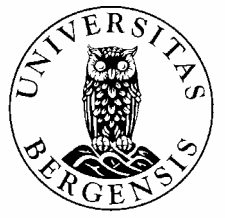
SIKKERHET
SOM TROLL AV ESKE

-- ABSTRAKTE MISBRUKSMØNSTRE
EN NY TILNÆRMING TIL SIKKERHETSKRAV

Hovedfagsoppgave i informasjonsvitenskap



Institutt for informasjonsvitenskap,
Universitetet i Bergen

AV GØRAN F. BREIVIK

BERGEN, NOVEMBER 2002

Figurliste:

Tabell-liste:

Ordliste:

Abstrakte misbruksmønstre – En ny type mønstre som er ment å støtte kommunikasjon, forståelse og enighet blant kravarbeidere. Mønstrene betegnes som abstrakte fordi de opererer på et konseptuelt nivå. De skiller seg fra andre mønstre ved at de er mindre implementeringsrettet, og mer kommunikasjonsrettet. Formålet med AMM er å formidle forståelse fra en kravarbeider til en annen og på den måten legge et bedre grunnlag for enighet.

AMM – Akronym for ”Abstrakte misbruksmønstre”

Applikasjonssikkerhet – En applikasjon eller annen type programs evne til å motstå sikkerhetsangrep

ASAC – Akronym for ”Application Security Attack Components”

BugTraq – En e-postliste for publisering av sikkerhetshull.

CERT/CC – “The CERT Coordination Center”. Et senter for sikkerhet relatert til Internett. Er tilknyttet the Software Engineering Institute ved Carnegie Mellon University.

ERP – Akronym for “Enterprise Resource Planning”. SUN og SAP er eksempler på slike omfattende administrasjonssystemer.

MC – Akronym for “Misuse Case(s)”

OWASP – Akronym for “Open Web Application Security Project”

PP – Akronym for “Protection Profile”, fra ISO 15408

ST – Akronym for “Security Target”, fra ISO 15408

UC – Akronym for “Use Case(s)”

Forord

Først og fremst vil jeg dedikere denne oppgaven til min sønn Vemund, min kone Anne Kristine og resten av familien, som alle har måttet ofre mye tid og krefter for at jeg skulle bli ferdig med denne oppgaven. Tusen takk, dere har vært beundringsverdig tålmodige.

Hovedfag ved Institutt for informasjonsvitenskap har vært et krevende og spennende prosjekt. Jeg har lært mye både om meg selv, om vitenskapelig arbeid generelt, om viktigheten av gode kolleger spesielt og ikke minst om det temaet jeg har skrevet om. Jeg vil gjerne få takke min veileder Andreas L. Opdahl for god støtte og konstruktiv kritikk. I tillegg har jeg fått uvurderlig støtte og hjelp av to gode venner og kolleger: Jan Otto og Øivind, tusen takk for en enorm innsats, jeg er svært takknemlig.

IFI har vært et trivelig, spennende og lærerikt sted på mange områder. Det har vært mange interessante diskusjoner, litt "for" sosialt miljø og i det hele tatt et fint sted å studere. Til "samboerne" mine Bernt, Olve og Stig, takk for en fin tid. Ellers vil jeg takke Maichen, Einar og alle dere andre som utgjør IFI, for et trivelig miljø.

Gøran F. Breivik

1 INTRODUKSJON

På tross av et enormt fokus på IT-sikkerhet de siste par årene er statistikkene stadig dystre på dette området. I følge NSOs mørketallsundersøkelse 2001 ble det gjort over 500 000 forsøk på datainnbrudd i Norge i fjor. 7500 av disse forsøkene var gjennomførte datainnbrudd (Kvammen, Hoel, Bakken, Torgersen, Østreng og Solstad '02). Naturligvis er folk opptatt av sikkerhet. Mange har en oppfatning av begrepet sikkerhet, men få er klar over hvor komplekst og sammensatt det området begrepet betegner egentlig er.

Tittelen på oppgaven spiller på en uttalelse fra Eugene Spafford. Han mener vi må ha sikkerhet i et hvert programvareprodukt allerede når vi tar det ut av esken (Spafford '01a). Ved å faktisk fokusere på noen av de forslagene sikkerhetseksperter foreslår for å øke sikkerheten, vil kanskje Spaffords ideal oppnås.

I denne oppgaven er begrepet sikkerhet brukt om IT-sikkerhet. Selv dette mer spesifikke begrepet er komplekst og sammensatt. Man kan sikre informasjonsteknologi mot elektromagnetiske pulser, sørge for at intranettet er beskyttet av brannmurer, at alle maskiner i selskapet har installert alle de siste oppdateringene, at alle ansatte benytter såkalt sikre passord og at all trafikken mellom ens egen bedrift og alle samarbeidspartnere er kryptert. Likevel blir selskapet utsatt for datainnbrudd. Hvordan kan det ha seg? En av grunnene er at mye av den programvaren en bedrift bruker har sikkerhetshull. Et sikkerhetshull oppstår når det er en logisk feil som kan utnyttes til å få programvare til å gjøre noe den ikke var tiltenkt. Dette kan man betegne som manglende applikasjonssikkerhet eller manglende programvaresikkerhet. Det er denne typen IT-sikkerhet generelt, og prosessen som ligger til grunn for den spesielt, som utgjør rammen for denne oppgaven.

1.1 BAKGRUNN OG MOTIVASJON

Så og si hver dag kan man lese minst én ny artikkel i massemedia relatert til en eller annen form for sikkerhet. Stort sett dreier det seg om et nytt virus som er på vei, eller det fortelles om en ung ”cracker” som har brutt seg inn eller blitt anklaget for å ha forbrutt seg mot annen manns eiendom ved hjelp av datamaskiner.

I både tidskrifter, dagspresse, ukepresse og ikke minst faglitteratur får man inntrykk av at brannmurer, nettverkssikkerhet og personlige brannmurer er alfa og omega hvis man ønsker å oppnå et akseptabelt sikkerhetsnivå. Populæroppfatningen av sikkerhet er i beste fall litt lite nyansert. Mange lag av sikkerhet befinner seg bak samlebetegnelser som sikkerhet, IT-sikkerhet og datasikkerhet. Hvis man går denne materien nærmere etter i sømmene, viser det seg at vanlige sikkerhetsproblemer vi sliter med i dag, i mange tilfeller er de samme problemene som har eksistert i mer enn 10 år (Spafford '01a).

Man er nødt til å tenke sikkerhet allerede når man planlegger et utviklingsprosjekt. Enten målet er å utvikle en enkel webapplikasjon eller et omfattende Enterprise Resource Planning (ERP) system, er man nødt til å tenke sikkerhet helt fra begynnelsen av, hvis man skal ha noe håp om at det ferdige produktet skal bli tilnærmet sikkert.

Sikkerhetsekspertene Schneier og Spafford er begge opptatt av hvordan man kan øke sikkerheten og peker på noen problemer som kan være årsaker til dårlig sikkerhet:

- Liten eller manglende forståelse for sikkerhet generelt både blant oppdragsgivere og utviklere
- Manglende kunnskap om sikkerhet blant utviklere.
- Dårlig metodisk tilnærming til sikkerhet.

Mange andre synes å være enige om at applikasjonssikkerhet er veien til ”security out of the box”, eller med andre ord at sikkerheten ikke skal være et tilleggsprodukt (Garfinkel og Spafford '97; OWASP '01; '02; Peteanu '01; Schneier '00; Spafford '01b; '01a; Viega og McGraw '02; Wheeler '02).

Sikkerhetskrav er som oftest definert som ikke-funksjonelle krav (bl.a.: Kotonya og Sommerville '98). Vi skal se i denne oppgaven at dette kanskje er i ferd med å endre seg. Som en mulig følge av fokuseringen på det ikke-funksjonelle aspektet ved sikkerhet er det en tendens til at sikkerhet kommer i siste rekke under systemutvikling (Yoder og Barcalow '97). Et program vil ha liten grad av sikkerhet hvis ikke sikkerhet er påtenkt fra begynnelsen av, dermed kan vi snakke om dårlig applikasjonssikkerhet (Bellovin '01; '02; Spafford '01b). Dette er et stort problem. Faktisk viser statistikk at

nærmere 85 % av rapporterte sikkerhetshull hos CERT/CC skyldes en eller annen form for logisk feil i programvaren, også kalt bugs (Bellovin '02; CERT/CC '02).

Med kunnskaper om Sindre og Opdahls Misuse Case-notasjon, kjennskap til patterns[1] (mønstre), tiltro til kravarbeid og en iver etter å lære mer om sikkerhet, ble utgangspunktet for denne oppgaven lagt.

1.2 PROBLEMSTILLING

Den overordnete problemstillingen for oppgaven er som følger:

Gjennom et utforskende studium å utvikle en teknikk som kan tilby en forenklet tilnærming til sikkerhet i kravarbeid.

For å utforske denne problemstillingen blir det først foretatt et litteraturstudium. På grunnlag av dette studiet blir problemstillingen presisert, før det blir utviklet et forslag til løsning kalt Abstrakte misbruksmønstre (AMM). Dette er en teknikk som har som formål å støtte kommunikasjon og forståelse for sikkerhetsangrep slik at medlemmer av et kravarbeidsteam lettere kan bli enige om sikkerhetskrav.

En antagelse som legges til grunn for oppgaven er at en forenklet tilnærming til komplekse problemer kan øke forståelse for sikkerhet, og at det dermed også kan føre til kommunikasjon rundt sikkerhet. For å oppnå en forenklet tilnærming til komplekse problemer blir detaljerte tekniske detaljer abstrahert til et mer generelt og forståelig nivå. Det forutsettes at dette i sin tur kan danne et bedre grunnlag for enighet om sikkerhetskrav blant kravarbeidere.

1.3 AVGRENSNING

Den form for sikkerhet som omtales i denne oppgaven vil i all hovedsak omfatte applikasjonssikkerhet. Dette området omhandler den interne sikkerheten i alle former for programmer. Applikasjonssikkerhet er et felt som har høstet lite anerkjennelse og popularitet inntil ganske nylig. Selv om flere har tatt opp emnet tidligere, er Viega og

[1] Med ”patterns” siktes det her til eksempelvis ”design patterns” og ”software patterns”.

McGraws bok ”Building Secure Software” den første boken med denne typen sikkerhet som hovedtema (Viega og McGraw '02). Hensynet til sikkerhetskrav under utvikling, enten det dreier seg om en enkel webapplikasjon eller et omfattende informasjonssystem, er avgjørende for sikkerheten i det ferdige produktet:

> **”You may have the world’s best firewall, but if you let people access an application through the firewall and the code is remotely exploitable, then the firewall will not do you any good …” (Viega og McGraw '02:2)**

For å illustrere konsepter utviklet i denne oppgaven tar jeg utgangspunkt i webapplikasjoner. Likevel er det verdt å merke seg at dette ikke er noen begrensning ved AMM. Teknikken kan også benyttes ved utvikling av andre typer applikasjoner og systemer. Webapplikasjoner ble valgt som utgangspunkt både for å begrense oppgavens omfang og fordi sikkerhetsproblemer knyttet til slike applikasjoner er representative også for andre typer applikasjoner. I tillegg er webapplikasjoner en applikasjonstype som er stadig mer utbredt og som også er ofte utsatt for sikkerhetsangrep.

1.4 OPPGAVENS UTFORMING

Undersøkelsesopplegget er av eksplorerende art og delt inn i to faser. Først utvikler jeg AMM på grunnlag av et utforskende litteraturstudium. Senere evaluerer jeg AMM i et kvalitativt og utforskende pilotstudium. Denne oppgavens hovedfokus ligger på arbeidet i den første fasen med å utvikle en teknikk som skal støtte innsamling, spesifisering og analyse av sikkerhetskrav i kravarbeid. Formålet med evalueringen i den andre fasen er å gi innspill og kommentarer til bruk og videre arbeid med teknikken. Arbeidet med oppgaven i sin helhet har foregått i en iterativ og inkrementell prosess. Spesielt under utviklingen av AMM foregikk det en konstant veksling mellom modelleringsforsøk og litteraturstudium.

2 METODE

I dette kapittelet presenteres det metodiske rammeverket for hver av de delene oppgaven er delt inn i. Mot slutten av kapittelet blir også oppgaven plassert inn i en større forskningsmetodisk sammenheng.

Hoveddelene i oppgaven er som følger:

Litteraturstudiet danner grunnlaget for konseptuelle fasen. Jeg har valgt å legge dette til et eget kapittel for å gjøre oppgaven mer oversiktlig.

Den konseptuelle fasen har fått denne betegnelsen fordi den dreier seg om å utvikle konseptet AMM.

Den evaluerende fasen beskriver et empirisk pilotstudium basert på dybdeintervjuer med kravarbeidere. Hensikten med studiet er først og fremst å gi grunnlag for nye ideer for videre utvikling og eventuelt videre evaluering av AMM.

2.1 METODE I DEN KONSEPTUELLE FASEN

Denne fasen etablerer grunnlaget for konseptet AMM. Da litteraturstudiet utgjør det teoretiske grunnlaget for konseptet som skal utvikles er det naturlig å starte med dette. Grunnlaget består av en syntese mellom forskningsfeltene sikkerhet, trusselmodellering, mønstre og kravarbeid.

På grunnlag av litteraturstudiet blir det utviklet et konsept som definerer teknikken AMM. Denne teknikken er utviklet for å støtte innsamling, spesifisering og analyse av sikkerhetskrav og kombinerer konsepter hentet fra mønstre og trusselmodellering. AMM danner også utgangspunktet for en mer omfattende metode for behandling av sikkerhetskrav i systemutvikling. Både teknikken AMM, og metoden som bygger videre på den, blir beskrevet utførlig i kapittel 4.

2.1.1 OPERASJONALISERING

Oppgavens eksplorerende utgangspunkt medførte at problemstillingen først ble operasjonalisert etter flere runder med grovkornet datainnsamling og analyse. Den første

perioden av forskningsprosjektet var sterkt preget av å være et nybrottsarbeid med en åpen tilnærming. En slik tilnærming har vært avgjørende for å danne en grunnleggende sikkerhetsforståelse, og for å se sammenhengene mellom sikkerhet og de andre forskningsfeltene trusselmodellering, mønstre og kravarbeid.

På grunn av den vide problemstillingen endte jeg opp med å lese mye litteratur som var perifer, men utfyllende og verdifull for min egen faglige forståelse. Spesielt gjelder dette sikkerhetsfeltet, hvor det meste av litteraturen er teknisk og pragmatisk orientert. Ved å lese denne typen litteratur i lys av problemstillingen, forstod jeg også noe av problemet med å formidle denne kunnskapen videre til personer uten IT-kompetanse. Sikkerhet er så teknisk detaljert at det umiddelbart kan være vanskelig å abstrahere kunnskap om det, i hvert fall på en måte som gjør det enkelt å forklare problemet for personer uten IT-teknisk bakgrunn. Nettopp dette problemet utgjør mye av kjernen til løsningen på problemstillingen.

Gjennom gjentatte runder med utvikling får jeg økt innsikt om styrker og svakheter ved ulike tilnærminger til de forskjellige forskningsfeltene. Dermed blir det etter hvert også mulig å se konturene av konseptet, AMM.

2.1.1.1 Innsamling av litteratur

I datainnsamlingen for litteraturstudiet ble det brukt ulike fremgangsmåter. Litteratur er samlet inn gjennom søk i forskjellige databaser tilgjengelig på web, ved leting på biblioteket, gjennom litteraturhenvisninger og ved å lete på viktige forskere og organisasjoners websteder (eks. Spafford, Wheeler, Anton, CERT, OWASP). Av sentrale webtilknyttede litteraturdatabaser vil jeg spesielt nevne:

Scientific Literature Digital Library (citeseer.nj.nec.com)

Den norske bibliotekdatabasen Bibsys (www.bibsys.no)

ScienceDirect® (www.sciencedirect.com)

The ACM Portal to computing literature (http://portal.acm.org)

Avanserte søkemuligheter i søkemotoren Google (www.google.com)

2.1.1.2 Presisering av forskningstemaet

Målet med oppgaven er i utgangspunktet: ”Gjennom et utforskende studium å utvikle en teknikk som kan tilby en forenklet tilnærming til sikkerhet i kravarbeid.” Da et aktuelt løsningskonsept begynte å ta form gjennom innsamling og gjennomgang av litteratur, var det naturlig å presisere problemstillingen. Det ble gjort med følgende to forskningsspørsmål:

Generelt:
Kan en teknikk som kombinerer konsepter fra trusselmodellering, mønstre og applikasjonssikkerhet være nyttig i arbeid med sikkerhetskrav?

Spesielt:
Kan en teknikk som kombinerer konsepter fra trusselmodellering, mønstre og applikasjonssikkerhet danne grunnlaget for bedre kommunikasjon, forståelse og enighet forbundet med sikkerhetskrav?

Formålet med en todelt presisering er å avgrense problemstillingen på en måte som gir spillerom for både den generelle og den spesielle tilnærmingen, samtidig som det er interessant å fokusere på en mer spesiell tilnærming. Som nevnt i introduksjonen er blant andre Spafford ('01b; '01a) opptatt av kommunikasjon og forståelse for sikkerhet. Al-Rawas og Easterbrook tar også opp kommunikasjon som et av hovedproblemene med kravarbeid ('96).

2.1.1.3 Forskningsenhet

Forskningsenheten i denne delen av oppgaven er det Glass, Ramesh og Vessey kaller for et ”abstrakt konsept” ('02), og er relativt vagt formulert som en metode[2], som for eksempel en datamodell (ibid:23).

Når det konseptuelle rammeverket tar form gjennom litteraturstudiet, blir også forskningsenheten mer veldefinert og kan mer spesifikt defineres som følger:

[2] Her anser jeg begrepet metode for å være mer generelt enn begrepet teknikk. Derfor mener jeg det også innbefatter forskningsenheten i denne oppgaven, som er en teknikk som kan regnes for å være en del av en metode.

Abstrakte misbruksmønstre er en kombinasjon av Misuse Case-diagrammer og mønstre. De har til hensikt å tilby en teknikk for innsamling, analyse og spesifisering av sikkerhetskrav ved å gi en forenklet og forståelig presentasjon av velkjente sikkerhetsangrep, samt løsninger for å unngå disse angrepene.

2.1.1.4 Utvalg

Etter at problemstillingen ble presisert, ble også litteraturen som danner utgangspunktet for utvikling av konseptet AMM, valgt på et mer presist grunnlag. Litteratur som omhandler flere av følgende forskningsfelt ble prioritert i utvalget, fremfor litteratur som kun omhandler ett av forskningsfeltene. Eksempelvis er litteratur som omhandler både kravarbeid og mønstre foretrukket fremfor litteratur som kun omhandler kravarbeid.

1. **Applikasjonssikkerhet**
2. **Trusselmodellering**
3. **Mønstre**
4. **Kravarbeid**

Dette utvalget er både en styrke og en svakhet med oppgaven. Det er en styrke fordi den teoretiske tilnærmingen som skal utgjøre en kombinasjon av konsepter fra forskjellige forskningsfelt, samtidig som det svekker en dypere teoretisk tilnærming til hver enkelt av disse konseptene. Det er med andre ord lagt vekt på å kombinere konsepter som til sammen kan danne grunnlaget for en løsning på problemstillingen, mer enn for eksempel å forske i dybden på årsaker til hvorfor akkurat mønstre kan bidra til å løse kommunikasjonsproblemet.

2.1.2 UTVIKLING AV KONSEPTET ABSTRAKTE MISBRUKSMØNSTRE

Flere ulike utforminger av AMM ble prøvd ut og diskutert med veileder og medstudenter. Først lå fokus på å innarbeide så mye som mulig av MC-notasjonen og å integrere dette med enkelte konsepter fra mønsterdomenet. Det tok lang tid å analysere de forskjellige klassifiseringene av sikkerhetsangrep og -feil, før et tilstrekkelig abstrakt, teknisk nivå var nådd. Mye av utviklingen forgikk ved prøving og feiling. Det vil si at en modell ble laget, gjennomtenkt, diskutert og forkastet. Til slutt valgte jeg å gå videre med en fortettet stil som gir seg uttrykk i det endelige forslaget. Resultatet er en teknikk som i hovedsak baserer seg på mønstre og i mindre grad på MC-notasjonen. Et annet resultat av den konseptuelle fasen er et forslag til en mer helhetlig metode for behandling av sikkerhetskrav. Denne metoden inneholder et sett med teknikker for å behandle sikkerhetskrav fra kravarbeid til design og implementering.[3]

2.2 METODE I DEN EVALUERENDE FASEN

For å få en indikasjon på konseptets nytte i praktisk kravarbeid ble det foretatt en empirisk pilotundersøkelse i form av dybdeintervjuer. Intervjuene, med påfølgende analyse og tolkning, ble foretatt etter kvalitative prinsipper. Målet med intervjuene var blant annet å danne et inntrykk av respondentenes forhold til problemstillingen, og et inntrykk av deres reaksjoner på løsningsforslaget. Det var også viktig å få ideer til videre utvikling og evaluering.

I den følgende delen av metodekapittelet beskrives prosessen fra problemstilling til tolkning i en metodisk kontekst. Disposisjonen av denne delen av kapittelet følger Helleviks beskrivelse av ”faser i forskningsprosessen” (Hellevik '00:34). Kort oppsummert er fasene:

- Valg og utforming av problemstillingen
- Valg av undersøkelsesopplegg

[3] Mer om dette i kapittel 4 som beskriver denne utviklingsprosessen i detalj, også forslaget til en mer helhetlig metode.

- Utvelging av data
- Innsamling av data
- Behandling av data
- Analyse av data
- Tolkning av resultatene
- Utarbeiding av forskningsrapport (ibid:34–40).

2.2.1 OPERASJONALISERING

I den konseptuelle fasen ble problemstillingen presisert og operasjonalisert for å komme frem til konseptet AMM. For å foreta en evaluering av AMM er det nødvendig å presisere og operasjonalisere problemstillingen igjen.

2.2.1.1 Forskningsenhet

Et springende punkt ved denne operasjonaliseringen er at forskningsenheten forandrer seg fra å være et abstrakt konsept til å være en gruppe personer, nærmere bestemt en gruppe kravarbeidere. Kravarbeidere anses for å være personer involvert i å avdekke, spesifisere og analysere kravene til et system. Det er viktig å understreke at det er gruppen som helhet som står i fokus. Det jeg ønsker å belyse er hvorvidt AMM kan fungere som mediator for formidling av forståelse for sikkerhetsangrep.

2.2.1.2 Forskningsspørsmål

På grunnlag av den endrete forskningsenheten blir dermed problemstillingen operasjonalisert gjennom følgende forskningsspørsmål:

Generelt:
Kan bruk av abstrakte misbruksmønstre være nyttig under behandling av sikkerhetskrav?

Spesielt:
Kan bruk av abstrakte misbruksmønstre føre til noen endring med hensyn til forståelse for og kommunikasjon rundt sikkerhetsangrep, og kan det i så fall gjøre det enklere for kravarbeidere å komme til enighet om sikkerhetskrav?

2.2.1.3 Variabler

I Klaus Pohls problematisering av kravarbeid i ”The Three Dimensions of Requirements Engineering” ('93), trekker han frem **enighet** som et vesentlig aspekt ved kravarbeidet. Slik jeg ser det, er dette et aspekt ved kravarbeidet som avhenger av god **kommunikasjon** og **forståelse** for de forholdene man står overfor. For som Pohl sier: ”… at the beginning of the RE process each person involved has his of her own personal view of the system … ” (Pohl '93:248). Videre sier han også at: ”… an agreement can only be gained through communication among the involved people … ” (Pohl '93:252).

Det er velkjent at kommunikasjon er et problem i kravarbeid, sier Al-Rawas og Easterbrook i artikkelen ”Communication Problems in Requirements Engineering: A Field Study” ('96). Videre sier de at:

> **”Knowledge acquisition and sharing can only be achieved through effective communication between the various stakeholders.”**

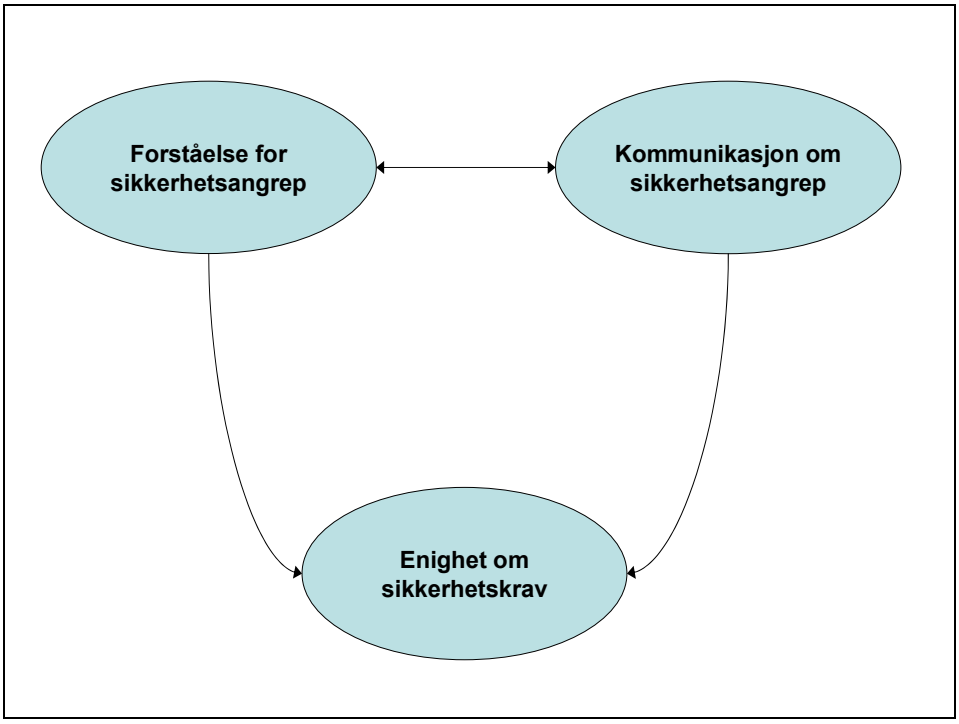
Det forutsettes i det videre at bedre forståelse for sikkerhet både vil føre til bedre kommunikasjon, og til et bedre grunnlag for enighet. Dette underbygges blant annet av følgende definisjoner av begrepene kommunikasjon og forståelse hentet fra Merriam-Webster Online (Merriam-Webster '02).

> **Communication: “A process by which information is exchanged between individuals through a common system of symbols, signs, or behaviour.”**
>
> **Understanding: ”To achieve a grasp of the nature, significance, or explanation of something.”**

Som man ser er det en sammenheng mellom de to begrepene, kommunikasjon og forståelse. For at kommunikasjon skal være vellykket må avsender oppnå forståelse hos mottaker.

Variablene kommunikasjon, forståelse og enighet er utledet på bakgrunn av Klaus Pohls artikkel. Mønstre anses å ha en kommunikasjonsbærende funksjon og dermed for å formidle forståelse (F.eks.: Coplien '96). Da konsepter fra mønstre også står sentralt i AMM, forutsettes det at et godt AMM også vil kunne gi et godt grunnlag for kommuni-

kasjon og forståelse, og dermed også for å skape enighet mellom kravarbeidere. La meg presisere: Det er her snakk om individuell forståelse for sikkerhetsangrep, kommunikasjon og enighet. Hvert individ må ha en forståelse for at gruppen skal kommunisere godt og dermed bli enige. Variablene, og sammenhengene mellom dem, danner grunnlaget for utformingen av intervjuguiden. Sammenhengene mellom variablene er illustrert i figuren under:



Figur 1 – Grafisk fremstilling av variabler.

De to variablene ”forståelse” og ”kommunikasjon” kan sies å være gjensidig avhengig av hverandre. For at kommunikasjonen om sikkerhetskrav skal være god, må også forståelsen være god. Hvis man skal oppnå god forståelse innad i en gruppe er man avhengig av alle individene i gruppen forstår. Dermed er det et avhengighetsforhold mellom det å formidle, eller kommunisere, forståelse og en felles forståelse. Hvis man forutsetter at en gruppe må ha en tilnærmet lik forståelse for å oppnå enighet, ser man også behovet for god kommunikasjon og forståelse. Hvis et kravarbeidsteam skal legge et godt grunnlag for å bli enige om hva som må gjøres med et bestemt sikkerhetshull, må alle bygge sin forståelse på et felles grunnlag. Hvis ikke kommunikasjonen innad i gruppen er god, vil det mest sannsynlig heller ikke være grunnlag for en felles forståel-

se. Hvis hvert individ forstår sikkerhetsangrepet på sin egen måte, er det sannsynlig at dette skaper et dårlig grunnlag for enighet.

2.2.1.4 Proposisjoner

På grunnlag av det spesielle forskningsspørsmålet er følgende proposisjoner definert:

1. Abstrakte misbruksmønstre kan formidle kunnskap om sikkerhetsangrep.

2. Abstrakte misbruksmønstre kan gjøre sikkerhetsangrep lettere å forstå.

3. Enklere formidling av forståelse for sikkerhetsangrep kan føre til et bedre grunnlag for enighet i et kravarbeidsteam.

2.2.1.5 Utvalg av respondenter

For å oppnå en viss bredde i utvalget ble det valgt respondenter fra følgende kategorier kravarbeidere: Sikkerhetseksperter, systemutviklere og domeneeksperter.

En **sikkerhetsekspert** er en person med erfaring fra arbeid knyttet til IT-sikkerhet generelt og eventuelt applikasjonssikkerhet spesielt. Primært vil det velges ut respondenter med kjenneskap til IT-sikkerhet fra en utviklers synsvinkel. Dette gjøres i et forsøk på å få evaluert også tekniske aspekter ved den foreslåtte representasjonsmetoden.

En **systemutvikler** er en person med en viss kjennskap til kravarbeid og analyse, men med hovedfokus på implementering. Denne klassen med respondenter vil få hovedfokus, da den anses for å utgjøre den største gruppen kravarbeidere.

Domeneeksperter er personer med inngående kunnskaper om det området et bestemt informasjonssystem utvikles innenfor. Dette trenger med andre ord ikke være en person med kunnskap om utvikling av informasjonssystemer.

Da evalueringen i denne oppgaven er et pilotstudium og regnes for å ha en sekundær og utfyllende posisjon, ble det kun gjort 6 intervjuer: 3 intervjuer med utviklere, 2 med

sikkerhetseksperter og 1 med en domeneekspert[4]. Hovedgrunnen til at utvalget ble slik sammensatt er tilfeldig. Det ble avgjort av hvilke personer som hadde anledning til å stille opp til intervju.

2.2.2 DATAINNSAMLING – INTERVJUER

På grunnlag av oppgavens eksplorerende tilnærming til problemstillingen ble dybdeintervjuer valgt som teknikk for datainnsamling i denne fasen. For som Burgess sier, så gir det forskeren en mulighet til å "… probe deeply, to uncover new clues, to open up new dimensions to a problem and to secure vivid, accurate, inclusive accounts that are based on personal experience" (Burgess '82:107, i Walker '85:4).

Som utgangspunkt for intervjuene ble det utformet en intervjuguide på grunnlag av de tre proposisjonene i 2.2.1.4. Formålet med intervjuguiden er, som Hellevik sier, å angi hvilke tema som skal dekkes i intervjuet og for å oppnå det han kaller et uformelt intervju (Hellevik '00:108). Walker sier at det å ha "an aide memoire" (intervjuguide) gjør at forskeren "… is free to follow up interesting ideas introduced by the informant" (Walker '85:5).

Intervjuene var inndelt i to deler. Den første delen var ment å kartlegge relevante opplysninger om bakgrunn og utdanning. Den andre delen var relatert til to eksempler på tilnærming til sikkerhetsangrep. Ett eksempel på AMM og ett modifisert eksempel hentet fra en database av "attack components"[5] (OWASP '01; '02). Formålet med å bruke to eksempler var å unngå at respondenten umiddelbart skulle gjenkjenne eksempelet som er utviklet av meg.

[4] Se for øvrig Tabell 9 side 76.

[5] Mer om dette i kapittel 3.

2.2.3 ANALYSE OG TOLKNING

Analysen og tolkningen foregår etter kvalitative prinsipper. Dette innebærer at både detaljer og overordnete konsepter vil analyseres og beskrives (Repstad '98; Walker '85). Repstads forklaring av hva kvalitativ metode er, gir også en god forklaring på hvordan denne metodiske tilnærmingen påvirker studiet:

> **"Kvalitative metoder handler om å karakterisere. Selve ordet kvalitativ viser til kvalitetene, det vil si egenskapene eller karaktertrekkene ved fenomener." (Repstad '98:13)**

Videre skriver han at det er teksten som står i sentrum for en kvalitativ undersøkelse. Tekst i form av forskerens nedtegnelser fra datainnsamlingen, noe som i dette tilfellet vil si transkriberte sitater fra intervjuene (ibid).

Sitatene klassifiseres på grunnlag av operasjonaliseringen, og deretter tolkes materialet eksempelvis på grunnlag av "… kombinasjoner av egenskaper som opptrer svært ofte …" (Hellevik '00:110). I Kvales oppfatning av kvalitativ analyse og tolkning vil det her være snakk om en meningskategorisering (Kvale '01:121-131). Det vil si at teksten analyseres og tolkes ved at den deles opp i hovedkategorier og underkategorier, med utgangspunkt i analysemetoden meningskategorisering (ibid:129). På dette grunnlaget diskuterer jeg interessante funn med utgangspunkt i AMM.

2.3 METODISK RAMMEVERK – MELLOM IS OG SE

Oppgaven kan generelt anses å følge en "teori før forskning"-strategi (Frankfort-Nachmias og Nachmias '96:41), og kan derfor sies å føye seg inn i en lang rekke forskningsarbeider som befinner seg et sted mellom "Information Systems research" (IS) og "Software Engineering" (SE).

IS oppfatter jeg på samme måte som Avison og Fitzgerald:

> **"The term 'information systems' has been defined as the effective design, delivery, use and impact of information technology in organizations and society" (Avison og Fitzgerald '95)**

SE forstår jeg slik det er definert i IEEEs standard 610.12:

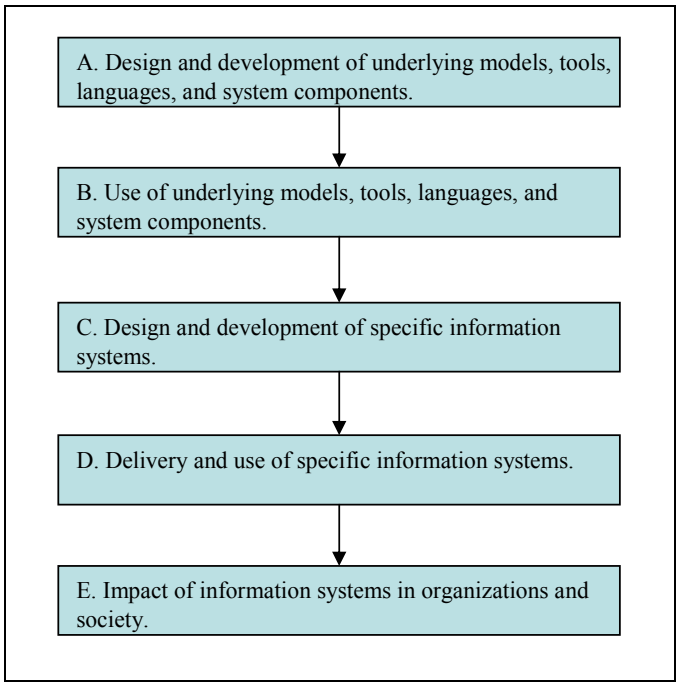
> **"(1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.**
>
> **(2) The study of approaches as in (1)." (IEEE '90)**

Mitt valg av metode er på ingen måte uvanlig. Glass, Ramesh og Vessey ('02) har på grunnlag av forskningstilnærming, analysemetode og forskningsenhet kategorisert 369 artikler fra relevante fagtidskrift innenfor IS og SE (ibid:4). I undersøkelsen fant de at rundt 50 % av artiklene nettopp var eksplorerende i sin tilnærming (ibid:20), baserte seg på konseptuell analyse som analysemetode (ibid:21) og hadde et abstrakt konsept[6] som forskningsenhet (ibid:22). Som vi har sett passer også den første fasen av denne oppgaven (den konseptuelle fasen) inn i nettopp denne beskrivelsen.

Nunamaker, Chen og Purdin ('91)introduserer en multimetodologisk tilnærming til IS-forskning som innbefatter elementer fra både IS og SE. De argumenterer for at "Sys-

[6] Begrepet abstrakt konsept " … is used to describe a concept such as a data model" (Glass et al. '02). Jeg anser begrepet for å omfatte beskrivelse av konsepter som modelleringsteknikker og -metoder, så vel som datamodeller.

tems Development" (SD) bør være ett av flere ledd i denne tilnærmingen. Ved å kombinere teoribygging, observasjon, eksperimentering og SD, vil det også være naturlig å innlemme SD som en del av IS forskningen. Utover dette mener de også at SD utgjør en nødvendig kjerne i denne tilnærmingen. Både teoribygging, eksperimentering og observasjon har en direkte tilknytning til SD, som Nunamaker et al. anser som sentralt i et nettverk av metodologier innen IS og SE. Burstein og Gregor tolker Nunamaker et als multimetodologiske tilnærming som fem sekvensielle steg:



Figur 2 – Phenomena of interest in IS (Burstein og Gregor '99)

Selv anser jeg oppgaven for å passe godt inn i det multimetodologiske rammeverket Nunamaker et al. beskriver. Mer spesifikt passer den inn som steg A og B av Burstein og Gregors fremstilling. Metodisk så vel som praktisk oppstod det derfor et skille i oppgaven. Det ble naturlig å dele den inn i to hovedfaser:

Den konseptuelle fasen svarer til steg A i figur 2, og **den evaluerende fasen** svarer til steg B i figur 2. Den evaluerende fasen evaluerer ikke direkte bruk, men indirekte. Jeg anser det som starten på evaluering av bruk, i og med at den gir innspill til gode ideer for videre utvikling og evaluering.

Denne inndelingen av oppgaven er gjort av to grunner. Først og fremst er det gjort på grunn av metodisk ulikhet som oppstår i det den ene fasen skal utvikle et abstrakt konsept i form av en teknikk eller et modelleringsverktøy. Dette kaller Nunamaker et al. ('91:94) for den teoribyggende delen. Burstein og Gregor ('99:125-126) beskriver steg B best som ”use of underlying models”, eller som i dette tilfelle: Evaluering av tenkt bruk.

2.3.1 VITENSKAPSFILOSOFISK TILNÆRMING

Epistemologisk plasserer oppgaven seg i den kvalitative tradisjonen, med postprosessuell, fortolkende og konstruktivistisk teori som bakgrunn. Dermed bygger oppgaven ontologisk på en relativistisk oppfatning, en oppfatning om at naturen er relativ til subjektet. Aksiologisk blir subjektets innvirkning på forskningen oppfattet som avgjørende. Derfor er det ikke noe mål i denne oppgaven å forsøke å komme frem til objektiv kunnskap, verken på grunnlag av det konseptuelle rammeverket, eller i den påfølgende evalueringen.

3 TEORI OG LITTERATURSTUDIUM

Formålet med dette kapittelet er å gjøre rede for litteratur som definerer det teoretiske rammeverket for konseptet Abstrakte misbruksmønstre. Først blir kravarbeid relatert til applikasjonssikkerhet, sikkerhetskrav og gjenbruk. Temaet sikkerhet blir definert og avgrenset i tråd med oppgavens kontekst. Deretter følger en gjennomgang av de sentrale temaene trusselmodellering, mønstre og gjenbruk.

3.1 SIKKERHET

Mediene skriver nærmest daglig om sikkerhetsproblemer. I den forbindelse kommer Viega og McGraw med en interessant kommentar:

> **"the media generally manages not to get to the heart of the matter... Behind every computer security problem and malicious attack lies a common enemy – bad software." (Viega og McGraw '02)**

Når et program skal utvikles er det viktig å ta hensyn til de krav som stilles til det ferdige produktet (Hoffer, George og Valacich '99; Jacobson, Booch og Rumbaugh '99; Kotonya og Sommerville '98; Spafford '01a). Dette er ikke alltid tilfelle når gjelder sikkerhet.

Mange sikkerhetseksperter (Bellovin '01; McGraw og Viega '00; Peteanu '01; Schneier '00; Spafford '01b; '01a; Wheeler '02) antar at veien til bedre sikkerhet generelt er å sørge for, som Spafford formulerer det: "security right out of the box" (Spafford '01b). Med det mener de at vi må sørge for å oppnå langt bedre sikkerhet innad i hvert enkelt program, enten det er en applikasjon eller et komplekst informasjonssystem. Bakgrunnen for dette er at de fleste sikkerhetsbrudd er et resultat av logiske feil i kildekoden (Bellovin '02; CERT/CC '02). Det vil si at man kan øke sikkerheten i programvare ved at man i størst mulig grad sørger for å unngå feil som kan føre til sikkerhetsbrudd.

Dette virker innlysende for mange systemutviklere og programmerere. Allikevel viser det seg at de samme elementære feilene går igjen gang på gang, år etter år. Bruce Schneier oppsummerer det med følgende utsagn i forordet til "Building Secure Software":

”We wouldn’t have to spend so much time, money, and effort on network security if we didn’t have such bad software security”. (Viega og McGraw '02:xix)

3.1.1 EN KORT INNFØRING I ”HACKER-SJARGONG”

En cracker, eller ”dark-side hacker” som de også kalles, blir beskrevet i ”The Jargon File” som en person som bryter sikkerheten i et system (Raymond '02). En ”script kiddie” blir på samme sted beskrevet som den typen cracker som står lavest på rangstigen. En person som gjør skade ved å benytte seg av skript og programmer skrevet av andre, uten å forstå hvordan ”exploit’en” virker (ibid). Mange hevder at dette er den vanligste typen cracker.

Det er ikke uvanlig at det lages såkalte ”exploits” [7] for å forenkle prosessen med å utnytte kjente sikkerhetshull (ibid). En exploit er et program som automatiserer angrep. Dette medfører blant annet at selv teknisk avanserte angrep kan foretas selv med meget lav grad av kunnskap. En exploit kan dermed føre til at et angrep blir mer tilgjengelig for ”script kiddies”.

”The Jargon File” nevner en rekke forskjellige typer hackere og crackere, men i prinsippet har man to typer, de såkalte ”white hats” og ”black hats”, eller henholdsvis de gode og de onde. Faktisk er det også personer som kaller seg ”grey hats”, eller i visse tilfeller samuraier. Dette er hackere som bryter sikkerheten i et system i lovlig øyemed (ibid). Her til lands er det ikke usannsynlig at enkelte personer i sikkerhetsbedrifter, Datatilsynet eller Økokrim kan regnes med i denne gruppen.

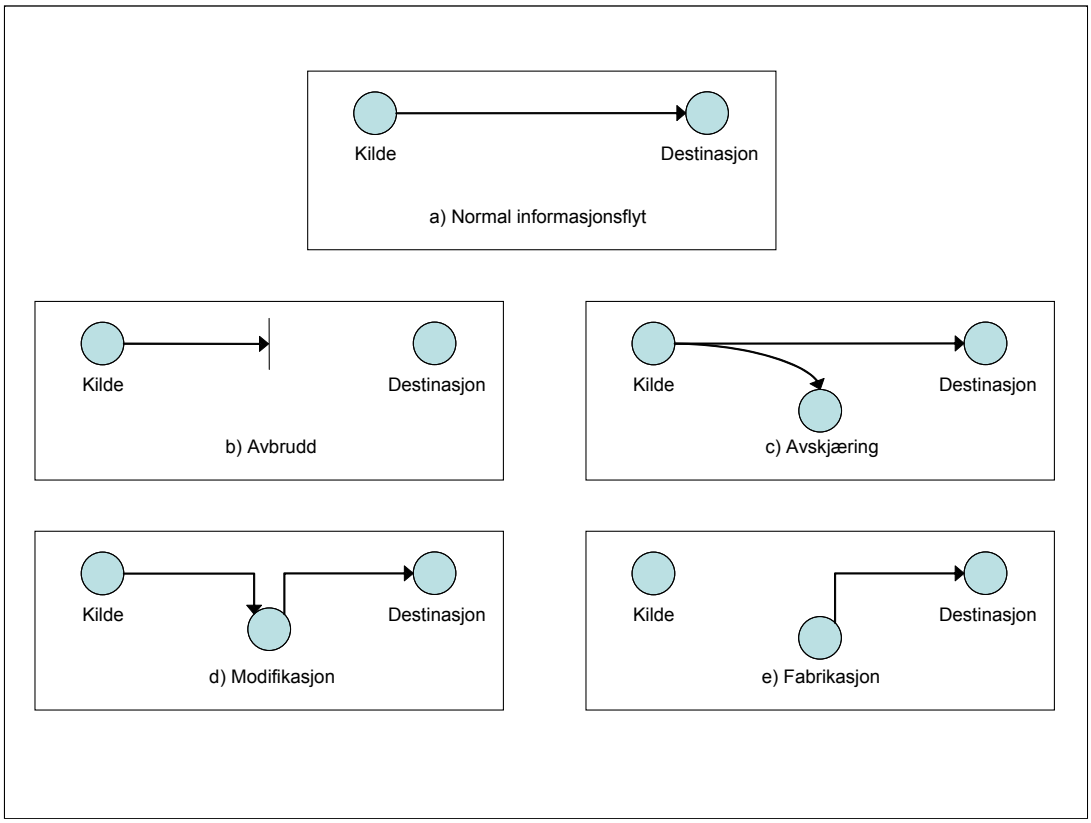
[7] En exploit er et ferdig utviklet skript som utnytter et sikkerhetshull, eller flere. Her har jeg brukt det engelske begrepet i mangel av noe godt norsk begrep som er dekkende. Se for øvrig 3.3.1.

3.1.2 NETTVERKSSIKKERHET VS. APPLIKASJONSSIKKERHET

Et sikkerhetsangrep er i følge William Stallings ('00):

> **"Any action that compromises the security of information owned by an organization."**

Han deler sikkerhetsangrep i fire generiske grupper: Avbrudd, modifikasjon, avskjæring og fabrikasjon (illustrert i figuren under).



Figur 3 – Fire generiske nettverksangrep (Stallings '00).

I praksis består gjerne ett angrep av flere av disse generiske komponentene. Ett eksempel er en angrepstype som kalles "Session Replay", som betegner et angrep hvor en uautorisert bruker gjentar en autentisert websesjon. Hvis en uautorisert bruker enten klarer å knekke koden i sesjons-id'en, eller på annen måte får fatt i en slik ID innenfor det aktuelle tidsspennet den er ment å vare, kan han belaste den autentiske brukeren med uautoriserte innkjøp, endre informasjon, eller lignende. Fordi webtrafikk foregår

over en tilstandsløs protokoll[8], er dette høyst aktuelt i dagens webbaserte IT-hverdag. Webapplikasjoner er derfor aktivt nødt til å opprettholde en innlogget tilstand. For å unngå å sende brukernavn og passord fram og tilbake ved hver bevegelse en bruker gjør, benytter man en det man kaller for en sesjons-ID. Det er eksempelvis et kodet sammendrag av passord og brukernavn i tillegg til en tilfeldig verdi. Denne ID'en brukes for å verifisere en innlogget bruker etter hvert som han beveger seg rundt i grensesnittet til den aktuelle applikasjonen uten at brukeren trenger å tenke noe særlig på det.

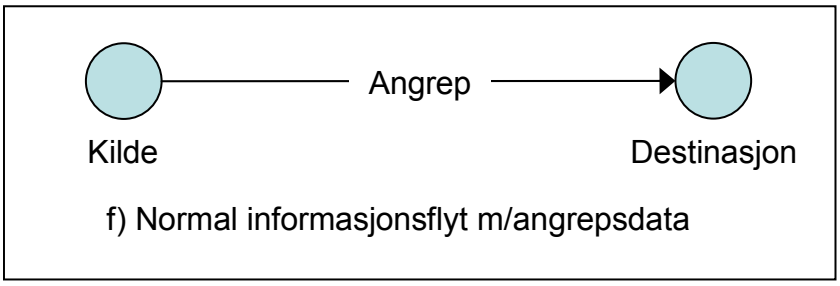
Dette skisserte angrepet er en kombinasjon av angrepene c) avskjæring og e) fabrikasjon (jfr. figuren over). Legger crackeren i tillegg til angrep b) avbrudd, vil det defineres som et "Session Hi-jacking" angrep i stedet. Først avlytter angriperen informasjonsflyten for å stjele en sesjons-id, eller et logisk mønster han kan bryte for å forutsi en fremtidig sesjons-ID. Når dette er gjort kan han f.eks. foreta en falsk innlogging med bruk av fabrikkerte data. Er det eksempelvis en e-handels løsning, kan resultatet være at han kan handle via regulære kunders konti. For å utføre dette angrepet manuelt, kreves det en god del kunnskap, og crackeren vil mest sannsynlig ikke være av typen "script kiddie". Derimot kan en script kiddie laste ned en exploit som gjør mye av jobben for ham.

