23.5.2012

Lausunto sovelluskehityksen VAHTI-ohjeesta

Sovelluskehityksen merkitys tietoturvallisuudessa on erittäin suuri. Ihminen on tietoturvallisuuden heikoin lenkki, mutta hyvin usein inhimilliset virheet muodostavat tietoturvariskin vain koska tietojärjestelmässä on heikkous. Kyse on kahden tekijän yhteisvaikutuksesta, eikä teknisiä asioita sovi väheksyä. Lisäksi monet viimeaikaisista tietoturvahaavoittuvuuksista ovat sellaisia, ettei käyttäjien ohjeistuksella ja koulutuksella voida korvata tietojärjestelmässä olevia puutteita.

Tietojärjestelmän tietoturvallisuuden perusta on järjestelmän valmistaminen eli sovelluskehitys.
VAHTI on laatinut uuden ohjeen sovelluskehityksen tietoturvallisuudesta. Ohje korvaa vanhan, jo pitkään hyvin palvelleen ja osin vanhentuneen ohjeen.

Luonnoksen (versio 0.9) oleva ohje sovelluskehityksestä on yleisesti hyvin laadittu. Se on hyvin kirjoitettu, rakenne on selkeä ja viittauksia muihin VAHTI-ohjeisiin on hienosti. Käytännön sovelluskehitystä helpottanee selkeät viittaukset tietoturvatasoihin, jotka on joissain sovelluskehitysprojekteissa koettu ongelmalliseksi. Ohjeen kattavuus on pääasiassa hyvä, ja se on selkeästi kirjoitettu. Suositusten numerointi ansaitsee erityiskiitoksen.

Ohjeen aihe koskee Nixua Oy:tä voimakkaasti. Yksi neljästä osastostamme, Develop, on erikoistunut turvalliseen sovelluskehitystyöhön, ja sen menetelmien kehittämiseen. Työtä teemme etupäässä suurissa kansainvälisissä yhtiöissä, mutta myös valtionhallinnossa. Emme oel osa valtionhallintoa, joten lausuntomme painottuu voimakkaasti ohjeen substanssiin – miten turvallisesti tehdä softaa – ei asian jalkauttamiseen julkishallinnossa.

Käytännön ohjelmistokehitystyössä hyvinä olemme pitäneet BSIMM3- ja OpenSAMM-kehyksiä, ja olemme antaneet kommenttimme näiden valossa. Kehyksiin voi tutustua osoitteissa http://bsimm.com/ ja http://www.opensamm.org/. Nämä ovat laajasti käytössä ja arvostettuja.

Hyvässä ohjeessa on vielä parantamisen varaa. Olemme jakaneet kommenttimme yleisiin, useaa sekä yksittäistä kohtaa koskeviin.

Olli-Pekka Soini
Senior Security Consultant
CISSP, CISA, CISM, CRISC

Ari Kesäniemi
Senior Security Architect
GSSP-Java

1 Yleiset kommentit

1.1 Organisointi ja roolit

Sovelluskehitystä tekevän organisaation sisällä toimivaa tietoturvatahoa tulisi korostaa nykyistä enemmän. Asiaa on käsitelty ohjeen kohdassa 2.4 ja vaatimuksessa OSK-006. Tätä suosittelee sekä BSIMM3 että OpenSAMM.

BSIMM3:

> *”The second most important role in a software security initiative after the senior executive is that of the Software Security Group. Every single one of the forty-two successful programs we describe in the BSIMM has a sizeable SSG. Carrying out the activities in the BSIMM successfully without an SSG is very unlikely (and has never observed in the field to date), so create an SSG before you start working to adopt the BSIMM activities. The best SSG members are software security people, but software security people are often impossible to find. If you must create software security types from scratch, start with developers and teach them about security.”*

Lisäksi: http://www.informit.com/articles/article.aspx?p=1434903 (”You Really Need a Software Security Group”)

OpenSAMM (s. 44) on samoilla linjoilla:

> *"Utilize security coaches to enhance project teams*
>
> *Using either internal or external experts, make security-savvy staff available to project teams for consultation. Further, this coaching resource should be advertised internally to ensure that staff are aware of its availability. [...] While coaches can be used at any point in the software life-cycle, appropriate times to use the coaches include during initial product conception, before completion of functional or detailed design specification(s), when issues arise during development, test planning, and when operational security incidents occur."*

Tietoturvataho voisi kulkea tietoturvaryhmän nimellä. Ohjeessa mainittu sisäinen tarkastus voi sinänsä hoitaa ainakin osaa tehtävistä, mutta terminä se luo mielikuvaa reaktiivisesta, jälkikäteen tapahtuvasta toiminnasta: tarkastuksesta, ei auttamisesta. Olennaista on, että taho toimii konkreettisesti ja aktiivisesti projektien kanssa, ja on helposti saatavilla ja lähestyttävissä. Sisäinen tarkastus voi toki toimia näin.

