

ponenter. Røstad så ikke noen åpenbare problemer med å benytte seg av åpen kildekode-komponenter i forhold til lukket kildekode og mente at for eksempel problemene vi hadde i forbindelse med integrering og universal innlogging nok også ville vært tilstede om vi hadde brukt lukket kildekode-komponenter. For å ha total frihet i hvordan man integrerte systemet, måtte man skrive komponentene selv fra bunnen av.

Universal innlogging

Et mer implementasjonsspesifikt spørsmål vi hadde gikk på bruken av universal innlogging og at en bruker skulle bli logget inn i alle komponentene våre gjennom en autentiseringskomponent. Vi hadde laget det slik at informasjonen om brukeren ble lagret i hver enkelt brukers «session» og når han var autentisert så ble han logget inn i de andre komponentene med informasjonen som var lagret der. Røstad syntes dette var en dårlig løsning fordi man ikke alltid kunne garantere at en «session» ble drept når en bruker var ferdig og ideelt sett burde derfor ikke den type informasjon lagres der. Hun forklarte at det nok var baksiden med å benytte seg av komponenter skrevet av andre, det være seg om de er åpen eller lukket kildekode kontra å egenutvikle hele systemet.

Vi var svært interessert i å høre om hun hadde noen forslag til andre måter å gjøre dette på, men hun hadde dessverre ingen forslag å komme med.

Til slutt anbefalte Røstad at vi avtalte et møte med Datatilsynet for å få deres synspunkter på systemet vårt og kravene til sikkerhet. Hun poengterte at det ville være lurt å klart kunne legge frem alle sikkerhetstiltak vi hadde implementert og formålet med systemet. Det var da lettere for Datatilsynet å kunne gi informasjon om hvordan man kunne forbedre systemet.

> «Du vil aldri i verden få et ja, men du vil kunne få et nei.»
>
> — Lillian Røstad om Datatilsynet.

4.2 Møte med Øyvind Røset

Etter vårt møte med Lillian Røstad, se Seksjon 4.1, planla vi å få et møte med Datatilsynet for å gi dem et innblikk i systemet vi utviklet, forklare sikkerhetsmekanismene bak systemet og få høre deres synspunkter på systemet. Arild Faxvaag, som er veileder for Anita Das, hadde kontakt med en

person fra Datatilsynet som skulle opp til Trondheim i forbindelse med et møte. I den forbindelse skulle han prøve å ordne det til slik at vi kunne treffe denne personen mens han var i Trondheim. Det ble etter hvert klart at denne personen ikke var den rette for oss å snakke med siden kvalifikasjonene hans lå mer mot juss-siden. Det ble derfor ikke satt opp noe møte og vi måtte prøve på nytt å få ordnet et møte med Datatilsynet. Denne prosessen dro ut i lengden og ble til slutt ikke noe av. Som et kompromiss satte Faxvaag oss i kontakt med personvernsombudet ved St. Olavs Hospital, Øyvind Røset.

Vi hadde på forhånd satt opp noen temaer vi ønsket å ta opp, sammen med noen spørsmål til hvert tema. Disse temaene var som følger:

- Bruk av SMS
- Helse og personsensitive opplysninger
- Sikkerhetskrav i forskningsystemer versus driftssystemer

Øyvind Røset er personvernsombud ved St. Olavs Hospital og vi avtalte å møte ham på St. Olavs 11. Mai for å ta en prat. Med på møtet var meg selv, Anita Das og Arild Faxvaag. Vi fortalte kort litt om prosjektet og hva det innebar og forklarte videre at vi ønsket å diskutere informasjons-sikkerhetsaspekter for en nettportal for kommunikasjon mellom pasienter og helsepersonell.

Ansvarsområde

Røset var svært opptatt av å få informasjon om hvem som sto ansvarlig for dette prosjektet og hvilken rolle sykehuset hadde i prosjektet. Det var viktig å avklare om systemet var ment for forskning eller om det skulle være en del av behandlingstilbudet til klinikken. Han trengte å plassere det i riktig «bås» i forhold til hvilke regler man måtte forholde seg til og hvilket ansvar som sykehuset skulle ha. Vi forklarte at NTNU og NSEP sto som ansvarlig for prosjektet og at målet med prosjektet var forskning og kunnskapsutvikling, men i det lengre løp var det et ønske om at et slikt system skulle kunne brukes som en del av en behandlingsprosess.

For å gi ham et bedre innblikk av systemet viste vi en demonstrasjon av systemet og funksjonaliteten som var tilgjengelig, og forklarte underveis de forskjellige komponentene og hva formålet med dem var.

Helse- og personsensitive opplysninger

CMS delen av systemet og presentasjonen av generell informasjon hadde han ingen spesielle kommentarer på, noe som var å forvente siden dette ikke var sensitiv informasjon og mye av informasjonen lå allerede åpent på sykehusets egne nettsider.

Forumet og muligheten for brukere å dele informasjon og erfaringer med andre brukere var han mer skeptisk til siden brukere da potensielt kunne dele helse- og personsensitive opplysninger med andre brukere av forumet. Vi spurte om dette kunne gjennomføres ved at brukere signerte en samtykkeerklæring hvor de ble gjort oppmerksomme på at informasjon skrevet i forumet kun ville være synlig for andre brukere av systemet. Røset mente at dette ikke var mulig, og forklarte at en person ikke kan samtykke seg vekk fra lovkrav og uansett hva en bruker sier seg enig i så må fremdeles lovene som gjelder for personvern overholdes [35].

> «Du skal ikke gå ut ifra at brukeren vet sitt eget beste»
>
> — Øyvind Røset

For å overholde lovene om personvern kunne ikke brukere ha mulighet til å dele helse- og personopplysninger, noe som i praksis betydde at man måtte hindre brukere i å kunne dele slik informasjon. Det ble foreslått to mulige løsninger på dette: Enten at moderatorer fjerner poster som inneholder helse- og personopplysninger eller at en redaktør må godkjenne alle poster før de blir publisert, for å forsikre seg om at de ikke inneholder sensitiv informasjon.

Røset kommenterte at den første løsningen ikke vill være tilstrekkelig fordi informasjonen allerede var blitt lagt ut og så fjernet og dermed var delt med andre. Den andre løsningen følte han var bedre og ved å gjøre det slik forsikret man seg om at informasjonen som ble delt verken avslørte helse- eller personopplysninger.

Røset var også skeptisk til formålet med direkte kommunikasjon mellom pasienter og helsepersonell i form av private meldinger. Han mente at straks pasienter og helsepersonell skal kommunisere slik så vil helsepersonell ha et ansvar for å svare, og da må de kunne dokumentere hva de har svart og da blir det en del av et behandlingsopplegg. Han trakk også frem at i en slik situasjon vil det måtte være en vurdering på om det var journalverdig informasjon.

Bruk av SMS

Den personlige kalenderen og muligheten for påminnelser via SMS var vi klar over at var problematisk, fordi SMS er en usikker kanal og det skulle derfor ikke kunne sendes helse- og personopplysninger i SMS. Røset bekreftet dette og viste til retningslinjene for bruk av SMS [36]. Vi presenterte den mulige løsningen som kom frem under møtet vi hadde med Lillian Røstad, hvor en bruker kunne velge fra et sett med predefinerte varslinger som kunne sendes ut, noe Røset var enig i at var en bedre løsning som sikret personvernet. Han trakk også frem et system de prøvde ut på sykehuset som hadde muligheten for at leger kunne sende SMS til pasienter med informasjon, hvor sykehuset hadde brukt samme løsning der legene kunne velge fra et gitt sett med predefinerte maler.

Forskning versus behandling

I starten av møtet var Røset som nevnt svært opptatt av å finne ut om prosjektet var forskningsorientert eller skulle om det være en del av behandlingsopplegget og være et system i drift. Han hadde også nevnt at det ble stilt forskjellige krav avhengig av dette. Vi spurte om kan kunne utdype hva som faktisk var forskjellen i kravene når et system var brukt i forskningsøyemed kontra i vanlig drift.

Røset forklarte at forskning gjerne skal prøve ut litt mer, ofte ting som ikke har blitt gjort før, men han presiserte at lovverket setter forskning på lik linje med alt annet. Han fortalte videre at et forskningsprosjekt for eksempel ikke ville trenge å ha de samme kravene til sporbarhet og logging mot journalsystemer. Det ville heller ikke være krav til oppetid på en slik tjeneste siden det ikke er et behandlingsrettet system, men et forskningssystem, så det ville ikke være fare for liv og helse om systemet ikke var tilgjengelig.

Kapittel 5

Sikkerhetskrav til systemet

I dette kapittelet vil jeg presentere sikkerhetskravene som stilles til systemet. Jeg vil se på hvilke sikkerhetskrav jeg hadde til systemet da jeg startet og hvilke nye sikkerhetskrav som kom frem som følge av intervjuene i Seksjon 4.1 og 4.2.

Jeg vil også se på de kravene som er stilt fra det offentlige til systemer som behandler helse- og personopplysninger og vurdere relevansen av kravene opp mot vårt system.

Arbeidet mot den ferdige kravlisten presentert i Seksjon 5.3 var en kontinuerlig prosess som gikk parallelt med utviklingsprosessen.

5.1 Kravliste

Da jeg startet dette prosjektet hadde jeg en kort liste med sikkerhetskrav gjengitt i Tabell 5.1.

ID	**Krav**
1	2-faktor autentisering av brukere
2	Rollebasert autorisering
3	Kryptert kommunikasjon mellom brukere og server
4	Skal kun ha tilgang til en innloggingsside om man ikke er autentisert
5	Første gang en bruker logger inn skal passordet endres slik at kun bruker kjenner til passordet

Tabell 5.1: Sikkerhetskrav

Denne listen var så klart ikke utfyllende nok, og måtte revideres for å speile de kravene til sikkerhet som stilles til systemer som behandler helse- og personopplysninger.

For å gjøre dette benyttet jeg meg av to metoder: Jeg intervjuet noen eksperter på området og jeg undersøkte dokumentasjon fra det offentlige om krav til systemer som behandler helse- og personopplysninger.

5.2 Kravliste fra Helsedirektoratet

Sosial- og helsedirektoratet har utarbeidet en «Norm for informasjonssikkerhet i helsesektoren», som er en samling av krav og retningslinjer for å oppnå tilfredsstillende informasjonssikkerhet i systemer som behandler helse- og personopplysninger. Tabell 5.2 viser en oppsummering av disse kravene hentet fra Norm for informasjonssikkerhet [37], og en vurdering av kravets relevans mot vårt system.

Nr	**Krav**	**Relevans for vårt system**
1.	Det skal registreres når autorisasjon tildeles og avsluttes og hvem som utfører dette, med mindre risikovurdering avdekker at dette ikke er nødvendig.	Sett i forhold til størrelsen på prosjektet, mengden brukere som vil benytte seg av systemet samt at tildelt autorisasjon ved registrering antagelig ikke vil være nødvendig å endre så ofte vurderer vi dette kravet som ikke relevant for vårt system.
2.	All autorisert bruk og forsøk på uautorisert bruk av informasjonssystemene skal registreres og registeret skal lagres i elektronisk form i minimum 2 år.	For å ivareta informasjonssikkerheten vil dette kravet være relevant for vårt system, og registrert data burde lagres frem til prosjektets slutt.
3.	Det skal føres hendelsesregistre (logger) over alle oppslag (lesing), utskrifter, endring, retting og sletting av helse- og personopplysninger.	Dette kravet er ikke relevant siden systemet ikke vil lagre helse- og personopplysninger som kan ses og endres av helsepersonell. Muligheten for utveksling av helse- og personopplysninger ligger i private meldinger som brukere kan sende til andre brukere og til helsepersonell.

Tabell 5.2 – Fortsetter på neste side

Tabell 5.2 – Fortsettelse fra forrige side

4.	Det skal ikke kunne endres opplysninger uten at det registreres hvem som har endret og hva som er endret. Dette krever for eksempel validering av alle felter i systemet.	Dette kravet er relevant for oss, og endringer av informasjon publisert i systemet burde registreres. Ved endring av brukerinformasjon vil det også være viktig å validere alle felter for å bevare konsistens.
5.	Nødrettstilgang skal kunne etableres som en mulighet for spesielt autoriserte brukere til å gi seg selv tilgang.	Tjenestene som systemet skal tilby tilsier ikke at det er noe behov for nødrettstilgang da dette primært er et informasjonssystem som ikke er del av en behandlingskjede.
6.	Årsak for bruk av nødrettstilgang skal registreres.	Ikke relevant for vårt system.
7.	Helse- og personopplysninger skal henføres til rett identifisert person. For eksempel ved bruk av farger for å illustrere hvilken journal som er åpen, bilde av pasient i journalen og i forbindelse med skanning og kobling til rett journal.	Dette systemet vil ikke være koblet mot et journalsystem, og helse- og personopplysninger skal kun forekomme i private meldinger mellom brukere. Dette kravet er derfor ikke relevant.
8.	Helse- og personopplysninger skal føres i henhold til kodeverket.	Det vil ikke føres helse- og personopplysninger i systemet, og slike opplysninger skal kun forekomme i private meldinger mellom brukere. Dette kravet er derfor ikke relevant.
9.	Autorisering skal skje selvstendig for hver enkelt rolle og autentisering må sikre identifisering i korrekt rolle i hvert enkelt tilfelle.	Det er viktig at brukere blir autorisert i riktig rolle ved autentisering slik at brukere kun har de rettigheter de skal ha. Dette kravet er derfor relevant for vårt system.
10.	Ulike ansettelsesforhold (samme person kan ha ulike roller innenfor samme virksomhet) skal identifiseres og ved behov gis ulike autentiseringskriteria.	Dette kravet er ikke relevant for vårt system, siden brukere kun vil tilhøre en spesifikk rolle og ikke flere.

Tabell 5.2 – Fortsetter på neste side

Tabell 5.2 – Fortsettelse fra forrige side

11.	Hendelsesregistrene skal enkelt kunne analyseres ved hjelp av analyseverktøy med henblikk på å oppdage brudd.	Det bør være enkelt å søke i hendelseregistere siden disse kan bli ganske store, og dette kravet er derfor relevant for oss.
12.	Hendelsesregistrene skal sikres mot endring og sletting av uautorisert personell.	For at hendelseregistrene skal kunne stoles på må de sikres mot endring og sletting, og dette kravet er relevant for vårt system.
13.	Systemet skal støtte muligheten for innsyn i (for å ivareta innsynsretten), retting, sletting og sperring av hele/deler av journaler.	Vårt system er ikke, og vil heller ikke være koblet til, et journalsystem. Dette kravet er derfor ikke relevant for vårt system.
14.	Alle systemer skal ha mekanismer som hindrer uautoriserte endringer av helse- og personopplysninger.	Siden helse- og personopplysninger kun potensielt vil bli skrevet i private meldinger mellom brukere, og ikke være registrert i systemet som informasjon om en pasient, er det ikke noen fare for uautorisert endring av opplysninger. Det er derimot viktig å sikre mot uautorisert innsyn.
15.	Autentisering av bruker fra og i interne systemer skal skje med minimum brukernavn og passord.	All bruk av systemet vil gå over Internett, og derfor vil alle måtte logge inn med 2-faktor-autentisering
16.	Passordfil skal krypteres.	Relevant for vårt system.
17.	Tildelt autorisasjon skal kunne tidsavgrenses.	I forhold til størrelsen på prosjektet og mengden brukere som vil benytte seg av det vil det ikke være nødvendig å kunne tidsavgrense autorisasjon.
18.	Det skal være mulig å gjøre en avsjekk av tildelte autorisasjoner.	Det må være mulig å fjerne autorisasjoner gitt til brukere i systemet, og dette er derfor relevant for vårt system.
19.	Passordet bør kunne byttes enkelt av bruker og tvunget skifte bør være teknisk mulig.	Dette kravet er relevant for vårt system, slik at bruker er den eneste som vet passordet sitt.

Tabell 5.2 – Fortsetter på neste side

Tabell 5.2 – Fortsettelse fra forrige side

20.	System som benyttes innen psykisk helsevern bør ha funksjonalitet for å kunne gi medlemmer av kontrollkommisjonen lesetilgang til journalen til enkeltpasienter, jf EPJ-standarden.	Ikke relevant for vårt system.

