PROSJEKTOPPGAVE

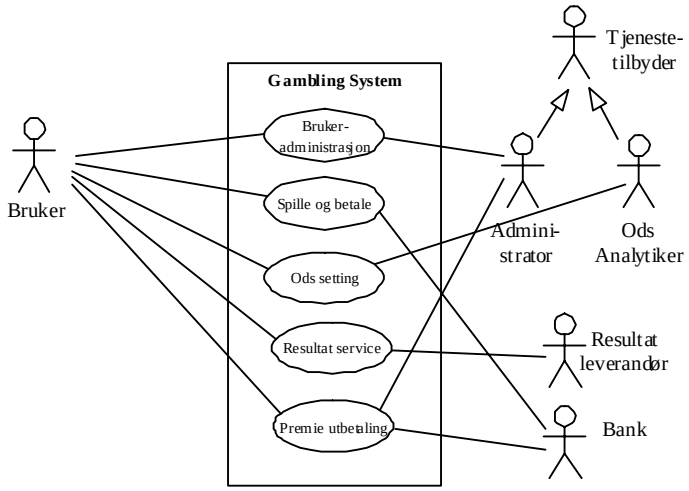
SYSTEMUTVIKLING OG SIKKERHETSARKITEKTUR

Av:
Ole Kasper Olsen (Leder)
Roar Sollie
Anders Wiehe
Morten Sporild
Fredrik Skarderud
Torkjel Søndrol
Ole Martin Dahl

Forord

Dette er en obligatorisk oppgave i faget "systemutvikling og sikkerhetsarkitekturer". I oppgaven benytter vi RM ODP [ISO10746] med UML-notasjon [Booch99] i relevante modeller og NS 5814 [NS5814] for risikoanalyse. Oppgaven skal være en innføring i iterativ, arkitekturbasert systemutvikling og risikoanalyse.

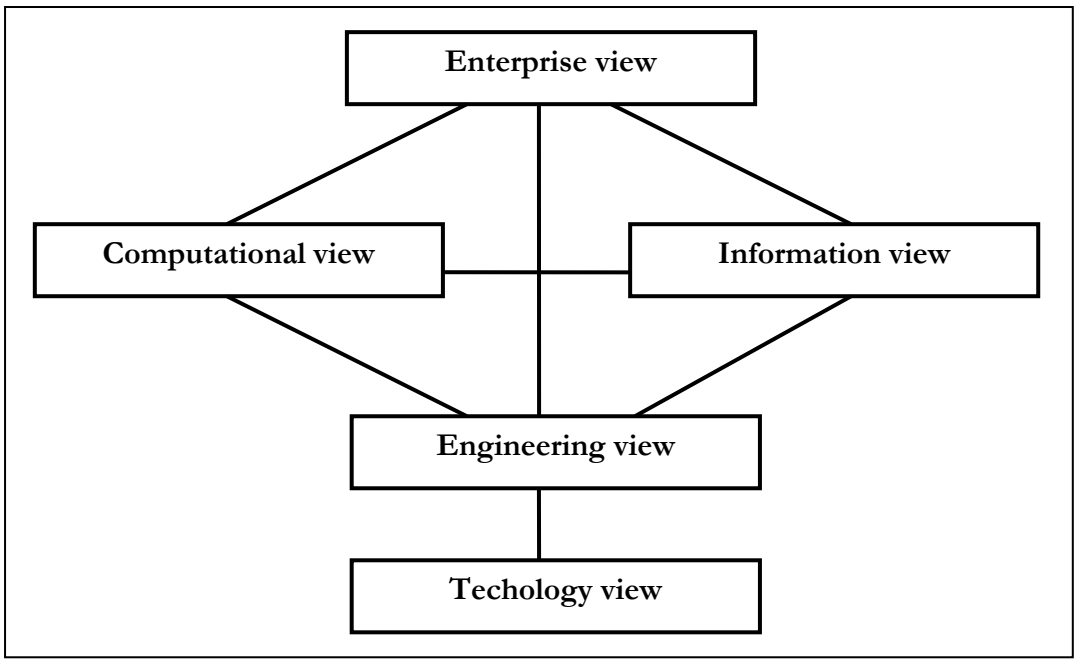
Oppgaven baserer seg på et forhåndsdefinert scenario, og dreier seg om et online betting-system, dvs. satsning på spill over Internett, hvor det eneste som var gitt er følgende Use Case-diagram:



Dette Use Case-diagrammet er grunnlaget for størstedelen av oppgaven.

Innholdsfortegnelse

1 Introduksjon



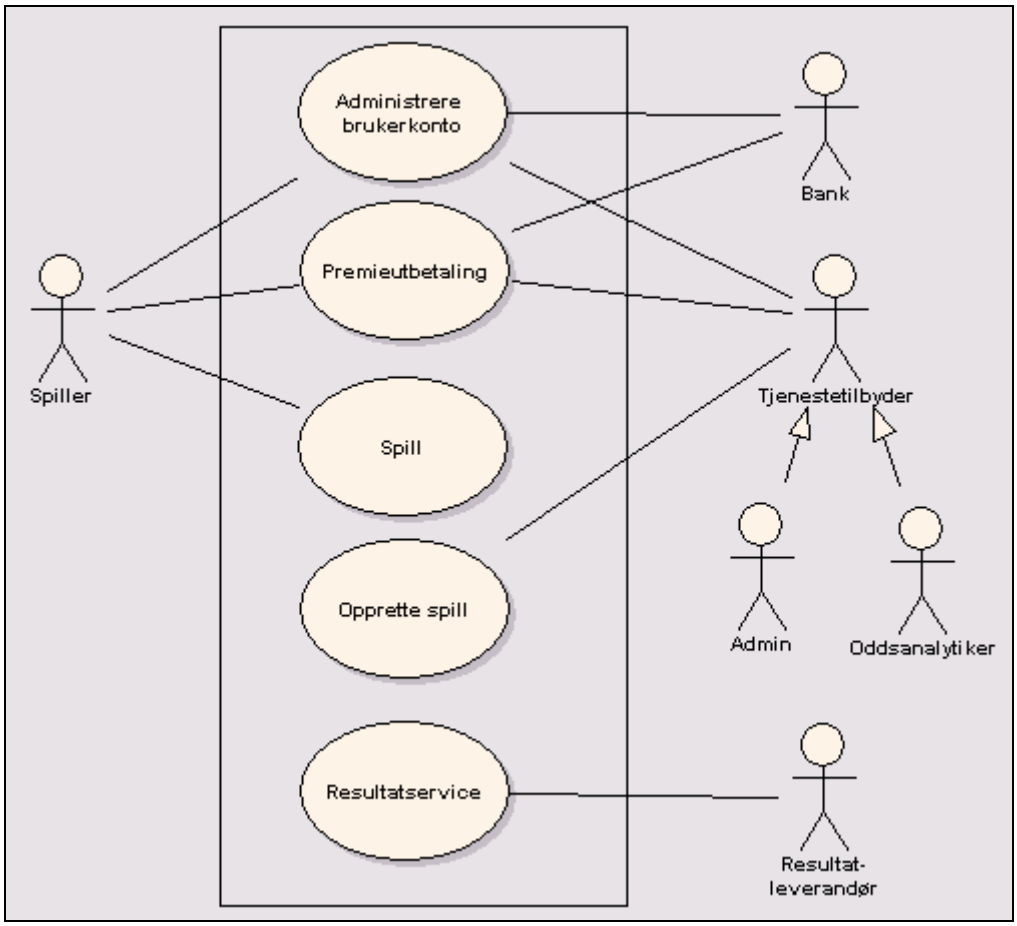
Figur 1 – RM ODP

RM ODP er grunnsteinen vi følger videre i denne oppgaven. Vi kommer til å analysere systemet gjennom de forskjellige view-ene basert på modeller utviklet i de forskjellige view-ene, og på denne måten finne ut hvilke deler som er sikkerhetskritiske.

Vi kommer først til å kartlegge overordnede krav til systemet i enterprise view, som deretter elaboreres i de påfølgende view-ene; computational og information. I engineering view blir systemets fysiske spredning detaljert. Technology viewet inneholder programkode og standarder denne baseres på. Dette er noe som faller utenfor vår oppgave.

2 Enterprise View

2.1 Use Case-diagram



Figur 2 – Use Case-diagram

2.1.1 Endringer i Use Case

Use Case-diagrammet beskrevet i oppgavebeskrivelsen fraviker noe fra det Use Case-diagrammet vi har valgt å arbeide ut i fra. Vi har gjort forandringer i det opprinnelige Use Case-diagrammet da vi var uenige i hvordan applikasjonen burde virke, samt noen aktørers rolle i mange Use Cases. Disse endringene ville i et reelt systemutviklingsprosjekt blitt utført i samarbeid med kunden, gjennom en produktiv diskusjon.

Den største forskjellen mellom det systemet vi ser for oss, og det som ble gitt i oppgaveteksten er at brukeren av applikasjonen, dvs. spilleren, har en lokal spillekonto hvor han eller hun kan deponere penger. Dette gjøres via Use Caset "*administrere brukerkonto*". Dette vil altså si at spilleren ikke vil behøve å foreta kredittkorttransaksjoner for hvert spill.

Vi er dessuten uenige i noen aktørers interesser i enkelte Use Cases og omfanget til enkelte Use Cases er endret. Vi har utvidet "*oddssetting*" til "*opprette spill*". Oddsanalytikeren setter da odds samtidig som han registrerer spillet i systemet. Videre har vi fjernet banken som en aktør i Use Caset "*spill*" (tidligere "*spill og betal*") da vi har valgt å bruke en lokal spillekonto som nevnt over. Vi har også valgt å ikke la brukeren være en aktør i Use Casene "*opprette spill*" og "*resultatservice*" da brukeren ikke har noen reelle mål i disse Use Casene.

Vi er også av den oppfatning at Use Caset "*administrere brukerkonto*" burde vært delt opp i to Use Cases, siden spiller og administrator har svært forskjellige mål mht det å administrere brukerkonto(er). Av denne grunn behandles dette Use Caset noe annerledes enn de andre ved utarbeiding av Use Case-beskrivelse og sekvensdiagram.

2.2 Aktører

2.2.1 Spiller

Spilleren er hovedbrukeren av applikasjonen, og den som utnytter applikasjonens tjenester. Spillerens hovedansvar i systemet er først og fremst å opprette sin egen brukerkonto i systemet slik at han senere kan logge inn og benytte seg av tjenestene tilbudt av applikasjonen. I disse tjenestene inngår det å overføre penger til lokal spillekonto, satse på spill og få utbetalt premiepenger.

2.2.2 Bank

Bankens oppgave i dette systemet er som et eksternt system å muliggjøre kredittkorttransaksjoner mellom spillerne og tjenestetilbyderen, i tillegg til girooverføringer fra tjenestetilbyderen til spilleren ved premieutbetalinger. Banken har ansvar for at dette gjøres på en sikker og feilfri måte.

2.2.3 Resultatleverandør

Resultatleverandøren er et eksternt system som har et tillitsforhold med applikasjonen, og som hele tiden gir innkommende resultater ved etterspørsel fra applikasjonen. Resultatleverandøren har ansvar for at resultatene kommer innen rimelig tid, og at de er korrekte.

2.2.4 Administrator

Administratoren er systemets "vaktmester". Denne aktøren har tilgang til alle brukerdata og kan endre disse ved behov. Det er også administratorens oppgave å ta aksjon ved ekstremsituasjoner. En slik situasjon kan være unormalt store eller mange utbetalinger til en spiller. Administratoren har altså ansvar for å administrere alle brukere, og den daglige driften av systemet. Denne aktøren har også et ansvar overfor seg selv, i og med at han er i en posisjon hvor han kan gjøre potensielt stor skade på systemet, og få tilgang til data av sensitiv karakter.

2.2.5 Oddsanalytiker

Aktøren oddsanalytiker er i realiteten en gruppe erfarne oddsanalytikere med én leder som har overordnet ansvar for at odds og spillfakta er så realistisk og korrekt som mulig. Oddsanalytikerne skal opprette spill i systemet, og angi odds, fakta og annen relevant spillinformasjon for disse.

2.3 Use Case-beskrivelser

2.3.1 Introduksjon

Vi har valgt å benytte oss av Cockburns Use Case-anbefaling [Cockburn01] i detaljering av Use Casene. Vi mener dette gir en meget konsis beskrivelse, og de får på en meget god måte fram aktørenes mål.

Cockburns modell benytter seg av en Use Case-beskrivelse, og i tillegg en beskrivelse av alternative scenarier. Beskrivelsen er ment å gi en utførlig beskrivelse av situasjonen det gjeldende

Use Case tar for seg. De alternative scenariene er ment å gi alternativer til enkelthendelser i hovedbeskrivelsen.

2.3.2 Use Case UC1: Administrere brukerkonto

Primæraktør: Spiller, administrator

Interessenter og deres interesser:
Spilleren ønsker å opprette en brukerkonto for å få tilgang til systemet, dessuten kunne oppdatere denne ved behov.
Spilleren vil også kunne betale penger inn på sin lokale konto.
Administratoren ønsker å kunne redigere enhver brukers personalia.

Beskrivelse:
Da dette er et Use Case som omfatter flere uavhengige scenarier, velger vi å ikke bruke samme fremgangsmåte som i de etterfølgende Use Cases, men heller beskrive disse prosaisk.

Spilleren vil først og fremst kunne opprette en brukerkonto, heretter kalt profil. Her kan spilleren registrere data som navn, adresse, telefonnummer, e-post, etc. Av sikkerhetshensyn skal ikke kontonummer lagres i profilen. Informasjonen i profilen skal brukeren også kunne endre og kunne hente opp ved behov. Ved endt oppretting av profil blir et automatisk generert passord sendt per e-post til brukeren for å validere e-postadressen.

Spilleren skal også kunne betale inn penger til sin lokale spillekonto via VISA/MasterCard-transaksjoner, som igjen kan satses på spill, så fremt spillekontoen har dekning.

Administratoren skal kunne gå inn og redigere en brukers personopplysninger, fortrinnsvis på oppfordring fra brukeren. Administrator skal som hovedregel ikke ha tilgang til å debitere eller kreditere brukerens spillekonto, men hvis det viser seg at denne lokale kontoen av en eller annen grunn ikke er korrekt, må det finnes en rutine for å ordne opp i dette.

2.3.3 Use Case UC2: Premieutbetaling

Primæraktør: Spiller, bank

Interessenter og deres interesser:
Spilleren ønsker å hente ut penger fra lokal spillekonto.
Banken ønsker å effektuere korrekt overføring.

Prebetingelser:
Spiller må være identifisert og autorisert.
Spiller må ha penger på lokal spillekonto.

Postbetingelser:
Overføring av penger ble registrert hos banken.

Beskrivelse:
1. Spiller skriver inn ønsket beløp.
2. Systemet kontrollerer beløpets størrelse.
3. Spiller oppgir kontoinformasjon.
4. Systemet foreviser kontoinformasjon for bekreftelse
5. Spiller godkjenner kontoinformasjon.
6. Systemet registrerer utbetaling hos banken.

7. Systemet debiterer spillers lokale spillekonto.
8. Systemet logger utbetalinger og transaksjoner.

Alternative scenarier:
2a. Spiller har ikke nok penger på lokal spillekonto:
    1. Systemet lar bruker taste inn beløp på nytt.
2b. Beløpet ønsket utbetalt er større enn en kontrollgrense:
    1. Systemet varsler administrator.
    2. Administrator kontrollerer utbetalingen.
    3. Administrator godkjenner utbetaling.
        3a. Administrator godkjenner ikke utbetaling:
            1. Loggføres som misbruk.
    4. Systemet registrerer utbetaling hos banken.
5a. Spiller godkjenner ikke kontoinformasjon.
    1. Systemet lar spilleren skrive inn kontoinformasjon på nytt.

2.3.4 Use Case UC3: Spill

Primæraktør: Spiller

Interessenter og deres interesser:
Spiller vil satse på et spill.

Prebetingelser:
Spiller må være identifisert og autorisert.
Spiller må ha penger på lokal spillekonto.

Postbetingelser:
Innsatsen ble registrert.
Lokal spillekonto ble debitert.

Beskrivelse:
1. Spilleren forevises tilgjengelige spill og spillhistorikk.
2. Spilleren velger et spill.
3. Spilleren registrer innsats.
4. Spilleren bekrefter innsats og spill.
5. Systemet lagrer innsats og debiterer spillekonto.

Alternative scenarier:
3a. Spilleren vil angre allerede registrert innsats:
    1. Systemet kontrollerer at ikke fristen for å angre innsats har gått ut
    2. Spilleren angrer innsats.
    3. Spilleren bekrefter.
    4. Systemet sletter innsatsen og krediterer spillekonto.
3b. Spilleren vil endre allerede registrert innsats:
    1. Systemet kontrollerer at ikke fristen for å endre innsats har gått ut.
    2. Spilleren endrer innsatsen.
    3. Spilleren bekrefter.
    4. Systemet oppdaterer innsatsen og krediterer/debiterer spillekonto.

2.3.5 Use Case UC4: Opprette spill

Primæraktør: Oddsanalytiker

Interessenter og deres interesser:
Oddsanalytikeren ønsker å opprette et spill.

Prebetingelser:
Oddsanalytiker må være identifisert og autorisert.

Postbetingelser:
Spillet ble opprettet.

Beskrivelse:
1. Oddsanalytikeren legger inn fakta og statistikk om spillet.
2. Oddsanalytikeren legger inn odds.
3. Systemet lagrer spillet med gitte data.
4. Oddsanalytikeren aktiverer spillet.
5. Systemet gjør spillet tilgjengelig for spillere.

Alternative scenarier:
1a. Fakta om spillet inneholder feil:
    1. Oddsanalytikeren retter opp feil.
    2. Systemet lagrer oppdateringer.
4a. Annullerer spillet:
    1. Oddsanalytikeren annullerer spillet.
    2. Systemet sletter spillet.
4b. Endrer spillet.
    1. Oddsanalytikeren endrer feilaktige data.
    2. Systemet lagrer nye data

2.3.6 Use Case UC5: Resultatbehandling

Primæraktør: Resultatservice

Interessenter og deres interesser:
Spilleren ønsker oppdaterte og korrekte resultater.

Postbetingelser:
Korrekte resultater er lagret i systemet.
Vinnende spillere har fått utbetalt gevinst til lokal spillekonto.

Beskrivelse:
1. Systemet etterspør resultat fra resultatservice.
2. Systemet lagrer resultater fortløpende.
3. Systemet behandler innkommende resultater.
4. Systemet krediterer vinnende spilleres lokale spillekontoer.
5. Systemet gir melding til alle deltagende spillere om resultatet.
Gjenta steg 1-5 med jevne intervaller for alle aktive spill

Alternative scenarier:
1a. Systemet oppnår ikke kontakt med resultatservice:

1. Systemet loggfører hendelsen.
2. Systemet prøver å oppnå kontakt på nytt.

2.4 Sekvensdiagram

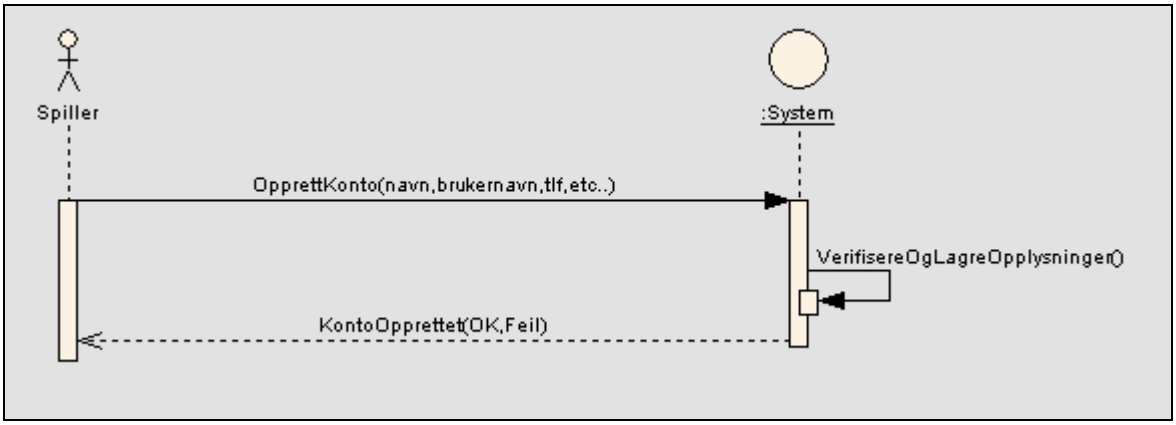
2.4.1 Introduksjon

Under utarbeidelsen av sekvensdiagrammene har vi basert oss på det vi mener er "*best practice*". For enkelte Use Cases har vi derfor laget flere enn ett sekvensdiagram, der vi så at dette var nødvendig for å lette forståelsen. Hovedsakelig skjer dette i Use Case 1, "*administrer brukerkonto*", hvor man har flere separate hendelsesforløp.

Vi har på dette stadiet i planleggingsarbeidet ennå ikke funnet konkrete klasser. Sekvensdiagrammene er derfor overordnede og viser kun interaksjon mellom allerede identifiserte aktører og systemet.

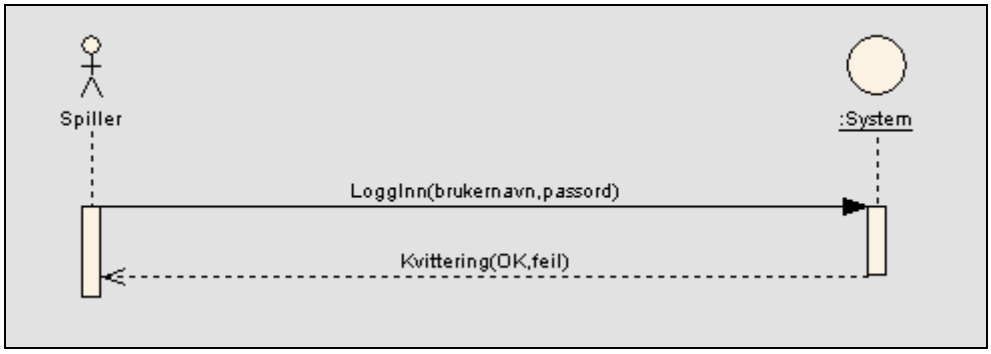
2.4.2 Sekvensdiagrammer

Alle sekvensdiagrammer forutsetter at bruker er logget inn, med unntak av sekvensdiagrammet for innlogging i UC1 (*Figur 4 – UC1: Logg inn*).



