Jari Pirhonen
AtBusiness Communications Oyj

Tietoturvallisia sovelluksia

Sovellusten tietoturvallisuus on noussut kuumaksi aiheeksi viime aikoina ja sovelluksilta halutaan enenevässä määrin myös tietoturvanäkökohtien huomioimista. Tämä on loistava trendi, koska käytännössähän sovellusprojekteja tehdään nimenomaan asiakaslähtöisesti ja tietoturvallisuus huomioidaan kunnolla vain, jos asiakas sitä vaatii. Toisaalta ei ole järkevää vaatia yleisesti ”tietoturvaa”. Tietoturva on subjektiivista, sovellusten käyttöympäristöt ja uhat ovat erilaisia, riskinsietokyky vaihtelee. Haluttu tietoturvataso täytyy määritellä.

Valitettavan usein nouseva tietoturvatietoisuus näkyy vain siinä muodossa, että asiakas ymmärtää kysyä tietoturvan perään ja sovellustoimittaja kertoo asian olevan kunnossa: ”Käytämme SSL-salausta ja palomuuria”. Tietoturva on kuitattu ja molemmat osapuolet ovat tyytyväisiä. Alku tämäkin, toivottavasti kuitenkin jatkossa tietoturvatarpeet määritellään sekä asiakas- että toimittajapuolella samalla tasolla kuin muutkin sovellusvaatimukset. SSL- ja palomuurisuojaukseen luottaessaan kannattaa muistaa, että palomuuria käytetään estämään pääsy muualle kuin käytettävään sovellukseen. SSL:n käyttö puolestaan vaatii myös sovellusta vastaan hyökkäävän käyttämään tietoliikenteen salausta – tämä ei yleensä ole ongelma. Yksinkertaistaen voisi sanoa, että SSL-salaus ja palomuuri takaavat sen, että vain meidän sovellustamme vastaan voidaan hyökätä ja vieläpä salattua yhteyttä käyttäen.

Median rummuttaessa tietoturvallisuuden ääri-ilmiöitä, kuten virukset, madot ja web-sivustojen hakkeroinnit, yritysten huomiokin helposti keskittyy sovelluskehityksen tietoturvallisuuden parantamisen kannalta epäoleellisiin asioihin. Tietoturvallisuuden kehittäminen on kuitenkin enimmäkseen riskien arviointia, koulutusta, ohjeistusta, käytäntöjen luomista, sopivien työkalujen käyttöönottoa ja dokumentointia.

Yrityksissä, joissa sovellusten tietoturvallisuuden edistämiseksi on jo tehty töitä, ollaan varmaankin kiinnostuneita SSE-CMM (Systems Security Engineering Capability Maturity Model) kehyksen tai Common Criterian käyttöönotosta. Kokemukseni perusteella monet yritykset ovat kuitenkin vasta alkutaipaleella sovellusten tietoturvallisuuden parantamisessa, joten siksi lähestyn asiaa perusteista lähtien. Tavoitteenani on herättää huomioimaan tietoturvallisuus sovelluksissa jo nyt, ennen kuin asiakkaat siihen pakottavat. Tietoturvallisuutta ei nimittäin saada mukaan hetkessä – tarvitaan aikaa toimintatapojen muutokseen sovellusprojektin kaikissa vaiheissa: määrittely, suunnittelu, toteutus, testaus ja käyttöönotto.

Sovelluskehityksen haasteita

Sovelluskehittäjien osa ei ole kadehdittava. Uusia protokollia, middleware-tuotteita ja työkaluja tulee jatkuvasti ja on täysi työ pysyä ajan tasalla uusien tekniikoiden osalta. Sitten vielä pitäisi huomioida tietoturva-asiat. Tämä on kuitenkin välttämätöntä toimivien, laadukkaiden sovellusten tekemiseksi.

Sovellusprojektin yleisimmät tavoitteet ovat toiminnallisuus, helppokäyttöisyys ja tehokkuus. Nämä mielletään usein ristiriitaisiksi tietoturvatavoitteiden kanssa: saatavuus, eheys, luottamuksellisuus, jäljitettävyys ja luotettavuus. Näin ei kuitenkaan tarvitse olla, kunhan kaikki tavoitteet huomioidaan yhdessä.

