Tatu Lamminmäki

Tietoturvamallien hyödyntäminen sovelluskehityksessä

Tietotekniikan
(ohjelmistotekniikka)
pro gradu -tutkielma
19. huhtikuuta 2008

JYVÄSKYLÄN YLIOPISTO
TIETOTEKNIIKAN LAITOS

Jyväskylä

Tekijä: Tatu Lamminmäki
Yhteystiedot: tplammin@cc.jyu.fi
Työn nimi: Tietoturvamallien hyödyntäminen sovelluskehityksessä
Title in English: Utilizing security patterns in application development
Työ: Tietotekniikan (ohjelmistotekniikka) pro gradu -tutkielma
Sivumäärä: 85
Tiivistelmä: Tutkielmassa keskitytään turvallisuustekniikan hyödyntämiseen sovelluskehityksessä tietoturvamalleja apuna käyttäen. WWW-pohjaiselle liikuntakalenterisovellukselle tehdään tietoturva-analyysi, jonka perusteella sille valitaan ja integroidaan optimaalinen tietoturvamalli.
English abstract: The thesis focuses on how to utilize security engineering in application development with cooperation of security patterns. A security analysis of a WWW-based sports calendar application will be made and based on it an optimal security pattern will be chosen. The chosen pattern will be also integrated to the application.
Avainsanat: tietotekniikka, turvallisuustekniikka, tietoturvamalli, liikuntakalenteri, pro gradu -tutkielma
Keywords: information technology, security engineering, security pattern, sports calendar, Master's thesis

Esipuhe

Monet pro gradu -tutkielmat osoittautuvat lähes elämää suuremmiksi projekteiksi; tämä tutkielma on yksi niistä. Tekoprosessi kesti yli kaksi vuotta, yli 700 päivää ja yötä. Välillä tutkielman teko saattoi jäädä kokonaan työelämän varjoon, ja joskus, yön pimeninä tunteina, epätoivokin saattoi vallata mielen. Kavereiden, ystävien, sukulaisten ja ohjaajien jatkuva kannustus auttoi kuitenkin ratkaisevalla tavalla aina kriittisillä hetkillä, josta suuri kiitos heille!

Pro gradu -projekti aloittiin valitsemalla tutkimuksen aihe. Aiheen valinta tapahtui käytännössä yhden päivän aikana vuoden 2006 alkupuolella käydessäni toisen tulevan graduohjaajani Tommi Kärkkäisen juttusilla tietotekniikan laitoksella. Tommilla oli aiheeksi useita vaihtoehtoja, joista valinta kohdistui turvallisuustekniikkaan, jota laitoksella ei aiemmin ollut tutkittu. Ajan kuluessa laaja aihe tarkentui turvallisuustekniikkaan liittyviin tietoturvamalleihin ja edelleen niiden soveltamiseen sovelluskehityksessä.

Töiden ohessa tutkielma eteni hyvin verkkaisesti, kunnes vuoden 2008 alussa totesin, että tarvitsen gradun tekemiseen enemmän aikaa. Otin asian esille työpaikallani ja sain mahdollisuuden tehdä gradua suurimman osan arkipäivistä, mistä kiitos esimiehilleni. Tämä mahdollisti gradun valmistumisen keväällä 2008.

Raimo Pitkänen antoi merkittävää asiantuntija-apua tutkielmassa käytettyyn liikuntakalenterisovellukseen liittyen ja uhrasi tähän paljon vapaa-aikaansa. Haluankin erityisesti kiittää Raimoa tästä korvaamattomasta avusta! Kiitän erikseen myös ohjaajiani Tommi Kärkkäistä ja Tuomo Rossia, jotka jaksoivat ohjauskerrasta toiseen kannustaa ja auttaa ongelmakohdissa. Lisäksi kiitän työkaveriani Tarja Pajusta tutkielman kieliopin tarkastamisesta.

Oli mielenkiintoista kirjoittaa aiheesta, josta entuudestaan tiesin suhteellisen vähän. Tietoturvamallien käyttö osoittautui mielenkiintoiseksi prosessiksi ja herätti mielenkiinnon myös suunnittelumallien käyttämiseen. Tulen luultavasti hyödyntämään malleja myös jatkossa.

Sanasto

Aitous (engl. *Authenticity*) Kaikkien järjestelmän käyttäjien ja tietojen tulee olla luotettavia ja oikeutettuja. Käyttäjien henkilöllisyys tulee todentaa, ja varmistaa, että järjestelmään tulevat syötteet tulevat luotettavalta taholta.

Altistus (engl. *Exposure*) Mahdollinen haitta tai menetys, mikä voisi syntyä onnistuneesta hyökkäyksestä. Se voi olla informaation menetys tai vaurioituminen, tai tietoturvamurrosta palautumiseen menetetty aika ja työmäärä.

Apache HTTP-palvelin (engl. *Apache HTTP Server*) on avoimeen lähdekoodiin perustuva HTTP-palvelinohjelma.

Etu (engl. *Asset*) Järjestelmän resurssi tai informaatio, jolla on arvoa ja jota tulee suojella.

Eheys (koskemattomuus) (engl. *Integrity*) Järjestelmän tieto on sille kuuluvassa esitysmuodossa eikä sitä pääsee muuttamaan muut kuin oikeutetut tahot heille sallituilla tavoilla.

Haavoittuvuus (engl. *Vulnerability*) Tietokonejärjestelmän heikkous, jota hyödyntämällä voidaan aiheuttaa menetystä tai haittaa.

HTTP (lyhenne sanoista Hypertext Transfer Protocol) Hypertekstin tiedonsiirtoprotokolla, jota selaimet ja WWW-palvelimet käyttävät.

HTTPS (lyhenne sanoista Hypertext Transfer Protocol over Secure Socket Layer) HTTP-protokolla, johon on lisätty salauskerros.

TCP/IP (lyhenne sanoista Transmission Control Protocol / Internet Protocol) Verkkokerroksen (IP) ja kuljetuskerroksen (TCP) yhteyskäytännöt, jotka yhdessä hoitavat sanomien reitityksen ja luotettavan tiedonkulun Internetissä.

Hyökkäys (engl. *Attack*) Järjestelmän haavoittuvuuden hyödyntäminen. Yleensä hyökkäys tulee järjestelmän ulkopuolelta ja sillä halutaan tieten tahtoen aiheuttaa vahinkoa.

Java Sun Microsystemsin kehittämä oliopohjainen ohjelmointikieli.

JSP (lyhenne sanoista JavaServer Pages) Java-pohjainen tekniikka dynaamisen WWW-sivujen luontiin.

Kiistämättömyys (engl. *Non-repudiation*) Elektonisesti solmitut sopimukset pitää pystyä todentamaan eikä tietoturvarikkomuksen tekijöiden tule pystyä jälkikäteen kiistämään sitä, mitä he ovat tehneet. Kiistämättömyyttä voidaan pitää vastuullisuuden osatavoitteena.

Kontrolli (engl. *Control*) Suojauskeino, jolla vähennetään järjestelmän haavoittuvuutta.

Kustannus (engl. *Cost*) kuvaa omaisuuden tai palveluiden rahallista, ajallista ja työnteollista kokonaiskäyttöä.

Käsiteltävyys (engl. *Manageability*) Järjestelmän kyky tulla käytetyksi tai kontrolloiduksi.

Käytettävyys (engl. *Usability*) Ohjelmistotuotteen kyky tulla ymmärretyksi, opituksi, käytetyksi ja kiinnostavaksi käyttäjälle, kun sitä käytetään määrätyissä tilanteissa. Käsitteellä voidaan ilmaista kuinka vaivatonta käyttäjän on operoida, valmistella syötteitä ja tulkita tuloksia järjestelmästä tai komponentista.

Käyttövarmuus (engl. *Dependability*) Järjestelmän kyky tarjota luottamuksen arvoista palvelua ja välttää sellaisia häiriöitä, jotka olisivat käyttäjän hyväksynnän kannalta liian toistuvia, vakavia tai pitkän katkoksen aiheuttavia.

Luottamuksellisuus (engl. *Confidentiality*) Tieto on vain siihen oikeutettujen tahojen saatavilla eivätkä ulkopuoliset pääse sitä näkemään.

MD5 on suosittu tiivistealgoritmi.

MySQL on suosittu ja tehokas SQL-tietokannan hallintajärjestelmä.

Nimettömyys (engl. *Anonymity*) Käyttäjää ei tule pystyä tunnistamaan muista käyttäjistä, mitä tarvitaan joissain tapauksissa, jotta yksityisyys voidaan turvata.

PHP (lyhenne sanoista PHP: Hypertext Preprocessor) Perlin kaltainen ohjelmointikieli, jota käytetään eritoten WWW-palvelinympäristöissä dynaamisten WWW-sivujen luontiin.

Saatavuus (käytettävyys) (engl. *Availability*) Tieto ja palvelut ovat tarvittaessa käytettävissä niihin oikeutetuille tahoille.

Salaisuus (engl. *Secrecy*) Tekninen termi, jota käytetään viitattaessa informaation pääsyä rajoittavien mekanismien (kuten tiedon kryptauksen tai pääsynvalvonnan) tehoon.

Siirrettävyys (engl. *Portability*) Ohjelmistotuotteen kyky siirtyä organisaatio-, laite- tai ohjelmistoympäristöstä toiseen.

Suoritusteho (engl. *Performance, efficiency*) ilmaisee ohjelmistotuotteen kyvykkyyttä antaa asianmukaista tehoa suhteessa käytettyihin resursseihin määrätyssä tilanteessa.

Tarkastettavuus (engl. *Auditability*) Kyky näyttää toteen järjestelmän asianmukainen toiminta menneisyydessä ja nykyhetkessä.

Tietoturvamallit (engl. *Security patterns*) ovat hyvin todistettuja valmiita malleja usein toistuviin tietoturvaongelmiin.

Turvallisuustekniikka (engl. *Security engineering*) on järjestelmien turvallisuuden rakentamiseen keskittyvä tieteenala, jonka tarkoitus on saattaa turvallisuusteoriat turvallisuuskäytännöiksi ja muuttaa uhkaavat tilanteet hyväksyttäviksi riskeiksi.

Uhka (engl. *Threat*) on seikka, joka mahdollisesti aiheuttaa menetystä tai haittaa, so. hyökkäykselle altis järjestelmän haavoittuvuus.

Vastuullisuus (engl. *Accountability*) Tietojen käsittelijät tai muuttajat tulee pystyä jäljittämään tietoturvarikkomusten varalta esimerkiksi järjestelmälokien avulla.

WWW (lyhenne sanoista World Wide Web) Internetissä käytetty hajautettu hypertekstijärjestelmä.

Yksityisyys (engl. *Privacy*) Kyky ja/tai oikeus suojella henkilökohtaisia salaisuuksia ja estää henkilökohtaisen tilan loukkaaminen.

Ylläpidettävyys (engl. *Maintainability*) Ohjelmistotuotteen sietokyky ohjelmatason korjauksille, parannuksille ja mukauttamisille sekä ympäristö-, toiminnallisuus- ja vaatimusmäärittelytason muutoksille.

Sisältö

1 Johdanto

Turvallisuus tulee usein huomioiduksi vasta, kun muut asiat on kunnossa. Näin on käynyt muun muassa auto- ja rakennusteollisuudessa. Myös ohjelmistokehityksessä on vasta viime vuosien aikana alettu painottamaan turvallisuuden merkitystä. Hyvänä esimerkkinä on Microsoft, joka vuonna 2002 lopetti kaikki normaalien ominaisuuksien kehittämisen Windows-käyttöjärjestelmään ja keskitti vapautuneet voimavarat seuraavan tuotteensa turvallisuuden parantamiseen [8, s. xxiii].

Pyrittäessä turvallisuuden parantamiseen törmätään vain harvoin täysin uudenlaisiin ongelmiin. Useimmat ongelmat on jo moneen kertaan ratkaistu ja vieläpä samoin menetelmin. Olisi siis hyvin järkevää koota tällaiset ratkaisumenetelmät yleiskäyttöiseen muotoon, jota järjestelmien kehittäjät voisivat käyttää apuna ongelmia ratkoessaan. Kysyntä ja tarjonta ovat kohdanneet toisensa turvallisuustekniikkaan läheisesti liittyvien tietoturvamallien muodossa.

Tässä tutkielmassa selvitetään mitä turvallisuustekniikalla ja siihen liittyvillä tietoturvamalleilla tarkoitetaan ja kuinka tietoturvamalleja voidaan hyödyntää sovelluskehityksessä.

Luvussa 2 kerrotaan yleisesti turvallisuustekniikasta. Luvussa 3 tutkitaan turvallisuustekniikkaa sovelluskehityksen näkökulmasta. Luvussa 4 annetaan kattava kuvaus tietoturvamalleista ja esitellään nk. ydinmallikokoelma. Luvussa 5 esitellään liikuntakalenterisovellus ja tarkastellaan sitä lähemmin turvallisuustekniikan ja tietoturvamallien näkökulmasta. Luvussa 6 esitetään tarkasti yksi tietoturvamalli. Luvussa 7 tutkitaan tietoturvamallin integroimisprosessia liikuntakalenterisovelluksen avulla ja lopuksi luvussa 8 tehdään yhteenveto tutkielmasta.

2 Turvallisuustekniikka

Luvussa kerrotaan mihin turvallisuustekniikkkaa tarvitaan, tarkastellaan sitä tieteenalana ja haetaan syitä turvallisuuden tarpeen korostumiseen.

2.1 Mihin turvallisuustekniikkaa tarvitaan?

Yksityisyyden ja omaisuuden turvaaminen ovat aina olleet tärkeitä asioita. Perinteiset turvamenetelmät — kuten lukot tai allekirjoitukset — ovat käyneet yhä enemmissä määrin vanhanaikaisiksi ja toimimattomiksi asioiden siirtyessä mekaanisesta maailmasta sähköiseen maailmaan. [1, s. xix]

Uusimpana suuntauksena on tehdä mahdollisimman monesta tietoverkosta, tietokoneesta, ohjelmistosta tai jopa yrityksestä yhteensopivia ja toisiinsa kytköksissä olevia. Tämä on huomattavasti lisännyt tuotteiden ja järjestelmien turvallisuuden merkitystä. [10, s. 30]

Uusia turvamenetelmiä on kehitetty jatkuvasti, mutta monesti ne eivät ole olleet riittävän hyviä ja niitä on jouduttu paikkaamaan uusilla versioilla. Samat virheet ovat toistuneet, koska jo toimiviksi tai toimimattomiksi havaittuja turvamenetelmiä ei ole ollut suunnittelijoiden tiedossa. Valitettavan usein turvamenetelmät ovat suojelleet vääriä asioita tai oikeita asioita väärällä tavalla. [1, s. xix-xx, 4]

Turvallisuustekniikka (engl. *security engineering*) on uusi tieteenala, jolla pyritään tuomaan järjestys tähän kaaokseen [1, s. xx]. Se toimii siltana yhdistäen salaustutkijat, käyttöjärjestelmän suojaamiseen erikoistuneet, varashälytinteollisuuden ja kaikki muut turvallisuusteknologian osaajat [1, s. 541].

Turvallisuustekniikan tietämystä hyödyntäen voidaan suunnitella ja toteuttaa järjestelmiä, jotka ovat suojassa ilkivallalta, virheiltä ja onnettomuuksilta [1, s. 3]. Sen avulla turvallisuusongelmat voidaan tunnistaa, niitä voidaan välttää ja niistä voidaan toipua [13, s. 722].

Turvallisuustekniikan tarkoitus on saattaa turvallisuusteoriat turvallisuuskäytännöiksi ja muuttaa uhkaavat tilanteet hyväksyttäviksi riskeiksi [3, s. 59].

2.2 Turvallisuustekniikka tieteenalana

Turvallisuustekniikka on uusi ja kehittyvä tieteenala. Tätä kuvastaa hyvin se, että ensimmäinen kattava turvallisuustekniikkaan liittyvä kirja julkaistiin vasta vuonna 2001 Ross Andersonin toimesta [1, s. xxiii]. Turvallisuustekniikalle ei myöskään vielä ole tarkkaa yleisesti hyväksyttyä määritelmää [3, s. 59] [10, s. 30].

Ross Anderson määrittelee turvallisuustekniikan tieteenalana vapaasti suomennettuna seuraavalla tavalla [1, s. 3]:

> Turvallisuustekniikka on tieteenala, joka keskittyy välineisiin, prosesseihin ja menettelytapoihin, mitä tarvitaan suunniteltaessa, toteutettaessa ja testattaessa kokonaisia järjestelmiä ja mukautettaessa olemassaolevia järjestelmiä, kun niiden ympäristö kehittyy.

Turvallisuustekniikassa tarvitaan poikkitieteellistä asiantuntemusta ja se on sidoksissa lukuisiin muihin tieteenaloihin, mukaanlukien [10, s. 31]:

- Yritystiede (engl. *Enterprise engineering*)
- Systeemitekniikka (engl. *Systems engineering*)
- Ohjelmistotekniikka (engl. *Software engineering*)
- Työtiede (engl. *Human factors engineering*)
- Tietoliikennetekniikka (engl. *Communications engineering*)
- Laitteistotekniikka (engl. *Hardware engineering*)
- Testaustiede (engl. *Test engineering*)
- Järjestelmien ylläpito (engl. *System administration*)

Turvallisuustekniikkaan kuuluva toiminta täytyy koordinoida useiden ulkoisten toimijoiden kanssa, sillä varmuus ja hyväksyntä toiminnan mukanaan tuomille vaikutuksille todennetaan yhdessä kehittäjien, integraattorien, ostajien, käyttäjien, puolueettomien arvioijien sekä muiden ryhmien kanssa. Nämä kytkökset ja tarvittavat laaja-alaiset organisaatioiden väliset kanssakäymiset tekevät turvallisuustekniikasta erittäin monimutkaisen ja omanlaisensa insinööritaidon tieteenalan. [10, s. 31]

Osaamista turvallisuustekniikassa tarvitaan muun muassa kryptografiasta ja tietokonejärjestelmien turvallisuudesta, laitteistojen peukaloinnin estosta, soveltavasta psykologiasta, organisaatioiden toimintatavoista, tilintarkastuksesta ja oikeustieteestä. Järjestelmien suunnittelutaidot kuten liiketoiminta-analyysi, ohjelmistosuunnittelu, -arviointi ja -testaus ovat tärkeitä turvallisuustekniikassa, sillä niiden avulla voidaan suojautua virheiltä ja onnettomuuksilta. Ne eivät kuitenkaan riitä, sillä myös ilkivallalta tulee pystyä suojautumaan. [1, s. 3]

Turvallisuuskysymysten korostuminen on kohottanut turvallisuustekniikan merkitystä tieteenalana. Turvallisuustekniikan tulisi olla ratkaisevassa roolissa, kun eri tieteenalojen asiantuntijat ja ryhmät yhdessä rakentavat, integroivat, operoivat, hallinnoivat, ylläpitävät ja kehittävät järjestelmiä ja ohjelmia, tai rakentavat, toimittavat ja kehittävät tuotteita. [10, s. 30]

Luvussa 3 tarkastellaan tarkemmin turvallisuustekniikan hyödyntämistä ohjelmistotekniikassa.

2.3 Miten turvallisuustekniikkaa sovelletaan?

Turvallisuutta on alettu käsittää laajemmassa mittakaavassa. Enää ei turvata vain valtiosalaisuuksia, vaan pyritään takaamaan myös esimerkiksi taloudellinen liiketoiminta, sopimusluonteiset päätökset, henkilökohtaiset tiedot ja Internetin turvallisuus. Tämän seurauksena on tarpeellista, että mahdolliset turvallisuustavoitteet (kts. kappale 3.5) huomioidaan ja päätetään joka sovellukselle. Turvallisuusasiat tulisi kohdistaa yritysten ja liiketoimintamenetelmien prosessien määrittelyyn, hallintaan ja uudelleenorganisointiin. Tällöin turvallisuustekniikka voidaan tarjota asiakkaalle järjestelmänä, tuotteena tai palveluna. [10, s. 30]

Useat turvallisuustekniikkaa tukevat tekniikat — kuten kryptografia, ohjelmistojen luotettavuus ja laitteistojen peukaloinnin estäminen — ovat jo nyt hyvin kehittyneitä, mutta tietotaito ja kokemus niiden tehokkaaseen soveltamiseen puuttuu. Hyvältä turvallisuustekniikan osaajalta vaaditaan, että hän ymmärtää järjestelmään mahdollisesti kohdistuvat uhat ja osaa käyttää sopivaa yhdistelmää teknologisia ja organisaatiollisia suojakeinoja hallitakseen ne. Päätöksenteossa auttaa suuresti tieto siitä, mikä aiemmin on toiminut ja etenkin se mikä ei. [1, s. xix-xx]

Eräs tapa soveltaa turvallisuustekniikaa ovat tietoturvamallit. Tietoturvamalleista kerrotaan kappaleessa 4.

2.4 Turvallisuustekniikan päämäärät

Turvallisuustekniikkaan voidaan määrittää seuraavat yleisesti hyväksytyt päämäärät [10, s. 30]:

- Saada tietämys hankkeen turvallisuusriskeistä.
- Laatia tasapuolinen joukko turvallisuustarpeita tunnistettujen riskien mukaisesti.
- Muuttaa turvallisuustarpeet turvallisuusneuvonnaksi yhdistettäväksi projektissa muiden tieteenalojen käyttämille toiminnoille ja kuvaukseksi järjestelmän konfiguraatiosta tai toiminnasta.
- Muodostaa luottamus tai varmuus turvallisuusmekanismien oikeellisuudelle ja tehokkuudelle.
- Päättää ovatko jäljelle jäävien haavoittuvuuksien käyttövaikutukset järjestelmään tai sen toimintaan siedettävät (hyväksyttävät riskit).
- Yhdistää eri insinööritaidon tieteenalojen ja erikoisalojen vaivannäkö yhteisymmärrykseksi järjestelmän luotettavuudesta.

3 Turvallisuustekniikka ohjelmistotekniikassa

Luvussa tarkastellaan turvallisuustekniikkaa ohjelmistotekniikan näkökulmasta kertomalla miten turvallisuustekniikka liittyy ohjelmistotekniikkaan, selvittämällä ohjelmistotekniikan turvallisuuskäsitteitä, määrittelemällä turvallisuustavoitteet ja käymällä läpi ohjelmiston turvallisuuden peruselementit.

3.1 Mitä turvallisuustekniikka tarjoaa ohjelmistotekniikkaan?

Moderni yhteiskunta on hyvin riippuvainen ohjelmistoista. 1990-luvulla Internet ja julkisten pakettikytkentäisten verkkojen käyttö synnytti uuden haasteen ohjelmistojen kehittäjille: suunnitella ja toteuttaa Internet-aikakaudelle turvallisia järjestelmiä [13, s. 718].

Kun yhä suurempi määrä entistä monimutkaisempia ja laajempia järjestelmiä liitetään Internetiin, kasvavat samalla niihin kohdistuvien ulkoisten hyökkäysten kirjo ja määrä [13, s. 718] [4]. Tällaisia hyökkäyksiä ovat esimerkiksi virukset, järjestelmien palveluiden luvaton käyttö ja järjestelmän sekä sen tietojen luvaton muuttaminen [13, s. 58]. Hyökkäyksien avulla voidaan väärinkäyttää järjestelmän laitteistoa, häiritä sen tarjoamia palveluita tai varastaa arkaluontoista tietoa. Ohjelmistot tulisi suunnitella siten, että ne kestävät ulkoiset hyökkäykset ja pystyvät toipumaan niistä [13, s. 718]. Ohjelmistokehittäjien tulisikin huomioida tietoturva ohjelmistokehityksen joka vaiheessa [2]. Tämä kaikki lisää turvallisuustekniikan merkitystä ohjelmistotekniikassa.