Det mest interessante med figuren over er det faktum at angrepene er en kombinasjon av nettverksangrep og applikasjonsangrep. Angrepene ville ikke vært mulig å gjennomføre, eller i det minste langt vanskeligere å gjennomføre, hvis applikasjoner hadde vært utviklet med applikasjonssikkerhet på dagsorden.

Det må anses som en logisk feil i applikasjonen når det for eksempel er mulig å foreta et sesjonsrelatert angrep. En sesjons-id må i første omgang være tidsbegrenset, og i andre omgang bør informasjonsflyten i sin helhet være kryptert. Det siste er et vesentlig prinsipp mange webapplikasjoner bryter. Hvis ikke informasjonsflyten krypteres i sin helhet, uavhengig av hvor i applikasjonen brukeren beveger seg, vil sesjons-id'en og all annen informasjon sendes i klartekst i det øyeblikket brukeren beveger seg inn på et ukryptert område i applikasjonen. For å illustrere angrepet i sin helhet ved hjelp av den

[8] At HTTP er tilstandsløs vil si at applikasjonen som implementerer den ikke får noen hjelp til å holde styr på hvor den er i en kommunikasjonsprosess.

samme ”notasjonen” som Stallings bruker i figuren over (figur 3), må det legges til en figur som viser den normale informasjonsflyten som bærer av angrepsdata og eventuelt som informasjonskilde. Dette gjøres for å understreke at informasjonsflyten i seg selv kan være sikker, men at dette ikke betyr at applikasjonen i enden er nødvendigvis er sikker av den grunn.



Figur 4 - Generisk angrep mot applikasjonssikkerheten.

Figuren skal illustrere hvordan angrep mot applikasjonssikkerheten må anses som noe annet enn et nettverksangrep. Enkelt forklart kan man si at et nettverksangrep dreier seg om å avbryte, avskjære, modifisere eller fabrikkere en informasjonsflyt. Et applikasjonsangrep går ut på å utnytte feil ved destinasjonsapplikasjonen, normal informasjonsflyt eller ei.

I sammenheng med for eksempel webapplikasjoner, kan et angrep foregå utelukkende via normal informasjonsflyt. En cracker kan lese mye ut av kildekoden i en dårlig sikret webapplikasjon. På den måten kan han skaffe seg informasjon om hvordan en applikasjon er bygd opp. Derfor kan denne typen angrep være vanskelig å avsløre ved å sikre infrastrukturen, for eksempel med brannmurer, kryptering og annen nettverkssikkerhet.

3.1.3 APPLIKASJONSSIKKERHET

”Application security”, ”software security” eller ”secure/safe programming/coding techniques” er begreper som betegner en applikasjons evne til å motstå sikkerhetsangrep, og hvordan utvikle sikker programvare (ibid:xxiii). Man kan si at applikasjonssikkerhet er hver enkelt applikasjons, eller hvert enkelt programs, enestående sikkerhet.

Applikasjonssikkerheten kan antas å øke proporsjonalt med reduksjonen av bugs, eller logiske feil, i programkoden (Spafford ’01). Denne oppfatningen forutsetter vel å merke at en logisk feil i prinsippet kan være utnyttbar, og dermed et potensielt sikkerhetshull.

3.1.3.1 Sikkerhetshull og sikkerhetsangrep

I det øyeblikket en logisk feil i en applikasjon som er utnyttbar blir funnet, blir den omtalt som en ”vulnerability” (sikkerhetshull).

Applikasjonen befinner seg i en usikker, eller sårbar tilstand (Bailey & Bishop 96). I det øyeblikket en cracker utnytter eller forsøker å utnytte sikkerhetshullet, betegnes dette som et sikkerhetsangrep.

Med begrepet sikkerhetshull forstås i denne sammenheng en hver logisk feil ved et informasjonssystem, eller deler av et slikt system som potensielt kan representere en trussel mot det aktuelle systemets, eller dets tilknyttede datas, konsistens og/eller integritet.

3.1.4 HVORDAN OPPNÅR MAN APPLIKASJONSSIKKERHET?

Spafford (2001) oppsummerer hvordan vi skal oppnå bedre applikasjonssikkerhet blant annet i følgende tre punkter:

> **"Understand the needs of the users**
>
> **Understand basic tenets of security**
>
> **Capture requirements for design and validation"**

Disse punktene tar opp tre viktige sider ved applikasjonssikkerhet. Spafford anser både kommunikasjon, forståelse og krav som viktige elementer ved applikasjonssikkerhet.

3.2 TRUSSELMODELLERING

For å spesifisere sikkerhetskravene til et system er det mange som er enige om at man bør gjøre dette ved å modellere potensielle sikkerhetstrusler mot systemet under utvikling (McDermot '01; McDermot og Fox '99; Moberg '00; Moore, Ellison og Linger '01; Schneier '99; '00; Sindre og Opdahl '00; '01; Sindre, Opdahl og Breivik '02). Dette kan med en samlebetegnelse kalles for trusselmodellering. Ved å modellere hva og hvem et system trues av, kan man også lettere danne seg et bilde av hva man kan og bør gjøre for å oppnå tilstrekkelig sikkerhet (Schneier '00).

3.2.1 KLASSIFISERING OG ABSTRAKSJON AV SIKKERHETSANGREP

Som grunnlag for utformingen av AMM var det naturlig å ta utgangspunkt i en eller flere klassifiseringer av misbruk. En klassifisering av misbruk har som formål å abstrahere sikkerhetsangrep til et mindre detaljert og mer generisk nivå. Formålet med dette er blant annet å kunne forenkle konseptet bak et sikkerhetsangrep, for å gjøre det mer forståelig for folk uten tilstrekkelig teknisk bakgrunn. Som Schneier sier, så er det ikke så stor forskjell på et sikkerhetsangrep i den fysiske verden og i den digitale verden, hvis man skreller bort alle de tekniske detaljene (Schneier '00). Man burde med andre ord kunne forstå et sikkerhetsangrep uavhengig av teknisk bakgrunn. Det viktige er at det er tilstrekkelig abstrahert, og på en måte som gjør det mer forståelig for personer

med en slik bakgrunn. De tekniske detaljene må forenkles på en slik måte at de beholder sin mening uten å virke vanskelige.

For å få generelt innblikk i sikkerhetsproblemer, kan man f.eks. ta for seg webstedet til The CERT Coordination Center (CERT/CC)(www.cert.org). Der finner man blant annet en egen side om ”Vulnerabilities, Incidents and Fixes”. Et annet alternativ er SecurityFocus’ arkiver over den anerkjente e-postlisten BugTraq (www.securityfocus.com) eller ”The Vulnerability Disclosure List” (www.vulnwatch.org). Ønsker man derimot mer spesifikk informasjon om sikkerhetsangrep relatert til spesielle applikasjoner, er SecurityFocus’ e-postliste ”webappsec@securityfocus.com” et bedre utgangspunkt.

Det er gjort en del arbeid med å utarbeide klassifiseringer og taksonomier for ”vulnerabilities” (sårbarhet), de første allerede på 1970-tallet (Bishop og Bailey '96). Det viser seg derimot at det kun er én klassifisering som tar utgangspunkt i sikkerhet spesifikt forbundet med webapplikasjoner. Denne klassifiseringen er det ”The Open Web Application Security Project” (OWASP) som opprettholder (OWASP '01; '02). OWASP er, som navnet tilsier, et prosjekt rettet mot sikkerhet i tilknytning til webapplikasjoner. På prosjektets websted foreslår de en klassifisering av sikkerhetsangrep under det de kaller ”Application Security Attack Components” (ASAC) (ibid).

OWASPs arbeid med klassifisering av sikkerhetsproblemer knyttet til webapplikasjoner gjennom ASAC, har ført til definisjonen av ni sårbarhetsklasser av det de kaller ”Attack Components” (angrepskomponenter). Hver av klassene inneholder et varierende antall angrepskomponenter. For eksempel har klassen ”Informational” fire angrepskomponenter. I sitatet under forteller de litt om hvilket grunnlag de har brukt for klassifiseringen sin:

> **”When analyzing problems described on Bugtraq it is evident that most problems are variants of common issues, but applied to different applications or systems using different parameters or targets.” (OWASP '01)**

En angrepskomponent beskriver et vanlig problem innenfor sikkerhet tilknyttet webapplikasjoner. Det vil si at hver angrepskomponent beskriver en vanlig komponent i et angrep på et webbasert system, og hvilke mottiltak som er kjent. Denne klassifiseringen

kan per i dag ikke anses for å være fyllestgjørende med hensyn til sitt formål, da den stadig revideres og alltid vil være mer eller mindre utdatert.

Ved en sammenligning mellom OWASPs klassifisering og lignende klassifiseringer (f.eks.: Aslam, Krsul og Spafford '96; Bishop '95; Howard og Fischbeck '97; Scambray, Kurtz og McClure '01), viser førstnevnte seg å være mer omfattende og mer spesifikk, uten å være mindre abstrakt. På grunnlag av denne sammenligningen synes OWASPs klassifisering å være av en slik kvalitet at den godt kan brukes som grunnlag i denne oppgaven, på tross av at den er uferdig.

Det er lite sannsynlig at en slik klassifisering noen gang vil bli ferdigstilt, fordi webapplikasjoner stadig er under utvikling. Dermed vil sikkerhetsproblemene som er knyttet til slike applikasjoner også være under utvikling. Disse forholdene henger nøye sammen. Betegnende for dette i forbindelse med andre applikasjoner, er Gary McGraws tordentale mot sikkerhetsoppdateringer: ”Det er jo som å gi crackerne et kart, slik at de lettere skal finne det aktuelle sikkerhetshullet” (fritt fra hukommelsen etter et innlegg McGraw holdt på SREIS 2002).

Når det er sagt kan det også legges til at Garfinkel og Spafford i boken ”Web Security & Commerce” allerede i 1997 skrev om en del av de problemene som nå er klassifisert og beskrevet gjennom OWASPs klassifisering (Garfinkel og Spafford '97). Av den grunn kan man kanskje hevde at utviklingen ikke går så raskt som man gjerne tror. I det minste er det en del problemer som går igjen og stadig er like aktuelle.

Marcus Ranum (Ranum '97) presenterer en kort oppsummering av det som må anses som superklasser av angrep. Ranums posisjon i sikkerhetsmiljøer (www.issa.org/-marcusranum.htm) gjør denne klassifiseringen verdt å kikke nærmere på. Av de angrepsklassene han lister opp, er det hovedsakelig én klasse som er interessant for denne oppgaven. Den klassen kaller han for ”Exploits”. Den inneholder angrep han definerer som å utnytte et sikkerhetshull i et program eller operativsystem (ibid:2). I forbindelse med denne klassen er også Ranum inne på dette med at ”badly written software is the norm” (ibid:14).

Ved å beskrive en sårbarhet i et system, beskrives også et potensielt sikkerhetsangrep som kan utnytte den sårbare tilstanden. For at en beskrivelse av et sikkerhetsangrep skal

egne seg i kravarbeid, er det nødvendig å benytte en abstrahert beskrivelse. Det er mange grunner til det, men den viktigste i denne forbindelse er at det kreves et høyt kunnskapsnivå om datasikkerhet hvis man skal forstå en detaljert teknisk beskrivelse uten at den er abstrahert. Skal man gi grunnlag for økt forståelse, kommunikasjon og enighet blant kravarbeidere er man nødt til å forenkle disse problemene. Hvis det er mulig å forklare sikkerhetsproblemer på en måte som er forståelig for en domeneekspert, har man klart å skape et slikt grunnlag.

En annen grunn til å abstrahere sikkerhetsangrep er at en teknisk detaljert beskrivelse fort kan føre til at kravspesifikasjonen blir påvirket av realiserings- og teknologibaserte løsninger, noe som er et stort problem i kravarbeid i følge Jackson og Zave ('97). Hvis man er for spesifikk i kravarbeidet, blir det i følge Jackson og Zave lett for at man lar dette styre den løsningen man ender opp med. Dette snevrer inn mulighetene og dermed sjansen for å få best mulig kvalitet i det ferdige produktet. Ved å abstrahere problemene til et nivå som fører til at folk uten teknisk IT-kompetanse forstår dem, kan man også oppnå å forminske dette problemet.

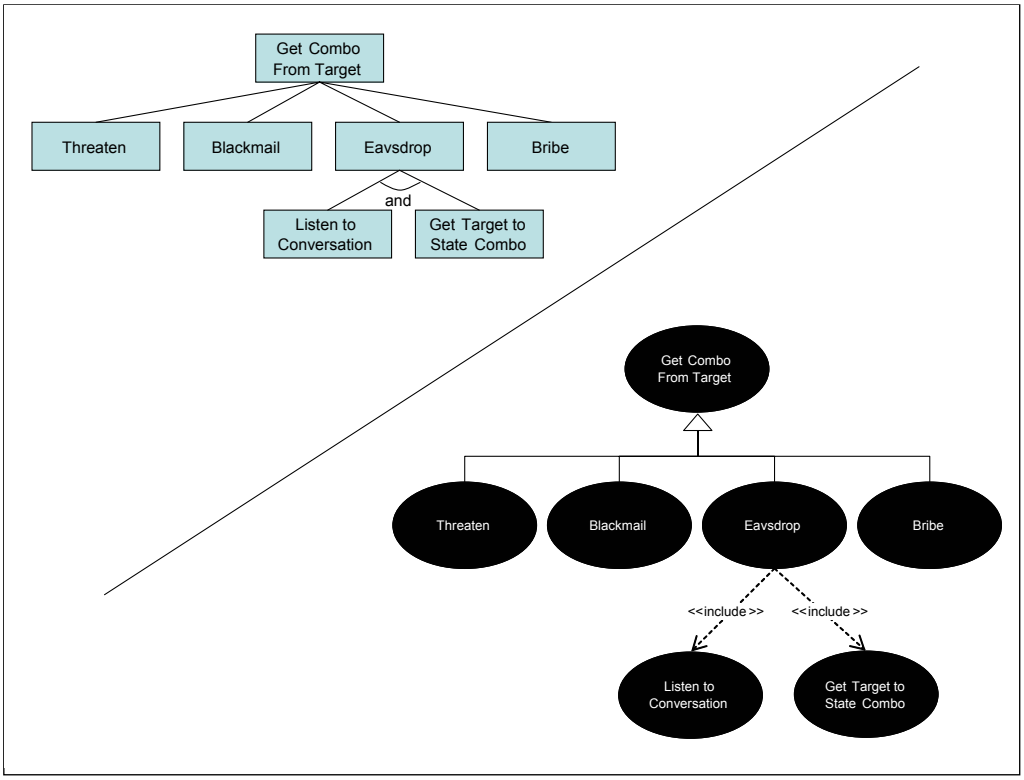
3.2.2 ANGREPSTRÆR

Det finnes ulike metoder for å modellere sikkerhetstrusler. Edward G. Amoroso lanserte en teknikk han kalte for ”Threat Trees” (Amoroso '94), som har mye til felles med Schneiers ”Attack Trees” (angrepstrær) (Moberg '00; Schneier '99; '00).

Konseptet for angrepstrær er enkelt. Man tar utgangspunkt i et potensielt mål ønsket oppnådd av en potensiell angriper. Dette målet representerer en rotnode i en trestruktur. Deretter dekomponerer man dette målet i mulige delmål. Disse delmålene representeres med hver sin løvnode. Etter behov fortsetter man denne dekomponeringen med flere lag løvnoder.

Angrepstrær er også blitt kalt for AND/OR-trær, som kommer av at man kan merke grenene mellom nodene med stereotypene ”AND” og ”OR”. Hvis en node er avhengig av mer enn én løvnode for å oppfylles, merker man grenene med stereotypen ”AND”. Er det flere måter å oppnå samme node på, merkes grenene til alternative noder med ”OR” (Moberg '00; Schneier '99; '00).

I figuren under er det vist to identiske angrepstrær i to forskjellige notasjoner. Den øverste notasjonen er den Schneier selv bruker, og den nedre notasjonen er MC-notasjonen. Meningen er å vise at det er fullt mulig å benytte MC-notasjonen til å tegne angrepstrær, hvis man ønsker det. I MC-notasjonen er spesialisering en mulig måte å uttrykke en ”OR” sammenheng på. For å uttrykke en ”AND” forbindelse kan man benytte assosiasjonstypen ”Include”. Dette vil også utforskes nærmere i kapittel 4.



Figur 5 - Eksempel på angrepstre i to notasjoner.

Bruce Schneier argumenterer med at angrepstrær blant annet kan være en god metode for å utarbeide forskjellige typer estimater. Ved f.eks. å tilegne verdier til enkeltnoder kan man summere og få beregnet pris- og/eller tidsestimater (Schneier '00:319-320).

3.2.3 ABUSE CASES, MISUSE CASES OG USE CASES

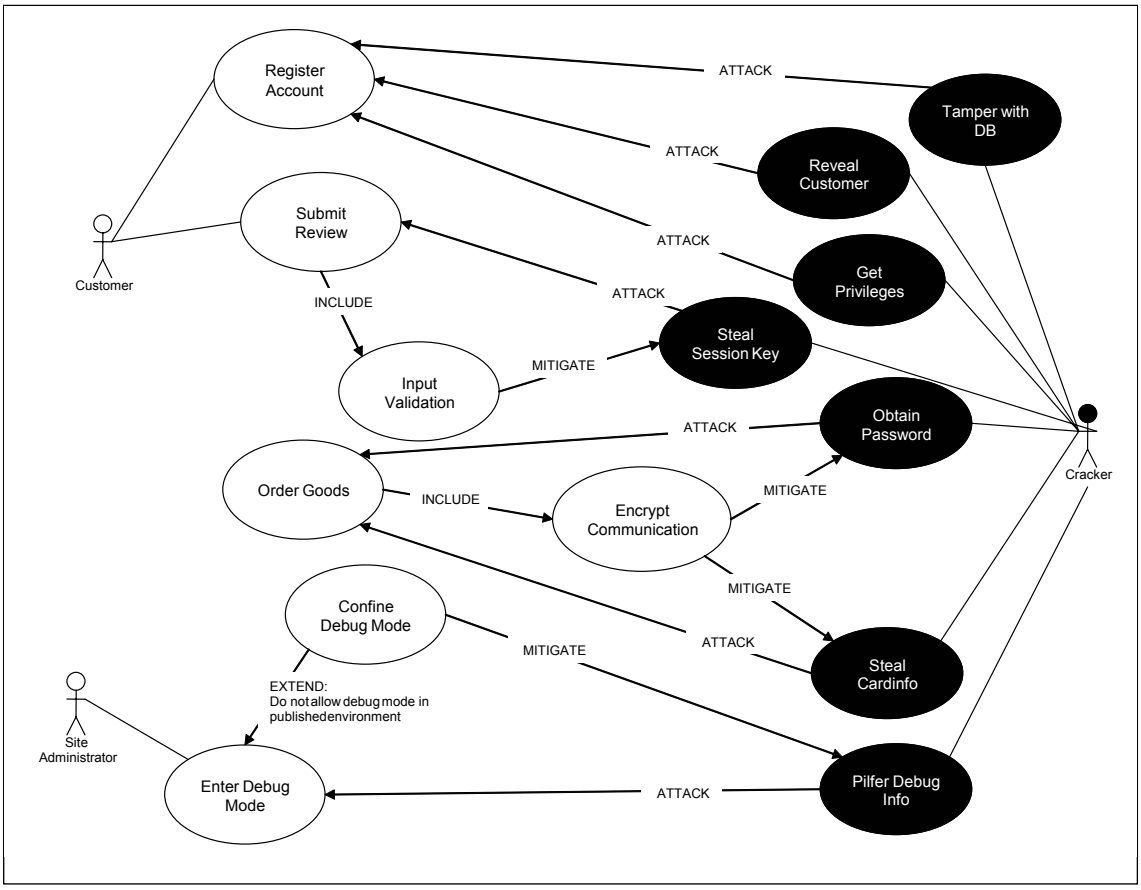
Misuse Cases (MC) (Sindre og Opdahl '00; '01; Sindre et al. '02) bygger på Use Case (UC) modellering, som er en del av ”The Unified Modeling Language” (UML). UC-modellering går kort sagt ut på å modellere **hva** systemet skal gjøre, eller med andre ord

beskrive systemets funksjonalitet (brukstilfeller). MC-modellering derimot, er ment å vise funksjonalitet som ikke skal være mulig (misbrukstilfeller), slik at man lettere skal kunne ta høyde for dette i utviklingen. MC-notasjonen er foreslått som en direkte utvidelse til UC-notasjonen, ikke som et alternativ eller separat supplement (Sindre og Opdahl '00:5).

UC-diagrammer er, som det står i OMG Unified Modeling Language Specification:

> **"… a graph of actors, a set of Use Cases, possibly some interfaces, and the relationships between these elements. The relationships are associations between the actors and the Use Cases, generalizations between the actors, and generalizations, extends, and includes among the Use Cases. The Use Cases may optionally be enclosed by a rectangle that represents the boundary of the containing system or classifier."**

MC-diagrammer er som nevnt en utvidelse av den ordinære UC-notasjonen. I tillegg til de vanlige elementene, tilfører MC-notasjonen elementene "misactors" og "Misuse Cases". Forbindelsene "attack" og "mitigate", understreker de forskjellige elementenes roller og gjør notasjonen meget intuitiv. Figuren på neste side viser et eksempel på integrasjonen mellom UC- og MC-notasjonene, og illustrerer likheter og forskjeller mellom disse. Den inneholder også språkelementer fra begge notasjoner, som blir nærmere forklart i teksten som følger.



Figur 6 – Eksempel på Misuse Case diagram (Sindre et al. '02)

Ut i fra figuren ser man at forbindelsen "mitigate" kunne vært dekket med en ordinær "include"-forbindelse hvis man hadde benyttet standard UC-notasjon. Ved å utvide med MC-notasjonen med "Mitigate" og "Attack" oppnår man å gi mer og viktig informasjon om MC'en. Mitigate-forbindelsen forteller for eksempel i figuren over at "Confine Debug Mode" har til hensikt å forhindre, så langt det lar seg gjøre, "Pilfer Debug Info". På samme måte viser forbindelsestypen "Attack" hvilket punkt som anses for å være angrepspunktet.

En UC-modell er et sett med UC-diagrammer som hver utgjør et scenario. En UC-modell kan ta form som ren tekstlig beskrivelse, eller som et grafisk diagram med beskrivelse (Fowler 00). Et slikt scenario beskriver sider ved systemets funksjonalitet tilknyttet bestemte mål for en eller flere aktører.

En aktør er en persons rolle som bruker av systemet, eller eventuelt et annet system eller del av et slikt. Et scenario kan innholde en eller flere aktører og et eller flere "Use Ca-

ses" (brukstilfeller). En aktør kan være involvert i et eller flere scenarier og et eller flere brukstilfeller (OMG '01).

Et brukstilfelle beskriver én mer eller mindre abstrakt funksjon i systemet (ibid). En slik funksjon kan eksempelvis være innlogging. Hvis scenariet for innlogging er avansert og inneholder mye funksjonalitet, kan det dekomponeres i flere nivåer, og dermed vise f.eks. "sjekk brukernavn", "sjekk gruppe", "sjekk passord" og så videre. Det er viktig å merke seg at et UC-diagram aldri skal vise *hvordan* systemet skal gjøre noe, men kun være en abstrakt beskrivelse av *hva* systemet skal gjøre.

MC-notasjonen som informasjonsbærer fører med seg forenklet kommunikasjon gjennom intuitivt forståelige modeller. Alistair Cockburn skriver i sin bok om UC'er at de "… normalt fungerer som et kommunikasjonsmiddel fra en person til en annen. Ofte mellom personer uten spesiell opplæring" (Cockburn '01).

McDermot og Fox lanserte i 1999 et MC-lignende konsept som de kaller for Abuse Cases (McDermot '01; McDermot og Fox '99). Det går i all hovedsak ut på å ta høyde for misbruk, eller "intrusion scenarios" som de selv kaller det, i en ordinær UC-modell. De har ikke endret noe på notasjonen og mener at separate UC- og Abuse Case-modeller vil være den beste tilnærmingen (ibid). Dette er en motsetning til Sindre og Opdahls MC-notasjon som legger vekt på at det nettopp er en fordel å kunne fremstille både UC'er og MC'er i kombinasjon i samme UC-modell (Sindre og Opdahl '00).

3.3 MØNSTRE

Ralph E. Johnson skrev en artikkel i 1992 om hvordan man kan bruke det han kaller for "patterns" til å dokumentere et rammeverk for gjenbruk innenfor programvare-design (Johnson '92). I denne artikkelen lanserer han begrepet "patterns" som en erstatning for det Christopher Alexander et al. kaller for "a pattern language" (Alexander, Ishikawa og Silverstein '77). Et "pattern language" er, i følge Alexander et al., et sett med mønstre som skal beskrive en spesifikk type problemer (Johnson '92).

I 1995 ble boken "Design Patterns – Elements of Reusable Object-Oriented Software" begynnelsen på en bølge av litteratur som omhandler gode og velprøvde løsninger på

vanlige designproblemer (Gamma, Helm, Johnson og Vlissides '95). Det var ikke et nytt konsept, men nytt innen ”software engineering”.

I følge Gamma et al., er Christopher Alexander opphavet til tankene rundt mønstre (Gamma et al. '95:2). Martin Fowler mener derimot at patterns som konsept ikke kan tilskrives noen, men at det må anses som allmenn konseptuell tenkning (Fowler '97).

Denne oppgaven har, som Fowler, heller ingen sterk tilknytning til Alexanders ”Patterns language” i sitt forslag til ny bruk av mønstre. Allikevel har Cristopher Alexander inspirert den generelle forståelsen for mønstre som ligger til grunn for denne teksten.

Alexanders definisjon av mønstre fungerer for eksempel som grunnlag for definisjonen av AMM (Alexander et al. '77):

Et misbruksmønster beskriver et angrep som forekommer ofte i en bestemt kontekst, samt kjernen i en løsning på dette angrepet, en løsning som er så generell at den kan gjenbrukes på en effektiv måte.

Et viktig element med et hvert mønster er at det skal følge en viss mal. Johnson et al. skriver at man ved å gi et mønster et navn, oppnår en umiddelbar utvidelse av en gruppes vokabular (Johnson '92:3). Fowler er enig i dette, men påpeker at det på ingen måte er noen nyvinning. Han mener abstraksjon og konseptuell tenking er en relativt vanlig måte å uttrykke komplekse tekniske løsninger på (Fowler '97). Med denne kritikken forsøker ikke Fowler å motsi Johnson et al., men heller påpeke at det er mange andre tekniske disipliner som arbeider på samme måte. Faktisk underbygger han påstanden til Johnson et al. Ved å utvide tradisjonen for å navngi et abstrahert konsept, gir han den nærmest en universell status.

Dette underbygges for eksempel av OWASP. Ut av stiftelseserklæringen til OWASP kan man lese at et meningsbærende navn også er viktig i utformingen av angrepskomponenter. Ved å gi et bestemt sikkerhetsproblem et navn vil man lettere kunne snakke om dette bestemte sikkerhetsproblemet, gjennom det OWASP kaller for en angrepskomponent, og ikke et mønster (OWASP ’02). De forsøker altså ikke å definere mønstre, men de bygger blant annet på ideer hentet fra mønstre. For å få en pekepinn om hvor OWASP stod på dette punktet sendte jeg en e-post til Mark Curphey. Spørsmål og svar lyder som følger:

GFB: "What is your position on design/analysis patterns?"

MC: "We are big fans of design patterns.....part of the aim of the project is to take the art out of security and show it's science ;-)"

Dette svaret tolker jeg som at OWASP anser mønstre for å være en mer vitenskapelig måte å fremstille sikkerhet på. Jeg ser samtidig ikke bort i fra at Curphey har ment at mønstre ikke er vitenskapelige nok, men i så fall virker det som om OWASP har langt igjen. Slik det er nå med alle fellestrekkene mellom vanlige mønstermaler og OWASPs ASAC, anser jeg den første tolkningen for å være den mest sannsynlige.

Gamma et al. definerer et designmønster som bestående av fire hovedelementer: Navn, problem, løsning og konsekvenser. Fowler derimot anser følgende for å være det mest essensielle:

"... a statement of context where the pattern is useful, the problem that the pattern addresses, the forces that play in forming a solution, and the solution that resolves those forces." (Fowler '97:6).

3.3.1 MØNSTRE OG KRAVARBEID

I sammenheng med kravarbeid er det også et behov for gjenbruk. Hovedgrunnene til at man ønsker å benytte seg av de samme løsningene om igjen er for å spare tid, og for å være sikker på at man får en god løsning på et problem. Dette gjelder for designmønstre og kan også overføres til kravarbeid. Det mest markante eksempelet på dette er ISO/IEC standard 15408 (15408), hvor det snakkes om å opprette "Protection Profiles" (ISO/IEC '99a; '99b; '99c). Men også uavhengig av sikkerhet har flere lansert ideer om gjenbruk av løsninger innenfor kravarbeid. Dyvonne Strysick presenterte på rOOts 2002[9] en ide hun kaller for "Pluggable Use Cases" som går ut på å modulisere UC'er og bygge opp et bibliotek av generelle byggeklosser som hun selv kaller det (Strysick '02:2). På grunnlag av et bibliotek av generelle UC-moduler kan man bygge opp mer spesifikke scenarier.

[9] rOOts er en konferanse som holdes i Bergen årlig. Akronymet står for Recent Object Oriented Trends Symposium.

En annen tilnærming som ligger tettere opp mot det man vanligvis anser som mønstre er Suzanne Robertsons ”Requirements Patterns via Events / Use Cases” (Robertson '96). Hun skriver ikke om rene kravmønstre, men det hun kaller for ”requirements process patterns”. Ideen er at man skal bruke UC’er for å oppdage og definere denne typen mønstre (ibid:2).

Andre har også nærmet seg mønstre i kravarbeid på samme nivå, det vil si prosessmønstre i stedet for kravmønstre. Ett eksempel er Rawsthorne, som har skrevet en artikkel om ”A Pattern Language for Requirements Analysis” (Rawsthorne '96). Et annet eksempel er CREWS prosjektet, som har laget det de kaller for ”method chunks” som har mye til felles med tradisjonelle mønstre à la de Gamma et al. ('95) presenterer (Rolland og Prakash '00).

Det er få som har tatt tak mer spesifikke kravmønstre, men OWASP har gjort nettopp dette i sine ”Application Security Attack Components”. De kaller det riktig nok for noe annet enn mønstre, men formålet er mye det samme. Det vil si at de ønsker å legge et grunnlag for å kunne gjenbruke gode løsninger på generelle problemer, som for deres del har en tilknytning til sikkerhet i webapplikasjoner (OWASP '01; '02).

3.3.2 MØNSTRE OG SIKKERHET

Innenfor applikasjonssikkerhetsdomenet har det eksistert mønstre i alle fall siden 1997. Da lanserte Yoder og Barcalow sine ”architectural patterns for enabling application security” ('97). Dette anses for å være de første mønstrene som kan klassifiseres som sikkerhetsmønstre. I den senere tid har blant andre Markus Schumacher (Schumacher ’02) og Sasha Romanosky (Romanosky ’01, ’02a og ’02b) bidratt til å øke fokus på dette området. I tillegg har nylig Network Associates Technology Inc. presentert et bibliotek med sikkerhetsmønstre på webstedet sitt (NAT '02).

De fleste sikkerhetsmønstre har veldig mye til felles med ordinære designmønstre. Det vil si at de dreier seg om designspesifikke løsninger på sikkerhetsproblemer. De er med andre ord ikke spesielt abstrakte, og egner seg på ingen måte i et kravarbeid. Det vill blitt for konkret, og man får nettopp det problemet som Jackson og Zave snakker om (Zave og Jackson '97).

3.4 KRAVARBEID

Etter hvert som etterspørselen etter sikkerhet øker blant oppdragsgivere som kjøper spesialutviklet og / eller –tilpasset programvare, øker også behovet for å behandle sikkerhetskravene på en bedre måte. Det er det denne oppgaven kikker nærmere på fra en litt annerledes synsvinkel. I dette underkapittelet tar jeg for meg to problemer i kravarbeid. Det ene har jeg allerede vært inne på, og det handler om kommunikasjon i kravarbeidet. Det andre problemet handler om sikkerhet, og tar for seg blant annet hva Eugene Spafford mener om den saken.

3.4.1 SIKKERHET I KRAVARBEID

Sammenhengen mellom kravarbeid og sikkerhet har ikke vært høyt prioritert i systemutvikling, og er det fortsatt ikke. På grunn av fokus på system- og nettverkssikkerhet har applikasjonssikkerhet blitt tillagt liten betydning i systemutvikling, men etter hvert som sikkerhet har blitt viktigere, har også applikasjonssikkerhet kommet i søkelyset. Dermed har systemutvikling generelt, og kravarbeidets betydning spesielt, også blitt viktig for å oppnå sikkerhet. Det betyr at systemutvikling og kravarbeid også får mye oppmerksomhet, og at gode metoder til utvikling av sikrere systemer er i ferd med å bli utarbeidet. Et eksempel på dette er ”Symposium on Requirements Engineering for Information Security” (SREIS), som ble avholdt for første gang i 2001. SREIS setter blant annet søkelyset på følgende:

> **”… there has been very little attention paid to helping consumers define and create their IT security requirements. There has also been insufficient effort to bring consumers and producers of IT products and systems together to build a better understanding of what customers need in the realm of IT security and what industry is able to deliver in a cost-effective manner.” (SREIS '02)**

I den publiserte artikkelsamlingen fra SREIS 2001 ser man nettopp det som sitatet refererer til. Standarder som ISO/IEC 15408-1-3:1999 ”Information technology -- Secu-

rity techniques -- Evaluation criteria for IT security"[10] (15408) og "BS 7799"[11] får en del oppmerksomhet. Av disse er det spesielt 15408 som er blitt tatt opp til diskusjon, fordi den tar for seg sikkerhetskrav spesielt og er beregnet for både konsumenter og systemutviklere. Utover det vil jeg påpeke at problemet sitatet ovenfor peker på, er akkurat det denne oppgaven tar for seg: Hvordan man kan lage et verktøy for behandling av sikkerhetskrav som er så enkelt og forståelig at selv sluttbrukere kan dra nytte av det.

3.4.2 SYSTEMUTVIKLING OG SIKKERHETSKRAV

I sammenheng med systemutvikling er det enighet om at det har vært en rådende tendens til å neglisjere sikkerhet. En av grunnene kan være en noe grov klassifisering av krav. Sikkerhetskrav har tradisjonelt vært én av flere klasser med det man kaller "non-functional requirements" (NFR) (f.eks. Kotonya og Sommerville '98). I IEEEs standard 830-1993 defineres det 14 kravkategorier, hvorav 13 er av typen ikke-funksjonelle krav (Kotonya og Sommerville '98). Krav til sikkerhet er bare en av mange kategorier NFR.

Denne oppgaven bryter med dette NFR-paradigmet. Ved å sammenligne den generelle oppfatningen av sikkerhetskrav slik den kommer til uttrykk gjennom diverse litteratur (eks. ibid, IEEE 830-1993, Zave og Jackson '97) med fremstillingen i standarden 15408, virker det som sikkerhetskrav gjennom denne har fått en sterkere posisjon de siste årene.

15408 setter sikkerhetskrav i en sterkere posisjon blant annet fordi den definerer 11 klasser av det den betegner som "Security functional components" (ISO '99b). En funksjonell komponent kan være del av en familie, som igjen er del av en klasse. Et eksempel kan være klassen "FDP:User data protection" som blant annet inneholder komponentfamilien "Rollback (FDP_ROL)" som igjen består av to komponenter "FDP_ROL.1 Basic rollback" og "FDP_ROL.2 Advanced rollback" (ibid:43-78). Begge

[10] ISO/IEC 15408-1-3:1999 "Information technology -- Security techniques -- Evaluation criteria for IT security" (ISO/IEC '99) er teknisk sett det samme som Common Criteria 2.1.

[11] BS 7799 er også kjent som ISO/IEC 17799:2000 "Information technology -- Code of practice for information security management" (ISO/IEC '00).

disse komponentene beskriver på en konseptuell måte hvordan operasjoner bør rulles tilbake, avhengig av hvilket arbeidsområde de har.

Beskrivelsen av en familie har mye til felles med en standard mønstermal, men er mer abstrakt i sin form. Det vil for eksempel si at den ikke er eksplisitt i sin tilnærming til problem og løsningsbeskrivelser, slik mønstre er. Oppbygning og konseptene med mønstre kommer jeg tilbake til senere i oppgaven.

Gjennom 15408 blir sikkerhet en vesentlig del av kravarbeidet og utviklingsprosessen. Den fremsetter det som vesentlig at sikkerhet bør inkorporeres i starten av utvikingen. Det faktum at dette også er blitt en internasjonal standard peker også på at det er bred enighet om at sikkerhetskrav er viktig.

For å gjøre et godt kravarbeid er det viktig å holde seg på et konseptuelt nivå og ikke være forutinntatt med hensyn til implementering (Zave og Jackson ’97). Når man nærmer seg sikkerhet, og spesielt programvaresikkerhet, blir det fort et detaljert og teknisk fokus. Innen programvaresikkerhet er det naturlig å snakke om ”buffer overflow” og ”race condition” i stedet for mer abstrakte problemer som ”Innputtmanipulering” og ”logiske hull”. Dette kan blant annet forklares med sikkerhetsproblemenes tekniske publikasjonskanaler (BugTraq, Vulnwatch, CERT/CC og så videre). Eksempelet under viser kun en innledende oversikt i en ”Vulnerability Note” fra CERT/CC:

> **”Buffer overflow vulnerabilities exists in the DNS stub resolver library used by BSD, ISC BIND, and GNU glibc. Other systems that use DNS resolver code derived from ISC BIND may also be affected. An attacker who is able to control DNS responses could exploit arbitrary code or cause a denial of service on vulnerable systems.” (CERT/CC ’02).**

Med denne typen problembeskrivelser kan det være vanskelig å få øye på mønstre og generelle problemer. Ser man på hva det egentlig dreier seg om, en ”buffer overflow”, så vet noen at dette er én av mange typer manipulering av innputt som kan føre til mer eller mindre alvorlige sikkerhetsbrudd.

Ideen med 15408 er at den skal fungere som et rammeverk for utvikling av generelle ”Protection Profiles” (PP). En PP skal beskrive et sett med generelle sikkerhetskrav innenfor et bestemt programvaredomene. På grunnlag av en slik PP utvikler man så mer

kontekstavhengige ”Security Targets” (ST) (ISO ’99a:31). En PP definerer et sett med sikkerhetskrav uavhengig av implementeringsdetaljer, og en ST inneholder sikkerhetskravene til ett bestemt IT-produkt eller -system (ibid:6, 37 og 43). For som det sies i del 1 - introduksjon og generell modell - så er 15408 ment å støtte utviklere ”… in identifying security requirements …”, og å gi sluttbrukere ”… an implemetation-independant structure termed the Protection Profile …” (ibid:8-9).

3.4.3 IMPLIKASJONER AV 15408

Både offentlige myndigheter i mange vesteuropeiske land, og det profesjonelle sikkerhetsmiljøet, synes å legge større og større vekt på sikkerhetskrav. ISO/IEC anser det som viktig å kunne knytte forbrukers krav til det ferdige produktet; validering i kravarbeidsterminologi. Sikkerhetskrav er satt på dagsorden og kommer foreløpig sterkest til uttrykk i 15408.

Et annet viktig aspekt ved denne standarden er det faktum at det går an å effektivisere ved å benytte seg av det standarden betegner som ”Protection Profiles” (PP). På et mer abstrakt nivå kan en PP anses for å være et slags mønster, en form for sikkerhetsmønster om man vil. Dette kan anses som at det legges opp til effektivisering av utviklingsprosessen ved å legge til rette for gjenbruk av sikkerhetskrav.

Et tredje og siste poeng som er verd å merke seg med 15408 er at den er ment som et rammeverk for forbruker.

En svakhet ved denne siden av standarden er dens tekniske tilnærming. Ved å velge en mindre teknisk tilnærming beskrevet med en terminologi nærmere opp mot dagligtale, vil det være enklere for flertallet å forstå også komplekse sikkerhetsproblemer.

4 KONSEPTUELL FASE

I forrige kapittel ble det gjort rede for de konseptene som ligger til grunn for den konseptuelle fasen. Her legges det vekt på å fremstille selve utviklingsprosessen, så vel som resultater av denne. Underveis vil forskjellige alternativer bli presentert og diskutert.

Jeg nevnte i introduksjonen at oppgaven er utført som en iterativ og inkrementell prosess. Dette gjelder kanskje spesielt i dette kapittelet. Jeg begynner med å forklare litt om prosessen frem mot AMM, før jeg går løs på de forskjellige iterasjonene.

4.1 PROSESSEN FREM MOT ABSTRAKTE MISBRUKSMØNSTRE

Prosessen frem mot AMM foregår mer eller mindre parallelt med litteraturstudiet. Første iterasjon leder frem mot de første mønstrene, som i all hovedsak er bygget opp på grunnlag av MC-notasjonen. Etter hvert fokuserer jeg mer på prinsipper fra mønstre og ender til slutt opp med et forslag som i all hovedsak er basert på mønstre i sin utforming og på trusselmodellering i sin tilnærming til sikkerhet.

I forrige kapittel viste jeg at det er avgjørende forskjeller på applikasjonssikkerhet og andre typer sikkerhet. Det er ikke dermed sagt at de forskjellige typene ikke har noe med hverandre å gjøre. Sikkerhet i sin helhet er et komplekst konglomerat av hensyn som må ivaretas. På tross av dette vil prosessen med å utvikle programvare, enten det er et omfattende ERP-system[12] eller en enkel webapplikasjon, bare ha direkte innvirkning på den interne sikkerheten i programmet som utvikles. Selvfølgelig kan det utarbeides f.eks. sikkerhetsrutiner relatert til programmet, men i så fall blir dette en tilleggsaktivitet og faller i beste fall inn under ikke-funksjonelle sikkerhetskrav.

Man kan argumentere for at mange typer sikkerhet rettmessig hører inn under et kravarbeid, men det er kun applikasjonssikkerhet som direkte berører det utviklede programmets egen sikkerhet. Rent fornuftsmessig kan man argumentere for at andre typer sikkerhet er bedre overlatt til eksperter på disse typene sikkerhet. En programmerer,

[12] ERP står for Enterprise Resource Planning system, slik som SUN og SAP.

systemerer eller databaseutvikler har i utgangspunktet ingen faglig bakgrunn som tilsier at han eller hun skal kunne sørge for strategiske sikkerhetsrutiner eller en sikker infrastruktur.

Mellom applikasjonssikkerhet og andre typer sikkerhet er det altså et skille. Grovt sett kan man si at skillelinjen går mellom funksjonelle og ikke-funksjonelle sikkerhetskrav. I den grad et sikkerhetskrav er funksjonelt, berører det applikasjonen direkte, noe som ikke er tilfelle med ikke-funksjonell sikkerhet. Det er ikke dermed sagt at grensetilfeller ikke forekommer.

La meg eksemplifisere med autentisering og hensynet til passord. Kravet fra oppdragsgiver er mest sannsynlig lite spesifikt, for eksempel ”Innloggingen må være så sikker som mulig” eller noe i den retning. Det er mange måter å løse dette på, men la oss anta at det innebærer bruk av passord. Da er det mange problemer å ta hensyn til: Hvilke kompleksitetskrav man skal stille til hvert passord, hvor mange dager passordet skal være gyldig før det må skiftes ut osv. Det er lett å se at dette både har funksjonelle og ikke-funksjonelle implikasjoner. Et interessant spørsmål å stille er hvilke oppgaver som bør tilfalle et utviklingsteam og ikke. Ideelt sett bør utviklingsteamet ha en sikkerhetsekspert som kan ta hensyn til de ikke-funksjonelle sikkerhetskravene relatert til systemet under utvikling, men det anser jeg for mindre viktig i denne oppgaven og vil ikke ta det opp til videre diskusjon her.

Når man går et programs interne sikkerhet etter i sømmene, beveger man seg fort inn på et teknisk nivå som ikke hører hjemme i et kravarbeid. Det er derfor avgjørende å få et innblikk i hvilke sikkerhetsproblemer og løsninger som er aktuelle innenfor det domenet en bestemt applikasjon utvikles. Ikke minst er det avgjørende å finne ut hvordan dette kan angripes fra et abstrakt og konseptuelt plan, slik at det lar seg forene med kravarbeid.

I oppgaven er det som nevnt tatt utgangspunkt i web- og e-handelsapplikasjoner. For å finne sikkerhetsproblemer relatert til disse typene applikasjoner ble e-postlisten BugTraq forsøkt analysert og kategorisert. Dette viste seg å være en meget omfattende operasjon og lå langt utover denne oppgavens omfang.

Dermed fokuserte litteraturstudiet i stedet på klassifiseringer av sikkerhetsangrep og sårbarhet. Gamle klassifiseringer så vel som nye, og dessuten én klassifisering som fortsatt er under utvikling, ble gjennomgått. Det viste seg at den mest relevante klassifiseringen med hensyn til applikasjonssikkerhet innenfor webapplikasjonsdomenet var den uferdige. Denne klassifiseringen er utviklet av OWASP, som fortsatt var i ferd med å utvikle ASAC når denne oppgaven ble ferdigstilt.

Da det sikkerhetsmessige utgangspunktet var på plass, startet modelleringen. Målet var å lage modeller av utvalgte angrepskomponenter fra ASAC-klassen ”Input Validation”. I tillegg til ASAC ble grunnlaget for modellene supplert med annen litteratur (OWASP ’01; ’02).

4.1.1 BALANSEN MELLOM MODELL OG TEKST

Det neste problemet som oppstod, var i forbindelse med utformingen av de abstrakte misbruksmønstrene. Hvilken rolle skal modellen spille? Bør den være først eller sist? Bør den være på et helt konseptuelt nivå, eller bør den være mer spesifikk og heller fungere som eksempel? Disse spørsmålene ble først besvart etter mye modellering, diskusjon med veileder og generell prøving og feiling. I det følgende underkapittelet presenteres et par eksempler i kronologisk rekkefølge. Hensikten er å vise hvilke steg som er gjennomgått i prosessen frem til det ferdige resultatet.

Tidlig i utviklingen av konseptet var det fokus på modellering og bruk av MC-notasjonen. Mønstre ble utformet på grunnlag av Sindre og Opdahls MC-maler. Av den grunn benyttet de første AMM få momenter hentet fra mønsterkonseptet. Etter egen refleksjon over formålet med mønstrene, samt diskusjon med veileder og medstudenter, ble det laget en annen tilnærming. Denne tilnærmingen satte mønsterkonseptet i sentrum. Under følger en kort redegjørelse for de to tilnærmingene.

4.1.2 MED SCENARIET I FOKUS

Både modellering etter UC- / MC-prinsipper og konsepter fra mønstre har som formål å lette kommunikasjon. Men modeller kan virke enklere å forstå enn en rent tekstlig beskrivelse. Derfor er det fokusert primært på MC-modellen i den første iterasjonen av AMM-utviklingen.

En MC-scenario består som nevnt av to deler: En MC-modell og en tilhørende tekstlig beskrivelse. Til å begynne med er det vanskelig å løsrive seg fra denne tankegangen. Det er grunnen til at malene til Sindre og Opdahl , hentet fra ”Templates for Misuse Case Description”, ble brukt som utgangspunkt for de første AMM’ene. Denne tilnærmingen er mer scenariobasert enn den påfølgende. Med scenariobasert mener jeg at den legger vekt på modellen og hva som foregår eller kan foregå i den. Denne tilnærmingen blir for teknisk til at en domeneekspert kan bruke forstå den og dermed kommunisere med den som verktøy.

Ved å legge vekt på scenarier oppstår det to problemer. Det ene er at scenariet ikke er ment for gjenbruk. I og med at scenarier beskriver én bestemt situasjon er det ikke ment å være generelt. Det andre er relatert til det kommunikative aspektet ved AMM. Scenarier er enkle for en utvikler eller andre med teknisk bakgrunn, men de kan likevel være vanskelige å forstå. Det skyldes blant annet terminologien de baserer seg på. Den er presis og beregnet på folk med IT-teknisk bakgrunn. Man kan si at forklaringsmodellen tilhører et teknologisk og ikke et allment miljø. Det blir med andre ord vanskelig for domeneeksperter og / eller sluttbrukere å forstå problemet på grunn av forklaringsmodellen som er valgt.