Sovelluskehityksen jatkuva kiire käyttöönotto- ja julkistusaikatauluista johtuen saattaa pakottaa tinkimään sovelluksen laadusta ja usein tinkiminen tapahtuu siellä, mitä huonoiten ymmärretään – tietoturvallisuudessa. Sovellusprojekteja suunnitellaan edelleen niin optimistisilla aikatauluilla, että monelta lienee jäänyt lukematta Frederick P. Brooksin klassikkoteos: ”The Mythical Man-Month”. Brooks mm. toteaa, että tuotteistetun ohjelmistokomponentin tekemiseen menee 9 kertaa enemmän aikaa kuin pelkän toiminnallisuuden ohjelmoimiseen ja Brooks ei edes tuonut esille tietoturvaa.

Sovellusprojekteilla ja tietoturvallisuudella on kyllä usein yksi yhteinen tavoite: yksinkertaisuus. Monimutkaisuus on tietoturvallisuuden pahin vihollinen. Valitettavasti sovellusympäristö muuttuu koko ajan monimutkaisemmaksi. Yritykset verkottuvat, sovelluksia halutaan käyttää aina ja kaikkialta erilaisten verkkoyhteyksien ja päätelaitteiden kautta, sovelluksia on avattava asiakkaille, kumppaneille ja mahdollisesti myös kilpailijoille. Haasteita riittää käyttöliittymä- ja sovellusarkkitehtuurisuunnittelijoille ohjelmoijista puhumattakaan. Samalla tietoturvahaasteet kasvavat. Ennen tietoturvallisuutta kuvattiin linnakemallilla – yrityksen sovellukset olivat suojassa yrityksen ”muurien” ja ”vallihautojen” takana. Nykyisin tietoturvallisuutta on ajateltava käyttäen mallina toria tai lentokenttää – huomio keskittyy käyttäjien hallintaan, todennukseen ja käyttöoikeuksiin.

Saatavuus on jo ajat sitten syrjäyttänyt luottamuksellisuuden useimpien yritysten tärkeimpänä tietoturvatavoitteena. Vastaavasti on esitetty näkemyksiä, joiden mukaan yritysten ensisijainen tavoite siirtyy tietojen ja sovellusten suojaamisesta selviytymiseen. Tärkeintä on, että palvelut toimivat, vaikka yksittäisiä palvelimia kohtaan hyökättäisiinkin tai verkkomato-epidemia kuormittaisi sähköpostia. Tämä vaatii turvallisten sovellusten lisäksi hyvää tietoturva-arkkitehtuuria, monistettuja järjestelmiä, tietoturvan monitorointia ja toipumissuunnitelmia.

Tietoturvallisuus sovelluskehityksessä

Parempien sovellusten aikaansaamiseksi on oleellista, että tietoturvatarpeet arvioidaan ja tavoitteet määritellään aivan kuten muutkin sovelluksen tavoitteet. Liian usein näkee määrittelydokumentteja, joissa tietoturvallisuus on ohitettu ylimalkaisilla huomautuksilla, kun taas muut sovelluskehittäjille tutummat asiat on kuvattu hyvinkin tarkasti. Tietoturvallisten sovellusten aikaansaaminen vaatii tietoturvallisuuden huomioimista koko sovelluskehitysprosessissa.

Sovelluskehityksen ammattilaisille on selvää, että virheiden korjaaminen on sitä edullisempaa, mitä aikaisemmassa vaiheessa ne huomataan. Kunnollinen määrittely ja suunnittelu säästävät kustannuksia sovelluksen koko elinkaaren aikana. Tietoturvaongelmat eivät tee poikkeusta tähän sääntöön – tietoturva-aukkojen korjaaminen jälkikäteen on kallista.

Yritys nimeltä @Stake teki vuonna 2002 mielenkiintoisen tutkimuksen 45 sovelluksen tietoturvaongelmista. Tulokset olivat hälyttäviä – tietoturvaongelmia löytyi paljon ja jopa 70% näistä oli suunnitteluvirheitä. Vakavista tietoturvaongelmista jopa puolet olisi huomattu huolellisemmalla suunnittelulla. Tyypillisimmät löydetyt ongelmat liittyivät käyttäjän todennuksen ja valtuutuksen virheellisyyksiin, riittämättömään syötteen tarkistamiseen ja istunnon hallinnan turvattomaan toteutukseen. *Kuva 1* listaa tutkimuksessa löydetyt 10 yleisintä tietoturvaongelmaa.

Sovelluksen tietoturvallisuutta ei siis saa jättää ohjelmoijan vastuulle. Toki ohjelmoijalla on oleellinen rooli, mutta ei ole ohjelmoijan tehtävä päättää tietoturvatoteutuksesta: miten käyttäjätodennus tehdään, miten istunto hallitaan, kuinka käyttäjävaltuudet hoidetaan, mitä salausalgoritmeja käytetään, kuinka sovelluksen saatavuus taataan, jne.