Muutokset ohjelmistojen kehittämistavoissa ja ohjelmistoarkkitehtuureissa ovat avanneet uusia mahdollisuuksia soveltaa turvallisuustekniikkaa. Kryptografiaa ja peukaloinninestolaitteistoja voidaan käyttää luomaan luottamusta ohjelmistotyökauluihin ja prosesseihin. Nämä mahdollisuudet kumpuavat siitä, että ohjelmistot enää harvoin ovat yhtenäisiä yhden valmistajan luomuksia. Sen sijaan ne ovat monimutkaisia, myöhäisessä vaiheessa nivottuja kokoelmia, jotka on koottu kaupallisista valmiselementeistä. [2]

Valmiselementit tarjoavat suuren kustannusedun verrattuna räätälöityyn ohjelmistoon. Suojellakseen osaamistaan valmiselementtien valmistajat myyvät ohjelmistokomponenttinsa käännettynä koodina, ilman lähdekoodia tai suunnitteludokumentteja. Tämän seurauksena ohjelmistoissa on mustalaatikkokomponentteja, joi-

hin ohjelmistokehittäjät joutuvat luottamaan. Eräs riskialttiimmista toimintavoista on mobiilikoodin myöhäinen käyttöönotto ohjelmissa. Mobiilikoodilla tarkoitetaan ohjelmaa, joka saadaan ulkoisesta järjestelmästä ja se voi saapua ja käynnistyä ilman käyttäjän toimenpiteitä [25]. Näihin ongelmiin voidaan tarttua turvallisuustekniikkaan kuuluvilla asioilla, esimerkiksi salaustekniikoilla, kuten nollatietotodistuksella (engl. *interactive proof*) tai peukaloinninestolaitteilla [2].

Ohjelmistoarkkitehtuurit, -suunnittelijat ja -testaajat ovat suuressa määrin autuaan tietämättömiä sovellusten turvallisuusongelmista. Peruskoulutus kriittisiin turvallisuusongelmiin voidaan antaa kuvailemalla ongelma ja osoittamalla sen tärkeys ja vaikutus. Kuitenkin vasta perusteellisella koulutuksella — käsittelemällä turvallisuustekniikaa, suunnitteluperiaatteita ja -suuntaviivoja, toteutusriskejä, suunnitteluvirheitä, analysointitekniikoita, sovellusten hyväksikäyttöä ja turvallisuustestausta — voidaan päästä todella turvallisiin ohjelmistoihin.[4]

3.2 Sovellusten turvallisuus

Sovellusten turvallisuus on keskeisessä osassa tietokonepohjaisten järjestelmien turvallisuusongelmissa. Sovellusten toteutusvaiheen virheet, kuten puskuriylivuodot, ja suunnitteluvaiheen virheet, kuten ristiriitaiset virhekäsittelyt, mahdollistavat hyökkäykset järjestelmiin varmasti vielä monia vuosia. Sovelluksen turvallisuutta voidaan tehostaa käyttämällä hyödyksi hyviä ohjelmistotekniikasta tuttuja toimintatapoja. Turvallisuus tulisi olla mukana heti sovelluksen kehityksen alkuvaiheessa, yleiset uhkat tulisi tiedostaa ja ymmärtää (mukaanlukien ohjelmistokielen virheet ja sudenkuopat), sovelluksen suunnittelu tulisi tehdä korostaen turvallisuutta ja kaikki sovelluksen osatekijät altistaa perusteellisille objektiivisille riskianalyyseille ja testauksille. On paljon helpompi suunnitella ohjelma turvalliseksi, kun alkaa pohtimaan ja korjaamaan turvallisuutta vasta juustonreikäisen ohjelman valmistuttua.[4]

Määriteltäessä ja suunniteltaessa sovellusten turvallisuutta tulee huomioida itse sovelluksen turvallisuuden lisäksi infrastruktuuri, jolle sovellus on rakennettu. Infrastruktuuriin kuuluu ohjelmasta riippuen [13, s. 718]:

- Käyttöjärjestelmä, esimerkiksi Linux tai Windows.
- Järjestelmässä ajossa olevat yleiset sovellukset, kuten Internet-selaimet ja sähköpostiohjelmat.
- Tietokannanhallintajärjestelmät, kuten MySQL.

- Väliohjelmistot, jotka tukevat hajautettua tietojenkäsittelyä ja tietokantaan pääsyä.
- Sovelluksen käyttämät uudelleenkäytettävät komponenttikirjastot.

Infrastuktuuriin komponentit tunnetaan hyvin ja ne ovat laajasti saatavilla. Niihin kohdistuukin suurin osa hyökkäyksistä [13, s. 718]. Erikoista on, että MITREn tekemien tutkimusten mukaan vuonna 2005 raportoiduista haavoittuvuuksista vain noin 25% liittyi infrastuktuuriin ja yli 75% viallisiin sovelluksiin [23]. Tästä voidaan päätellä, että infrastruktuurin turvallisuuteen kiinnitetään paljon huomiota.

Ohjelmiston turvallisuus on ohjelmiston laadullinen ominaisuus, jonka takaamiseksi järjestelmä suunnitellaan kestämään hyökkäykset. Infrastuktuurin turvallisuus on järjestelmänhallinnalle kuuluva asia, jonka takaamiseksi järjestelmä konfiguroidaan kestämään hyökkäykset [13, s. 719]. Järjestelmänhallinnan tehtäviin kuuluu myös tarkkailla hyökkäysiä ja raportoida niistä kehitystiimille [4].

Tehokkain ja edullisin tapa turvallisuuden takaamiseksi on suunnitella ja toteuttaa ohjelmistot niin, että ne ovat jo valmiiksi mahdollisimman turvallisia. Mitä enemmän turvallisuus on otettu huomioon ohjelmaa suunniteltaessa ja toteuttaessa, sitä vähämmän turvallisuus on riippuvainen järjestelmähallinnan konfiguroinnista: on parempi parantaa tauti kuin hoitaa oiretta. Järjestelmänhallinnasta ei kuitenkaan kannata tinkiä, sillä täysin turvallista ohjelmaa on mahdotonta tehdä ja hyökkäykset ovat aina mahdollisia. [4]

Sovelluksen turvallisuus on järjestelmänlaajuinen ongelma eikä sitä saavuteta vain lisäämällä sovellukseen irrallisia turvallisuusominaisuuksia. Esimerkiksi SSL-pohjainen tiedon salaus ei tarjoa kokonaisratkaisua tietoturvaongelmiin, eikä sen lisääminen ohjelmaan tee ohjelmasta tietoturvallista automaattisesti. Huomioon tulisi ottaa sekä turvallisuusmekanismit (kuten pääsynvalvonta) että turvallisuuslähtöinen suunnittelu, koska ne ovat usein erillisiä asioita. Tietoturvaongelma esiintyy todennäköisemmin järjestelmän perusasioissa (kuten tietokantamoduulin rajapinnassa) kuin jossakin tietoturvaominaisuuksissa. [4]

3.3 Turvallisuusmenetelmät vesiputousmallissa

Tämä kappale perustuu pääosin lähteeseen [4]. Vesiputousmalli on perinteinen tapa kuvata sovelluskehitystä. Kuvasssa 3.1 on esitetty vesiputousmalli, jossa on korostettu turvallisuutta lisäämällä malliin turvallisuutta parantavia menetelmiä. Seuraavassa kerrotaan tarkemmin näistä menetelmistä:

- Väärinkäyttötapauksien (engl. *abuse cases*) avulla kuvaillaan järjestelmän käyttäytymistä hyökkäyksen aikana ja ne auttavat turvallisuusvaatimusten määrittelyssä. Niiden tekeminen vaatii selkeää näkemystä siitä mitä pitää suojella, keneltä ja kuinka kauan.

- Riskianalyysi (engl. *risk analysis*) on välttämätön sovelluksen arkkitehtuuria määriteltäessä sekä luokkahierarkiaa suunniteltaessa. Turvallisuussuunnittelijat paljastavat riskit ja asettavat ne tärkeysjärjestykseen. Katso myös kappale 3.7.

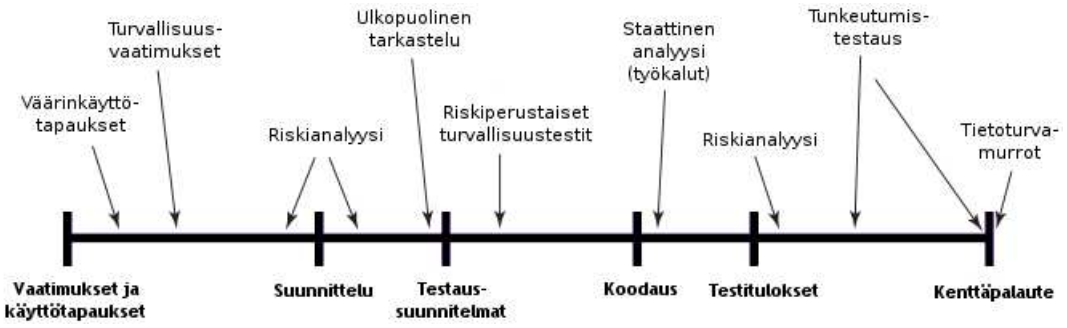
- Ulkopuolinen tarkastelu (engl. *external review*) on suunnittelutiimin ulkopuolista tarkastelua, jota tarvitaan usein riskianalyysin teossa.

- Staattisen analyysin työkalut (engl. *static analysis tools*) ovat työkaluja, joilla voidaan etsiä lähdekoodista yleisiä haavoittuvuuksia sovelluksen toteutusvaiheessa (koodausvaiheessa).

- Riskiperustaiset turvallisuustestit (engl. *risk-based security tests*) perustuvat hyökkäys- ja uhkamalleihin (engl. *attack patterns, threat models*). Ne mahdollistavat yhdessä normaalien toiminnallisuutta testaavien tekniikoiden kanssa sellaisten testaussuunnitelmien teon, joilla voidaan jäljittää tietoturvavaatimukset.

- Tunkeutumistestauksen (engl. *penetration testing*) tarpeellisuus tulee usein esille arkkitehtuuritason riskianalyyseissä. Tunkeutumistestaus auttaa ohjelman ymmärtämistä sen oikeassa ympäristössä. Tunkeutumistestaus on sitä hyödyllisempi, mitä enemmän sitä voidaan käyttää sovelluksen arkkitehtuuritasolla.

- Tietoturvamurtojen (engl. *security breaks*) tarkkailu ja raportointi ovat tärkeässä roolissa turvallisuuden ylläpidossa.



Kuva 3.1: Turvallisuusmenetelmät vesiputousmallissa [4]

3.4 Turvallisuuskäsitteet ohjelmistotekniikassa

Tietokonejärjestelmien ja ohjelmistojen turvallisuuskäsitteet on lueteltu taulukossa 3.1.

Taulukko 3.1: Turvallisuuskäsitteet [13, s. 720]

Termi	Kuvaus
Etu (engl. *Asset*)	Järjestelmän resurssi tai informaatio, jolla on arvoa ja jota tulee suojella.
Altistus (engl. *Exposure*)	Mahdollinen haitta tai menetys, mikä voisi syntyä onnistuneesta hyökkäyksestä. Se voi olla informaation menetys tai vaurioituminen, tai tietoturvamurrosta palautumiseen menetetty aika ja työmäärä.
Haavoittuvuus (engl. *Vulnerability*)	Tietokonejärjestelmän heikkous, jota hyödyntämällä voidaan aiheuttaa menetystä tai haittaa.
Hyökkäys (engl. *Attack*)	Järjestelmän haavoittuvuuden hyödyntäminen. Yleensä hyökkäys tulee järjestelmän ulkopuolelta ja sillä halutaan tieten tahtoen aiheuttaa vahinkoa.
Uhka (engl. *Threat*)	Seikka, joka mahdollisesti aiheuttaa menetystä tai haittaa, so. hyökkäykselle altis järjestelmän haavoittuvuus.
Kontrolli (engl. *Control*)	Suojauskeino, jolla vähennetään järjestelmän haavoittuvuutta.

Turvallisuusuhkat voivat kohdistua järjestelmän ja sen informaation luottamuksellisuuteen, eheyteeen sekä käytettävyyteen. Mikäli yksikin näistä vaarantuu, vaarantuvat myös toiset. Näistä kohteista kerrotaan tarkemmin luvussa 3.5.

Kontrolli voidaan jakaa kolmeen luokkaan. Se voi olla [13, s. 721-722]:

- Järjestelmän suunnitteluvaiheessa tehty suojauskeino, jolla hyökkäykset saadaan estettyä. Hyvänä esimerkkinä on järjestelmän sulkeminen pois ulkoverkosta.

- Järjestelmän toimintaa monitoroiva toiminnallisuus, jolla hyökkäyksiä voidaan tunnistaa ja torjua.
- Tukea ongelmista toipumiseen, kuten esimerkiksi tiedon varmuuskopionti ja peilaus tai vakuutusmenettely.

Yleisesti ottaen uhkan, kontrollin ja haavoittuvuuden välinen suhde voidaan esittää siten, että uhka torjutaan kontrolloimalla haavoittuvuutta. [9, s. 7]

3.5 Turvallisuustavoitteet

Tietokonejärjestelmien turvallisuus pitää sisällään kolme hyvin tärkeää tavoitetta: luottamuksellisuuden, eheyden ja käytettävyyden. Turvallista järjestelmää rakennettaessa pyritään löytämään sopusointu näiden tavoitteiden välille. [9, s. 10] Näiden tavoitteiden lisäksi voidaan vielä erikseen ottaa esille aitous, vastuullisuus, kiistämättömyys ja nimettömyys.

- Luottamuksellisuus (engl. *confidentiality*): Tieto on vain siihen oikeutettujen tahojen saatavilla eivätkä ulkopuoliset pääse sitä näkemään [9, s. 10] [5] [11] [12]. Luottamuksellisuuden määritelmä on osittain päällekkäinen salaisuuden (engl. *secrecy*) ja yksityisyyden (engl. *privacy*) kanssa [1, s. 10]:
  - Salaisuus on tekninen termi ja sitä käytetään viitattaessa informaation pääsyä rajoittavien mekanismien, kuten tiedon kryptauksen tai pääsynvalvonnan, tehoon.
  - Luottamuksellisuuteen sisältyy tietoon tulleiden muiden henkilöiden tai organisaatioiden salaisuuksien suojeleminen.
  - Yksityisyys on kyky ja/tai oikeus suojella henkilökohtaisia salaisuuksia ja estää henkilökohtaisen tilan loukkaaminen. Se on oikeus nähdä ja kontrolloida omat henkilökohtaiset tiedot [23].
- Eheys (koskemattomuus) (engl. *integrity*): Tieto on sille kuuluvassa esitysmuodossa eikä sitä pääsee muuttamaan muut kuin oikeutetut tahot heille sallituilla tavoilla [9, s. 10] [5] [11] [12].
- Saatavuus (käytettävyys) (engl. *availability*): Tieto ja palvelut ovat tarvittaessa käytettävissä niihin oikeutetuille tahoille [9, s. 10] [11].

- Aitous (engl. *authenticity*): Kaikkien järjestelmän käyttämien tietojen ja kaikkien järjestelmän käyttäjien tulee olla luotettavia ja oikeutettuja. Käyttäjien henkilöllisyys tulee todentaa ja lisäksi on varmistettava, että järjestelmään tulevat syötteet tulevat luotettavalta taholta [5]. Eheys ja aitous -käsitteet ovat hyvin lähellä toisiaan: aitoutta käytetään usein siinä yhteydessä, kun lähteen luotettavuuden varmistumisesta on kulunut vain vähän aikaa [1, s. 11].

- Vastuullisuus (engl. *accountability*): Tietojen käsittelijät tai muuttajat tulee pystyä jäljittämään tietoturvarikkomusten varalta esimerkiksi järjestelmälokien avulla [5] [11].

- Kiistämättömyys (engl. *non-repudiation*): Elektonisesti solmitut sopimukset pitää pystyä todentamaan [12]. Tietoturvarikkomuksen tekijöiden ei tule pystyä jälkikäteen kiistämään sitä, mitä he ovat tehneet. Kiistämättömyyttä voidaan pitää vastuullisuuden osatavoitteena [11].

- Nimettömyys (engl. *anonymity*): Käyttäjää ei tule pystyä tunnistamaan muista käyttäjistä [23]. Nimettömyyttä tarvitaan joissain tapauksissa, jotta yksityisyys voidaan turvata [9, s. 602] [1, s. 10].

Luottamuksellisuuden merkitys verrattuna saatavuuteen ja eheyteen on vähentynyt, ja varsinkin saatavuuden merkitys on kasvanut palvelunestohyökkäyksien vuoksi [1, s. 321-322, 541].

3.6 Ohjelmiston turvallisuuden peruselementit

Ohjelmistojen turvallisuus rakentuu kolmesta toisistaan riippuvaisesta peruselementistä, jotka ovat haavoittuvuudet, edut ja hyökkääjät [5]. Seuraavissa kolmessa alaluvussa kerrotaan tarkemmin näistä peruselementeistä. Luku perustuu pääosin lähteeseen [5].

3.6.1 Hyökkääjät

Ilman hyökkääjiä ei olisi myöskään hyökkäyksiä. Hyökkääjät voidaan jakaa kahteen kategoriaan: ulkopuolisiin hyökkääjiin sekä järjestelmän luottamuksen omaaviin sisäpiirin hyökkääjiin. Ulkopuoliset ovat luottamuksen puutteen vuoksi helpommin torjuttavissa. Sisäpiiriläisillä sen sijaan on luottamuksen lisäksi yleensä myös tietämystä järjestelmästä ja yleensä tietty päämäärä hyökkäyksiinsä toisin kuin ulkopuo-

lisilla. Lisäksi heidän motiivinsa ja menetelmänsä eroavat ulkopuolisten motiiveista ja menetelmistä.

Hyökkääjiä voidaan kategorisoida myös muun muassa määrätietoisuuden (suunnitelmallinen teko, sattumanvarainen), motivaation (sabotaasi, vakoilu, kosto), tietotaidon (amatööri, ammattilainen) ja fyysisen pääsyn (paikallinen, ulkoinen) perusteella. Hyökkääjä voi toimia aktiivisesti tai passiivisesti. Aktiivinen hyökkääjä toimii suoraan etuja manipuloiden, kun taas passiivinen pyrikii hankkimaan informaatiota etuja tarkkailemalla. [3, s. 92]

Hyökkääjällä tulee olla [9, s. 8]:

- Motiivi: syy tai halu suorittaa hyökkäys.
- Menetelmä: taito, tietämys, välineet ja muut tarvittavat asiat joiden avulla hyökkäys voidaan toteuttaa.
- Tilaisuus: aika ja pääsy suorittaa hyökkäys.

3.6.2 Edut

Edut (kuten resurssit ja informaatio) ovat syy siihen, miksi hyökkäyksiä tehdään. Etuja voidaan suojata esimerkiksi palomuurien, tunkeutumisen havaitsemisjärjestelmien, pääsynvalvonnan, käyttöoikeuksien, autentikoinnin ja kryptografian avulla.

Eduilla on jokin arvo jollekin taholle, esimerkiksi organisaatiolle tai henkilölle. Etu voi olla konkreettinen ja materialistinen, esimerkiksi ohjelmisto, järjestelmä tai verkko — mutta se voi olla myös abstraktimpi ja ihmisläheisempi, kuten tietämys tai osaaminen. [3, s. 91]

Mike Andrewsin ja James A. Whittakerin [5] mukaan paras etujen suojaustapa on nk. syvyyssuuntainen turvallisuusajattelu (engl. *defense in depth*). Ajatuksena siinä on antaa suoja useaan kerrokseen esimerkiksi siten, että verkkoa suojataan palomuureilla ja tunkeutumisen havaitsemisjärjestelmillä, käyttöjärjestelmää salasanoilla ja älykorteilla sekä yksittäisiä ohjelmia paremmilla suunnitteluvälineillä ja koulutuksella. Vaikka yhden kerroksen turvallisuus vaarantuisi, niin muut kerrokset pysyvät turvallisina.

3.6.3 Haavoittuvuudet

Haavoittuvuudet mahdollistavat hyökkäykset. Ne ovat järjestelmän suunnittelu- ja toteutusvirheitä — heikkouksia, joita hyödyntämällä hyökkääjä pääsee käsiksi jär-

jestelmän etuihin. Haavoittuvuuksia voi esiintyä palveluissa, ohjelmissa ja käyttäjän toiminnassa.

Palveluihin perustuvat haavoittuvuudet ovat usein vakavia turvallisuuden kannalta, sillä palvelut eivät yleensä vaadi käyttäjältä toimia. Hyökkääjän ja kohteen välille avautuu suora yhteys ja hyökkäyskoodi otetaan vastaan osana palvelua.

Ohjelmistohaavoittuvuudet voivat myös vaarantaa turvallisuuden vakavalla tavalla, vaikka niiden hyödyntäminen vaatii yleensä käyttäjän toimia. Esimerkiksi sähköpostissa oleva virus vaatii käyttäjän toimia, jotta se voisi hyödyntää sähköpostiohjelman haavoittuvuutta. Taidokkailla huijauksilla käyttäjät yritetään saada lankeamaan ansaan. Käyttäjä voi esimerkiksi avata sähköpostiviestissään olevan houkuttelevasti nimetyn liitetiedoston, jolloin ohjelmistohaavoittuvuutta päästään käyttämään.

Käyttäjä voi omalla toiminnallaan altistaa järjestelmän haavoittuvuuksille. Käyttäjä voi esimerkiksi konfiguroida tai käyttää tietokonetta sekä sen ohjelmia turvallisuutta vaarantavalla tavalla.

Käyttäjän manipuloinnilla (engl. *social engineering*) yritetään huijata käyttäjää antamaan henkilökohtaisia tietojaan hyökkääjille esimerkiksi väärennettyjen sähköpostiviestien tai Internet-sivujen avulla.

3.7 Riskinhallinta

Tehokkaaseen turvallisuustekniikan käyttöön kuuluu tietoturvariskien arviointi ja hallinta. Riski on mahdollinen ongelma, jonka järjestelmä tai sen käyttäjä voi kohdata. Se on mitä tahansa, joka voi uhata liiketoiminnan turvallisuutta. Riskinhallinta on järjestelmän etuihin kohdistuvien hyökkäysten aiheuttaminen menetysten ja niiden torjuntaan tarvittavien toimenpideiden kustannusten tasapainottamista siten, että toiminta olisi mahdollisimman kustannustehokasta. Riskinhallinta liittyykin liiketoimintaan, eikä tekniikkaan. Sovelluskehittäjien tulee antaa asiantuntevaa apua liiketoiminnasta — eli riskien arvoinnista — päättäville elimille. [9, s.506] [13, s. 722]

Riskien arviointi alkaa jo ennen järjestelmän hankintaa. Riskinhallinnan tulisi jatkua koko järjestelmän kehityksen ajan. Riskien arviointi on vaiheittainen prosessi sen suhteen kuinka paljon informaatiota järjestelmästä on saatavilla.

Alustavassa riskien arvioinnissa (engl. *preliminary risk assessment*) päätöksiä järjestelmän vaatimuksista, suunnittelusta tai toteutuksesta ei vielä ole tehty eikä tietoa mahdollisista haavoittuvuuksista vielä ole. Tarkoituksena on ensin arvioida, oikeuttavatko järjestelmän kehittämisestä saadut hyödyt järjestelmän kehittämiseen

liittyvät riskit ja tämän jälkeen tuottaa yksityiskohtaiset turvallisuusvaatimukset toteutettavalle järjestelmälle. Arviointi toimii pohjana järjestelmäalustan ja väliohjelmistojen valinnalle sekä ohjelmiston tarkemmille toimintavaatimuksille.

Elinkaarellisessa riskien arvioinissa (engl. *life cycle risk assessment*) järjestelmän arkkitehtuuri ja tietorakenne ovat saatavilla ja järjestelmäalusta sekä väliohjelmistot on valittu. Arvointi tapahtuu järjestelmän kehityksen elinkaaren aikana ja se vaikuttaa järjestelmän teknisiin suunnittelu- ja toteutuspäätöksiin. Se antaa kehitysprosessille tietoa turvallisuusvaatimustekniikasta. Tunnetut ja mahdolliset haavoittuvuudet yksilöidään ja tämä tietämys antaa päätöksentekijöille tietoa siitä, kuinka järjestelmän toiminnallisuus toteutetaan, testataan ja otetaan käyttöön. [13, s. 722-727]

4 Tietoturvamallit

Luvussa määritellään mitä turvallisuustekniikkaan läheisesti liittyvillä tietoturvamalleilla tarkoitetaan, kerrotaan mallien historiasta, esitellään tietoturvamallikokoelmat, selvitetään mallien luokitusperiaatteet ja suhteet, käydään läpi tietoturvamallin rakenne ja esitellään nk. ydinmallit.