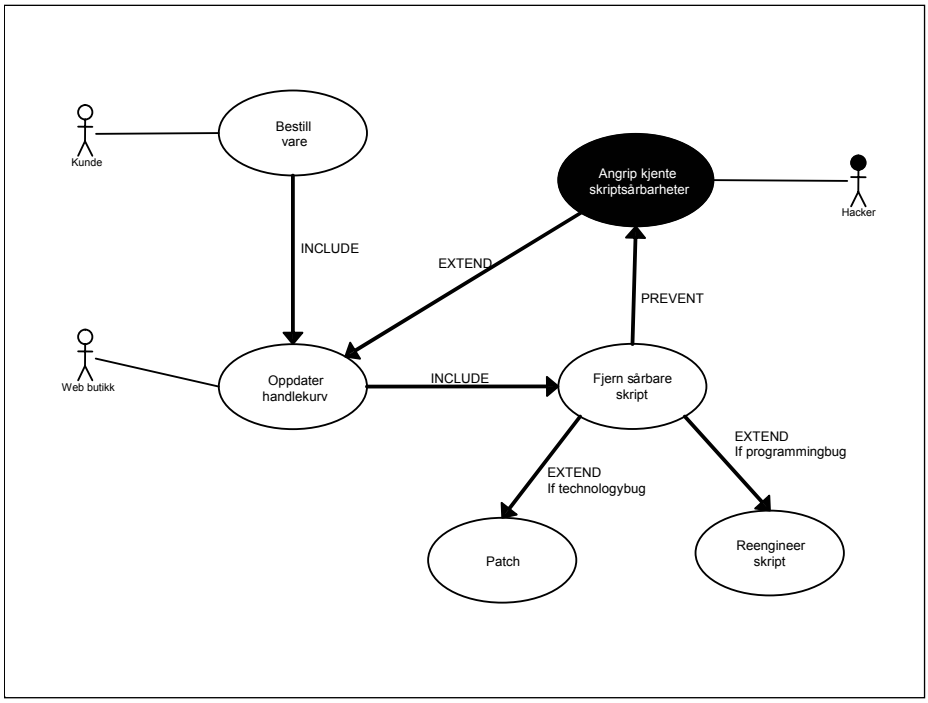
Dette strider med andre ord mot hovedhensikten med å basere AMM på mønstre. De er ment å ha en medierende effekt mellom kravarbeidere ved diskusjon av sikkerhetskrav. I tillegg er de ment å ha en effektiviserende effekt på kravarbeidet på grunn av muligheten for gjenbruk.

Dermed er det første utviklingsforslaget med hovedvekten på scenarier forkastet. Scenarier blir flyttet til en senere fase i det som viser seg å bli en metode med utgangspunkt i AMM.

På neste side følger et AMM med MC-modellen i fokus. Dette mønsteret er på engelsk, fordi det språket ble brukt i en periode helt i starten av hovedfaget. En diskusjon av problemer og løsninger på disse følger etter presentasjonen av mønsteret.

SCRIPTWHACKING

Model:



Name:

ScriptWhacking

Alias:

Script Hacking; Hacking Scripts; Exploiting Bugs; Server side Script Vulnerabilities

Summary:

A Hacker finds a script that has bugs or vulnerabilities in the site structure and uses this to gain access to what would otherwise be inaccessible to him.

Basic path:

bp0: (step bp0-1) The Hacker has used "Webspionage" (Web pilfering) to get to know the most intimate details about your firm's website and it's structure. (step bp0-2) This gives the hacker information about how to gain privileged access rights to your site by misusing a badly written cgi-script or a buggy asp-script. (step bp0-3) Getting this access the hacker can do all sorts of bad things, like (step bp0-4) steal credit card information from your trusting customers, for instance, thereby ruining your reputation for an extremely long time, maybe for life.

bp1: A less experienced hacker, often called a ScriptKiddie, (step bp1-1) uses a downloaded tool to scan your site for known script vulnerabilities, finds one or more vulnerabilities and (step bp1-2) follows a recipe on how to misuse this information. This can cause damage even greater than bp0, because you now probably have a complete novice rampaging through your web store causing all kinds of damage.

Alternative paths:

ap2: The hacker downloads a scanner tool from the Internet which tells him which vulnerable scripts your site uses (changes bp0-1).

ap3: The script kiddie shares his information with all members of his "hacker group" and several of the group's members launch a massive attack within a specifically planned timeframe (changes bp1-2).

ap4: The hacker does not compromise you or your customers at your point of sale, but uses the information gathered at your site to compromise your clients elsewhere (changes bp0-4 and bp1-2).

Tabell 1 – AMM med fokus på eksempelscenario.

En indikasjon på at dette ikke var den rette formen, er måten MC-malen nærmest er presset over modellen på. En MC-mal er i utgangspunktet en beskrivelse av ett enkelt misbrukstilfelle, mens den her er forsøkt brukt for å beskrive et helt scenario. Både problem og løsning, som er to essensielle enheter i mønsterkonseptet, kommer kun implisitt frem av navn og modell. Derfor blir denne tilnærmingen fort vanskelig tilgjengelig for personer uten sikkerhetsbakgrunn[13].

Et abstrakt misbruksmønster skal beskrive et konseptuelt nivå og vil derfor dra nytte av et forenklet, men mer konkret eksempel. Dette har flere fordeler. Én fordel er at det blir mindre tilfeldig hvordan problemet eksemplifiseres. En god løsning kan være å bygge opp AMM av en meget abstrakt del i form av den tekstlige mønsterdelen, og et litt mer konkret eksempel i form av MC-modell med forklaring. Eksempelet så vel som mønsteret tar ellers hensyn til at det skal være enkelt å formidle og så enkelt som mulig for en sluttbruker å forstå. En annen fordel er at det blir konsistens i måten mønsteret eksemplifiseres på. Man kan i større grad være sikker på at det eksemplifiseres korrekt når dette tilbys via mønsteret.

En av hovedhensiktene med abstrakte misbruksmønstre blir dermed å oppnå en gjennomført abstraksjon og forenkling, uten å miste kontakten med det tekniske grunnlaget. På dette området skiller abstrakte misbruksmønstre seg fra andre mønstre, også andre kravarbeidsmønstre. Abstrakte misbruksmønstre skal defineres på en slik måte at de enkelt lar seg formidle til personer med liten eller ingen teknisk bakgrunn. På den måten vil de danne et godt grunnlag for kommunikasjon mellom kravarbeidere med ulik bakgrunn. Dette kan så skape et bedre grunnlag for felles oppfatning av problemet og dermed forståelse hos den enkelte person, og videre vil det kunne føre til et bedre grunnlag for enighet kravarbeidere i mellom.

Tradisjonelt beskrives et mønster i tekstlig form og eksemplifiseres eventuelt med en modell for å gjøre problem og løsning klarere. Eksempler på dette finner man både i Gamma et al. ('95) og i Fowler ('97). I det neste eksempelet er dette fulgt opp, men der dukker nye problemer opp.

[13] Her har jeg skrevet sikkerhetsbakgrunn og ikke IT-teknisk bakgrunn, fordi det, selv for en utvikler, kan blir vanskelig å forstå problemet hvis han ikke kjenner det fra før.

4.1.3 MED MØNSTERET I FOKUS

I denne iterasjonen begynner konseptet å ta form. Etter at mønstre med utgangspunkt i MC-maler var forsøkt og forkastet, ble hovedfokus satt på mønsterdelen av AMM. Bygget på de tre prinsippene jeg nettopp nevnte, fremstår dette som et bedre forslag, og mer i samsvar med oppgavens problemstilling. Nå blir fokus lagt på det medierende aspektet i mønstrene, fremfor at det skal representere et eksemplifiserende scenario.

Forslaget til løsning trer klarere frem. AMM bør utformes med formidling som hovedformål, og bør tone ned scenariet til kun å være en forenklet eksemplifisering av problemet.

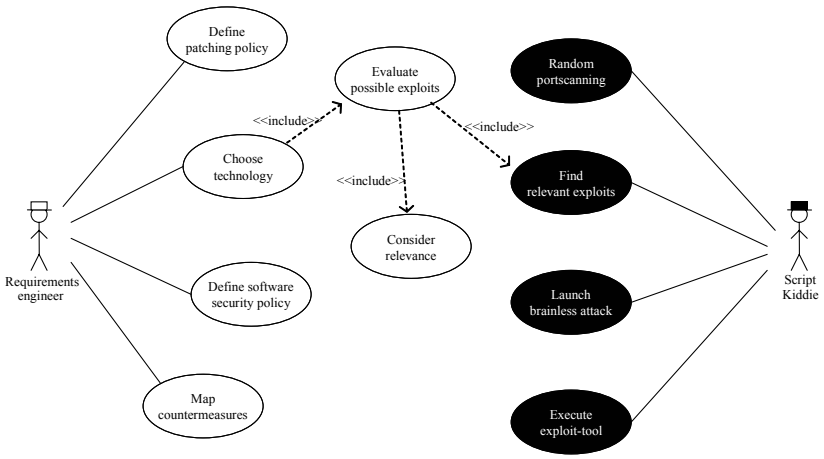
4.1.3.1 Mønsteret "KiddieKiller"

Nå følger et av de første forsøkene på å utforme mønster av denne typen. Mønsteret er kalt "KiddieKiller" og viser tilbake på begrepet "script kiddies"[14], og hvordan man skal gå frem for i størst mulig grad å unngå angrep fra denne klassen med crackere. Problemet er høyst aktuelt, men peker tydelig på behovet for et skille mellom utvikling av sikkerhet i en applikasjon og utvikling av gode sikkerhetsrutiner, både i utviklingsteam og i bedrifter.

[14] Se teorikapittelet for nærmere forklaring av begrepet "Script Kiddie".

KIDDIEKILLER

Model:



Problem:

A large number of the computer security incidents that are reported are performed by so called script kiddies. These are usually youngsters with no other incentive than to gain access to someone else's computer and cause harm. This means that the problem will apply to anyone that is at some point in time connected to the Internet. All machines connected to the Internet are at some point or another scanned for possible exploit-prone possibilities. New exploits turn up all the time. No network, machine or application is safe as long as the script kiddies are on the rampage.

Solution:

Make technological choices on behalf of the security issues known at any point in time.
Evaluate possible exploits in chosen technology.
Carefully consider the relevance in regard to the application under development.
Map possible countermeasures.
Define a software security policy.
Define a patching strategy.
Follow it all up throughout the development lifecycle.

Consequences:

This puts security on the agenda so that it will be a major consideration in the developing process.

Tabell 2 - AMM med fokus på mønsterkonsepter.

Mønsteret taler i stor grad for seg selv, men noen kommentarer er nødvendige. Blant annet er begrepet ”policy” brukt flere steder. Det er ikke tilfeldig og er på ingen måte i strid med den generelle holdningen i resten av oppgaven, som anser en ”policy” for å være svaret på et ikke-funksjonelt krav. Forklaringen er at dette mønsteret ble utviklet på et tidspunkt da litteraturstudiet fortsatt pågikk og forholdet mellom funksjonelle og ikke-funksjonelle sikkerhetskrav ikke var helt avklart.

Utover dette er det et annet problem som er verdt å kommentere. Det går ut på at det i trusselmodellering kan være vanskelig å opprettholde et skille mellom ulike modelleringsdomener. Dette gjelder spesielt ved utvikling av webapplikasjoner, men kan også være aktuelt i andre tilfeller. Utvikling av webapplikasjoner er spesiell fordi applikasjonstypen er dynamisk slik at utvikling gjerne fortsetter etter at applikasjonen er tatt i bruk.

Å hindre angrep fra script kiddies er et høyst relevant problem, men som tidligere påpekt faller det ikke direkte inn under det man kan kalle for applikasjonssikkerhet. Det må med andre ord klassifiseres som et ikke-funksjonelt krav. Som mange andre sikkerhetsproblemer handler det om å ha tilstrekkelig gode rutiner for å gjøre noe som påvirker applikasjonen indirekte. Et eksempel på dette er gode passordrutiner. Applikasjonssikkerheten kan være veldig god, men hvis ikke brukerne slutter å klistre en huskelapp for passord på skjermen, vil det ikke hjelpe stort. Dette er et godt eksempel på et sikkerhetsproblem som bør løses utenfor utviklingsarbeidet. Denne klassen med problemer representerer problemer som i større grad berører bedriftskultur og –rutiner, enn systemutvikling. Allikevel kan det være nødvendig å sette opp et sett med regler for bruk av en applikasjon, men det vil i så fall være opp til bedriften å omsette disse til praksis.

Hvis og når man modellerer ikke-funksjonelle krav, som policy-eksemplene i mønsteret over, ender man fort opp med å måtte trekke inn eksterne aktører fra flere forskjellige domener, eller ”verdener”, som Mylopoulos, Borgida, Jarke og Koubarakis snakker om i sin artikkel ”Telos: A Language for Representing Knowledge About Information Systems” (Mylopoulos, Borgida, Jarke og Koubarakis '90). De mener det eksisterer fire domener som er av interesse for kravarbeidet ved utvikling av informasjonssystemer: Forretningsdomenet (”subject world”) systemdomenet (”system world”), bruksdomenet

(”usage world”) og utviklingsdomenet (”development world”) (ibid:20). Slik jeg forstår det, er Mylopoulos et al. i mot å blande de ulike modellene i en og samme modell.

Det kan ha visse fordeler med å la disse domenene overlappe i sammenheng med trusselmodellering. I sin artikkel om utredning av sikkerhetsløsninger i tog skriver Ian Alexander for eksempel at kan det være en fordel å trekke inn utvikleren i MC-scenarier (Alexander '02). Grunnen til dette er at man må se sammenhengen mellom systemet under utvikling og det miljøet systemet skal operere innenfor. Dette er forhold som kan ha avgjørende innvirkning på hvordan systemer bør konstrueres.

Et eksempel er forholdet mellom mulige løsninger på en sikkerhetstrussel, den teknologien som velges for å utvikle et system og de mulighetene dette gir. Det handler om å ta hensyn til at de rette rutinene både under utvikling og under drifting av en webapplikasjon. Ved utvikling av en webapplikasjon er det vanskelig å skille mellom applikasjonssikkerhet og infrastruktursikkerhet. En av grunnene til dette kan være den kompleksiteten i en distribuert webapplikasjon. En webapplikasjon kan i de fleste tilfeller like korrekt omtales som et system, fordi det er en sammensetning av en rekke ulike mer eller mindre frittstående komponenter.

4.2 DET ENDELIGE RESULTATET

I dette underkapittelet presenteres det endelige resultatet av denne fasen. Det er blitt et kompromiss mellom scenario- og mønstertilnærmingen. For å kunne fungere som kommunikasjonsmiddel i et meget teknisk og vanskelig tilgjengelig domene, er de abstrakte misbruksmønstrene skrevet i et så enkelt og lite teknisk språk som mulig. Dermed bør også kommunikasjon mellom kravarbeidere med lite teknisk kompetanse kunne bidra med å øke forståelsen rundt den konseptuelle vurderingen og tenkningen som foregår under kravarbeidet. De vil kunne forstå de konseptuelle problemene og løsningene som kan være aktuelle i deres scenario, og dermed enklere kunne ta avgjørelser basert på forholdet mellom risiko og kostnad.

4.2.1 UTFORMING AV MØNSTERMAL FOR AMM

Her er det ikke tatt utgangspunkt i noen bestemt mønstermal, men i de mest vanlige elementene i forskjellige maler og beskrivelser. Som nevnt i kapittel 4 er dette de fire elementene navn, motivasjon, problem og løsning.

Forslaget til mønstermal for abstrakte misbruksmønstre er som følger:

Navn → Et beskrivende navn som er ment å implisere hvilket problem som løses.

Alias → Skal liste opp eventuelle andre navn problem - løsningsparet kan være kjent under.

Motivasjon → Konteksten problemet oppstår i (også kalt ”forces” eller kontekst). Dette er en meget viktig del av mønsteret. Det skal vise hvilke forutsetninger som må/kan være til stede for at problemet skal kunne oppstå.

Problem → Beskriver kort kjernen i problemet.

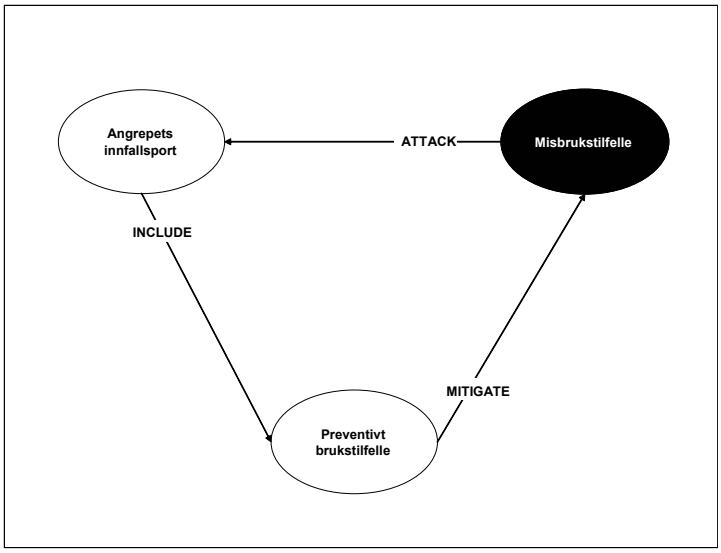
Løsning → Beskriver den velkjente løsningen på problemet. Det vil si at det skal være en løsning som er utprøvd og ansett som å være en god løsning på akkurat det problemet som mønsteret beskriver.

Eksempel → Modell av et representativt og eksemplifiserende MC-scenario med beskrivelse. Dette bør være noe mer konkret enn den tekstlige forklaringen.

Relaterte mønstre → Peker til andre mønstre som har en relevans i forhold til dette. En eventuell referanse til et annet mønster oppgis i form av navnet til det aktuelle mønsteret.

Tabell 3 – Mønstermal brukt i AMM

For å legge vekt på mønsterformen blir modellen tonet ned og plassert under overskriften eksempel, til fordel for de fire mønsterelementene. Eksempelet er for AMMs del en konseptuell skisse av det tenkte problemet, uttrykt med hjelp av MC-notasjonen. Figuren under representerer en metamodell som har til hensikt å vise hvordan notasjonen er tenkt benyttet:



Figur 7 - Metamodell for AMM-mal

Modellen presenterer en forenklet MC-scenario hvor formålet er å vise grafisk hvordan én bestemt UC blir angrepet og hvilket preventivt brukstilfelle som er ment å demme opp for det.

De andre elementene som er nevnt er med i malen fordi de gjør det enklere å sette mønstrene i sammenheng, i tillegg til at de øker forståeligheten. For eksempel kan det være viktig å se sammenhengen med andre AMM. Dette finner man i så fall umiddelbart i elementet "relaterte mønstre", som skal peke til andre relevante mønstre.

4.2.2 FRA ANGREPSKOMPONENTER TIL ABSTRAKTE MISBRUKSMØNSTRE

I det følgende presenteres en del sitater for å vise hvilket grunnlag den siste iterasjonen bygger på. Formålet med denne presentasjonen av sitatene er å vise noe av prosessen frem til de abstrakte misbruks- og brukstilfellene som brukes i det ferdige eksempelet på AMM. Dette AMM'et er også grunnlaget for den evaluerende fasen (kapittel 5).

For å komme frem til et mønster som kan brukes i pilotundersøkelsen, er det avgjørende å lage et relevant og troverdig eksempel. Utover disse føringene er det valgt ut noen representative sitater. Innholdet i disse sitatene blir abstrahert, og både preventive brukstilfeller og misbrukstilfeller er definert på grunnlag av denne abstraksjonen. I denne sammenheng kan et treffende sitat fra Martin Fowlers bok om analysemønstre være verdt å ha i bakhodet:

> **”Conceptual models are linked to interfaces (types), not implementations (classes)” (Fowler 01:2)**

Abstraksjonsnivået er forsøkt tilpasset et nivå for formidling som kan fungere under kravarbeid generelt, og som utgangspunkt for mer inngående scenariebygging med hjelp av UC-modeller spesielt. Det er altså meningen at mønstrene skal være lette å formidle og lette å forstå. De skal fungere som grunnlag for diskusjon og ideelt sett danne et godt grunnlag for å komme til enighet om bestemte konseptuelle løsninger.

4.2.2.1 Fokus på ”Innputtmanipulering”

For å eksemplifisere oppbygningen av et AMM, er det valgt ut én bestemt klasse med angrep fra OWASP-repertoar. Beskrivelse av problem og løsning presenteres og analyseres. På dette grunnlaget abstraheres misbrukstilfeller og preventive brukstilfeller. Klassen er valgt fordi den utgjør en klassisk gruppe angrep som får mye omtale i litteraturen. Som det følgende sitatet hentet fra Scambray, McClure og Kurtz viser, så dreier hele problemet seg om manglende validering og rensing av innputt fra brukeren. Brukeren kan manipulere innputten på en slik måte at han omgår den funksjonaliteten applikasjonen er tiltenkt.

> **”The basic problem arises from the inadequacy of sanitizing the input to a particular script. Without validation and sanitizing, it is possible for attackers to submit a particular character, along with a local command, as a parameter and have the web server execute it locally.” (Scambray et al. '01:573-574)**

Garfinkel og Spafford er også opptatt av dette problemet og sier f.eks. ”An astonishing number of security-related bugs arise because an attacker sends an unexpected value or an unanticipated format to a program …” (Garfinkel og Spafford '97:300)

Det sies å være god programmeringsskikk å programmere på en slik måte at systemet kun tillater de verdiene det er ment å behandle. Alt annet skal utelukkes på en eller annen måte. Dette bringer en tilbake til det faktum at de aller fleste sikkerhetsbrudd skyldes bugs, eller med andre ord dårlig programmering. Kurtz et al. ('01) opererer allikevel med et skille her og skriver f.eks. om sårbarhetstilfellet ”Showcode.asp”:

> **"… the difference with this vulnerability is that it wasn't a bug per se, but more an example of poor programming …" (Scambray et al. '01:283)**

Slik jeg oppfatter det er det uvanlig å skille mellom det som kalles for bugs og dårlig programmering, slik Scambray et al. gjør her. Både Garfinkel og Spafford, Wheeler, Viega og McGraw og flere, gir generelt inntrykk av at løsningen for å unngå bugs er å følge regler for god programmering (Garfinkel og Spafford '97; Peteanu '01; Viega og McGraw '02; Wheeler '02).

Garfinkel og Spafford skriver under overskriften "Rules To Code By" at de har kommet fram til en liste over gode prinsipper for programmering ('97:300). I utdrag fra denne listen legger de vekt på to hovedpunkter. Det ene er at man bør planlegge systemet grundig før man setter i gang, og det andre er at man må sjekke all innputt fra brukeren. Videre legger de vekt på hvor viktig det er å behandle absolutt alt en bruker sender inn til systemet (Garfinkel og Spafford '97).

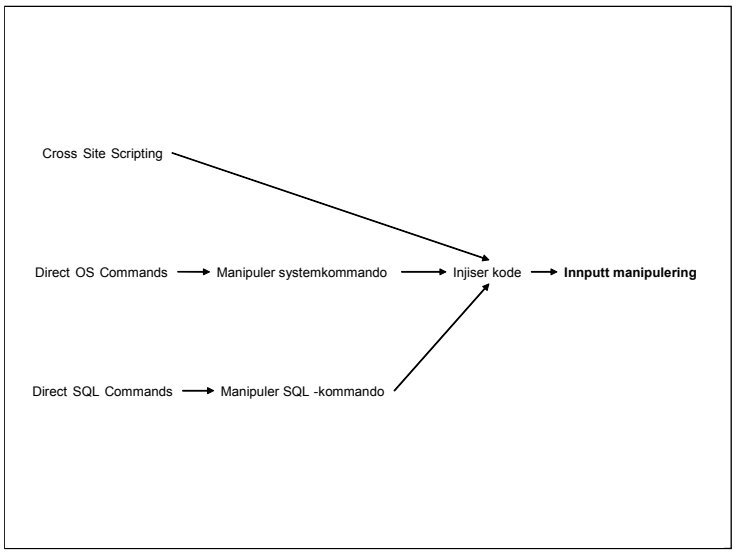
Problemer som følge av dårlig programmering kan altså i følge mange løses ved å ta hensyn til prinsipper for hvordan oppnå det motsatte, altså god programmering. Det er slike prinsipper som det, med hensyn til sikkerhet, kan være verdt å definere ved hjelp av AMM. Med andre ord understrekes det nok en gang at applikasjonssikkerhet handler om god programmering. Unngår du bugs, unngår du en masse sikkerhetsproblemer også.

Man kan kanskje si at AMM er et forsøk på å sette god programmeringsskikk i system, og å presentere det på en måte som kan virke forståelig for flest mulig kravarbeidere. Ved å presentere problemet så vel som løsningen er det kanskje lettere å få øye på mønsterets aktualitet. For kravarbeidere med forskjellig bakgrunn og vidt forskjellig begrepsapparat, vil det sannsynligvis være enklere å enes om en bestemt løsning hvis man får presentert et eksemplifisert problem som forklarer behovet og logikken bak den aktuelle løsningen. AMM er ment å fylle nettopp denne rollen.

4.2.3 ANALYSE AV INNPUTTMANIPULERING

I det følgende presenteres sitater fra ASAC i tabellform. Sitatene er tatt fra tre av angrepene som beskrives i klassen ”Input Validation”. Tabellene inneholder tre kolonner, der den første oppgir hvilken angrepskomponent det dreier seg om, den neste inneholder relevante sitat og den tredje kolonnen redegjør for hvilke vurderinger som er foretatt med hensyn til å abstrahere henholdsvis misbruks- og preventive brukstilfeller. Legg merke til at dette bare er eksempler som er ment å vise hvordan prosessen med å trekke ut relevante abstraksjoner til bruk i de abstrakte misbruksmønstrene har foregått. Det er med andre ord på ingen måte en utfyllende presentasjon av grunnlaget for det resulterende mønsteret Innputtmanipulering. Den første tabellen resulterer i misbrukstilfeller, deretter følger en annen tabell med deres respektive preventive brukstilfeller. Begge tabellene fungerer etter samme logikken. Andre kolonne presenterer et sitat fra henholdsvis problem- og løsningsbeskrivelse. Den siste kolonnen presenterer analyse av andre kolonne hvor henholdsvis misbrukstilfeller og preventive brukstilfeller defineres. Disse abstraherte tilfellene representerer deretter alle de tre komponentene på et mer abstrakt nivå.

Figuren under viser hvordan de spesifikke angrepskomponentene blir abstrahert opp til et felles misbrukstilfelle. Samme fremgangsmåte blir brukt i tabellen for preventive brukstilfeller:



Figur 8 - Viser logikken bak abstraksjonen i Innputtmanipulering

Angrepskomponent	Problembeskrivelse	Misbrukstilfeller
Cross-Site Scripting	"… Outside or third-party input data received by a web application resulting in a HTTP Client-Side language executing within the users web environment achieving the same level of access and privileges as the hosted domain. Client-Side Scripting (CSS) attacks require the execution of Client-Side Languages (JavaScript, Java, VBScript, ActiveX, Flash, etc.) within a user's web environment. …"	Utviklere kan lett overse at innholdet som puttes inn i et skjema bør evalueres før det legges ut i f.eks. en gjestebok. Hvis dette ikke blir gjort grundig, blir det lett for en hacker å legge til skadelig kode i et tilsynelatende "uskyldig" tekstfelt. Det kan føre til at neste man som mer eller mindre tilfeldig går inn på kommentarsidene får kjørt en "ulovlig" kode i sin browser. Ut i fra dette kan misbrukstilfellet "**Injiser kode**" abstraheres, eller enda mer abstrakt "**Innputtmanipulering**".
Direct OS Command	"… Executing of operating system commands with the aid of an unchecked input-parameters including form-parameters, cookies, http request-headers and malicious data in uploaded files. Succeeding with such a technique makes it possible to execute system-commands with the same permissions of the web-server (tier component). System commands are normally a very convenient feature. With little effort it is possible to add functionality to your web-application. …"	Hvis man bruker systemkommandoer i et webscript for å f.eks. få lagret noe persistent, må man være klar over de truslene dette utgjør. En hacker kan utnytte slike kommandoer hvis scriptet ikke evaluerer den innputten det får. Abstrahert kan man si at en hacker utnytter misbrukstilfellet "**Manipuler systemkommando**", eller enda mer abstrahert kan man si at han sender et kall ved hjelp av misbrukstilfellet "**Injiser kode**". Ytterligere abstrahert kan dette også anses som "**Innputtmanipulering**".
Direct SQL Commands	"… Direct SQL commands (also known as SQL injection) is a technique used to alter existing or create new SQL queries by exploiting vulnerabilities in inadequate or inexistent input validation routines. With SQL injection it is possible to retrieve data from the data bases and in some instances from the underlying operating system. …"	Ved å endre på SQL strenger i en database tilknyttet webapplikasjon kan en hacker få tilgang til informasjon han ikke skulle hatt tilgang til. Dette leder naturlig til misbrukstilfellet "**Manipuler SQL-kommando**", mer abstrakt kan det omtales som "**Injiser kode**" og igjen enda mer abstrakt "**Innputtmanipulering**".

Tabell 4 – Fra angrepskomponenter til misbrukstilfeller.

I tabellen ser vi at "Direct OS-" og "Direct SQL commands" i det minste har ett fellestrekk. De handler begge om manipulering av argumenter. På klientsiden i en webapplikasjon kan slike OS og SQL kommandoer være hardkodet eller generert gjennom grensesnittet på grunnlag av brukerens valg. "Cross Site Scripting" derimot er først et problem når utvikleren for eksempel ikke har forutsett at et felt for utfylling av fri tekst kan brukes på en skadelig måte hvis han eller hun ikke begrenser muligheter og rettigheter. Dette kan også gjelde de to andre komponentene. Dermed kan man si at alle de tre angrepskomponentene på sett og vis handler om en form for "injisering av kode" på ett eller annet tidspunkt.

Til slutt blir altså alle tre puttet i kategorien "Innputtmanipulering", som vi kan tenke oss inneholder langt mer enn bare de tre eksemplene. Det vil si flere av OWASPs eller

andre angrepskomponenter. Det vil si angrepskomponenter hvor både problem og løsning har fellestrekk på et abstrakt nivå med de som er trukket frem her.

La meg bruke ”Buffer overflow” som et eksempel på dette. Hvis man abstraherer dette angrepet tilstrekkelig, handler det i bunn og grunn om det samme: En form for Innputtmanipulering. Forskjellen ligger her i hvordan det løses, eller med andre ord hvilket preventivt brukstilfelle som må til for å forhindre akkurat denne typen angrep.

Uten å gå for mye i detalj kan jeg også trekke frem at det finnes to typer Innputtmanipulering:

1. Innputt som kommer direkte fra brukergrensesnittet, hvor denne manipuleres til å gå utover sin tiltenkte funksjon på grunn av manglende validering.

2. Bevisst manipulering direkte i kildekoden, for å få sendt ukorrekte eller uventede data til applikasjonen.

Denne siste typen angrep unngår mange former for tradisjonell innputt validering, fordi det ikke er innputt i tradisjonell forstand. I forbindelse med webapplikasjoner kommer man langt med et lokalt proxy og en teksteditor for å utføre begge typene Innputtmanipulering.

Et eksempel som kan illustrere dette er to måter å utføre et SQL-angrep på. Den første typen, via grensesnittet, går ut på at man for eksempel skriver inn et termineringstegn etterfulgt av en ny SQL-kommando i feltene for bruker-id og passord (Se AMM’et Innputtmanipulering på side 60, hvor dette er brukt som eksempel). Dette angrepet er det mer naturlig å forvente fordi det kommer fra brukergrensesnittet. Den andre typen angrep er mer sofistikert og derfor mer uventet. Det går ut på at man for eksempel stopper et http-kall ved hjelp av et proxy og manipulerer dette før det sendes videre. Her er det ikke snakk om manipulering av forventet innputt fra brukeren, via et tekstfelt eller andre grensesnittsobjekter, men av kildekoden direkte. Det ene kan anses som variabelmanipulering, det andre som kodemanipulering. Hvilket som er mest skadelig avhenger av crackerens mål, ikke av metoden som brukes. På et abstrakt konseptuelt nivå er det vanskelig å skille dem, fordi begge angrepene er av typen Innputtmanipulering og fordi begge angrepene avverges med de samme konseptuelle løsningene. Skillet mellom de to typene bør spesifiseres og analyseres på et senere tidspunkt.

Under følger tabellen for analyse og definisjon av de preventive brukstilfellene.

Angrepskomponent	Problembeskrivelse	Preventive brukstilfeller
Cross-Site Scripting	"1) HTML-aware: …" "… Create HTML TAG ALLOW lists. (Only allow certain safe HTML tags) Create HTML ATTRIBUTE ALLOW lists. (Only allow certain safe HTML Attributes) Filter any input Java-Script/Java/VBScript/ActiveX/Flash Related. Host HTML user input on an alternate domain. (acme.net instead of .com) Double and Single quotes should be filtered Remove all Null Characters" "2) HTML-Disable: …" "… Filter HTML enabling characters Double and Single quotes should be filtered Remove all Null Characters …" "3) HTML-Hybrid: …" "… Ensure all allowed tags and attributes are safe and limited. Double and Single quotes should be filtered Remove all Null Characters …"	I dette tilfellet vil man få en rekke preventive brukstilfeller avhengig av hvilket scenario man befinner seg innenfor. Det er med andre ord på sin plass med en abstraksjon for å få dette løsningskomplekset til å passe inn i et abstrakt misbruksmønster. Uansett hvilken løsning man velger å se nærmere på vil "**Valider, rens og tillat**" være beskrivende. Grunnen er at alle de spesifikke løsningene dreier seg om å sjekke innputt opp mot et tillatt sett med kodeelementer. Brukstilfellet "Valider, rens og tillat" er ment å implisere både en forsikring om at man kun tillater et begrenset antall validerte kode elementer, og at uønskede elementer renskes vekk.
Direct OS Command	" - Refrain from using system commands..." "... Validate and sanitize every user-variable passed to the web application..."	Brukstilfellet "**Valider, rens og tillat**" vil implisitt dekke både det å begrense bruk av systemkommandoer i det hele tatt, og det å begrense måten de blir brukt på.
Direct SQL Commands	"… Never pass unchecked user input to database queries. Validate and sanitize every user variable passed to the database. Check if given input has the expected data type. Quote user input which is passed to the database. …"	Som vi ser gjelder det samme for disse løsningene. All innputt må valideres, renses og eventuelt tillates hvis det skal være sikkert å slippe den gjennom. Mao. vil det abstrakte men preventive brukstilfellet "**Valider, rens og tillat**" også være dekkende her.

Tabell 5 – Fra angrepskomponenter til preventive brukstilfeller.

På et konseptuelt nivå er i dette tilfellet løsningene mer ensartede enn problemene. Både "Cross Site Scripting", "Direct OS Command" og "Direct SQL Command" kan sies å ha en abstrakt løsning som handler om tre felles hovedpunkter: Valider, rens og tillat.

Som det fremgår av tabellen, har jeg i mange tilfeller valgt å abstrahere problemer og løsninger til et så generelt nivå som mulig. Et hovedprinsipp for konseptuell modellering er at man ikke blir for spesifikk. Hvis man hadde fulgt klassifiseringen til OWASP,

ville det bli for spesifikt. Som jeg tok opp i teorikapittelet, så er det ingen god skikk å bli for teknisk allerede i kravarbeidet. Det fører til at man låser seg til bestemte løsninger alt for tidlig.

Formålet med å abstrahere OWASPs problem- og løsningskategorier er nettopp å fortette innholdet i en mer abstrakt modellkomponent. Selv om både problemer og løsninger virker forskjellige, er det mulig å abstrahere dem og dermed påpeke fellestrekkene.

4.2.4 DET ABSTRAKTE MISBRUKSMØNSTERET "INNPUTTMANIPULERING"

Det er viktig å merke seg at det i praksis mest sannsynlig vil være naturlig at sikkerhetseksperter utarbeider innholdet i de abstrakte misbruksmønstrene. I mønstersammenheng ellers gjøres dette på grunnlag av egne og andres erfaringer.

I sin mest abstrakte form har jeg altså kommet frem til følgende misbruks- og preventive brukstilfeller:

Misbrukstilfelle: "**Innputtmanipulering**"

Preventivt brukstilfelle: "**Valider, rens og tillat**"

Fortsatt aner man konturene av både problem og løsning, uten at de er for spesifikke eller teknisk ledende. Man kan si at de fanger opp hovedprinsippene både for dette bestemte sikkerhetsangrepet og for hva man bør gjøre for å unngå det.

Når det er sagt, så vil det være behov for å eksemplifisere dette paret av problem og løsning med et mer konkret scenario, slik jeg har vært inne på. I det endelige resultatet blir dette gjort ved å presentere et mindre abstrakt misbrukstilfelle enn det som er brukt som utgangspunkt for teksten. Det vil si at jeg velger å trekke frem "Manipuler SQL-kommando" for å konkretisere det abstrakte problemet noe. Allikevel er mønsteret presentert på et såpass abstrakt nivå at det sannsynligvis vil passe sikkerhetseksperter og utviklere dårlig, hvis de ser det isolert. Hvorvidt det kan oppfylle sin oppgave med å være kunnskapsbærer og kommunikasjonsmiddel, gjenstår å se. Kapittel 5 vil gi en indikasjon på dette.

I fremstillingen av det ferdige resultatet blir mønsteret delt over to sider. Den første siden presenterer den tekstlige delen, og den andre siden presenterer eksempel og beskrivelse av dette.

(Se neste side for del en av AMM'et Innputtmanipulering)

INNPUTTMANIPULERING

Navn:

Innputtmanipulering

Alias

Script Inadequacies; Input Validation Attacks

Motivasjon

Enhver applikasjon må ha en eller annen form for innputt for å kunne utføre en oppgave på grunnlag av brukerens ønsker og behov. Enhver form for innputt kan manipuleres. Den kan manipuleres innenfor de rammene utviklerens intensjon tilsier, eller alternativt på en måte som utvikleren aldri har tenkt på. Webapplikasjoner er komplekse informasjonssystemer som tilbyr mange måter å formatere innputt på; ASCII, Unicode, URL-encoding, og så videre.

Problem

Manglende kontroll av innputt i alle lag av applikasjonen.

Løsning

All innputt må valideres, sjekkes og renses på streng-, karakter- og byte-nivå for at kun lovlig innputt skal tillates.

Vær spesielt forsiktig med hva som sendes videre til f.eks. databaser, filsystemet og andre applikasjoner.

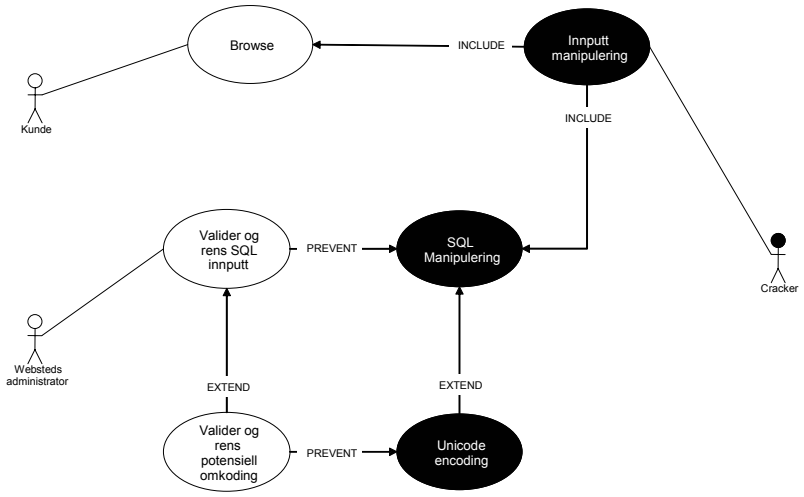
Eksempel

På neste side følger et eksempel som viser hvordan en sql-setning kan manipuleres til å oppnå uautorisert tilgang til en database.

Tabell 6 – Den tekstlige delen av Innputtmanipulering.

Her er det på sin plass å understreke viktigheten av punktet motivasjon. Mens problemet er spissformulert, skal motivasjonen gi en kort og konsis forklaring for hva som fører til problemet.

(Se neste side for del to av AMM'et Innputtmanipulering)



Forklaring

En bruker logger seg på en webapplikasjon med følgende data:

BRUKERNAVN: JohnDoe

PASSORD: Sunshine

Når brukeren trykker på OK, vil SQL variabelen se ut som følger:

SELECT * FROM users WHERE username='JohnDoe' AND password='Sunshine'

En cracker kan gjøre som følger:

BRUKERNAVN: '' OR username LIKE '%

PASSORD: ' OR password LIKE '%

Dermed følger:

SELECT * FROM users WHERE username='' OR username LIKE '%' AND password='' OR password LIKE '%'

Eventuelt kan crackeren gjøre manipuleringen noe vanskeligere å fange opp under validering. I stedet for å skrive prosenttegnet i ASCII, som kanskje ikke er tillatt, kan han velge en alternativ omkoding som f.eks. URL omkoding. Dermed skriver han "... username='' OR username LIKE '%20'..." som oversatt blir det samme som over.

Hvis ikke innputt valideres for lovlige verdier og renser vekk ulovlige verdier – sql konstruksjoner i dette tilfelle - vil manipulasjonen i eksempelet føre til at crackeren får en utskrift av hele brukertabellen med passord og det hele. Er da passordene lagret uten kryptering av noe slag, får crackeren tilgang til et verdifullt våpen.

(Eksempelet er fritt etter: www.silksoft.co.za og owasp.org)

Relaterte mønstre: Profilering eller Fingerprinting

Tabell 7 – Eksempeldelen av Innputtmanipulering.

4.2.5 DISKUSJON AV FORM PÅ LØSNINGSFORSLAGET

I det følgende vil jeg se nærmere på et par punkter som er spesielt interessante med hensyn til formen på det endelige resultatet.

4.2.5.1 Abstraksjon og generalisering

Et viktig element ved abstrakte misbruksmønstre er, som navnet tilsier, at de er abstrahert opp til et fortettet og lite teknisk nivå. Som jeg har vist er denne fortettingen basert på logisk begrepsorientert abstraksjon. Som eksemplene viser kan man bruke UMLs mulighet for generalisering til å tydeliggjøre dette. Fordelen med å benytte generalisering er at man får modeller som også viser sammenhengen mellom de abstrakte og de mer konkrete angrepene, som eksempelet i AMM'et Innputtmanipulering viser.

En generalisering i UML-notasjonen har mye til felles med angrepstrær. Et angrepstre kan brukes på mange måter. Man kan eksempelvis benytte denne notasjonen for å understreke komponeringen av abstraherte misbrukstilfeller. Da oppnår man også å forenkle analysearbeidet ved at man understreker sammenhenger mellom ulike angrep og løsninger. Dette er også til dels vist i AMM'et Innputtmanipulering. I eksempelmodellen har begge de to spesialiserte misbrukstilfellene, "SQL-manipulering" og "Unicode encoding", fått spesialiserte preventive brukstilfeller.

Hvis man benytter prinsippene fra angrepstrær i MC-notasjonen får man et verktøy som både er illustrerende og har et langt større bruksområde enn å kun fungere som en utvidelse av UC-notasjonen. Én fordel er altså at man kan bruke det samme verktøyet, MC-modellering, i flere faser og til flere oppgaver, noe jeg kommer nærmere inn på i kapittel 4.3.

En annen fordel er at man oppnår en bedre dybde i modellene ved å vise generaliseringshierarkiet i tillegg til for eksempel et abstrakt scenario. På den måten får man dokumentert hvordan man har tenkt når man har abstrahert. I tillegg får man en oppskrift på hvordan et AMM skal dekomponeres. Dette vil være til stor nytte i analyse- og design-fasen, hvor den jobben uansett måtte vært gjort for å komme frem til en teknisk og implementerbar løsning.

4.2.5.2 Problemorientert vs. løsningsorientert

Det er en viktig forskjell på tradisjonelle (design-) mønstre og abstrakte misbruksmønstre. Vanligvis er mønstre løsningsorienterte. Man kan argumentere med at AMM er det stikk motsatte, fordi de tar navn etter problemet og legger vekt på å fokusere på sikkerhetsangrepet og litt mindre på løsningen. Dermed kan man kalle AMM problemorientert, men som Schneier sier, så er det viktig å vite hva man skal beskytte seg mot (Schneier '00). Det er det som er meningen med AMM: Å formidle forståelse for sikkerhetsangrep, slik at man skal vet mer om hvilke sikkerhetsangrep man vil beskytte seg mot.

Én ulempe med en problemorientert tilnærming, er at den kunnskapen som formidles også kan misbrukes. Følger man den konseptuelle oppskriften i et AMM kan man lettere forstå og eventuelt gjennomføre det sikkerhetsangrepet som blir beskrevet. Dette er et velkjent problem i sikkerhetsmiljøet. Bugtraq og andre lignende fora for publisering av sikkerhetsproblemer blir ofte kritisert for å danne grobunn for det problemet de tilsynelatende vil til livs. Samtidig er det også velkjent at man bør vite hvordan en ”fiende” tenker for at man skal kunne utvikle et effektivt forsvar. Ved å forstå konseptene bak et angrep som blir beskrevet, kan man også lettere beskytte seg mot det.

4.2.6 EN KATALOG MED AMM

I sikkerhet generelt er en cracker målorientert og ikke prosessorientert. Derfor kan han benytte et utall forskjellige tilnærminger for å oppnå det han ønsker. Bruce Schneier, blant andre, trekker frem oppfinnsomhet som en viktig egenskap for en cracker. Dette sier noe om at en ikke bør utelukke noe misbrukstilfelle, verken under innsamling av krav eller senere i utviklingsprosessen.

For å unngå at enkeltmønstre brukes isolert bør de organiseres i en katalog. Å organisere relaterte mønstre i kataloger kan være til stor hjelp for utviklere og arkitekter, så vel som sluttbrukere, for å få oversikt over relaterte mønstre.

Et annet viktig aspekt ved kataloger med AMM’er er å skape en bibliotekfunksjon som både vil fungere som en form for organisasjonshukommelse og som et utvidet verktøy. Hvis man har en katalog med mønstre som er tilpasset det domenet man jobber innenfor, kan man effektivisere gjenbruksprosessen og dermed utviklingsprosessen, og man

vil få et bedre fokus på sikkerhet enn hvis man benytter AMM som et ad-hoc-verktøy. For at AMM skal fungere optimalt bør de spesifiseres på grunnlag av erfaring. Det vil gi seg uttrykk i at man får gode løsninger på velkjente problemer formulert i en standardisert form tilgjengelig også for mindre erfarne utviklere, kravarbeidere eller andre involvert i utviklingsprosessen.

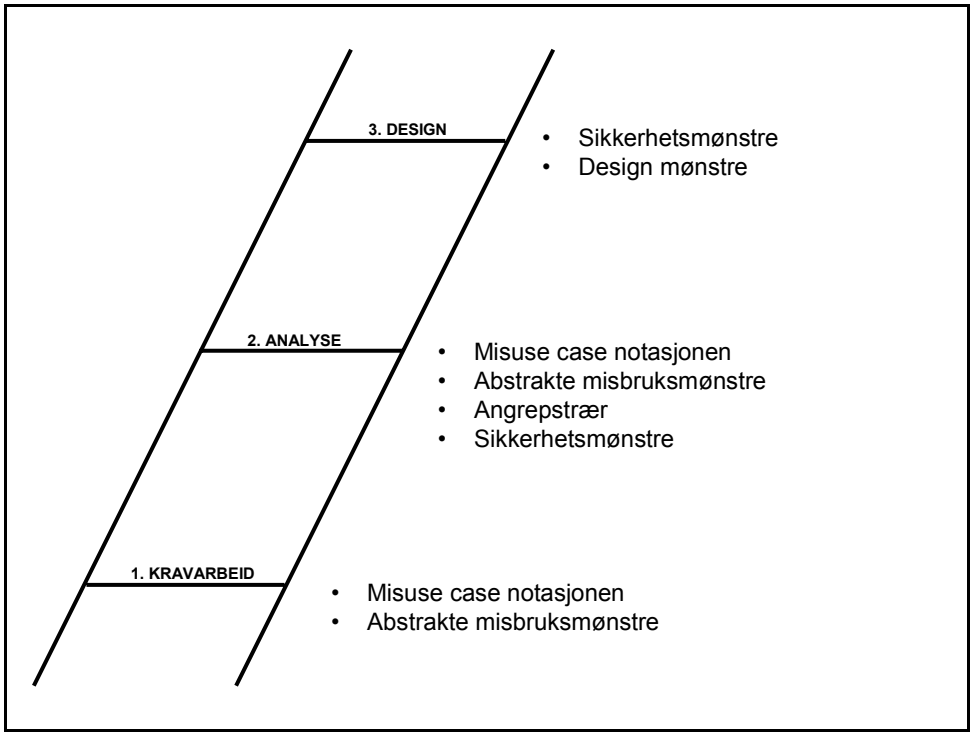
I 15408 finner man flere eksempler på katalogtilnærmingen. Der utgjør en PP et sett med generell sikkerhetskrav tilpasset én kategori programvare, for eksempel webapplikasjoner. Et slikt sett med generelle sikkerhetskrav er delt inn i klasser, familier og komponenter, ikke helt ulikt OWASPs ASAC, som kun deler inn sine komponenter i klasser. En katalog med AMM'er kan utformes på mange måter, men hensikten er at den skal fungere som et bibliotek av aktuelle domener innenfor et bestemt applikasjonsdomene, på samme måte som 15408. Spesielt interessant er det at 15408 også er ment å være en katalog som skal fungere som utgangspunkt for så vel sluttbrukere som sikkerhetseksperter. (ISO/IEC '99a; '99b; '99c)

4.3 AMM SOM DEL AV EN METODE

Abstrakte misbruksmønstre er ment som en konseptuell tilnærming til innsamling av sikkerhetskrav. Hvis denne teknikken skal få full effekt i et utviklingsprosjekt må også resten av prosessen fokusere på sikkerhet. Det vil si at de sikkerhetskravene som blir samlet inn også må bli tatt med videre i utviklingsprosessen og behandlet på en forsvarlig måte.