Sisäisen tarkastuksen yksikkö löytyy suurimmasta osaa julkishallinnon organisaatioista, ohjelmistokehityksen tukee kykenevä tietoturvaryhmä on harvinaisempi. Osaamista voi ostaa, ja suurissa kehityshankkeissa tämä on perusteltua. Joka tapauksessa, ehdotamme, että viittaukset sisäiseen tarkastukseen korvataan niiltä osin, kuin kyseessä ei ole varsinainen tarkastustoiminta

1.2 Ohjeet turvallisesta sovelluskehityksestä

Toteutusohje ei ole kovin kattava, jos sitä vertaa esimerkiksi OWASP Top 10 -listaan, puhumattakaan laajemmista listoista ja yleisesti tiedossa olevista tyypillisistä toteutusvirheistä. Painoa on annettu liiaksi lokien käsittelylle, joka toki on tärkeä asia, mutta ei oikeassa suhteessa muihin toteutusta koskeviin suosituksiin.

Toteutusohjeissa ei ole todennäköisesti järkevää listata konkreettisia toteutusasioita, vaan viitata yleisesti käytössä oleviin ohjeisiin, esim. OWASP:n Secure Coding Practices Quick Reference Guide (https://www.owasp.org/index.php/OWASP_Secure_Coding_Practices_-_Quick_Reference_Guide).

Olennaisinta toteutusohjeessa olisi kuitenkin painottaa oikeita työmenetelmiä ja -periaatteita,kuten

- defensiivinen ohjelmointi
- koodikatselmointikäytännöt (ks. BSIMM s. 33)
- turvallisen ohjelmistokehityksen ohjeistus ja referenssitoteutus (ks. BSIMM s. 29)
- arkkitehtuuriratkaisujen ja yhteiskäyttöisten komponenttien rooli tietoturvassa (esimerkiksi OWASP ASVS:ään ja *Quick Reference Guiden* -ohjeisiin suhteutettuna). Kunkin organisaation tulisi itse pystyä korostamaan toisaalta sovellusten vastuuta tietoturvasta ja toisaalta arkkitehtuuriratkaisujen merkitystä. (ks. BSIMM s. 27)

1.3 Uhka-analyysi

Uhka-analyysi ja siihen liittyvät muut aktiviteetit eivät ole ohjeessa selkeästi määritellyt, vaan toistuvat osittain ja useammassa kohdassa. Olisi järkevää sitoa uhka-analyysi, riskianalyysi ja tietoturvavaatimukset niihin vaiheisiin, joissa näitä tehdään, mutta miettiä ensin uhkien käsittely kokonaisuutena. Usein käytetty ja hyväksi havaittu jako on:

- Vaikutusanalyysi (BIA *Business Impact Analysis*) esiselvitys- tai määrittelyvaiheessa
- Tietoturvavaatimusten määrittely määrittelyvaiheessa tai viimeistään määrittelyn tarkennus –vaiheessa. Tämä tehdään edellisen kohdan vaikutusanalyysin perusteella
- Uhka-analyysin vuoro on suunnitteluvaiheessa, Toteutustiimin on järkevää olla tässä mukana, eli tämä tulisi tehdä vasta, kun toteutus on kilpailutettu tai toteutustiimi muuten valittu.
- Teknisen suunnitelman täydentäminen uhka-analyysissä tunnistettujen uhkien ja niihin liittyvien suojausten perusteella
- Riskien hallinta tunnistettujen uhkien perusteella koko projektin ja järjestelmän käytön ajan

Kannattaa kuitenkin huomioida, että tarkka työnkulku riippuu organisaation sovelluskehitysmallista, organisaation rakenteesta, arkkitehtuuriratkaisuista ja muista tekijöistä. Tärkeintä on kuitenkin saada uhkat mahdollisimman kattavasti kartoitettua ja sidottua teknisiin ratkaisuihin ja riskien hallintaan ja tätä kautta oikeisiin toteutusratkaisuihin.