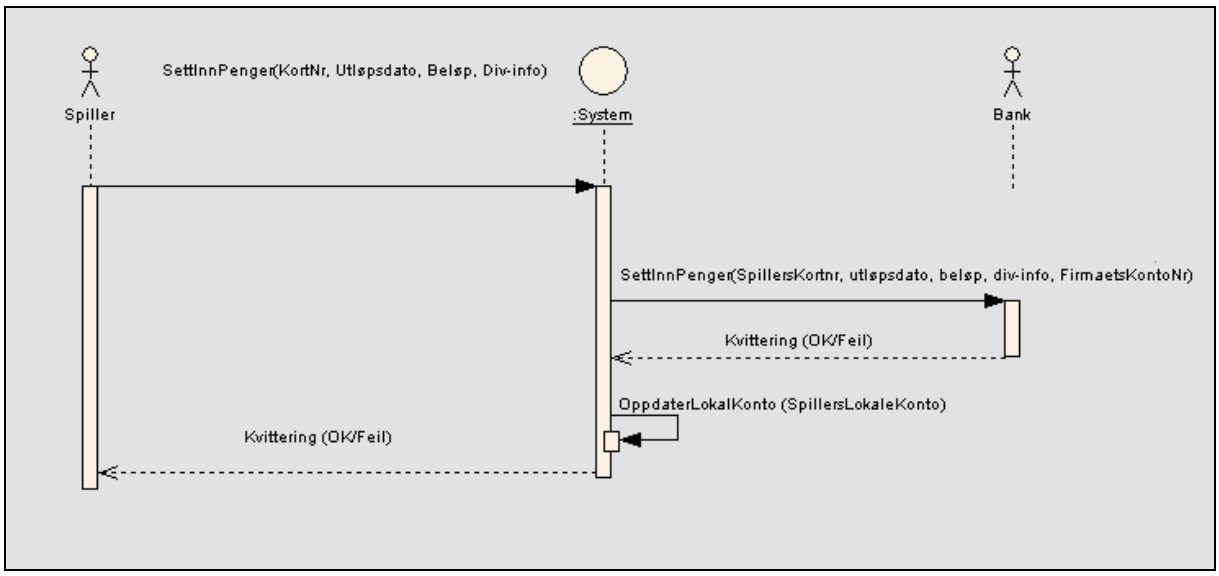
Figur 3 – UC1: Opprett profil

Figur 3 – UC1: Opprett profil viser hvordan en spiller oppretter en profil i applikasjonen for første gang. Systemet verifiserer informasjonen før den lagres. Når dette er gjort er det tenkt at brukeren skal motta innloggingsinformasjon via e-post. Sekvensdiagrammer for å endre brukerkonto er tilnærmet identisk med dette, med unntak av at man ikke i alle tilfeller vil få tilsendt ny innloggingsinformasjon med mindre spiller forspør dette.



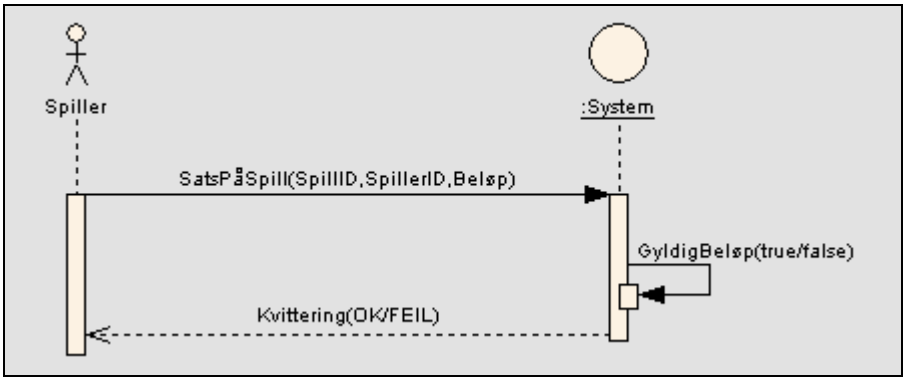
Figur 4 – UC1: Logg inn

Figur 4 – UC1: Logg inn viser innloggingsprosedyren for spilleren, men den er identisk for de andre brukerne av systemet, dvs. administrator og oddsanalytiker. Teknisk sett vil ikke innloggingen foregå på samme måte da spilleren vil bruke et Web-interface og tjenestetilbyderne vil benytte seg av en annen type klient, men informasjonsflyten mellom bruker og system er slik som vist i figuren over.



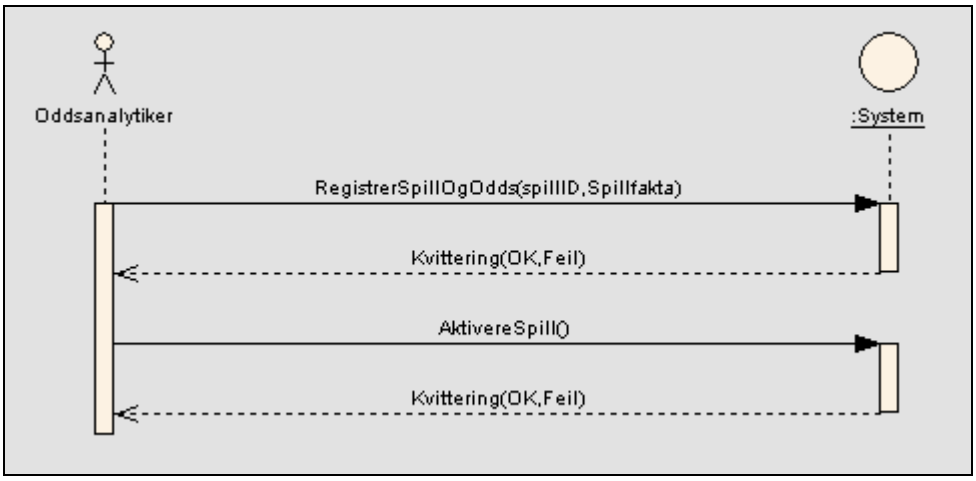
Figur 5 – UC1: Betal inn penger på lokal spillekonto

Figur 5 – UC1: Betal inn penger på lokal spillekonto detaljerer hvordan spilleren oppgir sine kredittkortopplysninger til systemet, som videre benytter seg av et kredittkortbetalingssystem for å utføre transaksjoner mellom spillerens kredittkortselskap og vår bank. Systemet oppdaterer spillerens lokale spillekonto når transaksjonen er godkjent.



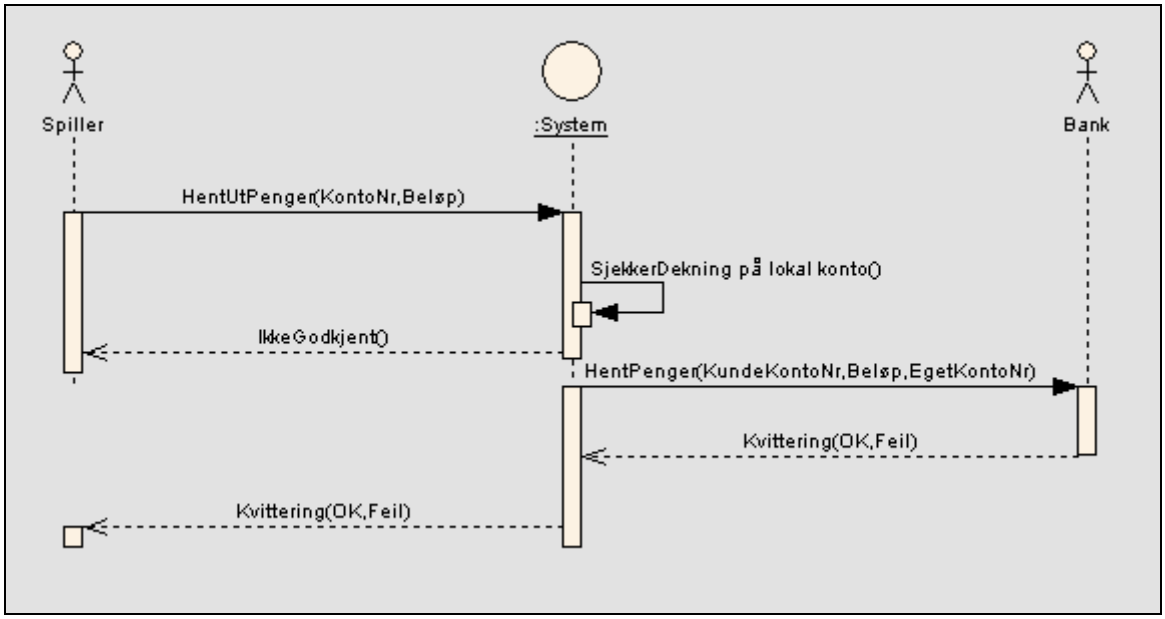
Figur 6 – UC2: Spille

Figur 6 – UC2: Spille viser hvordan en spiller kan satse på et gitt spill. Med funksjonen "*GyldigBeløp*" kontrollerer systemet om brukeren har nok penger på den lokale spillekontoen før satsningen godkjennes og penger trekkes fra spillerens spillekonto.



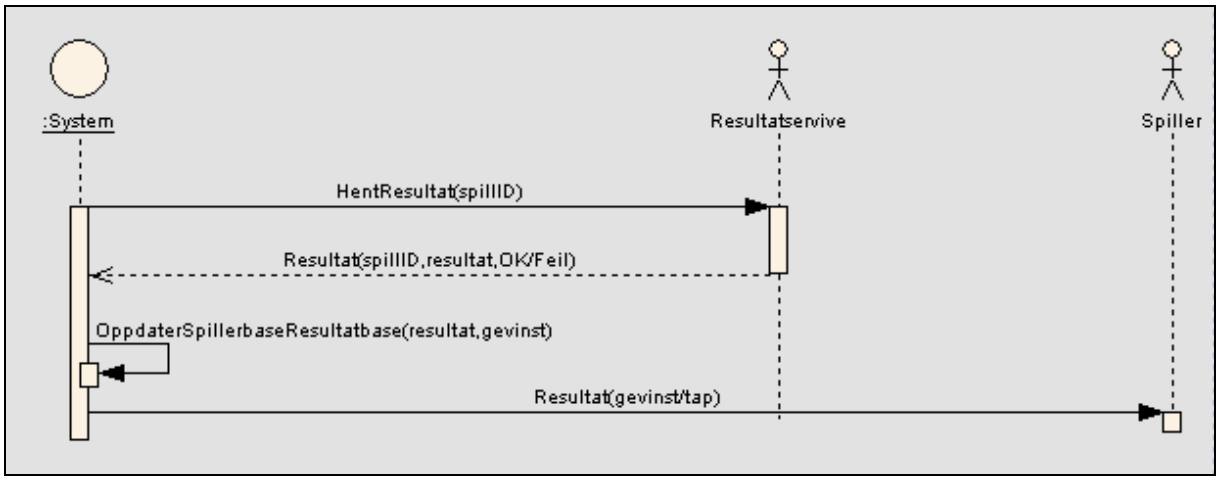
Figur 7 – UC3: Opprette spill

Figur 7 – UC3: Opprette spill viser hvordan oddsanalytikeren registrerer data og odds om et spill, som en spiller senere kan spille på. Parameteren "*spillfakta*" i funksjonen "*RegistrerSpillOgOdds*" inneholder alle relevante data om spillet, samt odds. Oddsanalytikeren må også aktivere spillet etter at han har fått kvittering på at nødvendig data er lagret.



Figur 8 – UC4: Premieutbetaling

Figur 8 – UC4: Premieutbetaling tar for seg situasjonen hvor spilleren vil ha utbetalt penger fra sin lokale spillekonto. Spilleren angir da kontonummer og et ønsket beløp, hvorpå systemet kontrollerer at det er dekning på spillekontoen. Spilleren har ansvar for å oppgi korrekt kontonummer, da det ikke er noen slike kontroller i systemet. Systemet bruker videre kontonummeret, i tillegg til allerede lagret personalia (navn og adresse), for å registrere en girooverføring fra vår bankkonto til spillerens.



Figur 9 – UC5: Resultatservice

Figur 9 – UC5: Resultatservice viser hvordan systemet henvender seg til Resultatservice for å få oppdaterte resultater for aktive spill som er ferdigspilt. Systemet gjør på bakgrunn av resultatet fra henvendelsen en løpende oppdatering av spill og spillekontoer hvor vinnende spillere får kreditert kalkulert gevinst. Alle deltakende spillere får melding om resultatet og eventuelle gevinster.

2.5 Sikkerhetsanalyse

Sikkerhetsanalysen baserer seg på modeller og beskrivelser i kapitlene 2.1 til 2.4.

2.5.1 Sikkerhetsanalyse av Use Case-diagram

2.5.1.1 Sensitiv informasjon

Det er mye sensitiv informasjon i systemet, dvs. informasjon som andre kan dra nytte av eller vil være kompromitterende for den som er eier av informasjonen.

Tabell 1 - Sensitiv informasjon

Hva	Hvorfor
Personalia gitt av spilleren, og passord	Her er mye sensitiv informasjon gitt av spilleren. Informasjon som telefonnummer og adressen til spilleren er sensitivt sammen med informasjon som for eksempel beløp han spiller for, penger han har på konto osv.
VISA og MasterCard informasjon	Dette er i høyeste grad sensitiv informasjon som kan brukes av en angriper for å forfalske transaksjoner eller overføre penger til egen bankkonto. Dette oppgis hver gang brukeren skal overføre penger til lokal spillekonto.
Kontonummer	Dette er sensitiv informasjon som ikke skal lagres i databasen. Kontonummeret må derfor oppgis hver gang spilleren skal overføre penger til bankkontoen sin.
Innloggingsinformasjon -en til oddsanalytiker	Denne informasjonen kan misbrukes av en angriper så han kan utgi seg for å være oddsanalytiker, og på denne måten sette egne odds på spill. Han kan også legge inn falske spill eller aktivere spill som ikke skal aktiveres enda.
Innloggingsinformasjon -en til administrator	Dersom en angriper får tak i dette vil han enkelt få tilgang til hele systemets database og den informasjonen som ligger lagret her. Dette

	vil blant annet gi han tilgang til sensitiv informasjon om spillere, samt informasjon om spill som kan misbrukes eller forfalskes.

2.5.1.2 Aktører og aksessrettigheter

Spiller

Spilleren må ha aksessrettigheter til å endre personopplysninger i sin egen profil. Han eller hun må også kunne ha tilgang til se spillfakta og odds til de spillene spilleren er interessert i. For å hente eventuelle penger spilleren har vunnet må han/hun kunne overføre penger fra sin premiepott til sin egen bankkonto ved og oppgi bankkontonummer.

Oddsanalytiker

Oddsanalytikeren skal ha tilgang til å sette opp spill i systemet. På hvert spill skal det kunne legges inn spillfakta, odds og deadline.

Administrator

Administrator må kunne ha full aksess til å lese og endre alle spillerkontoene samt å kunne stenge spillerkontoer ved misbruk.

Bank

Banken trenger ingen aksessrettigheter i vårt system for å betale ut og overføre penger.

Resultatleverandør

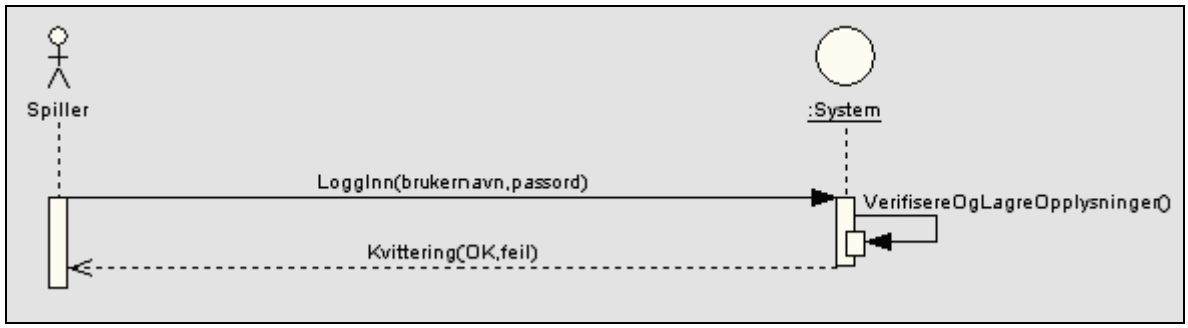
Resultatleverandøren trenger heller ingen aksessrettigheter for å sette resultat på de enkelte spill inn i systemet, men systemet henter data og lagrer resultater ut ifra denne.

2.5.2 Sikkerhetsanalyse av sekvensdiagram

Under følger revisjoner av enkelte sekvensdiagram som vi etter nærmere sikkerhetsmessig inspeksjon har endret for å bedre ivareta sikkerheten i systemet.

2.5.2.1 Logg inn

I denne sekvensen er det spillerens brukernavn og passord som er de mest sensitive dataene. Prosessen med å overføre disse dataene over Internett fra spillerens nettleser til vår applikasjon er meget utsatt. Ved å logge inn oppnår brukeren de privilegier som er nødvendige for å benytte de tjenester som applikasjonen tilbyr.



Figur 10 – UC1: Logg inn (revidert)

*Figur 10 – UC1: Logg inn (revidert)*Figur 10 – UC1: Logg inn viser den nye utgaven av *Figur 4 – UC1: Logg inn*. Etter revidering av sekvensdiagrammet har vi lagt til funksjonen *VerifisereOgLagreOpplysninger* som gjør kontroll på input (brukernavn og passord). Funksjonen bør hashe passordet og kontrollere om det er det samme som er lagret i databasen.

2.5.2.2 Opprett konto

Også i denne sekvensen vil sensitiv data bli overført over Internett. Her inngår blant annet spillerens brukernavn, passord, navn, e-postadresse og andre personalia.

Ved å opprette konto (profil) vil brukeren oppnå de privilegier som er nødvendige for å logge inn i systemet.

2.5.2.3 Endre brukeropplysninger

Identisk som 2.5.2.2.

2.5.2.4 Betal inn penger på lokal spillekonto

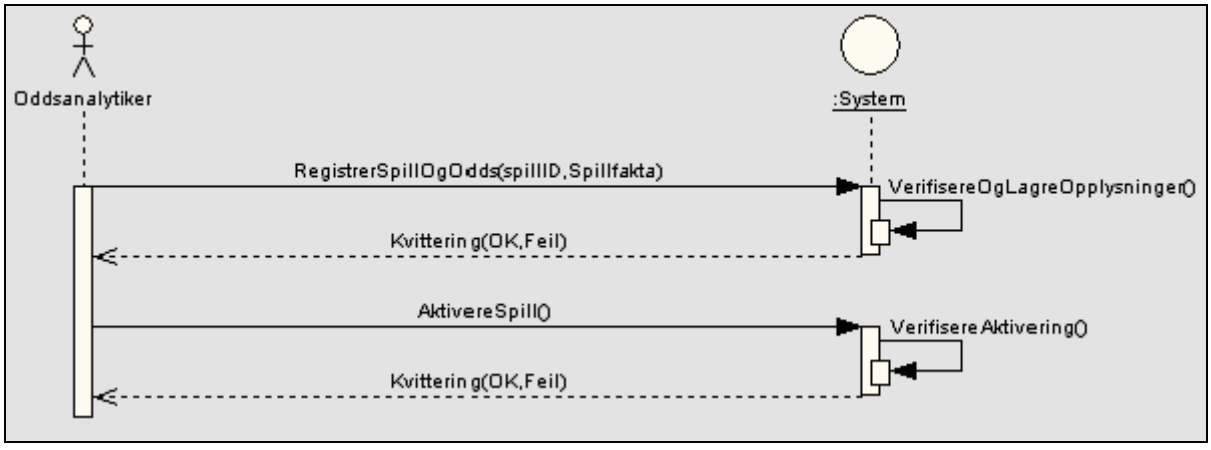
I denne sekvensen er det spillerens kredittkortnummer og kredittkortets utløpsdato som er sensitive. Disse blir i likhet med andre opplysninger i tidligere sekvenser, sendt over Internett fra spillerens nettleser til vår applikasjon. I tillegg vil applikasjonen kreditere spillerens spillekonto. Dette skjer på et internt nettverk, men dataene er av stor viktighet og er derfor svært utsatt.

2.5.2.5 Spille

Her er beløpet spilleren satser på et spill av sensitiv natur, spesielt med tanke på at noen kan forsøke å endre beløpet.

2.5.2.6 Opprette spill

Sensitive opplysninger vi har her er oddsen og ikke minst spillfakta. Grunnen til at vi anser spillfakta som sensitive data er at bruker på bakgrunn av disse dataene kan komme til å ta feil avgjørelser med tanke på satsning.



Figur 11 – UC3: Opprette Spill (revidert)

Figur 11 – UC3: Opprette Spill (revidert) viser den nye utgaven av *Figur 7 – UC3: Opprette spill*. Under revideringen av dette sekvensdiagrammet har vi lagt til funksjonene *VerifisereOgLagreOpplysninger* og *VerifisereAktivering*, som gjør de nødvendige kontrollene på inputen til systemet fra oddsanalytiker.

2.5.2.7 Premieutbetaling

Premieutbetalingen krever at brukeren skriver inn sitt kontonummer. Applikasjonen benytter seg av kontonummer og personalia for å registrere en overføring av penger hos banken.

2.5.2.8 Resultatservice

Resultatleverandøren skal her ikke ha noen privilegier i vår applikasjon. Alle henvendelser skal gå fra applikasjonen til resultatleverandøren.

2.5.3 Risikoanalyse

Risikoproblematikk rundt dette systemet, kan for eksempel være at systemet går ned eller at systemet blir angrepet av uønskede utenforstående hendelser.

2.5.3.1 Uønskede hendelser med årsakskjeder, sannsynlighet og skadeomfang

A. Strømbrudd/brudd i kommunikasjonslinjer/DoS:

Et strømbrudd eller kommunikasjonsbrudd kan føre med seg økonomiske konsekvenser da spill kan gå tapt på grunn av ukontrollerte systemstopp. Dette kan medføre systemfeil ved omstart av servermaskiner, og transaksjoner kan bli feil med inkonsistente data. Dette medfører også økonomiske tap for spillerne og selskapet, da de ikke får satset på planlagte spill. Skadeomfanget vil avhenge av hvor lang strømstansen eller kommunikasjonsbruddet er, men det kan her fort bli snakk om store tap av renommé og omsetning.

Tiltak for å begrense skadene vil være å innføre et databasesystem som støtter rollback av avbrutte transaksjoner (dvs. at databasen returnerer til en konsistent tilstand), sammen med en batteridrevet strømkilde (UPS) som kan tilføre strøm til systemet i tilstrekklig tid for kontrollert systemstans slik at alle transaksjoner vil få anledning til å bli avsluttet.