4.1 Yleiskatsaus tietoturvamalleihin

Kuten kappaleessa 2.3 todettiin, tietoturvatekniikat ovat nykyisin hyvin tunnettuja ja valmiita ohjelmistokehittäjille. Oikeuksientarkistamistekniikoiden ja salauskirjastojen olemassaolo ei kuitenkaan riitä, niitä on myös osattava hyödyntää ohjelmistoja kehitettäessä. Koska turvallisuutta on jälkikäteen hyvin vaikeaa lisätä, tulisi se tehdä jo ohjelmiston suunnitteluvaiheessa. Turvallisen ohjelman suunnittelu ei kuitenkaan ole yksinkertainen tehtävä ja se vaatii usein taitoja, joita kehitystiimillä ei normaalisti ole. Tätä tilannetta helpottamaan on kehitetty tietoturvamallit. [23]

Tietoturvamallit (engl. *security patterns*) ovat hyvin todistettuja valmiita malleja usein toistuviin tietoturvaongelmiin. Ne ovat kirjallinen keino ottaa talteen suunnitteluasiantuntijoiden oivallukset ja kokemus ja ilmaista ne aloittelijoille. [18] Tietoturvamalleista käytetään myös nimityksiä tietoturvan malliratkaisu ja turvamalliratkaisu [7].

Schumacher et al. [24] määrittelevät tietoturvamallin vapaasti suomennettuna seuraavalla tavalla:

> Tietoturvamalli kuvailee tietyn toistuvan tietoturvaongelman, joka ilmenee tarkoin määritetyissä asiayhteyksissä ja esittelee sille hyvin todistetun ratkaisun. Ratkaisu muodostuu yhdestä vuorovaikuttavasta roolijoukosta, joka voidaan järjestää moneen konkreettiseen suunnitelurakenteeseen, kuten myös prosessiksi luomaan yksi tällainen rakenne.

Tietoturvamallikokoelma (engl. *security pattern system*) on nimensä mukaisesti kokoelma tietoturvamalleja sisältäen ohjeet mallien toteuttamiseen, yhdistelyyn ja sopivaan käyttöön tietoturvallisten ohjelmistojen suunnittelussa ja toteutuksessa [6].

Malliajattelu pitää sisällään myös käsityksen mallikielestä (engl. *pattern language*), joka määritellään seuraavalla tavalla [24]:

> Mallikieli on verkko tiukasti yhteenpunoutuneita malleja, jotka määrittelevät menetelmän, jolla voidaan järjestelmällisesti ratkaista joukko yhteenkuuluvia, toisistaan riippuvaisia ohjelmistojen kehitysongelmia.

Mallikielet ovat nykytietämyksen valossa viimeisin ja täydellisin tapa soveltaa malleja. Tietoturvamalleille kunnollisia mallikieliä ei vielä ole. Mallikielen tulisi vastata seuraaviin kysymyksiin [24]:

- Mitkä ovat tärkeitä suunnittelu- ja toteutusongelmia, jotka esiintyvät kehittäessä tietyn tyyppistä sovellusjärjestelmää?
- Kuinka kaikki nämä ongelmat yhdistyvät toinen toisiinsa ja mikä on niiden paras mahdollinen ratkaisujärjestys?
- Mitkä vaihtoehtoiset mallit voivat auttaa ratkaisemaan kunkin ongelman tehokkaimmin muiden ongelmien läsnäollessa?
- Millä kriteereillä päätetään mikä vaihtoehtoisista malleista on soveltuvin ratkaisemaan nimenomaisen ongelman tietyssä tilanteessa?
- Kuinka valittu malli toteutetaan tehokkaimmin olemassaolevassa (vaillinaisessa) ohjelmistoarkkitehtuurissa?

4.2 Tietoturvamallien historia

Tietoturvamallien juuret juontavat 1970-luvun rakennusarkkitehtuurin ja kaupunkisuunnittelun periaatteisiin. Arkkitehti Christopher Alexander esitteli mallikäsitteen ensimmäisen kerran 1970-luvun lopussa julkaisemissa kirjoissaan [14] ja [15]. Alexander kehitti tiiminsä kanssa myös mallien konteksti-ongelma-ratkaisu rakenteen, joka tunnetaan myös Alexandrian rakenteena (engl. *Alexandrian form*) [3, s. 12].

Ward Cunningham ja Kent Beck julkaisivat vuonna 1987 raportin, jossa he — Christopher Alexanderin oppeja hyödyntäen — kertoivat käyttäneensä onnistuneesti viittä mallia käyttöliittymien suunnitteluun ja aikomuksestaan kuvata kokonainen mallikieli olioperustaisille ohjelmille [16].

Mallien kehittämiseen tähtäävät konferenssit, joita kutsutaan nimellä PLoP (engl. *Pattern Language of Programs*), saivat alkunsa vuonna 1994 [3, s. 13]. Vuonna 1995

Erich Gamma, Richard Helm, Ralph Johnson ja John Vlissides, jotka tunnetaan yhteisellä nimellä ”the Gang of Four” (GoF), julkaisivat kirjan [17], jossa esiteltiin oliopohjaiset ja uudelleenkäytettävät suunnittelumallit (engl. *design patterns*). Kirja sai aikaan malliajatuksen läpimurron [3, s. 13].

Joseph Yoder ja Jefferey Bercalow julkaisivat vuonna 1997 PLoP:n (mallien kehittämiskonferensin) yhteydessä ensimmäistä kertaa malleja, jotka olivat tarkoitettu tietoturvan parantamiseen [3, s. 99-101]. Tästä alkoi tietoturvaan liittyvien mallien aikakausi.

4.3 Tietoturvamallikokoelmat

Tähän mennessä on julkaistu ainakin neljä suurempaa tietoturvamallikokoelmaa [20] [23]:

- Security Patterns Repository [18], joka sisältää 26 mallia ja kolme mini-mallia. Mallit keskittyvät lähinnä verkkosovellusten turvallisuuteen. Malleista 13 ovat rakenteellisia, kuten myös mini-mallit, ja 13 malleista ovat proseduraalisia. Mini-mallit ovat lyhyttä vapaamuotoista turvallisuusasiantuntevaa keskustelua liittyen vain ongelmaan ja sen ratkaisuun. Rakenteellisia malleja voidaan hyödyntää itse sovellukseen sen suunnittelu-, arkkitehtuuri- ja toteutustasolla. Ne sisältävät usein rakennekaavioita ja vuorovaikutuskuvauksia. Proseduurisilla malleilla voidaan taas parantaa tietoturvakriittisen sovelluksen kehitysprosessia ja ne vaikuttavat yleensä kehitysprojektin organisointiin ja hallintaan.

- Security Design Patterns (SDP) technical guide [19], jossa on viisi ”saatavuusmallia” (engl. *available system patterns*) ja kahdeksan ”suojattua järjestelmämallia” (engl. *protected system patterns*). Saatavuusmallit huolehtivat ennustettavasta katkeamattomasta pääsystä järjestelmän tarjoamiin palveluihin ja resursseihin. Suojatut järjestelmämallit suojaavat arvokkaita resursseja luvattomalta käytöltä, julkitulolta tai muutokselta.

- Core Security Patterns [21], jossa on kahdeksan ”verkkokerrosmallia” (engl. *Web Tier Security Patterns*), seitsemän ”liiketoimintakerrosmallia” (engl. *Business Tier Security Patterns*), kolme ”verkkopalvelukerrosmallia” (engl. *Web Services Tier Security Patterns*) ja neljä ”henkilöllisyydenhallinta- ja palveluntoimitusmallia” (engl. *Identity Management and Service Provisioning*) [22]. Verkkokerroksessa ollaan vuorovaikutuksessa asiakkaan kanssa. Verkkokerrokseen kuuluvat mallit turvaavat infrastruktuurin ja sovelluksen asiakas-palvelin- ja

palvelin-palvelin -kommunikoinnin. Liiketoimintakerroksella eristetään liiketoimintalogiikka ja -prosessit. Liiketoimintakerroksen mallit vahvistavat liiketoimintalogiikkaan ja -prosesseihin kuuluvia turvallisuuspalveluita. Verkkopalvelukerroksella yhdistetään sovelluksen sisäinen ja ulkoinen infrastuktuuri. Henkilöllisyydenhallintakerrosella ilmoitetaan todennettu henkilöllisyys asianmukaisille palveluntarjoajille infrastuktuurissa [21]. Kokoelman mallit on suunnattu erityisesti Java-ympäristöön, mutta kokoelman malleja voidaan solveltaa myös yleiskäyttöisesti, kuten tietoturvamallikokoelmien teknillisessä selonteossa ”A system of security patterns” [23] on tehty.

- Security Patterns [24], joka käsittää noin 50 mallia. Mallien käyttämistä lähestytään arkkitehtuurin ja suunnittelun lisäksi myös riskin arvioinnin ja pienentämisen kannalta [23].

Tässä tutkielmassa käytetään tietoturvamallien pääasiallisena lähteenä teknillistä selontekoa, eräänlaista tietoturvamallijärjestelmää, A system of security patterns [23]. Tässä vuoden 2006 loppupuolella kirjoitetussa selonteossa on yksinkertaistettu mallin valitsemisprosessia järkiperäistämällä kaikki kirjoitushetkellä julkisesti saatavissa olleiden tietoturvamallikokoelmien mallit. Tämä on saatu aikaan karsimalla löydetystä 80 mallin joukosta päällekkäiset ja huonosti määritellyt tai kuvatut mallit. Mallit on myös luokiteltu sen mukaan mihin ohjelmiston kehitysvaiheeseen ne kuuluvat. Lopputuloksena on saatu 35 ”ydinmallia” (engl. *core patterns*), kts. taulukko 4.1. Toisin kuin muilla malleilla, ydinmalleilla ohjelmistoarkkitehdit ja -suunnittelijat voivat lisätä sovelluksen turvallisuutta suunnitteluvaiheessa. Kaikki tässä tutkielmassa käytetyt tietoturvamallit kuuluvat tietoturvamallijärjestelmän ydinmalleihin.

Ydinmalleille on määritelty yhtenäinen meta-informaatio helpottamaan oikean mallin valitsemista. Kuhunkin ydinmalliin on liitetty turvallisuustavoitteet (luottamuksellisuus, kontrolloitu pääsy, saatavuus jne.), jotka se täyttää ja joiden avulla mallin valitsemisperusteet voidaan jäljittää. Lisäksi ydinmalliin on liitetty lisähuomautuksia, joilla kuvataan sen sivuvaikutukset muihin ohjelman ominaisuuksiin. Näiden sivuhuomautusten avulla voidaan löytää hyvä kompromissi ydinmallien väliltä. Lopuksi ydinmallien väliset suhteet on esitetty selkeästi. [23]

4.4 Tietoturvamalliluokitukset

Tietoturvamalliluokitukset esitellään tässä tutkielmassa lähteen [23] ydinmalleille tehdyn luokituksen mukaisesti. Ydinmallit on luokiteltu kolmella eri tavalla, mitkä

käydään seuraavissa alaluvuissa läpi.

4.4.1 Kehitysvaiheluokitus

Kehitysprosessivaiheeseen perustuvassa luokituksessa jaetaan ydinmallit karkeasti kahteen ryhmään. Ensimmäiseen ryhmään kuuluvat mallit, jotka vaikuttavat vain sovelluksen ympäristöön, kuten infrastuktuuriin tai väliohjelmistoihin. Toinen ryhmä pitää sisällään mallit, jotka vaikuttavat itse sovelluksen suunnitteluun, ja jotka jaetaan edelleen osajoukkoihin sen perusteella, toimivatko ne paremmin korkean tason arkkitehtuurivaiheessa vai yksityiskohtaisemmassa suunnitteluvaiheessa [23]:

- Sovellusarkkitehtuuriryhmä: Mallien käytöllä on järjestelmänlaajuisia seuraamuksia. Tällaiset mallit tuovat sovelluksen uusia komponentteja, muuttavat olemassa olevia komponentteja tai lisäävät riippuvuuksia laajassa osassa sovellusta. Ryhmään kuuluu 12 ydinmallia.
- Sovellussuunnitteluryhmä: Mallien käytöllä on vain paikallisia seuraamuksia. Malli vaikuttaa vain pieneen osajoukkoon sovelluksen yksityiskohtaiseen suunnitteluun liittyvistä elementeistä. Ryhmään kuuluu 11 ydinmallia.
- Järjestelmäryhmä: Mallin vaikutukset kohdistuvat pääasiassa sovelluksen ympäristöön ja parhaimmassa tapauksessa vain siihen. Ryhmään kuuluu 12 ydinmallia.

4.4.2 Turvallisuustavoiteluokitus

Luokittelu perustuu turvallisuustavoitteisiin (kts. luku 3.5). Luottamuksellisuuden, eheyden, saatavuuden ja vastuullisuuden lisäksi on lähteessä [23] määritelty kolme alitavoitetta, jotka auttavat saavuttamaan tavoitteet. Alitavoitteita ovat:

- Turvallinen datan välitys (engl. *secure data transmission*): On luottamuksellisuuteen ja eheyteen liittyvä alitavoite, missä pyritään suojaamaan siirrettävä tieto.
- Valvottu sisäänpääsy (engl. *controlled Access*): On luottamuksellisuuteen ja eheyteen liittyvä alitavoite, missä pyritään rajoittamaan resurseihin pääsy vain valtuutetuille tahoille.
- Tunnistus (engl. *identification*): On vastuullisuuteen liittyvä alitavoite, missä pyritään varmistamaan käyttäjän henkilöllisyys. Tunnistus-alitavoite on välttämätön alitavoitteelle ”valvottu sisäänpääsy”.

Kaikkiin näihin turvallisuustavoitteisiin ja alitavoitteisiin liittyy ydinmalleja. Luottamuksellisuuteen liittyvät mallit kuuluvat automaattisesti myös alitavoitteisiin ”turvallinen datan välitys” ja ”valvottu sisäänpääsy”. Huomion arvoista on, että yksikään ydinmalli ei liity kiistämättömyyteen, nimettömyyteen tai yksityisyyteen (kts. luku 3.5).

4.4.3 Piirreluokitus

Piirreluokituksessa ydinmallit on luokiteltu sen mukaan, mihin piirteisiin mallilla on suotuisa tai vahingollinen vaikutus. Mallin suotuisa tai vahingollinen vaikutus piirteisiin saadaan vertailemalla mallin toteuttavaa järjestelmää sellaiseen vastaavalla turvallisuustoiminnoilla rakennettuun järjestelmään, jossa mallia ei ole otettu huomioon. Piirteisiin kuuluu turvallisuustavoitteet (kts. kappale 4.4.2) ja muita turvallisuuteen liittyviä ja liittymättömiä ominaisuuksia. [23]

Turvallisuustavoitteisiin kuulumattomia piirteitä ovat [23]:

- Käyttövarmuus (engl. *dependability*) tarkoittaa kykyä tarjota palvelua, johon voidaan oikeutetusti luottaa. So. järjestelmän kyky välttää sellaisia häiriöitä, jotka olisivat käyttäjän hyväksynnän kannalta liian toistuvia, vakavia tai pitkän katkoksen aiheuttavia [30].

- Siirrettävyys (engl. *portability*) tarkoittaa ohjelmistotuotteen kyvykkyyttä siirtyä organisaatio-, laitteisto- tai ohjelmistoympäristöstä toiseen [29].

- Ylläpidettävyys (engl. *maintainability*) ilmaisee ohjelmistotuotteen muunneltavuutta. Muunnoksia voivat olla ohjelmatason korjaukset, parannukset ja mukauttamiset sekä ympäristö-, toiminnallisuus- ja vaatimusmäärittelytason muutokset [29].

- Suoritusteho (engl. *performance*, *efficiency*) ilmaisee ohjelmistotuotteen kyvykkyyttä antaa asianmukaista tehoa suhteessa käytettyihin resursseihin määrätyssä tilanteessa [29].

- Käytettävyys (engl. *usability*) ilmaisee ohjelmistotuotteen kykyä tulla ymmärretyksi, opituksi, käytetyksi ja kiinnostavaksi käyttäjälle, kun sitä käytetään määrätyissä tilanteissa. Se ilmaisee kuinka helppoa käyttäjän on operoida, valmistella syötteitä ja tulkita tuloksia järjestelmästä tai komponentista [31].

- Käsiteltävyys (engl. *manageability*) ilmaisee kyvykkyyttä tulla käytetyksi tai kontrolloiduksi [32].

- Tarkastettavuus (engl. *auditability*) ilmaisee kyvykkyyttä näyttää toteen järjestelmän asianmukainen toiminta menneisyydessä ja nykyhetkessä [33].
- Kustannus (engl. *cost*) kuvaa omaisuuden tai palveluiden rahallista, ajallista ja työnteollista kokonaiskäyttöä [32].

4.5 Tietoturvamallien väliset suhteet

Lähteessä [23] on eroteltu viisi mallien välistä suhdetta (relaatiota), jotka positiivisimmasta negatiivisimpaan ovat: riippuu (engl. *depends*), hyötyy (engl. *benefits*), on vaihtoehtoinen (engl. *alternative*), haittaa (engl. *impairs*) ja on ristiriidassa (engl. *conflicts*).

Suhteille käytetään relaatiosääntöjä. Olkoon a, b ja c malleja, jotka kuuluvat mallijoukkoon A. Mallien välien suhde on symmetrinen, jos kaikille a ja b pätee: kun a on relaatiossa b:n kanssa, niin myös b on relaatiossa a:n kanssa. Tämä merkitään matemaattisesti seuraavasti:

$$\forall a, b \in \mathrm{A} | (a \mathsf{R} b) \Rightarrow (b \mathsf{R} a).$$

Mallien välinen suhde on transitiivinen, jos kaikille a, b ja c pätee: kun a on relaatiossa b:n kanssa ja b on relaatiossa c:n kanssa, niin a on relaatiossa c:n kanssa. Tämä merkitään matemaattisesti seuraavasti:

$$\forall a, b, c \in \mathrm{A} | ((a \mathsf{R} b) \wedge (b \mathsf{R} c)) \Rightarrow (a \mathsf{R} c).$$

Mallien väliset suhteet ovat seuraavat:

- Riippuu: suhteista lujin, sillä jos malli A on riippuvainen mallista B, niin A ei toimi oikein ilman B:tä. Suhde on transitiivinen, mutta ei symmetrinen.
- Hyötyy: malli A hyötyy mallista B, jos mallin B toteutus antaa lisäarvoa jo toteutetulle mallille A. Suhde ei ole transitiivinen eikä myöskään yleensä symmetrinen.
- On vaihtoehtoinen: malli A on vaihtoehtoinen mallille B, jos se vastaa toiminnaltaan B:tä. Malli A voidaan korvata mallilla B ja toisinpäin, ilman että se vaikuttaisi järjestelmän kokonaiskäyttäytymiseen tai -laatuun. Suhde on symmetrinen ja transitiivinen.

- Haittaa: malli A haittaa mallia B, jos A:lle kuuluva toiminta vaikeutuu toteuttamalla malli B. Sekä malli A että B voidaan toteuttaa, mutta toteutus on tehtävä huolella virhetilanteiden välttämiseksi. Suhde on symmetrinen, mutta ei transitiivinen.

- On ristiriidassa: malli A on ristiriidassa mallin B kanssa, jos B:n toteuttaminen A:n sisältämään järjestelmään aiheuttaa ristiriidan. Suhde on symmetrinen, mutta ei transitiivinen.

4.6 Tietoturvamallin rakenne

Tietoturvamallien yhtenäinen rakenne pitää sisällään kuvaavan nimen, kontekstin ja sukulaismallit, ongelman ja ratkaisun. Rakenteen pohjalta niistä haetaan tietoa ja niitä vertaillaan [6].

Tietoturvamallin pohja on perusta, jonka päälle malli rakentuu. Tässä tutkielmassa käytetään tietoturvamallipohjana lähteen [23] ”ydinmalleille” tehtyä pohjaa. Ydinmallien pohja perustuu Gang of Four -ryhmän suunnittelumallien pohjaan [17], ja laajentaa sitä hieman. Ydinmallin pohja koostuu kahdesta osasta. Ensimmäisessä osassa annetaan yleiskuva mallista, jonka avulla voidaan nopeasti arvioida soveltuuko malli käsillä olevaan ongelmaan. Pohjan ensimmäinen osa pitää sisällään seuraavat kohdat:

- Mallin nimi: kuvataan tietoturvamalli lyhesti, mutta selkeästi.

- Tarkoitus: kerrotaan mallin tarkoitus tiivistettynä. Kohdan tulisi vastata kysymyksiin: Mikä on mallin tarkoitus? Minkä ongelman malli ratkaisee?

- Tunnetaan myös (valinnainen): mainitaan muut nimet, joilla malli yleisesti tunnetaan.

- Soveltuvuus: kuvataan missä tapauksissa mallia voidaan käyttää. Kuvaus sisältää mallin vaikutusalueen ja kehitysvaiheen, johon se soveltuu parhaiten.

- Turvallisuustavoitteet: kerrotaan mallin pääturvallisuustavoite. Useiden tavoitteiden antaminen on sallittua, mutta sitä tulisi välttää. Jos malli vaikuttaa muihin kuin päätavoitteeseensa, voidaan se mainita nimiöt-kohdassa.

- Nimiöt (engl. *labels*): kuvataan mallin vaikutuksia (positiivisia ja negatiivisia) tehokkuuteen, käytettävyyteen sekä muihin piirteisiin ja turvallisuustavoitteisiin.

- Suhteet: annetaan lyhyt tiivistelmä mallien välisistä suhteista. Suhteet kuvataan tarkemmin mallipohjan toisessa osassa.

Mallipohjan toinen osa sisältää mallin yksityiskohtaisen kuvauksen. Tämä osa muistuttaa Gang of Four -ryhmän suunnittelumallipohjaa [17], ja pitää sisällään seuraavat kohdat:

- Ongelma: kuvataan ongelma, johon malli tarjoaa ratkaisun. Ongelman kuvauksessa voidaan mainita asiat, jotka johtavat ongelmaan.
- Esimerkki: esitetään esimerkki mallin käyttämisestä. Esimerkissä annetulla helpolla tapauksella olisi pystyttävä kartoittamaan tuleva ratkaisu.
- Ratkaisu: kuvataan täydellisesti ja yksityiskohtaisesti mallin tarjoama ratkaisu. Kuvaus sisältää ratkaisun staattisen rakenteen ja dynaamisen toiminnan (mieluummin graafisessa esitysmuodossa, kuten UML), osalliset vastuineen sekä yhteistoiminnan osallisten välillä.
- Toteutus (valinnainen): annetaan vinkkejä ja ideoita toteutukseen; voidaan sisällyttää esimerkkikoodia. Mahdolliset vaihtoehtoiset toteutustavat tulisi mainita.
- Piilevät vaarat (valinnainen): kerrotaan mallin käyttöön mahdollisesti liittyvistä piilevistä vaaroista ja riskeistä. Vapaavalintaisesti sisällytetään näille ratkaisu.
- Seuraamukset: kerrotaan mallin käytön tarjoamista eduista ja mahdollisista haitoista. Seuraamuksiin sisältyvät myös perusteellisemmat kuvaukset mallipohjan ensimmäisen osan nimiöistä.
- Sukulaismallit: merkitään ylös mallien väliset suhteet. Suhteista on kerrottu tarkemmin luvussa 4.5.
- Tunnetut käyttökohteet: mainitaan tunnettuja onnistuneita käyttökohteita mallille.

4.7 Ydinmallit

Taulukossa 4.1 on esitelty lyhyesti kaikki 35 ydinmallia ja kerrottu kuhunkin malliin liittyvät tavoitteet, suhteet ja piirteet.