Tabell 5.2: Sikkerhetskrav for systemer som behandler helse- og personopplysninger.

5.3 Revidert kravliste

For å få en større innsikt i kravene som stilles til systemer som behandler helse- og personopplysninger intervjuet jeg Lillian Røstad, Seksjon 4.1, og Øyvind Øyvind Røset, Seksjon 4.2. Etter samtaler med disse personene og en gjennomgang av kravene som er stilt fra det offentlige, reviderte jeg kravlisten og resultatet kan ses i Tabell 5.3. Noen av kravene fra Helsedirektoratet er reformulert for å bedre passe vårt system.

Det er den reviderte kravlisten i Tabell 5.3 som resten av rapporten forholder seg til.

ID	Krav
Autentiseringskrav	
1	Systemet skal overholde kravene stilt til nivå 4 autentisering
2	Skal kun ha tilgang til en innloggingsside om man ikke er autentisert
3	Autentisering må sikre identifisering i korrekt rolle i hvert enkelt tilfelle.
Autoriseringskrav	
4	Rollebasert autorisering
5	Det skal være mulig å fjerne tildelte autorisasjoner.
6	Systemet skal ha mekanismer som hindrer uautorisert innsyn i helse- og personopplysninger.
Krav til logging	
7	All autorisert og forsøk på uautorisert bruk av systemet skal registreres og lagres til prosjektets slutt.
8	Det skal ikke kunne endres opplysninger uten at det registreres hvem som har endret og hva som er endret.
9	Hendelsesregistrene skal enkelt kunne analyseres ved hjelp av analyse-verktøy med henblikk på å oppdage brudd.
10	Hendelsesregistre skal sikres mot endring og sletting av uautorisert personell.
Krav til personvern	
11A	Brukere skal ikke kunne legge ut helse- og personsensitiv informasjon i forumet
11B	Alle poster til forumet skal godkjennes av en moderator for å sikre at helse- og personopplysninger ikke postes
12A	Det skal ikke være mulig å sende ut helse- og personsensitiv informasjon via SMS
12B	Kalenderpåminnelser som sendes ut via SMS skal være forhåndsdefinerte, og bruker skal ikke selv kunne skrive hva som står i påminnelsen
Andre sikkerhetskrav	
13	Kryptert kommunikasjon mellom brukere og server
14	Første gang en bruker logger inn skal passordet endres slik at kun bruker kjenner til passordet
15	Det må være to uavhengige sikkerhetsbarrierer som må penetreres før uvedkommende skal kunne få tilgang til data
16	Passordfil skal ikke lagres i klartekst.

Tabell 5.3: Revidert kravliste

Kapittel 6

Endelig løsning

I dette kapittelet vil jeg presentere den endelige løsningen og komponentene den består av.

6.1 Introduksjon

Som nevnt i Seksjon 2.1 var det gjennomført et forprosjekt hvor det ble laget en prototype på systemet. Den reviderte kravspesifikasjonen, tilbakemeldinger fra brukere, og erfaringer fra dette prosjektet var grunnlaget for å starte utviklingen av dette systemet.

Mastergradstudentene Anders Rognstad og Tarjei Ormestøyl er en del av samme utviklingsprosjekt, men med et annet fokus. På grunn av dette vil en del av det som blir presentert i dette kapittelet være tilsvarende i deres oppgave.

6.2 Grafisk grensesnitt

I denne seksjonen vil vi presentere utseendet på nettsiden. Vi bruker noen skjermdumper for å forklare hvordan nettsiden er strukturert og for å forklare det fundamentale designet.

6.2.1 Logg inn-skjermene

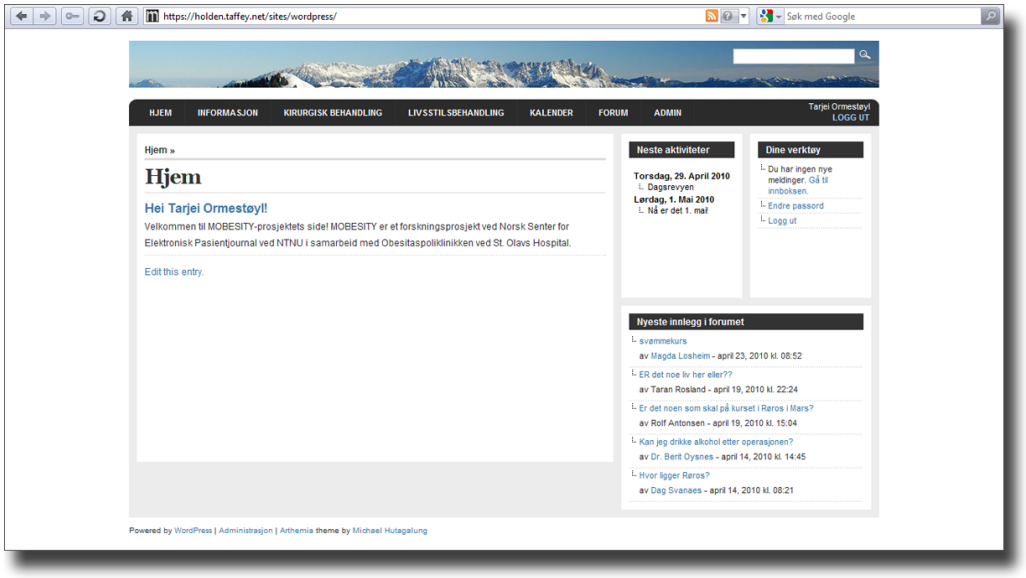
Figur 6.1 og Figur 6.2 viser logg inn-skjermene. Her må brukere skrive inn sitt brukernavn og passord etterfulgt av en engangskode som blir tilsendt på SMS. Disse er ganske enkle og bør være intuitive for de fleste.

Figur 6.1: Logg inn steg 1.

Figur 6.2: Logg inn steg 2.

6.2.2 Hjem-siden

Etter at brukeren har logget inn kommer han til Hjem-siden, vist i Figur 6.3. Dette er en startportal for å bruke nettsiden, og brukeren har snarveier til de viktigste funksjonene via denne siden. Til høyre ser en de nyeste forum-innleggene, oppkommende aktiviteter i kalenderen, eventuelle nye private meldinger, og mulighet for å endre passord.



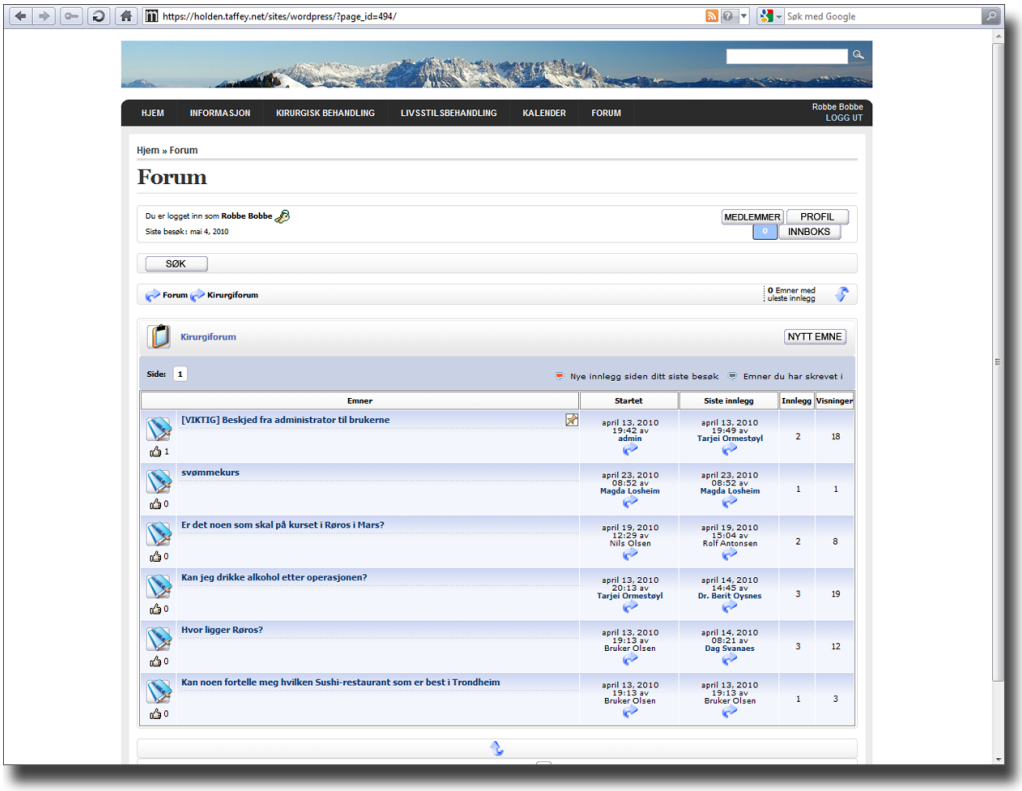
Figur 6.3: Hjem-siden til MOBESITY-nettsiden

6.2.3 Informasjon

Informasjonsdelen av nettsiden er ganske omfattende, siden den tar for seg mesteparten av all informasjon som blir formidlet til pasientene. Opprinnelig ble denne informasjonen nokså ukritisk kopiert inn i nettsiden fra en ekstern kilde. Dette viste seg å ikke fungere veldig bra, da det ble trøblete å navigere, og under en brukertest slet mange med å finne riktig informasjon. Informasjonsstrukturen ble revurdert, og endret slik at det er lettere å finne det man er på utkikk etter. Menyfanen «Informasjon» inneholder generell informasjon, «Kirurgisk behandling» inneholder ganske naturlig informasjon om kirurgisk behandling, og fanen «Livsstilsbehandling» inneholder informasjon om ikke noe annet enn Livsstilsbehandling.

6.2.4 Forumet

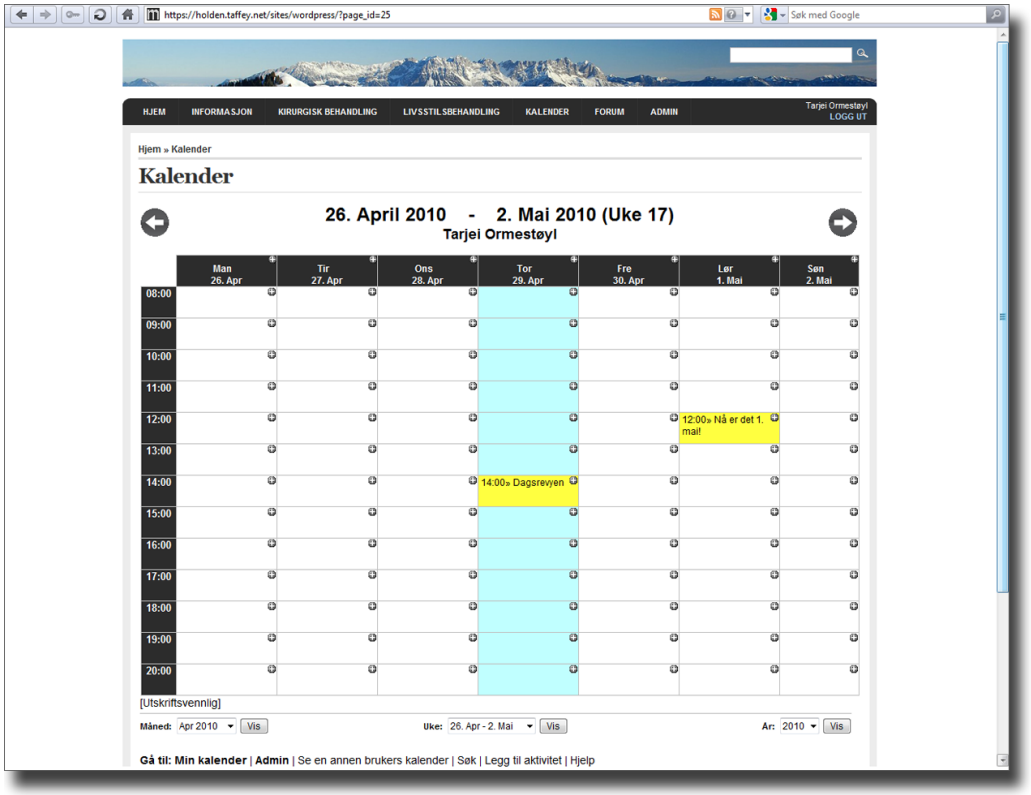
Et skjermdump av forumet er vist i Figur 6.4. Forumet har et forholdsvis standard forumutseende, og bør være kjent for brukere som har brukt lignende tjenester før. Øverst på forumsiden kan brukeren få opp en liste over forumets medlemmer, endre sin profil og besøke innboksen, der han kan motta og sende private meldinger.



Figur 6.4: Forumet til MOBESITY-nettsiden

6.2.5 Kalenderen

Et skjermdump av hovedsiden til kalenderen er vist i Figur 6.5. Her er ukevisning valgt, men det går også an å se måneds- og årsvisning av kalenderen. Brukeren kan legge til aktiviteter ved å enten trykke på plusstegnet på hver time, eller velge «Legg til aktivitet»-valget nederst. Figur 6.6 viser siden man får opp når man legger til en aktivitet. Valgene her er relativt standard kalenderfunksjonalitet som beskrivelser, tidsangivelse og eventuelle gjentagelser. Den tredje fanen gir mulighet for å spesifisere tidspunkt for å få påminnelse om aktiviteten per SMS. Lignende valg som disse er også aktuelle om man velger å redigere en eksisterende aktivitet.

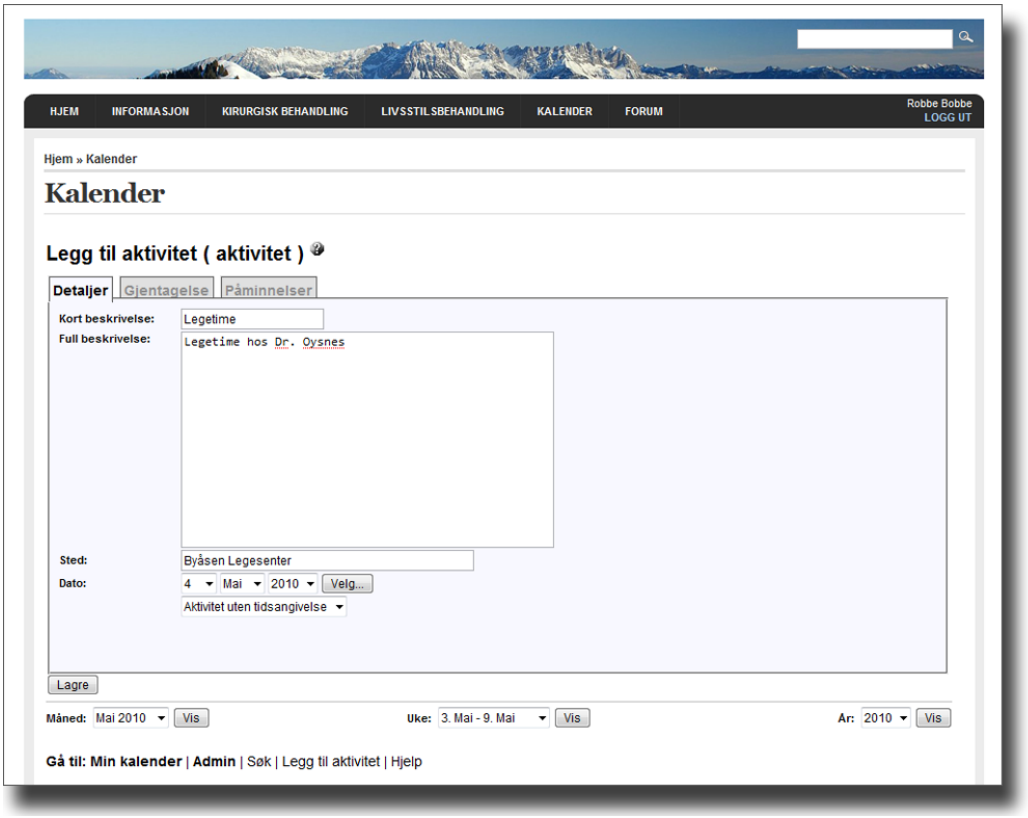


Figur 6.5: Kalender-siden til MOBESITY-nettsiden

6.3 Bruk av åpen kildekode

Målet med utviklingen var å tilfredsstille så mange funksjonelle krav som mulig i systemet vi laget. Samtidig ønsket jeg å tilfredsstille så mange sikkerhetskrav som mulig. For Anita Das, som har MOBESITY-prosjektet som PhD, var det vesentlig at systemet ble ferdig i den grad at det kunne testes på et større antall pasienter i realistisk bruk. For å utfylle så mange krav som mulig innså vi at å skrive systemet fra grunnen av ville ta for lang tid. Ved å kode alt selv ville vi kun klare å fullføre en brøkdel av systemkravene og produktet ville ikke være ferdig nok til å kunne brukes i noen form for testing.