Uhkien analysointi on BSIMM:ssä yksi kahdestatoista osa-alueesta (*practice*) ja siihen löytyy paljon lisää tietoa sieltä (BSIMM3 s. 25 alkaen). OpenSAMM:ssa myös uhkien analysointi on keskeinen asia ja siihen liittyy monia aktiviteettejä (ks. OpenSAMM s. 46 alkaen).

1.4 Luettavuus

Ohjeessa on paljon sisältöä muista ohjeista. Näitä kannattaisi karsia, tiivistää tai siirtää liitteisiin. Esimerkiksi lokiohjeen käyttöä voisi vähentää ja tyytyä viittaamaan siihen, mainiten lähinnä aihealueet, joita tulee huomioida. Tällöin myös kyseisen ohjeen päivittyessä ei tarvitse päivittää tätä ohjetta eli viittaus ei korruptoidu.

Lukemisen helpottamiseksi ohjeen jaottelua voisi kehittää, ja tuoda paremmin esille eri kohderyhmien tarpeet kuten projektipäälliköt ja sovelluskehittäjät). Projektipäällikön rooli on erittäin tärkeä, erityisesti suurissa projekteissa, ja mielellään lisäisimme ohjeeseen hänelle tarkoitettua sisältöä. Monet projektipäälliköt kokevat infoähkyä eli heidän on hallittava valtava määrä tietoa. Ei ole takeita, että aikaa 50-sivuisen ohjeen lukemiseen on, mutta 5-sivuiselle tiivistelmälle yleensä on.

Luettavuutta parantaisi tietoturvatasoviittaustaulukoiden (jotka ovat ehdottoman hyvä asia) siirtäminen tekstin sisältä pois esimerkiksi liitteeseen. Toisaalta viittaus tietoturvatasoihin ja muihin VAHTI-ohjeisiin

tulisi tehdä ohjetekstissä niin, että käy ilmi mikä on viittauksesta otettua sisältöä ja mikä ohjeen omaa tekstiä.

Joissain kappaleissa käytetään imperatiivia (esim. s. 36). Esimerkinomaisuutta kuvaavia lyhenteitä esim, mm. jne. voisi karsia.

2 Yksityiskohtaiset kommentit

Sivu 9: Luku Haasteita sovelluskehityksen tietoturvassa

Lista haasteista on hyvä, mutta lisäisimme siihen kohdan

- Ennalta määrittämättömiin tilanteisiin varautuminen

Kaikkeen ei osata varautua, vaikka kuinka hyvin tehdään. On varauduttava sellaiseen, josta ei vielä tiedetä, eikä voida tietää: on jokseenkin varmaa, että vastaan tulee ongelmia, joita ei ole osattu ennakoida. Tällöin järjestelmä, joka suunniteltu korjattavaksi eikä täydelliseksi on hyvä. Lisäksi järjestelmä on toteutettava niin, että se toimii hallitusti odottamattomissa tilanteissa.

s. 12: Kuva 3, Esimerkki sovelluskehityksen roolien tehtävistä hankkeen eri vaiheissa

Kuva on esimerkki, mutta olisi johdonmukaista, että kuvassa olisi mukana sisäinen tarkastus. Etenkin kun tälle toiminnolle on sälytetty vastuuta ohjeen muissa kohdissa. Näkisimme vielä parempana ratkaisuna erillisen tietoturvaryhmän, jota esitimme edellä.

s.14: Sisäinen tarkastus / tietojärjestelmätarkastus

Kuten edellä kerroimme, pidämme erillistä ryhmää parempana etenkin suurissa projekteissa. Kohta “Sisäisen tarkastus voi myös olla projektin tukena koko hankkeen elinkaaren ajan” on idealistinen: olisi hyvä, jos näin voisi olla, mutta käytännössä “voi” toteutuu harvoin.

s. 15: 2.6 Tietoturvallisuuden laadun varmistaminen / viimeinen kappale

Viimeinen kappale on erittäin tärkeä, ja olemme vahvasti samaa mieltä: katselmointeja, auditointeja tarkastuksia ja muita laadunvarmistustoimia ei ole aikataulutettu ja ne joudutaan usein tekemään kiireessä.