Taulukko 4.1: Ydinmallit

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
Sovellusarkkitehtuuriryhmän mallit					
1	Authentication Enforcer [21]	Valvottu sisäänpääsy	Hyötyy 33, 12 Vaihtoehto 5	−Nimettömyys +Ylläpidettävyys +Käsiteltävyys +Tarkastettavuus +Yksityisyys +Vastuullisuus +Siirrettävyys	You need to verify that each service request is from an authenticated entity.
2	Authorization Enforcer [21]	Valvottu sisäänpääsy	Hyötyy 1, 12 Vaihtoehto 5	+Ylläpidettävyys +Käsiteltävyys +Tarkastettavuus +Vastuullisuus +Siirrettävyys	Verify that requests for services are properly authorized at the method and link level.
3	Checkpointed System [19]	Eheys	Hyötyy 4 Haittaa 26	−Suoritusteho −Kustannus +Käyttövarmuus	Structure a system so that its state can be recovered and restored to a known valid state in case a component fails.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
4	Comparator-Checked Fault-Tolerant System [19]	Eheys		–Suoritusteho –Kustannus +Käyttövarmuus	Structure a system so that an independent failure of one component will be detected quickly and so that an independent single-component failure will not cause a system failure.
5	Container-Managed Security [21]	Valvottu sisäänpääsy Tunnistus	Vaihtoehto 2, 1	+Käsiteltävyys +Siirrettävyys +Eheys	You need a simple, standard way to enforce authentication and authorization in your J2EE applications and don't want to reinvent the wheel or write home-grown security code. Using a Container Managed Security pattern, the container performs user authentication and authorization without requiring the developer to hard-wire security policies in the application code.
6	Credential Tokenizer [21]	Tunnistus	Hyötyy 33	+Siirrettävyys +Käsiteltävyys +Käytettävyys	You need a flexible mechanism to encapsulate a security token that can be used by different security infrastructure providers.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
7	Input Guard [38]	Eheys	Hyötyy 8 Vaihtoehto 25	–Suoritusteho +Tarkastettavuus	Protect components from input that does not conform to the system specification.
8	Output Guard [38]	Eheys	Hyötyy 7	–Suoritusteho +Tarkastettavuus	Confine an error in the component that contains the fault which led to that error.
9	Replicated System [19]	Saatavuus	Riippuu 30	–Käsiteltävyys –Kustannus	Structure a system which allows provision of service from multiple points of presence, and recovery in case of failure of one or more components or links.
10	Secure Access Layer [37]	Eheys		+Ylläpidettävyys +Siirrettävyys	Application security will be insecure if it is not properly integrated with the security of the external systems it uses. On top of the lower-level security, build a secure access layer for communicating in and out of the program.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
11	Secure Logger [21]	Vastuullisuus	Hyötyy 33	−Suoritusteho +Luottamuksellisuus +Eheys +Ylläpidettävyys +Käsiteltävyys	Application events must be logged in a centralized way, and it should be impossible to alter log files.
12	Secure Service Facade [21]	Valvottu sisäänpääsy		+Tunnistus +Auditointi +Käsiteltävyys +Ylläpidettävyys	You need a secure gateway mandating and governing security on client requests, exposing a uniform, coarse-grained service interface over fine-grained, loosely coupled business services that mediates client requests to the appropriate services.
Sovellussuunnitteluryhmän mallit					
13	Controlled Object Factory [24]	Eheys	Riippuu 14	−Suoritusteho +Käsiteltävyys	This pattern addresses how to specify the rights of processes with respect to a new object. When a process creates a new object through a factory, the request includes the features of the new object. These features include a list of rights to access the object.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
14	Controlled Object Monitor [24]	Valvottu sisäänpääsy	Riippuu 2 Hyötyy 1	+Ylläpidettävyys +Käsiteltävyys +Tarkastettavuus +Vastuullisuus	This pattern addresses how to control access by a process to an object. Use a reference monitor to intercept access requests from processes. The reference monitor checks whether the process has the requested type of access to the object.
15	Full View with Errors [37]	Valvottu sisäänpääsy	Vaihtoehto 16 Ristiriita 16	−Käytettävyys +Ylläpidettävyys	Prevent users to perform illegal operations by showing an error message when the user tries to perform an illegal operation.
16	Limited View [37]	Valvottu sisäänpääsy	Vaihtoehto 15 Ristiriita 15	+Käytettävyys	Prevent users to perform illegal operations by hiding all operations that cannot be performed by the user.
17	Obfuscated Transfer Object [21]	Turvallinen datan välitys		+Käsiteltävyys +Ylläpidettävyys +Siirrettävyys	You need a way to protect critical data as it is passed within application and between tiers.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
18	Secure Session Object [21]	Tunnistus	Riippuu 21	+Siirrettävyys +Luottamuksellisuus +Suoritusteho	You need to facilitate distributed access and seamless propagation of security context and client sessions in a platform-independent and location-independent manner.
19	Security Association [19]	Turvallinen datan välitys	Riippuu 33, 20	+Suoritusteho	Define a structure which provides each participant in a secure communication with the information it will use to protect messages to be transmitted to the other party, and with the information which it will use to understand and verify the protection applied to messages received from the other party.
20	Security Context [24]	Tunnistus	Riippuu 24 Hyötyy 19	+Valvottu sisäänpääsy +Ylläpidettävyys	Provide a container for security attributes and data relating to a particular execution context, process, operation, or action.
21	Session [37]	Tunnistus	Hyötyy 23, 18, 22	–Nimettömyys –Yksityisyys +Käytettävyys	Many objects need access to shared values, but the values are not unique throughout the system.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
22	Session Failover [39]	Saatavuus	Riippuu 30, 21	–Suoritusteho –Kustannus	Avoid inconveniencing users that lose session data in a system restart.
23	Session Timeout [39]	Saatavuus	Riippuu 21	–Käytettävyys –Yksityisyys +Kustannus	Prevent the system from running out of resources because abandoned sessions are not cleaned up.
24	Subject Descriptor [24]	Tunnistus	Hyötyy 20	+Valvottu sisäänpääsy	Provide access to security-relevant attributes of an entity on whose behalf operations are to be performed.
Järjestelmäryhmän mallit					
25	Application Firewall [35]	Valvottu sisäänpääsy	Vaihtoehto 31, 7	–Suoritusteho +Käsiteltävyys +Tarkastettavuus	To filter calls and responses to/from enterprise applications, based on an institution access control policies.
26	Audit Interceptor [21]	Vastuullisuus	Riippuu 11 Hyötyy 12	–Suoritusteho +Tarkastettavuus +Ylläpidettävyys +Käsiteltävyys	You want to intercept and audit requests and responses to and from the Business tier, in a flexible and modifyable way.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
27	Controlled Process Creator [24]	Eheys		−Suoritusteho +Käsiteltävyys +Käyttövarmuus	This pattern addresses how to define and grant appropriate access rights for a new process, in an operating system in which processes or threads need to be created according to application needs.
28	Demilitarized Zone [24]	Valvottu sisäänpääsy	Riippuu 29	−Suoritusteho −Kustannus −Käsiteltävyys −Käyttövarmuus +Ylläpidettävyys	Any organization conducting e-commerce or publishing information over Web technologies must make their service easily accessible to their users. However, any form of Web site or e-commerce system is a potential target for attack, especially those on the Internet. A Demilitarized Zone (DMZ) separates the business functionality and information from the Web servers that deliver it, and places the Web servers in a secure area. This reduces the ”surface area” of the system that is open to attack.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
29	Firewall [36]	Valvottu sisäänpääsy	Hyötyy 28, 25 Haittaa 31	−Suoritusteho −Käyttövarmuus +Vastuullisuus	Control incoming and outgoing network connections, restrict access to certain hosts on the network level.
30	Load Balancer [39]	Saatavuus	Hyötyy 9, 22 Vaihtoehto 31 Haittaa 21	+Suoritusteho	Distribute the load from multiple users over several servers.
31	Reverse Proxy [40]	Saatavuus	Riippuu 28 Haittaa 33	−Ylläpidettävyys +Käsiteltävyys +Eheys	Protect your web server infrastructure on an application protocol level, without hindering accessibility.
32	Secure Message Router [21]	Turvallinen datan välitys		−Saatavuus +Käsiteltävyys +Ylläpidettävyys +Luottamuksellisuus +Eheys	Securely communicate with multiple partner endpoints using message-level security and identity-federation mechanisms.

Jatkuu seuraavalla sivulla...

Taulukko 4.1 – Jatkuu

Nro	Nimi	Tavoitteet	Suhteet	Piirteet	Kuvaus
33	Secure Pipe [21]	Turvallinen datan välitys	Hyötyy 19		You need to provide privacy and prevent eavesdropping and tampering of client transactions caused by man-in-the-middle attacks.
34	Server Sandbox [18]	Eheys		–Käsiteltävyys –Suoritusteho –Kustannus	Many site defacements and major security breaches occur when a new vulnerability is discovered in the Web server software. Yet most Web servers run with far greater privileges than are necessary. The Server Sandbox pattern builds a wall around the Web server in order to contain the damage that could result from an undiscovered bug in the server software.
35	Single Access Point [37]	Valvottu sisäänpääsy		+Käsiteltävyys	Reduce the ”attack surface” by imposing a single access point on the system, providing an ideal place to do access control and policy enforcement.

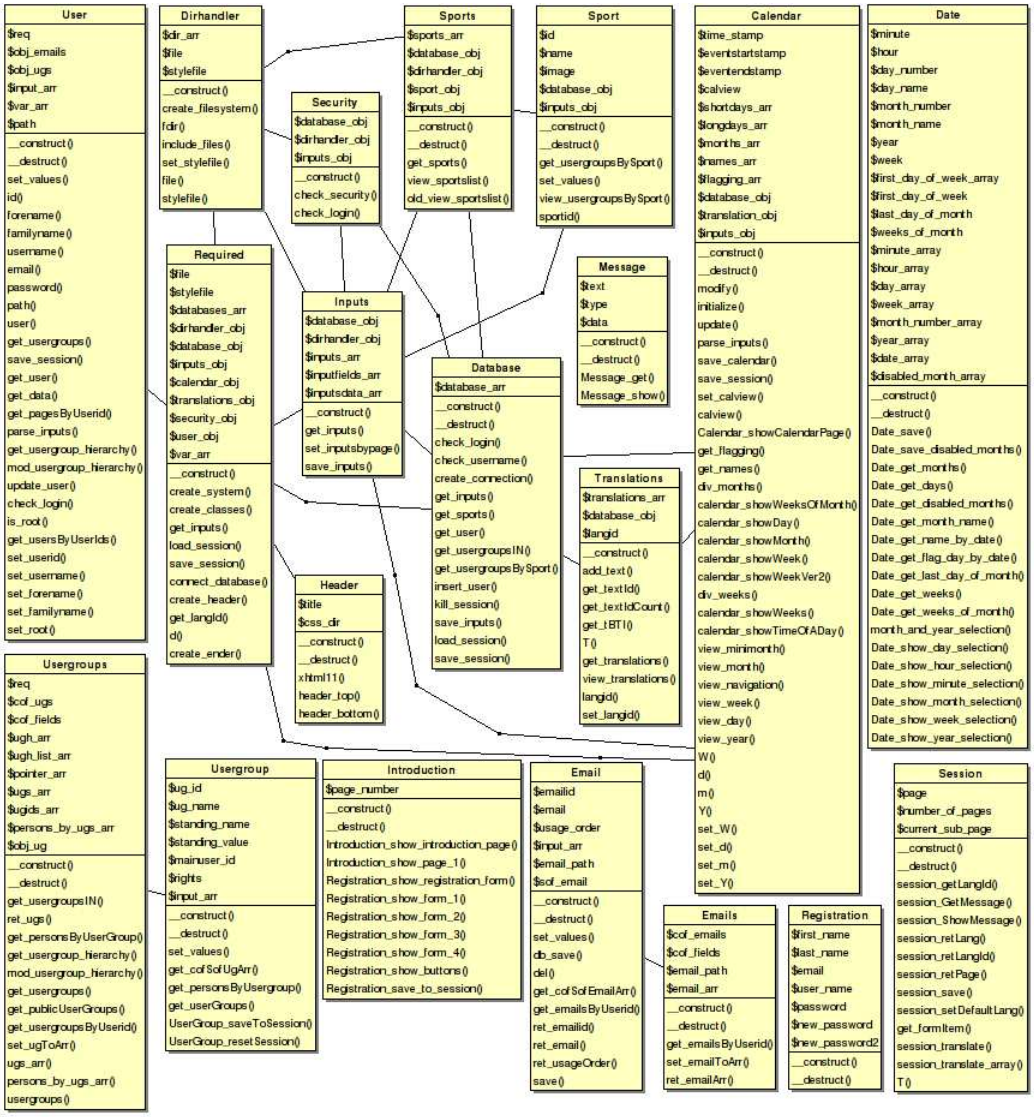
5 Liikuntakalenteri

Luvussa kerrotaan tutkielman esimerkkisovelluksena toimivasta liikuntakalenterisovelluksesta, tarkastellaan sitä turvallisuustekniikan näkökulmasta ja tehdään sille tietoturva-analyysi tietoturvamalleja silmälläpitäen.

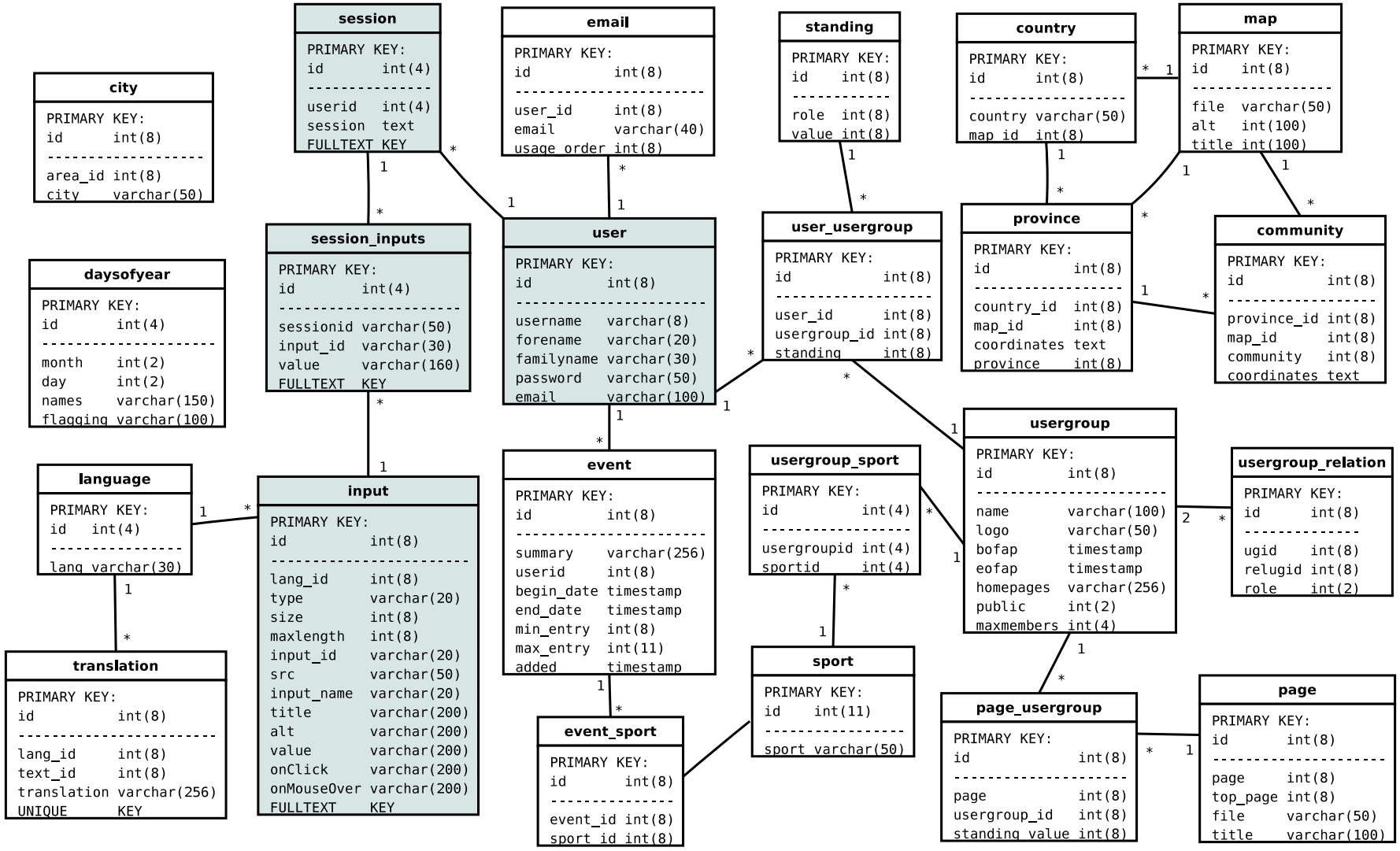
5.1 Liikuntakalenterisovellus

Liikuntakalenterisovelluksen on suunnitellut ja toteuttanut Raimo Pitkänen, ja hän on käyttänyt sitä esimerkkisovelluksena joulukuussa 2006 valmistuneessa pro gradu -tutkielmassa [26]. Liikuntakalenteri on sovellus, jolla yksittäishenkilöt tai ryhmät voivat luoda liikuntatapahtumia kalenteriin ja ilmoittautua niihin. Sovellus on WWW-pohjainen ja se on toteutettu PHP-kielellä olio-ohjelmoinnin periaatteiden mukaisesti. MySQL-pohjaisessa tietokannassa on 23 taulua [27]. Liikuntakalenterisovellus käyttää sessioita, mutta ei evästeitä [28]. Sovellus ei vielä ole valmiiksi asti toteutettu, mutta se on hyvin pitkälle suunniteltu [26]. Tässä tutkielmassa sovelluksesta käytetään pro gradu -versiota, johon on lisätty käytettävyystestien jälkeen tehdyt heuristiseen asiantuntija-arvioon perustuneet muutokset [28]. Sovellus toimii yhdellä palvelimella, jossa sijaitsevat niin Apache HTTP-palvelinohjelmisto kuin tietokantakin.

Liikuntakalenterin luokkakaavio on esitetty kuvassa 5.1 ja tietokannan kuvaus kuvassa 5.2.



Kuva 5.1: Liikuntakalenterin luokkakaavio



Kuva 5.2: Liikuntakalenterin tietokantakuvaus

Jotta luokkien toiminnasta saataisiin konkreettisempi käsitys, esitetään seuraavaksi liikuntakalenterissa oleva sovellustekninen toiminnallisuus, joka mahdollistaa WWW-sivulla näkyvien syöttökenttien sessiokohtaisen tallennuksen ja päivittämisen tietokantaan. Mukana tässä toiminnallisuudessa on luokkakaaviossa (kts. kuva 5.1) esitetyt luokat Required, Inputs ja Database. Toiminta on esitetty kuvan 5.3 sekvenssikaaviossa ja se etenee seuraavasti:

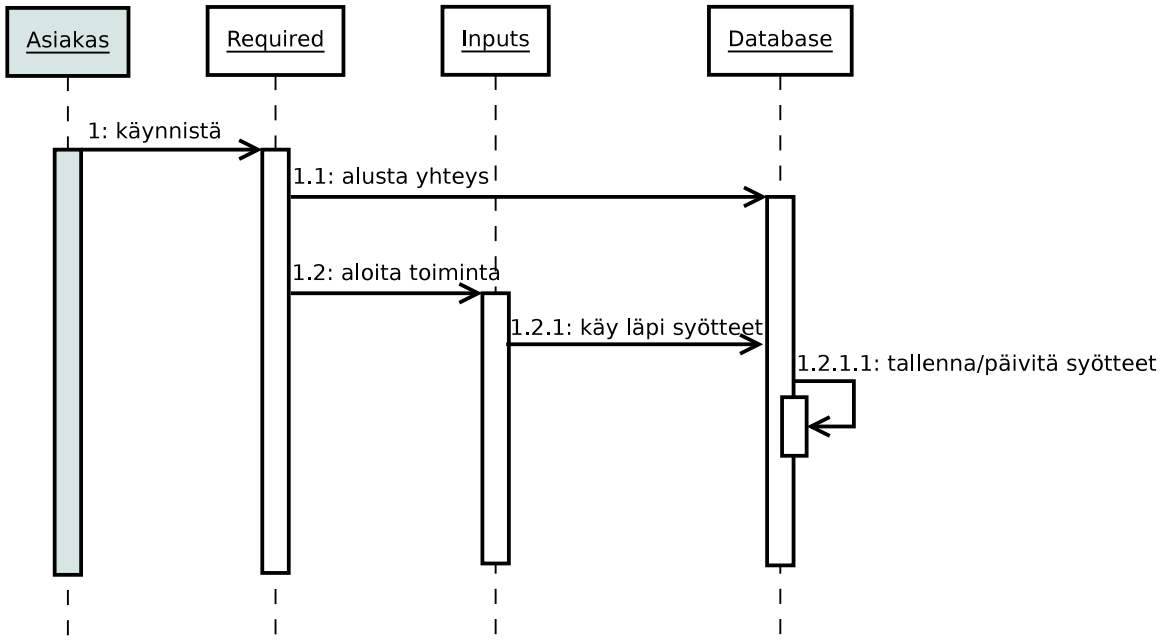
1. Asiakas lataa tai päivittää syöttökenttiä sisältävän www-sivun sisällön ja käynnistää samalla sovelluksen pääluokan Required.

2. Required-luokka alustaa tietokantayhteyden Database-luokan avulla.

3. Required-luokka pyytää Inputs-luokkaa toimimaan.

4. Inputs-luokka välittää Database-luokalle pyynnön syötteiden läpikäyntiä varten.

5. Database-luokka tekee sessio-tasolla kullekin syötteelle uuden session_inputs-taulun tietueen tai päivittää olemassa olevaa syötettä.



Kuva 5.3: Sekvenssikaavio, joka kuvaa syöttökenttien tallennusta ja päivitystä tietokantaan

5.2 Liikuntakalenteri turvallisuustekniikan näkökulmasta

Liikuntakalenterisovellus toimii WWW-pohjaisesti, ja se on siksi suurella riskivyöhykkeellä joutua hyökkäyksen kohteeksi. Sovelluksen suosion kasvaessa myös hyökkäyksen todennäköisyys kasvaa. Myös saatavuus-turvallisuustavoite korostuu luottamuksellisuuteen ja eheyteen verrattuna, sillä palvelunestohyökkäykset ovat yleisempiä suosituilla sivustoilla. Vaikka liikuntakalenteri ei vielä sisälläkään hyvin arkaluontoista tietoa, voi sitä tulevaisuudessa sovelluksen monimutkaistuessa järjestelmään tulla. Ei siis pidä aliarvioida riskejä, vaan pitää turvallisuustavoitteet korkealla alusta alkaen.

Sovellusta käyttävät pääasiassa urheilusta kiinnostuneet ihmiset. Käyttäjien joukossa voi siis olla täysin tietotekniikasta tietämättömiäkin. Tietoturvaominaisuudet eivät saa vaikeuttaa ainakaan oleellisesti järjestelmän käyttämistä. Järjestelmän käyttämisen tulisi olla mahdollisimman yksinkertaista. Sovelluksen käyttö jakautunee melko tasaisesti eri kellonaikoihin. On kuitenkin pidettävä mielessä mahdolliset käyttäjäpiikit varsinkin päiväsaikaan. Yksi palvelinkone voi suurilla käyttäjämäärillä alkaa tuottamaan ongelmia esimerkiksi liian suurien vasteaikojen muodossa.

Kuten kaikkien muidenkin sovellusten, niin myös liikuntakalenterisovelluksen tietoturvaa tarkasteltaessa tulee kiinnittää huomio itse sovelluksen lisäksi myös sen infrastruktuurin turvallisuuteen, tässä tapauksessa siis käyttöjärjestelmän, PHP-kielen, Apache HTTP-palvelinohjelmiston ja MySQL-tietokannan haavoittuvuuksiin. Tällaiset haavoittuvuudet johtuvat yleensä väärästä konfiguroinnista tai vanhoista ohjelmien versioista. On siis huolehdittava, että palvelinkoneeseen on asennettu uusimmat versiot kaikista tarvittavista sovelluksista ja tarkistettava sovellusten asetukset.

Liikuntakalenterisovelluksessa on jo olemassa turvallisuuteen liittyviä asioita. Autentikontimenetelmänä sovelluksessa toimii käyttäjätunnus ja salasana. Lisäksi sovelluksen käyttöä rajataan käyttäjäryhmien ja ryhmäkohtaisen oikeuksienhallinnan avulla [26].

5.2.1 Liikuntakalenterissa käytetyistä tietoturvatekniikoista

Liikuntakalenterissa suojataan käyttäjän salasana ennen sen tallentamista tietokantaan MD5-tiivistealgoritmilla [28]. MD5-tiivistealgoritmi (engl. *MD5 hash function*) muuntaa sille toimitetun informaation 128-bittiä pitkäksi tiivisteeksi. Tiivisteen avulla voidaan varmistaa tiedon eheys esimerkiksi tiedonsiirrossa. Muunnos on yksisuuntainen, eikä alkuperäistä syötettä pysty tulkitsemaan millään tavoin tiivisteestä [9, s. 77].

Yksisuuntaisuuden lisäksi tiivistealgoritmien tavoite on, että sama tiiviste eri syötteille esiintyisi äärimmäisen harvoin. Wang Xiaoyun ja Yu Hongbon [43] esittelemillä menetelmillä pystytään löytämään noin puolessa tunnissa kaksi syötettä, joille MD5-tiivistealgoritmi antaa saman tiivisteen. MD5-algoritmi sijoittuu tutkimuksessa tiivistealgoritmien keskivälille, kun mitataan niiden tehokkuutta.