Vi ønsket derfor å ta i bruk mest mulig åpen kildekode-komponenter i utviklingen av systemet, og selv skrive det som var helt spesifikt for MOBESITY.

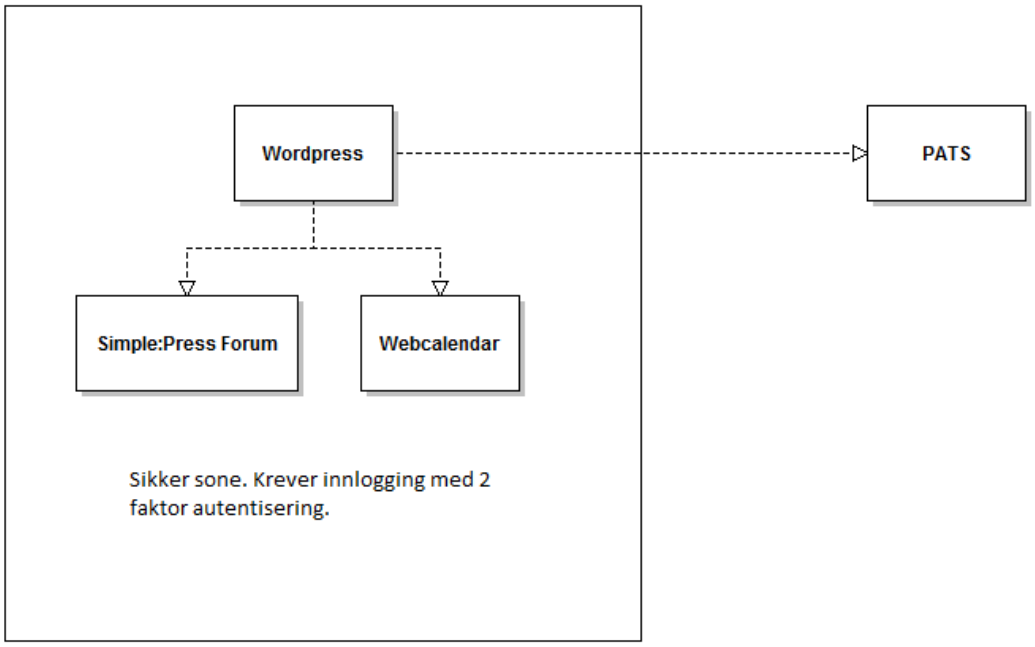


Figur 6.6: «Legg til aktivitet»-siden til MOBESITY-nettsiden

6.3.1 Valg av åpen kildekode-komponenter

For å få en oversikt over hva vi trengte av komponenter gjennomførte vi en idémyldring. Resultatet av denne ble en liste over komponenter som var nødvendige for å tilfredsstille de viktigste kravene til systemet. Vi kom fram til at de følgende punktene ville dekke mesteparten av funksjonaliteten som var ønsket.

- Et publiseringsverktøy for å formidle informasjon til brukere.
- Et forum for deling av informasjon og erfaringer.
- En kalender for påminnelser om spising, trening, vitaminer o.l.
- Et meldingssystem for private meldinger mellom brukere.
- En elektronisk dagbok.



Figur 6.7: Oversikt over komponentene i vårt system

En overordnet beskrivelse av de forskjellige komponentene vises i Figur 6.7, og er nærmere forklart i Seksjon 6.4, 6.5, 6.6 og 6.8.

6.4 Publiseringsverktøy: Wordpress 2.9.2

Den viktigste modulen i systemet er publiseringsverktøyet[1]. Etter råd fra veileder og etter en samtale med Professor Maria Letizia Jaccheri valgte vi å ta i bruk Wordpress som vår CMS-løsning. Vi så nærmere på Wordpress, og syntes dette virket som et godt verktøy å ta i bruk.

Denne modulen er på mange måter hovedmodulen, og de andre komponentene ble integrert inn i denne. Modulen håndterer også informasjonsbehandlingen på en god måte. Wordpress benytter seg av brukere med ulike roller og i tillegg eksisterer det et utall innstikk («plugins») til Wordpress, som vi har tatt i bruk og i noen tilfeller modifisert til å passe systemets behov.

Vi valgte å bruke Wordpress[2] versjon 2.9.2. Neste generasjon Wordpress (3.0)

[1] Kanskje bedre uttrykt på engelsk ved Content Management System (CMS).
[2] `http://wordpress.org/`

er like rundt hjørnet, og aller helst ville vi nok ta i bruk denne. Men siden denne versjonen kun var tilgjengelig i beta på dette tidspunktet, valgte vi å bruke den stabile 2.9 versjonen.

I ettertid kan det virke som om Wordpress ikke var det beste valget av publiseringsverktøy, og selv om vi fikk det anbefalt så burde en grundigere undersøkelse vært gjort. Det viste seg vanskelig å tilpasse visse aspekter ved Wordpress for å tilfredsstille kravene stilt til systemet, men heldigvis var det bare vanskelig, ikke umulig. På dette tidspunkt var det allerede lagt ned veldig mye tid i Wordpress og det var ikke lenger hensiktsmessig å bytte publiseringsverktøy.

Blant andre CMS-verktøy som er utbredt finnes Drupal[3] og Joomla[4]. Litt undersøkelser gjort i ettertid viser at av de tre nevnte systemer er Drupal kjent for å være enklest å tilpasse til egne behov. På en annen side går Drupal for å være minst brukervennlig å administrere for brukere som ikke er teknisk kyndige. Drupal tilbyr også en bedre løsning ved å støtte flere brukere der det er behov for brukerspesifikt innhold. Det som taler for Wordpress er at det i utgangspunktet er veldig enkelt å sette opp samtidig som det er enkelt å forholde seg til for nye brukere. For oss som utviklere hadde det vært en fordel med et publiseringssystem som var enklest mulig å tilpasse, men samtidig er det personer som ikke er spesielt teknisk kyndige som skal administrere systemet, og da er det en fordel at det er intuitivt og lett å forstå.

6.4.1 Innstikk og temaer

Wordpress tilbyr en stor mengde innstikk og temaer, hovedsaklig gjennom tredjeparter, og de fleste av disse er gratis å installere og ta i bruk. For å integrere de ulike komponentene bedre og tilpasse Wordpress etter behov tok vi i bruk noen av disse.

Auto Login Et selvlaget innstikk som er en metode for å autentisere brukeren i Wordpress gjennom vår logg inn-funksjonalitet, se Seksjon 7.5.1.

Exec-PHP Et innstikk som gjør det mulig å skrive PHP-kode i sider, «widgets» og poster[5].

[3] `http://www.drupal.org`
[4] `http://www.joomla.org`
[5] `http://bluesome.net/post/2005/08/18/50/`

Force SSL Et innstikk som krever at all kommunikasjon går over https-protokollen for å øke sikkerheten[6].

Simple:Press Foruminnstikket, presentert i Seksjon 6.5.

WP Security Scan Et innstikk for å søke etter sikkerhetshull på nettsiden[7].

I tillegg til innstikkene benytter vi oss av temaet Arthemia[8] for Wordpress. Temaet tilbyr et pent utseende kombinert med hendig «widget»-behandling og god navigering. Dette var et av de mest passende temaene vi kom over.

6.5 Kommunikasjonsverktøy: Simple:Press 4.2.2

Vi var på utkikk etter et kommunikasjonsverktøy som kunne ta seg av de ulike kommunikasjonsveiene mellom pasienter og helsepersonell. Opprinnelig falt valget på phpBB[9], som er en fullstendig internettforum-pakke og tilbyr en mengde funksjonalitet tilgjengelig gjennom moduler som kan installeres etter behov. Vi fikk phpBB anbefalt av veileder, og i tillegg hadde et av gruppemedlemmene erfaring med å sette opp og administrere denne forumpakken.

Da vi skulle integrere phpBB og Wordpress møtte vi likevel på et problem - det viste seg vanskelig å få til en sømløs integrasjon med felles innlogging i alle komponenter.

Før vi investerte for mye tid i å integrere phpBB undersøkte vi andre alternativer for kommunikasjon. Vi fant Simple:Press[10], som er en forumplugin til Wordpress. Dette forumet hadde all funksjonalitet vi trengte og var enkel å integrerte i Wordpress. I tillegg tillot den oss å bruke den samme brukerdatabasen på tvers av komponentene. Valget falt derfor på å bruke dette forumet i vårt system.

Simple:Press har standard forumfunksjonalitet, med mulighet for flere forum og brukergrupper. Simple:Press støtter også private meldinger mellom forummedlemmene. På denne måten tilfredsstilte vi flere krav gjennom å ta i bruk modulen, og vi slapp å involvere flere eksterne komponenter.

[6] `http://wordpress.org/extend/plugins/force-ssl/`
[7] `http://semperfiwebdesign.com/plugins/wp-security-scan/`
[8] `http://michaelhutagalung.com/2008/05/arthemia-magazine-blog-wordpress-theme-released/`
[9] `http://www.phpbb.com`
[10] `http://simple-press.com/`

6.6 Kalenderverktøy: WebCalendar 1.2.0

Når det kom til valg av kalenderfunksjonalitet hadde vi lært av tidligere feil og evaluerte de kalenderne vi kom over med tanke på støttet funksjonalitet, sikkerhet og brukervennlighet. Vi fant flere alternativer, som Kronolith[11], Chronos[12] og WebCalendar[13]. Av de vi fant vurderte vi WebCalendar som den kalenderen som passet best for vårt system.

WebCalendar er et åpen kildekode-prosjekt som tilbyr standard kalenderfunksjonalitet, samt en del funksjoner som vi fant nyttige. Det finnes blant annet støtte for flere brukere, norsk språk, påminnelser og ulike kalendervisninger [38]. Det er også en mulighet for å sende ut kalenderpåminnelser på e-post, og vi så her en mulighet for å modifisere dette til å sende ut SMS-påminnelser, noe vi var suksessfulle i å gjøre.

WebCalendar har vært i utvikling siden 2005 og har vært i «stable» siden 2007, og utifra antall «patcher» som er gitt ut virker det som det er nok utviklere som jobber med prosjektet.

WebCalendar er fortsatt under utvikling, og benytter seg av Sourceforge.net som vert for utvikling av åpen kildekode-prosjektet. Siden til WebCalendar finnes på `http://sourceforge.net/projects/webcalendar/`, og her kan en laste ned kalenderen, laste ned all kildekode, bli med på utvikling, diskutere i forumet og mer.

6.7 Elektronisk dagbok

Det var ønskelig for brukerne å kunne føre diverse dagbøker, for eksempel treningsdagbok, måltidsdagbok og vanlig dagbok, som igjen kunne deles med helsepersonell. Vi fant ingen åpen kildekode-komponenter som tilfredsstilte disse kravene og denne funksjonaliteten måtte derfor implementeres fra grunnen av.

Implementeringen ble påbegynt, men mangel på tid og ressurser førte til at andre moduler ble prioritert slik at kvaliteten på disse ble tilfredsstillende. Dagbok-funksjonaliteten kom derfor ikke med i det endelige systemet.

[11] `http://www.horde.org/kronolith/`
[12] `http://chronoss.sourceforge.net/`
[13] `http://www.k5n.us/webcalendar.php`

6.8 Universal innlogging

Vi trengte en komponent som på en sømløs måte integrerte brukerhåndtering og innlogging på tvers av de ulike komponentene systemet bestod av. I utgangspunktet tilbød Wordpress delvis en måte å gjøre dette på, men siden systemet skulle håndtere svært personsensitive opplysninger tilfredsstilte ikke dette kravene til sikkerhet.

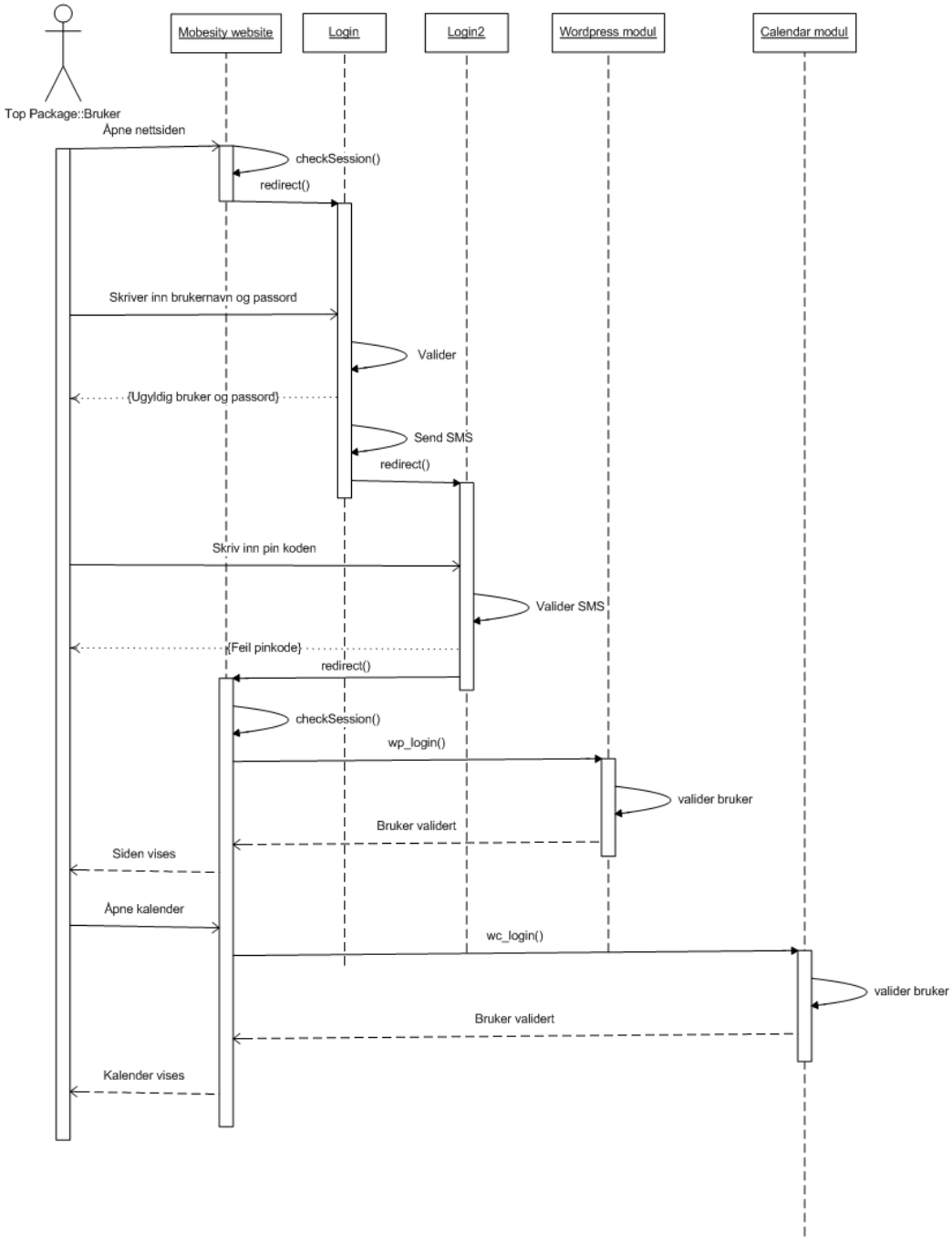
For å kunne realisere vårt system med bruk av åpen kildekode-komponenter måtte vi derfor finne en måte å tilfredsstille kravene som stilles til sikkerhet for et slikt system. For det første var det krav om 2-faktor-innlogging til et system som dette. Vi ønsket å benytte oss av en midlertidig SMS-kode for å løse dette, men vi fant ikke noen støtte for dette blant Wordpress-innstikkene vi undersøkte. Det var også nødvendig med ytterligere krav til sikkerhet enn det Wordpress kunne tilby, og vi trengte derfor en egen brukerdatabase til lagring av brukerinformasjon og for å autentisere brukere mot.