B. Uønsket tilgang til oddssetting:

Hvis en angriper får tilgang til oddssetting kan han forandre odds på spill og dermed påføre selskapet økonomisk tap og tap grunnet dårlig omdømme. Skadeomfanget er dermed stort.

Tiltak kan være krypterte forbindelser, og kombinasjon av autentiseringsmekanismer. Man kan kombinere noe man vet sammen med noe man har for å styrke sikkerheten. For eksempel et passord sammen med et smartkort.

C. Uønsket tilgang til profil:

Adgang til en spillers profil kan misbrukes til å overføre premieutbetalinger til angripers bankkonto, og en angriper kan satse på spill med penger som allerede står på offerets spillekonto. Dette medfører økonomisk tap for bruker. Skadeomfanget for bedriften kan variere ut i fra hvor modig og heldig en eventuell angriper er angående hvilke spill han eller hun satser på og hvor mye gevinsten er.

Tiltak vil være det samme som punkt B, i tillegg til å føre logg med IP-adresser og annet viktig informasjon for å identifisere avvik fra normalen.

D. Tilgang til database:

Skrivetilgang til database hvor data om balanse på lokal spillekonto er lagret. Misbruker kan endre saldo etter eget ønske. Har en angriper tilgang til systemets database vil han eller hun også kunne endre odds på enkelte kamper som det deretter kan satses mye penger på. Skadeomfang vil da høyst sannsynlig være stort både for selskap og spiller. Selskapet vil tape penger ved utbetaling av premie og spilleren som eier eventuell spillekonto som har blitt endret kan tape penger om misbruker har tappet spillekonto ved innbrudd i databasen.

Tiltak kan være fysisk og logisk sikre tilgang til databaseinformasjon. Databaseserveren skal kun være tilgjengelig via intranett, og ikke utad. Dessuten skal unormalt høye utbetalinger til enkeltspillere undersøkes av administrator i et forsøk på å avdekke misbruk.

E. Uærlige administratorer:

Administrator og oddsanalytiker kan utnytte sin makt i systemet til egen vinning. Oddsanalytikeren kan legge inn urealistiske odds på spill som han eller andre kan satse på og oppnå meget store premier. Skadeomfanget vil være stort i form av omsetningstap om det skjer.

Uærlige administratorer vil det være vanskelig å beskytte seg mot, og eventuelle tiltak kan være gode kontrollrutiner slik at en person ikke har eneansvar for systemkritiske operasjoner.

F. Feil/ingen resultatdata fra resultatservice:

Ukorrekte data fra resultatservice enten ved brudd, brukerfeil eller ondsinnet angrep, kan føre til feil utbetaling både i negativ og positiv retning for kunde og selskap.

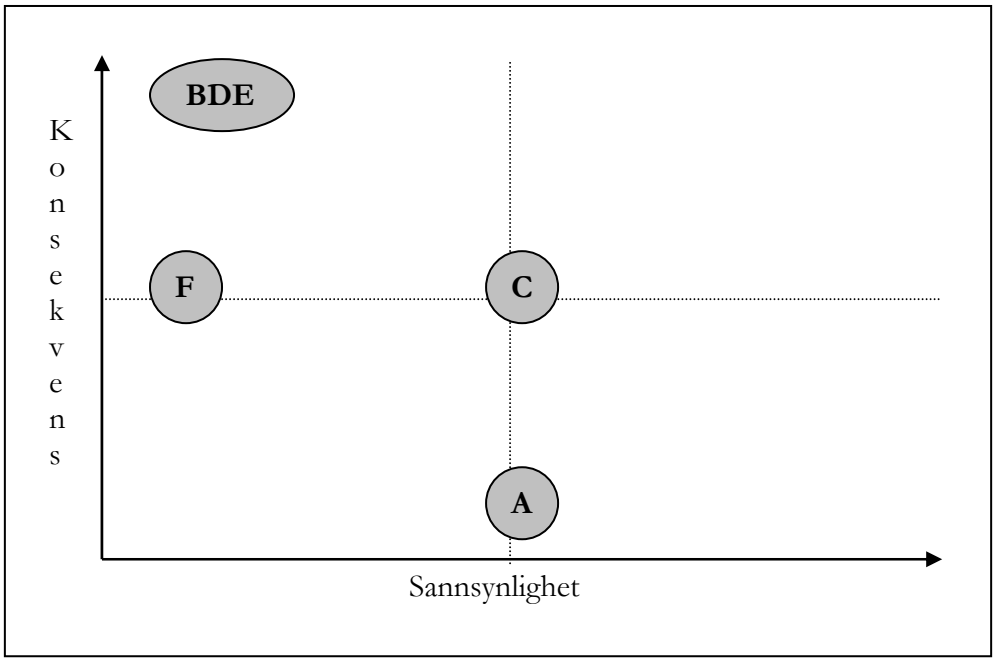
Tiltak kan være dobbel overføring av resultater og kryptering av alle ledd i forbindelsen, for eksempel opprette et virtuelt privat nettverk (VPN) mellom system og resultatleverandør.

Definisjoner av risikonivå

Høy sannsynlighet:	Noe som kan inntreffe flere ganger daglig.
Middels sannsynlighet:	Noe som kan skje med flere ukers mellomrom.
Lav sannsynlighet:	Det kan være opptil ett år mellom hver gang noe inntreffer.
Høy konsekvens:	Noe som setter bedriften/systemet ut av spill en periode. Kan ramme bedriften hardt økonomisk og stort tap av omdømme.
Middels konsekvens:	Noe som kan sette ut enkelte deler av systemet, men ikke hele. Flertallet av spillerne vil ikke bli rammet av stor grad. Kan økonomiske konsekvenser for bedriften og kan føre til tap av omdømme hvis situasjonen ikke håndteres profesjonelt og situasjonen inntreffer gjentatte ganger.
Lav konsekvens:	Noe som ikke rammer bedriften hardt økonomisk, men kan føre til en kortere nedetid av systemet og vil gi lite tap av omdømme.

Tabell 2 - Sannsynligheter og konsekvenser

	Sannsynlighet			**Konsekvens**		
	Høy	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
A. Brudd		√				√
B. Oddssetting			√	√		
C. Profil		√			√	
D. Database			√	√		
E. Administrator			√	√		
F. Resultatservice			√		√	



Figur 12 – Sannsynlighet-konsekvensgraf

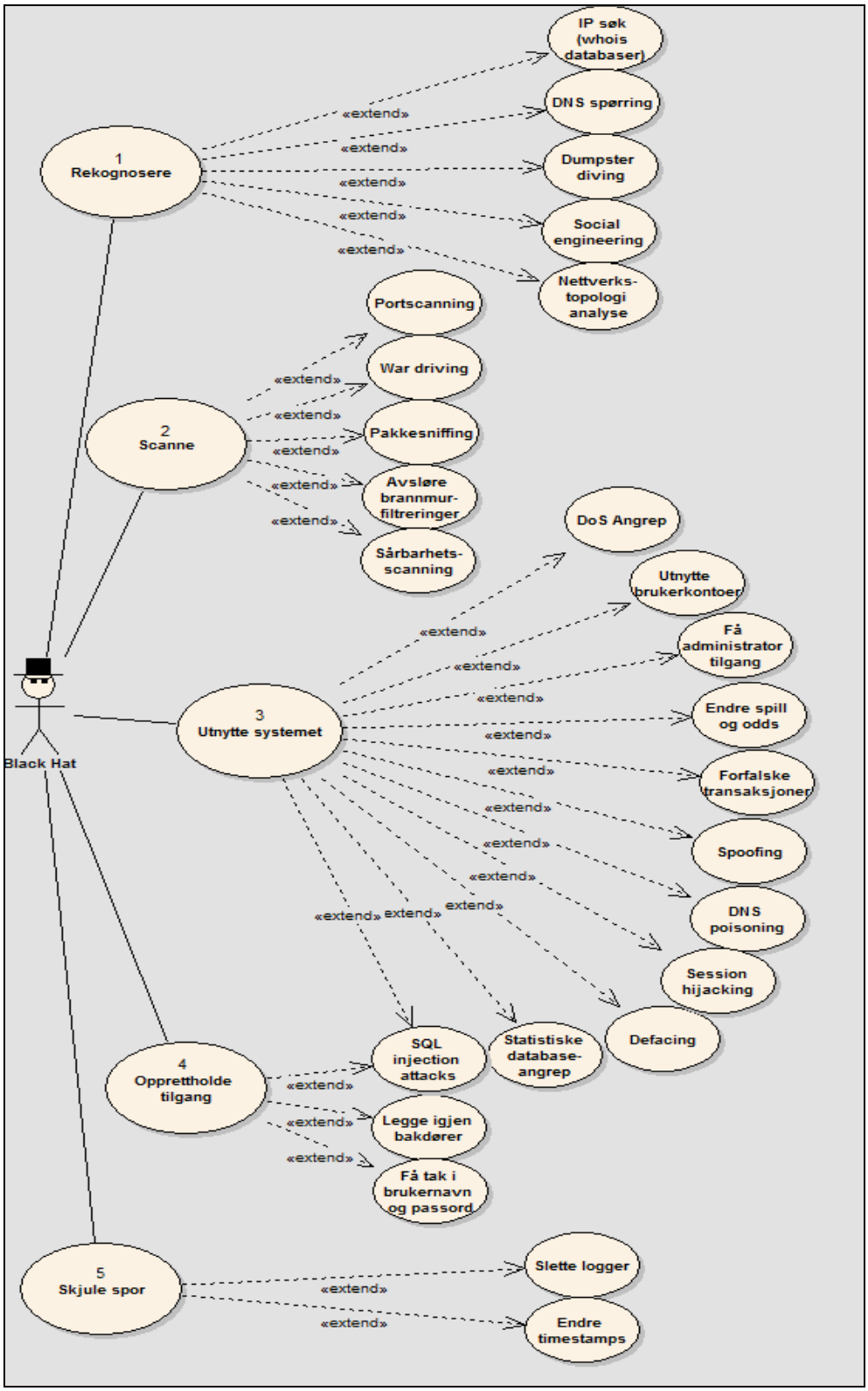
Figur 12 – Sannsynlighet-konsekvensgraf er basert på *Tabell 2 - Sannsynligheter og konsekvenser* og visualiserer på en god måte forholdet mellom konsekvens og sannsynlighet. Ut i fra dette kan vi gjøre en risikorangering.

2.5.3.2 Risikorangering

I følge Norsk Standard 5814 er risiko produktet av sannsynlighet og konsekvens, men det er umulig å tallfeste disse. Vi velger derfor å se på rangeringen mer ut fra skjønn. Ut fra *Tabell 2 - Sannsynligheter og konsekvenser*, *Figur 12 – Sannsynlighet-konsekvensgraf* og vår vurdering av sannsynlighet og konsekvens, følger denne risikorangeringen:

1. Uønsket tilgang til profil
2. Strømbrudd/brudd i kommunikasjonslinjer/DoS
3. Tilgang til database med lagret informasjon om lokal spillekonto
4. Uønsket tilgang til oddssetting
5. Uærlige administratorer
6. Feil/ingen resultatdata fra resultatservice

2.5.4 MisUse Case



Figur 13 – MisUse Case

Figur 13 – MisUse Case viser en ondsinnet angripers fremgangsmåter for angrep på systemet. Disse mulige angrepene er det viktig at systemet er beskyttet imot slik at sannsynligheten for et vellykket angrep blir minst mulig. Som det framgår av figuren er det fem hovedstadier av et angrep; rekognosering, scanning, utnytting av systemet, opprettholding av tilgang og skjuling av

spor. For å beskytte seg mot angrep er det viktig å hindre angrepet på et tidlig stadium, helst allerede under rekognoseringsdelen. For å vanskeliggjøre innsamling av informasjon om system bør minst mulig informasjon om systemet offentliggjøres på Internett og i andre offentlig tilgjengelige kanaler

Under forklares enkelte av angrepsmetodene i noe mer detalj.

2.5.4.1 Social Engineering

Når man benytter seg av social engineering utnytter man det svakeste ledd i ethvert systems sikkerhetskjede, nemlig mennesket. Social engineering går ut på å bruke innflytelse og overtalelsesevner til å lure et offer til å tro at man er en annen enn den man i virkeligheten er. Man kan etter å ha oppnådd et tillitsforhold, utnytte offerets godvilje til å skaffe til veie sensitiv informasjon, uten å benytte seg av tekniske hjelpemidler. For å forhindre eller begrense social engineering er opplæring av personale i alle deler av bedriften utrolig viktig.

2.5.4.2 Nettverkstopologianalyse

Ved å benytte seg av sofistikerte verktøy som for eksempel Cheops-ng, kan man få innsyn i bedriftens nettverksstruktur. Slik er viktig informasjon for en angriper når han eller hun skal finne mål for et rettet angrep.

2.5.4.3 War Driving

War Driving går ganske enkelt ut på å komme seg inn på bedriftens trådløse nettverk ved hjelp av en bærbar PC. En angriper trenger ikke å komme seg inn på bedriftens områder eller kontorer, han trenger kun å sitte trygt utenfor i en bil.

2.5.4.4 Session Hijacking

Ved pålogging til en Web-applikasjon blir en bruker tildelt en sesjonsvariabel. Denne variabelen identifiserer brukeren ovenfor HTTP-serveren. Hvis en angriper stjeler denne variabelen og setter denne i sin nettleser, vil han få tilgang til offerets profil, og alle rettigheter offeret har i systemet.

2.5.4.5 SQL Injection-angrep

SQL Injection er aktuelt over alt i systemet der man leser inn input fra brukeren, eller lar brukeren velge noe som så benyttes direkte i en SQL-spørring på serveren. Hvis en applikasjon ikke er godt nok programmert for å sjekke om inputdata er gyldig, kan en angriper med enkle grep klare å kjøre egne SQL-spørringer via inputfeltet, som for eksempel kan oppdatere spillekontoen med penger.

2.5.5 Tiltak og konklusjon

Det er åpenbart helt i fra starten at databasesikkerhet blir ekstremt viktig. Alle sensitive data bør krypteres i databasen, og man bør da helst kryptere sensitiv informasjon utenfor databasen og lagre de krypterte dataene i databasen. Kommunikasjonslinjene inn og ut fra applikasjonen er også av høyeste viktighetsgrad. Kryptert kommunikasjon vil i denne forbindelse være et krav. For å implementere dette kommer vi til å bruke HTTPS mellom spiller og vår applikasjonsserver, og VPN og IPSec mellom våre servere og serverne til banken og resultatleverandøren. Vi kommer heller ikke til å bruke et Web-interface for å administrere systemet da dette strengt tatt ikke er nødvendig, og kun åpner for flere sikkerhetshull med påfølgende sikkerhetsforanstaltninger. Man vil da kunne unngå å utføre viktige administrative oppgaver over Internett, men i stedet utføre dette over bedriftens intranett. Dette vil altså si at tjenestetilbyderens aktører vil benytte seg av en egenutviklet applikasjon. Hvis det i ettertid skal vise seg at administrator eller oddsanalytiker er

absolutt avhengig av tilgang til systemet hjemmefra vil vi komme til å se på VPN-løsninger også her.

2.6 Implementasjonsspråk

Slik systemet er designet i de foregående kapitlene, er det meget godt egnet for en klient/server-løsning hvor mesteparten av funksjonaliteten som spilleren kommer til å benytte seg av vil ligge på HTTP-serveren. Av denne grunn velger vi å benytte oss av PHP[1] på en Apache HTTP-server[2] til dette. Det finnes konkurrerende teknologier, blant annet fra Microsoft, men kombinasjonen Apache og PHP har den fremtredende konkurranseegenskapen at den er gratis, også i produksjonssammenheng.

Siden vi i kapittel 2.5.5 konkluderte med at det vil være sikkerhetsmessig risikabelt å la administratoren administrere systemet over Internett, vil det ikke være grunnlag for å utvikle en Web-applikasjon for slik funksjonalitet. Det vil være mer effektivt å benytte seg av et språk som egner seg bedre til vanlig applikasjonsutvikling slik som Java.

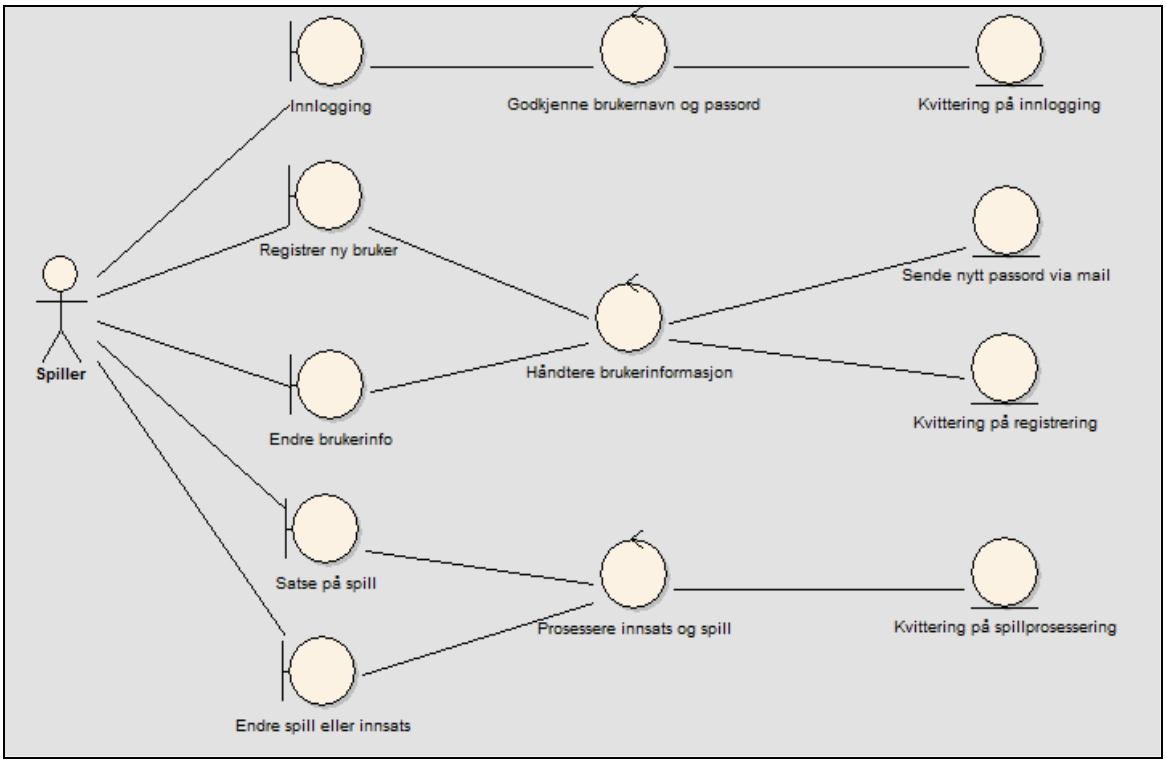
Som databaseserver vil det bli benyttet PostgreSQL[3] grunnet dennes utmerkede ytelse og sikkerhetsegenskaper (slik som *views* (se kapittel 0), noe som for eksempel MySQL[4] foreløpig ikke har).

I et reelt systemutviklingsprosjekt vil det også her bli diskutert med kunden hva som skal være implementasjonsspråk og maskinvareplattform, slik at det best mulig oppfyller kundens krav.

[1] http://www.php.net
[2] http://httpd.apache.org
[3] http://www.postgresql.org
[4] http://www.mysql.com

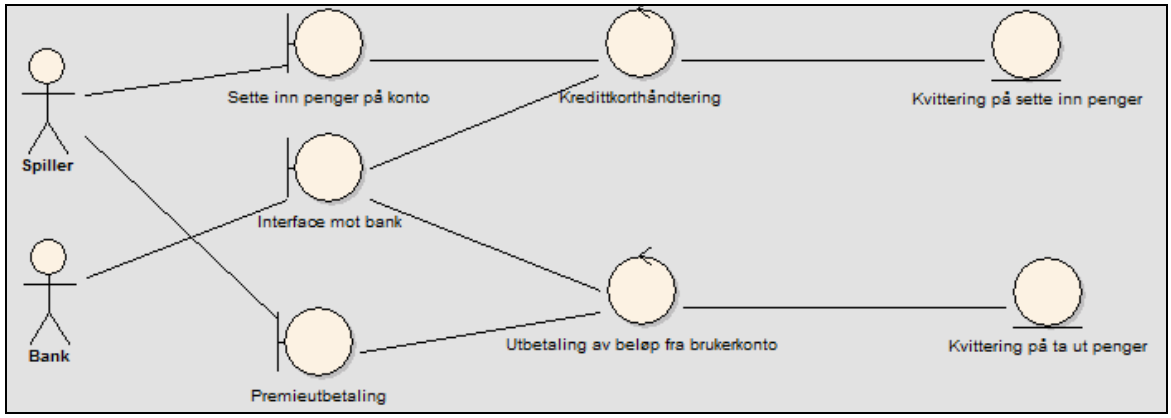
3 Computational View

3.1 BCE modeller



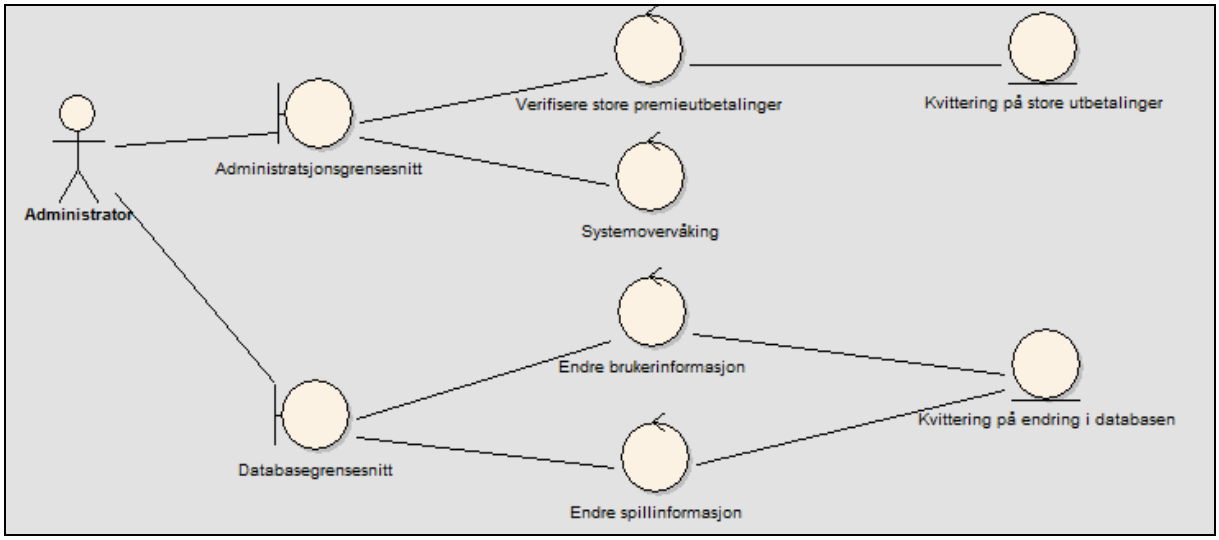
Figur 14 – Spillerens hovedfunksjoner

Figur 14 – Spillerens hovedfunksjoner viser hovedfunksjonene til en spiller og hvilke prosesser disse må ha. Spiller har i hovedsak fem grensesnitt. Det første tar for seg innlogging på systemet. Et annet grensesnitt er å registrere seg som ny bruker, her blir passord sendt via e-post. Det samme grensesnittet blir brukt ved endring av allerede registrert brukerinformasjon. Det fjerde grensesnittet tar for seg satsning på spill og en egen prosess tar seg av prosesseringen av innsatsen. Den samme prosesseringen brukes ved endring av spill eller innsats der tidsfristen ikke er gått ut.