Ehdotamme lisäystä tilanteisiin, joissa tietoturvavastaavaa ei nimetä (täysin perusteltu ratkaisu joissain tilanteissa): ”Mikäli tietoturva-asiantuntijaa ei nimetä, on vastuu tietoturva-asioista projektipäälliköllä tai tietoturvaryhmällä, jos tällainen on käytettävissä.”

s. 15: 2.7 Tietoturvallisuuden dokumentointi osana sovelluskehitystä

Kohdassa oleva lista on hyvä, ja siinä on hyvin todettu, että asioiden oikea paikka saattaa olla osana muuta dokumentaatioita. Ehdotamme, että listaan lisään seuraavat kohdat

- Tietoturva-arkkitehtuuri. Erityisesti, miten on varmistettu, että järjestelmä on noudattaa organisaation kokonaisarkkitehtuuria
- Noudatettavat standardit ja muut normit, ja noudattamisen aste eli onko noudattaminen pakollista, ohjeellista vai jotain näiden väliltä

Kohdan lisääminen sitoo toisiinsa tietoturvatyötä ja kokonaisarkkitehtuurityötä, johon velvoitetaan laissa valtionhallinnon tietohallinnosta. Listassa oleva kohta tietoturvakuvaus on usein asiallisesti lähellä tietoturva-arkkitehtuurin kuvausta.

s.17: Sopimukset

Kappale on hyvin kirjoitettu, mutta näkisimme siinä käsiteltävän perinteisen sopimuskonstruktion lisäksi kahta muuta: sopimukset laaditaan toimittajan vakiosopimusten pohjalle sekä pilvipalvelut (etenkin SaaS), joissa ei ole samassa mielessä softaa, joka pyörii käyttöpalveluntoimittajalla. SaaS-palveluiden merkitys tulee kasvamaan.

s.19: 3.1.1 Strategia ja resursointi kappale STR-002: Perustaso

Kappaleessa on lueteltu riskejä, joista "Ohjelman purkaminen lähdekoodiksi **(reverse engineering)"** ei ole riski vaan hyökkäystapa.

s. 23: 3.1.3 Osaaminen ja koulutus kappale OSK-001: Perustaso

Kappaleessa esitetään, että sovelluskehittäjille pidetään vähintään kerran vuodessa tietoturvakolutusta, jonka kestoksi annetaan esimerkinomaisesti 1-2. Käytännön tasolla tämän noudattaminen on monissa julkishallinnon yksiköissä vaikeaa, vaikka koulutustarpeesta olemme samaa mieltä. Koska kyse on tietoturvallisuuden perustasosta, ehdotamme harkittavaksi, riittäisikö vähempi.

Koulutus on perusteltua antaa uusille kehittäjille, ja merkittävien muutosten yhteydessä.

s. 23 OSK-003: Perustaso

Kirjallinen materiaali koulutuksessa on tärkeää, mutta kokemuksemme on, että hyödyllisempää on kertoa, kuka antaa lisätietoja asiasta. Tämä voi olla tietoturvapäällikkö tai joku tietoturvaryhmän (tai vastaavan) jäsenistä. Ehdotamme asian lisäämistä kappaleeseen.

s. 25 Standardit / STA-001: Perustaso

Ohjeessa ehdotetaan kaikkien palveluiden, sovellusten ja niissä käytettyjen ohjelmakomponenttien luokittelua suojaustason mukaan. Sovellusten ja palveluiden suhteen asia on selvä, ja lienee myös pääosin kunnossa. Ulkoisten ohjelmakirjastojen luokittelu on käytännössä huomattavasti hankalampi vaatimus, ja edellyttää tietoa, mikä sovellus käyttää mitäkin komponenttia ja mitä tietoa kukin välittää millekin komponenteille. Ohjelmakomponenttien osalta on jo olemassa vaatimukset UHM-004 ja SNT-003.

Ehdotamme, että tätä ei vaadita ohjelmakomponenttien osalta ainakaan perustasolla.

s. 27 3.1.5 Tekninen sovelluskehitysympäristö / TSK-004: Perustaso

Kappaleessa on ilmeisesti oletettu tilanne, jossa kehitysympäristö sijaitsee etäällä. Käytännössä on usein niin, että sovelluskehitysympäristö on lokaali (etenkin pienkehityksessä), jolloin tietoturvamielessä tärkeää on lokaalin kirjaston päivittäminen hallitusti, ei yhteydet tähän. Ehdotamme mainintaa asiasta.