Vi valgte følgelig å utvikle en egen komponent for å håndtere disse kravene. Dette var en tidskrevende oppgave, da vi måtte integrere denne funksjonaliteten på en god måte i de ulike åpen kildekode-komponentene. Funksjonaliteten som skulle oppfylles av denne komponenten var som følger:

1. Brukere skulle autentiseres med en 2-faktor-innlogging.
2. Brukere skulle kun se en innloggingsside om de ikke var logget inn.
3. Brukere skulle autentiseres og autoriseres i alle komponenter gjennom samme innlogging.
4. Brukere skulle ha ulike rettigheter i de forskjellige komponentene.
5. Oppretting, endring og sletting av brukere skulle propageres til alle brukerdatabasene til de forskjellige komponentene.

I Figur 6.8 er innloggingsmekanismen beskrevet ved hjelp av et sekvensdiagram. Når en bruker prøver å aksessere nettsiden, sjekker systemet om brukeren har en «session» på serveren, altså om brukeren er autentisert. Hvis brukeren ikke er logget inn vil denne sjekken feile og videresende brukeren til innloggingssiden vist i Figur 6.1.

Om brukeren har tilgang på siden sender han inn sitt brukernavn og passord og systemet sjekker først om brukernavnet finnes. Om brukernavnet finnes valideres passordet bruker sendte inn mot passordet som er lagret i systemet. Hvis valideringen er vellykket sendes det en sms med en tilfeldig generert PIN-kode til mobilnummeret som er registrert på brukeren og brukeren



Figur 6.8: Sekvensdiagram av innloggingsmekanismen

sendes videre til innloggingssteg to, se Figur 6.2. PIN-koden har begrenset levetid og når den tastes inn sjekkes det at den ikke er for gammel, og så om den er lik koden som ble sendt ut. Om PIN-koden er riktig så blir brukeren autentisert og videresendt til nettsiden.

Når brukeren er blitt autentisert gjennom vår komponent vet vi at brukeren har lov til å bruke systemet, men siden alle åpen kildekode-komponentene som er brukt har sin egen brukerdatabase og innlogging, må komponenten sørge for at brukeren blir autentisert i alle komponentene. Det er også brukerdatabasene til de forskjellige åpen kildekode-komponentene som autoriserer brukere. De bestemmer altså hvilke rettigheter en bruker skal ha. En administrator vil naturligvis ha flere rettigheter enn en vanlig bruker. Wordpress og forumet har sin egen brukerdatabase som beskriver hvilke rettigheter en bruker har og det samme har kalenderen. For å autentisere og autorisere brukere i Wordpress og forumet, har vi skrevet et innstikk til Wordpress som tar seg av dette. Måten det fungerer på er at hver gang en bruker prøver å laste en side i Wordpress så sjekkes det om brukeren har en gyldig «session», altså om brukeren har gjennomført 2-faktor autentisering. Dersom brukeren har dette, logges brukeren med det samme brukernavnet inn i Wordpress. Dette fungerer siden oppretting av brukere fører til at brukeren legges i alle brukerbasene og brukernavn er unikt. I kalenderen fungerer det litt anderledes: Siden denne er integrert i Wordpress, må man være autentisert i Wordpress for å få tilgang til den. Når brukeren prøver å åpne kalenderen vil kalenderen sjekke hvilken bruker som er autentisert ved hjelp av sessionen og logge den samme brukeren inn i kalenderen. Å logge ut fungerer mye på samme måten; vår komponent logger brukeren ut fra alle komponenter og avslutter så «sessionen».

På denne måten kan brukere lettere autoriseres i de forskjellige komponentene uavhengig av hverandre. En bruker kan ha rettigheter til å moderere forumet, men ikke til å endre informasjon som er lagt ut i Wordpress, eller ha administratorrettigheter for Wordpress og forumet, men ikke lov til å se alle brukere sine kalendere.

Komponenten måtte administrere alle brukerdatabasene slik at det å legge til nye brukere, endre eksisterende brukere og slette brukere kunne gjøres fra et grensesnitt og at endringene ble propagert til alle komponentene. For å gjøre dette har komponenten sitt eget grensesnitt hvor slike endringer kan gjøres som er tilgjengelig for administratorer i Wordpress. Når en bruker legges til, endres eller slettes herfra vil endringene gjøres i alle brukerdatabasene slik at brukeren opprettes med de spesifiserte rettigheter.

6.8.1 SMS-utsending

SMS-utsendingen vi bruker benytter seg av Telenor sitt PATS-rammeverk[14]. IDI har en avtale med Telenor om fri bruk av denne løsningen i forskningsøyemed og det egnet seg derfor godt til vårt formål. Rammeverket tilbyr blant annet en enkel Web Service for utsending av SMS som vi kunne bruke både i autentiseringskomponenten og til kalenderpåminnelser.

[14] `http://www.pats.no/?q=node/64`

Kapittel 7

Utviklingsprosess

I dette kapittelet vil jeg se på implementasjonen av systemet vårt

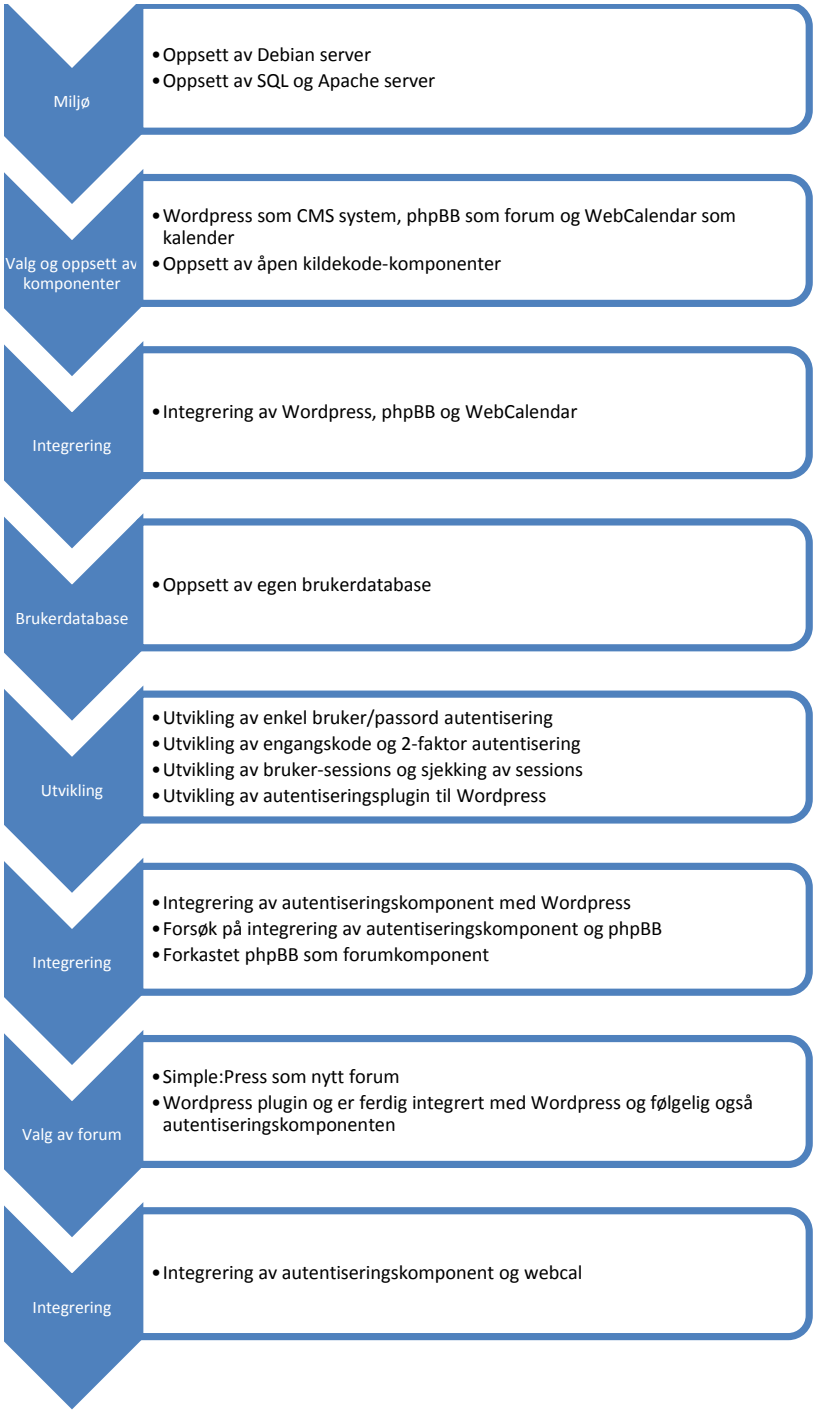
Før vi kunne sette igang trengte vi en server som kunne kjøre vår planlagte nettside. Vi var ikke ukjente med å sette opp servere og det tok ikke lange tiden å få opp en fungerende webserver. Vårt oppsett besto av en Debian[1] Linux-server som kjørte en Apache[2] webserver og en MySQL[3] databaseserver.

- *Oppsummeringer av mine erfaringer vises i bokser slik som denne.*

7.1 Oppstart

Ved oppstart i januar 2010 var prosjektets medlemmer allerede godt kjent med MOBESITY-prosjektet fra høstens fordypningsprosjekt. De funksjonelle kravene til systemet som la grunnlaget for utviklingen var basert på krav-spesifikasjonen til Anita Das, revidert av oss under fordypningsprosjektet gjennom samtaler med pasienter og helsepersonell samt en vurdering av hva som var gjennomførbart.

[1] `http://www.debian.org`
[2] `httpd://www.apache.org`
[3] `http://www.sun.com/mysql`



Figur 7.1: Tidslinje av utviklingsprosessen

Vi startet utviklingen i midten av januar og utviklet i iterasjoner og prøvde å ha ukentlige møter med veileder for å få tilbakemelding på implementert funksjonalitet. Vi fulgte ikke en mer spesifikk utviklingsmodell enn at vi baserte oss på ukentlige iterasjoner, og dette passet fint siden systemet var såpass komponentbasert og fordi komponentene fungerte uavhengig av hverandre i starten.

Ideelt sett i utviklingsprosjekter hvor informasjonssikkerheten er viktig å ivareta så bør man fokusere på sikkerhet gjennom hele utviklingsprosessen. Siden vi baserte oss på en såpass utstrakt bruk av åpen kildekode-komponenter så ble dette vanskelig å gjennomføre. Følgelig ble komponentene først satt opp og integrert sammen til en helhetlig nettside, og deretter fokuserte vi på å implementere sikkerhetstiltak for å tilfredsstille kravene som er stilt til informasjonssikkerhet.

Figur 7.1 viser en tidslinje over utviklingen.

7.2 Oppsett av åpen kildekode-komponenter

Som nevnt i Seksjon 6.4 så skulle vi bruke Wordpress som et CMS-system. Vi måtte derfor sette opp en Apache-server med Wordpress, noe som ifølge Wordpress sine egne nettsider ikke skulle ta mer enn fem minutter. Vi prøvde først å installere Wordpress fra Aptitude, men det viste seg å ta ganske mye lenger tid enn fem minutter. Installeringen skulle være ganske rett frem og selvforklarende, og man skulle kjøre en install.php fil via en nettleser som skulle installere Wordpress. Problemet var at denne installeringen tydeligvis ikke gikk som den skulle, for Wordpress fungerte ikke. Etter mye søking fant vi ut at flere hadde hatt samme problemet som oss, og løsningen for dem hadde vært å laste Wordpress ned direkte fra hjemmesiden og ikke bruke Aptitude eller andre pakkehåndteringssystemer. Og sant nok, da vi gjorde dette tok installeringen ikke mer enn fem minutter. Akkurat hva som var problemet fant vi aldri ut, men Wordpress fungerte som den skulle.

Vårt valg av forum falt på phpBB, som er en komplett internettforum-pakke. Denne hadde all funksjonalitet man kan regne med å finne i et forum og var svært utbredt brukt med et stort samfunn rundt seg. Installeringen av forumet var svært enkel og vi støtte ikke på noen problemer med oppsettet. Etter kort tid hadde vi et fungerende forum oppe å kjøre. Som nevnt var phpBB en omfattende forumpakke med mye funksjonalitet, og mye av denne funksjonaliteten hadde vi liten bruk for i vårt forum. Det viste seg enkelt å

tilpasse forumet og man kunne lett fjerne og legge til funksjonalitet via et grafisk grensesnitt. På denne måten kunne vi fjerne all funksjonalitet som våre brukere ikke ville trenge, og dette ga også et noe mer oversiktlig grensesnitt.

Valg av kalender falt på WebCalendar, se Seksjon 6.6, som tilbød standard kalenderfunksjonalitet samt støtte for flere brukere, påminnelser og norsk språk. Installeringen av denne modulen var også lettvinn, og vi støtte ikke på noen problemer i oppsettet av kalenderen.

- *Totalt sett var det svært enkelt å få de valgte åpen kildekode-komponentene installert og klare til bruk, med unntak av litt trøbbel med Wordpress som vi kanskje burde unngått ved å ha brukt kildekoden fra hjemmesiden deres.*

7.3 Integrering av åpen kildekode-komponenter

Når alle komponentene var installert og satt opp, var de tre separate websider og vi måtte derfor integrere dem sammen til en webside. Wordpress fungerte som «hovedsiden» og phpBB og WebCalendar måtte derfor integreres som en del av Wordpress.

For å integrere phpBB fant vi Wordpress innstikket «WP-United»[4]. Ved å bruke dette innstikket kunne vi integrere phpBB som en del av Wordpress, og det var også muligheter for kobling av brukerkontoer i Wordpress og brukerkontoer i phpBB. Med dette fikk vi en sømløs integrasjon mellom de to komponentene samtidig som brukerkontoer ble koblet på tvers av komponentene.

Men som vi skal se i Seksjon 7.5.2 viste det seg trøblete å integrere phpBB som en del av en universal innlogging. Vi fant et nytt forum, Simple:Press[5], som er et foruminnstikk til Wordpress. Dette forumet hadde all funksjonalitet vi trengte til vår bruk og siden det var et innstikk til Wordpress var det enkelt å integrere. Den tillot oss også å bruke samme brukerdatabase og innlogging som Wordpress, noe som ville gjøre jobben med å lage en universal innlogging litt lettere siden det ble én mindre innlogging å forholde seg til.

[4] `http://www.wp-united.com/`
[5] `http://simple-press.com/`

WebCalendar viste seg å være den vanskeligste komponenten å integrere sammen med de andre til en helhetlig side. Måten koden var strukturert på gjorde det vanskelig å sette seg inn i og endre koden. Det var også komplisert å bruke WebCalendar-metodene utenfor WebCalendar-komponenten, og kall til metoder i kalenderen gjort fra Wordpress var derfor ikke lov.

Wordpress legger heller ikke tilrette for å integrere andre komponenter utenom ved bruk av innstikk, og det å skrive om kalenderen til et Wordpress-innstikk var en jobb for stor til å ta fatt på.

Etter mer grundige undersøkelser rundt problemet, kom vi frem til at vi måtte ta i bruk HTML iFrame-elementet for å inkludere kalenderen i en side i Wordpress. Vi fikk da kalenderen som en side i Wordpress, men det var ikke en ideal løsning.

Når alle komponentene var integrert sammen til en felles nettside var utfordringen å lage en autentiseringsmekanisme som tilfredsstilte sikkerhetskravene og som fungerte som en felles innlogging. Brukere skulle kun behøve å logge inn én gang for så være autentisert og autorisert i alle komponentene, og dermd ikke ha behov for å logge inn i de forskjellige komponentene individuelt. Dette ser vi nærmere på i Seksjon 7.4.