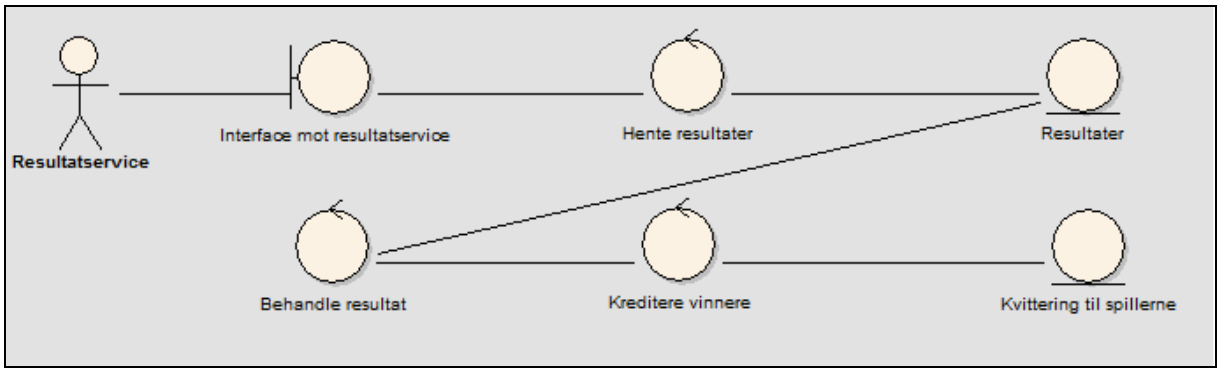
Figur 15 – Kommunikasjon mellom spiller og bank

Figur 15 – Kommunikasjon mellom spiller og bank viser hvordan transaksjonen mellom en spiller og banken foregår. Grensesnittene mot brukeren tar seg av premieutbetaling og overføring av penger til systemet (til spillekonto). Her må man ha en egen prosess som tar seg av kreditkorthåndtering og en prosess som tar seg av betaling til kundens bankkontonummer. Grensesnitt mot banken er kun en kommunikasjonskanal mot banken. Det er ikke noe brukergrensesnitt på grunn av at banken kun er en passiv aktør.



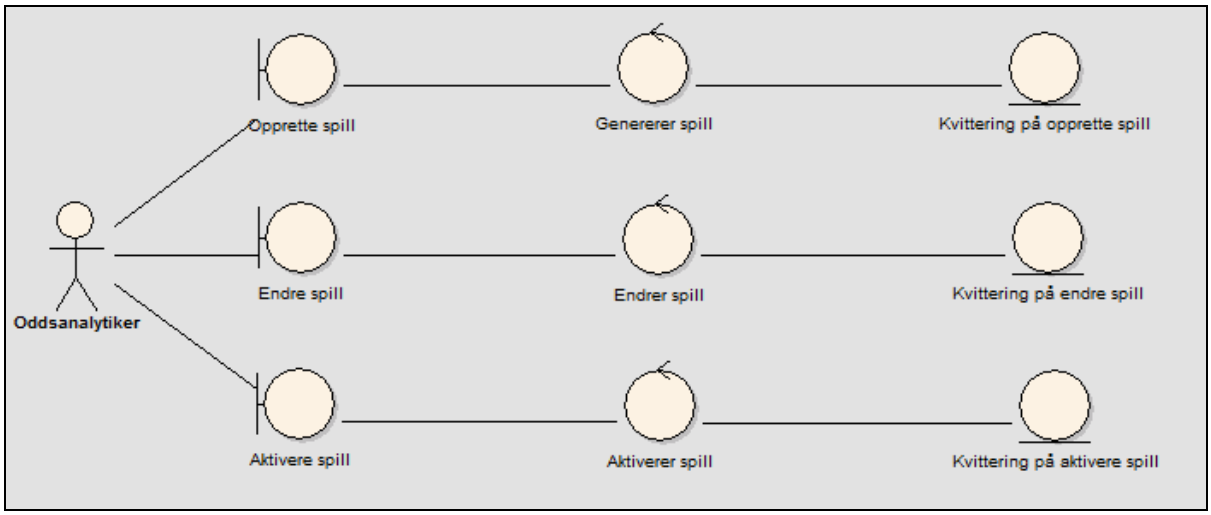
Figur 16 – Administratorens hovedfunksjoner

Figur 16 – Administratorens hovedfunksjoner viser grensesnittene til administrator og hvilke prosesser som kreves av administrator. Administrator har i hovedsak to grensesnitt. Et for systemovervåkning og verifisering av store premieutbetalinger, og et grensesnitt som tar seg av databasehåndtering, både spillerinformasjon og spillinformasjon.



Figur 17 – Resultatbehandling

Figur 17 – Resultatbehandling viser kommunikasjonsgrensesnittet mot resultatservice. En prosess henter resultat og behandler disse dataene. Til slutt utbetales premiene til vinnerne og spillerne får kvittering på e-post.

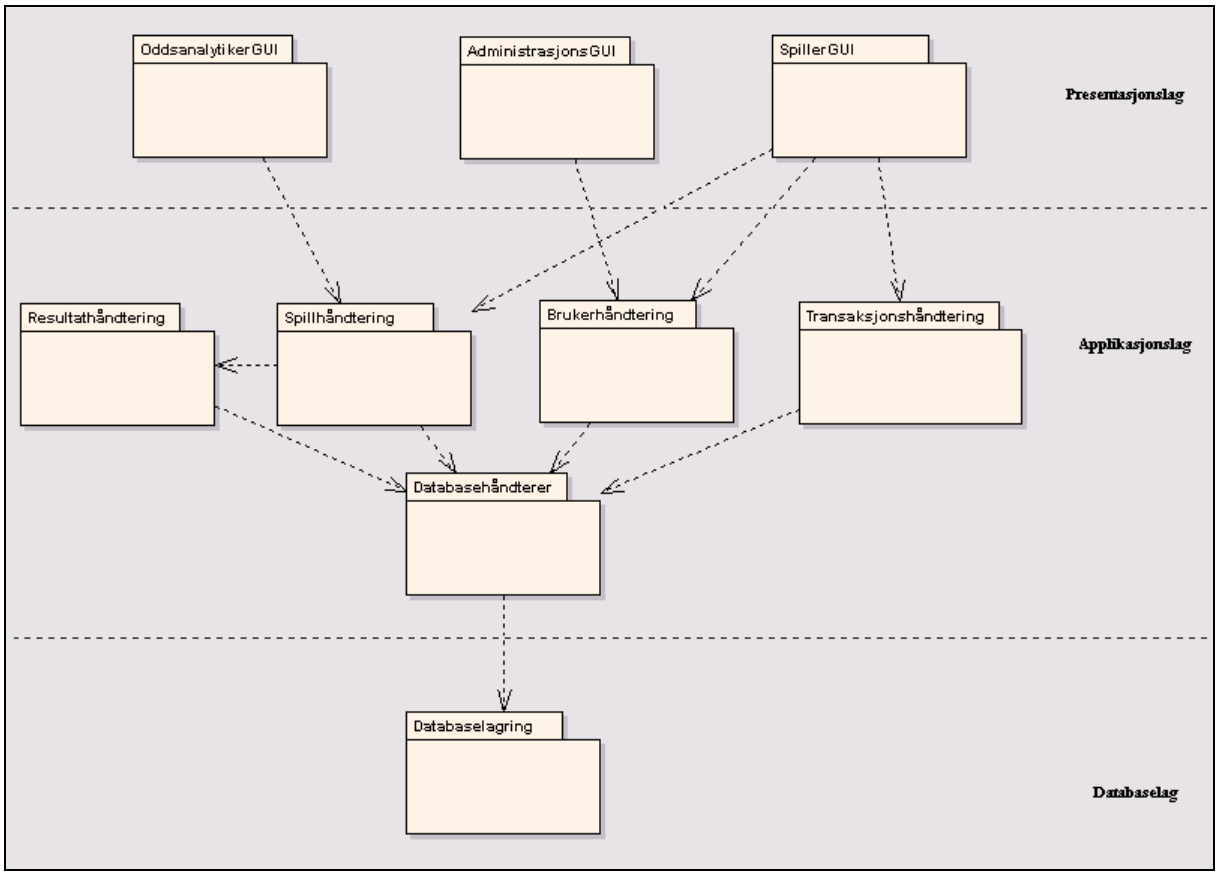


Figur 18 – Oddsanalytikerens hovedfunksjoner

Figur 18 – Oddsanalytikerens hovedfunksjoner viser grensesnittene mot oddsanalytikeren. Oddsanalytiker har tre grensesnitt mot systemet. Det ene er å opprette et spill med spillfakta og odds. Nummer to er å endre et spill som er lagt inn. Den siste grensesnittet oddsanalytikeren har er å aktivere et spill som allerede er lagt inn.

3.2 Lagdelt arkitektur

En vanlig måte å designe et system på, er å dele det opp i lag som er abstraksjonsnivåer. På denne måten er det lettere å få oversikt over systemet, og dets kritiske og ikke-kritiske deler.



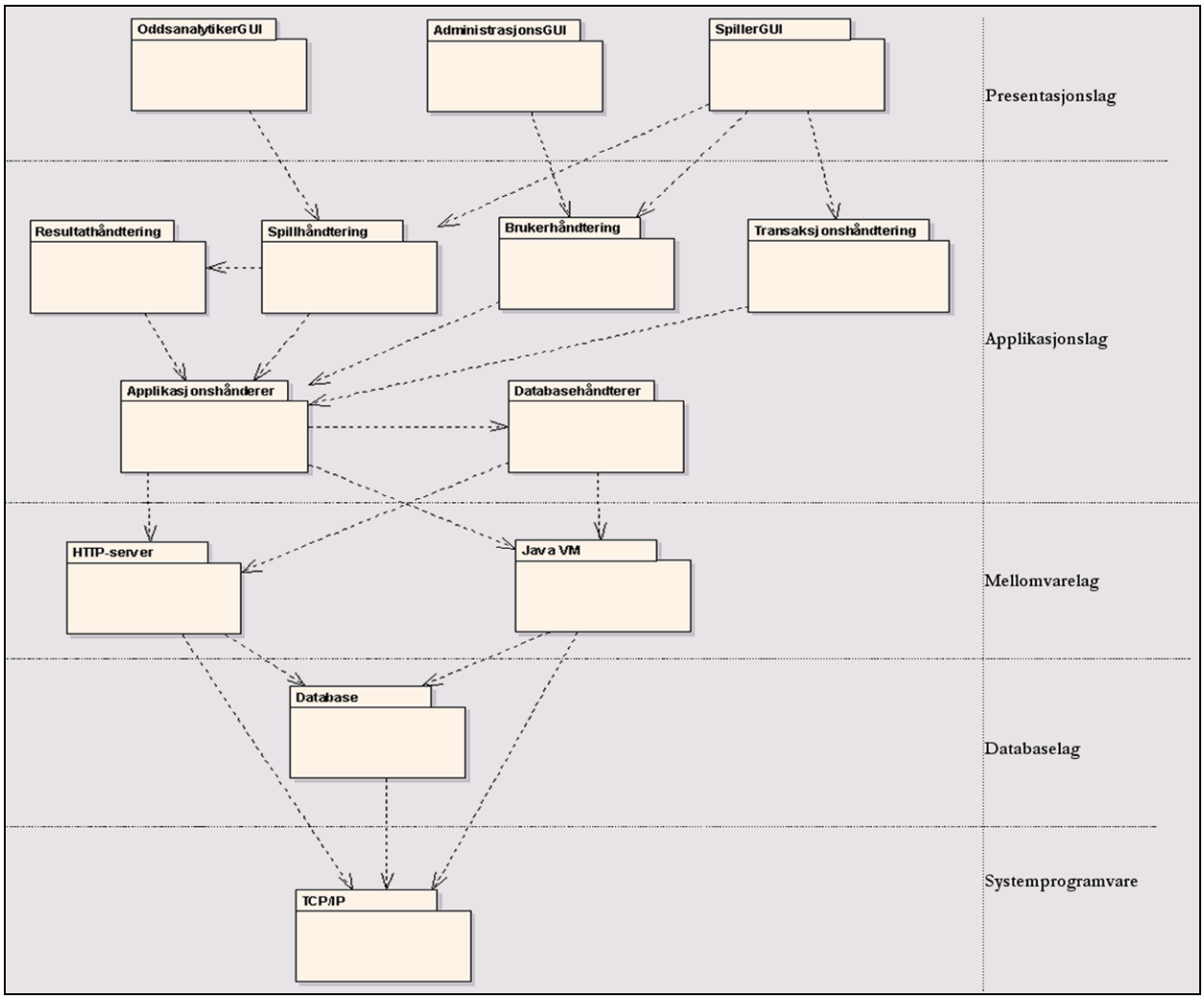
Figur 19 – Lagdelt arkitektur

Figur 19 – Lagdelt arkitektur viser en trelags struktur av systemet. Figuren går ikke i detalj på hver enkelt klasse i systemet, men den viser den overordnede strukturen i systemet med pakker, bestående av klasser, i de forskjellige lagene av systemet. Hvert lag i applikasjonen går alltid ned for å hente data og gjøre handlinger. Dette er i tråd med et softwarepattern eller et prinsipp som sier at klasser helst bare skal bruke klasser og henta data fra objekter som ligger lavere enn seg selv. Hvis klasser eller objekter som ligger lavere i arkitekturen gjør seg avhengig av klasser over dem selv blir det fort ukonsistens og problemer. Et eksempel kan være at tre GUI applikasjoner viser frem klokketiden og dato på tre forskjellige måter. Så endrer systemet innstillingen på klokka til vintertid. Hvis vi ikke hadde fulgt top-down prinsippet måtte systemet sendt ut den nye tiden opp til hver av de tre GUI applikasjonene. Men med top-down prinsippet ville hver GUI applikasjon gått ned selv og hentet den nye tiden. Da trengs bare endringen å skje et sted, men får konsekvenser for flere deler/applikasjoner av systemet.

Det øverste laget er presentasjonslaget med grafiske brukergrensesnitt mot forskjellige aktører i systemet. Her vil vi finne all interaksjon med brukerne, og et mål for disse klassene er at interaksjonen skal være enkel og forståelig mot brukeren.

Lag to er applikasjonslaget hvor selve systemets funksjonalitet ligger. Her er systemet delt inn i 5 pakker, *resultathåndtering*, *spillhåndtering*, *brukerhåndtering*, *transaksjonshåndtering* og *databasehåndtering*. Hver av pakkene inneholder klasser med attributter og funksjoner for å kunne utføre og kontrollere brukernes handlinger i systemet, og videre dirigere transaksjoner mot databasen. Databasehåndteringspakken vil fungere som et grensesnitt mot databaselaget.

Det siste laget er databaselaget, her ligger databasen hvor all informasjon ligger lagret. Dette er den mest essensielle delen av systemet. Skjer det noe feil her, eller data ikke blir riktig lagret, vil feilen fort forplante seg oppover i hele systemet. Når det gjelder sikkerhet er det viktig å tenke på dette i alle lagene av systemet. Denne første modellen ble satt opp veldig tidlig i utviklingsfasen. Siden vi på dette tidspunktet ikke hadde besluttet hvilke teknologiske løsninger vi ville velge viser denne modellen ikke noe utover dette, men hvordan systemet skal kommunisere nedover og bygge på hverandre. Etter vi besluttet å benytte oss av Java for å bygge hoveddelen av administrasjonsmodulene og webmoduler for spillerne har vi laget en litt mer utdypende modell.



Figur 20 – Lagdelt arkitektur (revidert)

I denne modellen har vi delt opp den lagdelte arkitekturen i to flere lag, henholdsvis mellomvarelag og systemprogramvare. I mellomvare laget finner vi HTTP-server og Java VM. Java VM vil fungere som en motor for å få kjørt Java-modulene, og HTTP-server vil ta seg av alle handlinger over Internett. Begge disse pakkene kommuniserer videre med databasen på databaselaget. På bunnen i denne modellen er systemprogramvare- og kommunikasjonslaget. Her har vi plassert pakken *TCP/IP* som er grunnsteinen i applikasjonen, når denne er den som skal sørge for all kommunikasjonen i systemet. I applikasjonslaget har vi plassert en ny pakke, *applikasjonshåndterer*. Denne klassen vil "samle" alle modulene og bestemme hvilke som er javamoduler og skal kjøres med Java VM, og hvilke som er webscrip og skal kjøres av HTTP-serveren.

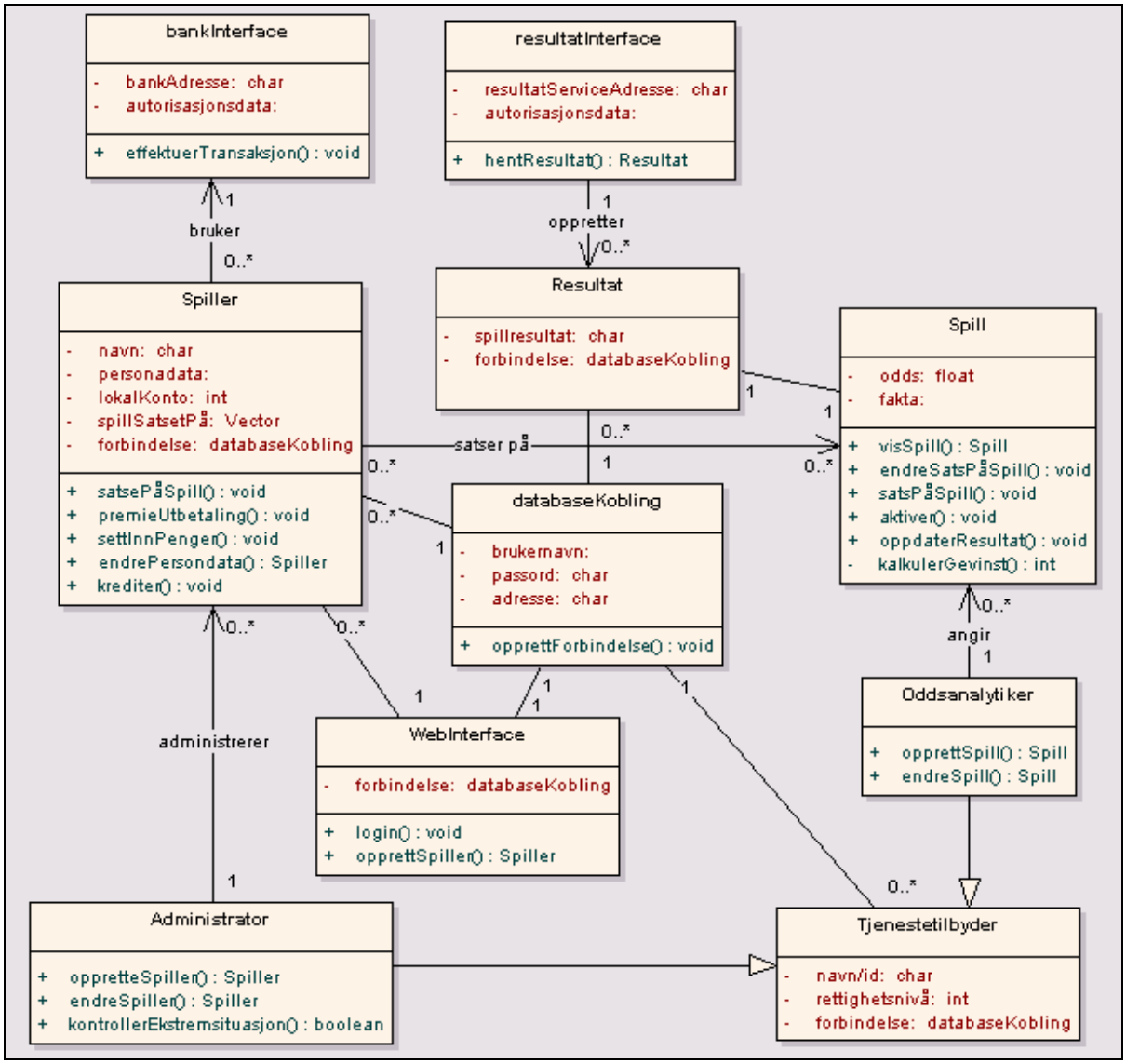
For å få en oversikt over klassene i systemet og hvordan de er knyttet til hverandre, har vi laget et UML-klassediagram. Diagrammet viser ikke hvilke klasser som er i hvilke pakker, men relasjonene mellom klassene og dens overordnede funksjoner og attributter.

4 Information View

4.1 Klassediagram

Klassediagrammet er ment å detaljere de viktigste informasjonselementene i systemet. Man kan se hvilke klasser som systemet er tenkt å inneholde, hvilke hovedattributter og -funksjoner de har.

Klassene representeres med tre hovedgrupper av attributter: navn, variabler og funksjoner eller metoder. De forskjellige variablene og funksjonene kan enten være private for klassen (anvist med minus (-)) eller offentlige for hele systemet (anvist med pluss (+)). Der det er hensiktsmessig er returtype fra funksjonen og type av variabel angitt.



Figur 21 – Klassediagram

Av figuren kan vi først og fremst se at klassen *tjenestetilbyder* er en generalisering av klassene *administrator* og *oddsanalytiker*. Dette betyr at de to klassene begge også er *tjenestetilbyderen*, og de arver alle attributter og funksjoner som *tjenestetilbyder* har. Grunnen til at det differensieres mellom *administrator* og *oddsanalytiker* er at disse to har meget forskjellige arbeidsoppgaver og mål i

systemet. Mens *administratoren* i hovedsak vil ha med *spiller* å gjøre, vil *oddsanalytikeren* i første rekke håndtere *spill*.

Klassen *webinterface* håndterer registrering og innlogging av brukeren. Det er denne klassen som lager en instans av *spiller* når en spiller logger inn. *webinterface* er en creator i systemet som definert i GRASP[5] [Larman01] da den oppretter en instans av *spiller* ved hjelp av initialiseringsdata den inneholder.