Etenkin Open Source –pohjaisessa kehittämisessä kirjastokomponenttien lataaminen (esim. ulkoisesta komponenttikirjastosta eli *repository*stä) on syytä tehdä keskitetysti, jotta tilanne pysyisi hallinnassa, eikä käytössä olisi monia eri versioita komponenteista, joiden tietoturvatasoa ei ole varmistettu. Ehdotamme mainintaa asiasta.

s. 30: 3.3 Sovelluskehityksen vaiheet ja tietoturvatasojen vaatimukset / ensimmäinen kappale

Kappaleessa on esitetty järjestelmäkehityksen osa-alueet implisiittisessä kronologisessa järjestyksessä. Kohta uhkien mallintaminen ei käytännössä toteudu esiselvityksen jälkeen, vaan sitä tehdään vaiheiden määrittely ja suunnittelu aikana, usein vielä muita vaiheen aktiviteettejä perässä. Uhkien mallintaminen kun edellyttää, että on tiedossa, mitä järjestelmä tulee tekemään, ja millä teknisillä ratkaisuilla se toteutetaan. Nämä asiat eivät kuulu esiselvitykseen.

Monasti esiselvitys jätetään tekemättä, jolloin järjestelmäkehityksen aloittaminen uhka-analyysillä vaikuttaa hullunkuriselta. Jos esiselvitys on tehty, on sen pohjalta mahdollista tehdä ylätason liiketoiminnallinen uhka-analyysi.

3.3.2 Uhkien mallintaminen

Kohta sisältää paljon asioita, joita ei ole käytettävissä esiselvityksen jälkeen, vaan vasta määrittely- tai suunnittelu vaiheessa.

Ehdotamme, että kohta uhkien mallintaminen siirretään vaatimusmäärittelyn jälkeen.

3.3.3 Vaatimusmäärittely

Kohdassa 3.3.2 edellytetään tehtävän uhka-analyysi ja kohdassa 3.3.3. riskianalyysi. Onko todella perusteltua edellyttää näiden molempien tekoa perustasolla?

Vaatimusmäärittelyssä ei yksittäisen vaatimuksen luokittelu ole aina helppoa. Onko esimerkiksi vaatimus tietystä vasteajasta tietoturvavaatimus vai suorituskykyvaatimus? Tämän asian kanssa projektipäällikkö painii, ja toivoisimme ohjeeseen lisää konkretiaa vaatimusmäärittelykohdassa. Tätä voisi olla esimerkit tietoturvavaatimuksesta ja sen käsittelystä.

Korostaisimme lisää seikkaa, että vaatimuksista suuri osa (muttei tietenkään kaikki) tulevat uhka- ja riskianalyysin pohjalta.

s. 33/ VTM-006: Korotettu taso

Väärinkäyttötapaukset (*abuse case)* ovat sinällään hyvä tapa hyökkäystapojen kartoittamiseen. Ohjelmistokehittäjät pitävät tapaa raskaana: jo keskikokoisessa vaatimusmäärittelyssä käyttötapausten määrä on kymmeniä. Lisäksi väärinkäyttötapauksilla ei voida kattaa monia hyökkäysskenaarioita, jotka eivät liity tiettyjen toimintojen väärinkäyttöön, vaan ne skenaariot löytyvät muulla tavalla (mm. uhkamallinnuksella).

Väärinkäyttötapaukset soveltuvat perinteiseen ohjelmistokehitykseen, jossa käyttötaukset ovat tärkeä väline, mutta heikommin uusiin kehitysmenetelmiin.

Ehdotamme, että väärinkäyttötapaukset mainitaan yhtenä mahdollisena työvälineenä. Toisaalta voi miettiä, onko vaatimus edes tarpeellinen, jos järjestelmälle on tehty huolellinen uhka-analyysi esimerkiksi STRIDE-menetelmällä.

s. 34 3.3.4 Suunnittelu / SNT-004: Perustaso

Kohdassa esitetyt asiat ovat hyviä. Haluaisimme korostaa, että ne ovat asioita, joita tulisi ratkaista arkkitehtuuri- tai muulla ylemmällä tasolla; yksittäinen sovellus vain toteutetaan näitä organisaation hyväksymiä periaatteita noudattaen.

s. 35 (ja muualla) sana hyökkäyspinta-ala.