Arkkitehtuurisuunnittelulla on oleellinen merkitys. Erityisesti on sovitettava yhteen toisiinsa vahvasti vaikuttavat tietoturva-arkkitehtuuri ja sovellusarkkitehtuuri. Sen sijaan esim. J2EE ja .NET Framework ympäristöjen tietoturvaominaisuuksilla ei ole merkittäviä eroja.

Sovelluskehitysprosessi

Seuraavassa esitän muutamia ohjeita tietoturvallisuuden parantamiseksi. Käytän pohjana vanhan tutun vesiputousmallin vaiheita: määrittely, suunnittelu, toteutus, testaus ja käyttöönotto. Sinällään ohjeet eivät ole riippuvaisia käytettävästä sovelluskehitysmallista. Käytettiinpä vaikka spiraalimallia tai XP-metodia, niin perusohje on sama: ennen toteutusvaihetta on hyvä tietää tavoiteltava lopputulos ja suunnitella tehtävät lopputulokseen pääsemiseksi. Sovellus on syytä myös testata varmistaaksemme, että tavoite ja lopputulos ovat yhtenevät.

Määrittely on tietoturvallisuuden kannalta oleellisin vaihe, jolloin päätetään tietoturvatavoitteet ja –taso. Tietoturvallisuus pitäisi perustaa sovelluksen riskianalyysiin, jossa arvioidaan sovellukseen kohdistuvat riskit ja mahdollisten vahinkojen seuraukset sekä päätetään vastatoimet. Ilman riskianalyysiä tietoturvaratkaisut ovat vain arvauksia. Riskianalyysiin käytettävä työmäärä vaihtelee sovelluksen käyttötarkoituksen ja osallistujien kokemuksen mukaan.

Keskusteluun on tärkeää saada mukaan sovelluksen käsittelemän tiedon omistaja, jolla pitäisi olla paras käsitys tiedon arvosta ja suojausvaatimuksista. Yrityksen tietoturvallisuusvastaavat osaavat neuvoa yrityksen tietoturvaohjeistuksen ja –käytäntöjen asettamista reunaehdoista, kuten sallituista verkkoprotokollista, sovellusten sijoittelumahdollisuuksista ja asennuskäytännöistä.

Olen havainnut olevan hyödyllistä miettiä useampia tietoturvaltaan ja kustannuksiltaan eroavia vaihtoehtoja. Tällöin tietoturvataso on helpompi valita ja kaikilla on selkeämpi ymmärrys valitusta ratkaisusta ja siitä, miten tietoturvallisuutta olisi voitu parantaa lisäkustannuksin.

Suunnitteluvaiheessa lyödään lukkoon tekniset tavat määrittelyn sanelemien tietoturvaratkaisujen toteuttamiseksi. Tekninen suunnitelma on ohjelmoijan käsikirjoitus. Sovelluksessa voi olla sekä toiminnallisia tietoturvavirheitä että loogisia ongelmia. Toiminnalliset virheet voi usein laskea ohjelmoijan osaamisen piikkiin, mutta loogiset virheet tulisi karsia suunnitteluvaiheessa. *Kuva 2* esittää tietoturvaan tähtääviä suunnitteluperiaatteita.

Tietoturvatoimintojenkin suunnittelun avuksi on jo olemassa valmiita malleja (security patterns) ja tietysti kannattaa hyödyntää alan tunnettuja parhaita käytäntöjä.

Toteutus on raakaa ohjelmointityötä. Ohjelmoijalla ei pitäisi olla liikaa vapauksia tietoturvaratkaisujen suhteen, vaan hän toteuttaa määrittelyn ja teknisen suunnitelman vaatimukset. Ohjelmoijan vastuulle jää käyttämiensä ohjelmointikielien ja työkalujen tietoturvapiirteiden ja haasteiden ymmärtäminen.

Tietoturvaohjelmointia ei juurikaan ole huomioitu alan koulutuksessa. Ohjelmoijalla täytyy olla hyvinkin erilainen näkökulma sen mukaan koodaako hän sovelluksen toiminnallisuutta, miettikö mahdollisia virhetilanteita ja niistä selviytymistä vai kirjoittaako koodia, jolla toteutetaan tietoturvaratkaisuja. Käsitykseni mukaan tietoturvaa taitavat ohjelmoijat ovat vielä harvinaisuus. Eräät tyypilliset tietoturvaongelmiin johtavat virheet kuten puskuriylivuoto ja syötteen tarkastamatta jättäminen eivät mielestäni taasen ole muuta kuin huonoa koodia.