Liikuntakalenterin yhteydessä MD5-tiiviste suojaa käyttäjien varsinaiset salasanat aina järjestelmän ylläpitäjiä myöten ja parantaa huomattavasti sovelluksen turvallisuustasoa.

Liikuntakalenterissa on valmius HTTPS-protokollan käyttämiseen ja tarvittava testaus protokollan käyttöön on jo tehty [28]. HTTPS-protokolla lisää TLS- tai SSL-salauskerroksen HTTP- ja TCP/IP-protokollan väliin ja turvaa asiakkaan ja palvelimen välisen liikenteen salakuuntelulta [44].

5.2.2 Liikuntakalenteri ja suojeltavat edut

Liikuntakalenteri ei sisällä esimerkiksi käyttäjän tilitietoja tai muuta tietoa, jonka vaarantuminen aiheuttaisi aineellista haittaa käyttäjälle. Liiikuntakalenterista voidaan kuitenkin löytää useita etuja, joita tulisi suojella. Niitä ovat ainakin:

- Järjestelmään tallennetut käyttäjien henkilötiedot (varsinkin käyttäjätunnus ja salasana) ja muut henkilökohtaiset asiat, mitkä tulisi suojella ulkopuolisilta tahoilta.
- Järjestelmään tallennetut ryhmäkohtaiset tiedot, joita ei tulisi muiden ryhmien nähdä.
- Järjestelmän saumaton sisäänpääsy, joka tulisi taata myös ruuhka-aikoina.
- Järjestelmän tietokannan eheys, joka tulisi säilyä joka tilanteessa.

5.3 Tietoturvamallit osaksi liikuntakalenteria

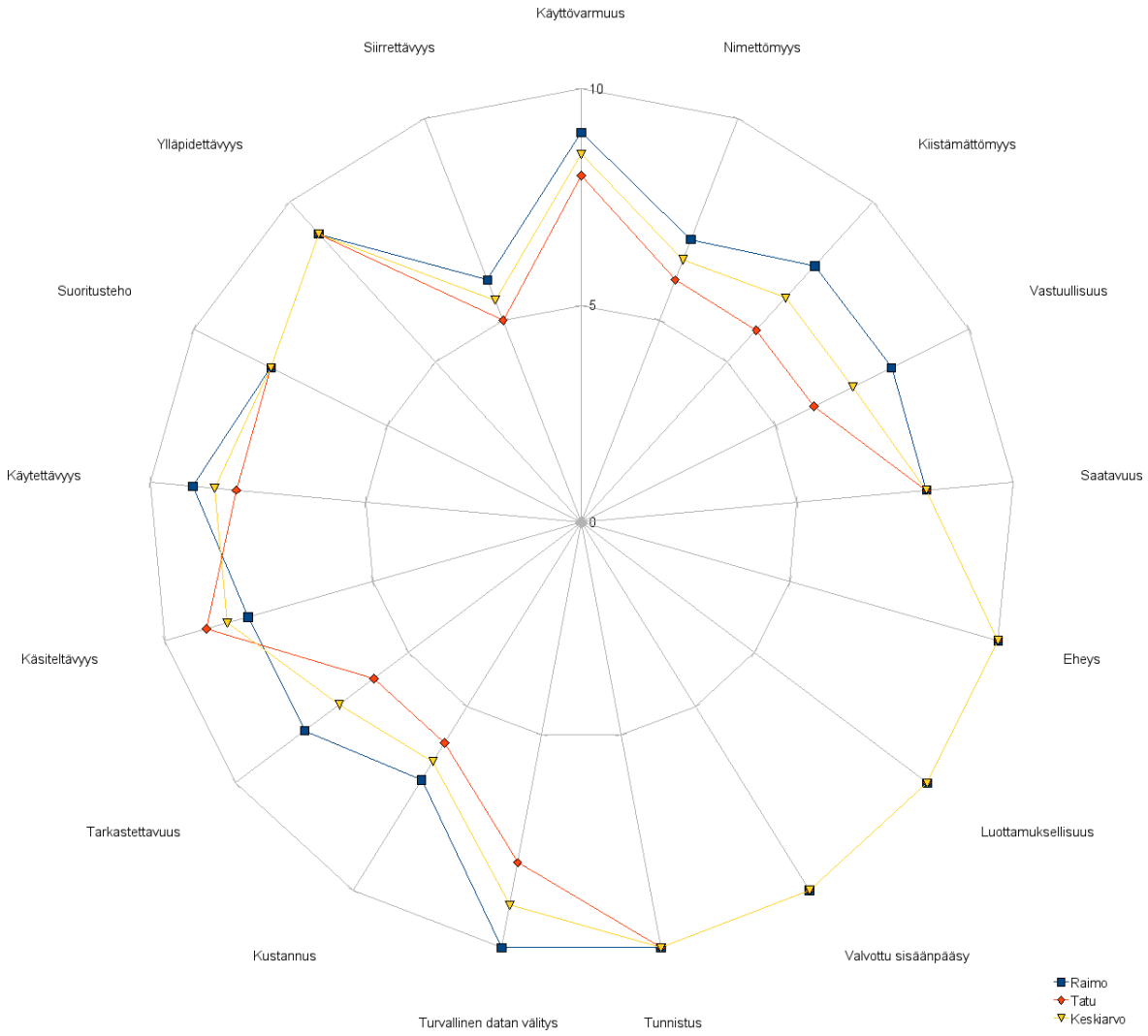
Tietoturvamallien soveltamiseksi ja valitsemiseksi ei ole olemassa yleisluontoista säännöstöä. Tässä tutkielmassa hyödynnetään taulukossa 4.1 esitettyjä malleihin liitettyjä tavoitteita, suhteita ja piirteitä. Niiden avulla sopivia malleja voidaan etsiä ja vastaavasti poissulkea vähemmän sopivia malleja. Lisäksi mallien valinnan apuna käytetään väärinkäyttötapauksia, joista kerrotaan tarkemmin luvussa 5.5.

Koska liikuntakalenterisovellus on suurimmaksi osaksi vielä suunnitteluvaiheessa, voidaan siihen luontevasti hyödyntää tietoturvamalleja jokaisesta kehitysvaihe-

ryhmästä (kts. 4.4.1). Pääpaino asetetaan kuitenkin sovellusarkkitehtuuriryhmään kuuluville malleille, koska niiden avulla päästään käsiksi sovelluksen perustaan ja saavutetaan suurimmat hyödyt. Liikuntakalenteria tehdessä tietoturvamallien olemassaoloa ei ole otettu huomioon, joten voidaan hyödyntää myös sellaisia malleja, jotka liittyvät sovelluksessa jo oleviin tietoturvaa parantaviin ominaisuuksiin, kuten pääsynvalvontaan. Tällä tavoin pystytään vertailemaan mallin esittämää ja nykyistä ratkaisua. Liikuntakalenterin käyttämään sessioon liittyvät mallit ja muut WWW-sovelluksiin sopivat mallit kannattaa tarkastella erityisen hyvin. Tutkimuksen kannalta mielekkäintä olisi tutkia muutamaa mahdollisimman erilaista mallia. Myös vaihtoehtoisten mallien tutkiminen olisi kiinnostavaa. On kuitenkin hyväksyttävä realiteetit ja ymmärrettävä ettei tutkimusta kaiken kattavassa mittakaavassa ole mahdollista toteuttaa. On parempi tehdä yksi asia hyvin kuin tehdä monta asiaa huonosti. Siksi onkin perusteltua rajata tutkimus koskemaan tiettyä liikuntakalenterin turvallisuusuhkaa ja eliminoida kyseinen uhka siihen parhaiten soveltuvilla malleilla.

5.4 Liikuntakalenteri suhteessa turvallisuuspiirteisiin

Tietoturvamalliin liitetyt piirteet antavat hyvän pohjan mallin valitsemisprosessiin. Liikuntakalenterin suhde eri piirteisiin arvioitiin suullisella kyselyllä esittämällä ensin turvapiirteen määritelmä ja sitten antamalla subjektiivinen arvosana kuinka tärkeänä kyseinen henkilö pitää piirrettä liikuntakalenterin kohdalla. Arvioitiin osallistui allekirjoittanut sekä Raimo Pitkänen. Arvioinnin tulokset Raimon, Tatun ja keskiarvon mukaan eriteltynä näkyvät kuvassa 5.4 olevassa graafissa. Graafista voidaan havaita, että erityisen tärkeinä pidetään eheyttä, luottamuksellisuutta, valvottua sisäänpääsyä ja tunnistusta (vastaukset sijoittuvat verkon ulkokehälle). Piirteistä vähiten tärkeimmiksi arvioitiin siirrettävyys ja kustannus (vastaukset sijoittavat lähimmäksi verkon keskipistettä). Eniten eroavaisuuksia arvioinnissa voidaan havaita kiistämättömyydellä, vastuullisuudella ja tarkastettavuudella. Vastauksista voidaan havaita, että Tatu on arvioinut piirteet usein vähemmän tärkeäksi kuin Raimo, mikä ainakin osittain johtuu siitä, että Raimo oli arvioinut eri piirteet toisistaan riippumattomasti, kun taas Tatu oli huomioinut eri piirteistä saatavat hyödyt ja niiden vaikutukset muihin piirteisiin.



Kuva 5.4: Turvallisuuspiirteiden tärkeys liikuntakalenterissa asteikolla 1-10

5.5 Väärinkäyttötapauksien soveltaminen liikuntakalenteriin

Tietoturvamallien valintaan vaikuttaa miltei yksinomaan se, millaisia turvallisuusuhkia järjestelmään kohdistuu. Väärinkäyttötapauksia käyttäen voidaan nähdä liikuntakalenteriin kohdistuvat turvallisuusuhkat selkeällä tavalla. Tässä tutkielmassa käytetään John McDermottin ja Chis Foxin [34] esittämää tapaa väärinkäyttötapauksien tuottamiseen. Jotta tutkimus saataisiin rajattua tiettyyn turvallisuusongelmaan, käydään tarkemmin läpi vain yksi väärinkäyttötapaus, jota tullaan jatkossa hyödyntämään tietoturvamallin valintaan.

Väärinkäyttötapausten teko aloitetaan määrittelemällä aktorit, jotka ovat ulkoisia tekijöitä, joista järjestelmään kohdistuu ainakin yksi uhka. Aktorien määrä on

pyritty rajaamaan tekemällä niistä mahdollisimman erillään olevia; esimerkiksi yhdistämällä samankaltaisia uhkia aiheuttavat aktorit yhdeksi yleisluontoisemmaksi aktoriksi. Taulukossa 5.1 esitellään aktorit ja liitetään kullekin resurssit, taidot ja tavoitteet.

Aktoreita tutkimalla voidaan voidaan heti havaita, että sovelluksen tämänhetkisessä vaiheessa merkityksellisin aktori on epäsuotuisa käyttäjä. Perusteluna tälle on se tosiseikka, ettei liikuntakalenteri ole vielä ollut toiminnassa ja siten saavuttanut suurta suosiota, jolloin se olisi houkutteleva kohde mainetta hakevalle kräkkerille. Liikuntakalenterilla ei myöskään tällä hetkellä ole kaupallista kilpailijaa, mikä on vaatimuksena sille, että kilpalijan palkkasoturista muodostuisi uhka. Epäsuotuisia käyttäjiä sovelluksessa voi olla käytännössä heti, kun sovellus otetaan käyttöön.

Aktorien esiintymistodennäköisyys muuttuu sovelluksen suosion kasvaessa. Ensiksi järjestys on todennäköisimmistä alkaen epäsuotuisat käyttäjät, mainetta hakeva kräkkeri ja kilpailijan palkkasoturi. Sovelluksen suosion kasvaessa mainetta hakeva kräkkerin todennäköisyys kasvaa. Kun sovellus on saavuttanut huippusuosion, on todennäköisintä että kilpailijat markkinaraon haistaessaan alkavat kohdistaa siihen uhkia. Tällöin kilpailijan palkkasoturi on todennäköisin aktori ja siten alkuperäinen esiintymistodennäköisyys on kääntynyt päälaelleen.

Taulukko 5.1: Liikuntakalenterin turvallisuutta uhkaavat aktorit

Aktori	Kuvaus	Resurssit	Taidot	Tavoitteet
Kilpailijan palkkasoturi	Kilpailevaa liikuntakalenteria kehittävän yrityksen palkkaama turvallisuustekniikan erityisosaaja.	Palkkasoturi saa yritykseltä käyttöön rahaa ja tietoa niin paljon kuin tarvitsee. Heillä on käytössään liikuntakalenterin dokumentointi kokonaisuudessaan sekä varhaisen vaiheen lähdekoodi. Palkkasoturi voi toimia yksin tai johtamassaan ryhmässä. Yritys on antanut heille kaksi kuukautta aikaa toteuttaa asettamansa tavoitteet.	Palkkasoturilla on kattava osaaminen kaikilta turvallisuustekniikan osa-alueilta ja erityisesti tietotekniikan alalta. Hän voi myös vapaasti saada käyttöönsä minkä tahansa alan asiantuntemuksen lisää avustajia palkkaamalla.	Ensimmäinen tavoite on saada liikuntakalenterin kehittäjät ja käyttäjät vakuuttumaan siitä, että liikuntakalenteri ei toimi ja sen käyttäminen on negatiivinen kokemus. Tähän pyritään tekemällä häirintää mahdollisimman huomaamattomasti ja jälkiä jättämättä. Toinen tavoite on varastaa liikuntakalenterin lähdekoodit sekä muu siihen käytetty tietotaito. Toiminnalla pyritään poistamaan liikuntakalenteri markkinoilta ja saamaan oma sovellus monopoliasemaan.

Jatkuu seuraavalla sivulla...

Taulukko 5.1 – Jatkuu

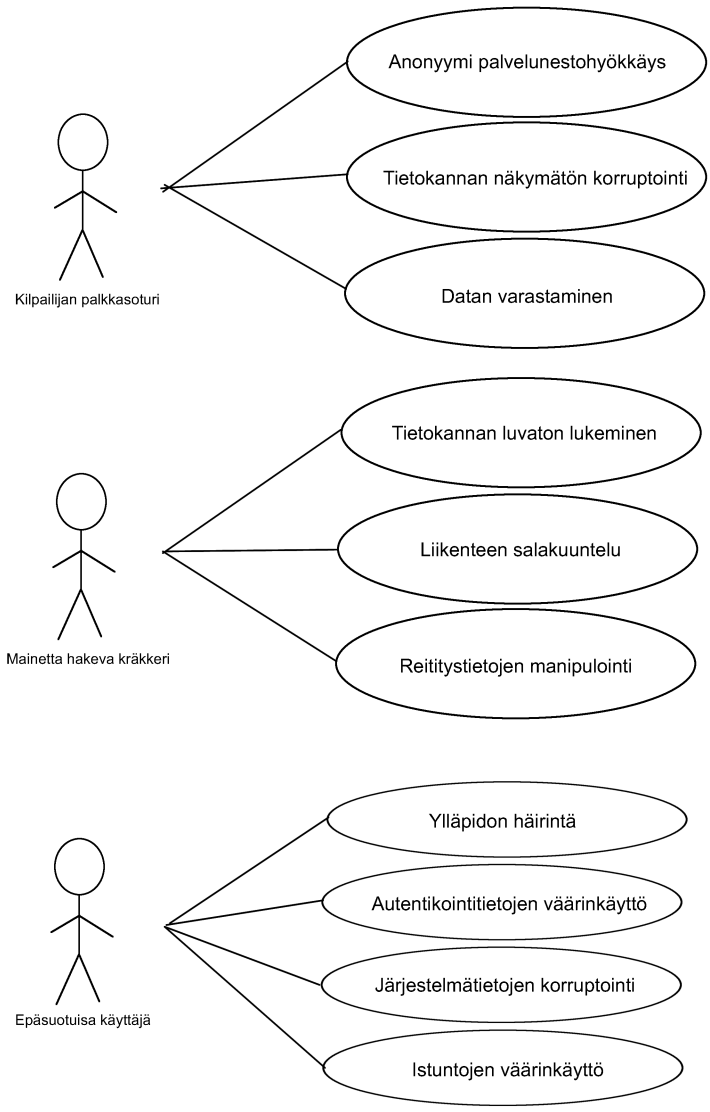
Aktori	Kuvaus	Resurssit	Taidot	Tavoitteet
Mainetta hakeva kräkkeri	Nuori tietotekniikkaa harrastava kräkkeri, joka haluaa päteä.	Kräkkeri voi toimia yksin tai kräkkeriryhmässä. Kräkkerit toimivat Internet-yhdeyden yli omalta kotikoneeltaan eivätkä he ole varanneet rahaa toimintaansa. Kräkkereillä on aikaa käytettävissään vuosia, mutta toiminta kohdistuu vapaa-ajalle ja on harrastepohjaista.	Kräkkerien tietotaito ei ole ammattimainen, mutta on kuitenkin hyvä. Kräkkerit matkivat olemassa olevia tunnettuja murtautumistekniikoita ja käyttävät apunaan olemassa olevia sovelluksia.	Kräkkerien tavoitteena on kartuttaa mainetta, havainnollistaa liikuntakalenterin huono tietoturva esimerkiksi levittämällä järjestelmän käyttäjätiedot vertaisverkkoon tai korvaamalla liikuntakalenterin sivut omalla mainossivulla. Rahallista hyötyä pyritään saamaan, jos se vain on mahdollista.

Jatkuu seuraavalla sivulla...

Taulukko 5.1 – Jatkuu

Aktori	Kuvaus	Resurssit	Taidot	Tavoitteet
Epäsuotuisa käyttäjä	Järjestelmään kirjautunut normaali käyttäjä tai ryhmän ylläpitäjä, joka tahattomasti tai tahallaan sekoittaa järjestelmää.	Käyttäjillä on pääsy Internettiin jollain tavalla. Järjestelmän käyttämiseen varataan keskimäärin puoli tuntia viikossa.	Epäsuotuisat käyttäjät voivat sekoittaa järjestelmää ainoastaan liikuntakalenterin käyttöliittymää hyödyntäen.	Epäsuotuisat käyttäjät joko huomaamattaan tai ilkeyttään sekoittavat järjestelmän eheyttä ja luottamuksellisuutta. He voivat esimerkiksi lisätä järjestelmään vääriä tapahtumia, ryhmiä tai käyttäjiä, ilmottautua tapahtumiin, joihin eivät todellisuudessa mene sekä ylläpitäjän muodossa poistaa tai väärinnimetä ryhmiä ja niihin ilmoittautuneita henkilöitä. He voivat jättää selaimen ja session auki liikuntakalenteriin esimerkiksi kirjastossa ja poistua tietokoneen äärestä. Lisäksi he voivat hukata tai kertoa ulkopuolisille käyttäjätunnuksen ja salasanansa järjestelmään.

Kun aktorit on selvitetty, tunnistetaan niiden väärinkäyttötapaukset. Diagrammissa 5.5 on kuhunkin aktoriin liitetty sille tunnistetut väärinkäyttötapaukset.



Kuva 5.5: Liikuntakalenterin väärinkäyttötapausdiagrammi

Aktorien väärinkäyttötapauksia tutkittaessa nähdään, että kilpailijan palkkasoturi on aktoreista selkeästi vaikein torjuttava. Seuraavaksi vaikeinpana voidaan pitää mainetta hakevaa kräkkeriä. Epäsuotuisan käyttäjän kohdistamia uhkia ei käytännössä pystytä täysin torjumaan, mutta niihin voidaan helpoimmalla tavalla puuttua esimerkiksi ohjeistusta parantamalla tai tehostamalla lokikäsittelyä.

Väärinkäyttötapausten tunnistamisen jälkeen ne määritellään. Kuten kappaleen alussa mainittiin, tässä tutkielmassa käsitellään tarkemmin vain yksi väärinkäyttötapaus ja siihen liittyvät turvallisuusongelmat. Aktoreja tutkiettaessa havaittiin epä-

suotuisan käyttäjän olevan tämänhetkisistä aktoreista todennäköisin. On siis mielekästä valita sille kuuluva väärinkäyttötapaus. Ohjeistusta parantamalla voidaan vähentää epäsuotuisten käyttäjien tahattomia väärinkäyttöjä, mutta tahallisia väärinkäyttöjä on useimmissa tapauksissa mahdotonta estää. Tällaiset väärinkäyttäjät tulisi kuitenkin saada vastuuseen teoistaan. Loki-käsittelyä parantamalla väärinkäytöt voidaan jälkikäteen havaita ja ryhtyä toimenpiteisiin. Liikuntakalenterissa ei ole hyödynnetty lokiominaisuuksia, mutta siinä on tallennettu tietokantaan tieto siitä, mitä tietoa kukin käyttäjä on kunkin session aikana käsitellyt [28].

Vastuullisuuden korostaminen vaikuttaisi hyvältä keinolta puuttua järjestelmän tietojen korruptointiin ja sitä kautta myös järjestelmän eheyteen, joka todettiin tärkeäksi kappaleessa 5.4. Toinen vaihtoehto olisi keskittyä pääsynvalvonnan tehostamiseen, sillä vaikka liikuntakalenteri ei pidä sisällään arkaluonteista tietoa, on tärkeää estää väärällä identiteetillä tehty toiminta — varsinkin ryhmien ylläpitäjien osalta. Pääsynvalvonta on kuitenkin jo tehty liikuntakalenteriin, joten se ei ole niin kiinnostava asia tutkittavaksi. Näiden seikkojen perusteella valitaan väärinkäyttötapaukseksi epäsuotuisan käyttäjän tapaus ”Järjestelmätietojen korruptointi”.

5.5.1 Järjestelmätietojen korruptointi - väärinkäyttötapaus

Kappaleessa kuvataan epäsuotuisan käyttäjän väärinkäyttötapaus järjestelmätietojen korruptointi. Väärinkäyttötapaukselle kerrotaan siihen liittyvä harmi, etuoikeusrajat ja loukkaava vuorovaikutus. Koska väärinkäyttötapaus rajoittuu käyttöliittymään ja koskee järjestelmää kokonaisvaltaisesti, voidaan väärinkäytössä käyttää hyödyksi käytännössä kaikkia sovelluksen käyttöliittymään liittyviä ominaisuuksia ja komponentteja.

Harmi: Käyttäjät tahallaan tai tahattomasti käyttävät järjestelmää normaalien tapojen vastaisesti.

Etuoikeusrajat: Käyttäjän toiminta rajoittuu sen mukaan onko hän normaalikäyttäjä (jokun ryhmän jäsen) ilman ryhmien ylläpitoon omaavia oikeuksia, vai omaako hän niiden lisäksi myös ryhmän ylläpitäjän oikeuksia. Ryhmien ylläpitäjiä koskevat kaikki normaalikäyttäjien toiminta ja niiden lisäksi heidän hallinnassaan olevien ryhmien ylläpitoon liittyvä toiminta.

Loukkaava vuorovaikutus: Epäsuotuisa käyttäjä käyttää apunaan sovelluksen tarjoamaa WWW-pohjaista käyttöliittymää ja joko tahallaan tai tahattomasti korruptoi järjestelmätietoja heikentäen järjestelmän eheyttä. Järjestelmätietojen korruptoinnissa on kyse käyttäjä-, tapahtuma- tai ryhmätietojen muuttamisesta,

poistamisesta tai lisäämisestä vahingollisella tavalla. Vahingollisina tai arveluttavina tapoina voidaan pitää esimerkiksi:

- Tekaistujen käyttäjien, tapahtumien tai ryhmien lisäämistä.
- Toistuvaa ilmoittautumista samanaikaisiin tapahtumiin.
- Jatkuvia poissaoloja tapahtumista, joihin on ilmoittautunut.
- Toistuvia kilpaileviin ryhmiin ilmoittautumisia.
- Liian pitkiä poissaoloja ryhmistä, joihin on ilmoittautunut.
- Jatkuvaa omien tietojen, tapahtumien tai ryhmän perustietojen muuttelua.
- Ryhmän jäsenten epäoikeutettua erottamista.
- Toistuvia ryhmiin tai tapahtumiin ilmoittautumisia ja perumisia.