- *Wordpress la ikke til rette for integrering av andre komponenter uten bruk av innstikk, og det var vanskelig å tilpasse andre komponenter til å bruke slike innstikk.*

7.4 Utvikling av universal innlogging

Siden sikkerheten i Wordpress, Simple:Press og WebCalendar ikke var tilstrekkelig i forhold til de krav som stilles for systemer som behandler helse- og personopplysninger, måtte vi legge til ekstra sikkerhetsmekanismer slik at kravene ble oppfylt. Hovedsakelig var det autentisering av brukere, se Seksjon 2.4.3, som ikke hadde adekvat sikkerhet. Vi var på utkikk etter en åpen kildekode-komponent som helt eller delvis tilfredsstilte kravene til sikkerhet, men vårt søk returnerte ingen resultater. Vi bestemte oss derfor for å utvikle denne komponenten selv og integrere den med systemet.

- *Våre krav til autentisering var ganske spesifikke og vi fant ikke noen åpen kildekode-komponenter som tilbød slik funksjonalitet.*

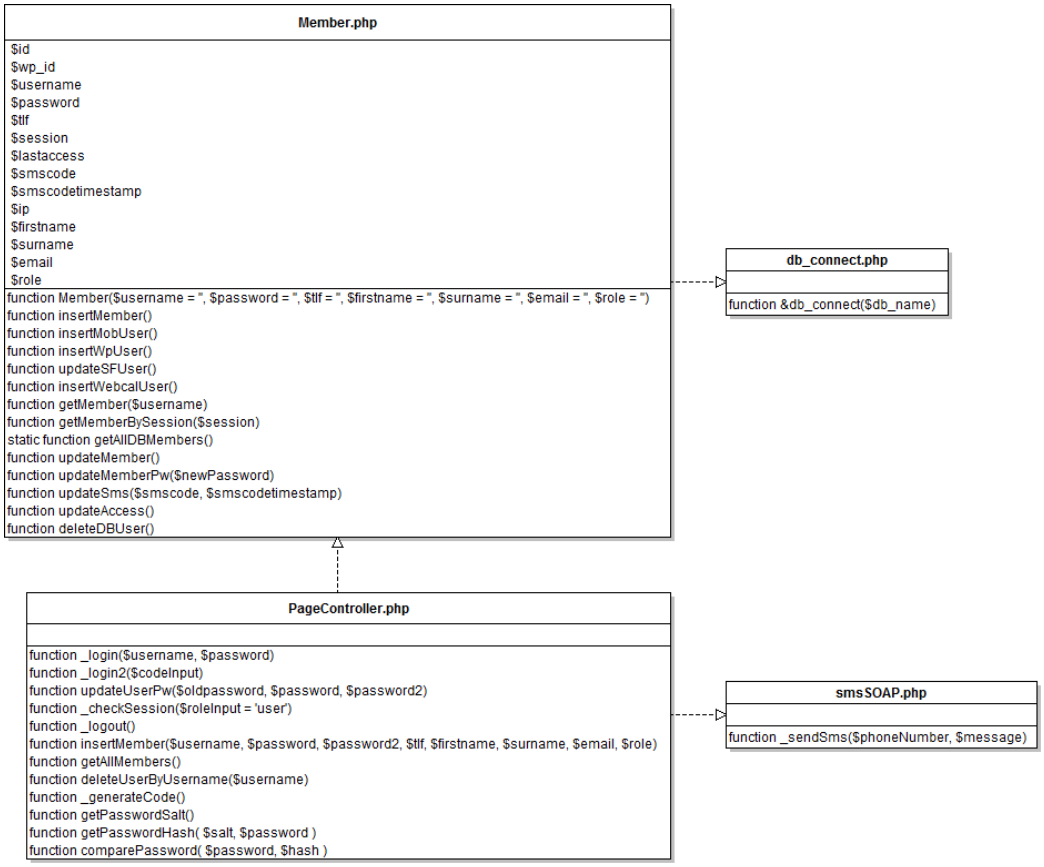
Vi startet prosessen med en idémyldring for å kartlegge hva vi ville trenge for å utvikle en slik komponent. Jeg hadde ikke mye erfaring med autentiseringsmekanismer fra før av, men den grunnleggende funksjonaliteten er å kontrollere om input fra bruker stemmer med det som er lagret i en database. Det første vi trengte for å kunne autentisere brukere i systemet vårt var derfor en brukerdatabase som kunne lagre relevant informasjon om brukere. Denne databasen er beskrevet i Tabell 7.1. Denne beskrivelsen er av den endelige databasen, og det var ikke alle feltene her som var med helt fra starten av.

Field	Type	Null	Key	Default	Extra
id	int(11)	NO	PRI	NULL	auto_increment
username	varchar(20)	NO	UNI		
password	char(255)	NO			
session	char(32)	YES		NULL	
ip	varchar(15)	YES		NULL	
tlf	varchar(12)	NO			
smscode	varchar(4)	YES		NULL	
smscodetimestamp	varchar(25)	YES		NULL	
firstname	varchar(32)	YES		NULL	
surname	varchar(32)	YES		NULL	
email	varchar(64)	YES		NULL	
lastaccess	varchar(25)	YES		NULL	
role	tinyint(1)	YES		NULL	

Tabell 7.1: Beskrivelse av brukerdatabasen

Når vi hadde laget vår database kunne vi begynne å utvikle funksjonaliteten til komponenten. Figur 7.2 viser et pseudo-klassediagram av det endelige systemet vi her beskriver utviklingen av.

Det første vi trengte var en måte å koble til databasen vår for å hente ut den informasjonen som lå lagret der. For å opprette en kobling til databasen skrev vi grensesnittet *db_connect.php*. Dette grensesnittet tilbyr en metode, *db_connect($db_name)*, som oppretter en databasetilkobling enten til vår brukerdatabase eller en av åpen kildekode-komponentene sine databaser og returnerer denne koblingen. Med en slik kobling på plass kunne vi sende spørringer og hente ut resultater fra spørringer mot de forskjellige databasene.



Figur 7.2: Klassediagram av autentiseringskomponenten

For å håndtere brukerinformasjonen vi hentet ut fra databasen trengte vi en klasse hvor denne informasjonen kunne lagres, og vi lagde derfor klassen *Member.php*. Denne klassen har variabler for alle feltene i brukerdatabasen vår, slik at vi kunne lage et Member-objekt for hver bruker vi hentet ut fra databasen. Metoden *getMember($username)* bruker grensesnittet for kobling mot databasen, henter ut all informasjon om den gitte brukeren og returnerer den.

Med dette på plass kunne vi skrive funksjonaliteten for å sammenligne brukernavn og passord som bruker skriver inn mot innholdet i databasen. Metoden *_login($username, $password)* tok inn brukernavnet og passordet som ble skrevet inn av bruker og opprettet så et nytt Member-objekt ved å hente ut informasjonen knyttet til det gitte brukernavnet. Hvis brukernavnet eksisterte i databasen så ble passordet oppgitt saltet og «hashet» på samme måte som passordet i databsen, og sammenlignet med passordet i databasen. Hvis passordene var like returnerte metoden true og brukeren var autentisert.

- *Startet med å lage funksjonalitet for å koble til brukerdatabase og sjekke om brukernavn og passord var korrekt.*

Som et krav til systemer som behandler helse- og personopplysninger måtte det være 2-faktor-autentisering av brukere, og vi måtte derfor implementere engangskoder ved innlogging av brukere. Ifølge Røstad var flere måter å realisere engangskoder i autentisering, enten med bruk av et kodeark, en kodebrikke slik de fleste bruker i nettbanken eller kode sendt på SMS til brukerens mobil. Det letteste for oss var å benytte seg av engangskoder sendt på SMS siden dette hadde minst krav til infrastruktur og var lettest å realisere. For å kunne gjøre dette trengte vi først en måte å sende SMS til brukerne våre. IDI har en avtale med Telenors PATS-rammeverk[6] og kan benytte dette i forbindelse med forskning. Rammeverket tilbød en enkel web service for utsending av SMS. Her fulgte det også med ferdig kode for hvordan man bruker grensesnittet for å sende SMS. Denne eksempelkoden var skrevet i Java og vi prøvde derfor å få Javakoden til å kjøre sammen med vårt system. Dette viste seg å være en omfattende jobb, siden hele vårt system var skrevet i PHP, og selv om vi kunne sende SMS ved å kjøre Javakoden så klarte vi ikke å integrere den med våre systemer. Vi måtte derfor skrive vårt eget grensesnitt for å bruke PATS-rammeverket og sende SMS. Grensesnittet *smsSOAP.php* tilbyr en metode, *_sendSms($phoneNumber, $message)*, som tar i bruk web servicen til PATS-rammeverket og sender en gitt melding til et gitt telefonnummer. Det viste seg greit å skrive om grensesnittet til PHP, og det var lett å integrere det nye grensesnittet med resten av systemet.

- *Utvidet funksjonaliteten slik at engangskode ble sendt på SMS, og denne måtte skrives inn for å fullføre autentisering.*

For å generere en tilfeldig kode implementerte vi en liten metode, *_generateCode()*, i *PageContoller.php* som benyttet seg av PHP sin *rand()* funksjon. Metoden returnerte et pseudo-tilfeldig tall mellom 1000 og 9999. Vi modifiserte så *_login($username, $password)* metoden slik at hvis brukernavn og passord var riktig, så ble engangskoden generert og sendt til telefonnummeret lagret i brukerens profil. Den genererte koden ble, sammen med klokkeslettet den ble opprettet, lagret i databasen og brukeren ble videresendt til en ny side hvor engangskoden måtte skrives inn. Metoden *_login2($codeInput)* sjekket så om koden brukeren skrev inn var lik den som var lagret i databasen og at

[6] `http://www.pats.no/?q=node/64`

det ikke var gått for lang tid mellom når koden ble generert og når brukeren skrev den inn. Om denne sjekken var vellykket så var brukeren autentisert i vår 2-faktor-autentiseringskomponent.

7.5 Integrering av autentiseringskomponenten

Med vår autentiseringskomponent på plass måtte vi sy sammen alle komponentene slik at de benyttet seg av den nye autentiseringen. Brukere skulle kun ha tilgang til innloggingssiden om de ikke var godkjente brukere. Dette innebærte at åpen kildekode-komponentene vi hadde brukt skulle være utilgjengelige for brukere som ikke var autentisert via vår komponent. For å få til dette trengte vi en måte å sjekke om en bruker som prøvde å få tilgang til nettsiden var logget inn eller ikke.

Siden HTTP er tilstandsløst[7] så er hver spørring til webserveren uavhengig av andre spørringer sendt tidligere. Dette betydde at vi trengte en måte å sjekke om brukere tidligere hadde autentisert seg og dermed skulle ha tilgang til nettsiden. For å realisere dette, benyttet vi oss av «sessions» slik at serveren kunne «huske» hvilke brukere som hadde logget inn.

For å kunne sjekke om en bruker var autentisert måtte hver bruker som logget seg inn få en session på serveren som indikerte om brukeren var logget inn og hvilke rettigheter brukeren skulle ha. Denne informasjonen ville lagres på serveren, mens brukeren ville få en cookie med en 32 sifret, tilfeldig generert hexadesimal ID som indikerte hvilken session på serveren som tilhørte brukeren. Vi laget metoden *_checkSession()* som sjekket om en bruker hadde en gyldig «session» og om den brukeren var logget inn. Hvis en bruker ikke var autentisert videresendte denne metoden brukeren til login-siden. Som en ekstra sikkerhetsmekanisme sjekker metoden også at brukeren ikke har vært inaktiv for lenge og at IP-adressen som brukes er den samme.

- *Benyttet «sessions» for å kontrollere om en bruker var autentisert.*

[7] Kanskje bedre uttrykt på engelsk ved Stateless [39]

7.5.1 Integrering med Wordpress

Wordpress er bygd opp slik at det skal være enkelt å forandre eller erstatte eksisterende metoder i den ved bruk av såkalte innstikk. Dette fungerer ved at det eksisterer et definert sett med metoder som er «pluggable», det vil si at de kan overstyres av egen kode. Når vi skulle integrere vår autentiseringskomponent med Wordpress var det derfor logisk å skrive et innstikk til Wordpress som gjorde dette.

Innstikket vårt, som vi kalte «auto_login()», la til ekstra funksjonalitet i metoden «init». Denne metoden ble kjørt før hver sidevisning i Wordpress og det var derfor den logiske plassen å sjekke om en bruker var autentisert. Innstikket brukte *_checkSession()* som sjekket om brukeren var autentisert og hadde lov å vise siden eller ei. Om denne sjekken feilet ble brukeren videresendt til login-siden.

Komponenten vi utviklet skulle være en universal innlogging og hvis en bruker var autentisert så skulle han være autentisert i alle komponenter i systemet. Siden denne sjekken ville gjøres for hver sidelasting måtte innstikket sjekke om brukeren var logget inn i Wordpress. Om det ikke var tilfelle ble brukernavnet hentet ut fra «session» og logget inn med Wordpress sine egne metoder.

På denne måten ville ikke en bruker få tilgang til Wordpress med mindre autentisering gjennom vår komponent var gjennomført.

- *Laget et innstikk for å bruke vår autentisering i Wordpress.*

7.5.2 Integrering med forum

Forumet vi hadde valgt å bruke, phpBB, viste seg å være den vanskeligste komponenten å integrere med autentiseringskomponenten. Selv om integreringen mellom Wordpress og phpBB ble ordnet av innstikket «WP-United», ble man ikke logget inn i phpBB når man logget inn i Wordpress. Dette betydde at vi måtte autentisere riktig bruker i forumet hvis 2-faktor-autentisering ble gjennomført. PhpBB hadde ikke støtte for innstikk slik som Wordpress og vi måtte benytte phpBB sine egne metoder for å autentisere brukere.

Vi forsøkte å bruke phpBB sine egne metoder fra vår autentiseringskomponent slik at når en bruker logget inn, så ble han også autentisert i forumet.

Det viste seg derimot at man ikke hadde tilgang til metodene utenfor phpBB, og vi kunne derfor ikke bruke disse fra vår autentiseringskomponent.

Vi forsøkte å modifisere autentiseringskoden til phpBB for å sjekke om en bruker var logget inn, men slik koden var organisert viste det seg vanskelig å gjennomføre. Da vi fant Simple:Press forumet, som lett lot seg integrere mot Wordpress og dermed også vår autentiseringskomponent, bestemte vi oss for å droppe phpBB og bruke dette i stedet.

- *phpBB var vanskelig å integrere med autentiseringskomponenten, og et bedre alternativ ble funnet.*

7.5.3 Integrering med WebCalendar

WebCalendar viste seg å være den modulen som var vanskeligst å modifisere for å støtte vår autentiseringskomponent. I motsetning til Wordpress hadde ikke WebCalendar noen støtte for innstikk, og det var komplisert å bruke WebCalendar-metodene utenfor WebCalendar-komponenten. Vi var derfor nødt til å modifisere kildekoden slik at autentiseringsmekanismen vår ble benyttet og ikke WebCalendar sin egen. Måten koden var strukturert på gjorde det vanskelig å sette seg inn i hvor denne funksjonaliteten måtte legges inn, men etter mye søking lokaliserte vi hvor autentisering ble håndtert i kalenderen.

Vi skulle gjerne hatt muligheten til å gjøre dette på en bedre måte enn å direkte modifisere kildekoden fordi når det kommer en oppdatering risikerer man at modifiseringene forsvinner. Løsningen var imidlertid tilfredsstillende fordi den fungerte slik den skulle.

- *Kalenderen støttet ikke innstikk slik som Wordpress, og kildekoden for autentisering i kalenderen måtte modifiseres.*

Kapittel 8

Resultater

I dette kapittelet vil jeg presentere resultatene fra den praktiske studien.

8.1 Implementering av sikkerhetskrav

Implementeringen av sikkerhetskravene presentert i Seksjon 5.3 ble gjort etter en grov prioritering av hva som var viktigst og hva som måtte være på plass først. Den reviderte kravlisten var ikke fullstendig når implementasjonen startet og ble til underveis i utviklingen.

Tabell 8.1 viser en oversikt over implementerte og ikke implementerte krav fra Tabell 5.3 samt en kommentar til kravet.

8.1.1 Autentiseringskrav

Krav 1 i Tabell 5.3 sier at systemet vårt skal overholde kravene stilt til nivå 4 autentisering. I følge Røstad kunne dette kravet virke litt ugjennomførlig siden ingen andre systemer i Norge i skrivende stund støttet en slik autentisering. Men kravet fra myndighetene ved behandling av helse- og personopplysninger sier at systemet må støtte nivå 4. Dette kravet ble selvfølgelig ikke realisert i vårt system, og ifølge Røstad hadde vi «sterk autentisering», men vi lå ikke på noe spesielt sikkerhetsnivå siden det var en del krav til infrastruktur rundt det å tilfredsstille et nivå.