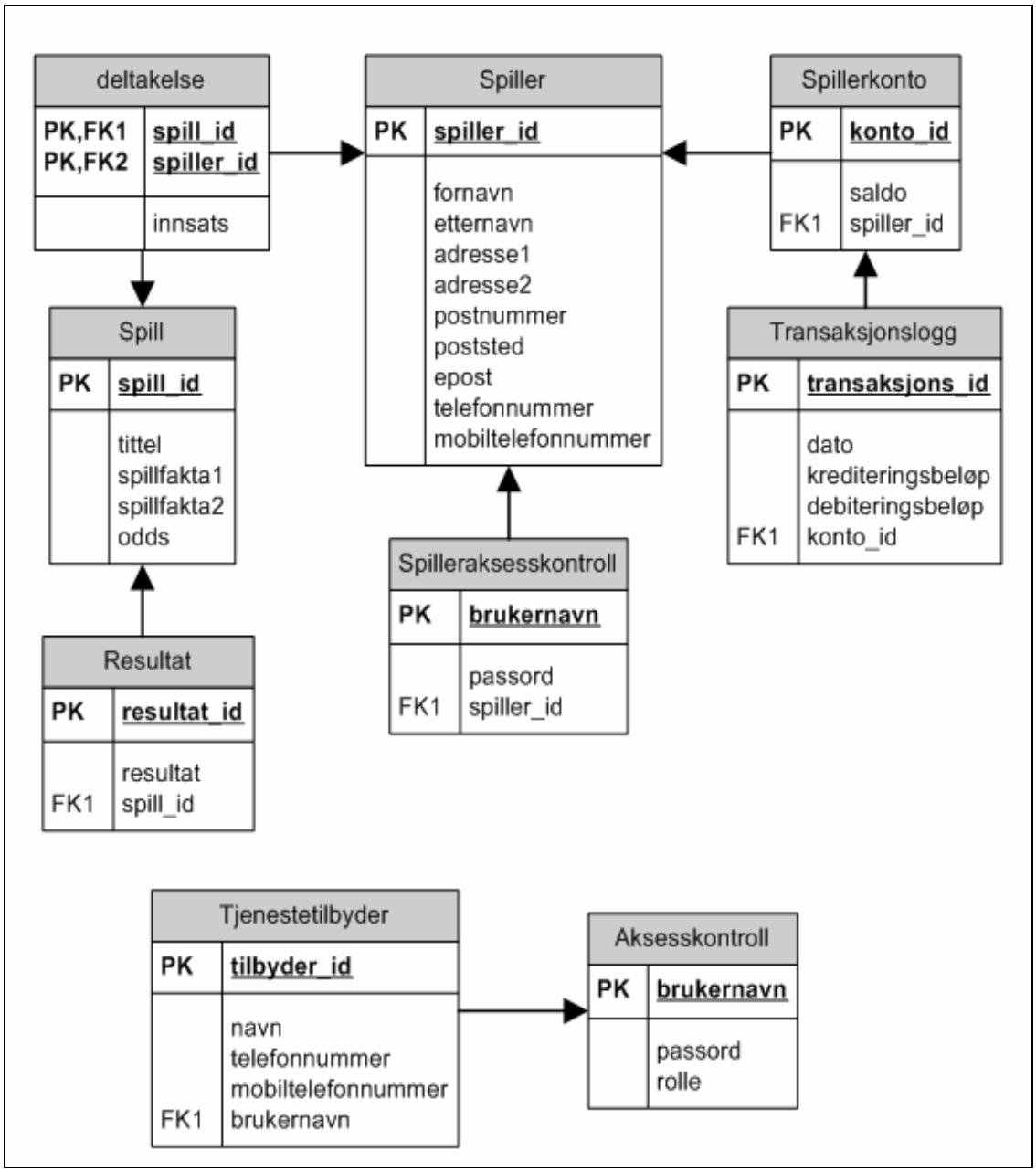
Klassen *databasekobling* er en controller i systemet slik som Larman har definert det i GRASP, og alle forespørsler til databasen må gå via denne. Vi kan da sikre integritet i databasespørringer og forbindelser på en bedre måte enn hvis databaseforbindelser ble opprettet i mange andre klasser i systemet, da alle slike kontroller kan foretas i *databasekobling*. Ved å benytte en kontrollerklasse er det også enkelt å logge alle forespørsler uten mye redundant kode. Som nevnt i kapittel 3.2 benyttes denne controlleren som et grensesnitt mellom to lag i arkitekturen.

4.2 ER-diagram for database

Det er viktig å få på plass et klassediagram før man går løs på implementasjonen av systemet. Like viktig er det å ha klart for seg den grunnleggende datastrukturen i databasen. Hvis man må gjøre store endringer i databasestrukturen under implementasjonen av applikasjonen er man i de fleste tilfeller nødt til å gjøre store endringer, noe som kan åpne for sikkerhetshull og andre feil som kan være vanskelige å oppdage.

Under utviklingen av denne databasemodellen er det lagt vekt på å spre informasjonen så mye som mulig, slik at det vil være noe vanskeligere for en angriper å endre gitt informasjon enn hvis alle sensitive data var plassert i samme tabell. Tanken er at ved bruk så skal det opprettes views som plukker relevant informasjon fra de forskjellige tabellene. Dette viewet skal det ikke være mulig å manipulere. Se kapittel 6.2.2 for mer informasjon angående databasesikkerhet i systemet.

[5] Genernal Responsibility Assignment Software Patterns



Figur 22 – ER-diagram for database

5 Engineering View

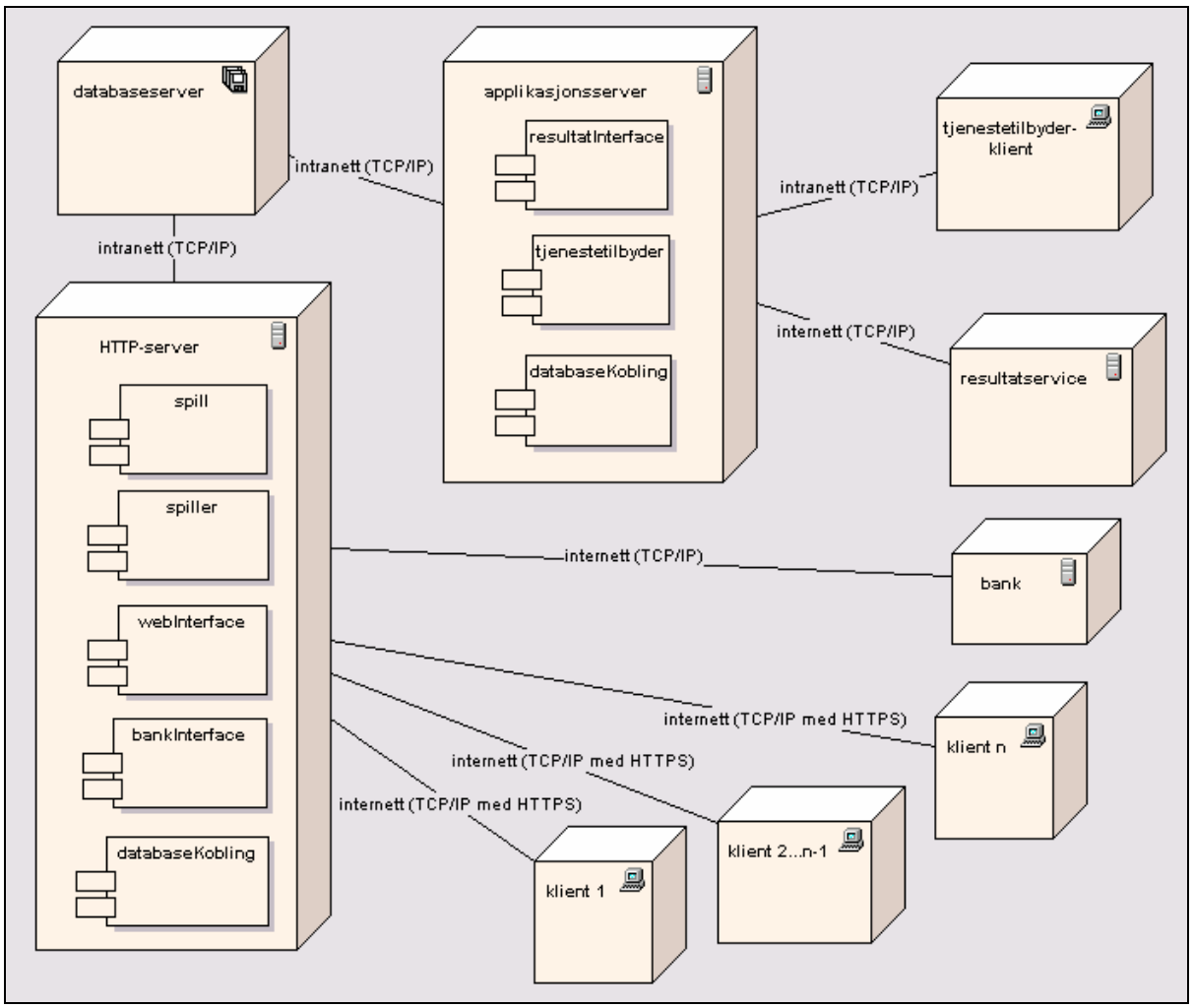
5.1 Deploymentdiagram

Klassediagrammet benyttes for å utvikle programvarekomponenter. Ofte blir disse spredt utover et antall maskiner når applikasjonen er ferdig utviklet, spesielt i distribuerte systemer slik som vår gamlingapplikasjon er et eksempel på.

Et deploymentdiagram benyttes nettopp for å vise hvordan den fysiske lokasjonen og fordelingen av ulike segmenter av systemet blir. Deploymentdiagrammet består av tredimensjonale kuber som representerer fysiske noder i systemet. Nodene er ofte enkeltmaskiner i et nettverk. På hver node ligger det komponenter av programvare, men i deploymentdiagrammet vises kun komponenter som inngår i vår applikasjon. Et ikon i kubenes høyre hjørne angir typen node; databaseserver, server eller klient-PC.

Vårt system er i høy grad et distribuert system, og derfor vil mye informasjon gå over Internett og intranett. Deploymentdiagrammet viser disse transmisjonskanalene som koblinger mellom nodene, påskrevet protokoll og nettverkstype.

Mange av komponentene i deploymentdiagrammet kan man finne igjen i klassediagrammet som selvstendige klasser, andre komponenter er samlinger av klasser.



Figur 23 – Deploymentdiagram

Som det fremgår av deploymentdiagrammet vil hovedvekten av applikasjonens komponenter og funksjonalitet ligge på HTTP-serveren. Disse komponentene vil generere dynamiske Web-sider som spilleren (klient 1 til *n* på diagrammet) vil kunne laste ned via sin nettleser. Det er av høyeste viktighetsgrad at det ikke er mulig å avlytte kommunikasjonen mellom klienten og HTTP-serveren, da det her blir overført mye sensitiv informasjon som for eksempel brukernavn, passord og kredittkortopplysninger. Derfor benyttes sikker, SSL-kryptert HTTP, også kjent som HTTPS. HTTP-serveren er koblet opp mot databaseserveren på et intranett, da innbrudd på databaseserveren fort kan være fatalt for bedriften.

I systemet har vi også en applikasjonsserver. På denne serveren ligger funksjonalitet som tjenestetilbyderen aksesserer via en klient. De komponentene det her er snakk om tar seg av ting som oddssetting og brukeradministrasjon. Av sikkerhetsgrunner har vi valgt å ikke tillate tjenestetilbyderen aksess til systemet fra et eksternt nett, og som det framgår av diagrammet vil intranettet benyttes.

Det ligger ingen komponenter fra vårt system på spillernes maskiner (klient 1 til *n*). Dette er fordi spilleren kommer til å benytte seg av en nettleser for å kjøre komponentene som ligger på HTTP-serveren. Komponentene blir aktivisert når spilleren forespør spesifikke, dynamiske Web-sider som for eksempel spillinformasjon, spillsatsning og brukerprofilendring. På diagrammet er det heller ikke tegnet inn noen komponenter på tjenestetilbyderens klient. I realiteten vil det være en liten komponent her som er et grensesnitt opp mot komponentene på applikasjonsserveren.

6 Sikkerhetsanalyse

6.1 Uønskede hendelser og årsaksanalyse

6.1.1 Trusler

Figurer og stoff er hentet fra [Jones02]. Vi deler her trusler inn i naturlige trusler og i angripere. De naturlige truslene er listet opp som følger:

Tabell 3 - Naturlige trusler

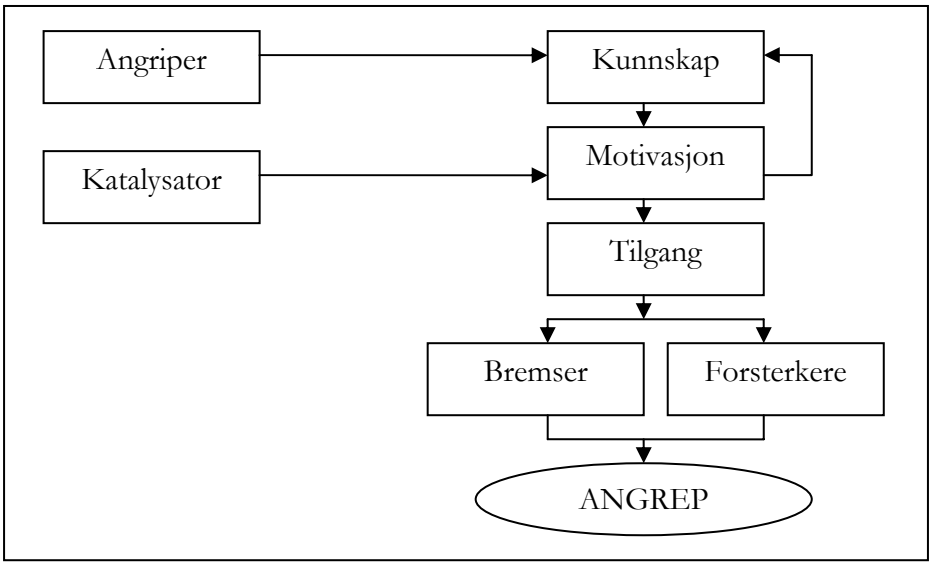
Naturlige trusler	Forklaring
Brann	Dette kan ha direkte ødeleggende effekt på systemet, men man har gode erfaringer innen sikring vha alarmer og utstyr.
Vind	De ødeleggende effektene avhenger svært av geografiske forhold, men man har bra erfaring i sikring.
Vann	Sannsynligheten for flom og oversvømmelse er minimale, og man har gode rutiner for sikring.
Ulykker	Systemet kan utsettes for uhell og ulykker ved uheldig bruk. Konsekvensene av uhell kan variere stort, og sannsynligheten for uhell vil være avhengig av personalets holdning.

For at et ødeleggende angrep skal finne sted må det finnes en angriper som må være motivert til å utføre og være i stand til å gjennomføre et angrep. Dette vil avhenge av følgende:

Tabell 4 - Forhold som fører til angrep

Faktorer	Forklaring
Kunnskap	Hvor dyktig en person eller en gruppe er, vil variere med kompleksiteten av angrepet. Hvilken type angrep angriperen vil komme til å utføre vil avhenge av målet til personen.
Motivasjon	Motivasjonen til å utføre et ødeleggende angrep mot systemet kan komme fra flere forskjellige situasjoner. Det kan være politiske, vedvarende, personlig vinning, religion, hevn, makt, terrorisme eller nysgjerrighet.
Tilgang	For å kunne utføre et angrep må angriperen få tilgang til systemet enten fysisk eller gjennom nettverket. Uten dette kan ikke et angrep utføres.
Katalysator	En katalysator er grunnen for at en angriper velger å utføre angrepet i det tidspunktet som han gjør. En katalysator er noe som enten påvirker angriperen eller målet for angrepet.
Hindringer	Det finnes flere faktorer som vil skremme en angriper fra å utføre angrepet ved et gitt tidspunkt. Dette kan enten være noe som påvirker angriperen eller målet for angrepet. Det kan for eksempel være at systemet er godt beskyttet og alle forsøk på angrep raskt blir oppdaget.
Forsterkere	Det vil være flere faktorer som oppfordrer en angriper til å angripe et visst mål i et gitt tidspunkt. Også dette vil enten være noe som påvirker angriperen eller målet for angrepet. Dette kan for eksempel være at systemet er dårlig beskyttet, og angriperen vil kunne være inne i systemet i lengre perioder uten å bli oppdaget.

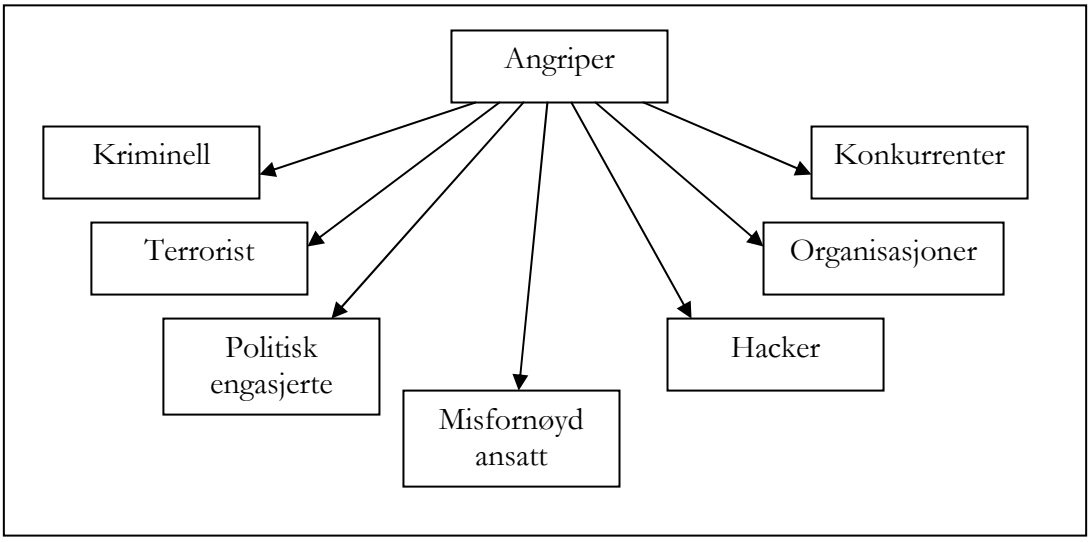
Forholdet mellom disse faktorene kan illustreres med denne tegningen:



Figur 24 – Forhold mellom trusselelementer

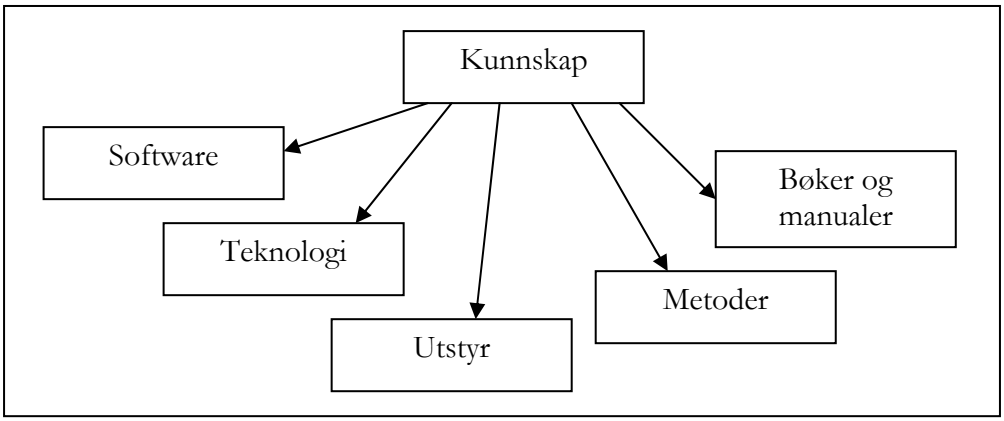
Figuren over viser faktorene som påvirker angriperne til å angripe et datasystem. Dersom en *angriper* skal være en virkelig trussel mot systemet, må han ha *kunnskap* til å utføre angrepet i tillegg til å kunne skaffe seg *tilgang*, enten fysisk eller elektronisk. Hvor mye kunnskap angriperen har vil påvirke hvor stor trussel angriperen kommer til å være. Angriperen vil bli svekket *hindringer* av som svekker hans muligheter til å utføre et vellykket angrep, og han vil styrkes av *forsterkere* som øker sannsynligheten for at angrepet skal bli vellykket. Det vil i tillegg finnes *katalysatorer* som vil gjøre at angriperen til syvende og sist utfører angrepet avhengig av hvilke *motivasjoner* han har. Vi skal nå beskrive de forskjellige faktorene mer i detalj.

Angripere kan deles inn i følgende kategorier:



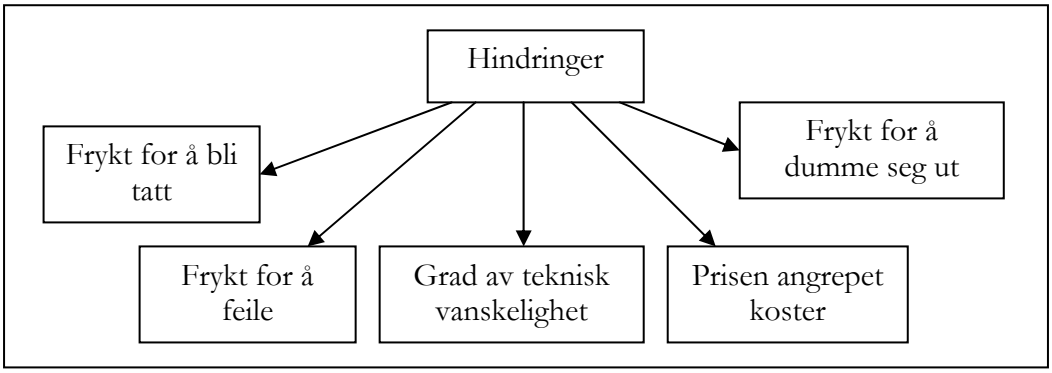
Figur 25 – Angriperkategorier

En angriper kan komme fra hvilken som helst av disse gruppene eller kombinasjoner av gruppene. Kunnskap kan deles inn i følgende tre:



Figur 26 – Angripers kunnskapskategorier

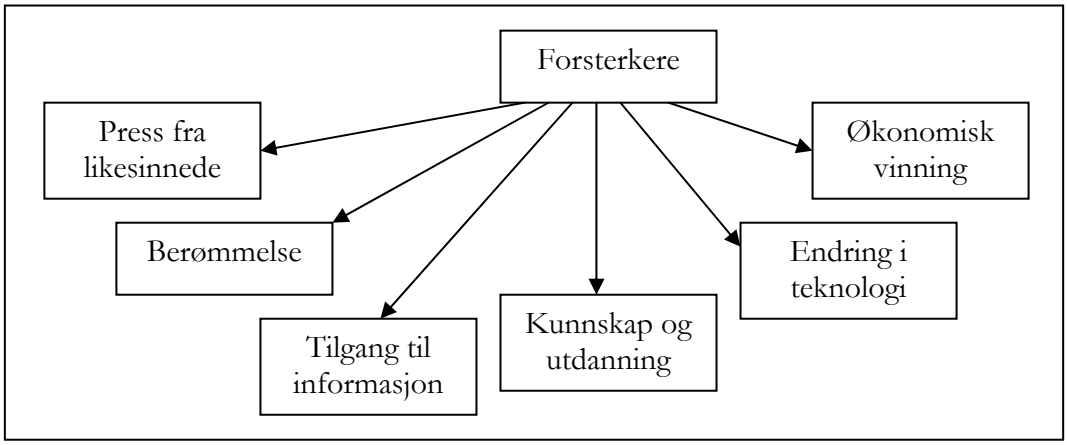
For å kunne utføre et vellykket angrep må angriperen ha de nødvendige kunnskapene til å ødelegge systemet og eventuelle erstatninger.