Monet väärinkäyttötapaukseen liittyvät asiat voidaan estää suunnittelemalla ja toteuttamalla järjestelmään ominaisuudet, joilla poissuljetaan väärinkäyttömahdollisuus. Esimerkiksi kilpailevat ryhmät voitaisiin linkittää siten, että tarkastusrutiineilla estettäisiin liittyminen molempiin ryhmiin yhtäaikaa. Joitakin asioita on kuitenkin mahdotonta estää ohjelmallisesti. Hyvänä esimerkkinä tästä on ryhmän jäsenen epäoikeutettu erottaminen. Tällaiset asiat tulisi voida havaita helposti järjestelmän ylläpitäjien toimesta ja saada väärintekijä vastuuseen teostaan. Tämä liittyy suoraan vastuullisuuden turvallisuustavoitteeseen ja sen johdosta lokitietojen tarkasteluun.

6 Turvattu Lokin Tuottaja -tietoturvamalli

Luvussa esitellään Turvattu Lokin Tuottaja -tietoturvamalli. Se, kuinka tähän malliin päädyttiin, on esitetty kappaleessa 7.1. Mallin tarkempi kuvaus perustuu lähteeseen [21] ja kuvaustapa kappaleessa 4.6 esitettyyn rakenteeseen. Ensiksi esitellään omassa kappaleessaan yleiskuva mallista, jonka jälkeen seuraa mallin yksityiskohtainen kuvaus siten, että jokainen mallipohjan kohta on omana kappalenaan.

6.1 Yleiskuva mallista

Turvattu Lokin Tuottaja -mallin yleiskuva perustuu lähteeseen [23] ja samat tiedot voidaan nähdä tiivistetyssä muodossa ydinmallitaulukon 4.1 mallissa 11.

- Mallin nimi: Turvattu Lokin Tuottaja
- Tarkoitus: Sovelluksen tapahtumat tulee merkitä lokiin keskitetysti, eikä lokitietoja pitäisi pystyä muuttamaan.
- Tunnetaan myös: -
- Soveltuvuus: Sovelluksen arkkitehtuuritaso
- Turvallisuustavoitteet: Vastuullisuus
- Nimiöt: +Luottamuksellisuus +Eheys +Ylläpidettävyys +Käsiteltävyys -Suoritusteho
- Suhteet: Hyötyy mallista Turvattu Putki

6.2 Ongelma

Kaikki sovelluksen tapahtumat ja niihin liittyvät tiedot tulee viedä lokiin turvatusti erityisesti virheiden korjausta varten. Tämä voi johtaa ylimääräiseen koodiin ja monimutkaiseen logiikkaan.

Kaikki luotettavat sovellukset edellyttävät turvallisen ja käyttövarman lokitoiminnallisuuden. Lokia voidaan tarvita rikosteknillisiin tutkimuksiin, ja se täytyy

suojata varkauksia ja peukalointia vastaan. Lokin käsittelyn tulee olla keskitetty, jotta ylimääräiseltä koodilta vältytään. Kaikki tapahtumat tulee merkitä lokiin asianmukaisesti useassa vaiheessa sovelluksen toimintaelinkaaren aikana. Joissakin tapauksissa lokiin tuleva tieto voi olla arkaluontoista ja sen näkeminen tulisi estää luvattomilta käyttäjiltä. Lokitiedon käyttö ja muokkaus tulisi estää pahansuovalta käyttäjältä, joka pyrkii tunnistamaan informaatioreitin. Ilman keskitettyä hallintaa voi koodi joskus monistua, ja tällöin muutosten ylläpito ja toimivuuden tarkkailu voi muuttua vaikeaksi.

Yksi onnistuneen tunkeutumisen yleinen perustekijä on hyökkääjän kyky peittää jälkensä. Yleensä tämä tarkoittaa paljastavien tapahtumien poistamista useista lokitiedostoista. Ilman lokijälkeä ylläpitäjällä ei ole todistusaineistoa tunkeutujan toimista, eikä siksi mahdollisuutta jäljittää tunkeutujaa. Estääkseen hyökkääjää murtautumasta aina uudelleen ja uudelleen on ylläpitäjän varotoimenpitein varmistettava, että lokitiedostoja ei voi muuttaa. Lokitietojen luottamuksellisuus ja eheys voidaan turvata salausalgoritmeja käyttäen. Salausalgoritmien hyödyntäminen tähän tarkoitukseen voi olla monimutkaista ja vaivalloista, mikä on omiaan lisäämään tarvetta keskitetylle lokitoiminnallisuudelle.

6.3 Esimerkki

Perustuen lähteeseen [21], kappaleessa kerrotaan esimerkkitilanteita, joissa mallista on apua.

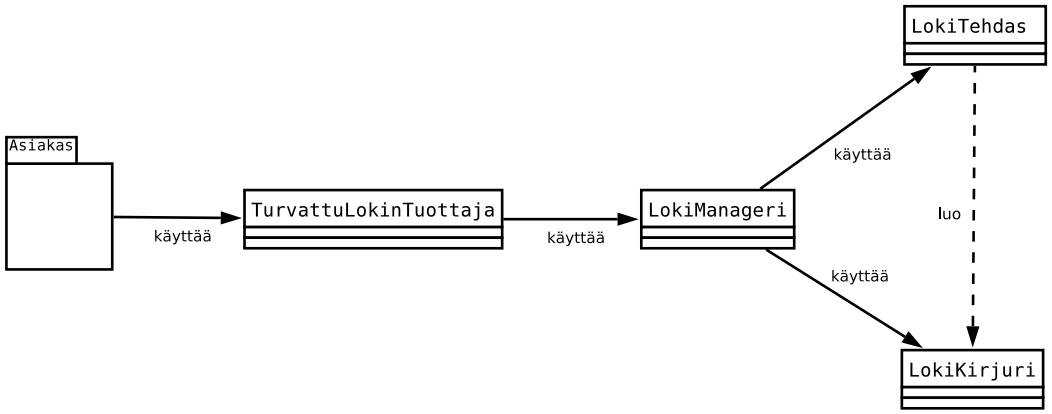
- Tarvitaan kirjata ylös arkaluonteista tietoa, joka tulisi olla suojassa luvattomilta käyttäjiltä.
- Halutaan varmistaa lokitiedon eheys, jotta voidaan ottaa selville onko tunkeutuja peukaloinut sitä.
- Virheen tai hyökkäyksen sattuessa halutaan tallentaa tulostus normaalilta toimintatasolta ja muilta tätä suuremmilta testaustasoilta.
- Halutaan keskitetty lokinvalvonta hallintatarkoituksiin.
- Halutaan soveltaa salausmenetelmiä lokitiedon luottamuksellisuuden ja eheyden varmistamiseen.

6.4 Ratkaisu

Käytetään Turvattu Lokin Tuottaja -mallia kirjaamaan ylös viestit turvallisella tavalla, jolloin niitä ei voi helposti muuttaa tai poistaa, ja tapahtumat eivät voi hävitä.

Turvattu Lokin Tuottaja -malli tarjoaa keskitetyn hallinnan lokitoiminnallisuudelle, jota voidaan käyttää useissa kohdissa kaikkialla sovelluksen pyynnöissä ja vasteissa. Keskittämällä hallinta annetaan keinot lokikäsittelijän toteutusyksityiskohtien erottamiseen sitä käyttävien kehittäjien koodista. Tapahtumien käsittelyä voidaan muuttaa ilman vaikutusta olemassa olevaan koodiin. Kehittäjät voivat esimerkiksi tehdä yksittäisen metodikutsun Java- tai JSP-koodissaan. Turvattu Lokin Tuottaja -malli pitää huolen siitä, kuinka tapahtumat kirjataan lokiin luotettavasti ja turvallisesti.

Kuvassa 6.1 esitellään Turvattu Lokin Tuottaja -mallin rakennetta luokkakaavion avulla.



Kuva 6.1: Tietoturvamallin Turvattu Lokin Tuottaja luokkakaavio [21]

Kuvassa 6.2 esitellään Turvattu Lokin Tuottaja -mallin osanottajat ja vastuut. Kaaviossa on seuraavat osanottajat:

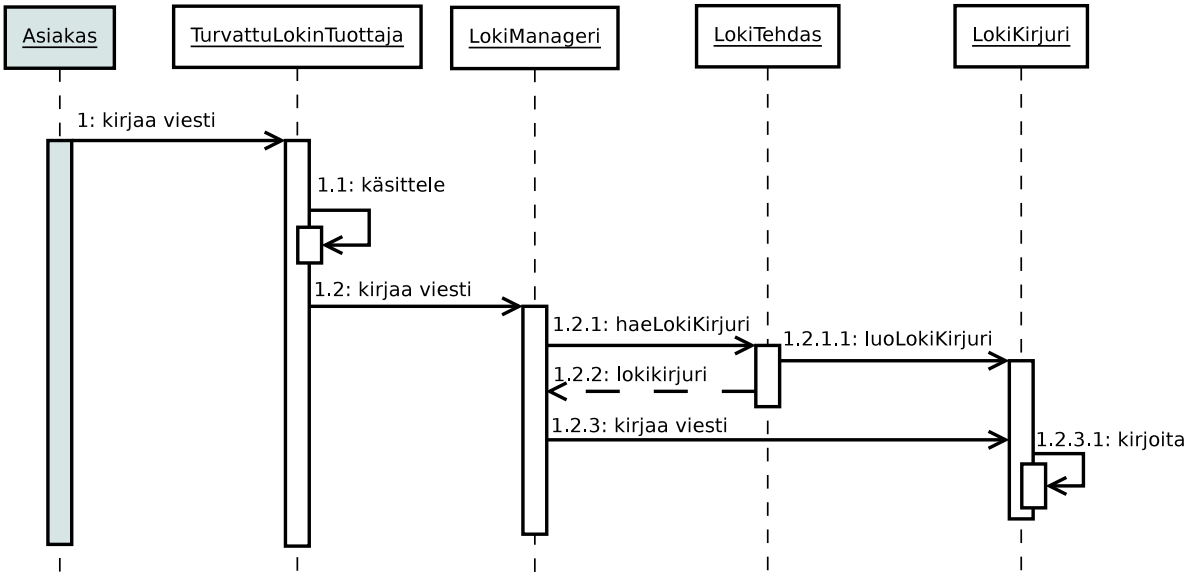
Asiakas lähettää pyynnön tiettyyn kohderesurssiin.

TurvattuLokinTuottaja on luokka, jota käytetään lokitiedon käsittelyyn turvallisella ja keskitetyllä tavalla.

LokiManageri hankkii LokiKirjurin ilmentymän LokiTehtaalta ja käyttää sitä viestien kirjaamiseen.

LokiTehdas vastaa LokiKirjurien luomisesta ja palauttaa LokiKirjuri-ilmentymät.

LokiKirjuri kirjoittaa lokiviestit kohteeseensa.



Kuva 6.2: Tietoturvamallin Turvattu Lokin Tuottaja sekvenssikaavio [21]

Asiakas käyttää TurvattuLokinTuottaja-luokkaa tapahtumien kirjaamiseen. TurvattuLokinTuottaja keskittää lokikirjauksen hallinnan ja eristää turvallisuusmekanismit, joita tarvitaan luvattomien lokimuutosten estämiseen.

1. Asiakas haluaa kirjata ylös tapahtuman käyttäen TurvattuaLokinTuottajaa.
2. TurvattuLokinTuottaja generoi järjestysnumeron ja liittää sen viestin eteen.
3. TurvattuLokinTuottaja antaa LokiMangerille järjestysnumeroidun tapahtumamerkkijonon kirjattavaksi.
4. LokiManageri hankkii kahvan LokiKirjurin ilmentymään LokiTehtaalta.
5. Lokitehdas luo LokiKirjuri ilmentymän.
6. LokiManageri delegoi varsinaisen lokikirjauksen LokiKirjurille.

Lokirjausprosessissa on kaksi osaa. Ensimmäinen osa sisältää kirjattavan tiedon turvaamisen ja toinen osa turvatun tiedon lokikirjauksen. TurvattuLokinTuottaja-luokka pitää huolta tiedon turvaamisesta ja LokiManageri-luokka hoitaa lokikirjauksen.

6.5 Toteutus

Turvattu Lokin Tuottaja -malli voidaan toteuttaa turvattu tiedonkeruu -strategialla (engl. *secure data logger strategy*) tai turvattu lokin varastointi -strategialla (engl. *secure log store strategy*), mitkä käsitellään seuraavaksi tarkemmin.

6.5.1 Turvattu tiedonkeruu -strategia

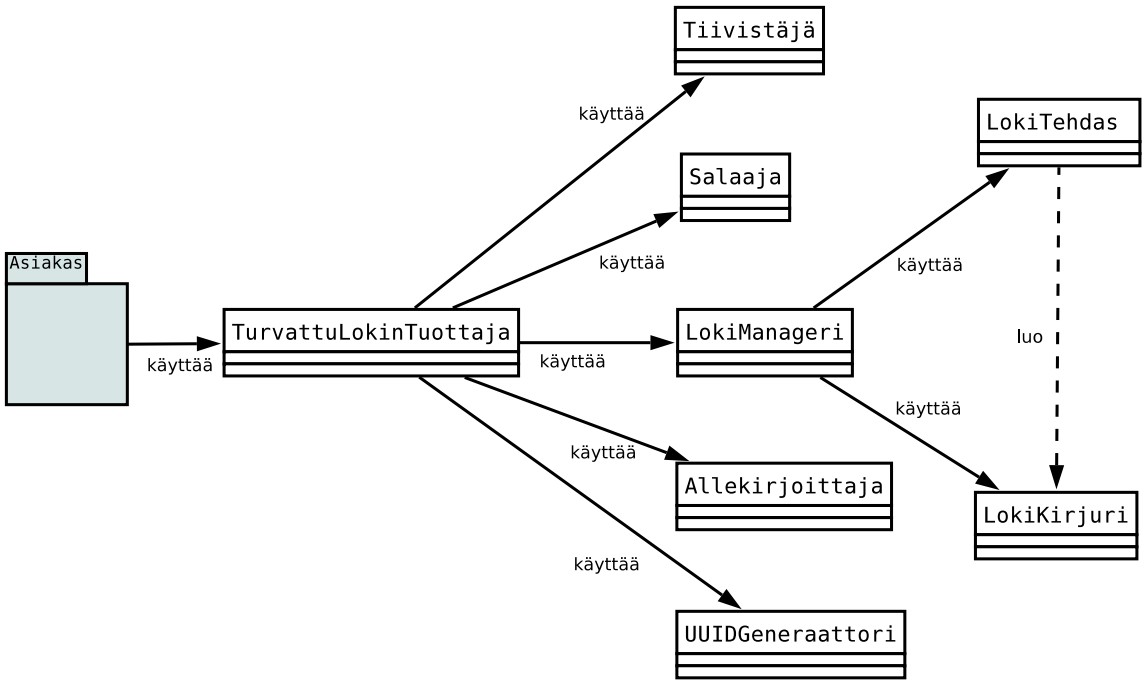
Turvattu tiedonkeruu -strategiassa turvataan itse tieto ennenkuin se välitetään tavanomaisella tavalla lokiin kirjattavaksi. Tällöin kaikki lokitietoon kohdistuvat muutokset ja poistot voidaan havaita ja pitää tieto turvassa lokikohteen (tiedoston, tietokannan tai viestijonon) vaarantuessa. Strategia on käyttökelpoinen, jos infrastruktuurin turvallisuus ei ole riittävä arkaluonteisten tietojen tai peukaloinnille alttiiden lokimerkintöjen suojaamiseen.

Tiedon turvaamisen avuksi Turvattu Lokin Tuottaja -malliin on esitelty neljä uutta luokkaa, Tiivistäjä, Salaaja, Allekirjoittaja ja UUIDGeneraattori, joissa käytetään salausmekanismeja ja suoritetaan useita toimintoja, mitkä varmistavat kirjatun tiedon luottamuksellisuuden ja peukaloinninkeston. Kuvassa 6.3 on esitetty Turvattu Lokin Tuottaja -mallin luokkakaavio höystettynä turvattu tiedonkeruu -strategialla ja kuvassa 6.4 sekvenssikaavio tapahtumaketjulle, jossa kuvataan tiedon turvaaminen ennen sen kirjaamista lokiin.

Tiedon turvaaminen koostuu kolmesta elementistä:

- Suojaa arkaluontoinen tieto (engl. *protect sensitive data*), missä varmistutaan siitä, että kaikki arkaluonteinen data tallentuu ja pysyy luottamuksellisena koko prosessin ajan. Esimerkiksi luvattomien henkilöiden ei tulisi avoimesti nähdä luottokortin numeroita. Itse tieto suojataan symmetrisen avaimen algoritmilla, ja symmetrinen avain julkisen avaimen algoritmeilla. Julkisen avaimen algoritmit ovat hyvin raskaita, eikä niitä kannata käyttää itse tiedon salaamiseen. Symmetrisen avaimen kunnollinen suojaaminen varmistaa sen, että arkaluontoinen tieto on suojassa lokeihin käsiksi päässeiltä hyökkääjiltä. TurvattuLokinTuottaja-luokan käyttämä SalausAvustaja-luokka vastaa annatun merkkijonon salaamisesta. Salauksen purku sen sijaan tulisi tehdä sovelluksen ulkopuolella ulkoisella apuohjelmalla, johon ei ole pääsyä sovelluksesta eikä sen palvelimelta. Tällä tavoin hyökkääjien pääsyä arkaluontoiseen dataan vaikeutetaan entisestään.

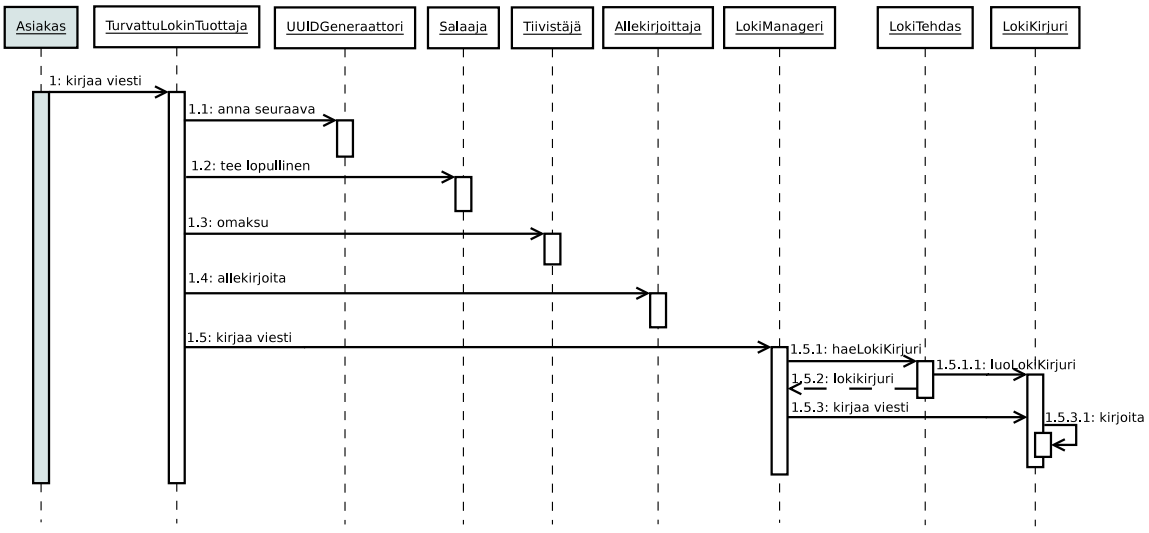
- Estä tiedon muuttaminen (engl. *prevent data alteration*), missä varmistutaan siitä, että tieto on peukaloinninkestävää. Esimerkiksi käyttäjätunnisteita tai ti-



Kuva 6.3: Tietoturvamallin Turvattu Lokin Tuottaja luokkakaavio turvattu tiedonkeruu -strategialla [21]

litapahtumien määriä ei pitäisi pystyä muuttamaan. Tietomuutokset voidaan välttää käyttämällä sähköposteistakin tuttuja sähköisesti allekirjoitettuja tiivistelmiä (engl. *digitally signed message digests*). Tiivistelmä luodaan ja allekirjoitetaan jokaiseen lokitiedoston viestiin. Allekirjoitus estää hyökkääjää muuttamasta viestiä ja muodostamasta seuraavaa tiivistelmää muuttuneesta tiedosta. Tähän tarkoitukseen TurvattuLokinTuottaja käyttää TiivistelmaAvustaja- ja SahkoinenAllekirjoitusAvustaja-luokkaa.

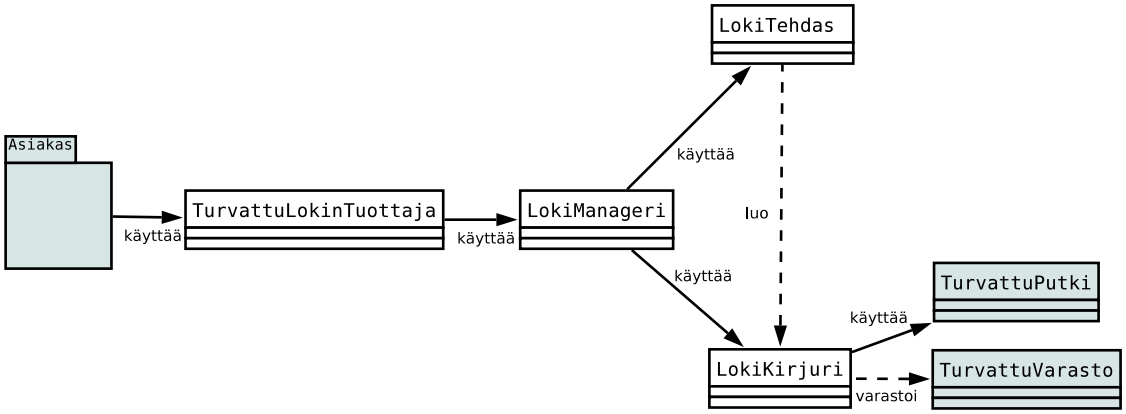
- Havaitse tiedon poisto (engl. *detect deletion of data*). Havaitaan lokista poistetut tapahtumat; selvät merkit siitä, että hyökkääjä on vaarantanut järjestelmän. Tiivistelmien tai sähköisten allekirjoitusten käyttö ovat hyödyttömiä, jos koko lokikirjaus ja sen mukana allekirjoitetut viestit poistetaan. Tiedon poistojen havaitsemiseen käytetään järjestysnumeroa, joka sisällytetään kaikkiin kirjauksiin ja joka on mukana allekirjoitetussa tiedossa. Jokainen aukko järjestysnumeroissa ilmaisee puuttuvaa kirjausta. Allekirjoitukset estävät hyökkääjää muuttamasta järjestyksessä seuraavia numeroita, jolloin ylläpitäjän on helppo havaita poistot lokitiedoista. TurvattuLokinTuottaja käyttää järjestysnumeroihin universaalisti uniikin tunnisteen väliohjelmistomallia (engl. *Universally Unique Identifier (UUID) Middleware pattern*).



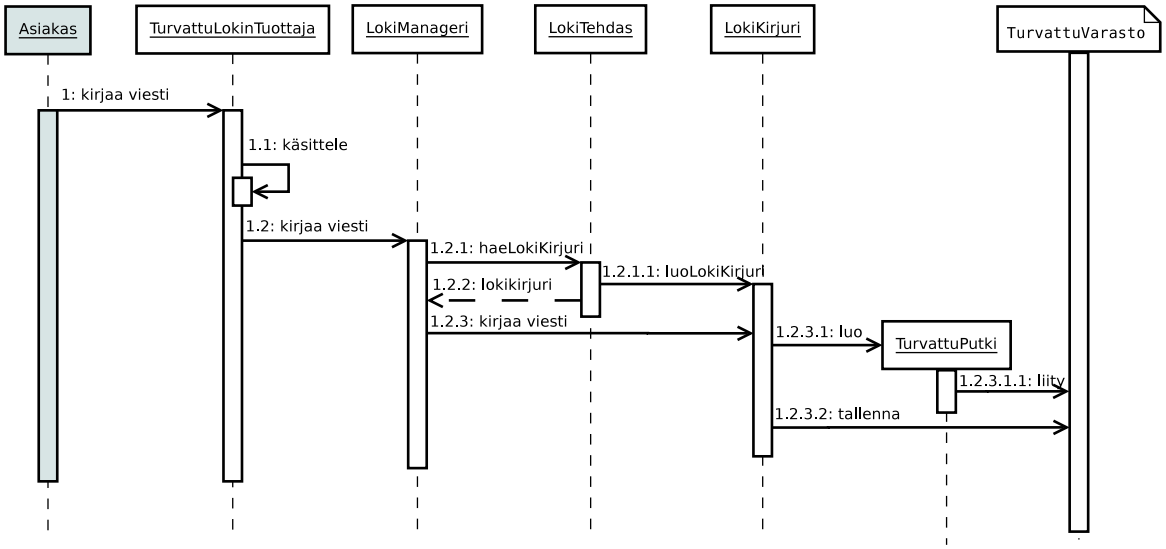
Kuva 6.4: Tietoturvamallin Turvattu Lokin Tuottaja sekvenssikaavio turvattu tiedonkeruu -strategialla [21]

6.5.2 Turvattu lokin varastointi -strategia

Turvattu lokin varastointi -strategia turvaa itse lokin peukaloinnilta, jolloin kaikki kirjattu tieto on varmuudella virheetöntä ja täydellistä. Lokitieto sijoitetaan turvattuun tietovarastoon, joka voidaan toteuttaa monenlaisilla valmistuotteilla tai tekniikoilla. Turvattu Putki -malli (engl. *Secure Pipe Pattern*, kts. ydinmallitaulukon 4.1 malli 33) on esimerkki tällaisesta tekniikasta. Sitä käytetään turvaamaan tieto peukaloinnilta sen siirtyessä turvattuun varastoon. Kuvassa 6.5 on esitetty luokkakaavio, jossa Turvattu Lokin Tuottaja -malli on toteutettu turvattu lokin varastointi -strategialla.