Mielestämme tavallisempi ja kielellisesti parempi on hyökkäyspinta.

s. 35 SNT-006: Perustaso.

Lista alkaa lyhenteellä esim ja päättyy jne. Ainakin jälkimmäisen voi poistaa. Hyökkäyspinta voi olla myös sovelluksen sisällä. Tämä ei ole aivan tavallista, mutta asian tiedostaminen auttaa siihen varautumista.

s. 36 SNT-007: Perustaso

Kappaleessa on tietoturvavaatimusten kokoamisesta. Tämä tulisi siirtää aiemmaksi.

s. 36 SNT-008: Perustaso

Ideaalitapauksessa tietoturvamekanismit on määritelty organisaatiotasoisella arkkitehtuuri- tai muulla ylemmässä tasolla. Asia olisi hyvä mainita, jotta vältetään tilannetta, jossa yksittäinen sovelluskehityshanke tekee suuria arkkitehtuuriratkaisuja.

s. 36 SNT-009: Perustaso

Tunnistusmenetelmän määrittelyssä on syytä välttää liikaa yksityiskohtaisuutta tunnistusmekanismin osalta, mikäli asiasta ei ole ylemmän tason linjausta. Omistajan tulee valita tunnistuksen taso, mutta toteutustavassa on syytä pitäytyä yleiseen määrittelyyn.

Onko erityistä syytä, miksi esimerkkinä ei mainita KATSO- ja VIRTU–tunnistusta?

s. 37 SNT-014: Perustaso

Tietoturvassa tuskin mistään muusta yksittäisestä aiheesta on kirjoitettu enempää kuin salasanojen hyvyydestä. VAHTIn tulisi harkita, onko järkevää, että ohjeet antavat kukin hieman toisistaan poikkeavia suosituksia, vaikka ne tässä ohjeessa ovatkin esimerkkejä. Ohjeen sivulla 40 on viittaus VAHTI-ohjeeseen, jossa vaatimukset ovat hieman erilaiset.

Luku 3.3.5 Toteutus

Luvusta puuttuu olennaisia asioita:

- Ns. defensiivinen ohjelmointi, joka on erittäin tärkeää
- toteutukseen liittyen *code review* on syytä mainita
- toteutusta tukemaan on syytä olla olemassa turvallisen kehityksen ohjeistus ja referenssisovellus
- arkkitehtuuriratkaisujen tulisi toteuttaa mahdollisimman suurelta osin tietoturvatoiminnot, pienentäen näin sovellusten vastuulla olevien asioiden määrää

Joitain teknisiä ohjeita puuttuu. Näillä on merkitystä, koska asiat ovat käytännön sovelluskehitystyössä tärkeitä. Niihin liittyvät tietoturva-ongelmat ovat hyvin tavallisia:

- injektiohyökkäysten välttäminen (oikea enkoodaus (*encoding*) oikeassa kontekstissa, *variable binding* jne): olennaista tunnistaa sovelluksesta kaikki kontekstit, joissa injektio voi syntyä
- XSS-suojautuminen on syytä mainita erikseen, ongelman yleisyyden ja vakavuuden vuoksi
- *insecure direct object reference* -ongelman välttäminen
- tiedostojen käsittelyn vaarat (esim. tiedostopolku täytyy validoida huolellisesti, jos se on saatu käyttäjältä)
- em. teknisten asioiden osalta järkevintä olisi viitata olemassa olevaan ohjeistukseen, esim. OWASP, CERT, SANS, sillä asiat muuttuvat alati.

s. 41 TOT-002: Perustaso

Onnistuneiden syötteenkäsittelyjen lokittaminen saattaa tuottaa valtavan määrän tietoa lokiin. Onnistuneilla toiminnoilla on harvoin samaa merkitystä kuin epäonnistuneilla, ja lokitiedon määrän kasvu heikentää sen käytettävyyttä, ja ruuhka-aikoina saattaa heikentää järjestelmän käytettävyyttä. Tämä asia on syytä pitää mielessä, jos myös onnistuneet tiedonsyötöt halutaan kirjoittaa ylös.