4.3.1 METODOLOGI

Ved å slå sammen og tilpasse noen teknikker for utredning av sikkerhetsproblemer, foreslår jeg her en ny metode for å fokusere på sikkerhet i både kravarbeid og analyse. Metoden er skissert i figuren under:



Figur 9 – Prosessmodell: Oppfølging av sikkerhetskrav fra kravarbeid til design.

Punkt én i figuren, kravarbeidet, skal illustrere at man kan ta utgangspunkt i teknikkene: AMM og MC-notasjonen. Dette innebærer at man tar utgangspunkt i kataloger med AMM og samler inn sikkerhetskrav på grunnlag av disse. Videre i kravarbeidet benytter man så MC-notasjonen for å spesifisere, analysere og utvikle MC-scenarier. I tillegg knytter man MC-scenariene opp mot resten av UC-modellen, ved å innlemme preventi-

ve brukstilfeller. På den måten får man innordnet sikkerhetskrav på en ryddig, sømløs og veldokumentert måte.

Punkt én og to faller i stor grad sammen med skjæringspunktet mellom kravarbeid og analyse. Her kan man for eksempel analysere UC-modellen og forsøke å finne ett eller flere AMM som er aktuelle. Hvis man finner et AMM som beskriver et angrep man er sårbar for, kan man spesifisere detaljerte MC-scenarier der man tar med de aktuelle truslene. For å utrede problemer vil angrepstrær være velegnede, for eksempel til å foreta en analyse for å spesifisere hvert enkelt misbrukstilfelle ut i fra et bestemt analysemål. Et slikt mål kan for eksempel være å komme frem til en balansegang mellom risiko og økonomiske aspekter.

Denne spesifiseringsanalysen kan gjøres ved at man anser hvert misbrukstilfelle som ett mål, som så brytes opp i mulige måter å nå dette målet på. Til dette er det naturlig å bruke notasjonen for generalisering, som er en del av UML[15]. På den måten kan man få en bedre oversikt over hva som behøves av tekniske løsninger, både med hensyn til design og til beslutninger – for eksempel beslutninger i forhold til hvilken type teknologi man bør ta i bruk for å avverge spesielt alvorlige trusler.

Man kan også tenke seg at man bruker et angrepstre for å vurdere økonomiske tap ved bestemte sikkerhetsangrep, satt opp mot valget mellom det å benytte kommersielle løsninger eller det å bruke gratis ”open source”-baserte løsninger. Det valget i seg selv kan også være et spørsmål om sikkerhet, og hvert av valgene har i så måte både fordeler og ulemper knyttet til seg som bør analyseres. I denne typen tilfeller kan en kombinasjon av notasjonene for angrepstrær og MC være et godt alternativ.

Hvis vi forutsetter at utviklingen følger Rational Unified Process – noe som ikke er usannsynlig ved bruk av UML - kan analysefasen se ut som følger: UC-modellen fra kravarbeidet konverteres til en analysemodell ved å utdype UC-scenarier. Når man så kommer til det punktet i prosessen hvor man begynner å ta avgjørelser på hvordan arkitekturen skal designes, vil det være naturlig å trekke inn sikkerhetsmønstre.

[15] Se figur 12 på side 78.

4.3.2 METODE-EKSEMPEL

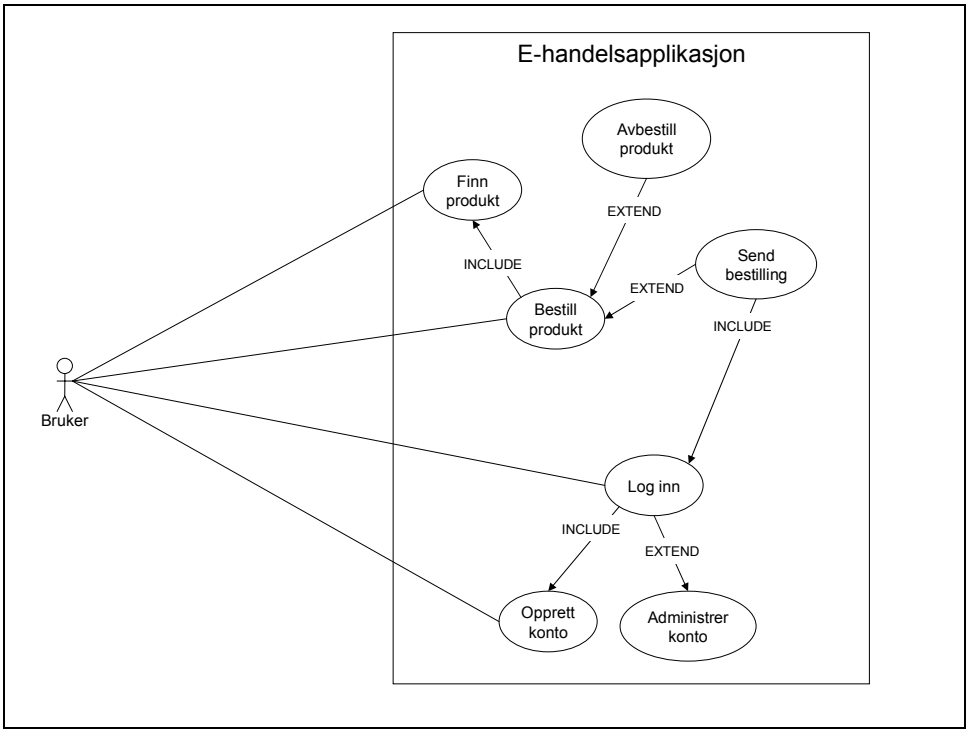
I denne delen vil jeg presentere et enkelt eksempel på hvordan metoden kan brukes i praksis. Hensikten er å vise hvordan man med utgangspunkt i en UC-modell kan ta tak i sikkerhet i kravarbeid, og deretter følge dette opp gjennom hele utviklingsprosessen.

4.3.2.1 Hvor skal man begynne?

Det kan utvilsomt være vanskelig å kaste seg ut i den komplekse prosessen med å formulere sikkerhetskrav. Derfor vil det være en fordel å begynne med å lage UC-modeller på vanlig måte før man går løs på sikkerhetsproblemene. Eksempelet som presenteres viser hvordan man kan benytte AMM, MC-notasjonen, MC-scenarier og angrepstrær for å oppnå en bred tilnærming til sikkerhetskrav i systemutviklingsprosessen.

4.3.2.2 Kontekstdiagram

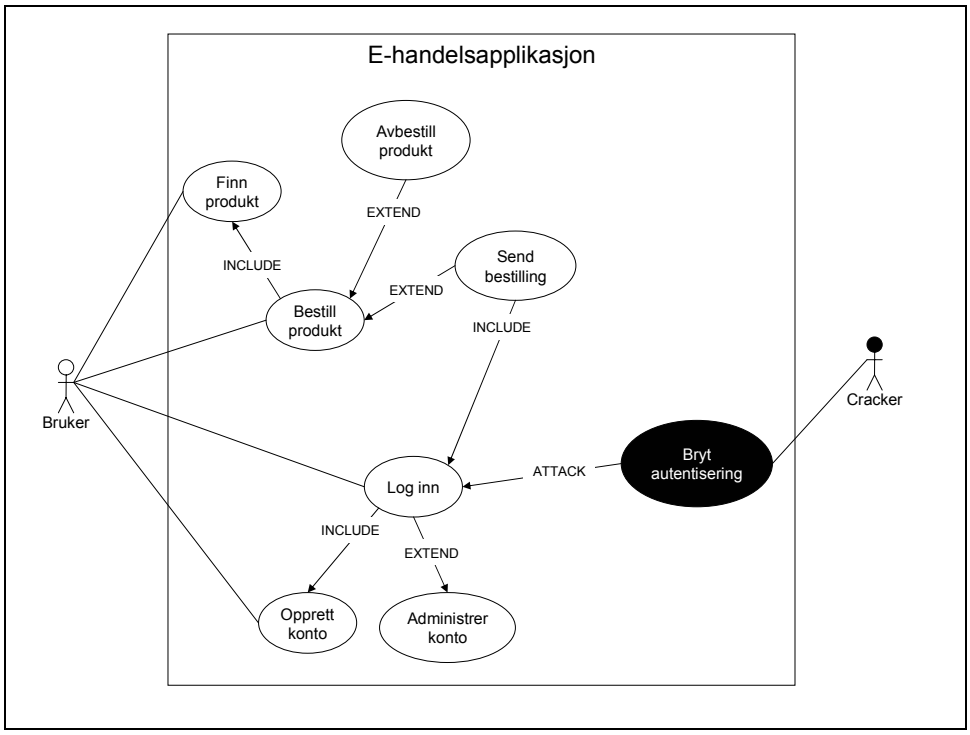
Eksempelet tar utgangspunkt i følgende kontekstmodell fra et tenkt e-handelssystem, supplert med eksempler på de forskjellige fasene i utviklingsprosessen: Kravarbeidet, analyse og design.



Figur 10 - Kontekstdiagram som utgangspunkt for eksempler.

4.3.2.3 Innsamling av sikkerhetskrav

La oss tenke oss at man begynner med å kikke på et mulig sikkerhetskrav som kan ha blitt formulert. En av domeneekspertene har sagt noe om at de ”må ha sikker log-inn”. Kravarbeidsteamet resonnerer seg frem til at sikkerhetskravet innebærer at man må beskytte seg mot at noen klarer å bryte autentiseringen. Dette integreres i UC-modellen fra figur 10, og vi får følgende modell:



Figur 11 – Sikkerhetskravet knyttes til UC-modellen.

I første omgang er det ikke noe AMM som kan knyttes direkte til dette kravet, og dermed må dette angrepet utredes nærmere, gjennom spesifisering og MC-scenario.

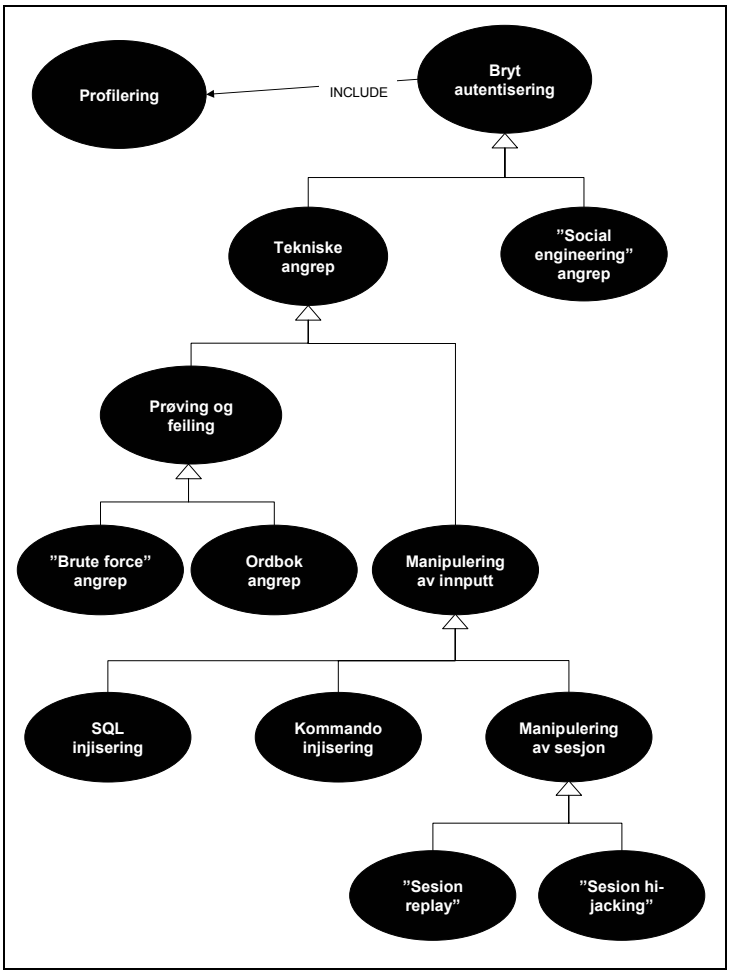
4.3.2.4 Angrepstrær til spesifisering

Bruce Schneiers angrepstrær er en enkel og effektiv måte å visualisere komponering og dekomponering på. Denne teknikken kan brukes i mange av fasene i et utviklingsprosjekt. Den egner seg for eksempel godt til å analysere og diskutere sikkerhet både på et konseptuelt nivå, og på et konkret og mer designrelatert nivå.

Teknikken er enkel og allsidig og kan dermed egne seg like godt til hjernemyldring som til en økonomisk analyse av risiko. Bruker man MC-notasjonen til å fremstille trærne, kan man også enkelt integrere dem med UC-modeller. På den måten får man tilført en

ny dimensjon til UC-modellen som langt på vei vil gi den informasjonen man trenger for å utvikle og implementere gode sikkerhetsløsninger.

Treet i figuren bygger videre på forrige underkapittel der kravet ”Må ha sikker log-inn” ble uttrykt i misbrukstilfellet ”Bryt autentisering” og relatert til UC-modellen. Her dekomponeres sikkerhetskravet slik at det blir lettere å spesifisere det:



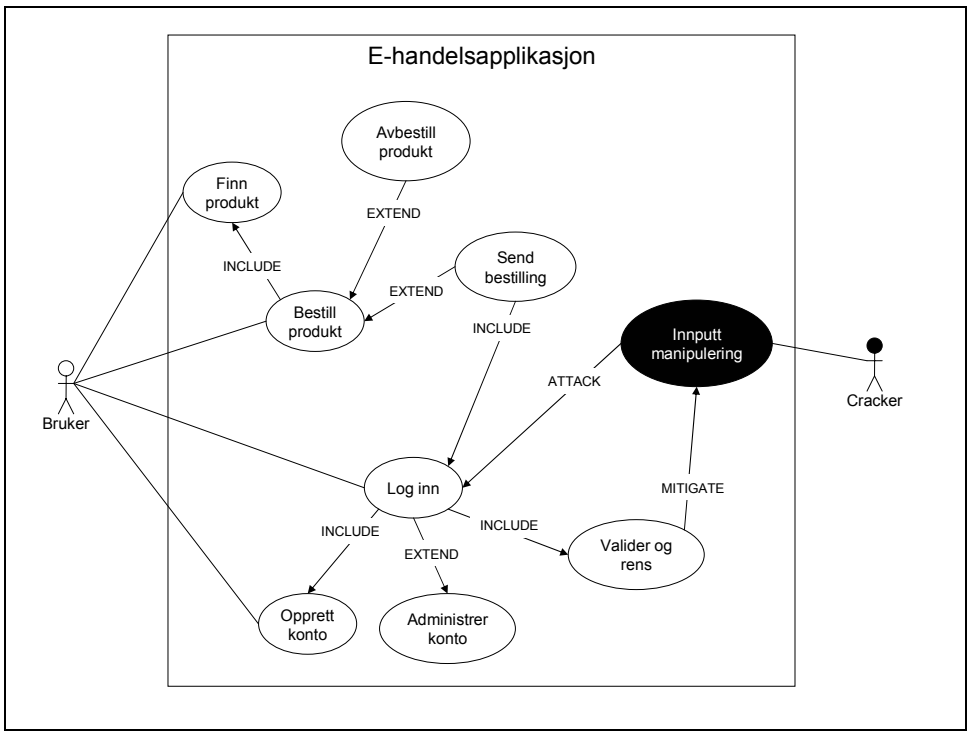
Figur 12 - Analyse av sikkerhetskrav med angrepstre.

Her ser man hvorledes kravarbeidsteamet kan benytte seg av et angrepstre for å spesifisere hva misbrukstilfellet ”bryt autentisering” innebærer. Deretter kan de vurdere om alle spesialiseringene er like relevante, for så for eksempel avgjøre hvilke av dem som vil volde mest skade for bedriften. Sannsynligvis har teamet en person som selv har vært med på å bygge opp bedriftens AMM-katalog, og som umiddelbart vil se at AMM’et ”Innputtmanipulering” vil kunne benyttes for å unngå forskjellige former for manipulering av innputt. Dermed har teamet raskt og enkelt funnet en gjenbrukbar

løsning på et av problemene knyttet til sikkerhetskravet ”Må ha sikker log-inn”. I tillegg vet én av medlemmene i teamet at AMM’et ”Profilering”[16] er standard å ta hensyn til når man utvikler en e-handelsapplikasjon, som i dette tilfellet.

4.3.2.5 Detaljering med MC-scenario

Når man har funnet ett eller flere AMM som kan brukes i sammenheng med dette utviklingstilfellet, integreres de i UC-modellen:



Figur 13 – AMM’et Innputtmanipulering blir integrert med UC-modellen.

Når den første delen av kravarbeidet er over og man har fått laget en spesifisert UC-modell med integrerte misbrukstilfeller, er det på tide å ta fatt på detaljene rundt sikkerhetsproblemene. Hvilke trusler innebærer de forskjellige scenariene? Til dette kan man benytte MC-notasjonen og dens scenariemaler til å spesifisere sikkerhetskravene. Igjen får man behov for mange av de opplysningene som kom frem under spesifiseringen av det overordnete misbrukstilfellet ”Bryt autentisering”, og lager eventuelt et nytt og mer spesifikt angrepstre hvis det er behov for det. Dermed får man et godt grunnlag for å

[16] For mer informasjon om AMM’et ”Profilering” se Vedlegg 3 – Ferdig eksempel på AMM (side 125).

utarbeide et detaljert MC-scenario hvor sikkerhetsproblemene blir relatert til akkurat dette spesifikke utviklingsprosjektet. Videre i dette eksempelet blir så misbrukstilfellet Innputtmanipulering spesifisert i et MC-scenario og analysert nærmere med det formål å skape et best mulig grunnlag for de påfølgende fasene design, implementering og testing. Ergo går gruppen i gang med å beskrive scenariet Innputtmanipulering.

Til bruk i denne delen av prosessen er det utviklet to forskjellige maler for spesifisering av MC-scenarier. Her velger jeg å bruke den enkle scenariemalen[17], "lightweight approach" (Engelsk terminologi er brukt i venstre kolonne fordi jeg ikke har funnet noen god oversettelse av alle begrepene):

(MC-scenario følger på neste side)

[17] Grunnen til at den enkle scenariemalen er brukt her er først og fremst plasshensyn, men også fordi den fungerer godt nok som illustrasjon i dette tilfellet.

Name	Logg inn
Iteration	n
Summary	Bruker logger seg inn med brukernavn og passord
Basic path of events	Steg 1: Systemet krever brukerkonto for å gå videre Steg 2: Bruker velger å logge inn Steg 3: Bruker fyller ut brukernavn og passord i de rette feltene Steg 4: Bruker trykker på ”logg inn” Steg 5: Systemet sjekker for Innputtmanipulering. Steg 6: Systemet validerer brukerdata mot database. Steg 7: Systemet godkjenner innlogging.
Alternative paths	Steg 2: Bruker har ikke konto. Systemet tilbyr bruker å registrere seg.
Exception points	Steg 3: Bruker fyller ut mangelfull informasjon. Steg 5: Systemet finner truende informasjon. Steg 6: Systemet finner ikke brukernavn og / eller passord i databasen. Steg 7: Systemet godkjenner ikke innlogging.
Extension points	“Administrer konto”
Triggers	“Send bestilling”
Assumptions	Bruker har konto.
Preconditions	Brukeren må ha en konto for å få godkjent innlogging.
Postconditions	Brukeren oppnår de rettigheter som innlogging medfører.
Threats	En cracker kan forsøke å lure innloggingsrutinene ved å manipulere innputt på flere forskjellige måter. • Injisere en fabrikkert SQL-streng • Ved å bryte koden i sesjons-id’en. • Injisering av kommandoer i operativsystemet.
Author	Gøran F. Breivik
Date	30.02.02

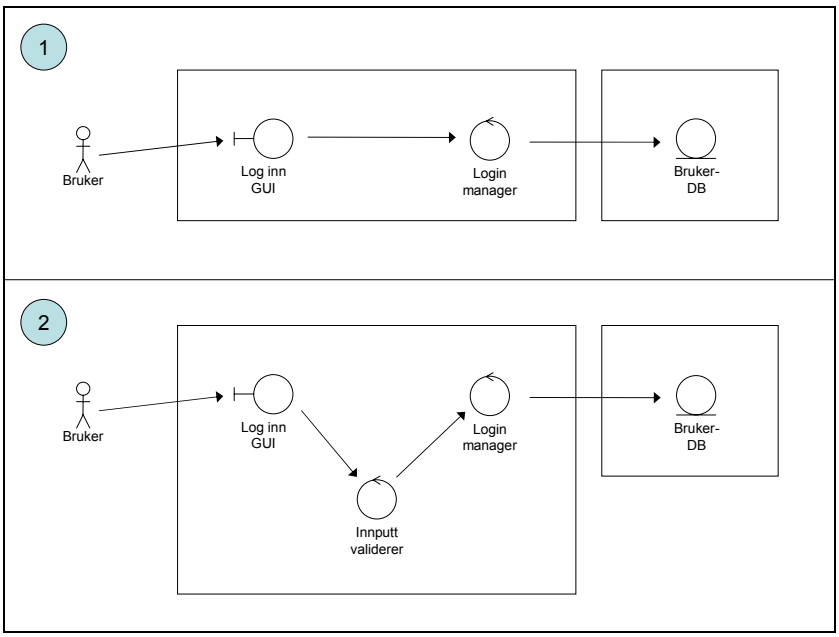
Tabell 8 – UC-scenario med trusler (Sindre et al. '02).

4.3.2.6 Innvirkning på design

Når arbeidet med analysen og overgangen til designfasen er i gang, blir det utarbeidet analysediagrammer[18]. I figuren under har jeg laget to forskjellige slike diagrammer. Det øverste diagrammet (1) er et analysediagram som viser hvordan systemdesignen kunne blitt seende ut hvis man ikke hadde brukt AMM’et Innputtmanipulering. I det neste diagrammet (2) er konsekvensen av AMM’et tatt med. Det viser en ekstra analyseklasse

[18] Denne diagramtypen brukes både i ICONIX og i RUP (Jacobson et al. '99; Rosenberg og Scott '01).

som tar seg av ”validering og rensing av innputt”, med andre ord den beskrevne løsningen i AMM’et Innputtmanipulering.



Figur 14 - Analysediagrammer med og uten resultatet av AMM’et Innputtmanipulering.

Ved å bruke denne metoden for behandling av sikkerhetskrav får en arkitektur som tar hensyn til sikkerhetskrav allerede fra begynnelsen av. I tillegg vil sikkerhet bli satt på dagsorden på et tidlig tidspunkt, noe som sannsynligvis også vil ha en innvirkning på de løsningene som blir utarbeidet, både med hensyn til arkitektur og med hensyn til implementeringsdetaljer.

4.4 SLUTTKOMMENTAR TIL DEN KONSEPTUELLE FASEN

I denne iterative og inkrementelle fasen har jeg forsøkt å vise hvordan utviklingen av AMM har foregått, fra de første skissene (vedlegg 1), via forskjellige eksempler på tidlige AMM (vedlegg 2) og til det ferdige resultatet (vedlegg 3). I og med at sikkerhetskrav, som andre krav, krever god oppfølging utviklingsprosessen gjennom, har jeg også utformet et forslag til en mer helhetlig metode. Til sammen tror jeg disse forslagene både kan være nyttige og danne grunnlag for bedre kommunikasjon, forståelse og enighet forbundet med sikkerhetskrav. Dermed anser jeg både det generelle og det spesielle forskningsspørsmålet i denne fasen for å være besvart.

5 EVALUERENDE FASE

Dette kapittelet gjør rede for evalueringen av ett abstrakt misbruksmønster. Denne evalueringen har til hensikt å gi et inntrykk av hvordan ideen og konseptet AMM blir oppfattet av kravarbeidere. Inspirasjon til nye ideer og tanker om videreutvikling er det viktigste formålet med denne undersøkelsen. Jeg starter med å ta for meg hvordan intervjuene er bygget opp og gjennomført. Deretter gjør jeg rede for og analyserer funn, før jeg avslutter med noen kritiske kommentarer til mitt eget evalueringsopplegg.

5.1 METODE OG PROBLEMSTILLING

Metodisk er denne fasen bygget opp slik det er beskrevet i metodekapittelet. Med utgangspunkt i oppgavens overordnete problemformulering, har jeg spesifisert to forskningsspørsmål, et generelt og et spesielt. På grunnlag av disse skal jeg generelt sett forsøke å belyse om AMM kan være nyttig i behandling av sikkerhetskrav, og spesielt om AMM kan formidle kunnskap om sikkerhetsangrep, gjøre slike lettere å forstå og om dette i så fall kan føre til et bedre grunnlag for enighet innad i kravarbeidsteamet.

5.2 OPPBYGNING

Intervjuene ble bygget opp rundt en sammenligning av to ulike tilnærminger til sikkerhetsangrep[19]:

- Et selvutviklet AMM
- En av OWASPs angrepskomponenter, oversatt til norsk

Ved å oversette OWASPs angrepskomponent hadde jeg altså to ulike tilnærminger til rådighet til fremstilling av sikkerhetsangrep. Begge tilnærmingene formidler i bunn og grunn det samme, men på to forskjellige abstraksjonsnivåer.

Som grunnlag for samtalen i intervjuene ble det utformet en intervjuguide[20]. I denne guiden er det formulert spørsmål og kontrollspørsmål som skal belyse problemstilling-

[19] Begge forslagene er å finne i sin helhet i vedlegg 5.

en. I begynnelsen av intervjuet stiller jeg noen enkle ja/nei spørsmål. Dette gir meg litt bakgrunnsinformasjon om respondenten.

5.2.1 RESPONDENTER

Seks respondenter ble intervjuet. I utgangspunktet var målet å få intervjuet to eller tre innenfor hver av de nevnte kategoriene domeneeksperter, utviklere og sikkerhetseksperter. Dette målet viste seg å være vanskelig å oppnå. Det var spesielt vanskelig å få fatt i domeneeksperter. De tidsrammene jeg har arbeidet innenfor gjorde at utvalget ble noe mindre enn det som først var hensikten. Det endte med at jeg intervjuet én domeneekspert, to sikkerhetseksperter og tre utviklere.

I resten av kapittelet vil jeg referere til den enkelte respondent med den koden jeg gir dem i tabellen under, for eksempel U1 for utvikler 1.

Respondent:	Kategori:	Beskrivelse:
U1	Utvikler	Mann 31 år, 8 års erfaring som utvikler. Jobber i det private næringsliv som seniorutvikler.
U2	Utvikler	Mann 52 år, 25 års erfaring som utvikler, jobber som sjefskonsulent i det private næringsliv.
U3	Utvikler	Mann 39 år, seniorutvikler i det private næringsliv.
S1	Sikkerhetsekspert	Mann 34 år, 17 års erfaring som programmerer. Jobber nå som sikkerhetsekspert i det private næringsliv.
S2	Sikkerhetsekspert	Mann 42 år, 18 års erfaring som systemansvarlig. Lang erfaring med programvaresikkerhet.
D1	Domeneekspert	Kvinne 47 år, ingen IT bakgrunn, men har vært involvert i kravarbeid som domeneekspert.

Tabell 9 - Oversikt over respondenter.

[20] Intervjuguiden er å finne bakerst i oppgaven som vedlegg 4.

5.2.2 INTERVJUMATERIALE

Det materialet jeg bruker som stimuli i intervjuet består som nevnt av to forslag som begge er ment å støtte kommunikasjon, forståelse og enighet i et kravarbeidsteam. Ved å bruke to tilnærminger med samme angrep som utgangspunkt, ble de to fremstillingene veldig like, men allikevel med så vesentlige forskjeller at dette i seg selv burde gi både positive og negative reaksjoner hos respondentene. To vesentlige forskjeller kan nevnes. Det er en vesentlig forskjell i abstraksjonsgrad - forslag 1 er for eksempel mer teknisk detaljert enn forslag 2. En annen vesentlig forskjell er et konsekvens av at forslag 2 er bygget opp som et mønster, mens forslag 1 ikke har dette som noe formelt mål i seg selv. Andre sider ved presentasjonsformen utgjør også viktige forskjeller. Forslag 2 har et forenklet MC-scenario med en kort beskrivelse som illustrasjon og eksempel, mens forslag 1 kun har kodeeksempler.

5.3 FUNN OG ANALYSE AV DATA

Jeg bygger min tolkning på to kilder: Transkriberte sitater (vedlegg 6), og egne inntrykk fra intervjuene. De transkriberte sitatene er plukket ut på grunnlag av relevans i forhold til problemstillingen, og på grunnlag av interessante problemer som dukker opp underveis i analysen.

Det viste seg at det spesielle forskningsspørsmålet ga få interessante funn. Derfor er analysen basert på det generelle forskningsspørsmålet, som gir en bredere tilnærming til funnene. Det vil si at jeg har gått gjennom sitatene med det mål for øyet å finne interessante problemer som har relevans for AMM, og som kan gi svar på om AMM kan være nyttig i behandling av sikkerhetskrav. Denne brede tilnærmingen, kombinert med oppgavens kvalitative utgangspunkt, fører til at tolkningen som følger er gjort på grunnlag av min egen subjektive innfallsvinkel.

Sitatene har jeg tolket i et videre perspektiv enn det problemstillingen legger opp til. Dette har jeg gjort fordi jeg etter hvert oppdaget at problemstillingen ble for teoretisk til at respondentene var i stand til å si så mye om den. Sitatene ble derfor kodet på grunnlag av interessante problemer respondentene tok opp:

- Flere hadde meninger om hvordan og hvilken **brukssituasjon** AMM ville passe best i.

- Flere var også opptatt av hvilken **rolle** som har og burde ha **ansvaret** for sikkerhet i en systemutviklingskontekst.
- Det ble også ment noe om hvordan AMM bør **presenteres**.
- Til slutt ser jeg nærmere på uttrykk for **holdninger** som kan ha en innvirkning på hvordan AMM vil bli mottatt.

5.3.1 BRUKSMÅTE OG BRUKSSITUASJON

I dette underkapittelet ser jeg nærmere på hvilke brukssituasjoner respondentene mener AMM kan egne seg i. Kan det for eksempel være en ide å bruke AMM som medieringsmiddel mellom domeneeksperter og andre medlemmer i kravarbeidsteamet? Jeg vil vise at respondentene, på tross av at de har liten tro på at det i det hele tatt vil være på sin plass å diskutere sikkerhet med domeneeksperter, har mange forslag til hvor og av hvem AMM bør kunne brukes.

5.3.1.1 AMM i analysefasen

I det følgende sitatet kommenterer respondent U1 de to forslagene, 1 (OWASP) og 2 (AMM):

> **"For meg så er forslag 1 rett på sak og for en utvikler så er det dét som trengs. Men skal en diskutere dette i et kravspesifikasjonsforum. En samling av folk som da er kjøpere av applikasjonen, folk som kommer inn som kvalitetssikrere og som leverandør, så tror jeg at forslag 2 er det som definitivt kan - kan favne bedre." (Sitat 1.4)**

Respondent U1 er i sitatet inne på AMM kan egne seg til diskusjon i kravarbeid, men poengterer tydelig at han selv foretrekker den mer tekniske tilnærmingen i forslag 1. Ved en annen anledning snakker han om hvor i prosessen sikkerhet diskutert:

> **"I forbindelse med sikkerhetsanalyse så må en gjøre seg en del "tanker" sammen med forretningssiden, som da er kunden for å si det sånn." (Sitat 1.1)**

> **"I kravspesifikasjonen vår, så kommer jo det fra forprosjektet da som har tenkt noe på dette her. Den vil jo for så vidt beskrive korrekt bruk av applikasjonen. Så i forbindelse med sikkerhetsanalysen så må en ta**

stilling til en del, hva skal jeg si… Misbruk enten ved at en får uautorisert tilgang, eller hva skal jeg si, ikke tiltenkt bruk av programmet. Om det er mulig og en del sånne ting." (Sitat 1.3)

I disse sitatene forteller respondenten at man må "gjøre seg en del tanker sammen med forretningssiden". Han betegner denne aktiviteten som sikkerhetsanalysen. Respondenten omtaler jobben med å utarbeide en kravspesifikasjon, som forteller hva en applikasjon skal gjøre og ikke skal gjøre, som forprosjektet. Den fasen han omtaler som sikkerhetsanalysen er en frittstående analyse som foregår i tilknytning til kravarbeidet.

Sikkerhetsanalysen i U1s bedrift foregår som en frittstående del av kravarbeidet[21] og utføres av folk med helt annen bakgrunn enn systemutvikling. Det fremkommer av neste sitat at det blant annet er revisorer som utfører en slik sikkerhetsanalyse i U1s bedrift:

"… du har jo medarbeidere i et sikkerhetsteam som egentlig har til oppgave å prøve å være kritiske og stille kritiske spørsmål, ikke sant? De har nødvendigvis ikke så god teknisk IT kompetanse, men de er veldig opptatt av det der at du, som sagt, følger en best practice. Og da tror jeg at forslag nr. 2 kan være veldig greit å bruke. Med tanke på den erfaringen jeg har med å sitte og diskutere sikkerhetsproblemstillinger med revisorer." (Sitat 1.4)

Hvis man forutsetter at revisorer vanligvis har liten eller ingen teknisk IT-bakgrunn, kan denne aktiviteten anses som en vel så interessant brukssituasjon som en ren kravarbeidsfase med domeneeksperter. Dermed vil de samme problemene som Pohl trekker frem i forbindelse med kravarbeid, også gjøre seg gjeldende i denne spesifiseringen av sikkerhetskrav. Slik sett kan AMM spille en vel så viktig rolle i denne brukssituasjonen, som i en mer ordinær kravarbeidsfase. Det er kommunikasjon med lite teknisk skolert personell AMM er beregnet for, uavhengig av faser.

[21] Dette ble bekreftet 20.11.02 på telefon med respondenten. Da ble det også bekreftet at revisorene han snakker om har økonomisk og ikke IT-teknisk bakgrunn, men han poengterer at folk med teknisk bakgrunn også er med på denne analysen.

Respondent U3 antyder i det følgende sitatet en annen bruksmåte for AMM enn den respondent U1 antyder:

> **"Vi har revisjoner, i gåseøyne, av løsninger. Der man går i gjennom fra a til å. Men det skjer jo ikke veldig ofte og vi er for så vidt i en prosess nå der vi har kjørt liksom i gang aktiviteter med å analysere den nye løsningen vår." (Sitat 1.12)**

Det kan virke som om respondent U3 har en annen oppfatning av behandlingen av sikkerhetsanalyse enn respondent U1, eller at hans avdeling følger andre rutiner forbundet med dette. Denne tolkningen er avhengig av hvor man plasserer aktivitetene sikkerhetsanalyse og revisjon i utviklingsprosessen. Vanligvis tilhører disse aktivitetene to forskjellige faser. Revisjon knyttes gjerne til testfasen, altså helt mot slutten av utviklingen, mens analysen som regel foregår i kombinasjon med kravarbeidsfasen.

Videre har denne tolkningen en interessant konnotasjon. Respondentene antyder her at det er folk med forståelse for sikkerhet, men som ikke nødvendigvis har teknisk IT kompetanse, som kan ha bruk for AMM. Både respondent U1 og respondent U3 antyder at folk som allerede driver med sikkerhet vil ha nytte av AMM.

Hva kan det skyldes at disse to respondentene mener at noen andre enn utviklerne selv bør ta i bruk AMM? Dette kan skyldes rolleinndeling i bedriften, eller kanskje ansvarsfordeling, noe jeg ser nærmere på i avsnitt 5.3.2. En annen grunn kan være manglende respekt for sikkerhet, noe som støtter Schneier og andre sikkerhetseksperters utsagn om lite bevisste holdninger til sikkerhet. Som nevnt i teorikapittelet er sikkerhet et tema som det undervises lite i ved universiteter og høyskoler. Dermed er også det generelle kunnskapsnivået om sikkerhet lavt blant mange utviklere. Dette fører til manglende respekt for emnets betydning i systemutviklingsprosessen, og kanskje også til at ingen føler at det direkte er deres ansvarsområde.

5.3.1.2 Kravarbeid eller design?

Selv AMM oppfattes av noen av respondentene som for teknisk og dermed bedre egnet for designfasen. Det er flere årsaker til dette, for eksempel at det ikke er tid eller plass til en såpass grundig gjennomgang av sikkerhetskrav med domeneekspertene, slik som respondent U1 gir uttrykk for i sitatet under:

> **"Det er ikke utypisk at du sitter med en relativt senior person fra kjøpersiden, som er da en domeneekspert innenfor det området du lager applikasjonen ikke sant. Og det er en ressursperson som skal utnyttes av forretningssiden eller av kunden samtidig. Og det er rift om den personen sin tid. Jeg har prøvd å bruke min tid. Når jeg har vært med i kravspesifikasjon, at jeg har brukt min tid til å beskrive. Jeg har drømt opp alt jeg komme på av ubehageligheter og så har jeg skrevet de ned, og på noen har jeg bare sagt at; Ok, det er min jobb, det fikser jeg. Jeg bare tenker på det når jeg skal kode. Og andre. Dette kommer til å koste penger ikke sant. Den vil forsinke prosjektet også videre ikke sant. Så du får en sånn for og i mot avveining her, men det har ikke vært noe struktur over det. Et sett med e-mail, som jeg samler i et arkiv da egentlig. Så jeg sender ut noen tanker med en e-post og et par powerpoint slides og så få en tilbakemelding fra (domeneekspert)." (Sitat 1.5)**

Her kommer det tydelig frem at respondent U1 har drevet med kravarbeid, men man får ikke inntrykk av at det har foregått særlig strukturert. Han sier selv "Så du får en sånn for og i mot avveining her, men det har ikke vært noe struktur over det." Kravarbeidet som beskrives ligner mer på en forundersøkelse som er lite strukturert og som samtidig virker lite grundig. Jeg tolker dette som om U1 indirekte sier at kravarbeid i praksis vanskelig kan gjøres så grundig som bruk av AMM impliserer. Respondent U2 understøtter dette i flere utsagn som antyder at både AMM og OWASPs angrepskomponenter passer bedre inn i en designkontekst enn en kravarbeidskontekst:

> **"For når jeg bare så på de skrivene her så. Oi, det her er nyttig i en design, i en realiseringsfase. Og også kanskje som et underlag for et kravarbeid. Sette krav til det her (forslagene). Du beskriver kravene på**

et litt mer generelt plan når du har med kunden, men du implementerer det eller realiserer det med hjelp av noe sånt." (Sitat 1.8)

Ved å bruke AMM som utgangspunkt for å utarbeide krav, mener respondenten teknikken kan være fruktbar, nærmest som et utgangspunkt for ideer.

5.3.1.3 Prosessdokumentasjon og legitimering

Det kan også være andre sider ved sikkerhet enn kvaliteten i seg selv. For eksempel kan det være vel så viktig å gi et inntrykk av at man har gode rutiner for å behandle sikkerhet, og for å legitimere ens eget seriøse forhold til sikkerhetsspørsmål som utvikler.

I det følgende sitatet snakker respondent U2 om denne typen situasjon. Han mener at AMM kunne vært brukt som en form for dokumentasjon for å understøtte påstanden om at "vi lager systemer som er innbruddssikre" (1.6). Dette tyder på at det også kan være viktig å underbygge en fasade som garanterer sikkerhet. Respondenten mener at AMM kan fungere godt som en form for legitimering av solide sikkerhetsrutiner overfor kunden, og vise at bedriften tar sikkerhet alvorlig:

"Om du kunne få en patternbeskrivelse. Noe à la det her (AMM). Det hadde vært noe det. For da kan du si at sånn (viser frem en fiktiv bunke med patterns) er det vi designer. Det er standard, så det gjør vi. Det savner jeg." (Sitat 1.7)

Videre sier han som følger om hvorvidt AMM vil egne seg som formidlingsmedium for sikkerhetskrav overfor domeneeksperter:

"Ikke i en kravspekk med kunden, men i et internt team som skal realisere kravet. Men da tror jeg det at kravene går på det at det skal være innbruddssikkert. Ekstremt høy sikkerhet på login, og så har du løsningen der." (Sitat 1.11)

Det virker som om respondent U2 er mer usikker på hvordan AMM kan brukes, men han ser tydeligvis mye potensial med hensyn til ulike brukssituasjoner. Det han er klokkeklar på, er at det ikke vil ha noe for seg å forsøke å få med domeneeksperter på å utarbeide sikkerhetskrav, uansett hvilken teknikk man skulle velge å bruke.

5.3.2 ROLLER OG ANSVAR

Hvorvidt respondentene i denne evalueringen har forstått hensikten med AMM, skal være usagt. Det kommer tydelig frem at de som forstår problemet med ”Innputtmanipulering”, med andre ord de IT-teknisk skolerte, ikke er villig til å påta seg jobben med å definere og arbeide med slike krav. I det følgende vil jeg forsøke å vise at avgrensede roller er en faktor i kravarbeidet, og at ansvar, i alle fall antydningsvis, skyves over på en annen rolle enn den man selv innehar.

Respondent U1 beskriver i følgende sitat i hvilken sammenheng sikkerhet tas opp i hans bedrift, og påpeker samtidig at det ikke er hans ansvar:

> **”I kravspesifikasjonen vår, så kommer jo det fra forprosjektet da som har tenkt noe på dette her. Den vil jo for så vidt beskrive korrekt bruk av applikasjonen. Så i forbindelse med sikkerhetsanalysen så må en ta stilling til en del, hva skal jeg si… Misbruk enten ved at en får uautorisert tilgang, eller hva skal jeg si, ikke tiltenkt bruk av programmet. Om det er mulig og en del sånne ting.” ”… men jeg er ikke den som setter opp kravet.”(Sitat 1.4)**

Her avslutter respondenten med å påpeke at det ikke er hans oppgave å sette opp sikkerhetskrav. Likevel sier han i de to neste sitatene at det er hans jobb som programmerer å sørge for at det ikke er mulig å foreta Innputtmanipulering som kan føre til sikkerhetsbrudd:

> **”Det er min jobb som programmerer at det ikke er mulig å putte inn ugyldige data. Så det må spesifiseres, hva skal jeg si, hva som er ugyldige data, eller ugyldige kombinasjoner av data. Hvis du tenker rent sånn skjermbildeorientert programmering da. Det er jo typisk på webforms og – så er det jo på den måten du egentlig tenker.” (Sitat 2.2)**

> **”Det er min oppgave som programmerer å – Du er nødt til å spesifisere faktisk helt ned på, hvis du skal gjøre det skikkelig så må du spesifisere helt ned på enkeltkarakterer. Hvilke tegn som skal være gyldig … Men det kommer som et motkrav opp i fra min side egentlig hvilke tegn som du ikke kan akseptere som innputt.” (Sitat 2.3)**

Disse utsagnene kan tyde på at respondenten ikke helt har forstått at meningen med det AMM'et han har fått utdelt er å sette fokus på nettopp den typen funksjonelle krav som vanligvis vil være hans ansvar. Det er tydelig at respondenten her mener at en gjennomsnittlig kunde ikke vil ha noen forutsetninger for å forstå konseptene som tas opp i AMM.

> **"Men når det gjelder innbruddssikkerhets mot hacking da. Altså den type sikkerhet da, det er altså noe som sjelden står i spekken. Det er ofte noe som overlates til den enkelte installasjons- og operative avdeling. Du har kryptering da som det eventuelt kreves. Det er også noe du bare slår av og på nesten. I en moderne server i dag." (Sitat 2.8)**

Her sier respondent U2 praktisk talt rett ut at det er helt opp til den enkelte utvikler, eller i beste fall utviklingsteamet, hvorvidt man tar hensyn til applikasjonssikkerhet eller ikke. Han sier med andre ord at det er en del av utviklerens rolle å ta hensyn til slikt under realisering av et system. Dette er et godt eksempel på at det kan være mange fordeler knyttet til det å bruke AMM under kravarbeidet. For det første får man dokumentert hvilke avgjørelser som er tatt, og til en viss grad hvordan man har tenkt. For det andre unngår man å gå i de samme fellene gang på gang, i større grad enn om man ikke har formalisert vanlige problemer gjennom mønstre.

Når man vil problemer som "Innputtmanipulering" til livs, er det viktig å ha en person med den rette forståelsen for problemet med i teamet, uavhengig om det er i kravarbeidsfasen eller i en annen fase. På den måten kan AMM fungere som en mediator når man forsøker formidle denne typen krav til en person uten den samme tekniske kompetansen som en programmerer.

Respondent U1 oppsummerer kort hvilke roller han mener AMM egner seg for og ikke:

> **"Måten den er skrevet på gjør det mulig for en, hva skal jeg si? Nå snakket vi om domeneekspert, men du har jo ulike domener som er involvert, men for kundesiden å forstå hvilke problemer som ligger her… Men også, hva skal jeg si, du har jo medarbeidere i et sikkerhetsteam som egentlig har til oppgave å prøve å være kritiske og stille kritiske spørsmål, ikke sant? De har nødvendigvis ikke så god teknisk IT kom-**

> **petanse, men de er veldig opptatt av det der at du, som sagt, følger en best practice. Og da tror jeg at forslag nr. 2 kan være veldig greit å bruke." (Sitat 2.4)**

Slik jeg tolker det er respondenten her vant til å ta seg av de funksjonelle sikkerhetskravene uoppfordret. Dette setter et av problemene med denne typen krav på dagsorden. For at kravene skal bli behandlet på forsvarlig vis bør man gjøre dette mer formelt enn det respondent U1 viser til. Hvis man overlater denne typen sikkerhetskrav til hver enkelt utvikler, vil det være personavhengig hvorvidt en applikasjon holder et høyt sikkerhetsnivå.

5.3.2.1 Perspektiver

Respondent U1 kommer med en meget interessant betraktning om presentasjonen av AMM overfor ulike roller:

> **"Kanskje det jeg sier er at du må ha flere versjoner. Ja, det er vel egentlig det jeg sier." (Sitat 3.3)**

Det respondenten faktisk mener med "flere versjoner" er at man må ha én AMM-versjon for hver enkelt rolle i kravarbeidet, tilpasset den rollens tekniske kompetanse. Det er en vanlig regel innenfor en hver form for tekstlig arbeid at man tilpasser teksten til mottakeren.

For å nå flere kategorier kravarbeidere kan det være en idé å utarbeide de samme AMM'ene på flere abstraksjonsnivåer, sett fra flere perspektiver, men med samme grunnleggende kjerne. På den måten kan man oppnå en kommunikasjon om problemet med grunnlag i hver enkelt respondents perspektiv på systemet.

I forbindelse med systemering er det vanlig å dekomponere komplekse problemer. Man begynner med å beskrive problemet abstrahert, og så jobber man seg nedover i problemet til det er tilstrekkelig spesifisert. Ved å følge denne prosedyren for AMM får man et sett med mønsterdokumenter som har ulik abstraksjonsgrad og dermed ulik innfallsvinkel. Man oppnår å få både et veldokumentert mønster og et godt formidlingsverktøy i flere faser av utviklingsprosessen. Sannsynligvis vil det også skape et godt utgangs-

punkt for kommunikasjon, forståelse og enighet blant en ofte broket gruppe med krav-arbeidere.

Hvis en utvikler får utdelt et mer teknisk og presist AMM, vil det kanskje føre til at han eller hun forstår problemet bedre, og at vedkommende dermed vil være bedre i stand til å forklare det til andre. Hvis den samme utvikleren kjenner innholdet av AMM'et beregnet for domeneeksperter, vil han kanskje også lettere kunne utdype det innholdet på grunnlag av "sitt eget" AMM.

5.3.3 PRESENTASJON

I denne delen vil jeg se nærmere på de av respondentenes utsagn som går direkte på presentasjon og form av AMM. Jeg tar tak i problemet med abstraksjon fordi dette står helt sentralt i konseptet. I tillegg har jeg sett litt på problemer med hensyn til form og terminologi.