Krav 2 i Tabell 5.3 beskriver at brukere ikke skal ha tilgang til annet enn en innloggingsside hvis man ikke er autentisert. Det er vår autentiseringskom-

Krav	Status	Kommentar
1	Delvis	Kravene som stilles til nivå 4 autentisering er ikke fult ut realisert av noen i Norge i skrivende stund.
2	Implementert	Brukere som ikke er autentisert ser kun en innloggingsside.
3	Implementert	Vår autentiseringskomponent sørget for dette.
4	Implementert	Brukere har ulike rettigheter basert på hvilken rolle de har i systemet.
5	Implementert	Administratorer kan fjerne autorisasjoner til enkeltbrukere.
6	Implementert	
7	Ikke implementert	Logging av hendelser ble prioritert hvis tiden tillot det, noe den ikke gjorde.
8	Ikke implementert	Logging av hendelser ble prioritert hvis tiden tillot det, noe den ikke gjorde.
9	Ikke implementert	Logging av hendelser ble prioritert hvis tiden tillot det, noe den ikke gjorde.
10	Ikke implementert	Logging av hendelser ble prioritert hvis tiden tillot det, noe den ikke gjorde.
11	Ikke implementert	Kravet kom sent i utviklingsprosessen og vi var tvunget til å prioritere andre krav, samtidig som vi antok at det antagelig kun var én konfigurasjon som skulle til.
12	Ikke implementert	Påminnelser med fritekst var implementert, men siden kravet kom sent i utviklingsprosessen hadde vi ikke tid til å forandre dette til forhåndsdefinerte valg.
13	Implementert	Det brukes SSL for å kommunisere mellom brukere og server.
14	Delvis implementert	Brukere kan endre passord, men det påtvinges ikke. Andre krav fikk prioritering
15	Ikke implementert	Mangel på erfaring og tid gjorde at dette kravet ikke ble prioritert.
16	Implementert	Passordfil saltes og «hashes» for å hindre at passord er i klartekst eller kan reverseres til klartekst og leses.

Tabell 8.1: Oversikt over implementerte krav

ponent som tilfredsstiller dette kravet, sjekker om en bruker er autentisert og videresender dem til innloggingssiden hvis den ikke er det.

Vi støtte på flere utfordringer i forbindelse med å realisere dette kravet. Vi var ikke kjent med autentiseringsmekanismer fra før av og det krevde en del analyse og lesing av dokumentasjon for å få et innblikk i hvordan man kunne gå frem og hvor vi skulle begynne. Vi måtte benytte oss av et rammeverk for å sende SMS, og integreringen av dette viste seg å være veldig vanskelig siden det var skrevet i Java, mens vår kode var i PHP. Det ble nødvendig å skrive om grensesnittet for å sende SMS til PHP, siden integreringen av Java mot PHP ikke ga noen resultater.

Vi måtte integrere autentiseringen med de andre komponentene, og dette resulterte i flere utfordringer. Integreringen mot Wordpress gikk forholdsvis greit siden man kunne bruke innstikk til å modifisere autentiseringen som ble brukt, men mot forumet phpBB og WebCalendar var det ikke så enkelt. I phpBB fikk vi ikke tilgang til noen av dens metoder i vår komponent og vi måtte modifisere autentiseringskoden. Dette var vanskelig og koden man måtte forandre var spredt utover flere filer og knyttet sammen med annen funksjonalitet. Vi fant et annet forum som erstattet phpBB siden vi ikke klarte å integrere det.

WebCalendar var også vanskelig og også der måtte vi modifisere autentiseringskoden for å benytte vår autentiseringskomponent. Autentiseringskoden her var litt mer oversiktlig, men det tok fremdeles mye prøving og feiling før vi kunne benytte egen autentisering.

Når alle komponentene krevde autentisering i vår komponent var kravet realisert og brukere så kun en innloggingsside hvis de ikke var logget inn.

8.1.2 Autoriseringskrav

Krav 4 i Tabell 5.3 var et av kravene vi hadde svært tidlig i prosessen, og det hadde kommet frem som et ønske om funksjonalitet på workshoppene vi hadde i fordypningsprosjektet, se Seksjon 2.1.2. Brukere skulle ha ulike roller med ulike rettigheter i systemet, for eksempel pasient og helsepersonell. Det viste seg å være en utfordring å få til rollebasert autorisering ved integrering av de forskjellige komponentene vi benyttet oss av, siden autorisasjonene en bruker har i de ulike komponentene blir håndtert av den aktuelle komponenten. Hvilke autorisasjoner en bruker skulle ha i en komponent måtte derfor settes i den aktuelle komponenten. Dette var problematisk, fordi det var ingen støtte i komponentene for å definere en felles rolle, noe som resulterte

i at vi selv måtte lage muligheten for å definere hvilken rolle en ny bruker skulle ha.

Vi utviklet funksjonalitet for å opprette nye brukere, hvor man skrev inn generell informasjon om brukeren, og i en nedtrekksmeny kunne man velge rollen en bruker skulle ha. Når brukeren ble opprettet, skulle den opprettes i alle komponenter, og autorisasjoner skulle settes basert på rollen som var valgt. Den eneste måten vi fant ut å gjøre dette på var å legge den nye brukeren til direkte i komponentenes databaser, og sette de riktige autorisasjonene basert på rollen. Dette var ikke lett og krevde mye testing for å forstå hvordan autorisasjon ble håndtert i de forskjellige komponentenes databaser, men kravet ble tilfredsstilt til slutt.

Krav 5 i Tabell 5.3 var allerede realisert i åpen kildekode-komponentene vi benyttet, og brukere med administratorrettigheter kunne fjerne rettigheter tildelt brukere. I forbindelse med implementasjonen av krav 5 la vi også til funksjonalitet for å slette brukere helt fra systemet. På samme måte som brukere ble lagt til i alle komponenter ved oppretting så ble de fjernet fra alle komponenter når de ble slettet. Det var relativt enkelt å fjerne brukere fra Wordpress, men vi måtte skrive om hele metoden for sletting av brukere i kalenderen for å fjerne alle spor av en bruker og dette var en stor jobb.

Brukere kunne sende private meldinger til hverandre, og disse hadde potensiale for å inneholde helse- og personopplysninger. **Krav 6** i Tabell 5.3 beskriver at det må være mekanismer som hindrer at uautoriserte brukere skal kunne se disse opplysningene. Det var derfor viktig at disse meldingene kun kunne ses av avsenderen og den tiltenkte mottageren. Vi benyttet Simple:Press sin funksjonalitet for private meldinger hvor hver bruker har en innboks der han kan se meldinger han har sendt og mottatt.

Vi la ikke til noen ytterligere sikring rundt private meldinger enn det som eksisterte i Simple:Press foruten at brukere måtte benytte vår autentiseringskomponent for å tilgang til systemet og dermed bli autorisert.

8.1.3 Logging

Kravene 7, 8, 9, og 10 i Tabell 5.3 som går på logging av aktivitet er ikke implementert i vårt system, mye grunnet mangel på tid. Det var en viss støtte for noen av kravene som var implementert i komponentene våre. Blant annet ble endring av informasjon publisert i Wordpress lagret i nye revisjoner slik at man kunne se forfatteren av alle revisjoner og tidligere revisjoner. Logging

av autorisert og uautorisert bruk var noe vi prioriterte å gjøre hvis vi hadde tid, men det viste det seg at vi ikke hadde.

8.1.4 Personvernkrav

Krav 11 i Tabell 5.3 ble til som en konsekvens av samtalene med Røstad og Røset. Det var opprinnelig meningen at brukere skulle kunne dele erfaringer de hadde med andre brukere av forumet og de skulle kunne legge ut opplysninger om deres helse hvis de ønsket det. Gjennom intervjuene, i Kapittel 4, kom det frem at dette ikke ville være en akseptabel løsning, og nettsiden skulle ikke tilrettelegge for at brukere kunne dele helse- og personopplysninger som kunne ses av alle de andre brukerne.

Dette kravet kom frem sent i utviklingsprosessen og frem til da var antagelsen at brukere skulle kunne ha mulighet til dette. For å tilfredsstille dette kravet ble det fremmet et forslag fra Røset hvor alle poster til forumet måtte godkjennes av en moderator før den ble synlig for alle brukere. Hvis posten inneholdt helse- eller personopplysninger skulle moderator slette posten eller fjerne opplysningene og så godkjenne den.

Jeg prioriterte å ferdigstille autentiseringskomponenten, siden det å stoppe alle poster for moderering antagelig kun var en konfigurasjon av forumrettigheter.

Krav 12 i Tabell 5.3 ble også til som en følge av samtalene med Røstad og Røset. Kalenderpåminnelser via SMS var et sterkt ønske fra brukerne, og brukere skulle kunne skrive inn påminnelser som de så fikk på SMS. Det kom frem i intervjuene at siden SMS er en usikret kanal, kunne brukere potensielt sende helse- og personopplysninger via SMS. Dette var ikke akseptabelt og måtte endres.

Kalenderpåminnelser var allerede implementert med støtte for fritekst-påminnelser ved å modifisere kildekoden slik at en SMS ble sendt i stedet for en e-post. For å tilfredsstille kravet fant jeg ut at predefinerte varslinger som ikke inneholdt helse- og personopplysninger var en mulig løsning. Grunnet tidsmangel ble dette ikke implementert.

8.1.5 Andre sikkerhetskrav

Krav 13 i Tabell 5.3 er realisert ved å kryptere kommunikasjonskanalen mellom klient og server ved bruk av SSL. Det var ingen stor utfordring å

implementere, og komponentene hadde innebygd støtte for bruk av SSL. Det var derfor smertefritt å få dem til å benytte seg av SSL. En utfordring vi støtte på, var å få vår egen autentiseringskomponent til å benytte seg av SSL. Det fungerte så lenge brukeren selv skrev HTTPS i adressefeltet, men hele meningen var å tvinge brukeren til å bruke SSL. Etter litt undersøking og prøving fikk vi rettet det slik at all trafikk til serveren ble tvunget til å benytte en kryptert kommunikasjonskanal.

Krav 14 i Tabell 5.3 som sier at brukere må endre passordet første gang de logger inn ble kun delvis implementert. Vi implementerte funksjonalitet for at brukere kunne endre passordet sitt, men de ble ikke tvunget til å endre passordet når de logget inn første gang. Det var prioriteringer av andre krav og tidsmangel som førte til at dette ikke ble gjort.

Krav 15 i Tabell 5.3 sier at inntrengere må bryte seg gjennom to uavhengige sikkerhetsbarrierer før de skal få tilgang til helse- og personopplysninger. Det stilles ikke noe videre krav til hva slags barrierer dette skal være, så lenge det er to, for eksempel brannmur, kryptering, sikker sone og så videre. Vi valgte å ikke prioritere dette kravet av to grunner: Vi hadde ikke mye erfaring med drifting av servere, altså oppsett av brannmur, sikker sone og lignende, og vi antok at det ville kreve endel tid å sette det opp, noe som ville gå utover implementasjonen av systemet.

Ved en eventuell testing av systemet antok vi at det ville være ekspertise på drifting og sikring av data som ville kunne sette dette opp raskere og bedre enn det vi ville.

Krav 16 i Tabell 5.3 sier at passordfil ikke skal lagres i klartekst, mens i kravlisten fra Helsedirektoratet står det at passordfil skal krypteres. Vi følte dette var motstridende med andre krav fremstilt av Helsedirektoratet siden de også sier at passord skal endres ved førstegangs innlogging slik at kun bruker kan vite passordet. Ved å kryptere passordet så vil man kunne dekryptere passordet dersom man har krypteringsnøkkelen, noe for eksempel en administrator vil kunne ha. For å sikre passordet valgte vi å salte og «hashe» passordet. Ved å gjøre dette vil man ikke kunne reversere og få ut passordet i klartekst.

Det var ingen spesielle utfordringer å realisere dette kravet, og det fantes mange gode eksempler på hvordan man kunne gjøre dette.

8.2 Utfordringer

I denne seksjonen vil jeg se på utfordringer knyttet til informasjonssikkerhet som jeg støtte på eller oppdaget eksisterte i løpet av prosjektets løp.

8.2.1 Krav

Når vi skulle implementere sikkerhetstiltak i tråd med de kravene som var stilt fra det offentlige til systemer som behandler helse- og personopplysninger, måtte jeg først finne ut hva disse kravene var. Det viste seg å være vanskelig å faktisk finne ut hva de spesifikke sikkerhetskravene var, og det krevde endel dokumentanalyse for å få formulert kravene.

Det var en stor hjelp å kunne konsultere med eksperter på området, og få deres synspunkter og tilbakemeldinger i forbindelse med spesifisering av kravene til informasjonssikkerhet.

8.2.2 Integrering

Det å integrere komponenter sammen til et helhetlig system var en stor utfordring. Slike komponenter er ikke alltid designet for å kunne integreres med andre, og avhengig av komponenten vil det kunne variere hvor lett eller vanskelig det er å integrere.

Som vi så med integreringen av åpen kildekode-komponentene var det svært lett å integrere Wordpress og phpBB, og vanskeligere å integrere WebCalendar. Siden både Wordpress og phpBB er to svært utbredte komponenter som er utstrakt brukt var det relativt smertefritt å integrere dem. WebCalendar er et mindre prosjekt og ikke så utbredt. Det var ikke tilrettelagt for noen integrering av disse komponentene så dette måtte vi gjøre selv.

Problemet med integrering av komponentene viste seg for fullt da vi skulle integrere vår egenutviklede autentiseringskomponent med resten av systemet. Siden Wordpress har en utstrakt støtte for innstikk var det den komponenten som var mest modifiserbar, og dette tillot oss å integrere vår autentiseringskomponent ved å lage et innstikk som for hver sidelastning sjekket om brukeren hadde autentisert seg i vår komponent. De andre komponentene hadde ingen støtte for innstikk, og var derfor ikke tilrettelagt for noen integrering av andre komponenter på lik linje med Wordpress. Dette ble tydelig når vi ikke fikk integrert phpBB med vår autentisering og endte opp med å finne et

alternativt forum. WebCalendar var også vanskelig å integrere med autentiseringen og vi måtte skrive om autentiseringskoden til kalenderen for å kunne benytte vår komponent i stedet.

8.2.3 Åpen kildekode

Den utstrakte bruken av åpen kildekode-komponenter i systemet er ikke uten utfordringer, selv om noen av utfordringene også ville eksistert hvis vi brukte lukket kildekode-komponenter eller COTS.

En utfordring ved å bruke nye åpen kildekode-komponenter er at de ikke nødvendigvis har blitt testet ordentlig gjennom «peer review». På den andre siden er det også en utfordring å bruke komponenter som ikke lenger vedlikeholdes og holdes oppdatert. Hvis komponenten er relativt ny risikerer man å bli en pionér som tar sjansen på ny software og finner feil, enten ved gjennomgang av koden eller ved at en sårbarhet blir utnyttet. Prosjekter som ikke lenger blir vedlikeholdt vil fremdeles kunne inneholde sårbarheter, og hvis det ikke er utviklere som finner og retter disse sårbarhetene så gir det en stor fordel til eventuelle angripere. Sårbarheter som eksisterer vil kunne utnyttes av angripere med mindre man selv kan fikse dem.

En annen risiko ved bruk av populære åpen kildekode-komponenter er at man er mer utsatt for angrep når det oppdages en sårbarhet, siden en angriper har mulighet til å angripe svært mange. En angriper kan ha muligheten til å automatisere angrep som utnytter en sårbarhet og rammer svært mange.

Det vil være viktig å holde komponentene oppdatert for ikke å være utsatt for angrep mot sårbarheter som er blitt fikset. I systemer hvor det brukes mange forskjellige komponenter, må alle disse holdes oppdatert hver for seg. Det kan være en utfordring å holde oversikten over om alle komponentene benytter nyeste versjon hvis det ikke finnes noen automatikk i oppdateringsprosessen.