Figur 27 – Angrepshinder

Hindringer er faktorer som hindrer en angriper i å utføre et vellykket angrep. Enkelte faktorer kan både være styrkende og hemmende. Dette kan være sikkerhetstiltakene som beskytter systemet. Dersom disse er svake vil det være styrkende, dersom de er sterke vil det virke hemmende.

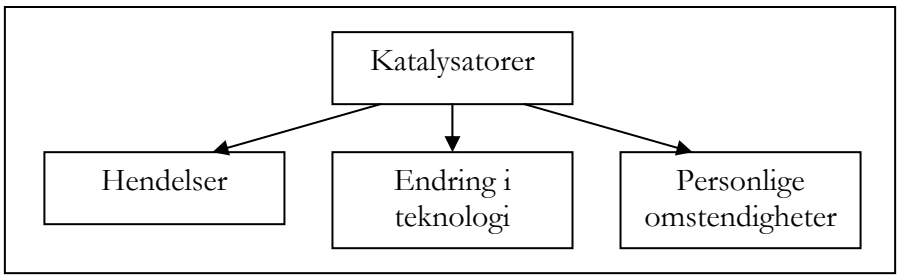
Forsterkere vil øke sannsynligheten for et vellykket angrep, og blir illustrert i treet under:



Figur 28 – Forsterkende omstendigheter for angrep

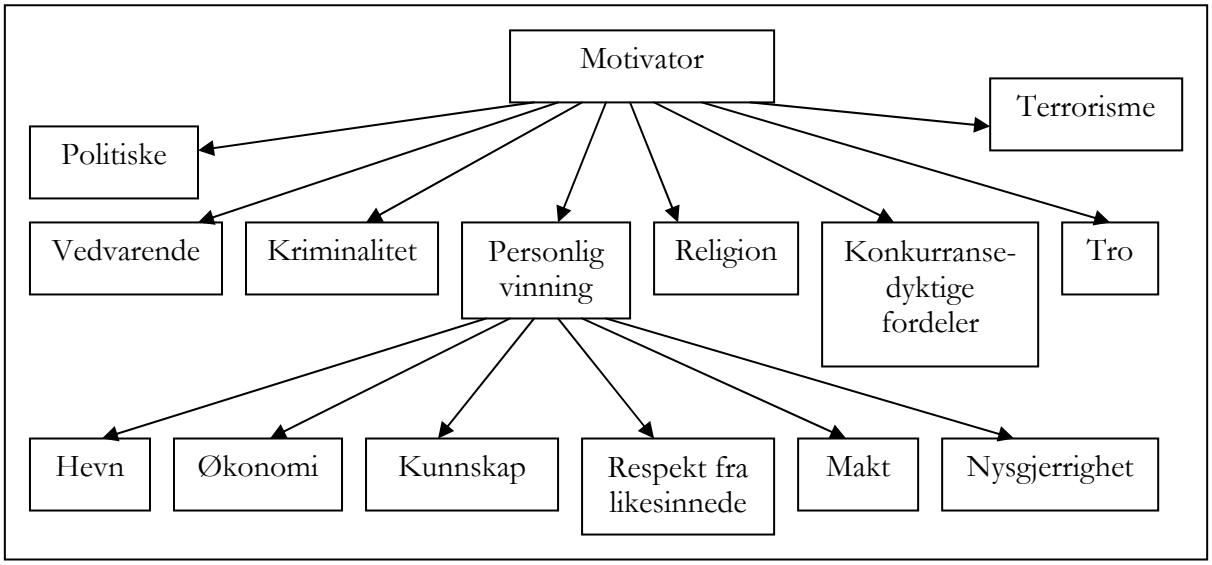
Hvilke av disse faktorene som vil forsterke muligheten for et vellykket angrep varierer, men faktorer som press fra likesinnede vil være inkludert. Her ønsker angriperen å vinne respekt fra sine likesinnede ved å utføre et vellykket angrep. Graden av utdanning og kunnskap en angriper

er i besittelse av, eller kan få tilgang til vil styrke selvtilliten til angriperen og også øke sannsynligheten for suksess.



Figur 29 – Angrepskatalysatorer

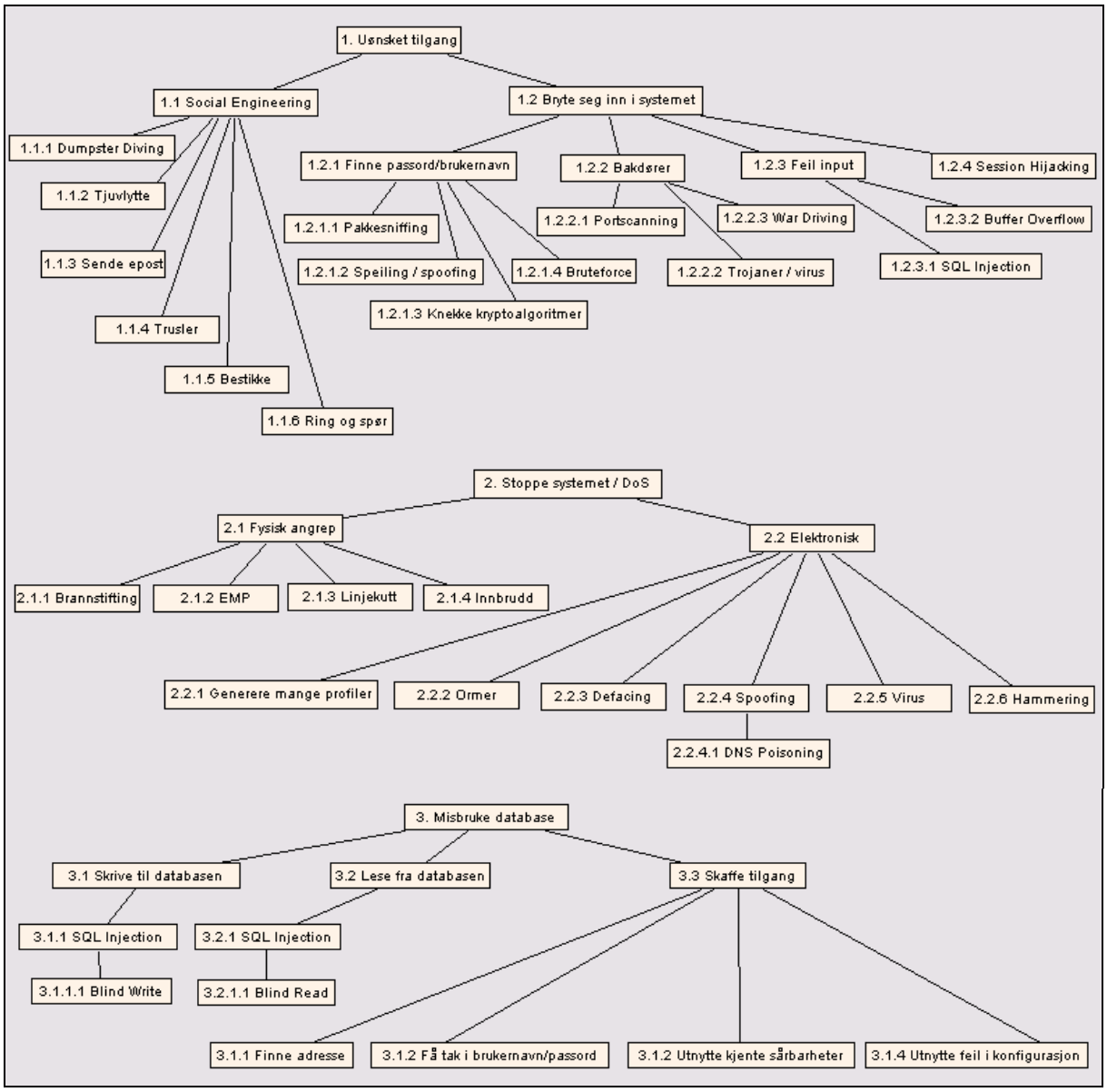
Det som får en angriper til å utføre angrepet på systemet i det tidspunktet angrepet utføres på, kan være avhengig av hendelser vedkommende opplever. Dette kan være at organisasjonen opptrer i media og uttaler noe angriperen er uenig i, eller en konflikt mellom angriperen og organisasjonen. En annen faktor kan være omstendighetene rundt angriperen, og at det gjøres endringer som kan påvirke muligheten for utførelse av et angrep. Et angrep kan også starte pga. at ny teknologi innføres som gir mulighet for å benytte angrepsmetoder som ikke før var mulig.



Figur 30 – Angrepsmotivatorer

Motivatorene er et område som påvirkes av et stort antall faktorer som vil avhenge av grupperingen eller kombinasjonen av grupperinger av områder angriperen påvirkes fra.

6.1.2 Angrepstre



Figur 31 – Angrepstre

*Figur 31 – Angrepstre***Error! Reference source not found.** viser et tre over hvordan en "black hat" kan nå ulike mål i vårt system. Røttene i trærne representer målene for en angriper ("black hat").

Tre 1 viser fremgangsmåter på hvordan en "black hat" kan få uønsket tilgang til vårt system for deretter kunne angripe integriteten og konfidensialiteten i systemet Måten å få tilgang til systemet deles inn i to hovedgrener; social engineering og å bryte seg inn i systemet.

"1.1 Social engineering" er ofte den enkleste måten for en "black hat" å få tilgang. Under social engineering er det mange fremgangsmåter en "black hat" kan benytte seg av. "1.1.1 Dumpster Diving"; lete gjennom søppel fra organisasjonen for å finne verdifull informasjon for å få tilgang til systemet. "1.1.2 Tjuvlytte"; tjuvlytte på telefonsamtaler i bedriften eller simpelthen å overhøre samtaler mellom ansatte. "1.1.3 Sende e-post"; sende e-post å spørre etter viktig informasjon (brukernavn og passord) for å få tilgang. "1.1.4 Trusler"; true til seg brukernavn og passord fra ansatte. "1.1.5 Bestikke"; bestikke ansatte for å tilgang til systemet. "1.1.6 Ring og spør"; ringe ansatte i bedriften å utgi seg for å være en person som har tilgang til å få oppgitt brukernavn og passord.

Den andre grenen (”1.2 Bryte seg inn i systemet”) i treet viser den tekniske måten å få tilgang til systemet på. ”1.2.1 Finne passord/brukernavn” deles inn i: ”1.2.1.1 Pakkesniffing”; lytte på trafikk i kommunikasjonskanalen for å finne eventuelle passord og brukernavn som sendes i klartekst. ”1.2.1.2 Speiling/spoofing”; lage falske grafiske brukergrensesnitt som brukere kan bli lurt til å taste brukernavn og passord inn i. ”1.2.1.3 Knekke kryptoalgoritmer”; knekke kryptering som brukes til å lagre/sende brukernavn og passord med for deretter å finne klarteksten fra chiffertekster eller hashverdier. ”1.2.1.4 Bruteforce”; permutere og prøve alle mulige kombinasjoner av tegn for å finne et passord og/eller eventuelt bruke ordlister for å gjøre dette lettere. ”1.2.2 Bakdører” deles inn i: ”1.2.2.1 Portscanning”; skanne systemet for åpne porter som kan gi tilgang til systemet. ”1.2.2.2 Trojaner/virus”; legge inn bakdører eller virus i programmer som installeres på systemet som legger igjen åpninger i systemet. ”1.2.2.3 War driving”; bruke en PC med trådløst nettverkskort for å få tilgang til trådløst LAN i bedriften. ”1.2.3 Feil input” deles inn i: ”1.2.3.1 SQL injection”; legge inn parametere i SQL spørringer får å tilgang til mer enn ønskelig. ”1.2.3.2 Buffer overflow” utnytte sårbarheter i koden til software for å få tilgang til systemet. ”1.2.4 Session hijacking”; vil si å være ta over åpne sesjoner til brukere for å få tilgang til systemet.

Tre 2 viser hvordan en ”black hat” kan hindre tilgjengeligheten til systemet. Vi deler dette målet inn i to hovedgrener; fysisk og elektronisk angrep.

”2.1 Fysisk angrep” er den første grenen i dette treet. Her vises hvordan en ”black hat” fysisk kan hindre tilgjengeligheten til systemet. ”2.1.1 Brannstifting”; vil si fysisk å sette fyr på systemet (servere). ”2.1.2 EMP”; vil si å bruke elektromagnetisk stråling for å ødelegge hardware i systemet. ”2.1.3 Linjekutt”; er å fysisk kutte kommunikasjonskanaler til/fra systemet. ”2.1.4 Innbrudd” vil si å bryte seg inne å stjele hardware som er systemet avhenger av.

Den andre hovedgrenen i å hindre tilgang til systemet er ”2.2 Elektronisk”. Altså hvordan en ”black hat” kan hindre tilgang for brukere elektronisk. ”2.2.1 Generere mange profiler”; vil si å produsere mange brukerprofiler slik at systemet blir overkjørt av prosessering og data. ”2.2.2 Ormer”; legge inn ondsinnede programmer som kopierer seg selv rundt i systemet og tar opp store ressurser. ”2.2.3 Defacing”; endre internettsiden til systemet slik at den ikke kan brukes av spillerne. ”2.2.4 Spoofing” og ”2.2.4.1 DNS Poisoning”; endre Internett-adressen til systemet slik at spillere ikke får tilgang til systemets korrekte Internettsider eller server. ”2.2.5 Virus”; legge inn virus i systemet som ødelegger programvare eller hindrer programvare i å fungere. ”2.2.6 Hammering”; oversvømme systemet med trafikk fra Internett slik at det blir utilgjengelig for brukerne.

Tre nummer 3 viser hvordan en angriper kan misbruke den sikkerhetskritiske databasen i systemet. Treet deles inn i tre hovedgrener: skrive til, lese fra og skaffe tilgang til databasen.

”3.1 Skrive til databasen” sier hvordan en ”black hat” kan skrive uønsket innformasjon til databasen uten tilgang. ”3.1.1 SQL Injection” vil si å legge inn ekstra informasjon i databasen vha. SQL setninger (`INSERT`/`UPDATE`). ”3.1.1.1 Blind Write”; legge til SQL utrykk til databasekall slik at man får skrevet inn i tabeller man ikke har tilgang til.

Den andre grenen (”3.2 Lese fra databasen”) sier hvordan en ”black hat” kan lese fra databasen. ”3.2.1 SQL injection” vil si å lese ekstra informasjon fra databasen vha. SQL setninger (`SELECT`). ”3.2.1.1 Blind write”; legge til SQL utrykk i databasespørringer slik at man får mer informasjon enn hva den opprinnelige SQL spørringen skal gi.

Den siste grenen i å misbruke databasen er ”3.3 Skaffe tilgang”. Det vil si å få tilgang til tabeller og informasjon som det ikke er meningen at man skal kunne ha. ”3.1.1 Finne adresse”; finne databaseserverens IP-adresse slik at man kan logge seg på fra en annen maskin (krever også 3.1.2). ”3.1.2 Få tak i brukernavn/passord”; finne brukernavn og passord til administrator av databasen. ”3.1.2 Utnytte kjente sårbarheter”; vil si å bruke kjente/ukjente sårbarheter i database Software for å få tilgang. ”3.1.4 Utnytte feil i konfigurasjon”; bruke feilkonfigurasjon til å få tilgang til databasen.

6.2 Systemets sikkerhetskritiske deler

Vi ser umiddelbart at all kommunikasjon mot bank er et viktig område mhp sikkerhet. Sikkerheten i denne kommunikasjonen er imidlertid en del av bankens tjenester, og ikke noe systemet vårt trenger å implementere. Systemet vårt må på den andre siden ha grundige kontroller som hindrer penger i å overføres innad i systemet dersom transaksjonene ikke godkjennes av banken. Grensesnittet mot banken må sikres slik at en angriper ikke klarer å sende falske godkjenninger tilbake til systemet og på denne utføre transaksjoner.

Databasen vil derfor antakelig være den mest utsatte delen av vårt system mht sikkerhet. En angriper må ikke klare å få tilgang til databasen og informasjonen som ligger lagret i denne. Dersom en angriper klarer å endre data her, vil dette forplante seg utover i hele systemet. En angriper må ikke klare å endre informasjonen som ligger i databasen. Systemet må ha sikkerhetsrutiner som hindrer spillere i å kunne spille uten å satse penger, eller å spille fra andre spillere sine kontoer.

Vi må også sikre brukerinnloggingen og brukerregistreringen, så en angriper ikke kan sende egne SQL-kall gjennom denne. Administratoren skal ha et eget system for å administrere og overvåke systemet. Det skal m.a.o. ikke gå an å logge seg inn som administrator fra samme grensesnitt som spillerinnloggingen, og en angriper vil dermed ikke klare å få administratortilgang uten dette eksterne systemet.

Vi må ha en sikker kommunikasjonskanal mot oddsanalytikeren, så en angriper ikke kan utgi seg for å være denne, for så å registrere falske odds og spilldata. I likhet med administrator benytter også oddsanalytikeren seg av et eget system for å legge inn odds i databasen.

Siden det er systemet som ber om resultater fra resultatservice vil dette vanskeliggjøre angrep denne veien. Sikkerheten vil her avhenge av resultatservice sin sikkerhetspolicy og troverdighet.

Ut fra deploymentdiagrammet ser vi at det er tre servere som må beskyttes. Disse er databaseserveren, HTTP-serveren og applikasjonsserveren. Her vil HTTP-serveren være den mest utsatte, siden det er denne som inneholder grensesnittet mellom spilleren og systemet.

6.2.1 Kommunikasjonskanaler

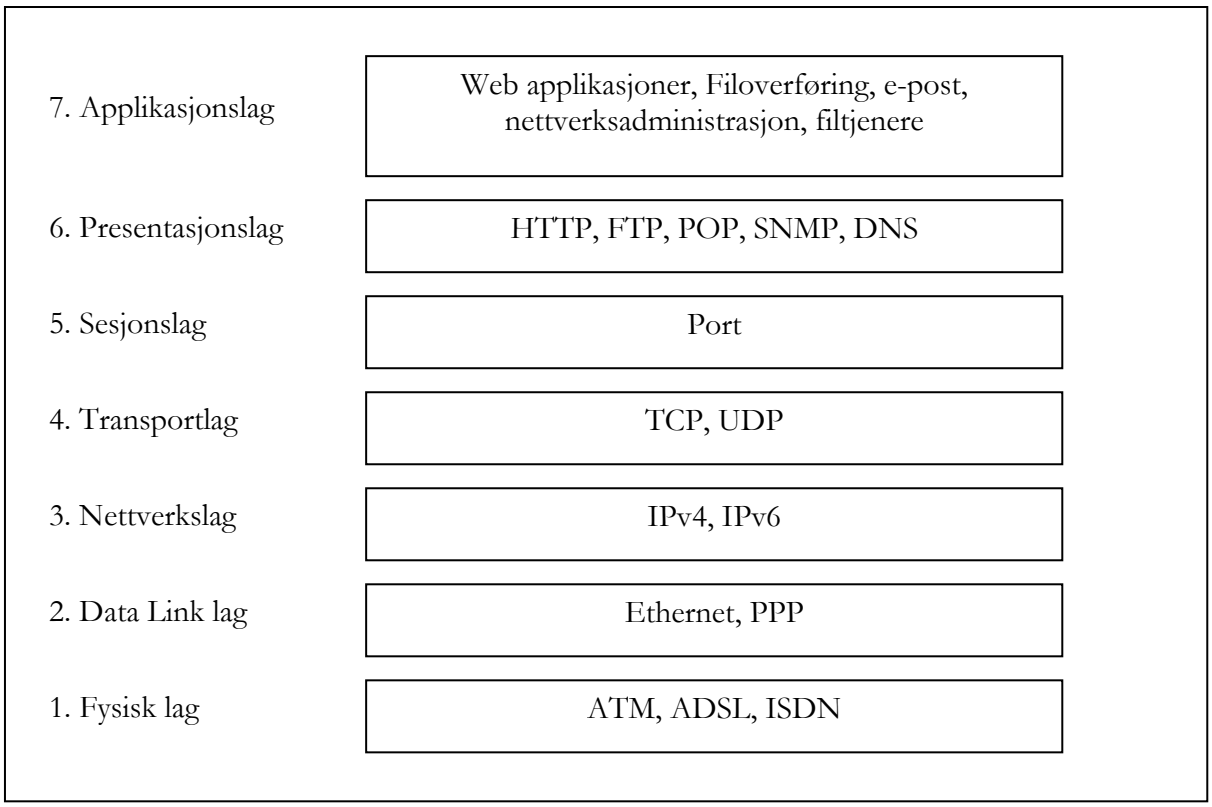
Da systemet vårt er et distribuert system med mange noder vil våre kommunikasjonskanaler både internt og til omverdenen være sårbare. Dette kan vi se helt klart fra vårt diagram i engineering viewet.

Under beskrives i mer detalj noen av de største sikkerhetshullene og -problemene man kan støte på angående implementasjon av våre valgte kommunikasjonskanaler.

6.2.1.1 TCP/IP

Mye av kommunikasjonen i systemet går over TCP/IP forbindelser. Sikkerheten i denne protokollen blir derfor en viktig del av den samlede sikkerhetsanalysen. OSI-modellen under

illustrerer hvordan TCP/IP er grunnlaget for sikkerhetsproblemene i de protokollene vi har forklart nedenfor.