Formen noe presenteres i kan være avgjørende både for forståelse og kommunikasjon. Det er også noe av bakgrunnen for å bruke en mønstermal i første omgang. Uten den ville det fort blitt mange uoversiktlige fremstillinger av allerede komplekse problemer.

Om formen på AMM sier Respondent U2:

> **"Jeg ser på dette her mer som et slags sikkerhetspattern nesten jeg (om forslag 2 – AMM). At det her er måter det kan gjøres på og som man må ta hensyn til i design. Og sånn sett så er dette her et av flere krav. Du har identifisert at dette her går an å gjøre og det må vi gjøre noe med." (Sitat 3.4)**

Her er respondenten selv inne på at AMM er formet som et mønster. Og at det må være et av flere krav er også helt korrekt i forhold til hvordan det var tenkt, men så kommer han i neste omgang til et springende punkt:

> **"Men jeg er litt usikker på hva du mener skjønner du. For du kommer med en løsning. En spekk er et krav, dette her er egentlig løsninger på krav." (Sitat 3.5)**

Han mener formen er feil i forhold til hvordan et krav bør være utformet. Siden AMM'et innholder en løsningsbeskrivelsen mener han tydeligvis at det ikke kvalifiserer som en kravbeskrivelse.

Kanskje er det den tilsynelatende fastlåste formen respondenten reagerer på. Det er mulig denne kan virke hemmende i og med at løsningen allerede er definert, men dette er som nevnt i teorikapittelet nettopp et av hovedhensiktene med mønsterkonseptet - å gjenbruke anerkjente løsninger på velkjente problemer. Det er ikke sikkert det er den løsningen man ønsker å bruke, men det er den mest anerkjente løsningen på det aktuelle problemet.

5.3.3.1 Abstraksjonsnivå

Det er i denne oppgaven lagt til grunn en oppfatning om at abstraksjon kan ha en positiv effekt på kommunikasjon. En forutsetning for dette har vært at sikkerhetsproblemene som har stått i fokus har vært for kompliserte til å forstå for personer med liten eller ingen IT-kompetanse. Etter å ha gjennomført pilotstudiet ble det klarere at formidling av applikasjonssikkerhet byr på store utfordringer, og at abstraksjon alene ikke vil være tilstrekkelig for å møte disse utfordringene.

Som en kontrast til det forrige sitatet er respondent U1 også i det følgende inne på noen interessante tanker om form:

> **"Jeg ville vel kanskje ha kjørt litt mer kjøtt på beina på nr. 2 for å få med meg utvikleren også, men jeg tror nr. 2 er det beste. For en annen grunn til det er at en må ikke bli for spesifikk her for da er jeg redd for at, i alle fall hvis du ikke har en erfaren utvikler. Og med det mener jeg at du må ha jobbet med dette her i veldig lang tid. Det å kunne ta et konkret tilfelle og utvide det og så tenke. Du produserer de samme feilene igjen og igjen altså, hvis ikke du passer deg. Ikke med de samme ti linjene med kode, men du bare gjør det på en annen måte. Hvis du er litt tankeløs her og der så har du plutselig introdusert problemstillinger du ikke tenkte på."**

Han mener AMM'et er for lite spesifikt. Ved å gjøre det litt mer konkret mener han at også utviklere vil være med. Tidligere i sitat 2.6 så vi at U1 lanserte ideen om variasjo-

ner av AMM knyttet til ulike roller og at dette ville være en god tilnærming. Her utdyper han dette med å påpeke hva han mener mangler for å få med seg utviklere. Det mest interessante er at han mener AMM kan være viktige for uerfarne utviklere. Det kan, som respondenten forteller, være vanskelig for en uerfaren utvikler å generalisere på grunnlag av én konkret løsning man har vært gjennom, for senere å trekke analogier til lignende løsninger. En av hovedhensiktene med konseptet AMM er at det kan gjøre det enklere å trekke slike analogier.

5.3.3.2 Notasjonsformer

I det neste sitatet er respondent U1 inne på ulike notasjonsformer:

> **"Jeg vil vel si at du faktisk er nødt til å ha begge deler fordi at noen mennesker tenker best med figurerer som de kan peke og tegne på etterpå, mens andre de liker å ha det veldig kort og konsist. Så jeg tror en konsis språkbruk uten alt for lange utredninger med figurer som fokuserer på kjernen i tingene er det som skal til. Og den notasjonen, hva skal jeg si, la oss kalle det den tekstlige fremstillingen og den billedlige fremstillingen bør standardiseres sånn at det blir enkelt å kommunisere. Så da er kanskje en UML-syntaks og sånt kanskje grei å bruke. Du må ha begge deler (tekst og figur), det er du bare nødt til altså." (Sitat 3.2)**

Her er han opptatt av at det er viktig å benytte både grafisk og tekstlig notasjon i et slikt mønster. Han påpeker at det vil være individuelt hva folk med ulik bakgrunn vil forstå og trives best med.

5.3.4 HOLDNINGER TIL BESVÆR

I denne kategorien av funn har jeg plassert sitater som gir et inntrykk av holdninger som kan virke inn på hvordan et verktøy som AMM vil bli mottatt. Negative holdninger til det fundamentet et verktøy bygger på vil selvfølgelig være avgjørende for hvor vellykket det vil være i bruk. Flere av de følgende sitatene er ytret i en mer eller mindre spøkefull tone, men inneholder sannsynligvis et visst snev av sannhet.

5.3.4.1 Holdninger til applikasjonssikkerhet

> **"Er det buddhistene som har Nirvana? Programmererne de har applikasjonssikkerhet." (Sitat 4.6)**

Her viser respondent U3 at han ikke har den store tiltroen til at applikasjonssikkerhet vil kunne bli noe annet enn et fjernt mål. (Vel å merke vil jeg ta høyde for at respondenten har en liberal tolkning av Nirvana.) Når han i neste sitat underbygger påstanden med å si at det er kompleksiteten i arkitekturen som skaper problemet, kan det virke som om han gir uttrykk for en viss maktesløshet:

> **"Du er jo bare ett tannhjul i maskineriet, ikke sant. Charlie Chaplin gir en god beskrivelse av hvordan det er blitt som det er. Fordi det er ingen som har totalansvaret, ikke sant. Du moduliserer alt mulig, ikke sant. I "design by contract" og sånt som det der. Du vet akkurat hva du skal levere, hva du får inn og hva resultatet skal bli. I en ideell verden kan det godt hende at hver komponent er flott og sikker og alt det der, men summen av alt, det er der problemet ligger." (Sitat 4.7)**

Som en motpol til denne holdningen, og for å videreføre respondentens analogi, vil jeg påstå at den eneste måten å oppnå "Nirvana" på for en programmerer må være å starte i det små. Uten at man går til oppgaven med hele sitt hjerte, vil nok Nirvana heller aldri oppnås. Spør en buddhist.

5.3.4.2 Holdninger til kravarbeid og systemering

I det neste sitatet skal vi se at respondent S1 gir et ganske klart inntrykk av hva han mener om kravarbeid:

> **"Jeg har prøvd å lese kravspesifikasjoner i den hensikt å implementere noe i henhold til en kravspesifikasjon da, som programmerer …" (Sitat 4.3)**

I det han sier "prøvd" er det lett å få et inntrykk av at han mener dette passer dårlig med hvordan han selv mener applikasjonsutvikling bør foregå. Det er lett å forstå at dårlig utført kravarbeid kan føre til negative holdninger hos dem som er nødt til å ta konsekvensene av det. På den annen side burde det være lett for en programmerer å se at et

godt kravdokument også vil gjøre rollen hans lettere å fylle. Slik det er i mange tilfeller i dag ligger veldig mye av sikkerhetsansvaret på programmereren, noe han ikke nødvendigvis er klar over, eller vil være seg bevisst. Det er som respondent U1 sier i sitat 2.2: Det er hans "jobb som programmerer å sørge for at det ikke er mulig å putte inn ugyldige data". Hvis man ikke planlegger systemene bedre med hjelp av et grundig kravarbeid, er det vanskelig å være uenig i den påstanden.

> **"Nei. Jeg vet at det finnes noen standarder for hvordan man skal formulere sånne ting, men det… Jeg er veldig praktisk anlagt altså, så jeg hever meg over sånt." (Sitat 4.4)**

Respondent S1 underbygger her sin tidligere uttrykte holdning til selv å være med på å definere krav (Sitat 4.3). Denne respondenten har kanskje litt av sine ord i behold på bakgrunn av at han til daglig driver opplæring i applikasjonssikkerhet, men allikevel utgjør holdningen et lite konstruktivt fundament med hensyn til å bygge applikasjoner på et solid kravarbeid. Det blir ekstra interessant når man ser hvilke holdninger to av de andre utviklerne ser ut til å gi uttrykk for. Når jeg for eksempel spør respondent U3 om han bruker UC-notasjonen i det daglige, svarer han:

> **"Nå må jeg skuffe deg litt: Minst mulig!" (Sitat 4.2)**

Som en kontrast til utviklernes holdninger er det på en måte befriende å se at man har mye å ta tak i. Respondent D1 svarer følgende på spørsmål om hvorvidt sikkerhetskrav ble diskutert i en bestemt kravarbeidssituasjon hun var med på:

> **"Nei. Men vi har jo ikke bedt om det da." (Sitat 4.5)**

Mener respondenten at det er hennes ansvar, eller forutsetter hun bare at utvikler tar seg av slikt? I det neste sitatet, fra D1, utdypes temaet. På spørsmål om hvorvidt sikkerhet ikke i det minste ble nevnt av utvikler, svarer respondenten:

> **"Vi var jo veldig opptatt av. Hvis vi henter et dokument i fra vårt område og legger ut på Internet. Vi turte jo nesten ikke å gjøre det. Så vi snakket jo sånn indirekte om det da. Og da sa de at det var helt uproblematisk." (Sitat 4.6)**

I lys av potensielt dårlige kravspesifikasjoner kan det være verdt å merke seg utsagnene fra respondent D1. Det er tydelig at hun og hennes medarbeidere har vært opptatte av sikkerhet, men de får beskjed om at det er ”uproblematisk”, i hvert fall det ene problemet de tar opp. Her kunne sikkerhet vært diskutert, men utvikler lot være. Hva som var grunnen til at utvikleren ikke diskuterte sikkerhet noe nærmere, kan en bare spekulere på.

Det er naturlig å la fagfolk ta seg av komplekse tekniske problemer, som i dette tilfellet er applikasjonssikkerhet. En domeneekspert vet at han eller hun vil ha en sikker applikasjon, og er kanskje spesielt opptatt av en bestemt detalj som i tilfellet over. La meg bruke en illustrerende analogi: Ingen forventer at man skal oppgi alle tekniske detaljer ved et hus man for satt opp. Slikt overlater man til fagfolk. Men man forventer at fagfolk diskuterer fordyrende alternativer med kunden.

Her er det to holdninger som er verdt å merke seg. Den ene holdningen tilhører domeneeksperten. Hun har en profesjonell tillit til fagfolkene hun er i kontakt med. Den andre holdningen er utviklerens holdning som uttrykkes både gjennom mangel på fokus og gjennom å påstå at det er helt uproblematisk med sikkerhet. Flere av årsakene som Schneier har pekt på som problematiske kan være grunnlaget for holdningen til utvikleren, men det kan selvsagt være tilfelle at akkurat denne utvikleren har ordene sine i behold selv om både statistisk sannsynlighet og ekspertuttalelser taler i mot det.

5.4 SLUTTKOMMENTAR TIL DEN EVALUERENDE FASEN

Evalueringen har avdekket mange interessante fakta, så vel som holdninger, som vil ha en avgjørende betydning for om man skulle forsøke å innføre en teknikk som AMM i utviklingsprosessen. Det er tydelig at kravarbeid gjerne blir tillagt mindre vekt i hverdagen enn det gjør i teorien. Vi har sett at kommunikasjon anses som problematisk, en diskusjon også fremskaffer mange gode forslag til forbedringer og alternative tilnærminger til å løse dette problemet.

Jeg vil hevde at denne evalueringen gir et indirekte svar på det generelle forskningsspørsmålet, og det er at det virker sannsynlig at AMM kan være nyttig under behandling av sikkerhetskrav. Det spesielle forskningsspørsmålet anser jeg derimot for ikke besvart. Man finner kun antydninger på at svaret kan være positivt. Dermed er det, slik jeg ser det, behov for en ytterligere evaluering for å besvare dette spørsmålet.

6 AVSLUTNING OG VIDERE ARBEID

Formålet med denne oppgaven har vært å utvikle teknikk som kan tilby en forenklet tilnærming til sikkerhet i kravarbeid. Ved å kombinere konsepter fra forskningsfeltene sikkerhet, trusselmodellering, mønstre og kravarbeid har jeg utviklet teknikken Abstrakte misbruksmønstre. I tillegg har jeg foreslått en metode som kan bygge videre på AMM for å bevare fokus på sikkerhet videre fremover i utviklingsprosessen.

6.1 KONSEPTUELL FASE

Det er bred enighet blant forskere om at et grundig kravarbeid er avgjørende for å utvikle et godt system. Som vi har sett gjelder dette i minst like stor grad hvis man ønsker å ta hensyn til applikasjonssikkerhet allerede i kravarbeidet.

I det foregående er det presentert ulike konsepter for tilnærming til effektiv formidling og dokumentasjon av forskjellige sider ved systemutviklingsprosessen. Komplekse sikkerhetsproblemer abstraheres med hjelp av en kombinasjon av en mønstermal og MC-notasjonen. Resultatet er en kombinasjon av både tekstlig og grafisk notasjon. Dette er på ingen måte en ny tilnærming til mønstre, men det er en ny tilnærming til både sikkerhetskrav spesielt og kravarbeid generelt. Tilnærming til sikkerhetsproblemene skjer gjennom å analysere og klassifisere sårbarhet og misbruk.

MC-notasjonen er en lett forståelig tilnærming til modellering av misbruk. Ved å utvide notasjonen med prinsipper fra angrepstrær blir det også lettere både å abstrahere og å fortette komplekse sikkerhetsproblemer. Om man har behov for komponering eller dekomponering er avhengig av hvilken fase man befinner seg i av utviklingsprosessen. I sin mest abstraherte form vil AMM egne seg best til innsamling av sikkerhetskrav i en tidlig fase. For å bringe problemene og løsningene fra AMM videre i utviklingsprosessen er det nødvendig med andre teknikker som kan bygge videre på AMM.

Presentasjonsformen, som er hentet fra mønstre, øker mulighetene for forenklet formidling av vanlige problemer og gjenbruk av gode løsninger. Bygger man opp et bibliotek av AMM'er og kombinerer disse med andre teknikker for mer omfattende trusselmodellering, analyse og design, kan dette danne utgangspunktet for en komplett metode

som vil effektivisere arbeidet med sikkerhetskrav på flere måter. Med denne dokumentasjonen for hånden vil det være enklere å ta avgjørelser med hensyn til hvilken effekt sikkerhetskrav vil ha på design og realisering av både enkle applikasjoner og komplekse systemer.

En av intensjonene med AMM er å fokusere på kommunikasjon mer enn på teknologi. På den måten blir komplekse tekniske detaljer skjult og dermed lettere å forstå for person uten IT-teknisk bakgrunn. Man ønsker å fremstille et komplekst teknisk problem på en lettfattelig måte. Et AMM skal være enkelt å formidle og ha en medierende effekt mellom kravarbeidere med ulik bakgrunn.

6.2 EVALUERENDE FASE

Intervjuene har ført til at jeg ser hele kravarbeidsprosessen i et nytt lys. Blant annet ser jeg at hovedfokuset for AMM kanskje bør ligge på kommunikasjon og forståelse i spesifiseringsarbeidet og i analysefasen, fremfor i uttrekningsfasen.

Videre innser jeg at operasjonaliseringen kunne vært mer vellykket. Den burde fokusert på å gjøre variablene og verdiene mer relatert til det praktiske arbeidet med krav. Den definisjonsmessige validiteten (Hellevik '00:51-52) i evalueringsfasen ble for dårlig. Med andre ord ble samsvaret mellom problemstillingen på teoriplanet og de operasjonaliserte variablene på empiriplanet ikke godt nok avklart.

Respondentene presenterer noen interessante synsvinkler på hvordan AMM kan brukes. Det kommer gode forslag til hvilke brukssituasjoner AMM kan ha en funksjon i, ulike rollers ansvar blir problematisert, og presentasjonsformen er gjenstand for både ros og ris. Respondentene viser også holdninger som både kan sies å underbygge behovet for en bedre tilnærming til sikkerhetskrav og som kan tenkes å skape problemer for å innføre en slik tilnærming.

Respondentene mener videre at AMM støtter formidling av forståelse, og at dette vil kunne føre til et bedre grunnlag for enighet kravarbeidere i mellom. Hovedproblemet består i det at de ikke er villig til å godta at dette vil fungere overfor domeneeksperter, eller med andre ord overfor personer uten verken teknisk IT-kompetanse eller forståelse

for sikkerhet. Derfor kommer de med en del forslag til andre sammenhenger der det kan være aktuelt å bruke AMM i stedet.

Det var en lærerik prosess å sakte men sikkert oppdage at kartet ikke nødvendigvis stemmer med terrenget når utgangspunktet baserer seg på teoretiske tilnærminger til praktiske problemer. Den spesielle problemstillingen ble ikke direkte besvart, kanskje fordi den var for dårlig operasjonalisert, men dette anser jeg for å være en av vanskelighetene med denne typen oppgave. Det er vanskelig å vite så mye om hvordan man skal angripe et nytt problem. Derfor er også oppgaven gjennomført med en utforskende tilnærming. Selv sitter jeg med et inntrykk av at evalueringen på mange måter var vellykket, og at den har tjent sitt hovedformål: Å gi inspirasjon til nye ideer og tanker om videreutvikling.

6.3 FORSLAG TIL VIDERE ARBEID

Det er spesielt mange sider ved dette konseptet som jeg mener bør få videre oppmerksomhet, rett og slett fordi behandling av sikkerhetskrav vil være av avgjørende betydning for den generelle IT-sikkerheten fremover. Først og fremst bør det utvikles en AMM-katalog innenfor et avgrenset applikasjonsdomene, slik at det blir mulig å gjennomføre et grundig kasusstudium. Det er viktig å finne ut om AMM vil fungere i praktisk bruk slik det er utformet i dag, og om form eller abstraksjonsnivå må revurderes. En annen tilnærming til et slikt kasusstudium kan være å jobbe videre med forslaget om AMM'er med utgangspunkt i forskjellige perspektiver, og få testet ut dette i et kasusstudium.

Et viktig aspekt ved en eventuell innføring av denne typen teknikk eller hel metode for behandling av sikkerhetskrav, er at det blir integrert i et CASE verktøy. Et forslag jeg ser på som interessant, spesielt for Institutt for informasjonsvitenskap, er å utvikle en CBR-basert[22] løsning som kan integreres med et allerede eksisterende verktøy for modellering av misbruk. Det bør la seg gjøre å implementere en form for automatisk mønstergjenkjenning, som automatisk vil forandre måten man lager UC-modeller på, og

[22] CBR er et akronym for Case Based Reasoning og er en del av forskningsfeltet kunstig intelligens.

sikkerheten vil mest sannsynlig bli bedre ivaretatt enn hvis man overlater dette til den enkelte utvikler.

På det teoretiske planet er det også en del å avklare. Motivasjonen for å benytte mønstre som utgangspunkt bør utforskes grundigere. Kanskje bør det defineres noen regler for hvordan man skal abstrahere misbrukstilfeller fra klasser av konkrete sikkerhetsangrep. For eksempel kunne det vært interessant å forsøke å utvikle AMM'er i samsvar med 15408.

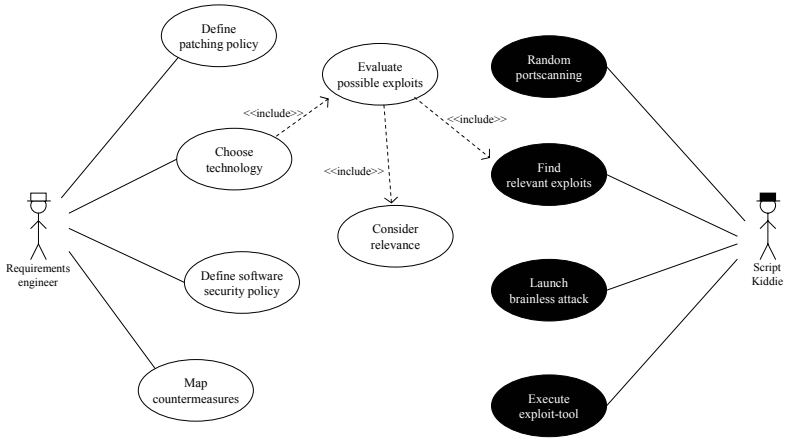
Mange sider ved oppgavens tema fortjener altså videre arbeid. Det faktum at oppgaven er utforskende har nettopp åpnet for videre arbeid med en rekke interessante problemer. Oppgavens hovedbidrag til faget er et grunnlag for videre arbeid.

6.4 SIKKERHET SOM TROLL AV ESKE

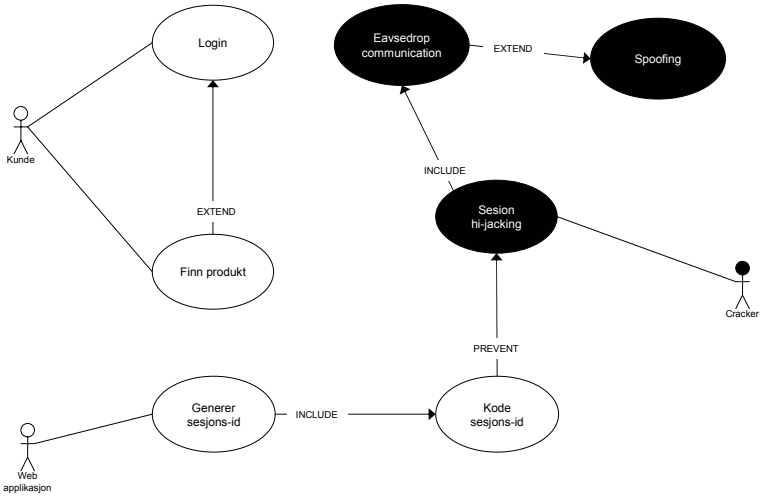
Tittelen på denne oppgaven er kanskje noe optimistisk, men jeg velger allikevel å tro at både AMM, og resten av metoden som er foreslått, representerer et steg i riktig retning med hensyn til å oppnå høyere applikasjonssikkerhet, og dermed også sikkerhet generelt.

I introduksjonen viste jeg Spaffords utsagn om at fokus på sikkerhet i tilknytning til applikasjonsutvikling vil gi "security right out of the box". Jeg vil gå litt lenger og hevde at mange faktisk vil bli overrasket over hvor stor innvirkning på sikkerheten det vil ha å fokusere på applikasjonssikkerhet allerede i kravarbeidet. Mange vil oppleve at man oppnår sikkerhet som troll av eske.

VEDLEGG 1 – UTVIKLINGSSKISSER



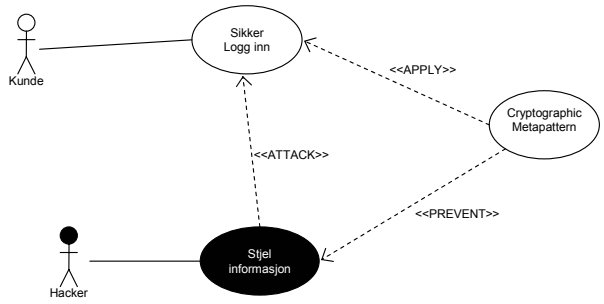
1



2

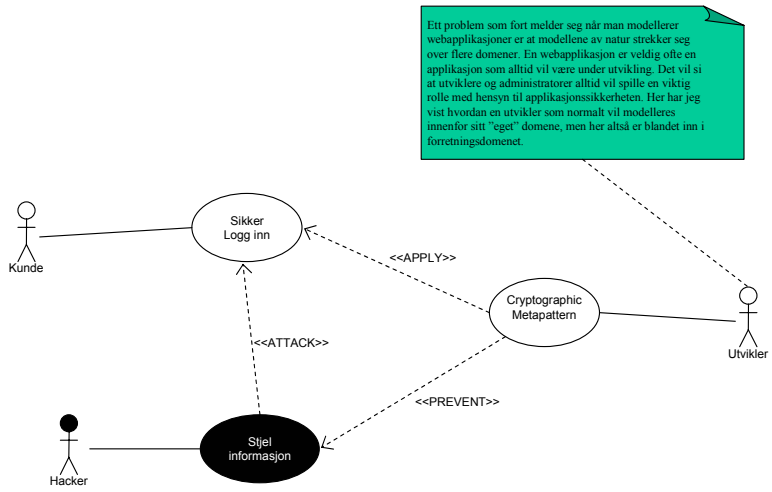
(Bellovin 89; Morris 85; Pennington 01; RFC 2616 99; Security Focus 27.10.00)

Normal Misuse Case notasjon



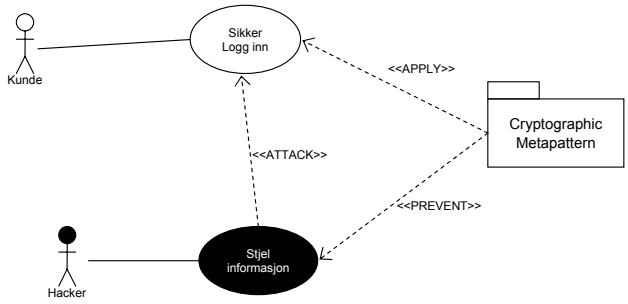
3

Misuse Case modellering på tvers av modelleringsdomener.



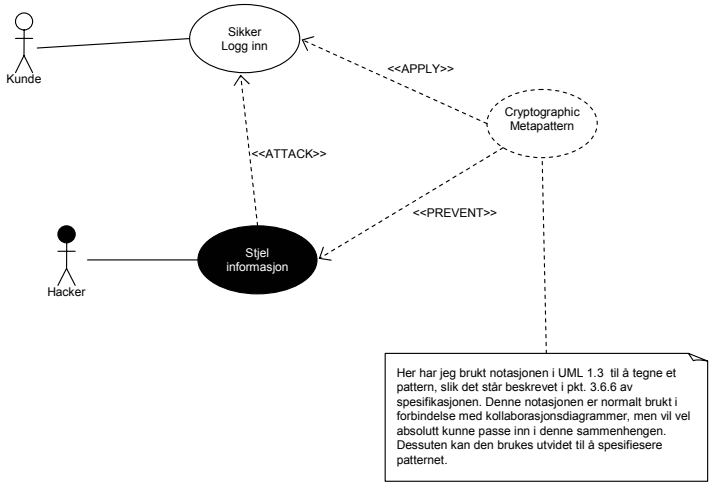
4

Utvidet Misuse case notasjon med mulig patterns notasjon i form av en komponent.

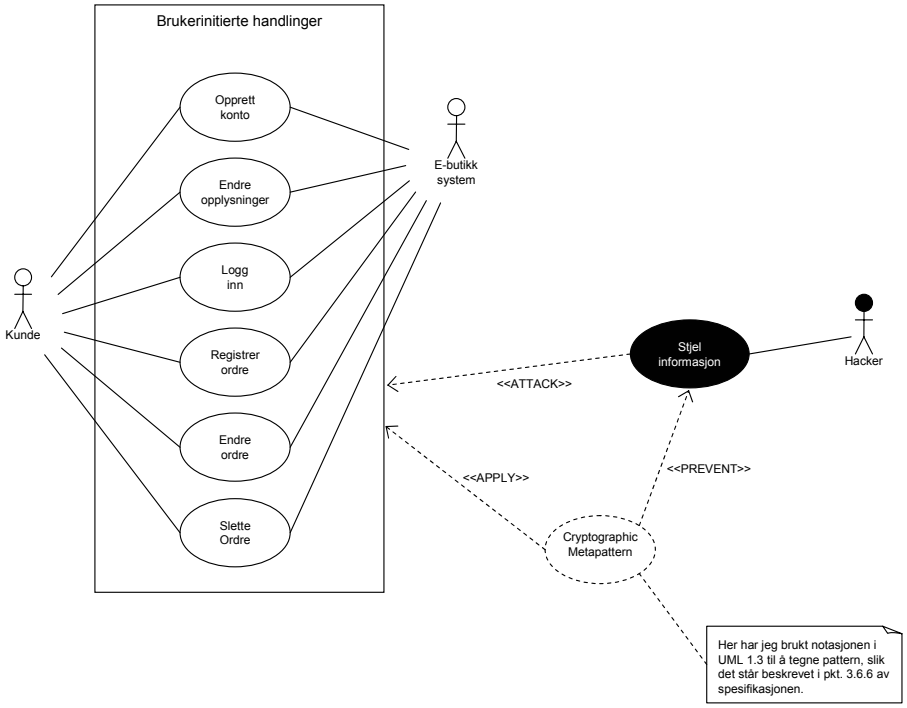


5

Utvidet Misuse Case m/patterns notasjon fra UML (1.3 +)

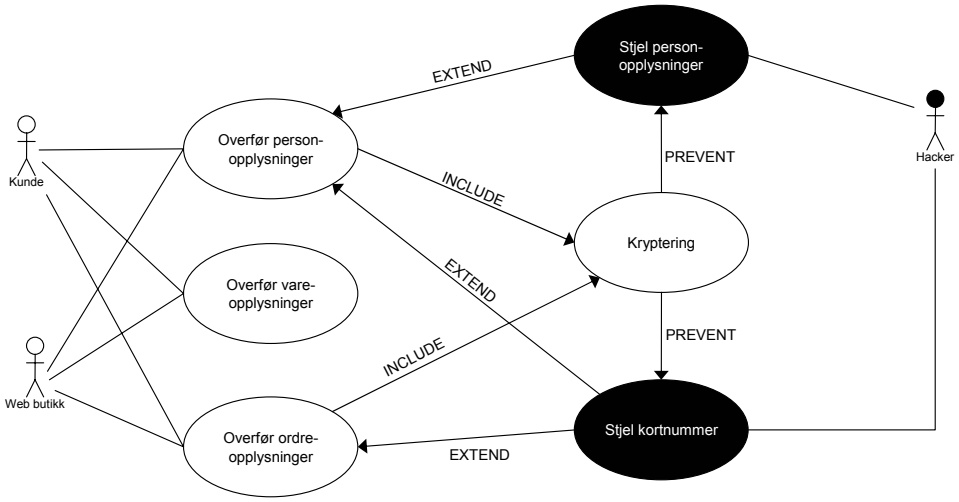


6



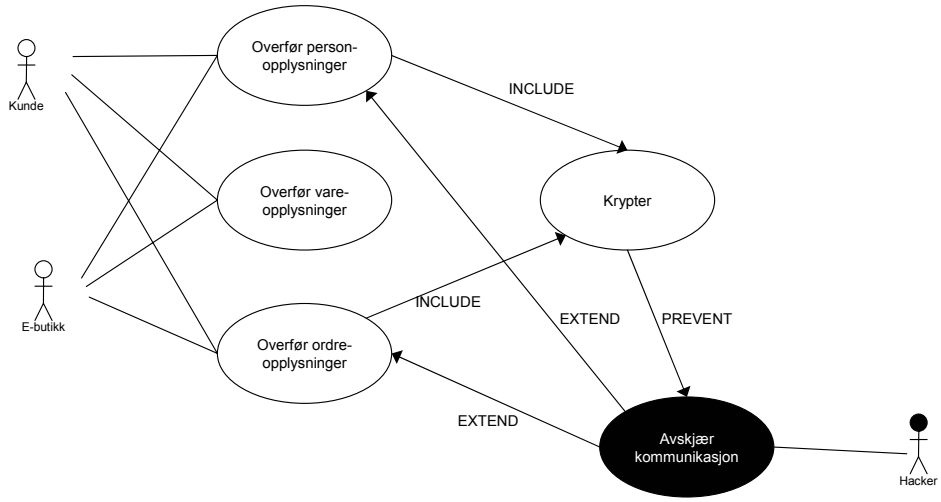
7

Misuse Case nr. 1



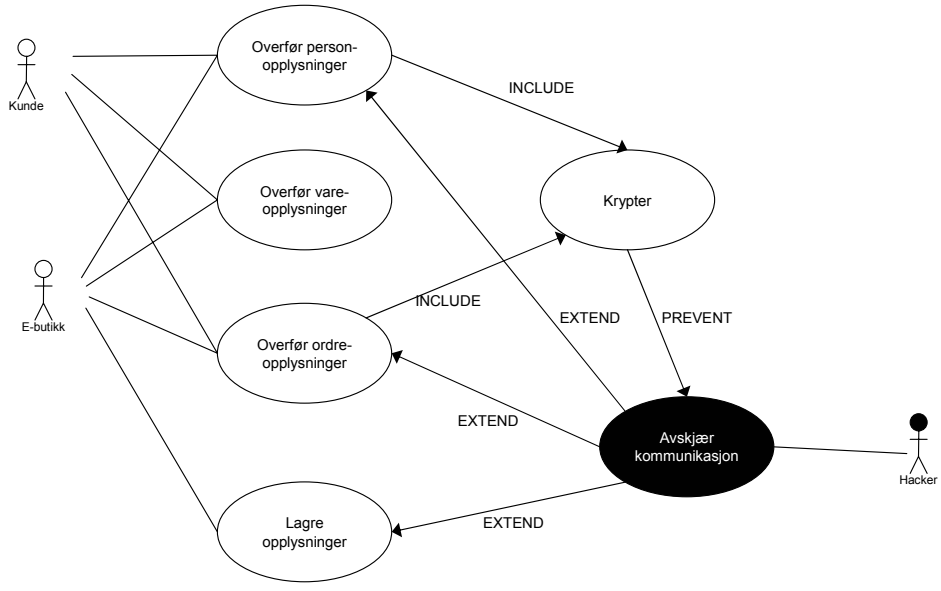
8

Misuse Case nr. 2



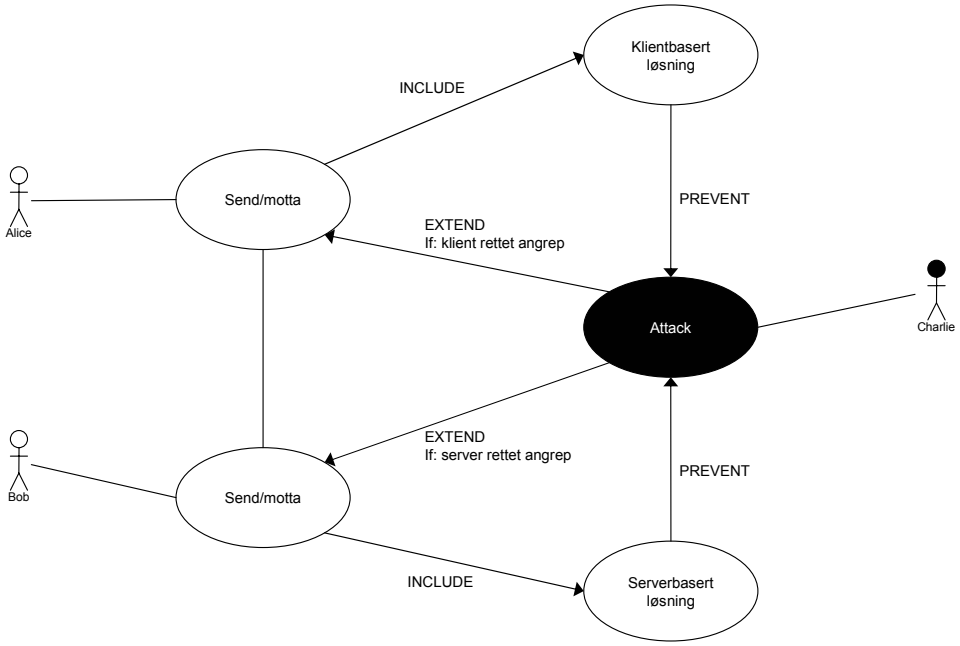
9

Misuse Case nr. 2



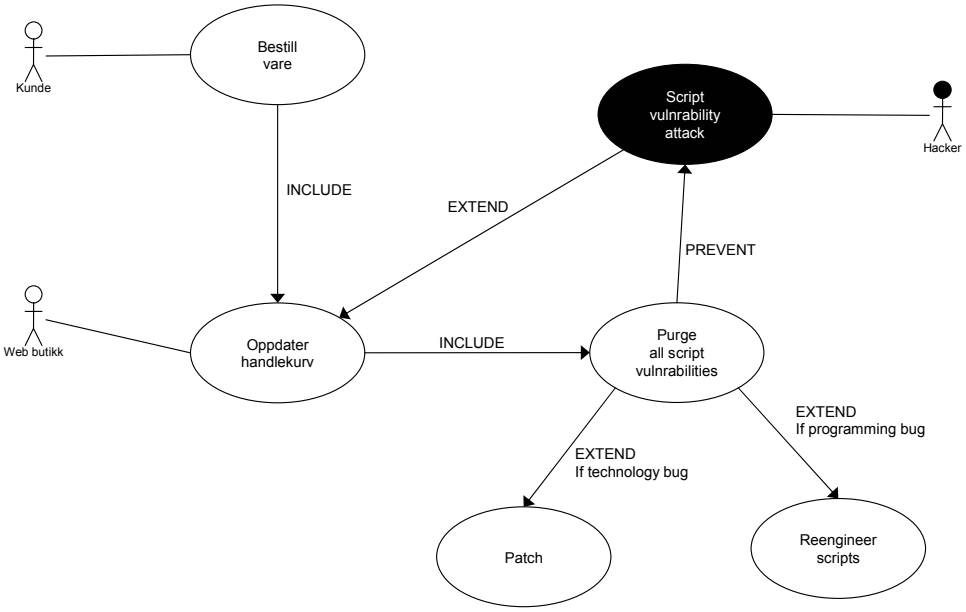
10

Mulig grunnlag for et metapattern for e-handel???



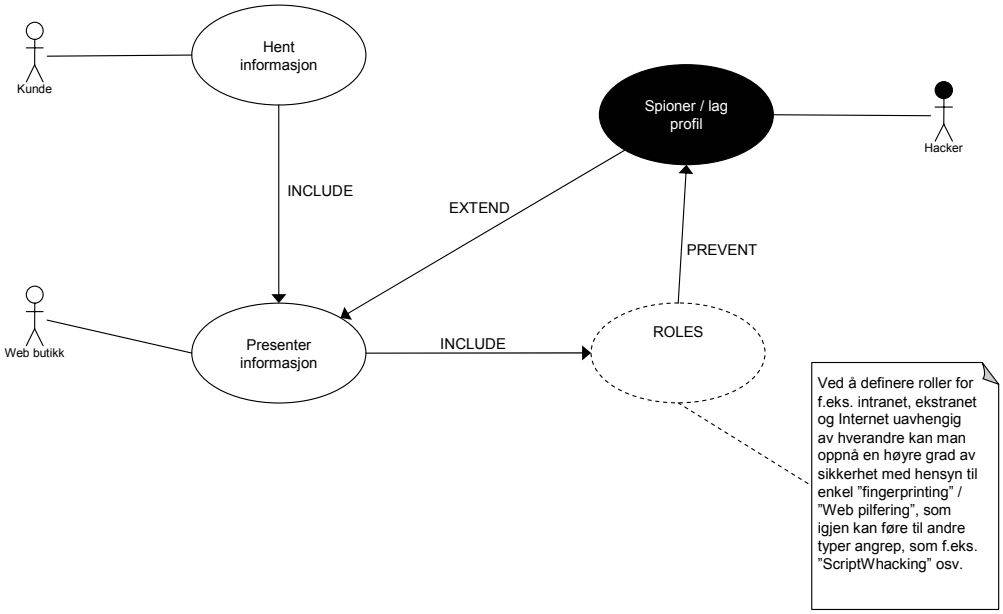
11

ScriptWhacking



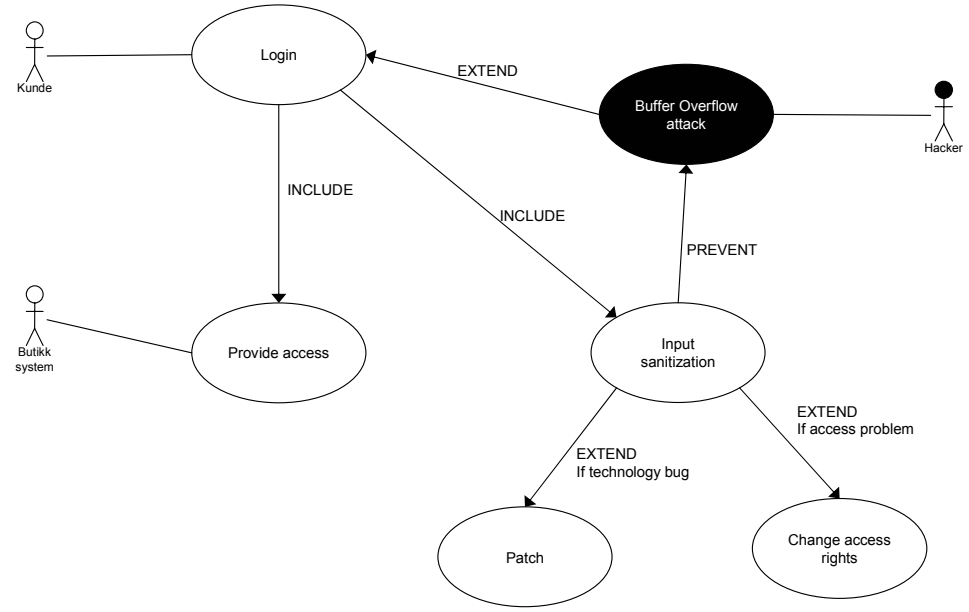
12

Webspionage



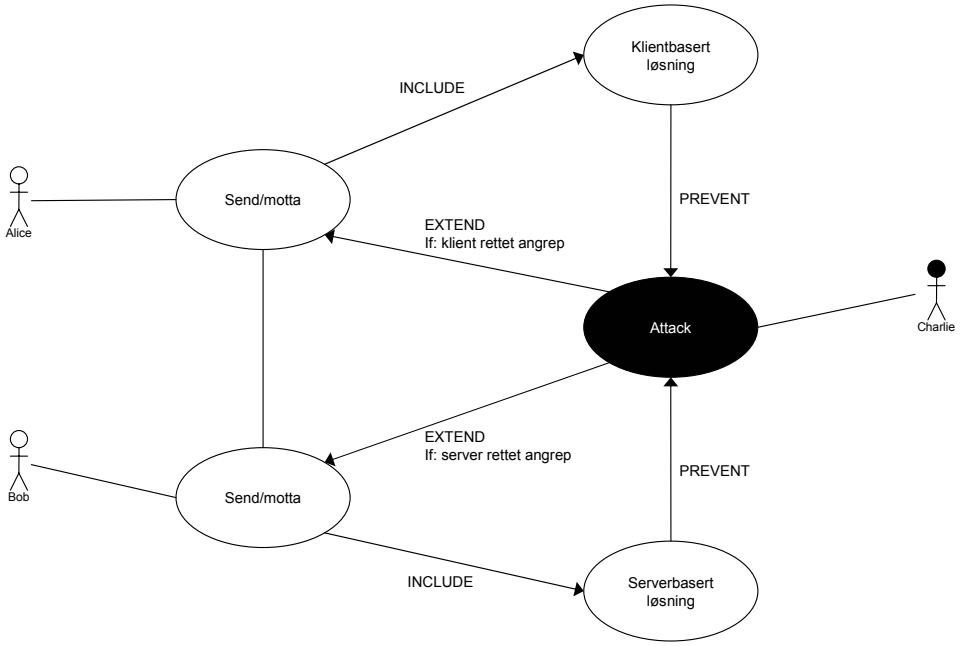
13

BufferSwamping



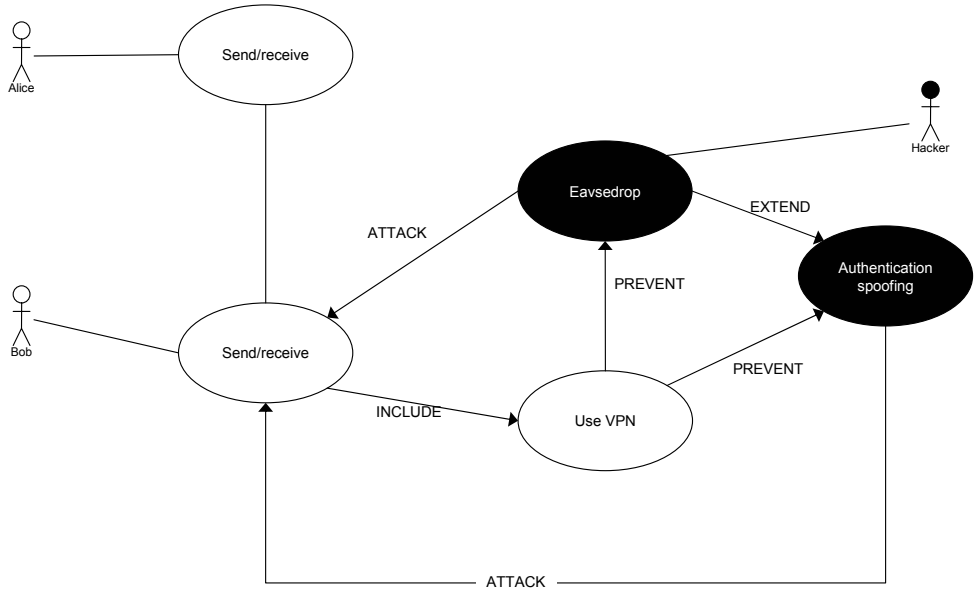
14

Mulig grunnlag for et metapattern for e-handel???