Kuva 6.5: Tietoturvamallin Turvattu Lokin Tuottaja luokkakaavio turvattu lokin varastointi -strategialla [21]



Kuva 6.6: Tietoturvamallin Turvattu Lokin Tuottaja sekvenssikaavio turvattu lokin varastointi -strategialla [21]

Turvattu lokin varastointi -strategia ei vaadi Turvattu tiedonkeruu -strategiassa esitettyä tiedon käsittelyä, vaan se käyttää Turvattu Putki -mallia ja turvattua tietovarastoa, esimerkiksi tietokantaa, jota TurvattuVarasto-objekti havainnollistaa. Kuvassa 6.6 on esitetty Turvattu lokin varastointi -strategian sekvenssikaavio. Ainoat muutokset Turvattu Lokin Tuottaja -mallin alkuperäiseen sekvenssikaavioon (kts. kuva 6.2) on Turvattu Putki -mallin ja TurvattuVarasto-kohteen käyttöönotto. LokiKirjuri luo turvallisen yhdeyden TurvattuVarasto -kohteeseen käyttäen TurvattuPutki-luokkaa, jonka jälkeen LokiKirjuri kirjaa viestit normaalisti. TurvattuVarasto vastaa siitä, että lokitiedostoja ei päästä peukaloimaan. Se voidaan toteuttaa:

- Tietokannalla, jossa on annettu vain lukuoikeudet lokin kirjauksen tekevälle käyttäjälle.
- Erilliseen turvattuun laatikkoon vain kirjoituskyvyn omaavalla kuuntelijalla.
- Millä tahansa muulla tavoin, mikä estää lokimerkintöjen poistot, muutokset tai luvattomat lisäykset.

6.6 Piilevät vaarat

Turvattu Lokin Tuottaja -malli edellyttää sovellukseen tulokohdan tietojen keruulle, mikä voi aiheuttaa piileviä vaaratilanteita ainakin avainten hallintaan ja luottamuksellisuuteen liittyen:

- Avainten hallinta. Turvattu Lokin Tuottaja -mallissa on salattava joko itse data tai luotava turvallinen kanava turvattuun lokivarastoon. Salauksessa käytettyjen salausavaimien salassapito on elinehto sille, että Turvattu Lokin Tuottaja -malli toimisi. Mikäli avain tai salasana avaimen hakuun (esimerkiksi avainvarastosta) joudutaan pitämään koodissa, tulisi koodin tulkitsemista vaikeuttaa.

- Luottamuksellisuus. Käytettäessä Turvattu lokin varastointi -strategiaa Turvattu Lokin Tuottaja -mallissa tulee varmistaa, että yhteys turvattuun varastoon on turvallinen. Riittämätön suojaus avaa hyökkääjälle mahdollisuuden muuttaa tietoa siirron aikana.

6.7 Seuraamukset

Turvattu Lokin Tuottaja -mallin käyttäminen parantaa lokitiedostojen luottamuksellisuutta ja eheyttä ja helpottaa tietojen keruuta kaikista tietoa sisältävistä sovelluksen tapahtumista, käyttäjän pyynnöistä ja vasteista.

Lisäksi sen vaikutuksia ovat:

- Tiedon keruun hallinnan keskittäminen. Turvattu Lokin Tuottaja edistää uudenlleenkäyttävyyttä ja ylläpidettävyyttä keskittämällä tiedon keruun hallinnan ja erottamalla toteutusyksityiskohdat sovellusliittymästä. Tämän ansiosta kehittäjien on mahdollista käyttää tiedonkeruuvälineitä kaikkialta sovellusliittymästä, riippumatta itse tiedon keruuseen rakennetuista turvallisuustoiminnosta. Sovelluksen turvallisuustaso paranee turvallisuusmenetelmien väärinkäyttämisen vaikeutuessa.

- Huomaamattomien lokimuutosten torjuminen. Turvattu Lokin Tuottaja -mallia hyödyntämällä voidaan estää lokimuutokset ja lokitiedostoja tutkimalla havaita tietomurrot. Ne ovat ensiaskeleet tunkeutujien löytämiseen ja tietomurtojen torjumiseen.

- Suoritustehon heikentyminen. Turvattu Lokin Tuottaja -malli vaikuttaa suoritustehoon paljon suoritusaikaa vaativien salausalgoritmien takia. Salausalgoritmeja tulisi käyttää mahdollisimman tehokkaasti ja välttää tarpeetonta tiedonsalausta, jotta järjestelmään saatavuus pysyisi hyvänä.

- Laajennettavuuden edistäminen. Turvallisuus on jatkuvasti kehittyvä prosessi. Jotta sekä nykyisiä että tulevia uhkia vastaan voitaisiin suojautua, koodin tulee olla mukautuvaa ja laajennettavaa. Turvattu Lokin Tuottaja -malli

mahdollistaa edelletyn laajennettavuuden piilottamalla toteutusyksityiskohdat yleiskäyttöisen rajapinnan taakse. Ohjelmakoodin eliniän kasvaessa myös sen käyttövarmuus paranee testauksen ja ohjelmavirhekorjausten johdosta.

- Käsiteltävyyden kohentuminen. Keskitettyä tiedon keruun hallintaa on helpompi käsitellä ja seurata. Turvattu Lokin Tuottaja -mallissa kaikki tarvittava turvallisuuskäsittely tehdään ennen varsinaista tiedon kirjaamista, mikä mahdollistaa kunkin toiminnon hallinnan itsenäisesti, vaarantamatta kokonaisturvallisuutta.

6.8 Sukulaismallit

Turvattu Lokin Tuottaja -mallin sukulaismalleja ovat [21]:

- Abstrakti Tehdas -suunnittelumalli (engl. *Abstract Factory Pattern*), missä annetaan rajapinta yhteenkuuluvien tai riippuvaisten olioperheiden muodostukseen erittelemättä niiden konkreettisia luokkia [17, s. 99].

- Turvattu Putki -tietoturvamalli (engl. *Secure Pipe Pattern*, kts. ydinmallitaulukon 4.1 malli 33), mikä takaa tiedon eheyden ja yksityisyyden säilymisen sen kulkiessa asiakkaan ja palvelimen tai palvelimien välillä.

6.9 Tunnetut käyttökohteet

Tietoa Turvattu Lokin Tuottaja -mallin käytöstä tunnetuissa käyttökohteissa ei luonnollisesti ole julkisesti saatavilla. Mallia kuitenkin käytetään: esimerkiksi Yolande et al. [41] soveltavat muiden mallien ohessa myös Turvattua Lokin Tuottaja -mallia terveydenhuoltoalan sovellukseensa.

7 Tietoturvamallin integrointi

Luvussa tutkitaan tietoturvamallin integroimisprosessia liikuntakalenterisovelluksen avulla.

7.1 Tietoturvamallin valitseminen liikuntakalenterisovellukseen

Tietoturvamallien valitseminen jo olemassa olevaan sovellukseen vaatii tarkkaa selvitystyötä. Aiemmin luvussa 5 on käsitelty liikuntakalenterista seuraavat asiat:

- Missä vaiheessa sovellus on, eli minkä kehitysvaiheen malleja voidaan käyttää järkevällä tavalla.
- Mitä tietoturvamalleissa kerrottuja asioita sovelluksessa jo on.
- Mitä turvallisuustavoitteita ja piireitä sovelluksessa halutaan korostaa.
- Mitä uhkia sovellukseen kohdistuu.

Väärinkäyttötapauksiin pohjautuneessa uhkakartoituksessa (kts. kappale 5.5) havaittiin, että vastuullisuuden lisäys on tällä hetkellä tehokas ja mielekäs tapa nostaa sovelluksen turvallisuustasoa. Koska vastuullisuus hyötyy suuresti hyvästä lokikäsittelystä, niin tarkastellaan ydinmalleja (kts. taulukko 4.1) näistä lähtökohdista.

Vastuullisuuteen voidaan vaikuttaa positiivisesti ydinmalleilla 1, 2, 14 ja 29, mutta se on jopa päätavoitteena malleilla 11 ja 26. Kun malleja 11 (Secure Logger) ja 26 (Audit Interceptor) tarkastellaan lähemmin, havaitaan mallin 26 riippuvan mallista 11. Mallia 26 ei siis voi käyttää ennenkuin malli 11 on toteutettu. Koska aiemmin päätettiin keskittyä vain yhteen malliin, on luonnollinen valinta käytettäväksi malliksi 11, eli Secure Logger. Mallin suomenkielinen nimi on Turvattu Lokin Tuottaja, ja se on esitelty yksityiskohtaisesti luvussa 6.

7.2 Turvattu Lokin Tuottaja -mallin käyttöönotto liikuntakalenterissa

Koska Turvattu Lokin Tuottaja -mallin käyttöönotto on laaja prosessi, se hajautetaan kappaleessa neljään eri osaan. Ensimmäisessä osassa selvitetään, mitä tietoja

liikuntakalenterista tulisi kirjata lokiin ja miksi. Kun lokiin kirjattava tieto on selvillä, päätetään toisessa osassa minne lokitiedot varastoidaan ja sen nojalla mallissa käytettävä toteutusstrategia. Kolmannessa osassa kerrotaan tarkemmin mallin integroimisesta liikuntakalenteriin ja neljännessä osassa havainnollistetaan esimerkkitapauksen avulla, miten lokitoiminnallisuus toteutetaan liikuntakalenteriin.

7.2.1 Lokiin kirjattavat tiedot

Järjestelmätietojen korruptointi -väärinkäyttötapauksen (kts. kappale 5.5.1) nojalla lokitietojen avulla tulisi helpolla tavalla pystyä selvittämään kaikki tapahtumat, joita tietty käyttäjä on tehnyt kunkin session aikana. Koska käyttäjätunniste ja siihen liittyvä salasana voivat joutua vääriin käsiin, tulisi käyttäjätunnus-sessio -parin yhteyteen liittää sitä tarkentavaa tietoa, mikä auttaisi paikallistamaan väärintekijän ja antaisi tietoa siitä onko jokin käyttäjätunniste vaarantunut. PHP-koodissa on mahdollista selvittää ennaltamäärättyjen muuttujien (engl. *predefined variables*) avulla mm. käyttäjän IP-osoite, selain ja käyttöjärjestelmä [42]. Näiden tietojen perusteella voidaan väärinkäyttäjä paikallistaa ja ohjelmallisesti tutkia todennäköisyyksiä sille, käyttääkö jotain käyttäjätunnusta useampi henkilö. Jos tiedot sessiosta toiseen jatkuvasti vaihtuvat, on todennäköisempää se, että käyttäjäkin vaihtuu. Myös sessioiden kestot alkamis- ja päättymisaikoineen tulisi kirjata lokiin. Ideaalitilanne olisi, jos lokitietojen pohjalta järjestelmän tila voitaisiin palauttaa siihen, mikä se oli ennen väärinkäyttöä. Näiden tietojen pohjalta lokissa tulisi siis olla:

- Käyttäjä- ja sessiotunniste
- IP-osoite-, selain- ja käyttöjärjestelmätieto
- Session alkamis- ja päättymisaika
- Sessiotapahtuman aika
- Sessiotapahtumaan liittyvät tiedot:
  - Edellisen sivun osoite
  - Nykysivun osoite
  - Käyttäjän tekemät valinnat ja kentiin kirjoittamat tiedot
  - Tapahtumaan liittyneet tietokantaa muuttaneet SQL-lauseet

7.2.2 Lokin varastointi

Liikuntakalenteriin ei ole varsinaisesti toteutettu lokiominaisuuksia. Sen tietokannassa on kuitenkin esimerkiksi sessiokäsittelyyn liittyviä tauluja, joita voidaan vähäisin muutoksin hyödyntää lokitietojen keräämisessä. Tuntuisikin mielekkäältä käyttää tätä kantaa lokivarastona, eikä tallentaa lokitietoja erilliseen varastoon. On kuitenkin huomattava, että tässä tapauksessa tietokannan merkitys korostuu, koska tietokannan pettäessä myös sovelluksen lokihistoria on mennyttä. Lokitietojen turvaamiseksi tietokanta on syytä varmuuskopioida säännöllisesti, jolloin pahimmassakin tapauksessa menetetään vain tiedot edellisestä varmuuskopiosta alkaen. Sovitaan, että tämä on hyväksyttävä riski, ja käytetään lokivarastona liikuntakalenterin olemassa olevaa tietokantaa. Tällöin Turvattu Lokin Tuottaja -malli tulee toteuttaa Turvattu tiedonkeruu -strategiaa käyttäen (kts. kappale 6.5.1).

Tietokantaa tarkemmin tarkasteltaessa voidaan nähdä, että käyttäjätunniste- ja sessiotiedot on tallennettu tauluihin user, session ja session_inputs (kts. liikuntakalenterin tietokantakuvaus 5.2, jossa kyseiset taulut on korostettu). User- ja sessiontaulu ovat vain eräänlaisia perustietovarastoja ja itse oleellinen tieto, eli käyttäjän kirjaamat syötteet, löytyvät taulusta session_inputs. Session_inputs-taulun tietojen perusteella on mahdollista yhdistää käyttäjä-, sessio- ja tapahtumatieto, sillä tauluun on tallennettu kaikki tietyn session kenttien tuoreimmat syötteet. Itse käyttäjä saadaan poimittua session-taulun avustuksella, sillä yksi sessiotunniste kuuluu aina tietylle käyttäjälle. Taulua käytetään sovelluksessa hakemaan tiedot syöttökenttiin, jotta session sisällä sivulta toiselle liikkuminen olisi mahdollisimman vaivatonta ja tietoja ei pääsisi hukkumaan. Sessioon liittyviä tietoja ei ole salattu, mutta ne on pakattu tiiviimpään muotoon käyttäen PHP:n serialize-toimintoa.

Session_inputs-taulua ei suoraan voida käyttää lokitauluna, sillä sessioon liittyvät tiedot tulisi pystyä nopeasti noutamaan taulua käyttäen ja niiden salaaminen lokitarpeita varten heikentäisi merkittävästi sovelluksen suorituskykyä. Kyseinen johtopäätös voidaan tehdä, koska session_inputs-taulun tietojen salaus vaatisi vastaavasti niiden purkamisen niitä noudettaessa. On myös huomattava, että session_inputs-taulussa on tallessa vain viimeisimmät sessioon liittyvät kirjaukset ja session päättyessä siihen liittyvät tiedot poistetaan. Tapahtumahistoriaa ei siis ole. Näiden seikkojen johdosta päädytään tilanteeseen, missä mitään tietokannan taulua ei voida käyttää suoraan lokivarastona. Kantaan on siis luotava uusi taulu lokia varten. Olkoon uuden taulun nimi session_log, johon lisätään tietoa aina, kun session_inputs-taulun tietoja päivitetään, ja tilanteissa, joissa käyttäjän toimet aiheuttavat tietokannan tietojen muutoksen, poiston tai lisäyksen. Määritellään

session_log-tauluun seuraavat kentät:

- logid. Lokin juokseva ja yksilöivä tunnistenumero.
- userid (käyttäjätunniste). Käyttäjätunnisteen ”kopioimisella” lokitauluun saadaan merkittävä nopeusetu käyttäjäkohtaisiin hakuihin.
- sessionid (sessiotunniste).
- ipaddress (käyttäjän IP-osoite).
- clientinfo. Tietopaketti, jossa on mukana mm. tiedot käyttäjän käyttöjärjestelmästä ja selaimesta. Tieto saadaan PHP:n ennaltamäärätystä muuttujasta HTTP_USER_AGENT, ja se voi olla esimerkiksi muotoa Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586) [42].
- logdatetime (lokin kirjoitushetki). Indeksoimalla userid, sessionid ja logdatetime (tai logid) voidaan nopeasti hakea session alkamis- ja päättymisajat (olettaen ettei samaa sessiotunnistetta voi tulla useaan kertaan yhdelle käyttäjälle).
- logdata. Sisältää sessiotapahtumaan liittyvän datan salattuna. Data voi olla joko SQL-komento tai syöttökenttätunniste–arvo-pari. Mukana on lisäksi tiedon validoinnin mahdollistamiseksi kaikki tietueen muihin kenttiin tallennettu tieto sekä tiedot nykyisestä ja edellisestä käyttäjän sivusta.

Tietokantahakujen mahdollistamiseksi vain logdata-kenttään kirjattava tieto salataan. On myös syytä otaksua, että muiden kenttien tieto ei ole niin arkaluontoista, että sitä tarvitsisi salata.

7.2.3 Lokin integrointi

Turvattu Lokin Tuottaja -mallin lisääminen liikuntakalenterisovellukseen Turvattu tiedonkeruu -strategialla aloitetaan toteuttamalla luokat TurvattuLokinTuottaja, UUIDGeneraattori, Salaaja, Tiivistäjä, Allekirjoittaja, LokiManageri, LokiTehdas ja LokiKirjuri. Luokista Allekirjoittaja, Salaaja ja Tiivistäjä esitetään PHP:lla toteutetut esimerkkitoteutukset, mutta muiden luokkien tarkempi toteutus sivuutetaan ja ja niistä kerrotaan vain liikuntakalenterin kannalta oleellisimmat asiat. Kaikista luokista tulisi tehdä yleiskäyttöiset ja perintää hyödyntäen tehdä niiden aliluokkiin liikuntakalenterin tarvitsemat erityispiirteet. Erityisen paljon liikuntakalenterikohtaista toiminnallisuutta tulee TurvattuLokinTuottajan ja LokiKirjurin aliluokkiin. Tutkielmassa viitataan jatkossa luokkiin niiden varsinaisilla nimillä, jotta aliluokkien nimet eivät monimutkaistaisi kerrontaa.

TurvattuLokinTuottaja-luokka toimii kapellimestarina lokiviestien välittämisessä, ja se tulisi olla käytettävissä mistä tahansa sovelluksesta. TurvattuLokinTuottaja-luokkaan tehdään KirjaaLokiin-metodi, jolle välitetään lokiin kirjattava tieto, so. SQL-komento tai syöttökenttätunniste–arvo-pari. Syöttökenttätunniste–arvo-paria varten KirjaaLokiin-metodia kuormitetaan, jotta se voi ottaa vastaan joko kaksi tai vain yhden parametrin. Muita session_log-taulun tarvitsemia tietoja ei välitetä, sillä niiden tuottamisesta ja toimittamisesta eteenpäin huolehtii TurvattuLokinTuottaja-luokka. Tällä tavoin KirjaaLokiin-kutsu saadaan mahdollisimman yksinkertaiseksi. Prosessin lopussa LokiKirjuri-luokka tekee sille toimitetusta lokitiedosta uuden tietueen session_log-tauluun.

Ohjelmalistauksissa 7.1, 7.2 ja 7.3 on esitetty esimerkkitoteutukset luokille Allekirjoittaja, Salaaja ja Tiivistäjä. Esimerkkitoteutukset pohjautuvat löyhästi lähteessä [21] Turvattu Lokin Tuottaja -mallille esitettyihin Java-pohjaisiin esimerkkikoodeihin. Luokat vaativat toimiakseen erillisiä PHP-komponentteja: Kaikki kolme vaativat OpenSSL-paketin asentamisen ja Salaaja-luokka vielä lisäksi mcrypt-kirjaston.

Ohjelmalistaus 7.1 (Allekirjoittaja-luokan PHP-pohjainen esimerkkitoteutus.)

```
<?php
class Allekirjoittaja {

   public static final function allekirjoita(&$viesti)
   {
      # Haetaan henkilökohtainen (private) avain tiedostosta.
      $fp = fopen("/polku/avaimen/hakemistoon/avain.pem", "r");
      $hkavain = fread($fp, 8192);
      fclose($fp);
      $hkavaintunniste = openssl_get_privatekey($hkavain);

      # Allekirjoitetaan viesti hkavaintunnistetta ja
      # SHA-1 -tiivistealgoritmia käyttäen.
      # Allekirjoitus välittyy allekirjoitus-parametrissa.
      openssl_sign($viesti, $allekirjoitus, $hkavaintunniste, OPENSSL_ALGO_SHA1);

      # Vapautetaan avain muistista.
      openssl_free_key($hkavaintunniste);

      # Lokikirjausta varten allekirjotus koodataan Base64-algorimilla
      # ja palautetaan koodattu allekirjoitus.
      return base64_encode(string $allekirjoitus);
   }

}
?>
```

Ohjelmalistaus 7.2 (Salaaja-luokan PHP-pohjainen esimerkkitoteutus.)

```
<?php
class Salaaja {
```

```
   public static function noudaAvainVarastosta()
   {
      # Luokka on tarkoitettu julkisen avaimen salauksella
      # salatun avaimen noutamiseen varastosta.
      # Toteutus sivuutetaan.
   }

   public static final function salaa(&$viesti)
   {
      # Haetaan salausavain varastosta.
      $avain = $this->noudaAvainVarastosta();

      # Avataan moduuli DES-salausalgoritmille ja moodille cfb.
      # Algoritmi ja moodi saadaan oletushakemistoista, koska
      # toinen ja neljäs parametri ovat tyhjiä.
      $td = mcrypt_module_open('des', '','cfb', '');

      # Huolehditaan, että avain ei ole liian pitkä.
      $avain = substr($avain, 0, mcrypt_enc_get_key_size($td));

      # Haetaan IV:lle koko avatun moduulin algoritmille ja moodille.
      $iv_size = mcrypt_enc_get_iv_size($td);

      # Luodaan IV (eli alustusvektori) satunnaislukulähteelle.
      $iv = mcrypt_create_iv($iv_size, MCRYPT_DEV_RANDOM);

      # Alustetaan puskurit salausta varten.
      mcrypt_generic_init($td, $avain, $iv);

      # Salataan viesti.
      $salattu = mcrypt_generic($td, $viesti);

      # Sammutetaan salausmoduuli.
      mcrypt_generic_deinit($td);
      mcrypt_module_close($td);

      # Lokikirjausta varten salaus koodataan Base64-algorimilla
      # ja palautetaan koodattu salaus.
      return base64_encode(string $salattu);
   }
}
?>
```

Ohjelmalistaus 7.3 (Tiivistäjä-luokan PHP-pohjainen esimerkkitoteutus.)

```
<?php
class Tiivistäjä {

   public static final function tiivista(&$viesti)
   {
      # Tiivistetään viesti SHA-1-tiivistealgoritmilla.
      # Tässä voitaisiin käyttää myös MD5-tiivistealgoritmia, jolloin
      # funktiokutsu olisi md5(string $viesti).
      $tiiviste = sha1(string $viesti);
```

```
      # Lokikirjausta varten tiiviste koodataan Base64-algorimilla
      # ja palautetaan koodattu tiiviste.
      return base64_encode(string $tiiviste);
   }
}
?>
```

Tutkitaan seuraavaksi, mistä TurvattuLokinTuottaja-luokkaa tulisi kutsua sovelluksessa. Liikuntakalenterin koodia tarkastelemalla voidaan havaita, että Session_inputs-taulua käsitellään vain Database-luokassa. Database-luokka on tarkoitettu rajapinnaksi sovelluksen ja tietokannan välille. Database-luokan kautta tietokantaan tallennetaan ja siitä poistetaan tietoa. Sovelluksen arkkitehtuuri ei täysin toteudu tältä osin, sillä koodia läpikäymällä havaitaan seuraavat asiat:

- Tietokantaan kohdistuu poistoja luokissa Database ja Email. Email-luokassa tapahtuvat poistot liittyvät tauluun email, jossa Email-oliota vastaava tieto poistetaan taulusta. Database-luokassa poisto liittyy session tuhoamiseen. Session tuhoamisessa session_input-taulusta poistetaan sessioon liittyvät tiedot.