Figur 32 – Oppbygning av OSI-modellen

Selv om utviklingen av TCP/IP ble sponset av Department of Defense, er det mange kjente sikkerhetsblemmer i denne protokollen. Noen av disse kommer av at hostene stoler på IP-adresser som autentifisering.

Noen av de mest kjente sikkerhetsproblemene i TCP/IP:

Forutseing av TCP sekvensnummer:
Hvis man klarer å forutse sekvensnummeret til TCP sekvensen, kan man bruke dette til å spoofe en maskin på det lokale nettverket.

Det er derfor viktig at disse sekvensnumrene ikke genereres sekvensielt, men at man bruker en kryptografisk algoritme til å generere dem.

Source-ruting:
Den letteste måten å utnytte IP på er source-ruting. Det vil si at man snapper opp en TCP open request, og satser på at neste IP pakke rutes på samme måte tilbake til nettverket. Angriperen kan bruke denne informasjonen til å velge en IP på nettverket som tilhører en pålitelig maskin. Lykkes dette, kan angriperen få samme muligheter på systemet som maskinen på det lokale nettverket.

Forsvar mot denne type angrep kan være å ikke sette fast traversering gjennom Internett.

Angrep mot RIP (Routing Information Protocol):

RIP blir brukt for å spre ruting informasjon på det lokale nettverket, spesielt broadcast meldinger. Informasjon som blir sendt ut blir vanligvis ikke kontrollert. Dette kan en angriper utnytte til å sende falsk rutinginformasjon til en maskin på nettverket, og oppgi falsk rute slik at pakkene blir sendt til angripers maskin istedenfor rett maskin.

Slike angrep kan enkelt beskyttes ved at man bruker en gateway som filtrerer IP pakker basert på avsender- eller mottakeradresse.

Angrep mot EGP (Exterior Gateway Protocol):
EGP benyttes for kommunikasjon mellom indre- og ytre gatewayer. Et mulig angrep vil basere seg på å utgi seg for å vær en annen ytre gateway. Dette kan forhindres ved at de indre gatewayene har en liste over lovlige ytre gatewayer de kommuniserer med.
En annen type EGP angrep går ut på at man kan utnytte at en gateway er nede og si at din gateway er tigjengelig.

Angrep mot ICMP (Internet Control Message Protocol):
ICMP redirect meldingen brukes til å si fra om alternative ruter, men vanskeligheten ligger i at redirect meldingen må bindes til en spesiell tilkobling i motsetning til RIP.
ICMP kan også brukes til rettede DoS (Denial of Service) angrep, ved å bruke ICMP meldingene destiantion unreachable og time to live exceeded.

For å beskytte seg mot denne typen angrep, må man være nøye med å sjekke at ICMP meldingen virkelig tilhører en tilkobling.

E-post:
Elektronisk post er sannsynligvis den mest sårbare tjenesten på Internett som bygger på TCP/IP. Noen e-postservere har open relaying, og man kan derfor ikke stole på at e-post kommer fra den man umiddelbart tror den kommer fra. Verken autentisering mot POP-3 serveren eller innholdet i en e-post sendes kryptert, det er derfor ikke vanskelig å lese av innhold i en e-post eller snappe opp passordet, om man får tak i en slik pakke.

Ved nyregistrering i vårt system, vil brukerne få tilsendt en e-post med et automatisk generert passord. Da e-post går ukryptert over Internett, vil dette være et forholdsvis stort sikkerhetshull i vår applikasjon. En måte å forhindre dette på er ved bruk av offentlig nøkkelkryptografi og signering av e-post, men det er urealistisk at alle våre brukere vil ha nok datateknisk kompetanse for å kunne bruke slike løsninger. Som en liten sikkerhetsforanstaltning vil ikke e-posten med passordet inneholde brukernavnet til spilleren. Dette for å vanskeliggjøre at en angriper lett kan skaffe seg både brukernavn og passord, og dermed logge inn på systemet.

Alternativ til denne løsningen er å tvinge bruker til å sette nytt passord ved første gangs pålogging på systemet. Brukeren vil da være inne på en HTTPS side, og er sådan beskyttet. Brukeren har ved dette tidspunket ingen penger på lokal spillekonto, og man risikerer ingenting om en angriper får tak i det første passordet som blir sendt på e-post.

DNS (Domain Name System)
DNS er en distribuert database prosess som mapper maskinnavn til IP adresser. En inntrenger som spoofer DNS serveren, kan utnytte dette til en mengde angrep. Dette kan være DoS angrep, eller innsamling av passord ved at man setter opp en falsk side.
Man jobber med systemer for å løse dette problemet med kryptografi, vad at både forespørselen og svaret bærer en kryptert nøkkel, og disse brukes til å generere en sjekksum.

SNMP (Simple Network Management Protocol)
Dette er en protokoll som er laget for å hjelpe nettverksadministrasjon, og brukes til å sende kontrollmeldinger mellom de ulike komponentene i nettverket. Tilgang til å sende denne typen meldinger må beskyttes nøye.

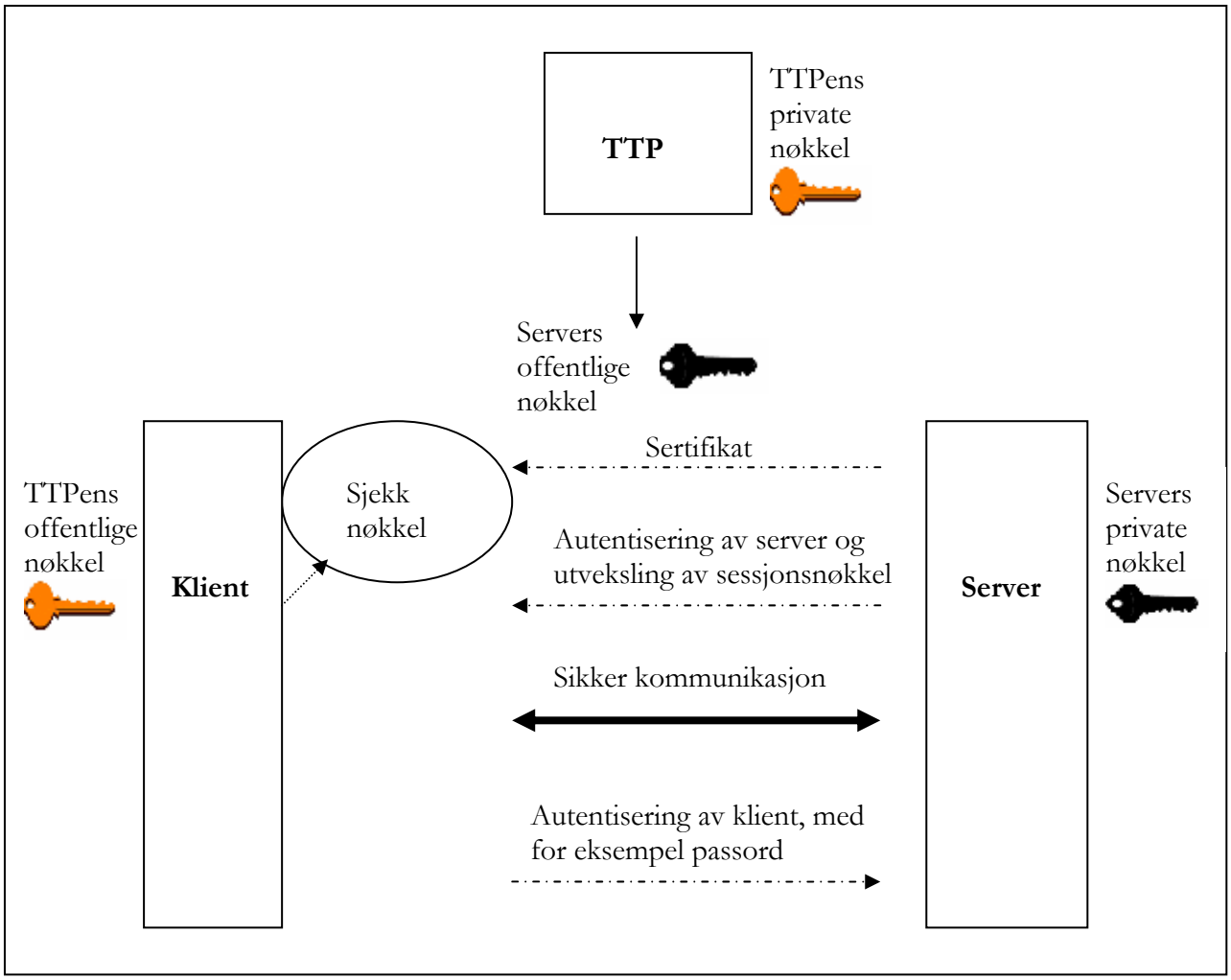
6.2.1.2 VPN og IPSec

Sikkerhet på IP-nivå er viktig for arkitekturen for alle internettbaserte sikkerhetssystemer. IPSec, IP Security, er utviklet for IPv6 men er også kompatibel med IPv4. Den er definert i [Kent98], er mer lik en sikkerhetsarkitektur enn en protokoll og ble laget for å støtte høykvalitets kryptobasert sikkerhet for IPv4 og IPv6.

Sikkerhetsmuligheter som er inkludert i IPSec er bl.a. aksesskontroll, integritetskontroll, kontroll av datas opprinnelse og konfidensialitet. IPSec kan integreres i servere, brannmurer og rutere. Den er nedenfor transportlaget i OSI-modellen (Open System Interconnection) og kan derfor gjøres transparent for brukere og applikasjoner. IPSec er protokollen som i størst grad benyttes både til autentisering av tunnel og til kryptering i VPN, den ble altså laget for å være algoritmeuavhengig og støtter derfor flere krypterings- og autentiseringsalgoritmer. Dette gjør at selskaper som skal benytte VPN selv kan velge hvilket nivå av sikkerhet som skal implementeres og også hvilke grad av overhead man får på nettet sitt. Gamblingselskapet og banken kan dermed selv bestemme hvilke algoritmer som skal benyttes for de forskjellige delene. Dette bestemmes før oppstart av tunnelen, og kalles autentiseringsfasen. Tunnelen er en virtuell linje som rutes gjennom Internett, og som gjør at nettverket virker transparent for brukeren. Før autentiseringsfasen "forhandler" de to endepunktene om hvilke nivå av sikkerhet som skal kjøres, ved at de begge parter har et minstekrav. De har sannsynligvis også algoritmer av høyere nivå på sikkerhet, og man velger den med høyest sikkerhet som begge støtter. Oppnås ikke minstekrav til en av partene kommer man aldri til autentifiseringsfasen og tunnelen blir aldri opprettet.

6.2.1.3 HTTP med SSL

HTTPS er HTTP-protokollen over en sikker SSL (Secure Socket Layer)/TLS (Transport Layer Security) -forbindelse. TLS er en protokoll som skal gi kryptering og sertifisering på transportlaget, slik at data kan gå gjennom en sikker kanal uten betydelige endringer programmer på klient og server. Forgjengeren til TLS, SSL, bruker privat nøkkel for å kryptere data og beskytte sensitiv informasjon. Ved ønske om kommunikasjon mellom nettleser og server vil klienten opprette en TCP-forbindelse til server og sjekke servers sertifikat som blir returnert. Dette sertifikatet inneholder informasjon om organisasjonen, servernavn, utløpsdato og er signert av en TTP (Trusted Third Party) som nettleseren stoler på. Under kan man se hvordan autentisering ved hjelp av TTP og sertifikater foregår.



Figur 33 – SSL/TLS

Ved feil i sertifikatet vil brukeren få beskjed fra nettleseren om at noe er galt, mange brukere vil ikke forstå meldingen og trykke OK eller akseptere feilen for å komme seg videre. Dette kan åpne for man in the middle attack som lager sitt eget sertifikat som ser gyldig ut og som da får tilgang til all trafikk mellom bruker og server som for eksempel brukernavn og passord til tjenesten. Dette innebærer også mulighet for endring av data sendt mellom server og nettleser, som for eksempel å endre kontonummer det skal utbetales til. All trafikk mellom server og nettleser overføres kryptert slik at sensitiv informasjon overført over HTTPS er beskyttet mot innsyn. Her brukes sterk kryptering og det vil være vanskelig for en tredjepart å få innsyn eller endre data under kommunikasjonen uten å ha kontroll over et endepunkt, servermaskinen eller maskinen hvor nettleseren kjører.

Ved oppsett av HTTPS-serveren må det lages et sertifikat og private og offentlige servernøkler. De private servernøklene må beskyttes slik at de ikke kommer i hendene på uvedkommende som da kan få innsyn i all trafikk generert av serveren. Sertifikatet må også sendes til en tredjepart, som de aktuelle nettleserne stoler på, for signering. Dette må som regel fornyes hvert år.

6.2.2 Database

Databaser kan fort bli utsatt for angrep, selv i tilfeller hvor en angriper ikke har fysisk tilgang til databaseserveren. En angriper kan for eksempel utnytte sikkerhetsmangler i Web-grensesnittet som muliggjør kjøring av vilkårlige SQL-setninger. Slike sikkerhetshull kommer alltid av for dårlig sjekking av inputdata fra brukeren, eller inputdata som det antas at brukeren ikke har kontroll

over. Man kaller disse angrepene "SQL Injection Attacks", da de baserer seg på å sette inn fremmed SQL-kode inn i SQL-setninger som blir bygget dynamisk av Web-applikasjonen.

Framgangsmåten er å sette inn en apostrof (') enten inn i en inputboks på Web-siden, eller i URL-en hvor data blir sendt med. Apostrofen kan i tilfeller hvor programmereren ikke har tatt nok sikkerhetsforanstaltninger føre til en syntaktisk feil når SQL-setningen prøves kjørt. Denne feilmeldingen kan en angriper ofte utnytte for å finne ut hva slags relasjoner (tabeller eller views) og kolonner som finnes i databasen, som igjen kan utnyttes hvis angriperen ønsker å sette inn nye data, eller endre allerede eksisterende data.

Databasen er ikke kun sårbar overfor fjernangrep. Hvis en angriper kan komme seg inn på lokalnettverket, vil han kunne ha fysisk kontakt med databasen. Her er det bare et brukernavn og passord som står mellom angriperen og full tilgang til databasen.

For å gjøre både fjern- og lokalangrep vanskeligere, vil det lønne seg å benytte seg av views når data skal hentes fram for visning på Web. Et view er en måte å vise data fra flere forskjellige tabeller i databasen, slik at det for en bruker framstår som en helt vanlig tabell. Bruker man `WITH CHECK OPTION` i SQL-setningen når viewet opprettes vil ikke en bruker kunne skrive data til noen felter i viewet, for eksempel via Injection Attacks.

Med et view kan man altså med enkle grep hindre en Web-bruker, spilleren, i å få tilgang til de relasjonene som viewet er basert på, og dermed forhindre innsetting av data. Viewet er derimot fortsatt sårbart for uthenting av mer data enn det som spilleren egentlig skal ha tilgang til, og det er derfor ikke nok å implementere sikkerhetsforanstaltninger på databasenivå, men også på applikasjonsnivå. Siden den Web-baserte delen av applikasjonen skal implementeres med PHP vil det her være utrolig enkelt å beskytte seg mot Injection Attacks, nemlig ved hjelp av funksjonen `addslashes`. Problemet er at det ofte er mange variable som sendes med til SQL-setningen, og absolutt alle disse må sikres vha `addslashes` for at applikasjonen skal være usårbar overfor slike angrep.

I tillegg til å benytte views der dette er mulig, må også databasen deles opp så mye som mulig, slik at sensitiv data ikke står sammen med data som ikke er like sensitivt. Med dette menes at data som spillekontosaldo og navn ikke skal stå i samme tabell, da dette gjør det lettere for en angriper å endre en persons saldo. Det skal ikke påstås at det å dele opp dataene i flere tabeller gjør det veldig mye vanskeligere for en angriper å endre data, men det er et ekstra hinder.

Da det ikke er mulig å være 100% sikker på at databasen ikke kan misbrukes, kan ikke passord lagres i klartekst. Lagrer man spillernes passord i klartekst i databasen, vil også en uærlig administrator kunne få tilgang til en annens profil ved å lese passordene. En angriper som går inn bakveien, vil også kunne lese ut passordene og ikke minst endre passordene. Slik kan en angriper kapre en spillers profil. For å hindre misbruk av passord skal alle passord krypteres ved hjelp av en kryptografisk hashfunksjon før innsettelse i databasen. En hashfunksjon tar en input av vilkårlig lengde (alt fra et passord til hele filer på flere megabytes) og returnerer en unik hashverdi på en bestemt lengde. Da det advares mot å benytte MD5 som kun har 128 bits hashverdi, skal det benyttes SHA1 som har 160 bits hashverdi. Vi kunne også ha brukt den europeisk utviklede RIPMD-160 som også har 160 bits hashverdi, da denne anslagsvis tilbyr like god sikkerhet som SHA1.

Alle databasetransaksjoner må logges slik at man i tilfelle misbruk kan gå tilbake og rekonstruere hendelser basert på loggen. Dette er nyttig både i tilfeller hvor uvedkommende kommer seg inn på databaseserveren, og i tilfeller hvor utro tjenere i bedriften kan endre spilleres spillekonto-

saldo. Det må være automatiserte rutiner for å ta jevnlige sikkerhetskopier av loggen, dessuten skal kun databaseprosessen ha tilgang til å skrive til den, dvs. ingen som bryter seg inn på systemet skal kunne slette eller redigere loggen.

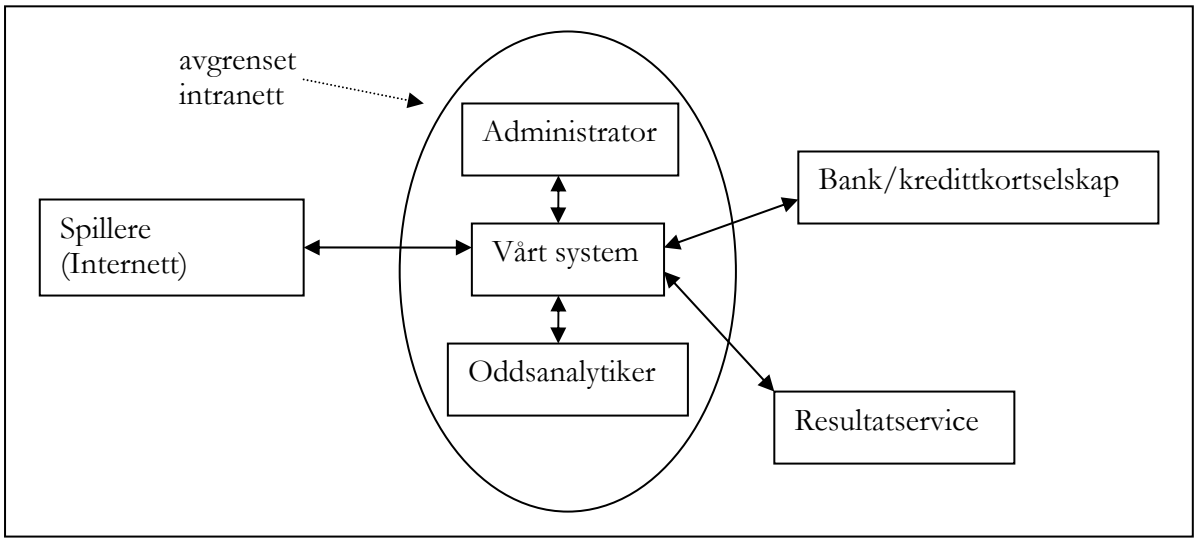
Ved bruk skal systemet benytte seg av fire separate brukerkontoer på databasen. To vil være i bruk av Web-applikasjonen, mens de to resterende vil være i bruk av administrator og oddsanalytiker. Den ene av Web-applikasjonens brukerkontoer skal kun ha leserettigheter på viewet som blir opprettet med informasjon som er nødvendig for at spilleren kan logge på systemet, og slik at spill og informasjon angående brukeren kan vises. Den andre brukerkontoen til Web-applikasjonen skal ha begrenset skrivetilgang til spillerens profilinformasjon, samt tabeller som muliggjør satsning på spill.

I tjenestetilbyderens applikasjoner skal administrator ikke ha skrivetilgang til oddssetting, men kun spillerinformasjon og annen informasjon som er nødvendig for å administrere systemet på en god måte. Likeledes skal oddsanalytikeren ikke ha mulighet til å skrive til tabeller som inneholder data om spillere og deres innsatser og kontobeholdning. Oddsanalytikeren skal kun ha tilgang til å skrive til tabellen som omhandler selve spillene.

Det er altså viktig å ikke gi mer tilgang enn det som strengt tatt er nødvendig.

6.3 Systemets omgivelser

Det er viktig å ha klart for seg hvilke omgivelser et system vil befinne seg i før man setter i gang med implementasjonen. Skal systemet være tilgjengelig via Internett, slik som vårt, må man anta at systemet befinner seg i fiendtlige omgivelser, og man må ta alle mulige sikkerhetsforanstaltninger..



Figur 34 – Systemets omgivelser

Systemet er koblet opp mot to eksterne systemer, se *Figur 34 – Systemets omgivelser*. Disse er banken og resultatservice. Grensesnittet mot disse systemene må være sikret for å hindre angrep denne veien. Koblingen mot resultatservice vil være over Internett via TCP/IP. Det samme vil koblingen mot banken være. Disse koblingene må benytte en sikker løsning.

For å få et godt bilde av sikkerheten til systemet er det ikke nok å bare se på hvilke omgivelser systemet har. Man må også se på omgivelsenes omgivelser, noe som i praksis vil si omgivelsene til banken og resultatservice. Det hjelper ikke om systemet vårt har all verdens sikkerhet dersom

bank eller resultatservice er fullstendig usikret. Siden det faller noe utenfor oppgaven å analysere omgivelsenes omgivelser analyseres, er dette en meget viktig del av et systemutviklingsprosjekt og må gjøres i et reelt tilfelle. Når det gjelder omgivelsene til spillerne må man anta at de sitter på vanlige hjemme-PC-er, som vanligvis er meget sårbare overfor hvilket som helst type angrep. Det vil være fornuftig å gå ut i fra at spillernes PC-er vil være enkle å kontrollere for en angriper.

6.4 Identifiserte risikoer

Her følger en liste over risikoer som er kartlagt i løpet av prosjektperioden, enten gjennom diagrammer og modeller eller gjennom tekstlige sikkerhetsbetraktninger. Noen av disse ble identifisert alt under kapittel 2.5, men de fleste risikoene har blitt identifisert senere i prosjektet. Risikoene identifisert i kapittel 2.5 er merket med stjerne (*).