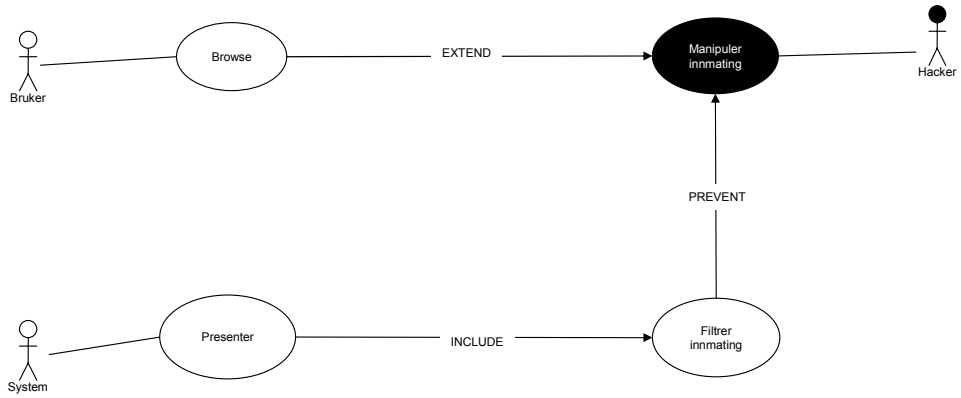


15

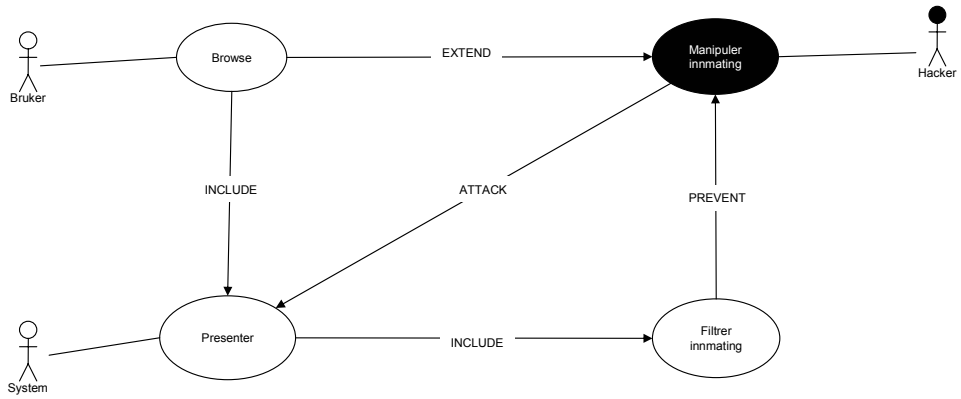
Wireless NotWorking (eks. nett basert på 802.11)



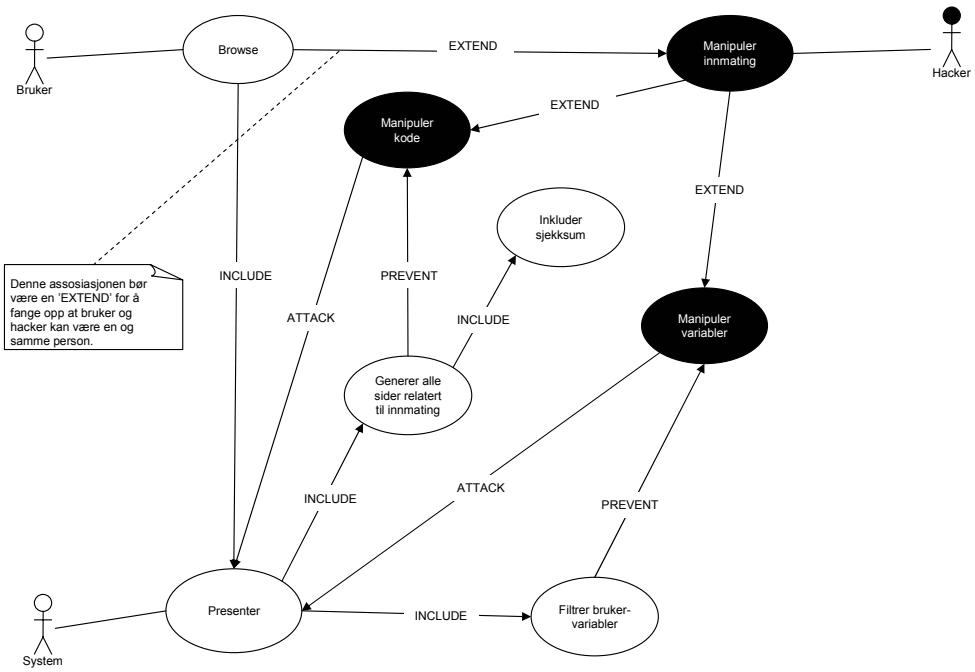
16


Bruker
Browse
EXTEND
Manipuler innmating
Hacker
PREVENT
System
Presenter
INCLUDE
Filtrer innmating

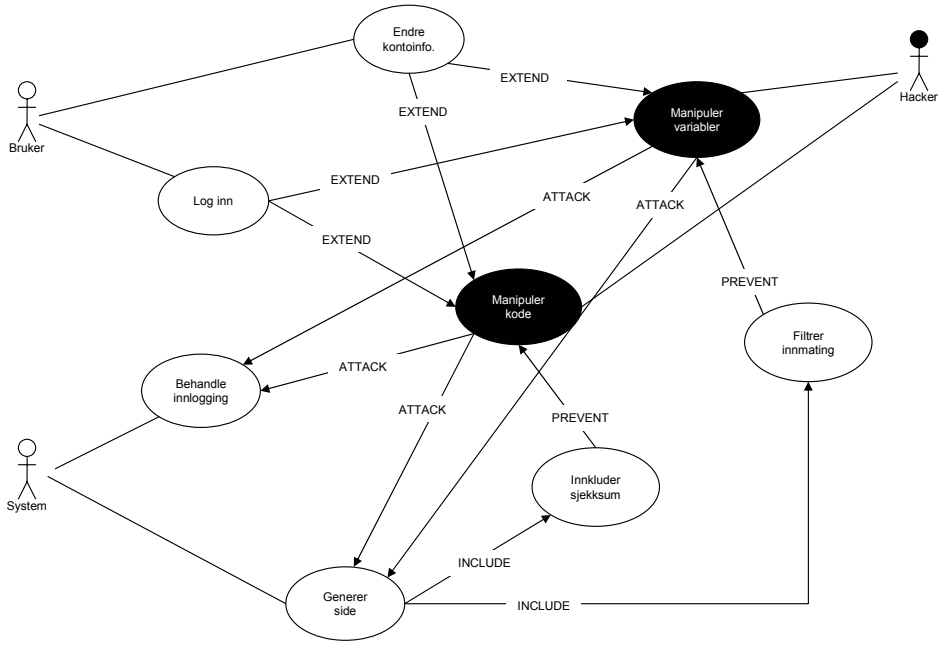
17


Bruker
Browse
EXTEND
Manipuler innmating
Hacker
INCLUDE
ATTACK
PREVENT
System
Presenter
INCLUDE
Filtrer innmating

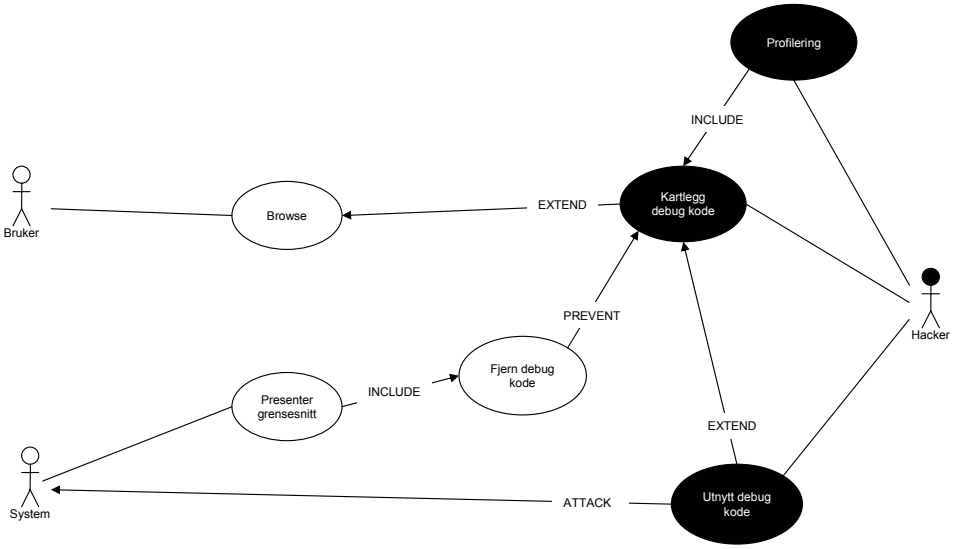
18


Bruker
Browse
EXTEND
Manipuler innmating
Hacker
EXTEND
Manipuler kode
EXTEND
Inkluder sjekksum
Denne assosiasjonen bør være en 'EXTEND' for å fange opp at bruker og hacker kan være en og samme person.
INCLUDE
PREVENT
ATTACK
INCLUDE
Manipuler variabler
Generer alle sider relatert til innmating
INCLUDE
ATTACK
PREVENT
System
Presenter
INCLUDE
Filtrer bruker-variabler

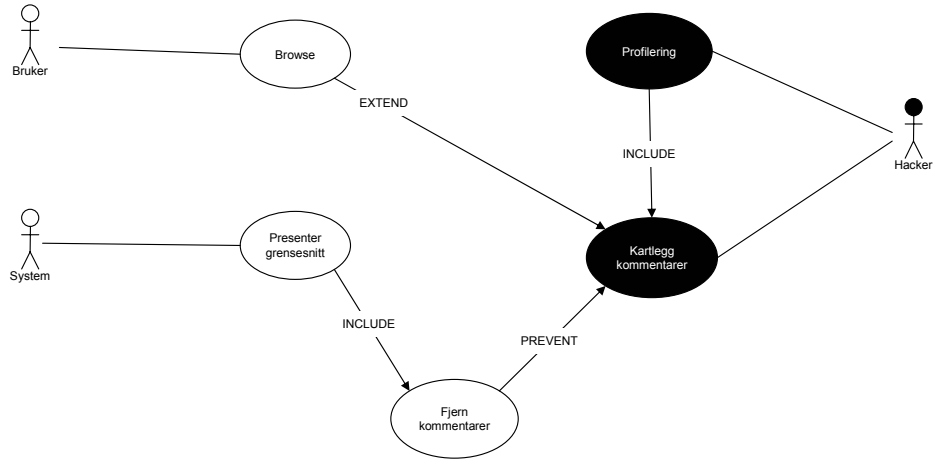
19


Endre kontoinfo.
EXTEND
EXTEND
Hacker
Bruker
Manipuler variabler
Log inn
EXTEND
ATTACK
ATTACK
EXTEND
PREVENT
Manipuler kode
Filtrer innmating
Behandle innlogging
ATTACK
ATTACK
PREVENT
System
Innkluder sjekksum
INCLUDE
Generer side
INCLUDE

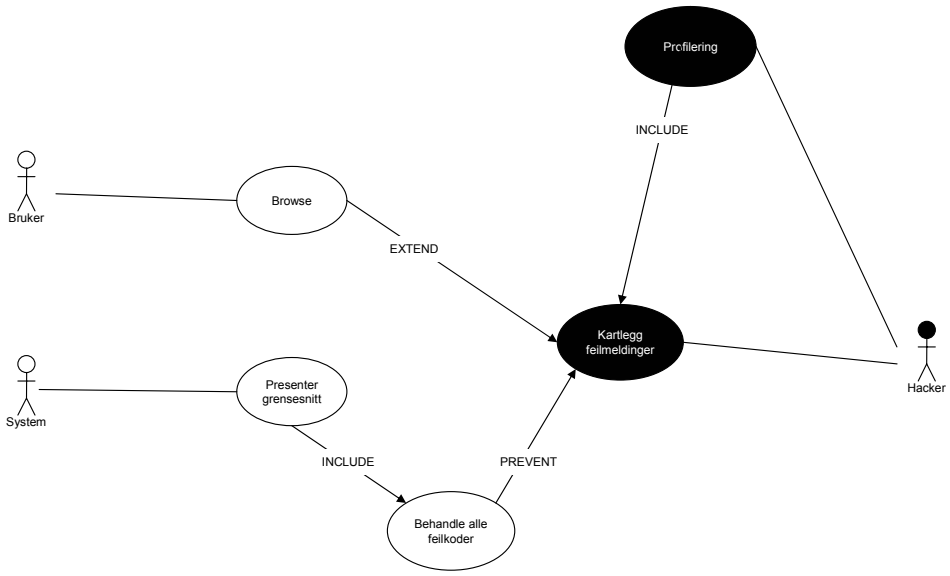
20


Profilering
INCLUDE
Bruker
Browse
EXTEND
Kartlegg debug kode
Hacker
PREVENT
Fjern debug kode
Presenter grensesnitt
INCLUDE
EXTEND
System
ATTACK
Utnytt debug kode

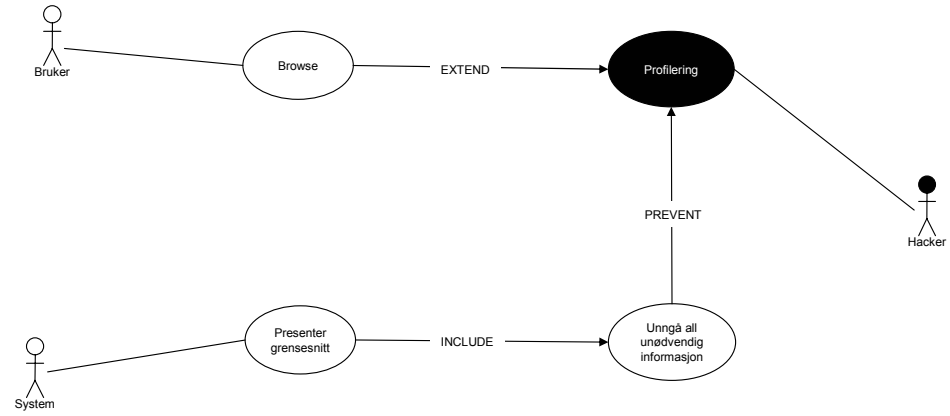
21


Bruker
Browse
Profilering
EXTEND
Hacker
INCLUDE
System
Presenter grensesnitt
Kartlegg kommentarer
INCLUDE
PREVENT
Fjern kommentarer

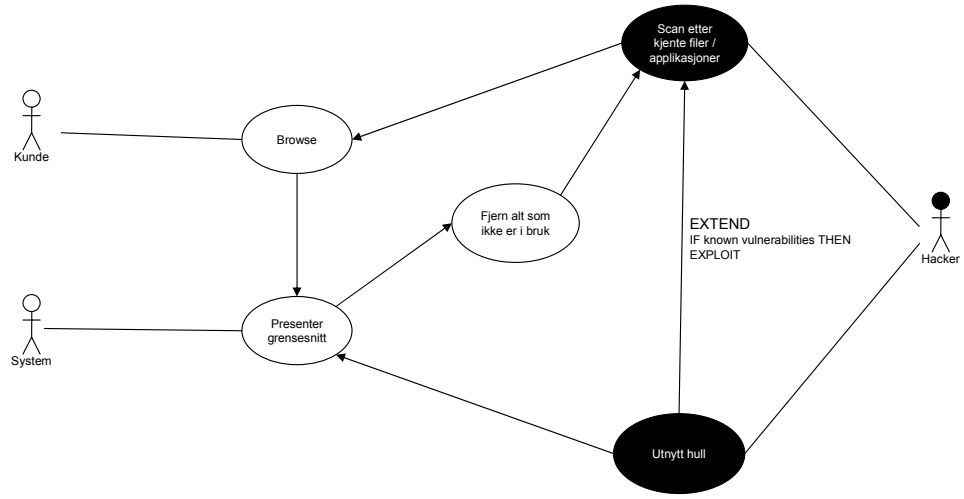
22


Prcfilering
INCLUDE
Bruker
Browse
EXTEND
Kartlegg
feilmeldinger
Hacker
System
Presenter
grensesnitt
INCLUDE
PREVENT
Behandle alle
feilkoder

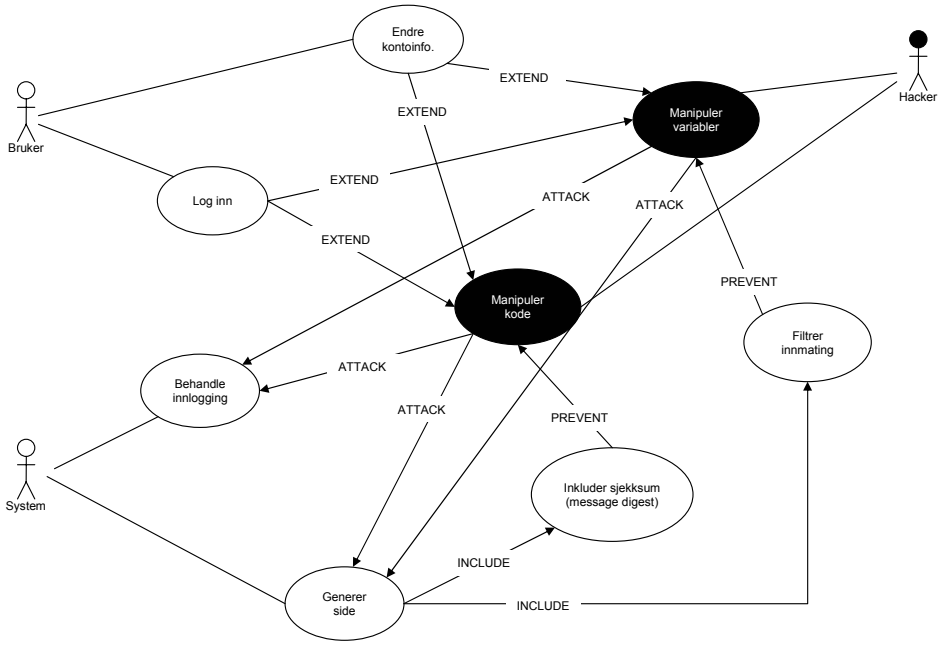
23


Bruker
Browse
EXTEND
Profilering
Hacker
PREVENT
System
Presenter
grensesnitt
INCLUDE
Unngå all
unødvendig
informasjon

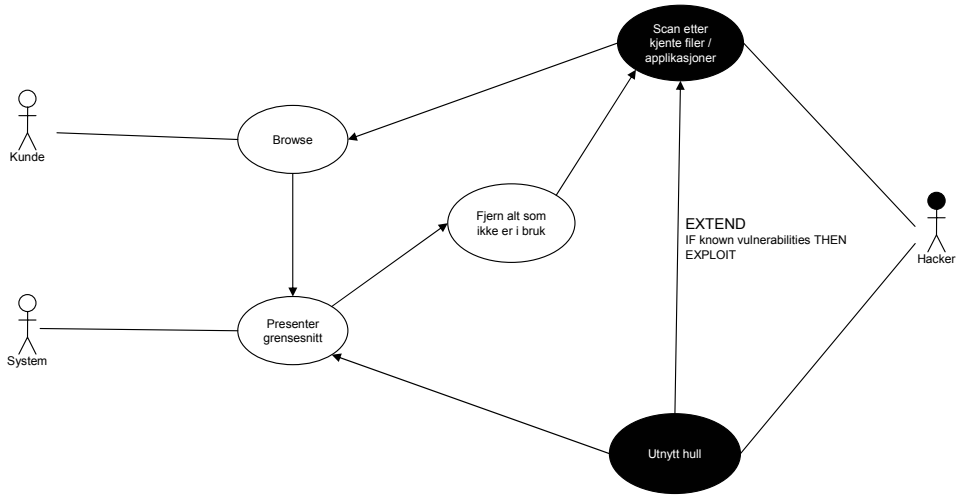
24


Scan etter kjente filer / applikasjoner
Kunde
Browse
Fjern alt som ikke er i bruk
EXTEND
IF known vulnerabilities THEN EXPLOIT
Hacker
System
Presenter grensesnitt
Utnytt hull

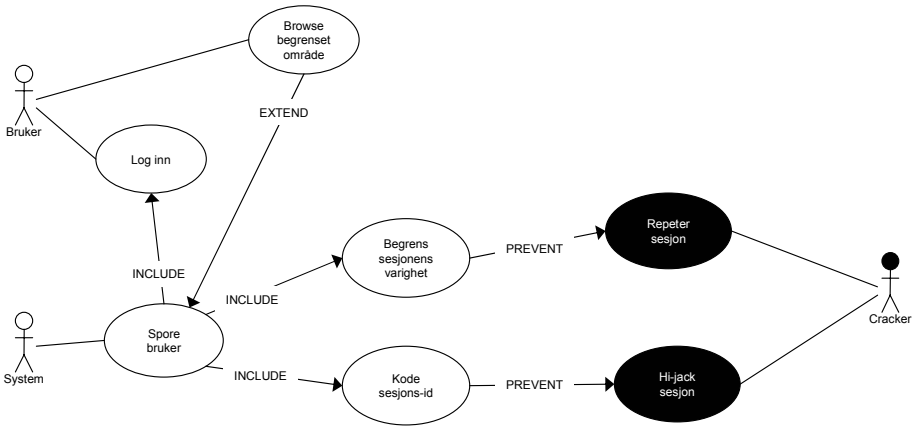
25


Endre kontoinfo.
EXTEND
Hacker
Bruker
EXTEND
Manipuler variabler
Log inn
EXTEND
ATTACK
ATTACK
EXTEND
PREVENT
Manipuler kode
Filtrer innmating
ATTACK
Behandle innlogging
ATTACK
PREVENT
System
Inkluder sjekksum (message digest)
INCLUDE
Generer side
INCLUDE

26

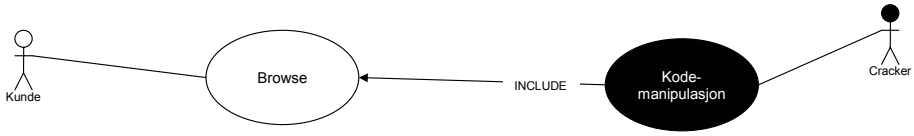

Scan etter kjente filer / applikasjoner
Kunde
Browse
Fjern alt som ikke er i bruk
EXTEND
IF known vulnerabilities THEN EXPLOIT
Hacker
System
Presenter grensesnitt
Utnytt hull

27


Browse begrenset område
Bruker
EXTEND
Log inn
INCLUDE
INCLUDE
Begrens sesjonens varighet
PREVENT
Repeter sesjon
Cracker
System
Spore bruker
INCLUDE
Kode sesjons-id
PREVENT
Hi-jack sesjon

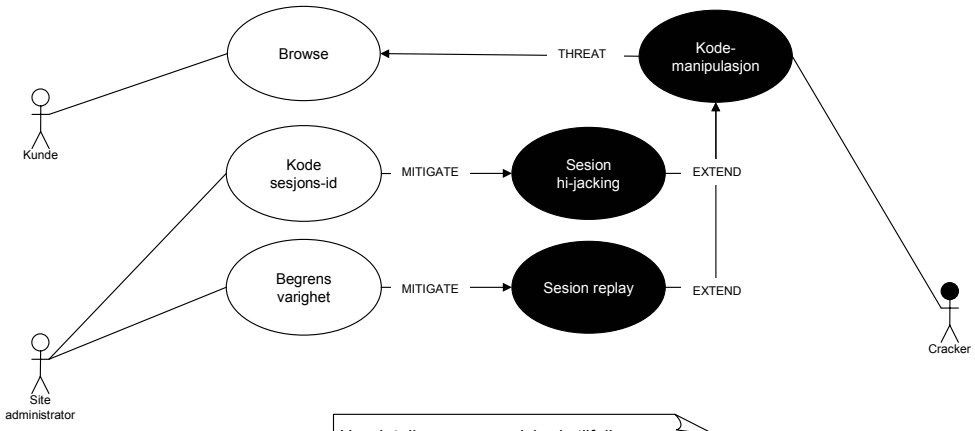
28

KodeManipulason I



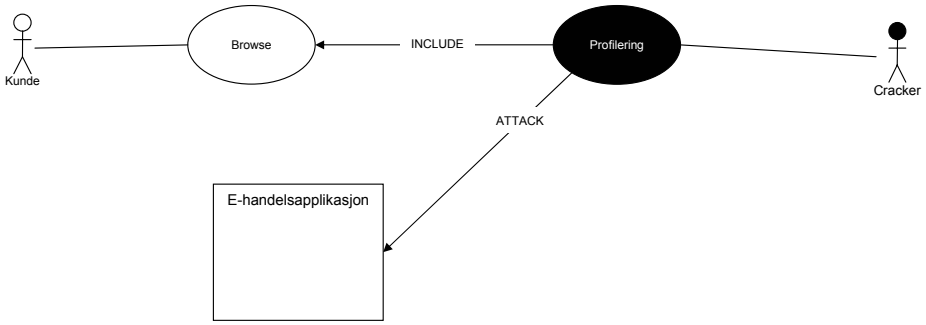
29

KodeManipulason II

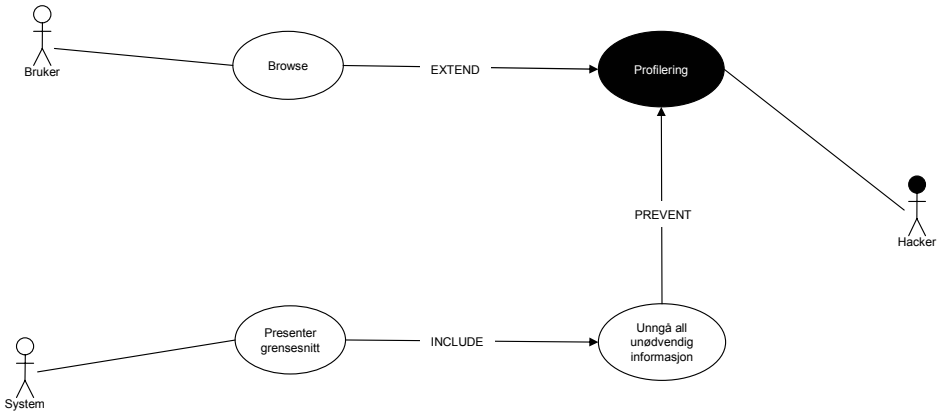


Her detaljeres en av misbrukstilfellene. Dette skal følge opp overskriftene i angrepstreet. F.eks. vil ”Sesion Hi-jacking” foregå ved at en cracker overvåker sesjons-id’en for å finne et mønster. Deretter vil han overta en sesjon i det øyeblikk han har forutsagt den riktige id’en (Krause & Tipton 00:74)

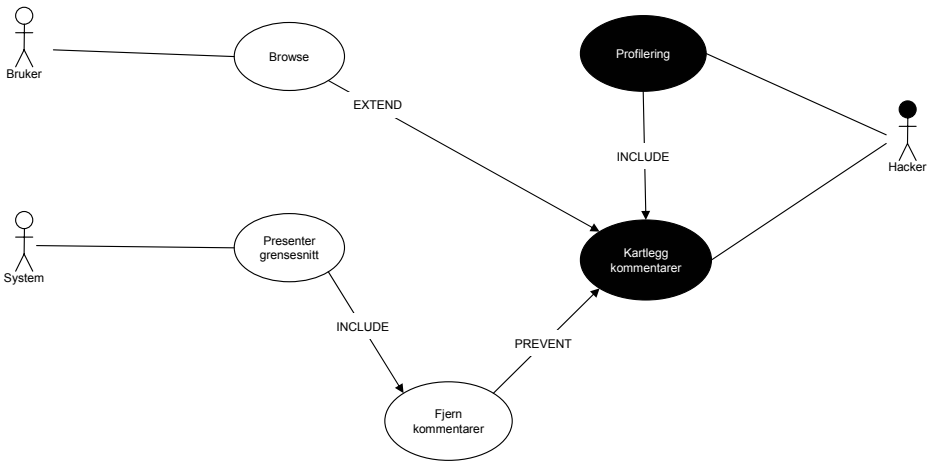
30


Kunde
Browse
INCLUDE
Profilering
Cracker
ATTACK
E-handelsapplikasjon

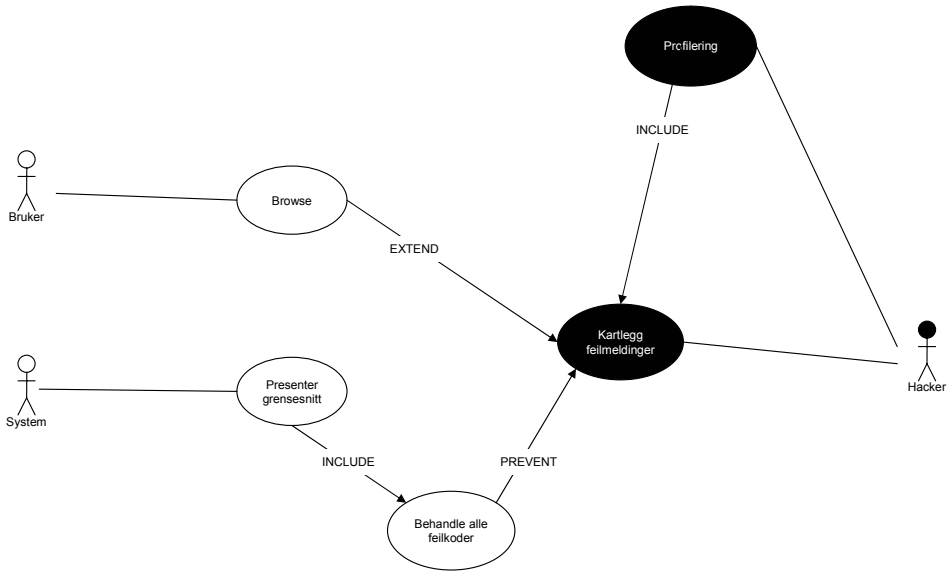
31


Bruker
Browse
EXTEND
Profilering
PREVENT
Hacker
System
Presenter grensesnitt
INCLUDE
Unngå all unødvendig informasjon

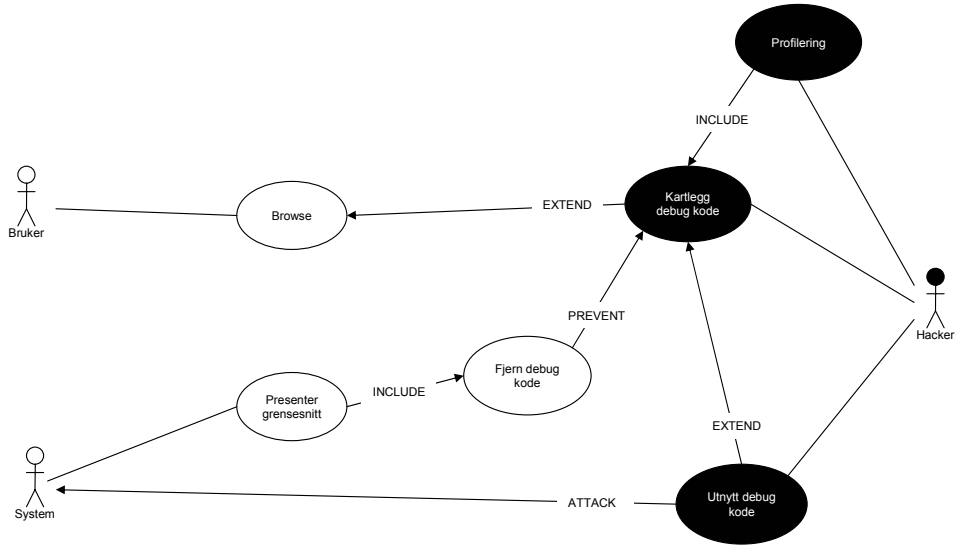
32


Bruker
Browse
EXTEND
Profilering
INCLUDE
Hacker
System
Presenter grensesnitt
Kartlegg kommentarer
INCLUDE
PREVENT
Fjern kommentarer

33

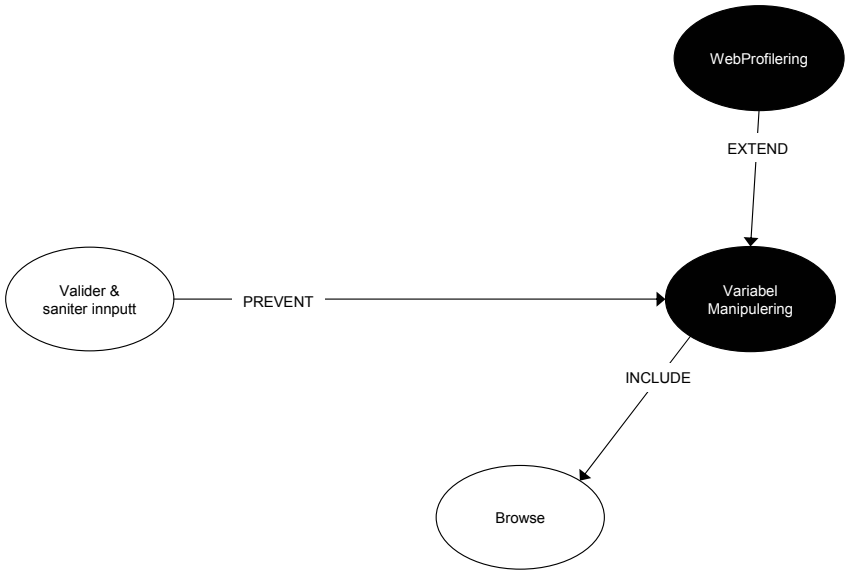

Profilering
INCLUDE
Bruker
Browse
EXTEND
Kartlegg feilmeldinger
Hacker
System
Presenter grensesnitt
INCLUDE
PREVENT
Behandle alle feilkoder

34

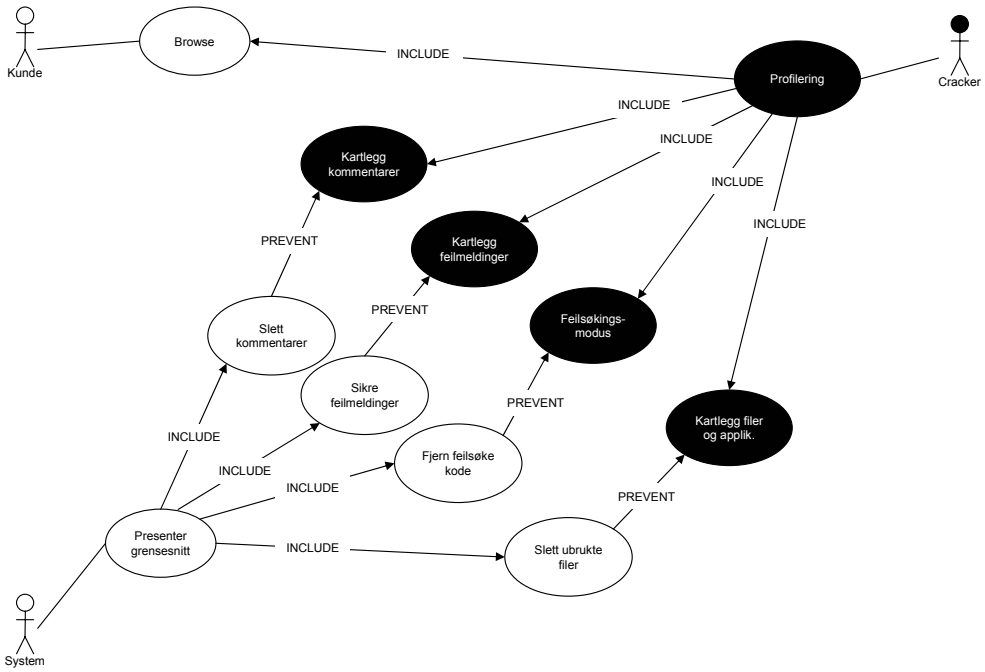


35

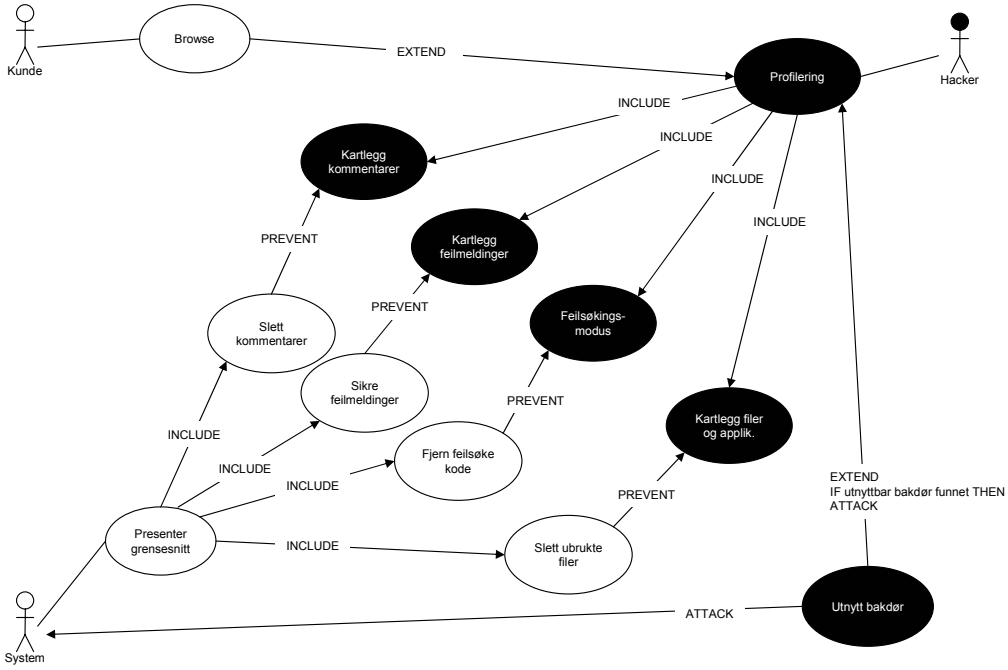
AMM Nr. 4 – Variabel Manipulering



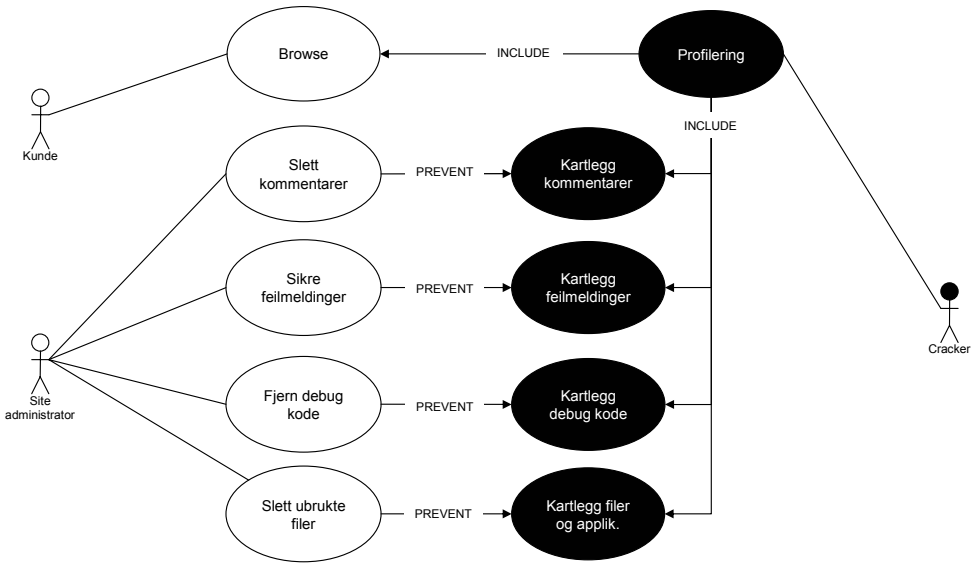
36


Kunde
Browse
INCLUDE
Profilering
Cracker
INCLUDE
INCLUDE
INCLUDE
INCLUDE
Kartlegg kommentarer
Kartlegg feilmeldinger
Feilsøkings-modus
Kartlegg filer og applik.
PREVENT
PREVENT
PREVENT
PREVENT
Slett kommentarer
Sikre feilmeldinger
Fjern feilsøke kode
Slett ubrukte filer
INCLUDE
INCLUDE
INCLUDE
INCLUDE
Presenter grensesnitt
System

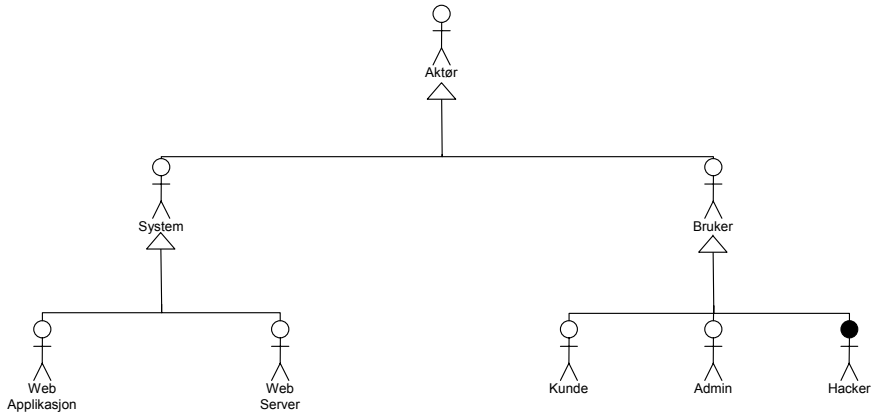
37


Kunde
Browse
EXTEND
Profilering
Hacker
INCLUDE
INCLUDE
INCLUDE
INCLUDE
Kartlegg kommentarer
Kartlegg feilmeldinger
Feilsøkings-modus
Kartlegg filer og applik.
PREVENT
PREVENT
PREVENT
PREVENT
Slett kommentarer
Sikre feilmeldinger
Fjern feilsøke kode
Slett ubrukte filer
INCLUDE
INCLUDE
INCLUDE
INCLUDE
Presenter grensesnitt
EXTEND
IF utnyttbar bakdør funnet THEN
ATTACK
Utnytt bakdør
ATTACK
System

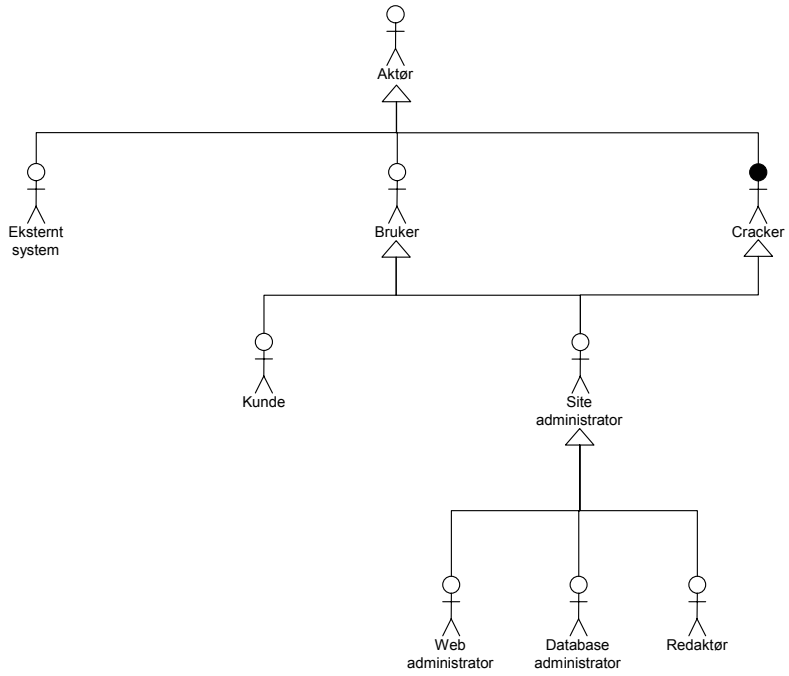
38


Browse
INCLUDE
Profilering
Kunde
INCLUDE
Slett kommentarer
PREVENT
Kartlegg kommentarer
Sikre feilmeldinger
PREVENT
Kartlegg feilmeldinger
Cracker
Site administrator
Fjern debug kode
PREVENT
Kartlegg debug kode
Slett ubrukte filer
PREVENT
Kartlegg filer og applik.

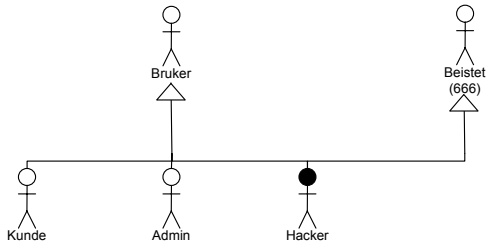
39


Aktør
System
Bruker
Web Applikasjon
Web Server
Kunde
Admin
Hacker

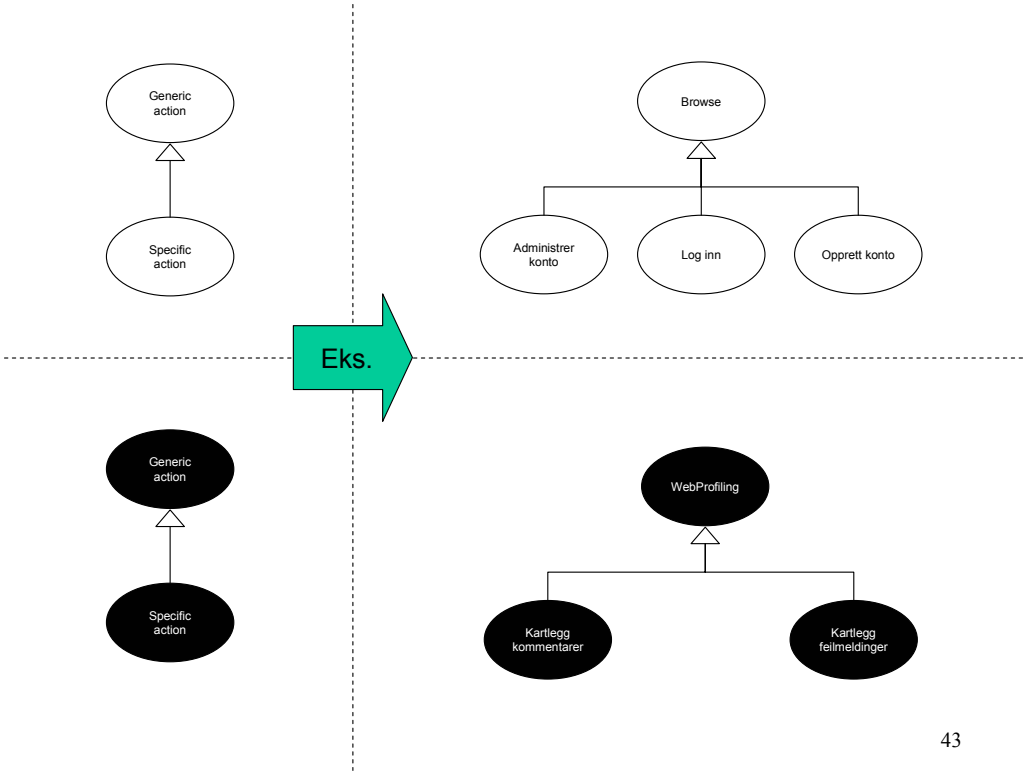
40


Aktør
Eksternt system
Bruker
Cracker
Kunde
Site administrator
Web administrator
Database administrator
Redaktør

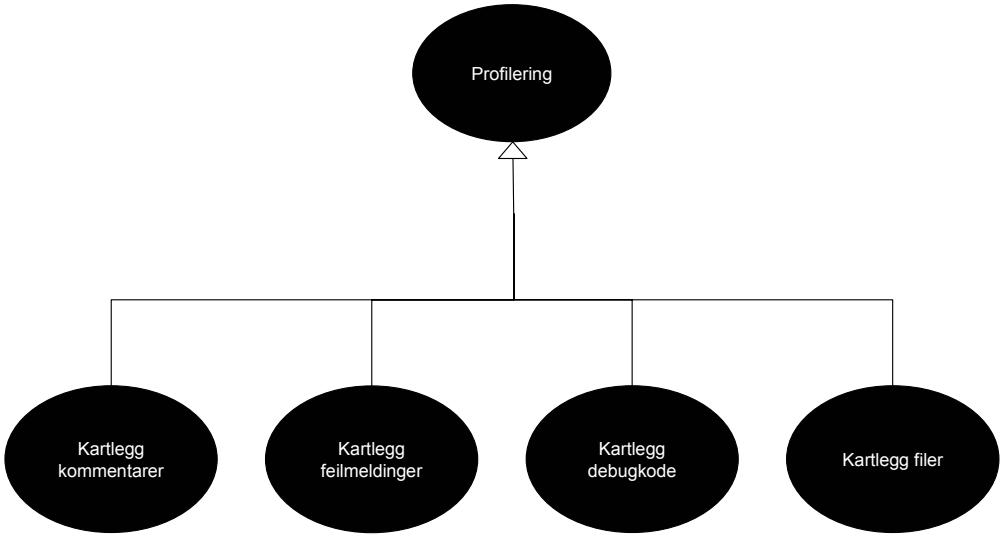
41


Bruker
Beistet (666)
Kunde
Admin
Hacker

42


Generic action
Specific action
Eks.
Browse
Administrer konto
Log inn
Opprett konto
Generic action
Specific action
WebProfiling
Kartlegg kommentarer
Kartlegg feilmeldinger

43

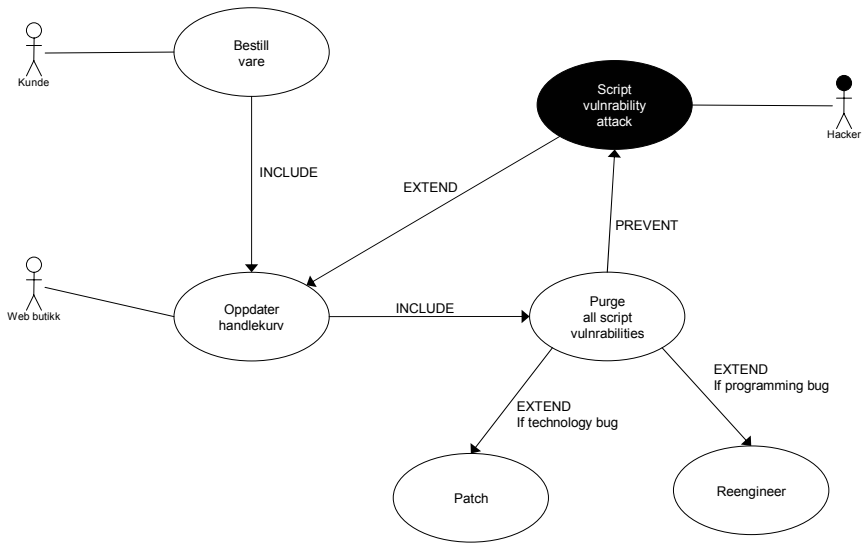

Profilering
Kartlegg kommentarer
Kartlegg feilmeldinger
Kartlegg debugkode
Kartlegg filer

44

VEDLEGG 2 – TIDLIGE EKSEMPLER PÅ AMM

SCRIPTWHACKING

Model:



Name:

ScriptWhacking

Alias:

Script Hacking; Hacking Scripts; Exploiting Bugs; Server side Script Vulnerabilities

Summary:

A Hacker finds a script that has bugs or vulnerabilities in the site structure and uses this to gain access to what would otherwise be inaccessible to him.

Author:

Gøran F. Breivik

Date:

25.11.2002

Basic path:

bp0: (step bp0-1) The Hacker has used "Webspionage" (Web pilfering) to get to know the most intimate details about your firm's website and its structure. (step bp0-2) This gives the hacker information about how to gain privileged access rights to your site by misusing a badly written cgi-script or a buggy asp-script. (step bp0-3) Getting this access the hacker can do all sorts of bad things like (step bp0-4) steal credit card information from your trusting customers, for instance, thereby ruining your reputation for a long time.

bp1: A less experienced hacker, often called a ScriptKiddie, (step bp1-1) uses a downloaded tool to scan your site for known script vulnerabilities, finds one or more vulnerabilities and (step bp1-2) follows a recipe on how to misuse this information. This can cause damage even greater than bp0, because you now probably have a complete novice rampaging through your web store causing all kinds of damage.