- Tietokantaan tehdään lisäyksiä luokissa Database, Email ja Translations. Email-taulussa tallennetaan Email-oliolle kerrottu uusi sähköpostiosoite email-taulun tietueeksi. Translations-luokassa translation-tauluun lisätään uudelle käännökselle tietue. Database-luokassa tietokantalisäykset liittyvät uuden käyttäjän lisäämiseen järjestelmään ja ryhmiin, käyttäjän syöttämien uusien arvojen tallentamiseen session_inputs-tauluun ja uuden session tallentamiseen session-tauluun.

- Tietokannan tietoja päivitetään luokissa Database, User ja Session. User-luokassa päivitetään käyttäjän perustietoja, kuten nimeä tai salasanaa. Session-luokassa päivitetään language-taulun oletuskieltä session pohjalta. Database-luokassa päivitykset kohdistuvat session_inputs-tauluun, johon päivitetään sessioon liittyviä käyttäjän syöttämiä arvoja.

Kaikki edellä mainitut tapaukset on syytä kirjata lokiin, jolloin TurvattuLokinTuottaja-luokkaa tulee kutsua luokista Database, Email, Translations, User ja Session, joissa tulisi olla attribuuttina viite TurvattuLokinTuottaja-olioon.

7.2.4 Esimerkkitapaus

Tutkitaan esimerkkitapauksen avulla, kuinka Turvattu Lokin Tuottaja -malliin ja sen Turvattu Tiedonkeruu -strategiaan pohjautuva loki toteutetaan liikuntakalenteriin.

Otetaan esimerkkitapaukseksi jo aiemmin liikuntakalenterista kuvattu toiminta, jossa lisätään tai päivitetään session_inputs-tauluun käyttäjän syöttämät arvot (kts. sekvenssikaavio 5.3).

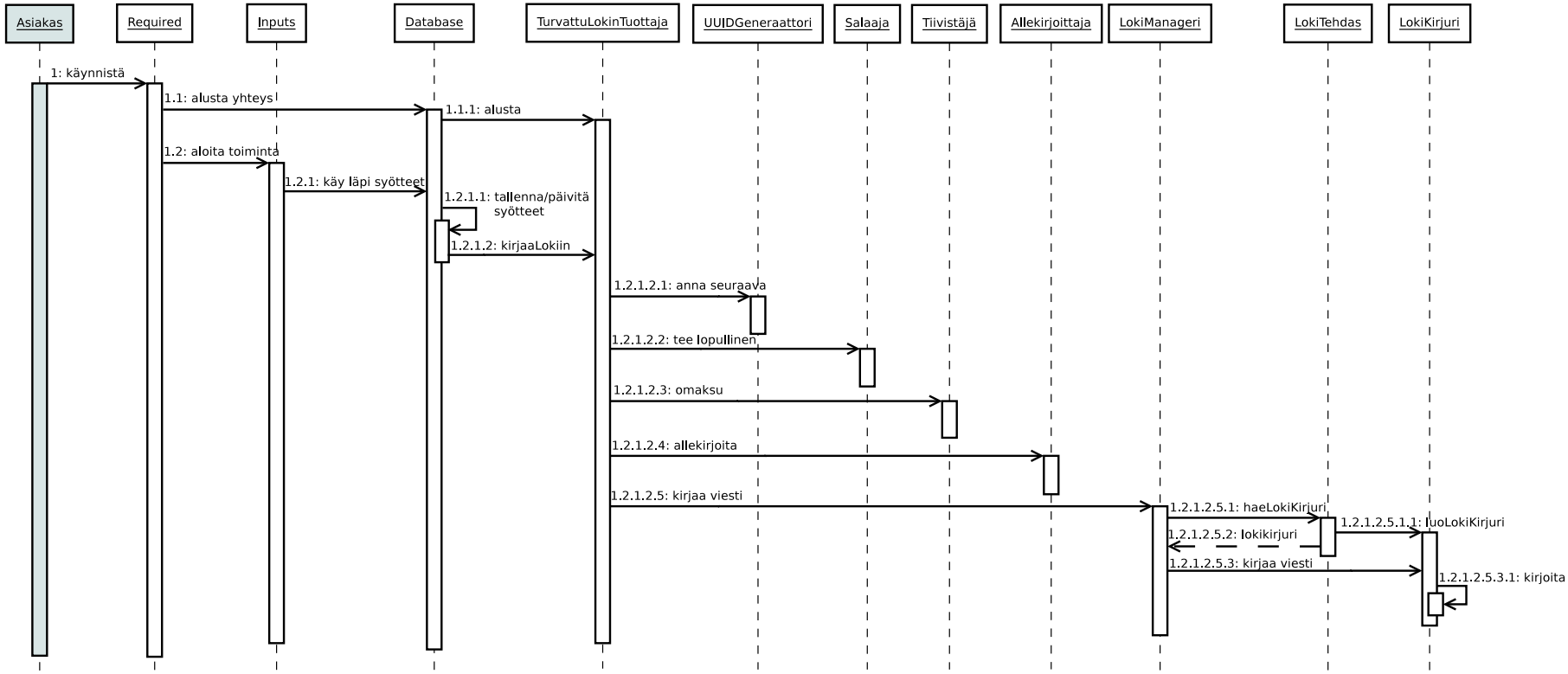
Syöttöjen lisäys ja päivitys tapahtuu Database-luokan metodissa save_inputs. Uutta lokitoiminnallisuutta varten metodin koodia tulisi täydentää siten, että siinä kutsuttaisiin TurvattuLokinTuottaja-luokan KirjaaLokiin-metodia jokaiselle tallennettavalle syötölle ja vielä erikseen tallennuksen suorittavalle SQL-komennolle. Ohjelmalistauksessa 7.4 esitetään save_inputs-metodin koodi, johon on lisätty tällainen lokitoiminnallisuus. Lisätyt rivit on kommentoitu lisäämällä rivien loppuun ”#lisätty”-kommentti. Koodissa oletetaan, että Database-luokassa on aiemmin alustettu TurvattuLokinTuottaja-olio muuttujaan turvattulokintuottaja_obj.

Ohjelmalistaus 7.4 (Database-luokan save_inputs-metodi lokitoiminnallisuudella.)

```
function save_inputs()
{
   if (is_array($_POST) && count($_POST) > 0)
   {
      foreach ($_POST as $input_id => $value)
      {

         if (is_array($value)) { $value = serialize($value); }

         $this->turvattulokintuottaja_obj->kirjaaLokiin($input_id,$value); #lisätty

         $sql  = "SELECT COUNT(*) FROM session_inputs ";
         $sql .= "WHERE sessionid = '".session_id()."' AND ";
         $sql .= "input_id = '".$input_id."'";

         $res = mysql_query($sql);

         $row = mysql_fetch_row($res);
         if ($row[0] > 0)
         {
            $sql  = "UPDATE session_inputs SET value = '".$value."' ";
            $sql .= "WHERE input_id = '".$input_id."' AND sessionid = '".session_id()."'";
            #echo $sql;
            $res = mysql_query($sql);
         }
         else
         {
            $sql  = "INSERT INTO session_inputs (sessionid, input_id, value) ";
            $sql .= "VALUES ('".session_id()."', '".$input_id."', '".$value."')";
            #echo $sql;
            $res = mysql_query($sql);
         }
         $this->turvattulokintuottaja_obj->kirjaaLokiin($sql); #lisätty
      }
   }
}
```

Tarkastelemalla Database-luokan save_inputs-metodia voidaan havaita, että myös pelkkä SQL-komentojen kirjaus lokiin riittäisi, sillä SQL-komennossa on mukana myös syöttökenttätunniste–arvo-parin sisältämä tieto. Syöttökenttätunniste–arvo-pari helpottaa kuitenkin lokin analysointia ja on siksi hyvä tallentaa lokiin erikseen.

Kuvassa 7.1 havainnollistetaan sekvenssikaavion avulla, kuinka Turvattu Lokin Tuottaja -malli ja sen Turvattu tiedonkeruu -strategia toimii yhdessä syöttökenttien tallennus- ja päivitystoiminnallisuuden kanssa. Sekvenssikaaviossa oletetaan, että sovelluksen arkkitehtuuri on yhdenmukainen ja vain Database-luokkaa käytetään tietokantamuutoksien tekoon. Silloin Database-luokka voi huolehtia myös TurvattuLokinTuottaja-olion perustamisesta. Ilman olettamusta tulisi TurvattuLokinTuottaja-olio luoda jo Required-luokassa.



Kuva 7.1: Sekvenssikaavio, joka kuvaa syöttökenttien tallennusta ja päivitystä tietokantaan ja niiden tallentamista lokiin Turvattu Lokin Tuottaja -mallin Turvattu tiedonkeruu -strategiaa käyttäen.

7.3 Johtopäätöksiä

Turvattu Lokin Tuottaja -malli soveltuu hyvin liikuntakalenteriin, sillä sitä varten ei tarvitse perustaa kuin yksi uusi taulu tietokantaan. Lisäksi osittain jo valmiiksi keskitetyn tietokannan käsittelyn johdosta TurvattuLokinTuottaja-luokkaa ei tarvitse kutsua useasta kohtaa sovellusta. Työläimmäksi osioksi jää mallin tarvitsemien luokkien toteuttaminen.

Uusi loki on tehokkaan salauksen ansiosta turvallinen ja se kattaa käytännössä kaikki muutokset, mitä järjestelmään kohdistuu. Se mahdollistaa (ainakin teoriassa) järjestelmän palauttamisen sessiota edeltäneeseen tilaan, koska myös kaikki tietokantaan kohdistuneet muutokset ovat tiedossa. Tietokantaan tallennettava datamäärä kasvaa kuitenkin näin kattavan lokitiedon johdosta erittäin suureksi. Tähän olisi syytä varautua esimerkiksi session_log-taulun sovittua vanhempien tietueiden automaattisella poistolla.

Turvallisimmastakaan lokista ei ole hyötyä, jos lokista ei pysty helpolla tavalla havaitsemaan oleellisia asioita, eli mahdollisia väärinkäytöksiä tai järjestelmävikoja. On erittäin työlästä, ellei mahdotontakin, käydä käsin läpi massiivinen lokidata ja poimia sieltä arveluttavat tiedot. Onkin välttämätöntä tehdä kelvolliset työkalut lokin analysointiin, joiden avulla lokista voidaan ohjelmallisesti poimia epäilyttävät kohdat ja esittää ne selkeällä tavalla järjestelmän ylläpitäjälle. Lokityökalujen olisi syytä jatkuvasti tarkkailla lokia ja ilmoittaa ylläpitäjälle, jos epäilyttävää toimintaa havaitaan. Työkalujen tulisi myös antaa selkeä raportti kunkin käyttäjän toimista valitulla ajanjaksolla, jolloin myös ihmissilmin voitaisiin todeta väärinkäytökset. Tämä voi osoittautua paljon suoritustehoa vaativaksi toimenpiteeksi, sillä analysointia varten tulee jokaisen session_log-taulun tietueen logdata-kentän salaus purkaa. On myös muistettava varmistua jälleen siitä, että purettu data ei joudu vääriin käsiin.

Jo olemassa olevan tietokannan käyttö lokitarkoituksiin saattaa pidemmällä ajanjaksolla tuoda esiin odottamattomia ongelmia suoritustehon ja tallennuskapasiteetin riittävyyden suhteen. Tällöin joudutaan mahdollisesti siirtymään erillisen turvatun lokivaraston käyttöön. Erillinen lokivarasto vaatii Turvattu lokin varastointi -strategian käyttöönoton, mikä voidaan tehdä kooditasolla seuraavalla tavalla:

- TurvattuLokinTuottaja-luokasta poistetaan viittaukset luokkiin UUIDGeneraattori, Salaaja, Tiivistäjä ja Allekirjoittaja.
- Lisäämällä Turvattu Putki -mallin kuvaama toiminnallisuus LokiKirjuri-luokan ja uuden lokivaraston välille.
- Muuttamalla LokiKirjuri-luokka käyttämään uutta lokivarastoa ja Turvattu

Putki -mallia.

Edellä mainitut muutostarpeet voidaan havaita vertailemalla Turvattu Lokin Tuottaja -mallin toteutusstrategioita. On kuitenkin syytä harkita vakavasti suoraan erillisen varaston ja Turvattu lokin varastointi -strategian käyttöä, varsinkin jos erillinen varasto voitaisiin toteuttaa vähäisin kustannuksin.

8 Yhteenveto

Tässä tutkielmassa turvallisuutta tarkasteltiin turvallisuustekniikaa, tietoturvamalleja ja liikuntakalenteri-esimerkkisovellusta apuna käyttäen. Tietoturvamalleista tarkasteluun otettiin useista eri mallikokoelmista koottu ydinmallikokoelma. Ydinmallien luokitus käytiin läpi perusteellisesti ja se pyrittiin esittämään lukijalle mahdollisimman selkeällä tavalla.

Tietoturvamallien integroimisprosessi olemassa olevaan sovellukseen havaittiin (karkealla tasolla) kaksivaiheiseksi prosessiksi. Ensimmäisessä vaiheessa tulee tehdä tietoturva-analyysi, jotta sopivat tietoturvamallit voidaan löytää. Toinen vaihe sisältää mallien varsinaisen integroinnin ja toteutuksen.

Liikuntakalenterisovellukselle valittiin tietoturva-analyysin perusteella yksi konkreettinen uhka, joka tulisi torjua. Torjumisen edellytykset selvitettiin, ja niiden perustella valittiin uhkan torjumiseen parhaiten sopiva tietoturvamalli ydinmallien joukosta. Valinta kohdistui Turvattu Lokin Tuottaja -malliin. Mallin integroimisprosessista liikuntakalenteriin esitettiin kattava ja selkeä korkeamman tason esitys. Apuna käytettiin muun muassa luokka- ja sekvenssikaavioita. Esitys sisälsi mallin tarkastelua osittain myös matalalla kooditasolla. Mallin toimintaa tarkasteltiin myös esimerkkitapauksen avulla.

Koska tutkielmassa pyrittiin keskittymään tietoturvamallien hyödyntämiseen sovelluskehityksessä, jätettiin mallin yksityiskohtainen toteutus liikuntakalenteriin tekemättä. Haittapuolena tästä on empiiristen tutkimustulosten puuttuminen mallin käytännön vaikutuksista sovelluksen toimintaan. Luonnollinen tapa jatkaa tutkimusta olisi toteuttaa Turvattu Lokin Tuottaja -malli kokonaisuudessaan liikuntakalenteriin ja tarkastella kuinka malli ja sille tutkielmassa esitetty ingeroimisprosessi käytännössä toimivat. Lisäksi voitaisiin tarkastella muun muassa mallin vaikutuksia sovelluksen suorituskykyyn. Jatkossa olisi myös mielenkiintoista integroida muita tietoturvamalleja liikuntakalenteriin.

Yksi tutkielman aihetta sivuava laajempi tutkimuskohde on uusien tietoturvamallien tuottaminen. Olisi hyvin haasteellista toteuttaa tietoturvamalli sellaiselle turvallisuustavoitteelle, jolle ei vielä ole olemassa yhtäkään siihen nimenomaan keskittyvää tietoturvamallia.

Tämä tutkielma osoitti minulle, että tietoturvamalleista on todella hyötyä. Uskon ja toivon, että tietoturvamallien käyttö sovelluskehityksessä tulee kasvamaan

ja sovellusten turvallisuustaso tämän ansiosta nousemaan.

Lähteet

[1] Anderson Ross, *Security Engineering, A Guide to Building Dependable Distributed Systems*, John Wiley & Sons Inc, New York, 2001.

[2] Premkumar T. Devanbu ja Stubblebine Stuart, *Software Engineering for Security: a Roadmap*, saatavilla PDF-muodossa <URL: `http://www.cs.ucl.ac.uk/staff/A.Finkelstein/fose/finaldevanbu.pdf`>, viitattu 14.10.2006.

[3] Schumacher Markus, *Security Engineering with Patterns, Origins, Theoretical Model and New Applications*, Springer-Verlag, Berlin Heidelberg, 2003.

[4] McGraw Gary, *Software security*, Security & Privacy Magazine, Volume 2, Issue 2, s. 80-83, Mar-Apr 2004.

[5] Andrews Mike ja Whittaker James A., *Computer Security*, Security & Privacy Magazine, Volume 2, Issue 5, s. 68-71, Sept-Oct 2004.

[6] Schumacher Markus ja Roedig Utz, *Security Engineering with Patterns*, saatavilla PDF-muodossa <URL: `http://www.cs.ucc.ie/misl/publications/files/plop01schumacher.pdf`>, 27.7.2001.

[7] Tietoturvallisuuden perus- ja jatkokurssi, Tampereen teknillinen yliopisto / Tietoliikennetekniikka, *Tietoturvasuunnittelun malliratkaisut*, saatavilla WWW-muodossa <URL: `http://sec.cs.tut.fi/maso/materiaali.php?id=473`>, 12.4.2005.

[8] Howard Michael ja LeBlanc David, *Writing Secure Code, Practical strategies and techniques for secure application coding in a networked world*, Microsoft Press, Redmond, Washington, 2003.

[9] Pfleeger Charles P. ja Pfleeger Shari Lawrence, *Security in Computing*, Prentice Hall Professional Technical Reference, Upper Saddle River, New Jersey, 2003.

[10] Systems Security Engineering Capability Maturity Model (SSE-CMM) Project, *Systems Security Engineering Capability Maturity Model SSE-CMM*

Model Description Document Version 3.0, saatavilla PDF-muodossa <URL: http://www.sse-cmm.org/docs/ssecmmv3final.pdf>, 15.6.2003.

[11] Outback Software, Ltd., *Computer security*, saatavilla WWW-muodossa <URL: http://www.outbacksoftware.com/security/security.html>, viitattu 14.5.2006.

[12] Stanley A. Klein ja James N. Menendez, *Information Security Considerations in Open system Architectures*, IEEE Transactions on Power Systems, Vol. 8, No. 1, s. 224-230, February 1993.

[13] Sommerville Ian, *Software Engineering, 8th edition*, Addison-Wesley, Pearson Education, Hawlow Essex, 2007.

[14] Alexander Christoper, *The Timeless Way of Building*, Oxford University Press, New York, 1979.

[15] Alexander Christoper, Ishikawa Sara, Silverstein Murray, Jacobson Max, Fiksdahl-King Ingrid ja Angel Shlomo, *A Pattern Language: Towns, Buildings, Construction*, Oxford University Press, New York, 1977.

[16] Kent Beck ja Cunningham Ward, *Using Pattern Languages for Object-Oriented Programs*, OOPSLA-87 workshop Technical Report No. CR-87-43, Apple Computer Inc., Tektronix Inc., 17.9.1987.

[17] Gamma Erich, Helm Richard, Johnson Ralph ja Vlissides John, *Design Patterns: Elements of Reusable Object-Oriented Software*, Addison-Wesley, Pearson Education, Upper Saddle River, New Jersey, 1995.

[18] Kienzle Darrell M., Elder Matthew C., Tyree David ja Edwards-Hewitt James, *Security Patterns Repository Version 1.0*, saatavilla PDF-muodossa <URL: http://www.scrypt.net/čeler/securitypatterns/repository.pdf>, viitattu 12.8.2007.

[19] Blakley Bob, Heath Craig ja The Open Group Security Forumin jäsenet, *Security Design Patterns Technical Guide*, The Open Group, U.K., 2004.

[20] Schumacher Markus, *Security Patterns Homepage*, saatavilla WWW-muodossa <URL: http://www.securitypatterns.org/patterns.html>, viitattu 12.8.2007.

[21] Steel Christopher, Nagappan Ramesh ja Lai Ray, *Core Security Patterns*, Best Practices and Strategies for J2EE, Web Services, and Identity Management, Prentice Hall PTR, Upper Saddle River, NJ, 2005.

[22] Nagappan Ramesh, *Core Security Patterns: Best Practices and Design strategies for J2EE, Web Services, Identity Management and Service Provisioning*, saatavilla WWW-muodossa <URL: `http://www.coresecuritypatterns.com/patterns.htm`>, viitattu 13.8.2007.

[23] Yskout Koen, Heyman Thomas, Scandariato Riccardo ja Joosen Wouter, *A system of security patterns*, saatavilla PDF-muodossa <URL: `http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW469.pdf`>, viitattu 12.8.2007.

[24] Schumacher Markus, Fernandez-Buglioni Eduardo, Hybertson Duane, Buschmann Frank ja Sommerlad Peter, *Security Patterns*, Integrating Security and Systems Engineering, John Wiley & Sons Ltd, Chichester, England, 2006.

[25] Chess David M., *Security Issues in Mobile Code Systems*, G. Vigna (Ed.): Mobile Agents and Security, LNCS 1419, pp. 1–14, Springer-Verlag, Berlin Heidelberg 1998.

[26] Pitkänen Raimo, *Käytettävyyssuunnittelu osana ohjelmistoprosessia*, Pro gradu -tutkielma, Jyväskylän yliopisto, Tietotekniikan laitos, joulukuu 2006.

[27] Pitkänen Raimo, *Liikuntakalenteri*, Erikoistyö, Jyväskylän yliopisto, Tietotekniikan laitos, elokuu 2006.

[28] Pitkänen Raimo, *Personal Communication*, syksy 2007 - kevät 2008.

[29] Losavio, Francisca, Dinarle Ortega ja María Pérez, *Towards a Standard EAI Quality Terminology*, Proceedings of the XXIII International Conference of the Chilean Computer Science Society (SCCC'03), s. 195-203, 6-8 November 2002.

[30] Avizienis Algirdas, Laprie Jean-Claude ja Randell Brian, *Fundamental Concepts of Dependability*, Research Report N01145, LAAS-CNRS, April 2001.

[31] Abran Alain, Khelifi Adel, Seffah Ahmed ja Suryn Witold, *Usability Meanings and Interpretations in ISO Standards*, Software Quality Journal, 11, s. 325-338, Kluwer Academic Publishers, 2003.

[32] Princeton University, *WordNet Search - 3.0*, saatavilla WWW-muodossa <URL: http://wordnet.princeton.edu/perl/webwn>, viitattu 15.3.2008.

[33] European Central Bank, *Report on Electronic Money*, saatavilla PDF-muodossa <URL: http://www.ecb.int/pub/pdf/other/emoneyen.pdf>, Aug 1998.

[34] Fox Chris ja McDermott John, *Using abuse case models for security requirements analysis*, Computer Security Applications Conference, 1999. (ACSAC '99) Proceedings. 15th Annual, s. 55-64, Jun-Oct 1999.

[35] Delessy-Gassant Nelly, Fernandez Eduardo B., Larrondo-Petrie Maria M. ja Rajput Saeed, *Patterns for Application Firewalls*, PLoP, 2004.

[36] Schumacher Markus, *Firewall Patterns*, EuroPLoP, 2003.

[37] Barcalow Jeffrey ja Yoder Joseph, *Architectural Patterns for Enabling Application Security*, PLoP, 1997.

[38] Saridakis Titos, *Design Patterns for Fault Containment*, EuroPLoP, 2003.

[39] Sorensen Kristian Elof, *Session Patterns*, EuroPLoP, 2002.

[40] Sommerlad Peter, *Reverse Proxy Patterns*, EuroPLoP, 2003.

[41] Berbers Yolande, Hovsepyan Aram, Joosen Wouter ja Van Baelen Stefan, *Generic Reusable Concern Compositions (GReCCo): Description and Case Study*, Katholieke Universiteit Leuven, Department of Computer Science, Heverlee, Belgia, Jan 2008.

[42] The PHP Group, *PHP: Predefined Variables - Manual*, saatavilla WWW-muodossa <URL: http://fi.php.net/reserved.variables>, 25.11.2007.

[43] Wang Xiaoyun ja Yu Hongbo, *How to Break MD5 and Other Hash Functions*, Shandong University, Jinan 250100, China, 2007.

[44] Rescorla E., *HTTP Over TLS*, saatavilla TXT-muodossa <URL: http://www.ietf.org/rfc/rfc2818.txt>, RFC 2818, May 2000.