8.2.4 Implementering

Det var en utfordring å realisere en felles innlogging over alle komponentene, og vår komponent måtte sørge for å logge brukeren inn i alle de andre komponentene når brukeren ble autentisert. For å få til dette fikk hver bruker en «session» lagret på serveren når de var autentisert og en «cookie» med en 32-sifret hexadesimal ID som indikerte hvilken «session» som var deres. For å kunne logge brukeren inn i de andre komponentene måtte vi lagre brukerens

informasjon i «session», slik at vi kunne hente ut brukernavn og passord og logge brukeren inn i de forskjellige komponentene.

Selv om informasjonen ikke ligger lagret hos brukeren var ikke dette en ideell løsning, men som Røstad sa, så er det nok «baksiden med å benytte seg av komponenter skrevet av andre». For å ytterligere sikre at ingen kunne stjele en annen brukers «session» la vi også til sjekker slik at brukeren måtte ha samme IP-adresse som når han først ble autentisert.

Kapittel 9

Diskusjon

Dette kapittelet diskuterer funnene fra denne studien ut ifra problemstillingen og forskningsspørsmålene. Videre vil resultatenes kvalitet diskuteres med tanke på validitet, generaliserbarhet og realibilitet. Til slutt vil jeg vurdere nytteverdien til forskningsmetodene som er benyttet i denne studien.

9.1 Problemstilling og forskningsspørsmål

I denne seksjonen vil jeg diskutere og besvare våre forskningsspørsmål som ble presentert i Seksjon 1.3.1.

9.1.1 Forskningsspørsmål 1

Hvilke krav stilles til informasjonssikkerhet i systemer som behandler helse- og personopplysninger?

For å svare på dette spørsmålet utførte jeg intervjuer av to sikkerhetseksperter. Ved å snakke med eksperter kunne jeg få en bedre forståelse av hvilke regler og retningslinjer slike systemer måtte forholde seg til. Jeg gjorde også en grundig litteraturanalyse for å avklare hvilke krav vårt system måtte forholde seg til.

Det viste seg å være vanskelig å finne ut hva de spesifikke kravene til informasjonssikkerhet for systemer som behandler helse- og personopplysninger faktisk var. Helsedirektoratet har laget en norm for informasjonssikkerhet i helsesektoren [30], og denne var til god hjelp for å få en klarhet i kravene til

systemet, selv om mye av informasjonen ikke direkte beskrev hvilke krav som ble stilt og det måtte derfor på mange punkter tolkes hva de spesifikke kravene skulle være.

For å komme frem til den endelige listen med sikkerhetskrav som skulle svare på dette forskningsspørsmålet utviklet jeg kravlisten kontinuerlig og parallelt med utviklingen av systemet. Jeg startet prosjektet med en liten liste med krav jeg antok systemet måtte tilfredsstille og reviderte denne etter hvert. Det første utkastet til sikkerhetskrav er gjengitt i Tabell 5.1.

For å få en bedre forståelse av kravene som stilles til behandling av helse- og personopplysninger analyserte jeg dokumentasjon fra det offentlige om krav til systemer som behandler sensitiv informasjon [17, 24, 27, 28, 30]. Jeg satte sammen en oversikt over kravene som stilles til slike systemer og gjorde en vurdering av relevansen til kravene i forhold til vårt system. Denne oversikten er gjengitt i Tabell 5.2.

For å gi meg en innsikt i hva kravene innebærte og hvilke som var relevante for vårt system var kompetansen til Lillian Røstad og Øyvind Røset uvurderlig. Ved å intervjue disse ekspertene fikk jeg et bedre innblikk i informasjonssikkerhet og sensitive opplysninger, og kravene til systemer som behandler slike opplysninger. Ved å få tilbakemeldinger på vårt system kunne jeg også få en bekreftelse, eller avkreftelse, på om løsningene våre var tilfredsstillende, selv om dette kun var deres meninger og ikke en offisiell godkjenning av systemet.

Med deres innsikt og kompetanse fikk jeg en dypere innsikt i hvilke krav som stilles til systemer av denne typen og jeg reviderte kravlisten vår og kom frem til den endelige kravlisten for informasjonssikkerheten til vårt system.

Denne kravlisten, som er svaret på forskningsspørsmål 1, er gjengitt i Tabell 5.3.

9.1.2 Forskningsspørsmål 2

Hvilke utfordringer er knyttet til å realisere disse kravene i prosjekter med utstrakt bruk av åpen kildekode, og hvilke tiltak kan gjøres for å redusere disse?

For å svare på dette spørsmålet brukte jeg en praktisk studie der jeg forsøkte å implementere sikkerhetskravene som er stilt til slike systemer for å selv oppleve utfordringene et slikt prosjekt kan føre med seg. Jeg utførte også en litteraturanalyse for å være klar over utfordringer jeg kunne møte på, og bevist på utfordringer jeg ikke erfarte.

Utfordringene jeg fant kan tematiseres i forhold til:

- Integrering av eksisterende åpen kildekode-komponenter. Seksjon 9.1.2.1
- Integrering av egenutviklede komponenter mot de eksisterende åpen kildekode-komponentene. Seksjon 9.1.2.2
- Nye vs. velutprøvde åpen kildekode-komponenter. Seksjon 9.1.2.3
- Bruk av åpen kildekode-komponenter som ikke lenger vedlikeholdes. Seksjon 9.1.2.4
- Populære og utbredte åpen kildekode-komponenter. Seksjon 9.1.2.5

9.1.2.1 Integrering av åpen kildekode-komponenter

En av hovedutfordringene jeg opplevde i prosjektet var integrasjonen av de forskjellige komponentene, både integreringen av åpen kildekode-komponentene sammen til et helhetlig system og integreringen av vår egen sikkerhetskomponent.

Ved bruk av åpen kildekode-komponenter i systemutviklingsprosjekter er det ikke noen selvfølge at komponentene er tilrettelagt for integrering med andre komponenter, og hvis slik integrering ikke er tilrettelagt i det hele tatt under utviklingen av komponenten kan det være en møysommelig prosess å tilpasse komponentene til hverandre og tilrettelegge dem for integrering.

Av komponentene vi tok i bruk i vårt system var Wordpress mest tilrettelagt for integrering av andre komponenter ved bruk av innstikk, og utfordringen framfor oss var derfor å integrere de andre åpen kildekode-komponentene med denne. Forumet phpBB, som er et svært utbredt forum, hadde allerede fått utviklet et innstikk til Wordpress som integrerte de to på en sømløs måte. Kalenderen derimot var det ingen som hadde ingen utviklet noe slikt for, og vi måtte derfor ta oss av integreringen selv; noe som viste seg å være en vanskelig jobb.

Tanken var å integrere WebCalendar i Wordpress ved å bruke innstikk på den samme måten som phpBB ble integrert. Det viste seg derimot å være en formidabel utfordring å realisere. Innstikket som integrerer Wordpress og phpBB var ganske avansert, og er faktisk et eget åpen kildekode-prosjekt. Bruken av åpen kildekode betyr at man mye raskere kan få funksjonalitet på plass, men størrelsen på koden til komponentene gjør det vanskelig å få en god nok innsikt til å forstå hvordan det hele henger sammen. Dermed ble det vanskelig å få til en god integrering av WebCalendar og Wordpress på samme måte som phpBB var integrert.

Det vil ikke alltid finnes komponenter som lar seg integrere med hverandre uten noen konfigurasjon eller modifisering, men ofte kan det være flere komponenter med tilsvarende funksjonalitet og som varierer i modifiserbarhet.

9.1.2.2 Integrering av egenutviklet komponent

Integreringen av vår egen sikkerhetskomponent var heller ikke uten utfordringer. Integreringen mot Wordpress slik at den benyttet vår autentisering var relativt grei og ble gjennomført ved å lage et innstikk som benyttet seg av vår komponent. På den annen side viste problemene seg ved integrering mot phpBB og WebCalendar.

Et viktig poeng når man utvikler et system basert på ferdigkomponenter oppsummeres fint med uttrykket «Kill your darlings» [40]. Betydningen av dette er enkel; ikke bli for knyttet til koden din. Dette er også relevant i utvikling som benytter ferdigkomponenter i den grad man ikke vil gi slipp på en komponent fordi det er lagt mye tid og ressurser i den. Dette kan fort bli en fallgruve hvor man ignorerer alternative og kanskje bedre komponenter.

Vi opplevde selv dette dilemmaet i vårt valg av forum. Forumet vårt, phpBB, var integrert med Wordpress ved hjelp av et innstikk. Men selv om Wordpress benyttet seg av vår komponent betydde ikke det at phpBB gjorde det samme. Vi måtte derfor finne en måte å integrere forumet slik at brukeren ble logget inn når han ble autentisert i vår komponent. Dette var en vanskelig oppgave å få til, og vi så etter andre, enklere løsninger og fant Simple:Press. Denne komponenten var et innstikk til Wordpress og integrerte seg lett med vår komponent; dermed forkastet vi bruken av phpBB.

Det burde vært gjort en grundigere undersøkelse av hvilke alternativer vi hadde fra starten av, slik at vi hadde spart endel tid. Det ville nok vært mulig å få phpBB til å benytte vår autentisering, men sett i forhold til tiden vi hadde til rådighet var det mer fornuftig å ta i bruk Simple:Press.

Kalenderen, i likhet med phpBB, hadde ingen støtte for innstikk eller lignende funksjonalitet og siden vi ikke fant en tilfredsstillende komponent hvor det var lettere å integrere vår autentisering, var vi nødt å modifisere kildekoden for autentisering i WebCalendar slik at den benyttet vår komponent.

En annen utfordring ved å integrere forskjellige komponenter sammen er at det ofte er såpass mye kode at man har ikke oversikt over all funksjonalitet, og integrering kan derfor føre til uforutsette konsekvenser.

9.1.2.3 Nye åpen kildekode-komponenter

Prosjekter som tar i bruk relativt nye åpen kildekode-komponenter som ikke har vært lenge i bruk møter en utfordring siden slike komponenter ikke nødvendigvis har blitt testet ordentlig gjennom en «peer review»-prosess. I slike tilfeller vil komponenten potensielt kunne inneholde mange feil som ikke er blitt oppdaget enda. Ved å være noen av de første som tar i bruk slike systemer risikerer man å være den som finner endel nye feil i systemet; forhåpentligvis ved gjennomgang av koden, men i verste fall ved at systemet blir utsatt for et angrep [6].

Det bør legges til grunne en god risikovurdering før man tar i bruk et slikt system, og en gjennomgang av kildekoden til systemet kan være en god idé hvis man velger å ta i bruk svært nye åpen kildekode-komponenter. Det bør også ses på andre komponenter som innfrir kravene og som har eksistert lenger.

9.1.2.4 Åpen kildekode-komponenter som ikke vedlikeholdes

På motsatt side av skalaen er en annen utfordring ved å bruke åpen kildekode som ikke lenger blir vedlikeholdt av utviklere. I en slik komponent vil ikke sårbarheter som oppdages bli fikset med mindre man gjør det selv. Og siden det ikke blir vedlikeholdt så vil sårbarheter som oppdages antagelig bli oppdaget av mulige angripere som ikke melder fra om sårbarheten, men heller utnytter den. Et slikt scenario gir angripere en fordel siden de da er de eneste som kikker på koden.

Også i en slik situasjon bør det utføres en risikovurdering av bruken av systemet og en vurdering av alternative komponenter som tilfredsstiller kravene og som fremdeles har et aktivt utviklermiljø.

9.1.2.5 Utbredte og populære åpen kildekode-komponenter

Bruken av populære åpen kildekode-prosjekter kan også være en utfordring i seg selv på grunn av den utstrakte bruken av dem. Siden det er svært mange som bruker samme system er antallet potensielle ofre for et angrep svært stort og dermed også et attraktivt mål. Når en sårbarhet oppdages vil, litt avhengig av hva slags sårbarhet som er funnet, angripere kunne automatisere angrepet til å ramme flest mulig som benytter seg av den spesifikke åpen kildekode-komponenten.

For å redusere denne utfordringen vil det være svært viktig å holde komponenten oppdatert med de siste fiksene fra utviklere. Å benytte seg av en gammel versjon fordi den nye ikke er kompatibel med andre komponenter eller modifikasjoner som er gjort, vil innebære en risiko for at man er sårbar for angrep som har blitt fikset.

9.2 Resultatkvalitet

I denne seksjonen vil jeg diskutere kvaliteten av resultatene som kom frem i den praktiske studien med hensyn på validitet, generaliserbarhet og reliabilitet.

9.2.1 Validitet

Validiteten sier noe om i hvilken grad resultatene man har oppnådd er som en følge av metodene man har brukt og om konklusjonene basert på resultatene er gyldige. Oates [32] definerer validitet som at hypotese A er den riktige årsaken til handling B, og er ikke påvirket av andre faktorer.

9.2.1.1 Forskningsspørsmål 1.

Jeg undersøkte dokumentasjon produsert som veiledere for informasjonssikkerhet i helsesektoren for å formulere våre spesifikke krav til sikkerhet i vårt system [30]. Et spørsmål om validiteten av kravene er om jeg har forstått dokumentasjonen slik som det var tenkt av forfatterne. På bakgrunn av at jeg har presentert mine sikkerhetskrav til to sikkerhetseksperter i mine intervjuer og fått deres synspunkter og kommentarer på dem anser jeg mine antagelser som fornuftige.

Angående intervjuene mine er det noen punkter som kan påvirke validiteten. Det første er om intervjuobjektene er kvalifiserte til å kunne svare på mine spørsmål, og følgelig om svarene deres er pålitelige. Det andre er om intervjuobjektene har forstått mine spørsmål slik jeg mente å stille dem. Altså om svaret de gir harmonerer med spørsmålet jeg stilte.

Forutsetningene til de to personene som ble intervjuet tror jeg ikke påvirker validiteten til studien og begge har lang fartstid og gode kvalifikasjoner innenfor informasjonssikkerhet. Det er vanskelig å si med sikkerhet om spørsmålene

mine har blitt oppfattet slik jeg tenkte, men noen store misforståelser har jeg ikke grunn til å tro eksisterer.

9.2.1.2 Forskningsspørsmål 2.

Komponentene vi valgte å bruke i utviklingen av systemet er ikke nødvendigvis representative for alle åpen kildekode-komponenter. Det er tydelige forskjeller i arkitektur, kodekvalitet og modifiserbarhet i de forskjellige komponentene og jeg kan vanskelig si at komponentene som er valgt er representative for åpen kildekode. Det er verdt å trekke frem at selv om en komponent er åpen kildekode, betyr ikke dette mye annet enn at kildekoden er tilgjengelig for lesing og modifisering, og derfor vil det uansett være svært vanskelig å finne komponenter som kan sies å være representative for fri programvare generelt.

Validiteten til utfordringene jeg har oppdaget underveis i studien kan påvirkes av det faktum at jeg er student og derfor ikke har veldig mye erfaring sammenlignet med andre. Mer erfarne utviklere ville muligens hatt en annen tilnærming til oppgaven og kunne dermed oppnådd forskjellige resultater enn det vi gjorde.

9.2.2 Generaliserbarhet

Generaliserbarhet sier noe om hvorvidt resultatene fra en studie eller et prosjekt kan generaliseres slik at de også er valide for andre prosjekter [32].

9.2.2.1 Forskningsspørsmål 1.

Som et ledd i innhentingen av sikkerhetskravene, intervjuet jeg to sikkerhetseksperter beskrevet i Kapittel 4. Det var overensstemmelse mellom dem på like spørsmål og jeg har grunn til å tro at svarene fra intervjuobjektene er generaliserbare. Jeg kunne likevel tenkt meg et intervju med Datatilsynet også for å ytterligere styrke mitt resultatgrunnlag.

Kravene til sikkerhet som er kommet frem i dette prosjektet er svært spesifikke for systemer som behandler helse- og personopplysninger, samtidig som de ikke er fullstendige da jeg kun har sett på krav som implementeres i vårt system. Slik sett er kravene presentert i Seksjon 5.3 ikke så generaliserbare for prosjekter som bruker åpen kildekode-komponenter, men kan brukes som retningslinjer i prosjekter som skal behandle sensitiv informasjon.

9.2.2.2 Forskningsspørsmål 2.

De resultater og konklusjoner som er oppnådd gjennom denne studien har visse begrensninger som kan gjøre de mindre generaliserbare. Resultatene stammer fra mine erfaringer med de komponentene som er brukt i studien og er derfor karakterisert av egenskapene til enkeltkomponentene. Min tilnærming til oppgaven og utviklingsprosessen vi har gjennomført er heller ikke generaliserbar, og det er rimelig å anta at andre utviklere med et annet erfaringsgrunnlag ville angrepet problemet på en anderledes måte.