Alternative paths:

ap2: The hacker downloads a scanner tool from the Internet which tells him which vulnerable scripts your site uses (changes bp0-1).

ap3: The script kiddie shares his information with all members of his "hacker group" and several of the group's members launches a massive attack within a specifically planned timeframe (changes bp1-2).

ap4: The hacker does not compromise you nor your customers at your point of sale, but uses the information gathered at your site to compromise your clients elsewhere (changes bp0-4 and bp1-2).

Triggers:

tr1: TRUE

Preconditions:

pc1: The business logic is server-based.

Worst case threat (postcondition):

wc1: The Hacker compromises large parts of the customer base, thereby making your trust disappear completely, possibly taking your business with it. Additionally it will make a good scare story in the media.

Capture guarantee (postcondition):

cg1: No compromise inflicted, thereby business continues as usual.

Potential misuser profile:

Any person who have the skills to download a program and follow a recipe, either for fun or for profit.

Stakeholders and threats:

sh1: web-shop

- Reduced trust and thereby reduced turnover.
- Reduced customer base because of a few isolated incidents.

sh2: customer

Loss of privacy.

Economic fraud.

Scope:

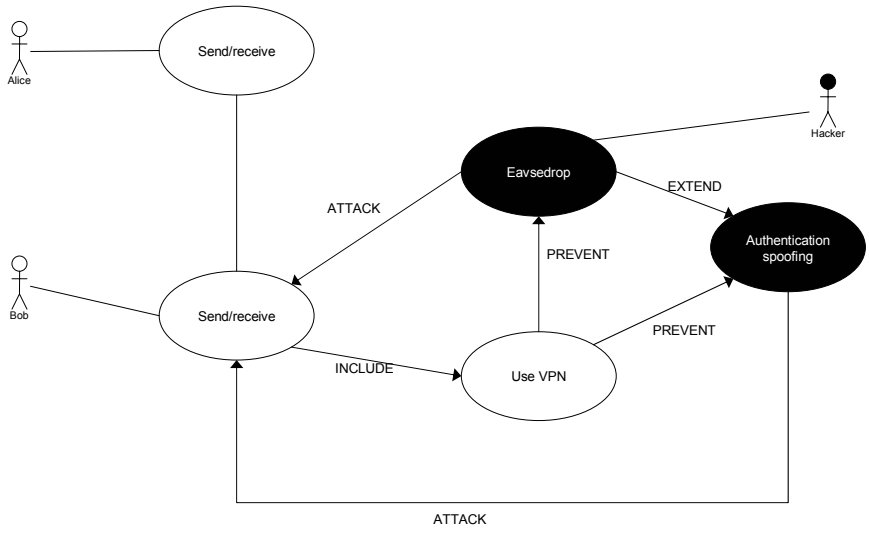
Web based business applications.

Abstraction level:

Mis-user goal / sub-function

WIRELESSNOTWORKING

Model:



Name:

Wireless NotWorking

Alias:

None

Summary:

The hacker gains access to a wireless network by eavesdropping communication between two branches of a large company.

Author:

Gøran F. Breivik

Date:

25.11.2002

Basic path:

bp0: (step bp0-1) The Hacker uses a standard wireless network card in his computer which he uses to scan for wireless networks. He uses a technique called wardriving to find an accesspoint and (step bp0-2) begins his eavesdropping as soon as he gets contact. From now on it is just a matter of time before he can (step bp0-3) get root (admin access), and controll over the entire network, (step bp0-4) using it for whatever purpose he wants.

Alternative paths:

ap2: Instead of gaining access to the network he just continues to listen to get marketable information (changes bp0-3).

Capture points:

cp1: ???

Extension points:

ep1: Extends "Crack encryption".

Triggers:

tr1: TRUE

Preconditions:

pc1: Some part of the business' network is wireless.

pc2: The physical layout of the network is available for scanning.

Assumptions:

as1:

Worst case threat (postcondition):

wc1: Deleting or in other ways destroying business critical data.
ws2:

Capture guarantee (postcondition):

cg1: Considerable less danger of being compromised.

Related business rules:

br1: Her er jeg usikker! Man kan få inntrykk av at det dreier seg om et annet nivå, men det er et faktum at denne typen angrep foregår og et annet faktum at det er vanskelig, for ikke å si unødvendig å generalisere angrepene ytterligere. Dermed sitter jeg igjen med at dette punktet kanskje ikke trenger å være like viktig hver gang.

Potential misuser profile:

Any person who have the skills to download a program, follow a recipe and the funds to buy a wireless network card. It is probably not an average hacker today, but this kind of hacking gains popularity with the technology involved.

Stakeholders and threats:

sh1: Network owner

- Reduced trust and thereby reduced turnover.

Scope:

All web-based business applications.

VEDLEGG 3 – FERDIG EKSEMPEL PÅ AMM

Under følger et eksempel på en fullstendig iterasjon fra utviklingen av konseptet AMM.

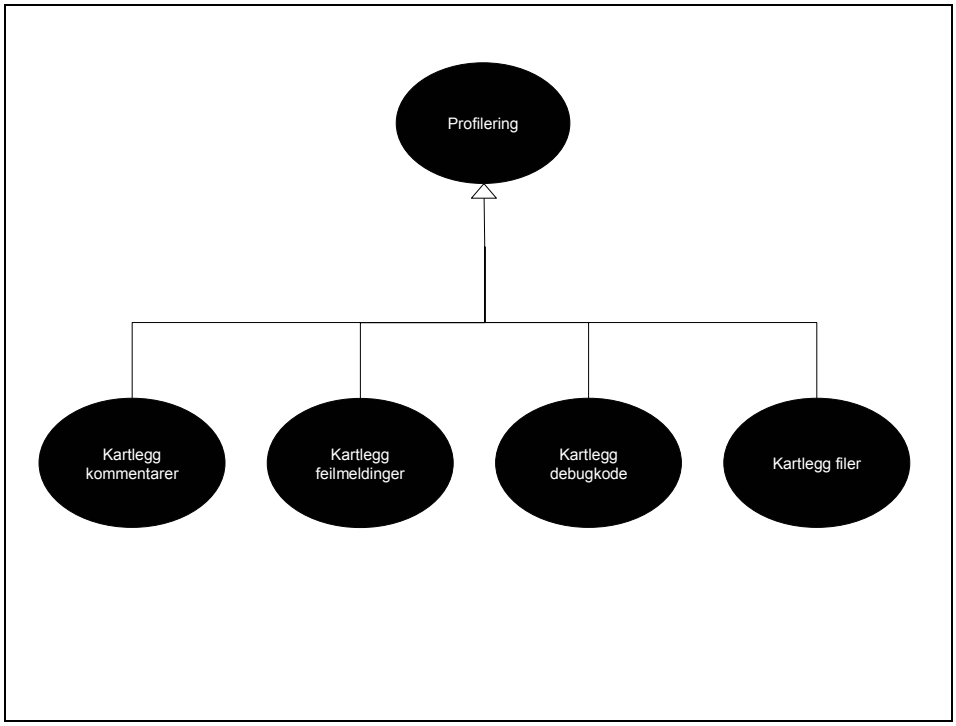
GRUNNLAG FOR AMM'ET PROFILERING

Profilering av potensielle ofre, eller "Footprinting" som det heter i hacker-slang, er første skanse for ethvert hacker-forsøk, enten det foregår manuelt eller med en nedlastet "vulnerability scanner". Det å unngå profilering er med andre ord første skanse for et hvert forsvar mot hackere. En cracker vil alltid, i den grad han er av det kyndige slaget, bruke en del tid på å finne ut mest mulig om sitt offer. Ved å danne seg en profil av offeret sitt, kan resten av jobben bli langt lettere. Som gjennomgangen av angrepskomponentene til OWASP viser (under), kan alvorlighetsgraden i et profileringsangrep avhenge av hvilken informasjon hackeren er ute etter og hvilken informasjon han finner. Hvis han f.eks. skulle finne informasjon som tilsier at det er enkelt å komme seg inn på serveren med administrator rettigheter, vil han mest sannsynlig benytte seg av denne umiddelbart. Et eksempel er "Debug Codes" (feilsøkingskoder) som kan sette serveren i feilsøkingsmodus, og dermed gjøre den mottakelig for utnytting gjennom å gi en administrators rettigheter til hackeren. Den siste setningen på OWASPs side om "Informational & Privacy Violations" illustrerer hovedgrunnen til å ta hensyn til denne klassen med sikkerhetshull.

> **"These attack components could be seen as the least technical, easiest to obtain and often the most useful to an attacker!"**

Angreps komponent	Problembeskrivelse	Misbrukstilfeller
Client-Side Comments (CSC)	” Comments are sometimes left in from the HTML development stage and can contain debug information, cookie structures, problems associated with development and even developer names, e-mails and phone numbers.”	For å bygge opp en profil av et potensielt offer, kan en hacker kartlegge alle kommentarer som er ”gjenglemt” i koden, som kan uttrykkes med misbrukstilfellet ”Kartlegg kommentarer”.
Error Codes (EC)	”Reading and understanding of error messages and/or error codes is the first step for successfully exploiting web application vulnerabilities!” ” If the web application does not proceed to catch exceptions, error messages and error codes they are sent to the webservers HTTP Response or other public output.”	”Kartlegg feilmeldinger” er dekkende for det misbrukstilfellet som beskrives i denne angrepskomponenten. Ved å kartlegge feilmeldinger får en hacker mer eller mindre nyttig informasjon om systemet feilmeldingene relateres til.”
Debugging Codes (DC)	”... An attacker will try various “obvious” debug constructs to see if the production sanitation process has been 100% successful. Any lapse can result in a devastating vulnerability due to the power that most debug constructs utilize.”	Hvis man skulle glemme å fjerne debug-mekanismer i koden, kan disse misbrukes. De forteller først og fremst en hacker mye om systemet. Dette kan uttrykkes gjennom misbrukstilfellet ”Kartlegg debug-konstruksjoner”.
File & Application Enumeration (FAE)	”File / Application Enumeration is a common technique that is used to look for files or applications that may be exploitable or be useful in constructing an attack.”	Skulle noen glemme å fjerne gamle cgi-script (Garfinkel & Spafford 97), standardapplikasjoner eller lignende, vil en hacker finne ut dette ved å kartlegge filer på alle tenkelige måter. Misbrukstilfellet ”Kartlegg filer” beskriver denne prosessen.

Hele denne klassen med angrepskomponenter dreier seg om å samle mest mulig informasjon med det formål å få laget en profil av offeret for så i andre rekke å angripe. Alle komponentene anses som såpass like på et abstrakt nivå at de i fellesskap kan uttrykkes med misbrukstilfellet "**Profilering**". Dermed oppstår det et generaliseringshierarki med Profilering som rotnode og de fire angrepskomponentenes respektive misbrukstilfeller som løvnoder.

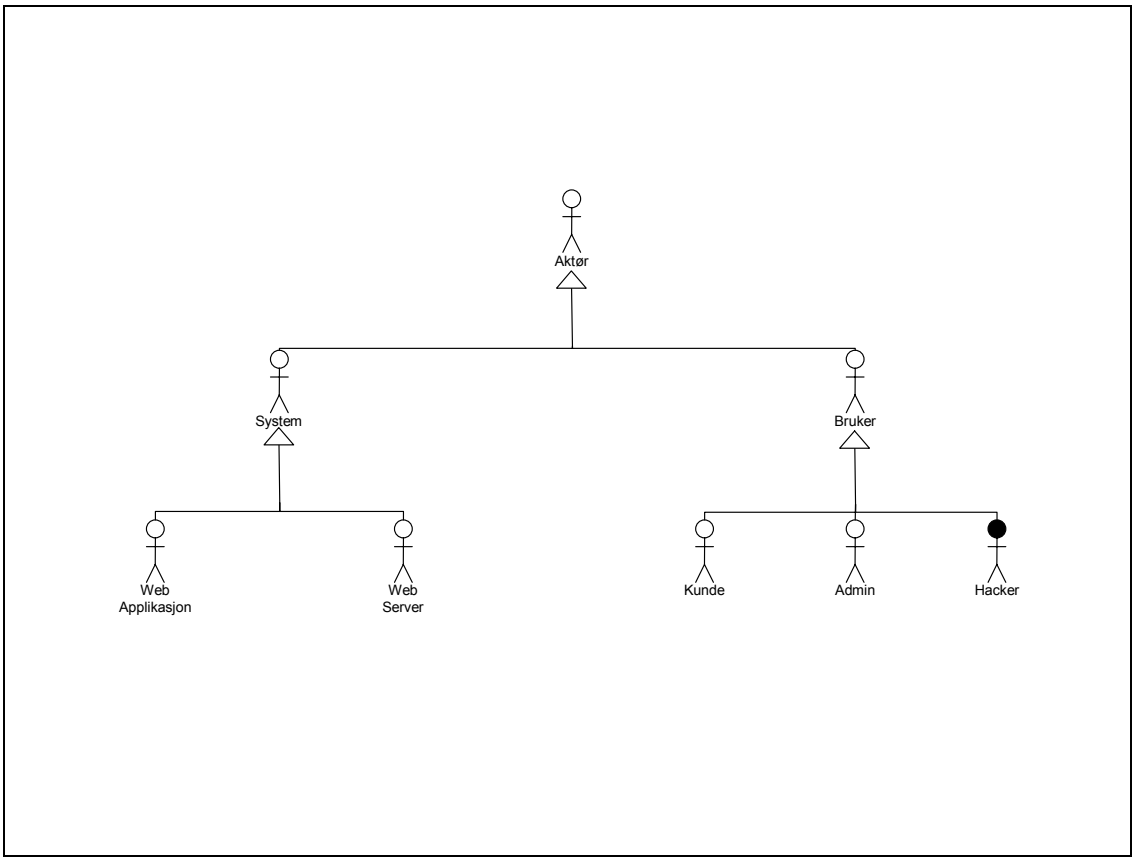


Figur 15 - Generaliseringshierarkiet Profilering

I neste omgang blir komponentene gjennomgått for å abstrahere preventive brukstilfeller som kan tilby en løsning på de problemene misbrukstilfellene utgjør. Sitatene i tabellen er hentet i all hovedsak fra angrepskomponentenes beskrivelse av mottiltak, men ikke nødvendigvis. Der det virker hensiktsmessig tas også sitater fra andre deler av komponentene i bruk.

Angreps komponent	Beskrivelse av mottiltak	Preventive brukstilfeller
Client-Side Comments (CSC)	”It is good practice to tie the filtering process to sound deployment methodologies so that only known good pages are ever released to production.”	For å unngå at en hacker skal få tilsynelatende uviktige fragmenter av informasjon fra koden, bør man ”Slette kommentarer”.
Error Codes (EC)	- ”Do not output error messages and exceptions in a production environment.” - ” If the web application does not proceed to catch exceptions, error messages and error codes they are sent to the webservers HTTP Response or other public output.”	Brukstilfellet ”Behandle feilkoder” forteller implisitt at det er nødvendig å fange opp feilkoder og ”exceptions” for å unngå at det blir fanget opp en hacker. Ved å fange opp alle exceptions, vil de verken være tilgjengelige eller utnyttbare.
Debugging Codes (DC)	- ”Test the site for debug commands.” - ”Sometimes a comment such as ’uncomment and set to 1 for debug mode’ will be left in just to make the attacker’s job easier. Finding debug elements is not easy, but once one is located it is usually tried across the entire web site by the potential hacker.”	Brukstilfellet ”Fjern debug konstruksjoner” bør være dekkende for å unngå sikkerhetshull tilknyttet til denne typen konstruksjoner.
File & Application Enumeration (FAE)	”Remove all sample files from your web server. Ensure that any unwanted or unused files are removed.”	Ved å fjerne alle filer som ikke er i aktiv bruk, slik som backup-filer, og script som ikke lenger er i bruk, vil man unngå en mengde utnyttbare hull. Brukstilfellet ”Slett ubrukte filer” er ment å illustrere dette.

Så langt er både misbruks- og preventive brukstilfeller definert for klassen ”Informational”. Et tredje nødvendig element i et Misuse Case diagram er aktørene. Ikke alle tenkelige aktører i en webapplikasjon, men mer eller mindre generiske aktører som kan brukes til å modellere roller innenfor konteksten; generiske angrep mot en generisk webapplikasjon. Følgende generaliseringshierarki over roller vil brukes som utgangspunkt i det videre:

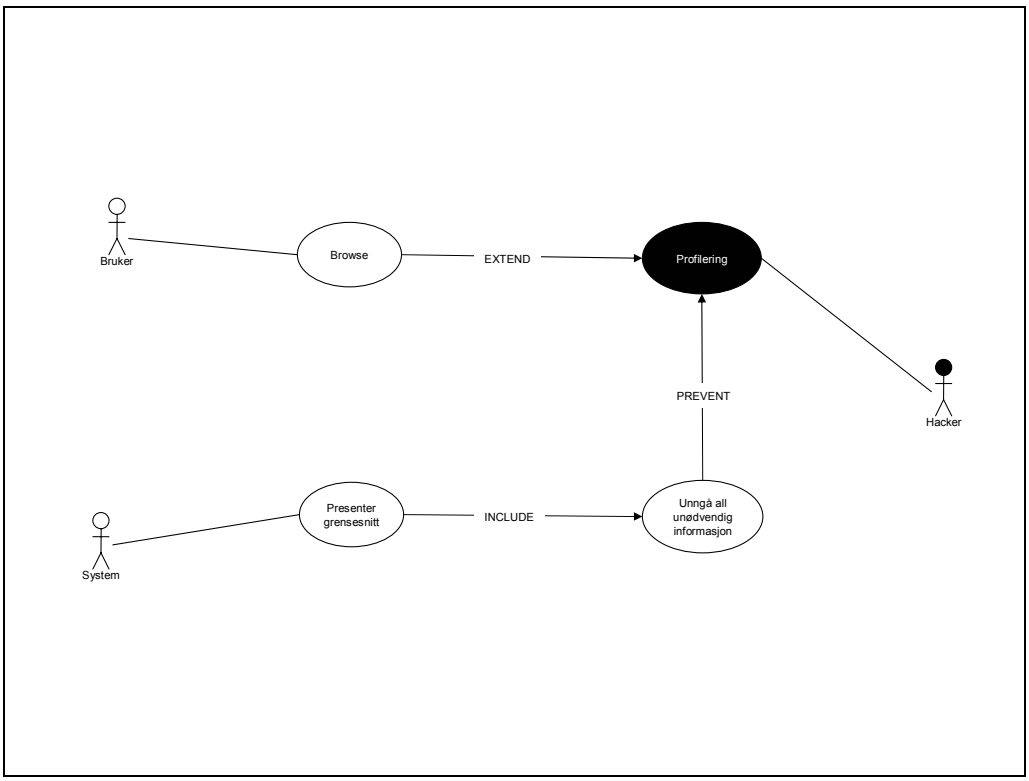


Figur 16 - Generaliseringshierarki for aktør-roller

Dette hierarkiet er tatt med her hovedsaklig av to grunner, det første er for å understreke at en aktør er en abstrakt rolle. Som hierarkiet viser kan en aktør både være av typene ”Bruker” som er en person og av typen ”System” som er et dataprogram eller en kombinasjon av slike. Den andre grunnen til dette hierarkiet er å vise at en ”Bruker” både kan være en faktisk ”Kunde” og/eller en ”Hacker”.

PROFILERING

Modell:



Name: Profilering

Alias: Footprinting; Web-Pilfering; Webspionage

Summary:

En hacker vil alltid gjøre mye for å finne ut mest mulig om sitt offer. Ved å foreta en eller flere av handlingene beskrevet som ”Extension Points” (EP?, 5 stk under eget punkt), vil han oppnå alt fra å få små biter med informasjon om systemet og dets utviklere, eiere, el.l. Hvis det er direkte utnyttbar informasjon (jfr. EP5), som f.eks. at det lar seg gjøre å kjøre systemet i feilsøkingsmodus (jfr. EP2), vil hackeren oppnå administrator rettigheter på serveren og dermed kunne gjøre hva som helst. Et annet eksempel som kan ha tilsvarende følger er EP4. EP1 og 3 derimot, er mindre kritiske, men allikevel kritiske nok til at det f.eks. kan foretas et ”Social Engineering”-angrep mot bedriften, eller at hackeren kan finne ut så mye om systemet at det lar seg gjøre å finne en bakdør/hull inn i systemet.

Author: Gøran F. Breivik

Date: 29.11.2001

Preconditions: PC1: Systemet er basert på internett-teknologi.

Scope: Alle web-baserte applikasjoner.

Basic path:

BP0: En hacker begynner sin søken etter kunnskap om sitt offer ved å ”spille” kunde og går inn på startsiden til systemet (step bp0-1). Så fort hackeren kommer inn på startsiden begynner han å lete etter interessant informasjon, både i kildekoden og i den informasjonen som presenteres (step bp0-2). Ved å benytte seg av kommentarer i kildekoden velger han å foreta et ’social engineering’-angrep (step bp0-3).

BP1: En hacker scanner tjeneren etter filer han vet er mulig å utnytte fordi de har et kjent sikkerhetshull eller bakdør (step bp1-1). Når han finner en slik fil utnytter han denne umiddelbart (step bp1-2).

Extension points:

EP1: Kartlegg feilmeldinger

Ved å fremprovosere feilmeldinger av alle tenkelige og utenkelige typer kan en hacker få masse informasjon om hvordan en webapplikasjon er bygget opp. Slik kartlegging av feilmeldinger kan fortelle om kvaliteten på kildekoden, om filstrukturer, teknologi, og så videre All informasjonen hackeren får nyttig i planleggingen av et angrep.

EP2: Feilsøkingsmodus

Det å sette et system i feilsøkingsmodus innebærer at man også får administrators rettigheter på systemet. Skulle et system inneholde slike muligheter i operativ sammenheng, åpnes det for muligheten for misbruk og innbrudd i systemet.

EP3: Kartlegg kommentarer

Fordi kildekoden til webapplikasjoner er såkalt åpen kildekode, vil alle kommentarer uansett tiltenkt formål være umiddelbart tilgjengelig for alle som har tilgang til websiden i utgangspunktet. Ved å kartlegge disse får en potensiell hacker masse informasjon han kanskje ikke burde ha, og som kan føre til at et angrep blir lettere å gjennomføre.

EP4: Kartlegg filer

Det finnes et utall mer eller mindre kjente filer som inneholder hull og bakdører. Disse kan man ofte laste ned verktøy for å søke etter. Ved å kartlegge filer av alle typer på en tjener, vil en hacker kunne få masse informasjon som både kan være direkte utnyttbar eller informativ nok til å bidra til et angrep.

EP5: Utnytt bakdør

Hackeren har funnet en bakdør til systemet gjennom EP2 eller -4 og utnytter denne.

Worst case threat (postcondition):

Mulighet for profilering fører til at en hacker for administrator rettigheter på systemet. Dette kan han oppnå hvis en av de fire misbrukstilfellene ikke blir tatt hensyn til. F.eks. kan resultatet av at man ikke fjerner kommentarer fra kildekoden føre til at hackeren får informasjon nok til å utføre et vellykket ’social engineering’-angrep, som i verste fall gir ham full tilgang til systemet. Alternativt finner han filer eller feilsøkingskonstruksjoner som gir ham direkte tilgang til systemet uten merarbeid. Slik sett kan man hevde at PP3 og 4 er viktigere enn de to andre.

Stakeholders and threats:

SH1: Systemeier

Tap av inntekt både gjennom nedetid, tap av kunder og tap av goodwill.

SH2: Kunde

Brudd på personvernet og misbruk av personlige opplysninger.

Prevention points:

PP1: Slett kommentarer

PP2: Sikre feilmeldinger

PP3: Fjern feilsøkingskode

PP4: Slett ubrukte filer

VEDLEGG 4 – INTERVJUGUIDEN

Personalia og bakgrunn

Personalia:

Navn

Alder

Bakgrunn:

Stilling

Relevant utdanning

Har du erfaring med eller kjennskap til en eller flere av følgende temaer?

System modellering?

UML generelt og/eller Use Case modellering spesielt?

”Patterns” (”design patterns” eller lignende)?

Datasikkerhet?

Applikasjonssikkerhet?

JA: Hva legger du i begrepet applikasjonssikkerhet?

NEI: Hva forstår du med begrepet applikasjonssikkerhet?

Sikkerhetskrav?

Å beskrive sikkerhetsangrep?

Har du vært involvert i kravarbeid, eller har du på annen måte kjennskap til kravarbeid?

Hvis ja:

Har du brukt Use Case modellering i forbindelse med kravarbeid?

Har du vært med på spesifisering av krav til sikkerhet / sikkerhetskrav?

Hvis ja:

Hvordan ble sikkerhetskrav spesifisert? Ved å beskrive korrekt bruk av applikasjonen eller mulig misbruk av applikasjonen?

Hvordan mener du sikkerhetskrav burde spesifiseres? Ved å beskrive korrekt bruk av applikasjonen eller ved å beskrive mulig misbruk av applikasjonen?

Kan du gi en kort begrunnelse?

Hvis nei:

Hvordan mener du sikkerhetskrav bør spesifiseres? Ved å beskrive korrekt bruk av applikasjonen eller ved å beskrive mulig misbruk av applikasjonen?

Kan du gi en kort begrunnelse?

1. Forståelse for sikkerhetsangrep blant kravarbeidere.

Er etter din mening forståelse for sikkerhetsangrep viktig eller uviktig del av kravarbeidet?

Hvilke av de presenterte forslagene til fremstilling av sikkerhetsangrep virker lettest å forstå?

Forslag 1 eller forslag 2?

Hva mener du gjør det forslaget lettere å forstå enn det andre?

Begrunnelse

Mener du noen av fremstillingsmetodene kompliserer/forenkler forståelsen for sikkerhetsangrep blant kravarbeidere?

Forklaring

Savner du noe som kunne gjort sikkerhetsangrep mer forståelig, og i så fall hva?

Eksempelvis:

Burde forklaringene vært mer eller mindre detaljert?

Burde det vært lagt større eller mindre vekt på grafisk fremstilling?

Burde det vært flere eller færre eksempler?

2. Kommunikasjon av sikkerhetsangrep blant kravarbeidere.

(Med kommunikasjon menes her formidling av forståelse mellom kravarbeidere.)

Er etter din mening kommunikasjon om sikkerhetsangrep viktig eller uviktig for kartlegging av sikkerhetskrav?

Hvorfor?

Hvilken av de to fremstillingene gjør kommunikasjon om sikkerhetsangrep enklest?

Begrunnelse

Kan du nevne hva ved det andre forslaget som gjør kommunikasjon vanskeligere?

Vil hvem du skal kommunisere med ha noen innvirkning på hvilket forslag du ville valgt?

Savner du noe i forslagene som kunne gjort kommunikasjon om sikkerhetsangrep enklere?

3. Enighet om sikkerhetskrav.

(Med enighet menes her enighet mellom oppdragsgiver og oppdragstaker. Spesifikt enighet om hva et krav innebærer og dokumentasjonen av denne enigheten (kontrakt).)

Hvor viktig er etter din mening enighet om krav for kvaliteten på en kravspesifikasjon? (Uviktig | Litt viktig | Ganske viktig | Veldig viktig)

Stiller det seg annerledes med sikkerhetskrav? Hvor viktig er enighet om sikkerhetskrav for kvaliteten på en kravspesifikasjon? (Uviktig | Litt viktig | Ganske viktig | Veldig viktig)

Hvilke av forslagene mener du vil gjøre det lettest å komme til enighet om sikkerhetskrav?

Begrunnelse

Hva mener du er grunnen til at det andre forslaget gjør det vanskelig å komme til enighet om sikkerhetskrav?

Savner du noe i fremstillingen som kunne gjort prosessen med å komme til enighet om sikkerhetskrav enklere, og i så fall hva?

VEDLEGG 5 – EKSEMPLER BRUKT I INTERVJUENE

FORSLAG NR. 1 (ANGREPSKOMPONENT HENTET FRA OWASP)

Overordnet forklaring av beskrivelsesmetoden:

Klasse

Én klasse innholder pr. i dag mellom 2 og 7 angrepskomponenter.

Komponent

Navngir en angrepskomponent.

Beskrivelse

En kort overordnet beskrivelse av sikkerhetsangrepet.

Analyse

Inngående beskrivelse av problemer knyttet til sikkerhetsangrepet.

Eksempler

Selvforklarende.

Mottiltak

Korte beskrivelser av hva som kan gjøres for å forhindre sikkerhetsangrepet.

Klasse

Innputt validering

Komponent

Direkte injeksjon av SQL kommando

Beskrivelse

Direkte injeksjon av SQL kommando er en teknikk hvor en angriper endrer eksisterende, eller konstruerer SQL kommandoer får å få utilsiktet tilgang til data eller eventuelt til å eksekvere system kommandoer på vertsmaskinen. Dette angrepet er avhengig av å utnytte ikke-eksisterende eller dårlig designede valideringsrutiner.

Analyse

Databaser er fundamentale komponenter i en hver webbasert applikasjon. De muliggjør lagring av data, slik som sesjonsinformasjon om en bruker, kundeopplysninger, instillinger og innholdselementer. Webapplikasjoner samarbeider med databaser med hjelp av ”Structured Qeury Language” (SQL) for å konstruere websider med spesialtilpassede data for hver enkelt bruker.

Dette oppnås ved at applikasjonen tar i mot brukerinnputt kombinert med statiske parametere for å bygge en SQL-spørring. Denne spørringen brukes for å hente ut korrekte data fra databasen, som applikasjonen deretter presenterer for brukeren.

Sikkerhetsmodellen som brukes av mange webapplikasjoner antar at en SQL spørring er tiltrodd kommando. Det betyr at SQL spørringer har muligheten til å omgå tilgangsrutiner, og dermed forbigå standard autentiserings- og autoriseringskontroller. I noen tilfeller kan SQL spørringer gi tilgang til vertssystemets operativsystemkommandoer.

Potensiell utnyttelse

Applikasjoner som ikke validerer og/eller saniterer brukerinnputt kan potensielt bli utnyttet på et utall måter:

- Forandre SQL-verdier:
- ”Concatenate” SQL-utsagn
- Legge funksjons- og/eller prosedyrekall til et utsagn
- “Typecaste” og ”concatenate” uthentede data

Eksempler følger på neste side:

Eksempel 1 – Forandre SQL verdier

Original Mysql-spørring:

```
UPDATE usertable SET pwd='$INPUT[pwd]' WHERE uid='$INPUT[uid]';
```

Deformert ”http request”:

```
http://www.none.to/script?pwd=ngomo&uid=1'+or+uid+like'%25admin%25';--%00
```

Eksempel 2 – Kombinere SQL-utsagn

Original PostgreSql-spørring:

```
SELECT id,name FROM products WHERE id LIKE '%$INPUT[prod]%';
```

Deformert ”http request”:

```
http://www.none.to/script?0';insert+into+pg_shadow+usename+values+('hoschi');--
```

Eksempel 3 – Legge funksjonskall og/eller lagrede prosedyrer til et utsagn

Original MSSQL-spørring:

```
SELECT id,name FROM products WHERE id LIKE '%$INPUT[prod]%';
```

Deformert ”http request”:

```
http://www.none.to/script?0';EXEC+master..xp_cmdshell(cmd.exe+/c);--
```

Eksempel 4 – “Typecast” og kombiner uthentede data

Original DB2-spørring:

```
SELECT id,t_nr,x_nr,i_name,last_update,size FROM p_table WHERE size =
'$INPUT[size]';
```

Deformed ”http request”:

```
http://www.none.to/script?size=0'+union+select+'1','1','1',concat(uname||'-
'||passwd)+as+i_name+'1'+'1'+from+usertable+where+uname+like+'25
```

Nå følger en kort oppsummering av problemer med populære databasesystemer:

MySQL

Støtter 'INTO OUTFILE'

Kjører ofte som ”rot”

De færreste moduler og biblioteker støtter sammensatte utsagn

Oracle

Subselects er mulig

UNION er mulig

Kommer med mange lagrede prosedyrer som standard (utf_file!)

Sammensatte utsagn ikke tillatt

DB2

Subselects er mulig

UNION er mulig

Lagrede prosedyrer

Sammensatte utsagn er ikke tillatt

Postgres

Støtter COPY (i superbruker-modus)

Subselects er mulig

UNION er mulig

Lagrede prosedyrer

Sammensatte utsagn er mulig

MS SQL

Subselects er mulig

UNION er mulig

Lagrede prosedyrer

Sammensatte utsagn er mulig.

Mange farlige lagrede prosedyrer som standard (xp_cmdshell, sp_adduser)

Mottiltak

- Tillat aldri usjekket brukerinnputt i databasespørringer.
- Valider og rens en hver brukervariabel sendt til databasen.
- Sjekk for forventet datatype.
- Sett brukerinnputt i anførselstegn.

FORSLAG NR. 2 (EGENUTVIKLET AMM)

Overordnet beskrivelse av fremstillingsmetoden.

NAVN

Følger ”pattern” ideen om navngiving for å forenkle kommunikasjon om det mønsteret beskriver.

ALIAS

Alternative navn problemet også kan være kjent under.

MOTIVASJON

Beskriver hva som fører til problemet, eller hvordan det oppstår.

PROBLEM

En kort beskrivelse av problemet.

LØSNING

En kort beskrivelse av overordnete løsninger.

EKSEMPEL

Selvforklarende.

RELATERTE MØNSTRE

Peker til andre mønstre som kan ha relevans for dette.

NAVN

Innputtmanipulering

ALIAS

ScriptInadequacies; InputValidationAttacks

MOTIVASJON

En hver applikasjon må ha en eller annen form for innputt for å kunne utføre en oppgave på grunnlag av brukerens ønsker og behov. Enhver form for innputt kan manipuleres. Den kan manipuleres innenfor de rammene utviklerens intensjon tilsier, eller alternativt på en måte som utvikleren aldri har tenkt på. Webapplikasjoner er komplekse informasjonssystemer som tilbyr mange måter å formatere innputt på; ASCII, Unicode, URL-encoding, og så videre

PROBLEM

I en webapplikasjon kan innputt manipuleres på mange måter og på mange nivåer. Dette fører til en mengde potensielle sikkerhetshull.

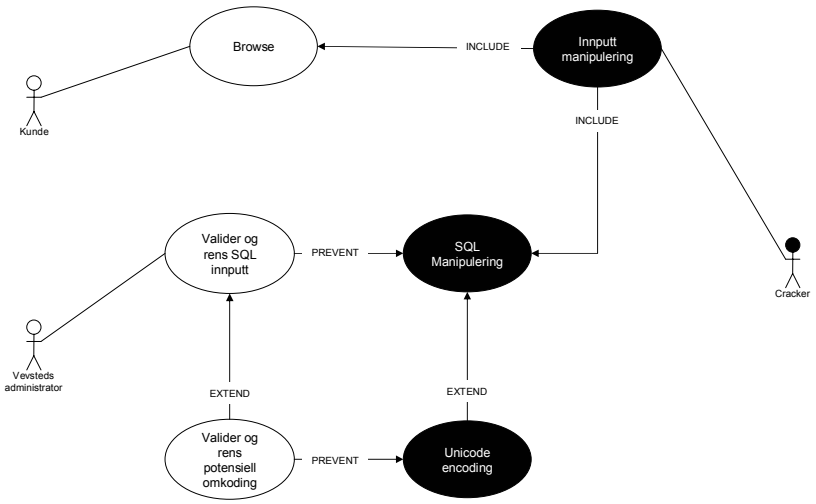
LØSNING

All innputt må valideres, sjekkes og renses på streng, karakter og byte nivå for at kun lovlig innputt skal tillates.

Vær spesielt forsiktig med hva som sendes videre til f.eks. databaser, filsystemet og andre applikasjoner.

EKSEMPEL

Under følger et eksempel som viser hvordan en sql kan manipuleres til å oppnå uautorisert tilgang til en database: (Se neste side)



Forklaring:

En bruker logger seg på en webapplikasjon med følgende data:

BRUKERNAVN: JohnDoe

PASSORD: Sunshine

Når brukeren trykker på OK, vil SQL variabelen se ut som følger:

SELECT * FROM users WHERE username='JohnDoe' AND password='Sunshine'

En cracker kan gjøre som følger:

BRUKERNAVN: '' OR username LIKE '%

PASSORD: ' OR password LIKE '%

Dermed følger:

SELECT * FROM users WHERE username='' OR username LIKE '%' AND password='' OR password LIKE '%'

Eventuelt kan crackeren gjøre manipuleringen noe vanskeligere å fange opp under validering. I stedet for å skrive prosenttegnet i ASCII, som kanskje ikke er tillatt, kan han velge en alternativ omkoding som f.eks. URL omkoding. Dermed skriver han "... username='' OR username LIKE '%20'..." som oversatt blir det samme som over.

Hvis ikke innputt valideres for lovlige verdier og renser vekk ulovlige verdier – sql konstruksjoner i dette tilfelle - vil manipulasjonen i eksempelet føre til at crackeren får en utskrift av hele brukertabellen med passord og det hele. Er da passordene lagret uten kryptering av noe slag, får crackeren tilgang til et verdifullt våpen.

(Eksempelet er fritt etter: www.silksoft.co.za og owasp.org)

RELATERTE MØNSTRE: "Profilering"

VEDLEGG 6 – TRANSKRIPSJONSTABELL

Respondent U1:	
Mann 31 år. 8 års erfaring som utvikler. Jobber nå i en stor, norsk bank som seniorutvikler.	
Har du vært med å beskrive sikkerhetsangrep:	
Brukssituasjon Trusselanalyse	”I forbindelse med sikkerhetsanalyse så må en gjøre seg en del ”tanker” sammen med forretningssiden, som da er kunden for å si det sånn. Så må man sette seg ned å tenke på en del situasjoner, for å si det sånn, så det blir vel en form for sikkerhetsangrep en tenker på da. Hvordan kan et system misbrukes for eksempel.”
Risikoanalyse	”… så må man sette seg ned å se på hvor sannsynlig det er, hvor ille det er hvis det går galt og hvor mye penger koster det. Hvor mye renommé mister en på det og masse sånne ting.”
Abstraksjon Trusselanalyse Risikoanalyse	”Vi går ikke ned og beskriver et sikkerhetsangrep i detalj, hvordan det kan gjennomføres, eller hvordan det kan tenkes gjennomført. Men for eksempel hvordan et system kan tenkes brukt i svindelsammenheng. Hvis vi for eksempel glipper på autentisering for eksempel og slipper igjennom en person som ikke skulle vært inne i systemet og sånt. Hva galt kan den personen få gjort, og en del sånne situasjoner…”
Teknologi	”… eller hvis det er en bug i koden som viser kontolister, eller kontotransaksjoner for feil person. Hvilke konsekvenser har det for banken. En del sånne ting har jeg vært med å tenkt på.”
Bruk av Use Case i kravarbeid:	
Abstraksjon Brukssituasjon	”De Use Casene jeg har laget har aldri vært avanserte. Bare noen få skjermbilder ikke sant. Det er jo det du lager Use Case på egentlig. Ett skjermbilde er ett Use Case stort sett. Kanskje du har et svarbilde eller noe sånt i tillegg.”
Har du vært med på å spesifisere sikkerhetskrav?	
	”Ja”
Hvordan beskrives sikkerhetskrav?	
Rollefordeling Brukssituasjon	”I kravspesifikasjonen vår, så kommer jo det fra forprosjektet da som har tenkt noe på dette her. Den vil jo for så vidt beskrive korrekt bruk av applikasjonen. Så i forbindelse med sikkerhetsanalysen så må en ta stilling til en del, hva skal jeg si… Misbruk enten ved at en får uautorisert tilgang, eller hva skal jeg si, ikke tiltenkt bruk av programmet. Om det er mulig og en del sånne ting.” ”… men jeg er ikke den som setter opp kravet.”
Hvordan bør man beskrive sikkerhetskrav?	
Rollefordeling	”Det er min jobb som programmerer at det ikke er mulig å putte inn ugyldige data. Så det må spesifiseres, hva skal jeg si, hva som er ugyldige data, eller ugyldige kombinasjoner av data. Hvis du tenker rent sånn skjermbildeorientert programmering da. Det er jo typisk på webforms og – så er det jo på den måten du egentlig tenker.”
Spesifisere hva som er ugyldig eller hva som er gyldig?	
Teknologi	”Du kan egentlig ikke i alle tilfelle spesifisere hva som er gyldig, men du for eksempel si om det er numeriske felter eller alfanumeriske felter.”
Kravspekk Rollefordeling	”Det er min oppgave som programmerer å – Du er nødt til å spesifisere faktisk helt ned på, hvis du skal gjøre det skikkelig så må du spesifisere helt ned på enkeltkarakterer. Hvilke tegn som skal være gyldig … Men det kommer som et motkrav opp i fra min side egentlig hvilke tegn som du ikke

	kan akseptere som innputt.”
	”Det kan ikke komme som et krav fra forretningssiden eller fra kunden sin side som et krav til sikkerhet. Det kommer bare som et generelt krav som sier at applikasjonen må være sikker teknisk sett.”
Kommentar til forslag 1 (OWASP)	
	”Dette er den tradisjonelle akademikerens forsøk på å sette noe i båser.”
Videre:	
Kravkonflikter	”Det som hadde vært interessant. Når du begynner på den måten der så kan du tenke patterns og pattern languages. Ok, hvis du velger det ene så får du problemer med noe annet, det er ikke helt unaturlig”.
Deretter kommer respondenten inn på andre muligheter enn det forslaget tar opp og forsøker å gi inntrykk av at dette ikke er noe nytt for ham…	
Er forståelse for sikkerhetsangrep en viktig/uviktig del av kravarbeidet?	
Kravarbeid Trusseltype	”Det er en viktig del av kravarbeidet. Det trenger ikke være ondsinnete angrep, det kan være uhell som gjør at du viser data for ”naboen”.”
Risikoanalyse	”Det kan ødelegge veldig mye av renommeet ditt som bedrift hvis du ikke har god kvalitet på applikasjonen.”
Forståelse	”Så, ja absolutt det er en viktig del av forståelsen til en person som skal utarbeide en kravspesifikasjon. At forståelse av sikkerhet finnes.”
Hvilke av forslagene synes du virker lettest å forstå?	
	”Jeg likte best forslag 1.” (OWASP)
Kommentarer til forslag 2 (AMM)	
Forståelse Rollefordeling Brukssituasjonen Kommunikasjon Sluttbruker	”Måten den er skrevet på gjør det mulig for en, hva skal jeg si? Nå snakket vi om domeneekspert, men du har jo ulike domener som er involvert, men for kundesiden å forstå hvilke problemer som ligger her… Men også, hva skal jeg si, du har jo medarbeidere i et sikkerhetsteam som egentlig har til oppgave å prøve å være kritiske og stille kritiske spørsmål, ikke sant? De har nødvendigvis ikke så god teknisk IT kompetanse, men de er veldig opptatt av det der at du, som sagt, følger en best practice. Og da tror jeg at forslag nr. 2 kan være veldig greit å bruke. Med tanke med den erfaringen jeg har med å sitte og diskutere sikkerhetsproblemstillinger med revisorer.”
Rollefordeling Forståelse	”For meg så er forslag 1 rett på sak og for en utvikler så er det det som trengs, men skal en diskutere dette i et kravspesifikasjonsforum. En samling av folk som da er kjøpere av applikasjonen, folk som kommer inn som kvalitetssikrere og som leverandør, så tror jeg at nr. 2 er det som definitivt kan, kan favne bedre.”
Hva er det som gjør det bedre enn forslag 1 i en sånn sammenheng?	
Abstraksjon	”Jeg tror ikke dette her med SQL her gir så mye mening. Det å sitte å se på at ok her hiver vi på en fnutt her og med noen kjappe raske greier så klarer man å få til å kjøre en annen operasjon enn det vi ville gjøre. Du får evt. revisoren til å nikke å si ”ja, det er uheldig”, men han har ikke skjønt hva det dreier seg om. Og når du da kommer til den andre her og da du egentlig begynner å snakke om ting på et mer sånn overordnet nivå der du sier det at du har en kunde som sitter å browser sant, og så får du noe Innputtmanipulering som gjør at en hacker sant. Eller la oss kalle det en tredjeperson som her ikke har noen relasjon til den personen som egentlig er inne her. At han kan ta over om ikke nødvendigvis ta over sesjonen, men ta over informasjonen til den her stakkaren som trur han er trygg og trygg for dataene sine. Det gjør vel for så vidt han site administratoren også. Han går vel rundt å trur at han

	har data som er trygge."
Presentasjon Abstraksjon Gjenbruk	"Jeg ville vel kanskje ha kjørt litt mer kjøtt på beina på nr. 2 for å få med meg utvikleren også, men jeg tror nr. 2 er det beste. For en annen grunn til det er at en må ikke bli for spesifikk her for da er jeg redd for at, i alle fall hvis du ikke har en erfaren utvikler. Og med det mener jeg at du må ha jobbet med dette her i veldig lang tid. Det å kunne ta et konkret tilfelle og utvide det også tenke. Du produserer de samme feilene igjen og igjen altså, hvis ikke du passer deg. Ikke med de samme ti linjene med kode, men du bare gjør det på en annen måte. Hvis du er litt tankeløs her og der så har du plutselig introdusert problemstillinger du ikke tenkte på."
Programvaresikkerhet	"For mange år siden når jeg begynte. Så det hadde jeg ikke lært meg den gangen. Det der med å dø på en estetisk måte. La sesjonen dø på en estetisk måte hvis det oppstår en feil."
Kunne forslag 2 ha vært mer eller mindre detaljert?	
Abstraksjon	"Det som også er å si om dette her. Det er at du har jo lagt det opp. Du har jo tatt opp akkurat det som jeg nevnte. Du ser dette her som et pattern også. Ja, du relaterer det mot andre, for å si det sånn. Andre patterns og sånne ting. Det tror jeg er viktig fordi at, som jeg sa før. Det kan fort bli et valg mellom to onder når du skal gjøre ting."
Er det noe poeng å legge mer eller mindre vekt på grafisk fremstilling?	
Presentasjon Modell	"Jeg vil vel si at du faktisk er nødt til å ha begge deler fordi at noen mennesker tenker best med figurerer som de kan peke og tegne på etterpå, mens andre de liker å ha det veldig kort og konsist. Så jeg tror en konsis språkbruk uten alt for lange utredninger med figurer som fokuserer på kjernen i tingene er det som skal til. Og den notasjonen, hva skal jeg si, la oss kalle det den tekstlige fremstillingen og den billedlige fremstillingen bør standardiseres sånn at det blir enkelt å kommunisere. Så da er kanskje en UML syntaks og sånt kanskje grei å bruke. Du må ha begge deler (tekst og figur), det er du bare nødt til altså."
Hvor viktig er det å komme til enighet i et kravarbeidsteam?	
Enighet	"Det er veldig viktig."
Mener du at det henger sammen med formidling og forståelse?	
Kostnad Enighet Risikoanalyse	"Ja, for å si det sånn. Det som er problemstillingen her. Det er at ting koster. Og da må du gjøre en for og i mot avveining. Det som mangler i den her for å få en kunde til å akseptere at dette er en sikkerhetsrisiko som han må forholde seg til. Det er, la oss kalle det kostnaden ved det. Her står det, liksom sånn, hvordan du løser det og masse sånt. Men nederst, ikke sant, så må du liksom skrive på kostnad. Altså: Kostnad av det her: "Nedetid – null timer"; "Tap av renommé – JA", og på noen skriver du "Ja, veldig mye". Du må prøve å kvantifisere ting på en sånn måte for menigmannen da. For den som er kunde slik at de kan forstå hva problemstillingen går ut på. Overfor en kunde så er du nødt til å sette opp en eller annen form for cost-benefit, ofte for å få det med i en analyse altså. Så det mangler egentlig for å komme helt til enighet om akkurat det kravet her. Det gjelder begge."
Kommunikasjon Enighet	"Selv om du tror du klarer å kommunisere i gjennom og "Ja, nei, det er ikke noe problem". Så kommer du etterpå, så viser det seg at det er et problem. Jeg har vært borti sånne situasjoner også."
Rollefordeling Brukssituasjon Enighet Kommunikasjon	"Det er ikke utypisk at du sitter med en relativt senior person fra kjøpersiden, som er da en domeneekspert innenfor det området du lager applikasjonen ikke sant. Og det er en ressursperson som skal utnyttes av forretningssiden eller av kunden samtidig. Og det er rift om den personen sin

	tid. Jeg har prøvd å bruke min tid, når jeg har vært med i kravspesifikasjon at jeg har brukt min tid til å beskrive. Jeg har drømt opp alt jeg komme på av ubehageligheter og så har jeg skrevet de ned, og på noen har jeg bare sagt at; Ok Det er min jobb. Det fikser jeg. Jeg bare tenker på det når jeg skal kode. Og andre. Dette kommer til å koste penger ikke sant. Den vil forsinke prosjektet også videre ikke sant. Så du får en sånn for og i mot avveining her. Men det har ikke vært noe struktur over det. Et sett med e-mail, som jeg samler i et arkiv da egentlig. Så jeg sender ut noen tanker med en e-post og et par power point slides og så få en tilbakemelding fra (domeneekspert).”
Presentasjon Rollefordeling Abstraksjon	”Kanskje det jeg sier er at du må ha flere versjoner. Ja, det er vel egentlig det jeg sier.”
Respondent U2:	
Mann 52 år. 25 års erfaring som utvikler, jobber nå som sjefskonsulent i et Norsk konsulentfirma.	
Hva forstår du med begrepet applikasjonssikkerhet?	
	”Jeg vil kalle det design og god programmeringsskikk jeg altså”
Har du noe kjennskap til sikkerhetskrav?	
Kravarbeid	”Det som jeg tenker på. Sikkerhetskrav er jo mer sånn sikkerhetspolicy, tenker jeg da når du sier det. For det er klar at vi som er i sånn andre tilbuds-fase vi får en del sikkerhetskrav altså hvor det står at det skal være sånn og sånn. Og det kan være alle typer sikkerhet, så vi ser hvordan vi skal løse, men sånn rent assosiativt når du sier sikkerhetskrav så tenker jeg ofte på sikkerhetspolicies.”
Har du noe kjennskap med å beskrive sikkerhetsangrep?	
	”Når du sier sikkerhetsangrep da får jeg litt assosiasjoner mot Internet og hacking jeg da. Det er ofte det jeg tenker på med sikkerhetsangrep. Før i tida så var det ikke noe angrep. Da måtte du innenfor veggene da for å komme til systemet. Så du kan si at sikkerhetsangrep er et begrep som har kommet med Internet. Du kan liksom sitte ute og angripe noen ondskapsfullt.”
	”Vi tenker sånne standardløsninger med antivirus, planlegger patching og evt. standardkryptering. Men noe utover de standardtingene der tenker vi egentlig ikke på.”
Har du vært med på å spesifisere sikkerhetskrav?	
Rollefordeling	”Ja, vi gjør det (tar tak i sikkerhetskrav i kravarbeidet), men helt sånn spesifikt så har jeg vel faktisk aldri omformulert kundekrav til sikkerhets-krav. Jeg har aldri fått kravene. For veldig ofte så, vi får jo ofte en litt sånn overordnet kravspekk vi. Hvor vi utformer en sånn litt mer detaljert krav-spekk og design. Og i den overgangen så ligger ofte sikkerhetskravene nedfelt. En har lest mange og implementert mange, men å lage det selv har jeg faktisk ikke gjort.”
Har de sikkerhetskrav du har sett vært spesifisert via korrekt bruk eller misbruk?	
Kravarbeid (prosess) Rollefordeling	”Vel sikkerhet mot å kunne skrive noe feil. Altså det at du hindrer feil i en validering. Det går stort sett på datasikkerhet da, eller datakonsistens, som er en type sikkerhet. Men den andre går også på. Det er mest. Er altså tilgang til data. Hvem som skal ha tilgang til data, altså autorisasjon. Når det gjelder sånn Internet type da har vi hatt en del forespørsler når det gjelder på elekt-ronisk signatur og om det er noen som kan noe om det. Men når det gjelder innbruddssikkerhets mot hacking da. Altså den type sikkerhet da, det er altså noe som sjelden står i spekken. Det er ofte noe som overlates til den enkelte installasjons og operative avdeling. Du har kryptografering da som det evt. kreves. Det er også noe du bare slår av og på nesten. I en moderne server i dag.”