Tabell 5 - Identifiserte risikoer

Nr	Risikoene omkring spiller
1	Spiller kan oppgi falske personopplysninger, for på denne måten å skjule identiteten sin
2	Spiller kan stjele eller bli frastjålet brukernavn og passord*
3	Spiller kan forfalske e-postadressen sin, eller lese andres e-post for å stjele andre spilleres passord

Nr	Risikoene omkring systemet
4	Systemet kan ha feil i sikkerhetstiltakene
5	Det kan utføres social engineering mot de ansatte
6	Utro tjenere kan bevisst legge inn sikkerhetshull i software
7	Det kan utføres angrep mot systemets programvare og infrastruktur
8	Systemet kan registrere feil beløp fra brukerkonto ved spill
9	Systemet kan settes ut pga. strømbrudd, brudd i kommunikasjonslinjer eller DoS-angrep*

Nr	Risikoene omkring oddsanalytiker
10	Oddsanalytiker kan bli frastjålet brukernavn og passord, og dermed kunne sette egne odds*
11	Oddsanalytiker kan bevisst eller ubevisst legge inn feil odds eller spilldata

Nr	Risikoene omkring administrator
12	Administrator kan bli frastjålet brukernavn og passord*
13	Administrator kan bevisst misbruke sine administratorprivilegier*
14	Administrator kan bevisst eller ubevisst registrere feil i spillerkontoer eller registrere feil spillinformasjon

Nr	Risikoene omkring resultatservice
15	Resultatservice kan legge inn feil resultater bevisst eller ubevisst*
16	Resultatene fra resultatservice kan manipuleres under overføring eller i ettertid
17	Det kan oppstå feil på linjen under overføring av resultater

Nr	Risikoene omkring banken
18	Banken kan motta transaksjoner det ikke er dekning for, eller ikke blir godkjent av andre grunner
19	Banken kan utbetale feil beløp eller til feil bankkonto
20	Utbetalinen fra banken kan bli forsinket
21	Banken kan motta en forfalsket ordre om utbetaling

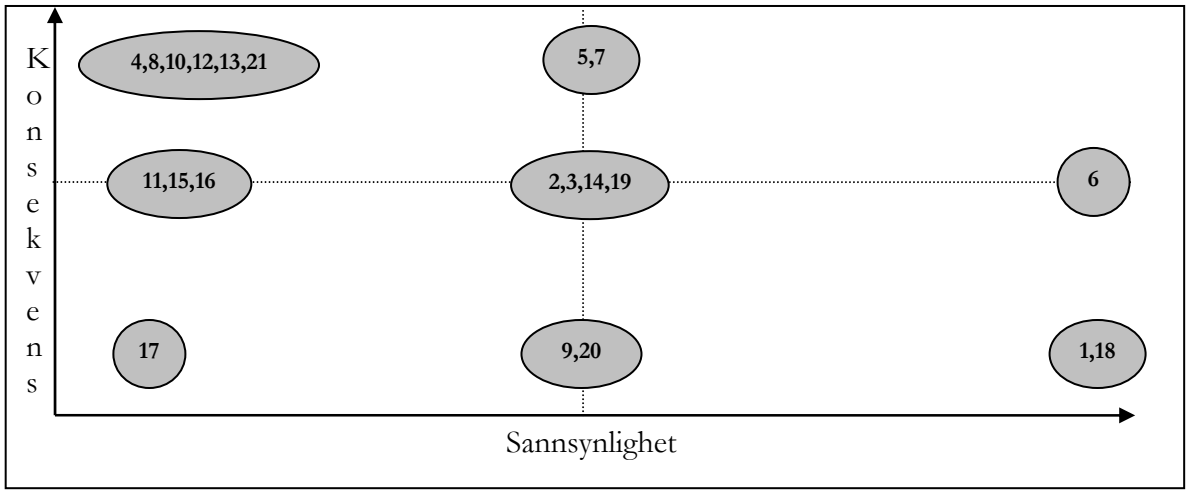
6.5 Rangering av risikoer

Vi følger risikonivåene definert i kapittel **Error! Reference source not found.**, og bruker denne definisjonen til å samle de forskjellige risikoene i tabellen under

Tabell 6 - Rangering av risikoer

	Sannsynlighet			Konsekvens		
Spiller	**Høy**	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
1. Falske spilleropplysninger	√					√
2. Stjele spillers brukernavn/passord*		√			√	
3. Spoofing/sniffing av e-post		√			√	
System	**Høy**	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
4. Feil i sikkerhetstiltak			√	√		
5. Social engineering		√		√		
6. Sikkerhetshull	√				√	
7. Angrep mot SW/infrastruktur		√		√		
8. Registrering av feil beløp			√	√		
9. Strømbrudd/DoS*		√				√
Oddsanalytiker	**Høy**	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
10. Bli frastjålet brukernavn/passord*			√	√		
11. Registrering av feil data/odds			√		√	
Administrator	**Høy**	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
12. Bli frastjålet brukernavn/passord*			√	√		
13. Misbruke privilegier*			√	√		
14. Registrere feil informasjon		√			√	
Resultatservice	**Høy**	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
15. Registrere feil resultater*			√		√	
16. Resultater manipuleres			√		√	
17. Feil på linjen			√			√
Bank	**Høy**	**Middels**	**Lav**	**Høy**	**Middels**	**Lav**
18. Motta ugyldige transaksjoner	√					√
19. Feil utbetaling		√			√	
20. Forsinket utbetaling		√				√
21. Motta forfalsket transaksjon			√	√		

Disse risikoene kan nå samles i en ny Sannsynlighet-Konsekvensgraf. Dette vises i figuren under



Figur 35 – Sannsynlighet-Konsekvensgraf

Vi ser av grafen at vi har 6 tilfeller med lav sannsynlighet og høy konsekvens. Disse kan få alvorlige følger for systemet og organisasjonen dersom de inntreffer for ofte, og det bør derfor iverksettes sikkerhetstiltak for å hindre disse. Vi ser også at vi har 4 tilfeller med middels sannsynlighet og konsekvens. Også disse bør det iverksettes sikkerhetstiltak mot, men tilfellene med høy konsekvens bør prioriteres. Vi ser at vår design av systemet har unngått tilfeller med høy sannsynlighet og konsekvens. Dette er veldig bra, men hvorvidt dette virkelig er reelt er vanskelig å si med vår erfaring, og uten å faktisk implementere systemet. Tilfellene med lav konsekvens er ikke noe som trengs å prioriteres samme hvor sannsynlig det er at de inntreffer. De vil uansett alene ikke skape store ødeleggelser for bedriften, men kunne være med på å kunne gi bedriften et dårlig omdømme dersom de ses sammen med andre tilfeller med høyere konsekvens.

Tabell 7 - Rangert risiko

Pri	Risikoer
1	5. Social engineering
2	7. Angrep mot SW/infrastruktur
3	4. Feil i sikkerhetstiltak
4	8. Registrering av feil beløp
5	10. Bli frastjålet brukernavn/passord* (oddsanalytiker)
6	12. Bli frastjålet brukernavn/passord* (administrator)
7	13. Misbruke privilegier*
8	21. Motta forfalsket transaksjon
9	6. Sikkerhetshull
10	2. Stjele spillers brukernavn/passord*
11	19. Feil utbetaling
12	14. Registrere feil informasjon
13	3. Spoofing/sniffing av e-post
14	11. Registrering av feil data/odds
15	15. Registrere feil resultater*
16	16. Resultater manipuleres
17	1. Falske spilleropplysninger
18	9. Strømbrudd/DoS*
19	18. Motta ugyldige transaksjoner
20	20. Forsinket utbetaling
21	17. Feil på linjen

Det er ekstremt vanskelig å måle risikoer opp mot hverandre, da både sannsynligheten og konsekvensen må tas med i beregningen. Vi ser derfor heller ingen grunn til å kvantifisere risikoene nærmere. Dette gjør at denne listen må ses på som veiledende, ikke bestemmende. Listen baserer seg på *Figur 35 – Sannsynlighet-Konsekvensgraf*.

6.6 Identifiserte mottiltak

6.6.1 Mottiltak mot angrep i angrepstreet

Tallene i *Tabell 8 - Mottiltak* refererer til indekseringen i *Figur 31 – Angrepstre*. Mottiltakene er oppført, tabulatorisk og beskriver kort de enkelte mottiltak.

Tabell 8 - Mottiltak

1 Uønsket tilgang	
1.1 Social Engineering	Generelt (gjelder alle undernoder): Veldefinert sikkerhetspolicy og opplæring i bruk av denne.
1.1.1 Dumpster diving	All sensitiv informasjon skal makuleres før kasting
1.1.2 Tjuvlytte	Begrens sensitiv prat på åpne kanaler.
1.1.3 Sende epost	Ha policy på å ikke sende e-post med sensitiv informasjon.
1.1.4 Trusler	Ikke oppgi for mye informasjon om ansatte offentlig. Pass på at ingen person sitter med all informasjon angående systemet og dets virkemåter.
1.1.5 Bestikke	Ikke oppgi for mye informasjon om ansatte offentlig.
1.1.6 Ring og spør	Ha policy på å ikke oppgi senestiv informasjon over telefon.
1.2 Bryte seg inn i systemet	Generelt(gjelder alle undernoder): holde kompleksiteten til systemet på et minimum
1.2.1 Finne passord og brukernavn	Generelt (gjelder alle undernoder): bruk sikre krypteringsalgoritmer.
1.2.1.1 Pakkesniffing	Bruke IPsec og HTTPS.
1.2.1.2 Speiling/spoofing	Digitale signaturer.
1.2.1.3 Knekke kryptoalgoritmer	Bruke sikre krypteringsalgoritmer med lang nøkkel
1.2.1.4 Brute force	Sperre kontoer etter et gitt antall mislykkede forsøk. Three finger salute. Ha lang nøkkel i krypteringsalgoritmer.
1.2.2 Bakdører	
1.2.2.1 Portscanning	Ingen andre åpne porter enn nødvendig. Sikre tjenester på porter som er i bruk.
1.2.2.2 Trojaner/virus	Bruke signerte og korrekte filer fra leverandør. Integritetssjekkede pakker.
1.2.2.3 War driving	Bruke MAC adresser til å identifisere klienter i WLAN. Eventuelt unngå å bruke WLAN.
1.2.3 Feil input	Generelt (gjelder alle undernoder): Kontrollere input.

1.2.3.1 SQL injection	
1.2.3.2 Buffer overflow	Ikke bruke funksjoner uten grensesjekking.
1.2.4 Session hijacking	Bruke lange og tilfeldige sesjons ID'er.

2 Stoppe systemet/DOS	Generelt (gjelder alle undernoder): Ha backupsystem (Speiling av systemet).
2.1 Fysisk angrep	Generelt (gjelder alle undernoder): Ha god fysisk sikring av serverpark.
2.1.1 Brannstifting	Brannvern
2.1.2 EMP	Faraday bur.
2.1.3 Linjekutt	Backup linjer. UPS.
2.1.4 Innbrudd	Innbruddssikring, dvs alarmer, vaktselskap, etc.
2.2 Elektronisk	Generelt (gjelder alle undernoder): IDS
2.2.1 Generere mange profiler	Gjøre det vanskelig å lage profil via script. Bruke registreringskontroll for eksempel at brukeren må taste inn verdier som er tegnet inn i et bilde ved oppretting av profil.
2.2.2 Ormer	Brannmur og antivirus programvare. Filtrerings programvare.
2.2.3 Defacing	Sikker webserver.
2.2.4 Spoofing	
2.2.4.1 DNS poisoning	Systemet må autentisere for brukeren. Sertifikater.
2.2.5 Virus	Antivirus programvare. Automatisk oppdatere programvare.
2.2.6 Hammering	Brannmurløsning med pakkefiltrering.

3 Misbruke databasen	Generelt (gjelder alle undernoder): sikker og testet databaseløsning.
3.1 Skrive til databasen	Generelt (gjelder alle undernoder): kontrollere input, bruke VIEWs der det er mulig.
3.1.1 SQL injection	
3.1.1.1 Blind write	Bruke WITH CHECK OPTION i databasekall.
3.2 Lese fra databasen	Generelt (gjelder alle undernoder): kontrollere input, bruke VIEWs der det er mulig.
3.2.1 SQL injection	
3.2.1.1 Blind write	Bruke WITH CHECK OPTION i databasekall.
3.3 Skaffe tilgang	Generelt (gjelder alle undernoder): Sikkerhetspolicy, se også tre1. Konfigurere databasen korrekt.
3.1.1 Finne adresse	Ikke vis feilmeldinger om databaseserver til brukerne.
3.1.2 Få tak i brukernavn/passord	Kryptere/hashe brukernavn og passord. Ikke legge

	brukernavn og passord i kildekode.
3.1.3 Utnytte kjente sårbarheter	Bruke oppdatert databasesoftware.
3.1.4 Utnytte feil i konfigurasjon	Sjekke konfigurasjon og teste denne.

6.6.2 Sikkerhetspolicy

En sikkerhetspolicy er sammen med god opplæring av ansatte det beste mottiltaket mot social engineering. Innholdet i sikkerhetspolicyen detaljerer retningslinjer for hvordan en ansatt skal oppføre seg i forskjellige situasjoner.

Under følger et utkast til hva sikkerhetspolicyen til systemet må inneholde. Den endelige sikkerhetspolicyen skal ferdigstilles når systemet er ferdig utviklet og før systemet settes i bruk. Enkelte elementer fra denne sikkerhetspolicyen er basert på Kevin Mitnicks erfaringer som hacker og social engineer [Mitnick02].

6.6.2.1 Hva skal beskyttes

I hovedsak skal hele systemet beskyttes. Det mest sikkerhetskritiske vil være databasen hvor all informasjon om brukere og spill ligger lagret. Andre kritiske deler av systemet er kommunikasjonskanalene, grensesnitt mot bank og resultatservice, innlogging til systemet og servere.

Dokumentasjon og informasjon om systemet bør holdes konfidensielt så lenge det ikke er strengt tatt nødvendig for offentliggjøre denne informasjonen.

6.6.2.2 Hva skal det beskyttes mot og hvorfor

Det må beskyttes mot uønsket aksess både innenfra og utenifra organisasjonen. Man må hindre at uvedkommende får mulighet til å endre og/eller stjele informasjon lagret i systemet. Uvedkommende i denne sammenheng kan være eksterne angripere eller utro tjenere internt i bedriften. Dette innebærer også utro tjenere i de eksterne systemene vi kommuniserer mot; bank og resultatservice.

Dersom systemet er ustabilt og dette blir kjent for allmennheten, vil det medføre tap av omdømme. Tap av omdømme er noe systemet må være beskyttet imot. Lengre gjentagende perioder hvor system er nede må hindres. Ved gjentatte nedetidperioder vil bedriften miste mange kunder til konkurrerende bedrifter, dette vil også gi økonomiske konsekvenser og tap av omdømme. Nedetid kan komme av systemfeil, strømbrudd, DoS-angrep og virus.

Offentlig dokumentasjon og informasjon om systemet skal ikke kunne gjøres tilgjengelig for angriper eller konkurrenter. Dette for å begrense innsyn i hvordan systemet er oppbygd og hvor/hvordan det kan angripes for uvedkommende. Hvor mer en angriper vet om systemet jo lettere blir det å angripe. Noe dokumentasjon kan likevel være hensiktsmessig å offentliggjøre. Dette kan være for eksempel være systemoppgraderinger som endrer noe av funksjonaliteten til systemet.

6.6.2.3 Hvordan skal det beskyttes

Arkitekturen til systemet er med på å gjøre systemet sikrere. Man bør holde kompleksiteten til systemet så lav som mulig slik at man har oversikt over alle moduler og komponenter. Brannmurløsninger er viktig for å beskytte imot uønsket aksess til systemet.

Antivirusprogramvare skal beskytte mot virus som kan trenge inn i systemet. Dette vil kun gjelde kjente virus og hvis antivirusprogramvaren stadig blir oppdatert.

Bruk av sikre krypterte kommunikasjonskanaler mot aktørene i systemet er viktig for å hindre at uvedkommende kan lese ut sensitiv informasjon. Dette gjelder spesielt mot bank, resultatservice og spiller.

Strømbrudd bør være beskyttet med bruk av nødstrømløsninger (UPS). Nødstrømløsningen må kontinuerlig testes og vedlikeholdes.

Overvåkning av systemet er viktig for å oppdage feil og anomalier så tidlig som mulig. Her er det viktig å overvåke så mange aspekter av systemet og dets brukere som mulig, dog uten å bryte personvernlover.

Dokumentasjon og informasjon om systemet skal ikke offentliggjøres på Internett eller gjennom andre media uten administrators samtykke. Informasjon som skal være utilgjengelig for offentligheten skal kun distribueres gjennom administrator til vedkommende som ønsker materialet. Eventuelt materiale som skal kastes må makuleres. Dette gjelder ikke bare papir, men også disketter og CD-ROM-er.

Opplæring av aktørene som bruker systemet er nødvendig. Sikkerhetspolicyen skal være velkjent for administrator, oddsanalytiker, resultatservice og bank.

6.6.2.4 Ansvarsroller

Tabell 9 - Ansvarsroller

Administrator har følgende ansvarsroller
Overordnet ansvar for sikkerheten i systemet. Han har rollen som IT-sikkerhetsansvarlig for gamblingsystemet ligger hos systemadministratoren
Ansvar for sikkerheten til maskinvare og programvare på servere og arbeidsstasjoner innenfor systemet. Dette inkluderer infrastrukturen på "nett-nivå". Systemadministrator kan sees på som eier av systemet
Overordnet ansvar for å sette opp standarder for utstyr som brukes på "nett-nivå"
Kun administrator har administratortilgang til systemets servere og må holde passord hemmelig
Ansvar for sikker avlogging når han/hun ikke bruker systemet
Overvåking av systemet etter feil og sikkerhetsbrudd. Administrator skal så tidlig som mulig oppdage uregelmessigheter og oppfange andre tegn på sikkerhetsbrudd og potensielle farer for sikkerheten. Administrator vil derfor forsøke å avverge brudd på sikkerheten eventuelt i samarbeid med de ansvarlige for de enkelte deler av systemet
Ansvar for å videreformidle informasjon om sikkerhet til alle aktører via websider og e-postlister. Dette vil være både generell informasjon, varsling av farer og informasjon om mottiltak
Ansvar for backup av systemet. Backuprutiner skal følges. Dette innebærer blant annet kryptering av all backupdata
Administrator kan ikke misbruke rollen som systemadministrator for oppgaver som ikke er i organisasjonens interesse
Administrator har taushetsplikt ovenfor personopplysninger han/hun har tilgang til
Administrator er den eneste som har tilgang til alle brukernes brukernavn og passord. Brukernavn og passord skal kun sendes til kunde pr. e-post og dette gjelder kun e-postadressen som er lagret i profilen til den enkelte spiller. Administrator har ansvar for all teknisk support til spillere og de andre aktørene i systemet.
Deaktivering av spiller profiler er det kun administrator som kan gjøre. Dette kan kun gjøres etter bekreftelse fra spiller vha e-post eller skriftelig brev. Profilen skal etter deaktivering

beholdes i databasen i minimum 6 mnd. Dette for å eventuelt gjenopprette denne profilen hvis spiller kommer tilbake.

Oddsanalytiker har følgende ansvarsroller
Han må holde brukernavn og passord hemmelig
Han må ikke misbruke tilgangen til systemet
Han har ikke mulighet til å være spiller i systemet

Spiller har følgende ansvarsroller
Han må holde brukernavn og passord hemmelig
Han har ansvar for sikker avlogging når han/hun ikke bruker systemet
Det er spillers ansvar å lagre e-postadresse i sin profil som er reelle og i bruk. Det er bl.a. denne e-postadressen som benyttes for å motta nytt passord
Ved ønske om deaktivering av spillerprofil er det spillers ansvar å betale ut penger i profil til bank

7 Konklusjon

Gjennom arbeidet med prosjektet har vi lært mye om identifisering og analysering av sikkerhet rundt IT og ovenfor dette spesielle caset. Vi har fått trening i å benytte oss av RM ODP i utvikling av systemer, og vi har lært en god del om det å utvikle systemer med tanke på sikkerhet. Vi har sikkert ikke klart å identifisere absolutt alle svakheter og tilfeller hvor systemet kunne falle for et angrep. Dette vil nok i hovedsak være umulig, men vi føler at vi har funnet de trusselelementer som er sentrale. Vi føler også at vi har analysert systemet såpass grundig og tatt hensyn på sikkerheten i alle deler av prosjektet på en slik måte at de aller fleste angrep mot systemet ikke ville lykkes dersom implementasjonen av systemet blir basert rundt denne dokumentasjonen.

Videre arbeid i dette prosjektet vil være å implementere systemet. Under implementeringen er det viktig å bruke dokumentasjonen som her ligger til grunne. Det er lett og være for rask når det kommer til implementering, og deler av dokumentasjonen kan fort bli glemt i prosessen. Rutiner på hvordan implementering skal foregå på bakgrunn av dokumentasjonen er viktig. Det er også viktig å hele tiden videre fremover kjøre kontinuerlige tester på systemet. Dette er et helt annet aspekt i prosjektet for å øke sikkerheten og meget viktig for å komme frem til et godt sluttprodukt.

Som en siste kommentar til gjennomføringen av prosjektet føler vi at vi har gjort et godt stykke arbeid, og at gjennomføringen har vært god. Alle sitter igjen med mye lærdom, og føler at dette var en riktig måte å lære mye av fagets emner på. Ved å utføre dette prosjektet har vi mer erfaring å ta med ut i arbeidslivet og delta på andre reelle prosjekter, da alle vi som har gjort denne oppgaven kommer rett fra ingeniørlinjen, og ikke har lignende erfaring fra arbeidslivet.

8 Referanser

Booch99 *The Unified Modeling Language User Guide*, Grady Booch, James Rumbaugh & Ivar Jacobson, Addison Wesley 1999

Cockburn01 *Writing Effective Use Cases*, Allistair Cockburn, Addison Wesley 2001

ISO10746 ISO/IEC 10746, *Information technology – Open Distributed Processing – Reference Model*, http://www.iso.org

Jones02 Identification of a Method for the Calculation of Threat in an Information Environment, Andrew Jones, 2002

Kent98 *Security Architecture for the Internet Protocol*, RFC-2401, S.Kent & R. Atkinson, http://www.ietf.org/rfc/rfc2401.txt

Larman01 *Applying UML and Patterns, An Introduction to Object-Oriented Analysis and Design*, Craig Larman, Prentice Hall PTR 2001

Mitnick02 *The Art of Deception: Controlling the Human Element of Security*, Kevin D. Mitnick & William L. Simon, John Wiley & Sons 2002

NS5814 *Krav til risikoanalyser/Requirements for risk analyses*, NS 5814, http://www.standard.no

9 Figurliste

10 Tabelliste