Likevel er det nærliggende å tro at mine resultater og erfaringer jeg har gjort gjennom denne studien vil være gode retningslinjer som jeg tror kan generaliseres til andre prosjekter med utstrakt bruk av åpen kildekode. Vår erfaring med integrering av forskjellige komponenter og problemene vi støtte på for å integrere dem på en god måte, understreker viktigheten av å gjøre grundige undersøkelser av komponentene man ønsker å bruke, og hvilke muligheter og begrensninger de fører med seg. Tiltakene jeg har identifisert for å redusere utfordringene ved slik bruk av åpen kildekode vil kunne være gyldige for andre prosjekter med lignende bruk av åpen kildekode.

9.2.3 Reliabilitet

Reliabiliteten sier noe om hvorvidt man ville fått de samme resultatene dersom man gjentok studien eller prosjektet flere ganger med de samme omstendighetene [32].

9.2.3.1 Forskningsspørsmål 1.

For innhentingen av kravene tror jeg reliabiliteten er god fordi eksperter fra to forskjellige organisasjoner, med forskjellig bagrunn, er enige. Skulle jeg eller noen andre utført det en gang til med like omstendigheter mener jeg at kravene til sikkerhet ville blitt ganske like.

9.2.3.2 Forskningsspørsmål 2.

Når jeg ser på hvilke utfordringer jeg støtte på gjennom studien var dette svært avhengig av hvilke komponenter som ble valgt. Det var ingen av oss som hadde mye erfaring med noen av komponentene og hadde vi utført studien

igjen er det gode muligheter for at vi kunne endt opp med å bruke andre komponenter og dermed fått andre resultater.

Hvis noen andre skulle utført studien med de samme omstendighetene som oss ville det være sannsynlig at de valgte andre komponenter enn oss, spesielt hvis noen hadde erfaring med for eksempel et annet publiseringsverktøy enn Wordpress. Den endelige løsningen og utfordringene som ble oppdaget ville da kunne blitt radikalt anderledes.

9.3 Metodediskusjon

I denne seksjonen vil jeg vurdere gjennomføringen og nytteverdien av forskningsmetodene jeg benyttet i studien. Hovedforskningsmetoden i dette prosjektet var utviklingen av et egenomsorgssystem for forskningsprosjektet MOBESITY basert på åpen kildekode-komponenter.

9.3.1 Intervju

Intervjuene av de to sikkerhetsekspertene fungerte bra for å få en klarhet i kravene som er stilt til systemer som behandler helse- og personopplysninger. Jeg fikk tilbakemeldinger på valg jeg hadde gjort og funksjonalitet vi hadde implementert og dette bidro til å forme det endelige systemet. Jeg fikk innsikt i kravene som ble stilt til informasjonssikkerhet rundt helse- og personopplysninger og hva kravene faktisk innebærte. Det ble også tid til diskusjoner under intervjuene og dette var svært nyttig for å få en forståelse av hva kravene faktisk innebærte for vårt system og sikringen av informasjon.

På denne måten fungerte intervjuene som et verktøy for å kartlegge sikkerhetskravene som var relevante for vårt system.

Den opprinnelige planen var at et av intervjuene skulle være med Datatilsynet for å presentere en grundig beskrivelse av systemet og sikkerhetsarkitekturen, og ha en grundig gjennomgang av sikkerhetsmekanismene i systemet for å kunne få en godkjenning til utprøvning av systemet på en begrenset brukergruppe. Omstendigheter utenfor min kontroll gjorde at dette møtet ikke kunne gjennomføres i tidsrommet jeg hadde tilgjengelig.

9.3.2 Systemutviklingscase

Gjennom systemutviklingscaset fik jeg verdifull erfaring med integrering av åpen kildekode-komponenter til et helhetlig system. Utviklingen av en autentiseringskomponent og modifiseringen av åpen kildekode-komponentene for å benytte den nye autentiseringen bydde på mange utfordringer som testet min kompetanse og ga meg ny innsikt.

Vi utviklet systemet i iterasjoner med ukentlige møter og tilbakemeldinger fra veileder på progresjon. Første prioritet var å få opp et fungerende system med alle de åpne kildekode-komponentene integrert sammen til en helhet. Med åpen kildekode-komponentene integrert startet vi arbeidet med vår egen komponent for å tilfredsstille sikkerhetskravene systemet hadde.

Utviklingsmodellen fungerte godt og vi realiserte nesten alle de funksjonelle kravene som var stilt til systemet og mange av sikkerhetskravene. Vi holdt oss til planen og ferdigstilte systemet til planlagt tid.

Utviklingen av systemet var nødvendig for kunne gi meg et bilde av hvilke sikkerhetsutfordringer som eksisterte i prosjekter med utstrakt bruk av åpen kildekode som behandlet helse- og personopplysninger.

Kapittel 10

Konklusjon

I dette kapittelet vil jeg konkludere svarene til min problemstilling og mine forskningsspørsmål og se på arbeidet som kan gjøres videre med prosjektet og systemet.

10.1 Problemstilling

Problemstillingen til dette prosjektet var å se på: *Hvilke krav som stilles til systemer som behandler helse- og personopplysninger og hvilke utfordringer som eksisterer i forbindelse med å implementere kravene i prosjekter med utstrakt bruk av åpen kildekode.*

For å kunne svare på min problemstilling utførte jeg en praktisk studie hvor jeg utviklet et egenomsorgssystem for overvektspasienter ved St. Olavs Hospital med utstrakt bruk av åpen kildekode-komponenter.

10.1.1 Forskningsspørsmål 1

Hvilke krav stilles til informasjonssikkerhet i systemer som behandler helse- og personopplysninger?

For å kartlegge hvilke krav som stilles til behandling av helse- og personopplysninger gjennomførte jeg en litteraturanalyse av dokumentasjon, hovedsakelig utgitt av det offentlige som beskrev de relevante krav til informasjonssikkerhet basert på gjeldende lovgivning. For å få et bedre innblikk i de gjel-

dende krav og konsekvensene de hadde for systemer som skulle tilfredsstille kravene, samt diskutere vår tilnærming til implementeringen av kravene i vårt system, intervjuet jeg to sikkerhetseksperter; Lillian Røstad og Øyvind Røset.

Kravene jeg fant til sikkerhet gjennom min litteraturanalyse er gjengitt i Tabell 5.2, og ved hjelp av intervjuene med eksperter og drøfting av relevans kom jeg frem til den endelige kravlisten til sikkerhet for vårt system. Denne er gjengitt i Tabell 5.3. Kravlisten fokuserer på sikkerhetskrav som må implementeres i et system, mens krav som stilles til infrastruktur og drifting anses som utenfor prosjektets omfang.

Kravene jeg har funnet er ganske spesifikke for vårt system, men jeg føler likevel at de kan brukes som utgangspunkt til å utforme retningslinjer for andre prosjekter som behandler sensitive opplysninger.

10.1.2 Forskningsspørsmål 2

Hvilke utfordringer er knyttet til å realisere disse kravene i prosjekter med utstrakt bruk av åpen kildekode, og hvilke tiltak kan gjøres for å redusere disse?

Gjennom mine erfaringer med utviklingen av systemet har jeg identifisert flere utfordringer knyttet til å realisere kravene til sikkerhet i prosjekter som benytter seg av åpen kildekode.

Den største utfordringen ved å realisere disse kravene var integreringen av komponentene til et helhetlig system, og integreringen av vår egen komponent i dette systemet. Det viste seg å være en svært varierende grad av modifiserbarhet i komponentene vi benyttet, noe som betydde at noen komponenter lett lot seg integrere med hverandre og andre var vanskelige eller på grensen til umulig med tanke på tiden vi hadde til rådighet.

Ved utvikling slik som dette bør det være fokus på å gjøre en grundig undersøkelse av komponentene man vil benytte seg av, og få en god oversikt over hvilke muligheter og ikke minst begrensninger disse komponentene pålegger systemet. Det vil være god praksis å se på lignende komponenter og sammenligne deres egenskaper for å kunne velge de komponentene som passer best for systemet man skal lage. Samtidig bør ikke utviklere bli for knyttet til komponentene som benyttes, og hvis problemer oppstår eller andre komponenter viser seg å være bedre, er det viktig å kunne gi slipp på en komponent.

Det bør være fokus på integrering av alle komponenter tidlig i utviklingsfasen slik at problemer som oppstår som en konsekvens av integrering avdekkes tidlig. Det vil være naturlig å bruke en iterativ utviklingsmodell som tillater fleksible krav, slik at muligheter og begrensninger som komponentene pålegger systemet kan påvirke kravutformingen.

Svært nye åpen kildekode-prosjekter som ikke nødvendigvis er blitt ordentlig testet og gamle prosjekter som ikke lenger vedlikeholdes innebærer en risiko å ta i bruk i systemer hvor sikkerhet er såpass sentralt. Det vil være en utfordring å ta i bruk slike komponenter, spesielt komponenter som ikke vedlikeholdes siden man selv må fikse eventuelle sikkerhetshull man kan oppdage eksisterer.

Det bør ligge til grunne en grundig risikoanalyse før man tar i bruk slike komponenter som forsvarer bruken av disse kontra aktive åpen kildekodeprosjekter.

Utfordringene og tiltakene jeg har identifisert vil kunne bidra til å minimere sikkerhetsproblemer i utviklingsprosjekter basert på utstrakt bruk av fri programvare.

10.2 Videre arbeid

Videre i dette prosjektet bør det gjøres en fullstendig testing av sikkerheten i systemet for å identifisere feil og sårbarheter som eksisterer. En slik testing bør gjennomføres av personer med ekspertise innenfor denne type testing for å sikre mest mulig valide resultater.

Det bør være en presentasjon av systemet for Datatilsynet og få deres synspunkter på systemet som helhet og løsningene som er valgt. Videre bør systemet godkjennes av Datatilsynet for uttesting på en begrenset pasientgruppe som er neste steg i Anita Das sin PhD.

Bibliografi

[1] Wikipedia. Open source. http://en.wikipedia.org/w/index.php?title=Open_source&oldid=369676537, 2010. Sist besøkt 20. Mai 2010.

[2] Redpill Linpro. Open source. http://www.redpill-linpro.no/Om-oss, 2010.

[3] Friprogsenteret. Friprog. http://www.friprog.no/friprogsenteret/, 2010.

[4] Anita Das. Kravspesifikasjon mobesity. Part of Ph.D., NTNU, 2009.

[5] E. Raymond. The cathedral and the bazaar. *Knowledge, Technology & Policy*, 12(3):23–49, 1999.

[6] C. Payne. On the security of open source software. *Information Systems Journal*, 12(1):61–78, 2002.

[7] J.H. Hoepman and B. Jacobs. Increased security through open source. *Communications of the ACM*, 50(1):83, 2007.

[8] B. Witten, C. Landwehr, M. Caloyannides, and A. DARPA. Does open source improve system security? *IEEE Software*, 18(5):57–61, 2001.

[9] John Lettice. German armed forces ban ms software, citing nsa snooping. *The Register*, 2001.

[10] N. McAllister. The Spy Who Hacked Me: Will Open Source Be The Hero Of International Security? *SF Gate*, 15, 2001.

[11] Brett Smith, Free Software Foundation. A quick guide to gplv3. http://www.gnu.org/licenses/quick-guide-gplv3.html. Sist besøkt 26. April, 2010.

[12] Eclipse Foundation. Eclipse public license - v 1.0. http://www.eclipse.org/legal/epl-v10.html, 2004.

[13] G. Moody. The greatest OS that (n) ever was. *Wired Magazine*, 1997.

[14] R. Anderson. Security in open versus closed systems–the dance of Boltzmann, Coase and Moore. *at Open Source Software Economics*, pages 127–142, 2002.

[15] E. Levy. Wide open source. *SecurityFocus. com. Electronic document. http://www.securityfocus.com/news/19*.

[16] K. Thompson. Reflections on trusting trust. In *ACM Turing award lectures*, page 1983. ACM, 2007.

[17] administrasjons-og kirkedepartementet Fornyings. 4 sikkerhetsnivåer for autentisering og uavviselighet. `http://www.regjeringen.no/nb/dep/fad/dok/lover-og-regler/retningslinjer/2008/rammeverk-for-autentisering-og-uavviseli.html?id=505958`, 2008.

[18] Datatilsynet. Sikkerhetsbestemmelser. `http://datatilsynet.no/upload/Dokumenter/infosik/veiledere/SV100_00.pdf`.

[19] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 16. `http://www.helsedirektoratet.no/normen/faktaark/alle/16_etablering_av_l_sning_for_meldingskommunikasjon_663584`, 2010. Sist besøkt 23. Juni 2010.

[20] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 33. `http://www.helsedirektoratet.no/normen/faktaark/alle/33_bruk_av_e_post__663364`, 2006. Sist besøkt 23. Juni 2010.

[21] Wikipedia. Ebxml. `http://en.wikipedia.org/w/index.php?title=EbXML&oldid=369157028`, 2010.

[22] Harald Jørgensen. Avvikling av trygd-helsepostkassen 1.juni. `http://www.helsedirektoratet.no/samspill/fagnytt/avvikling_av_trygd_helsepostkassen_1_juni__407814`. Sist besøkt 4. Desember 2009.

[23] Arnstein Vestad. Veiledning for innføring av ebxml og pki i helseforetak. `http://kith.no/upload/1601/R12-05Veiledning-ebXML-PKI.pdf`, 2005.

[24] Datatilsynet. Kryptering. `http://www.datatilsynet.no/templates/article____889.aspx`, 2001. Sist besøkt 4. Desember 2009.

[25] Wikipedia. Brute force attack. http://en.wikipedia.org/w/index.php?title=Brute_force_attack&oldid=329487002. Sist besøkt 5. Desember 2009.

[26] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 14. http://www.helsedirektoratet.no/normen/faktaark/alle/14_tilgangsstyring__663594, 2006. Sist besøkt 23. Juni 2010.

[27] A. Vestad and M. Alsaker. Sikkerhet i webløsninger - autentisering og tilgangskontroll. 2003.

[28] T. Huston. Security issues for implementation of e-medical records. 2001.

[29] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 10. http://www.helsedirektoratet.no/normen/faktaark/alle/10_bruk_av_databehandler__ekstern_driftsenhet__663174, 2006. Sist besøkt 23. Juni 2010.

[30] Helsedirektoratet. Norm for informasjonssikkerhet. http://www.helsedirektoratet.no/normen, 2010.

[31] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 17. http://www.helsedirektoratet.no/normen/faktaark/alle/17_fysisk_sikring_av_omr_der_og_utstyr__663574, 2010. Sist besøkt 23. Juni 2010.

[32] B.J. Oates. *Researching information systems and computing*. Sage Publications Ltd, 2006.

[33] G. McGraw. *Software security: building security in*. Addison-Wesley Professional, 2006.

[34] Datatilsynet. Sikkerhetsbestemmelsene i personopplysningsforskriften med kommentar. http://www.datatilsynet.no/upload/Dokumenter/infosik/veiledere/SV100_00.pdf, 2000.

[35] Justis og politidepartementet. personopplysningsloven. http://www.lovdata.no/all/hl-20000414-031.html, 2001.

[36] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 42. http://www.helsedirektoratet.no/normen/faktaark/alle/42_bruk_av_sms_i_pasientkontakt_versjon_1_0_663294, 2010. Sist besøkt 19. Mai 2010.

[37] Sosial- og helsedirektoratet. Norm for informasjonssikkerhet i helsesektoren, faktaark 38. http://www.helsedirektoratet.no/normen/faktaark/tekniske/38_sikkerhetskrav_for_systemer_663324, 2010. Sist besøkt 21. Mai 2010.

[38] k5n. Webcalendar. http://www.k5n.us/webcalendar.php, 2010. Sist besøkt 03. Mai, 2010.

[39] W3C. W3c http 1.1 specification. http://www.w3.org/Protocols/rfc2616/rfc2616.html, 1999.

[40] Jay Fields. Kill your darlings. http://blog.jayfields.com/2009/03/kill-your-darlings.html, 2009.