Presentasjon (Risikoanalyse)	”Og så er det alltid en diskusjon om, kan du bryte deg inn eller kan du ikke bryte deg inn. Og da har du da den vurderingen om hvor stor sjanse det er da.”
Introduksjon av eksempler:	
Forståelse for sikkerhetsangrep blant kravarbeidere:	
Kravarbeid (prosess) Abstraksjon Forståelse vs. Kunnskap	”Ja, det er viktig. Men ofte å gå ned på dette her det, så langt ned tenker man ofte ikke i kravarbeid. Du tenker ofte på det som en helt generell sak. Det skal ikke være mulig å bryte seg inn, type ting. Begge forslagene går på måter å bryte seg inn på. Men jeg visst ikke om de måtene der altså. Men jeg er ikke så veldig teknisk heller da.”
Hvilket av forslagene virker lettest å forstå?	
	”Jeg ser på dette her mer som et slags sikkerhetspattern nesten jeg (om forslag 2 – AMM). At det her er måter det kan gjøres på og som man må ta hensyn til i design. Og sånn sett så er dette her et av flere krav. Du har identifisert at dette her går an å gjøre og det må vi gjøre noe med.”
Rollefordeling Kravarbeid (prosess)	”Men jeg tror aldri du vil få den typen frem i et kravspekkteam altså.”
Nei, dette er jo på ingen måte vanlig å gjøre.	
Rollefordeling Kunnskap Kommunikasjon / Forståelse	”Nei, grunnen til at man ikke gjør det er at man vet ikke om det. Det er ikke alle som er hacker spesialister her. Vi har en som driver å kikker litt på hvordan det gjøres. Og ned på det her, det er det sjelden vi har tenkt på. Jeg tror ikke vi har tenkt på det jeg.”
Hvis du er i et møte med kunden som sier at han vil ha en sikker applikasjon. La oss si at du er interessert i å avdekke sikkerhetskrav. Hvilket av de to forslagene ville du ha starta med eller begynt med?	
Forståelse	”Å binde sql kommandoer rett til innputt, det. Det er dårlig design altså. Derfor så ville jeg ikke nevnt dette her sånn. Da ville jeg bare si at sånt gjør vi ikke.”
Se litt bort i fra innholdet og litt mer på formen. Hva tror du vil egne seg best i en sånn situasjon (uttrekning av krav i møte med kunden)?	
Brukssituasjon	”Da skulle jeg tro at type patterns som er sånn som jeg oppfatter det litt mer generelt. Og at det er det du liksom bruker i en spekk. Det her (forslag 1) er litt sånn konkret, liksom litt mer implementeringsrettet.”
Kravarbeid (prosess) Presentasjon	”Men jeg er litt usikker på hva du mener skjønner du. For du kommer med en løsning. En spekk er et krav, dette her er egentlig løsninger på krav.”
Målet er at også kunden skal kunne detaljere sikkerhetskravene sine mer.	
Kravarbeid (prosess) Abstraksjon Brukssituasjon	”Det tror jeg ikke du for til. Kunden er ikke på det nivået her. De er på et nivå at du kan få innputt validering, ja. Det skal de ha, men der er det stopp. Og så skal ikke applikasjonen din hackes. Så det er opp til deg å finne ut hvordan det er. Du klarer aldri å få dem til å si at du skal gå på den type, den type, eller gå så langt ned.”
	”Det går på seriøsiteten i firmaet ditt. Du sier at vi lager systemer som er innbruddssikre og da sørger vi for det og det og det og det.”
Kravarbeid (prosess) Abstraksjon	”Du får ikke 100% kravspekk. Du leverer ikke det du har i kravspekken i dag. I dag er kravspekken et utgangspunkt. I RUP så er kravspekken et utgangspunkt og så jobber du da frem, og så endrer du da kravspekken og så har du endringsrutiner. Det å gå helt ned. Du går ikke så langt ned i kravspekken.”
Så du synes begge forslagene går for dypt?	

Brukssituasjon	”De går for dypt, men både ja og nei. Det som jeg skulle ønske meg. Akkurat som jeg har den pattern boka mi for design. Om du kunne få en patternbeskrivelse. Noe a-la det her. Det hadde vært noe det. For da kan du si at sånn (viser frem en fiktiv bunke med patterns) er det vi designer. Det er standard så det gjør vi. Det savner jeg.”
Brukssituasjon Abstraksjon Kommunikasjon	”For når jeg bare så på de skrivene her så. Oi, det her er nyttig i en design, i en realiseringsfase. Og også kanskje som et underlag for et kravarbeid. Sette krav til det her (forslagene). Du beskriver kravene på et litt mer generelt plan når du har med kunden, men du implementerer det eller realiserer det med hjelp av noe sånt.”
Så da vil du si at ingen av disse forslagene egner seg?	
Brukssituasjon	”Ikke i en kravspekk med kunden, men i et internt team som skal realisere kravet. Men da tror jeg det at kravene går på det at det skal være innbruddssikkert. Ekstremt høy sikkerhet på login, og så har du løsningen der.”
Respondent S1:	
Mann 34 år. 17 års erfaring som programmerer. Driver nå eget firma med applikasjonssikkerhet som forretningsområde.	
Har du kjennskap til patterns?	
Rollefordeling	”Jeg har kjennskap til patterns i den forstand at jeg har funnet opp en del patterns selv, som jeg ikke visste noe om at noen hadde gitt navn til. Det er vel sånn de fleste som har programmert en stund opplever det tror jeg. Men jeg forstår hva som er hensikten med patterns.”
Hva legger du i begrepet applikasjonssikkerhet?	
	”Det er for å heve det over det sikkerhetsnivået som folk flest tenker på, og det er infrastruktur. Så oppå der hvor noe er spesialskrevet programvare. Logiske feil, det er typisk. Eller mangler, mangler på ett eller annet, som gjør at noe kan misbrukes.”
Har du noe forståelse av sikkerhetskrav?	
Kravarbeid (prosess) Forståelse… (for kravarbeid)	”Nei. Jeg vet at det finnes noen standarder for hvordan man skal formulere sånne ting, men det. Jeg er veldig praktisk anlagt altså så jeg hever meg over sånt.”
Har du kjennskap til kravarbeid?	
Kravarbeid (prosess) Rollefordeling	”Jeg har prøvd å lese kravspesifikasjoner i den hensikt å implementere noe i henhold til en kravspesifikasjon da, som programmerer, men jeg har ikke laget kravspesifikasjoner nei.”
Hvordan bør sikkerhetskrav beskrives?	
Kravarbeid (prosess)	”Hvis hensikten er å få folk som ikke har sikkerhetskunnskap til å gå med på kravene eller forstå hvorfor kravene må være der. Så tror jeg det er veldig bra å bruke sånne virkelighetseksempler for å presentere det. Men hvis det er for å formidle ting til folk som jobber med dette her til daglig så tror jeg det er bedre å bruke en terminologi som man er enig om, så slipper man å bruke alt for mye tid på det. Men terminologien finnes jo ikke fra før så. Og det er jo det OWASP har startet å prøve å gjøre noe med.”
Kommentarer til forståelse:	
Arkitektur	”Hvis kravspesifikasjonen skal skissere en struktur på systemet så vil deler av det kunne omhandle sikkerhet. For eksempel så synes jeg det er veldig smart å lage ett eller annet sikkerhetslag som har ansvaret for all innputt validering, og gjemmer hele innputten for de andre lagene. Det er noe som godt kunne vært med i en kravspesifikasjon antageligvis.”

Hvilke av de to forslagene synes du virker lettest å forstå?	
Presentasjon (form)	”Jeg synes jo dette her virker litt ryddigere da. Det vil si forslag nr. 2 som er ditt forslag. Jeg synes det virket mest oversiktlig det forslag nr. 2.”
Savner du noe som evt. kunne gjort det enda enklere å forstå?	
Presentasjon Abstraksjonsnivå Brukssituasjon	”Jeg synes ikke det kan gjøres noe klarere enn det her, men jeg har jo innsett at det er mange som allikevel ikke vil forstå da. Så hvis jeg savner noe så måtte det være en kobling til et praktisk eksempel hvor man kan prøve det ut, og det går jo ikke an å ha det på papir da.”
Presentasjon (konsekvenser)	”En sånn liste over forskjellige ting som en inntrenger kan gjøre hvis han for til et vellykket angrep. For eksempel i dette tilfellet da. Sql-manipulering. Dette kan brukes for å forandre informasjon som ligger i databasen, uten direkte tilgang til databasen. Det kan brukes til å hente ut informasjon som ligger i databasen, det kan brukes til ødelegge data og sånne ting.”
Så du ville brukt forslag 2 til å formidle dette her, men hva kunne vært gjort bedre?	
Presentasjon Abstraksjon	”Jeg ville nok ha krydret det litt med noen flere eksempler og hvis jeg skulle forklare det til noen så ville jeg ha hostet opp flere enn bare en sånn brukernavn og passord SQL kanskje.”
Så litt mer detaljer mener du hadde vært bra?	
Behov Forståelse	”Ja noe som gjør at de folka som ser det for første gang som ikke har tenkt så mye sånne tanker før og det gjelder jo egentlig alle firmaer i norge. De jeg har vært borti i alle fall. At de har litt mer ting å leke med i hodet for at brikkene skal falle på plass.”
Så kunne jeg tenke meg å få noen tanker rundt enighet.	
Brukssituasjon	”Ja, i og med at jeg synes dette (forslag 2) på en måte var litt klarere inndelt, så ville jeg vel brukt det i alle sammenhenger tror jeg.”
Respondent D1:	
Kvinne 47 år. Ingen IT bakgrunn, men har vært med på kravarbeid. Jobber som saksbehandler i det offentlige.	
Ble krav til sikkerhet diskutert i det kravarbeidet du var med på?	
	”Nei. Men vi har jo ikke bedt om det da.”
Men de nevnte det heller ikke?	
	”Det kan jeg ikke huske. Men vi hadde mye diskusjon med dem.”
	”Vi var jo veldig opptatt av. Hvis vi henter et dokument i fra vårt område og legger ut på Internet. Vi turte jo nesten ikke å gjøre det. Så vi snakket jo sånn indirekte om det da. Og da sa de at det var helt uproblematisk. For du kutter den strengen med en gang du har tatt det inn, altså den linken”
Er det noe som helst sted i applikasjonen hvor en bruker kan legge inn informasjon?	
	”Nei.”
Er det noe dere har diskutert?	
	”Ja. Men det vi diskuterte, det er ikke ut i fra et sikkerhetsmessig synspunkt, men fra et kapasitetsmessig synspunkt her.”
Var sikkerhet i det hele tatt diskutert i den sammenhengen?	
	”Nei.”
Det var ikke noe salgsargument fra deres side at de hadde sikkerhet i høysetet?	
Kravarbeid (prosess)	”Da er jeg nødt til å se etter, for det husker jeg ikke. (Blar i et dokument).

	Nei, ikke i denne, men de svarte jo på det vi ba om ikke sant. Men det kan jo hende at det er sikkert for det. Og vi spurte ikke om det heller.”
	”Jeg husker jo ikke alt vi snakket om for vi snakket jo i timevis. Men vi snakket om de notatene, brevene som vi skal legge ut. Og da snakket vi indirekte om det, men det kan jo hende at vi snakket om andre ting også.”
Brukte dere noen form for modellering av noe slag, når dere snakket om systemet?	
Kravarbeid (prosess) Kommunikasjon Forståelse	”Hva mener du? Jeg vet ikke hva det vil si. Vi laget en tekst og så laget vi strukturen i Word. Vi hadde ikke noe sånt visuelt da.”
Presentasjon (form / terminologi)	”Jeg så det for meg på en måte. Hvordan det skulle bli. Hvor du skulle klikke og sånt. Så vi skrev det med ord. Men vi hadde ikke noen sånne fine ord som du har, altså som modellering og sånt. Men vi fikk jo en del skryt for det at alle skjønte det.”
Hvis du skulle tatt tak i sikkerhet. Hvordan burde man formidlet det?	
Presentasjon (form / terminologi)	”Jeg tror jeg ville hatt litt tekst først som ikke var for vanskelig. Som var oversatt til menneskespråk og beskriver forskjellige begreper. Og kommer med en del problemstillinger. Og så ville jeg snakket med de som har skrevet det, fordi da kunne jeg spurt om ting jeg ikke forstår. Det må være tekst hvor de folkene som har skrevet det har skrevet det med tanke på folk som ikke skjønner et kvekk. For det er ikke poenget å imponere noen ikke sant.”
Resten av intervjuet er ødelagt pga. tekniske problemer.	
Respondent S2:	
Mann 42 år. 18 års erfaring som systemansvarlig, mye erfaring med sikkerhet.	
Har du noe kjennskap til applikasjonssikkerhet?	
	”I veldig stor grad så bruker jo vi standard operativsystemer og for så vidt standard programvare. Og driver i liten grad med egenutvikling av program-vare.”
	”Folk har blitt veldig mye flinkere til å lage applikasjoner de siste par årene, for å si det som det er. En skal ikke gå så veldig langt tilbake før en del sånne hensyn som en i dag er blitt flinkere til å ta. Programmering, typisk i c da ikke sant på Unix maskiner som det jeg har holdt på med. Hvor det har vært en rekke med feil som har vært utnyttbare for eksterne og for så vidt interne brukere. Som går på slumsete programmering, egentlig sånn buffer overflow type problemstillinger. Og da gjerne i Set UID programmer. Så jeg kjenner jo liksom til de feilene og en god del av det opprydningsarbeidet som stod på.”
	Dette intervjuet førte ikke med seg noe særlig nyttig informasjon utover det at dette med applikasjonssikkerhet er et viktig problem å ta tak i.
Respondent U3:	
Mann 39 år. Seniorkonsulent.	
Har du noe kjennskap til UML og Use Case modellering?	
	”Ja”
Jobber du med det til daglig?	
Holdninger	”Nå må jeg skuffe deg litt: Minst mulig!”
Kommentar til hvorfor Use Case ikke har noe for seg:	
Presentasjon (form)	”Aktuelt problem: Vi skal registrere en ny bruker. Registrer bruker, hva

	skjer? Jo. Brukeren registrerer seg inn i systemet og får en respons tilbake. Hvis jeg beskriver det om det er med bilde eller tekst eller gud vet hva, så hadde det ikke gitt meg så veldig mye mer altså.
Hva legger du i begrepet applikasjonssikkerhet?	
Forståelse	”Det kan være flere ting, men jeg har ikke et klart inntrykk av det begrepet der. Men du kan si det at kvaliteten på koden i applikasjonen er god. At ikke ting skjer som ikke skulle ha skjedd. At tilgangen til applikasjonen er regulert. For eksempel.”
Holdninger	”Er det buddhistene som har Nirvana? Programmererne de har applikasjonssikkerhet.”
Holdninger	”Du er jo bare ett tannhjul i maskineriet ikke sant. Charlie Chaplin gir en god beskrivelse av hvordan det er blitt som det det er. Fordi at det er ingen som har totalansvaret ikke sant. Du moduliserer alt mulig ikke sant. I ”design by contract” og sånt som det der. Du vet akkurat hva du skal levere, hva du får inn og hva resultatet skal bli. I en ideell verden så kan det godt hende at hver komponent er flott og sikker og alt det der, men summen av alt, det er der problemet ligger.”
Kjenner du begrepet sikkerhetskrav?	
	”Nei.”
Å ta hensyn til sikkerhet i krav kjenner du til det?	
Rollefordeling	”Ja, ja. Vi har jo en egen rolle i våre prosjekter. En sikkerhetsansvarlig i prosjektet, som tar for seg alle sikkerhetsrelaterte aktiviteter i prosjektet skal da liksom den personen ta ansvaret for. Å beskrive hvordan det er tatt hensyn til punkter som er beskrevet i basis sikkerhetskrav som er en lefse av dimensjoner i vår bedrift. Og i og med at vi er den typen bedrift som vi er så har vi vel en del sikkerhetskrav som går utenpå det mange andre baler med.”
Har du noe erfaring med å beskrive sikkerhetsangrep?	
Brukssituasjon	”Ja, det går ikke en dag uten at vi beskriver mulige sånne. Vi ser jo det at en sikker løsning bygget på de komponentene vi har i dag, det er faktisk umulig å oppnå. Så det er liksom sikker nok. Det er det vi streber etter.”
Du sier at dere bruker sikkerhetsangrep aktivt. Og beskrivelsen etter hva jeg har skjønt, er ikke annet enn muntlig. Det er ikke sånn at dere setter opp beskrivelsen formelt?	
Kommunikasjon	”Nei, det er ikke sånt vi jobber med som en oppgave som det står et prosjektnummer xyz og tolv timer denne uken og sånt altså. Det blir litt sånn nidkjærhet også fordi vi har en løsning ikke sant. Som vi har fått en del pepper for. Men vi ser jo at alternativene er ikke noe bedre så vidt vi kan se. Det er mye sånn oss mot dem, holdt jeg på å si. Men vi jobber jo i en bedrift hvor det er kniving mellom, av og til føles det sånn, forretningssiden og IT siden. De kommer med sine krav som av og til ikke henger på greip.”
Brukssituasjon	”Vi har revisjoner, i gåseøyne, av løsninger. Der man går i gjennom fra a til å. Men det skjer jo ikke veldig ofte og vi er for så vidt i en prosess nå der vi har kjørt liksom i gang aktiviteter med å analysere den nye løsningen vår. Applikasjonssikkerhet. Der er ordet applikasjonssikkerhet. Det var det vi snakket om i sted ja. Men det går altså på totalløsningen. Ikke på infrastrukturen. Ikke på operativsystemet, nettverk eller, men altså på implementeringen.”
Tenk deg at du er del av et kravarbeidsteam. Og blant alle kravarbeiderne så skal du forsøke å kommunisere rundt akkurat det problemet som er skissert her. Hadde du kunnet velge ett av alternativene som bedre enn det andre?	
Kommunikasjon	”Ja, altså jeg synes denne her var veldig bra jeg (forslag 1). Nei, jeg synes den er mer beskrivende akkurat for problemstillingen. Den beskriver bedre

	for meg hva som er problemet.”
Hvis du tenker på kravarbeidsteamet som helhet da?	
Abstraksjon	”Dette gir jo en form for et bilde da (forslag 2). Hvis man skal snakke med en eller annen hvem som helst, så ville han ha problemer med å skjønne hva du mente med dette her også (forslag 2).”
Er du enig i at hvilken av disse metodene du bruker vil være avhengig av hvilken bakgrunn du har?	
Rollefordeling	”Ja, det er jo enig i. Det sier seg selv fordi at jeg er helt sikker på at vår Use Case ekspert ville ha foretrukket forslag 2.”

VEDLEGG 7 – KODINGSTABELL

1.0 Bruksområde og brukssituasjon		
Underkat.:	**Sitat:**	**Resp. nr.:**
Analyse	**1.1** ”I forbindelse med sikkerhetsanalyse så må en gjøre seg en del ”tanker” sammen med forretningssiden, som da er kunden for å si det sånn.”	U1
	1.2 ”I kravspesifikasjonen vår, så kommer jo det fra forprosjektet da som har tenkt noe på dette her. Den vil jo for så vidt beskrive korrekt bruk av applikasjonen. Så i forbindelse med sikkerhetsanalysen så må en ta stilling til en del, hva skal jeg si… Misbruk enten ved at en får uautorisert tilgang, eller hva skal jeg si, ikke tiltenkt bruk av programmet. Om det er mulig og en del sånne ting.” ”… men jeg er ikke den som setter opp kravet.”	
	1.3 ”… du har jo medarbeidere i et sikkerhetsteam som egentlig har til oppgave å prøve å være kritiske og stille kritiske spørsmål, ikke sant? De har nødvendigvis ikke så god teknisk IT kompetanse, men de er veldig opptatt av det der at du, som sagt, følger en best practice. Og da tror jeg at forslag nr. 2 kan være veldig greit å bruke. Med tanke med den erfaringen jeg har med å sitte og diskutere sikkerhetsproblemstillinger med revisorer.”	
	1.4 ”For meg så er forslag 1 rett på sak og for en utvikler så er det det som trengs, men skal en diskutere dette i et kravspesifikasjonsforum. En samling av folk som da er kjøpere av applikasjonen, folk som kommer inn som kvalitetssikrere og som leverandør, så tror jeg at nr. 2 er det som definitivt kan, kan favne bedre.”	
	1.5 ”Ja, det går ikke en dag uten at vi beskriver mulige sånne. Vi ser jo det at en sikker løsning bygget på de komponentene vi har i dag, det er faktisk umulig å oppnå. ”	U3
Kravarbeid vs. design	1.6 ”Det er ikke utypisk at du sitter med en relativt senior person fra kjøpersiden, som er da en domeneekspert innenfor det området du lager applikasjonen ikke sant. Og det er en resursperson som skal utnyttes av forretningssiden eller av kunden samtidig. Og det er rift om den personen sin tid. Jeg har prøvd å bruke min tid. Når jeg har vært med i kravspesifikasjon, at jeg har brukt min tid til å beskrive. Jeg har drømt opp alt jeg komme på av ubehageligheter og så har jeg skrevet de ned, og på noen har jeg bare sagt at; Ok, det er min jobb, det fikser jeg. Jeg bare tenker på det når jeg skal kode. Og andre. Dette kommer til å koste penger ikke sant. Den vil forsinke prosjektet også videre ikke sant. Så du får en sånn for og i mot avveining her, men det har ikke vært noe struktur over det. Et sett med e-mail, som jeg samler i et arkiv da egentlig. Så jeg sender ut noen tanker med en e-post og et par power point slides og så få en tilbakemelding fra (domeneekspert).”	U1
	1.7 ”De Use Casene jeg har laget har aldri vært avanserte. Bare noen få skjermbilder ikke sant. Det er jo det du lager Use Case på egentlig. Ett skjermbilde er ett Use Case stort sett.”	
	1.8 ”Kunden er ikke på det nivået her. De er på et nivå at du kan få innputt validering, ja. Det skal de ha, men der er det stopp. Og så skal ikke applikasjonen din hackes. Så det er opp til deg å finne ut hvordan det er. Du klarer aldri å få dem til å si at du skal gå på den type, den type, eller gå så langt ned. Det går på seriøsiteten i firmaet ditt. Du sier at vi lager systemer som er innbruddssikre og da sørger vi for det og det og det og det.”	U2
	1.9 ”Om du kunne få en patternbeskrivelse. Noe a-la det her (AMM). Det hadde vært noe det. For da kan du si at sånn (viser frem en fiktiv bunke med patterns) er det vi designer. Det er standard, så det gjør vi. Det savner jeg.”	

	1.10 ”For når jeg bare så på de skrivene her så. Oi, det her er nyttig i en design, i en realiseringsfase. Og også kanskje som et underlag for et kravarbeid. Sette krav til det her (forslagene). Du beskriver kravene på et litt mer generelt plan når du har med kunden, men du implementerer det eller realiserer det med hjelp av noe sånt.”	
	1.11 ”Ikke i en kravspekk med kunden, men i et internt team som skal realisere kravet. Men da tror jeg det at kravene går på det at det skal være innbruddssikkert. Ekstremt høy sikkerhet på login, og så har du løsningen der.”	
	1.12 ”Da skulle jeg tro at type patterns som er sånn som jeg oppfatter det litt mer generelt. Og at det er det du liksom bruker i en spekk. Det her (forslag 1) er litt sånn konkret, liksom litt mer implementeringsrettet.”	
Test	1.13 ”Vi har revisjoner, i gåseøyne, av løsninger. Der man går i gjennom fra a til å. Men det skjer jo ikke veldig ofte og vi er for så vidt i en prosess nå der vi har kjørt liksom i gang aktiviteter med å analysere den nye løsningen vår.”	U3

2.0 Roller og ansvar		
Underkat.:	**Sitat:**	**Resp. nr.:**
test	2.1 ”I kravspesifikasjonen vår, så kommer jo det fra forprosjektet da som har tenkt noe på dette her. Den vil jo for så vidt beskrive korrekt bruk av applikasjonen. Så i forbindelse med sikkerhetsanalysen så må en ta stilling til en del, hva skal jeg si… Misbruk enten ved at en får uautorisert tilgang, eller hva skal jeg si, ikke tiltenkt bruk av programmet. Om det er mulig og en del sånne ting.” ”… men jeg er ikke den som setter opp kravet.”	U1
	2.2 ”Det er min jobb som programmerer at det ikke er mulig å putte inn ugyldige data. Så det må spesifiseres, hva skal jeg si, hva som er ugyldige data, eller ugyldige kombinasjoner av data. Hvis du tenker rent sånn skjermbildeorientert programmering da. Det er jo typisk på webforms og – så er det jo på den måten du egentlig tenker.”	
	2.3 ”Det er min oppgave som programmerer å – Du er nødt til å spesifisere faktisk helt ned på, hvis du skal gjøre det skikkelig så må du spesifisere helt ned på enkeltkarakterer. Hvilke tegn som skal være gyldig … Men det kommer som et motkrav opp i fra min side egentlig hvilke tegn som du ikke kan akseptere som innputt.”	
	2.4 ”Måten den er skrevet på gjør det mulig for en, hva skal jeg si? Nå snakket vi om domeneekspert, men du har jo ulike domener som er involvert, men for kundesiden å forstå hvilke problemer som ligger her… Men også, hva skal jeg si, du har jo medarbeidere i et sikkerhetsteam som egentlig har til oppgave å prøve å være kritiske og stille kritiske spørsmål, ikke sant? De har nødvendigvis ikke så god teknisk IT kompetanse, men de er veldig opptatt av det der at du, som sagt, følger en best practice. Og da tror jeg at forslag nr. 2 kan være veldig greit å bruke.”	
	2.5 ”For meg så er forslag 1 rett på sak og for en utvikler så er det det som trengs, men skal en diskutere dette i et kravspesifikasjonsforum. En samling av folk som da er kjøpere av applikasjonen, folk som kommer inn som kvalitetssikrere og som leverandør, så tror jeg at nr. 2 er det som definitivt kan, kan favne bedre.”	
	2.6 ”Kanskje det jeg sier er at du må ha flere versjoner. Ja, det er vel egentlig det jeg sier.”	
	2.7 ”Ja, vi gjør det (tar tak i sikkerhetskrav i kravarbeidet), men helt sånn spesifikt så har jeg vel faktisk aldri omformulert kundekrav til sikkerhetskrav.	U2

	Jeg har aldri fått kravene. For veldig ofte så, vi får jo ofte en litt sånn over-ordnet kravspekk vi. Hvor vi utformer en sånn litt mer detaljert kravspekk og design. Og i den overgangen så ligger ofte sikkerhetskravene nedfelt. En har lest mange og implementert mange, men å lage det selv har jeg faktisk ikke gjort."	
	2.8 "Men når det gjelder innbruddssikkerhets mot hacking da. Altså den type sikkerhet da, det er altså noe som sjelden står i spekken. Det er ofte noe som overlates til den enkelte installasjons og operative avdeling. Du har kryptografering da som det evt. kreves. Det er også noe du bare slår av og på nesten. I en moderne server i dag."	
	2.9 "Men jeg tror aldri du vil få den typen frem i et kravspekkteam altså. Nei, grunnen til at man ikke gjør det er at man vet ikke om det. Det er ikke alle som er hacker spesialister her. Vi har en som driver å kikker litt på hvordan det gjøres. Og ned på det her, det er det sjelden vi har tenkt på. Jeg tror ikke vi har tenkt på det jeg."	
	2.10 "Jeg har kjennskap til patterns i den forstand at jeg har funnet opp en del patterns selv, som jeg ikke visste noe om at noen hadde gitt navn til. Det er vel sånn de fleste som har programmert en stund opplever det tror jeg. Men jeg forstår hva som er hensikten med patterns."	
	2.11 "Nei. Jeg vet at det finnes noen standarder for hvordan man skal formulere sånne ting, men det. Jeg er veldig praktisk anlagt altså så jeg hever meg over sånt."	S1
	2.12 "Jeg har prøvd å lese kravspesifikasjoner i den hensikt å implementere noe i henhold til en kravspesifikasjon da, som programmerer, men jeg har ikke laget kravspesifikasjoner nei."	
	2.14 " Nei. Men vi har jo ikke bedt om det da."	
	2.15 "Jeg så det for meg på en måte. Hvordan det skulle bli. Hvor du skulle klikke og sånt. Så vi skrev det med ord. Men vi hadde ikke noen sånne fine ord som du har altså, som modellering og sånt."	D1
	2.16 "Ja, ja. Vi har jo en egen rolle i våre prosjekter. En sikkerhetsansvarlig i prosjektet, som tar for seg alle sikkerhetsrelaterte aktiviteter i prosjektet skal da liksom den personen ta ansvaret for."	
	2.17 "Nei, det er ikke sånt vi jobber med som en oppgave som det står et prosjektnummer xyz og tolv timer denne uken og sånt altså. Det blir litt sånn nidkjærhet også fordi vi har en løsning ikke sant. Som vi har fått en del pepper for. Men vi ser jo at alternativene er ikke noe bedre så vidt vi kan se. Det er mye sånn oss mot dem, holdt jeg på å si. Men vi jobber jo i en bedrift hvor det er kniving mellom, av og til føles det sånn, forretningssiden og IT siden. De kommer med sine krav som vi av og til ikke henger på greip."	U3
	2.18 "Ja, det er jo enig i. Det sier seg selv fordi at jeg er helt sikker på at vår Use Case ekspert ville ha foretrukket forslag 2."	
3.0 Presentasjon		
Underkat.:	**Sitat:**	**Resp. nr.:**
Abstraksjon	3.1 "Jeg ville vel kanskje ha kjørt litt mer kjøtt på beina på nr. 2 for å få med meg utvikleren også, men jeg tror nr. 2 er det beste. For en annen grunn til det er at en må ikke bli for spesifikk her for da er jeg redd for at, i alle fall hvis du ikke har en erfaren utvikler. Og med det mener jeg at du må ha jobbet med dette her i veldig lang tid. Det å kunne ta et konkret tilfelle og utvide det også tenke. Du produserer de samme feilene igjen og igjen altså, hvis ikke du passer deg. Ikke med de samme ti linjene med kode, men du bare gjør det på en annen måte. Hvis du er litt tankeløs her og der så har	U1

	du plutselig introdusert problemstillinger du ikke tenkte på."	
	3.3 "Kanskje det jeg sier er at du må ha flere versjoner. Ja, det er vel egentlig det jeg sier."	
	3.5 "Men jeg er litt usikker på hva du mener skjønner du. For du kommer med en løsning. En spekk er et krav, dette her er egentlig løsninger på krav."	
	3.6 "Jeg synes jo dette her virker litt ryddigere da. Det vil si forslag nr. 2 som er ditt forslag. Jeg synes det virket mest oversiktlig det forslag nr. 2."	
	3.7 "Jeg synes ikke det kan gjøres noe klarere enn det her, men jeg har jo innsett at det er mange som allikevel ikke vil forstå da. Så hvis jeg savner noe så måtte det være en kobling til et praktisk eksempel hvor man kan prøve det ut, og det går jo ikke an å ha det på papir da."	
	3.8 "En sånn liste over forskjellige ting som en inntrenger kan gjøre hvis han for til et vellykket angrep. For eksempel i dette tilfellet da. Sql-manipulering. Dette kan brukes for å forandre informasjon som ligger i databasen, uten direkte tilgang til databasen. Det kan brukes til å hente ut informasjon som ligger i databasen, det kan brukes til ødelegge data og sånne ting."	S1
	3.9 "Jeg ville nok ha krydret det litt med noen flere eksempler og hvis jeg skulle forklare det til noen så ville jeg ha hostet opp flere enn bare en sånn brukernavn, passord sql kanskje."	
Terminologi	3.4 "Jeg ser på dette her mer som et slags sikkerhetspattern nesten jeg (om forslag 2 – AMM). At det her er måter det kan gjøres på og som man må ta hensyn til i design. Og sånn sett så er dette her et av flere krav. Du har identifisert at dette her går an å gjøre og det må vi gjøre noe med."	U2
	3.10 "Jeg tror jeg ville hatt litt tekst først som ikke var for vanskelig. Som var oversatt til menneskespråk og beskriver forskjellige begreper. Og kommer med en del problemstillinger. Og så ville jeg snakket med de som har skrevet det, fordi da kunne jeg spurt om ting jeg ikke forstår. Det må være tekst hvor de folkene som har skrevet det har skrevet det med tanke på folk som ikke skjønner et kvekk. For det er ikke poenget å imponere noen ikke sant."	D1
	3.11 "Jeg så det for meg på en måte. Hvordan det skulle bli. Hvor du skulle klikke og sånt. Så vi skrev det med ord. Men vi hadde ikke noen sånne fine ord som du har, altså som modellering og sånt. Men vi fikk jo en del skryt for det at alle skjønte det."	
Notasjon	3.2 "Jeg vil vel si at du faktisk er nødt til å ha begge deler fordi at noen mennesker tenker best med figurerer som de kan peke og tegne på etterpå, mens andre de liker å ha det veldig kort og konsist. Så jeg tror en konsis språkbruk uten alt for lange utredninger med figurer som fokuserer på kjernen i tingene er det som skal til. Og den notasjonen, hva skal jeg si, la oss kalle det den tekstlige fremstillingen og den billedlige fremstillingen bør standardiseres sånn at det blir enkelt å kommunisere. Så da er kanskje en UML syntaks og sånt kanskje grei å bruke. Du må ha begge deler (tekst og figur), det er du bare nødt til altså."	U1

4.0 Holdninger		
Underkat.:	**Sitat:**	**Resp. nr.:**
Systemering	4.1 "De Use Casene jeg har laget har aldri vært avanserte. Bare noen få skjermbilder ikke sant. Det er jo det du lager Use Case på egentlig. Ett skjermbilde er ett Use Case stort sett."	U1
	4.2 "Nå må jeg skuffe deg litt: Minst mulig! (Om UC)"	U3

Krav	4.3 ”Jeg har prøvd å lese kravspesifikasjoner i den hensikt å implementere noe i henhold til en kravspesifikasjon da, som programmerer …”	S1
	4.4 ”Nei. Jeg vet at det finnes noen standarder for hvordan man skal formulere sånne ting, men det… Jeg er veldig praktisk anlagt altså, så jeg hever meg over sånt.”	
	4.5 ”Nei. Men vi har jo ikke bedt om det da.”	D1
	4.6 ”Vi var jo veldig opptatt av. Hvis vi henter et dokument i fra vårt område og legger ut på Internet. Vi turte jo nesten ikke å gjøre det. Så vi snakket jo sånn indirekte om det da. Og da sa de at det var helt uproblematisk.”	
Applikasjons-sikkerhet	4.7 ”Er det buddhistene som har Nirvana? Programmererne de har applikasjonssikkerhet.”	U3
	4.8 ”Du er jo bare ett tannhjul i maskineriet ikke sant. Charlie Chaplin gir en god beskrivelse av hvordan det er blitt som det det er. Fordi at det er ingen som har totalansvaret ikke sant. Du moduliserer alt mulig ikke sant. I ”design by contract” og sånt som det der. Du vet akkurat hva du skal levere, hva du får inn og hva resultatet skal bli. I en ideell verden så kan det godt hende at hver komponent er flott og sikker og alt det der, men summen av alt, det er der problemet ligger.”	

LITTERATURLISTE

Alexander, C., Ishikawa, S., et al. (1977). A Pattern Language Towns, Buildings, Construction. New York, Oxford University Press.

Alexander, I. (2002). Initial Industrial Experience of Misuse Cases in Trade-Off Analysis. IEEE Joint International Requirements Engineering Conference, Essen.

Al-Rawas, A. og Easterbrook, S. (1996). Communication Problems in Requirements Engineering: A Field Study. Proc. of the First Westminster Conference on Professional Awareness in Software Engineering, London.

Amoroso, E. G. (1994). Fundamentals of Computer Security Technology. Englewood Cliffs, N.J., Prentice Hall.

Aslam, T., Krsul, I., et al. (1996). Use of a Taxonomy of Security Faults. In 19th National Information Systems Security Conference Proceedings. Baltimore**:** 551-560.

Avison, D. E. og Fitzgerald, G. (1995). Information Systems Development : Methodologies, Techniques and Tools. London, McGraw-Hill.

Bellovin, S. (2001). "Computer Security - an End State." Communications of the ACM **44**(3): 131-132.

Bellovin, S. (2002). The State of Software Security. Florham Park, NJ, AT&T Labs Research.

Bishop, M. (1995). A Taxonomy of Unix System and Network Vulnerabilities, Department of Computer Science at the University of California at Davis.

Bishop, M. og Bailey, D. (1996). A Critical Analysis of Vulnerability Taxonomies, Department of Computer Science at the University of California at Davis.

Burstein, F. og Gregor, S. (1999). The Systems Development or Engineering Approach to Research in Information Systems: An Action Research Perspective. 10th Australasian Conference on Information Systems.

CERT/CC (2002). Cert/Cc Statistics 1988-2002 [Internet]. CERT Coordination Center, Carnegie Mellon University. Tilgjengelig på: http://www.cert.org/stats/cert_stats.html [20.11 2002].

Cockburn, A. (2001). Writing Effective Use Cases. Boston, Addison-Wesley.

Coplien, J. (1996). Software Patterns. New York, SIGS Books & Multimedia.

Fowler, M. (1997). Analysis Patterns : Reusable Object Models. Menlo Park, Calif., Addison-Wesley.

Frankfort-Nachmias, C. og Nachmias, D. (1996). Research Methods in the Social Sciences. London, Arnold.

Gamma, E., Helm, R., et al. (1995). Design Patterns : Elements of Reusable Object-Oriented Software. Reading, Mass., Addison-Wesley.

Garfinkel, S. og Spafford, G. (1997). Web Security & Commerce. Cambridge, O'Reilly.

Glass, R. L., Vessey, I., et al. (2002). "Research in Software Engineering: An Empirical Study." Information & Software Technology **forthcoming**.

Hellevik, O. (2000). Forskningsmetode I Sosiologi Og Statsvitenskap. Oslo, Universitetsforl.

Hoffer, J. A., George, J. F., et al. (1999). Modern Systems Analysis and Design. Reading, Mass., Addison-Wesley.

Howard, J. D. og Fischbeck, P. S. (1997). An Analysis of Security Incidents on the Internet 1989-1995. Carnegie Mellon University. Dept. of Engineering and Public Policy. Pittsburgh, PA, Carnegie Mellon University**:** xxviii, 292 p. :.

IEEE (1990). Ieee Standard Glossary of Software Engineering Terminology. Piscataway, NJ, The Institute of Electrical and Electronics Engineers, Inc.

ISO/IEC (1999a). 15408-1 Information Technology — Security Techniques — Evaluation Criteria for It Security — Part 1: Introduction and General Model, International Standardization Organisation / International Engineering Committee.

ISO/IEC (1999b). 15408-2 Information Technology — Security Techniques — Evaluation Criteria for It Security — Part 2: Security Functional Requirements, International Standardization Organisation / International Engineering Committee.

ISO/IEC (1999c). 15408-3 Information Technology — Security Techniques — Evaluation Criteria for It Security — Part 3: Security Assurance Requirements, International Standardization Organisation / International Engineering Committee.

Jacobson, I., Booch, G., et al. (1999). The Unified Software Development Process. Reading, Mass., Addison-Wesley.

Johnson, R. (1992). Documenting Frameworks Using Patterns. OOPSLA'92, Vancouver, BC, ACM SIGPLAN Notices.

Kotonya, G. og Sommerville, I. (1998). Requirements Engineering : Processes and Techniques. Chichester, Wiley.

Kvale, S. (2001). Det Kvalitative Forskningsintervju. Oslo, Gyldendal Akademisk.

Kvammen, O., Hoel, B. A., et al. (2002). Mørketallsundersøkelsen 2001 (Kortversjonen), Næringslivets sikkerhetsorganisasjon (NSO)**:** 16.

McDermot, J. og Fox, C. (1999). Using Abuse Case Models for Security Requirements Analysis. 15th Annual Computer Security Applications Conference, Phoenix, Arizona, US.

McDermot, J. (2001). Abuse-Case-Based Assurance Arguments. 17th Annual Computer Security Applications Conference, New Orleans, Louisiana.

McGraw, G. og Viega, J. (2000). Make Your Software Behave: Assuring Your Software Is Secure. Don't Wait Till a Costly Security Breach. [Internet]. IBM Developer Works. Tilgjengelig på: 18:04:10 ftp://www6.software.ibm.com/software/developer/library/assurance.pdf [24.11 2001].

Merriam-Webster (2002). Merriam-Webster Online - the Language Center [Internet]. Merriam-Webster, Inc. Tilgjengelig på: http://www.m-w.com/ 2002].

Moberg, F. (2000). Security Analysis of an Information System Using Attack Trees-Based Methodology. Department of Computer Engineering. Gøteborg, Chalmers University of Technology**:** 38.

Moore, A. P., Ellison, R. J., et al. (2001). Attack Modeling for Information Security and Survivability. Pittsburgh, PA, Software Engineering Institute, Carnegie Mellon University**:** 30.

Mylopoulos, J., Borgida, A., et al. (1990). "Telos: A Language for Representing Knowledge About Information Systems." ACM Transactions on Information Systems **8**(4): 325-362.

NAT (2002). Nat Patterns Repository [Internet]. Network Associates Technology, Inc. Tilgjengelig på: http://patterns.nailabs.com/repository.html [21.11 2002].

Nunamaker, J. F., Chen, M., et al. (1991). "Systems Development in Information Systems Research." Journal of Management Information Systems **7**: 89-106.

OMG (2001). Omg Unified Modeling Language Specification 1.4, Object Management Group.

OWASP (2001). Application Security Attack Components [Internet]. Tilgjengelig på: http://www.owasp.org/asac 2001].

OWASP (2002). A Guide to Building Secure Web Applications, The Open Web Application Security Project (OWASP).

Peteanu, R. (2001). Best Practices for Secure Development, Razvan's Application Security Page [Online].

Pohl, K. (1993). The Three Dimensions of Requirements Engineering. 5th Int. Conf. of Advanced Information Systems Engineering, Paris, Springer.

Ranum, M. J. (1997). Internet Attacks, Ranum, M. J.

Rawsthorne, D. A. (1996). A Pattern Language for Requirements Analysis. Proc. on The Eleventh Annual ACM Conference on Object-Oriented Programming Systems, Languages and Applications, San Jose.

Raymond, E. S. (2002). The on-Line Hacker Jargon File, Version 4.3.3 [Internet]. 4.3.3. Raymond, Eric S. Tilgjengelig på: http://tuxedo.org/~esr/jargon/ [20.11 2002].

Repstad, P. (1998). Mellom Nærhet Og Distanse : Kvalitative Metoder I Samfunnsfag. Oslo, Universitetsforl.

Robertson, S. (1996). Requirements Patterns Via Events/Use Cases. PLoP'96.

Rolland, C. og Prakash, N. (2000). "From Conceptual Modeling to Requirements Engineering." Annals of Software Engineering **10**: 151-176.

Rosenberg, D. og Scott, K. (2001). Applying Use Case Driven Object Modeling with Uml : An Anotated E-Commerce Example. Boston, Addison-Wesley.

Scambray, J., Kurtz, G., et al. (2001). Hacking Exposed : Network Security Secrets & Solutions. Berkeley, Calif., Osborne/McGraw-Hill.

Schneier, B. (1999). "Attack Trees - Modeling Security Threats." Dr. Dobb's Journal(24): 21-29.

Schneier, B. (2000). Secrets and Lies : Digital Security in a Networked World. New York, John Wiley.

Sindre, G. og Opdahl, A. L. (2000). Eliciting Security Requirements by Misuse Cases. TOOLS Pacific 2000, Sydney, AU.

Sindre, G. og Opdahl, A. L. (2001). Templates for Misuse Case Description. 7th Intl Workshop on Requirements Engineering, Foundation for Software Quality (REFSQ'2001), Interlaken, CH.

Sindre, G., Opdahl, A. L., et al. (2002). Generalization/Specialization as a Structuring Mechanism for Misuse Cases. Symposium on Requirements Engineering for Information Security (SREIS '02), Raleigh, US.

Spafford, E. (2001a). Observations on Security & Assurance, Center for Education and Research in Information Assurance and Security.

Spafford, E. (2001b). The Challenge of Secure Software, Center for Education and Research in Information Assurance and Security.

SREIS (2002). Sreis Homepage [Internet]. SREIS. Tilgjengelig på: www.sreis.org [24.11 2002].

Stallings, W. (2000). Network Security Essentials : Applications and Standards. Upper Saddle River, N.J., Prentice Hall.

Strysick, D. (2002). Pluggable Use Cases. Recent Object Oriented Trends Symposium, Bergen, rOOts.

Viega, J. og McGraw, G. (2002). Building Secure Software : How to Avoid Security Problems the Right Way. Boston, Addison-Wesley.

Walker, R. (1985). Applied Qualitative Research. Aldershot, Gower.

Wheeler, D. A. (2002). Secure Programming for Linux and Unix Howto, Wheeler, David A.

Yoder, J. og Barcalow, J. (1997). Architectural Patterns for Enabling Application Security. the 4th Conference on Patterns Language of programming (PLoP'97).

Zave, P. og Jackson, M. (1997). "Four Dark Corners of Requirements Engineering." ACM Transactions on Software Engineering and Methodology **6**(1): 1-30.