s. 41 TOT-003: Perustaso (ja muualla)

Lokeista on kirjoitettu melko paljon, ja on harkittava, voitaisiinko viitata VAHTI-ohjeeseen.

s.42 TOT-006: Perustaso

“Sovellus tulee toteuttaa siten, että autentikoituneen käyttäjän istuntoa ei voi kaapata luvattomasti.” Näin luonnollisesti on. Haluamme taas korostaa arkkitehtuurin merkitystä ja jättää vähemmän yksittäisen sovelluskehittäjän harteille.

s.42 TOT-007: Perustaso

On kova vaatimus edellyttää kaikilta pieniltä sovelluksilta mahdollisuutta kytkeytyä ulkoiseen käyttäjähallintaan. Mahdollisuus on erittäin suositeltava, mutta onko todella niin, että edes perustasolla ei muita ratkaisuja voida ajatella? Vanhat suurkonejärjestelmät eivät tätä vaatimusta yleensä täytä.

s. 43 3.3.6 Testaus

Testaus on erittäin tärkeä vaihe, erityisesti suurissa, mutkikkaissa järjestelmissä. Ohjeessa esitettyjen asioiden lisäksi tulee testata toteutuneiden tietoturvatapausten (*incidents*) pohjalta. Ehdotamme lisäystä.

s. 44 TST-004: Perustaso

CERT ja muita esitettyjä lähteitä käytetään rutiininomaisesti auditoineissa. Haluamme kuitenkin korostaa, että yhtä tärkeää on näiden lähteiden hyödyntäminen ohjelmiston toteutustyön aikana.

Tietoturvaan liittyvien testien ja katselmointien käsittely on tehtävä kuten muidenkin

projektiin liittyvien auditointien ja katselmointien käsittely. Korjausehdotukset on käsiteltävä kuten muutkin muutokset ja laatupoikkeamat. Näkisimme tästä maininnan ohjeessa.

s. 44 TST-005: Korotettu taso

Automatisoitu testaus on tärkeää aivan kuten ohjeessa on. Kokemuksemme on, että parhaat tulokset saadaan yhdistämällä tämä “staattiseen” koodin arviointiin. Näin korotetun tason ohjelmistoissa. Sovelluskehitysvaiheessa porttiskanneri on harvoin yhtä käyttökelpoinen muut esitetyt menetelmät.

TST-006: Korotettu taso

Ehdotamme, että kohtaan lisätään arkkitehtuurin arviointi. Kyse ei ole kohtaan teknisestä asiasta, ja arkkitehtuurin hallinnan merkityksen takia (laki julkishallinnon tietohallinnosta) halaisimme nähdä asian omana kohtanaan.

S. 45 TOT-011:Korkea taso

“**Esim. kaikki tietyt** haavoittuvuudet tulee korjata koodista ennen kuin julkaisu voidaan tehdä.” on lihavoidulta kohdaltaan hieman epämääräinen ilmaus tärkeästä asiasta. Ehdotus uudesta “Voidaan edellyttää esimerkiksi, että kaikki haavoittuvuudet, joiden CVSS-arvo on yli 3, tulee olla korjattuna, ennen kuin julkaisu voidaan tehdä”.

3.3.7 Käyttöönotto ja ylläpito

Tuotantoympäristössä päivitysten seurannan ja asentamisen lisäksi pitäisi olla mekanismi, jolla voidaan reagoida nopeasti yllättäviin tietoturvatilanteisiin, estäen tunnistetun haavoittuvuuden hyväksikäytön, mikäli varsinaista korjausta ei ole käytettävissä tai sitä ei voida asentaa. Tällainen mekanismi on esimerkiksi sovelluspalomuuri, joka voi suodattaa HTTP-liikennettä konfiguroitavien sääntöjen mukaisesti.

3.3.8 Käytöstä poisto

Kappale on liian lyhyt suhteessa asian merkitykseen. Siinä on sisäänrakennettuna ajatus, että järjestelmän käyttö ja sen tarjoamat palvelut loppuvat. Tavallista kuitenkin on, että on tehty uusi järjestelmä tarjoamaan samoja palveluita, jolloin kyse on vaihdosta. Vanha saattaa jopa jäädä elämään jossain muodossa, ainakin, jotta on olemassa kohde varmuuskopiolta palautuksille.