Perusongelmilta vältytään, jos ohjelmoijien käytössä on kokoelma testattuja, tietoturvanäkökohdat huomioivia valmiskomponentteja ja valmiiksi mietitty kehys tai malli sovelluskehitykseen. Tällöin harvat tietoturvaohjelmointiresurssit voidaan kohdistaa komponenttien tekemiseen ja varsinaisten sovellusohjelmoijien mahdollisuus virheisiin vähenee.

Testauksessa on toiminnallisuuden lisäksi huomioitava tietoturvan testaus. Tietoturvaongelmat johtuvat usein siitä, että sovellukseen on hiipinyt ylimääräistä toiminnallisuutta. Sovellusta voidaan siis käyttää väärin tavalla, jota sovelluksen toteuttajat eivät ole osanneet arvioida.

Ross Andersson ja Roger Needham ovat osuvasti kuvanneet testaamisen ongelmaa termeillä “Murphy’s computer” ja “Satan’s computer”. Usein testaamista tehdään olettaen, että käyttäjä on hyväntahtoinen ja virhetilanteet ovat lähinnä huolimattomuusvirheitä tai taitamattomuutta (Murphy). Tietoturvaa testattaessa pitäisi olettaa, että käyttäjä on pahantahtoinen, sovelluksen väärinkäyttöön kaikkia sääntöjä rikkoen pyrkivä (Satan).

Tarkistuslistoja voi käyttää yleisimpien ongelmien nopeaan löytämiseen, mutta ei kannata luottaa pelkästään niihin. Kuormitustestaukseen, asennusten turvallisuuden testaukseen ja yleisempien ohjelmointivirheiden löytämiseen on olemassa hyviä työkaluja. Järeämpänä keinona voi käyttää ulkopuolisella teetettävää sovelluksen auditointia. Mikään ei kuitenkaan

korvaa ammattitaitoista testaajaa. Tietoturvaa testaavan on oltava hyvin tekniikkaa ja ohjelmointia tunteva. Testaaja pyrkii rikkomaan sovelluksen suunnittelijan ja ohjelmoijan virheelliset oletukset, käyttää sovelluksen käyttöliittymien ulkopuolisia omia testiohjelmia ja hyökkää sovellusten ja komponenttien välisiin rajapintoihin.

Koodikatselmointi lienee yksi tehokkaimmista sovelluksen laatua ja tietoturvaa parantavista keinoista. On ihmeen motivoivaa tehdä huolellista jälkeä, jos tietää saavansa hutiloinnista armotonta kritiikkiä kollegoiltaan.

Käyttöönotossa on erityisesti huomioitava tietoturvalliset, kovennetut ja dokumentoidut asennukset. Monilta tietoturvaongelmilta olisi vältytty, jos käyttöjärjestelmistä, sovelluspalvelimista ja tietokannoista olisi asennusvaiheessa poistettu turha toiminnallisuus, esimerkkisovellukset ja oletusarvoiset käyttäjätunnukset ja salasanat.

Nykyisissä sovellusympäristöissä on niin paljon asennettavia komponentteja erilaisine asetuksineen, että usein pyrkimyksenä on vain saada sovellus toimimaan ja toimivaan ympäristöön ei haluta koskea. Tietoturvalliset asennukset edellyttävät suunnittelua ja hyviä asennusohjeita.

Tietoturvallinenkin sovellusympäristö rapistuu hoitamattomana. Ylläpidossa ja tietoturvakorjausten asentamisessa korostuu hyvän dokumentaation ja suunniteltujen käytäntöjen merkitys.

Yhteenveto

Pyrittäessä laadukkaampiin sovelluksiin on tietoturvallisuus otettava mukaan yhdeksi laatukriteeriksi. Tietoturvallisuus on kuitenkin määriteltävä ja suunniteltava sovellukseen alusta pitäen. Ensimmäisenä askeleena tietoturvallisuuden parantamiseksi sovelluskehityksessä voisi terävöittää sovelluksen tietoturvallisuuden määritystä ja testausta. Näin olisi annettu sovelluksen tietoturvatavoitteet ja lisäksi testausvaiheessa saataisiin ongelmat kiinni.

Tietoturvallisuuden kunnollinen huomioiminen sovelluskehityksessä vaatii vuosien työn organisaatiossa. Sovelluskehitysprosessia ei voi muuttaa kertaheitolla, vaan tässäkin eteneminen pienin askelin lienee se helpoin tie.

Otan mielelläni sähköpostitse vastaan palautetta esittämistäni näkökohdista. Samoin olen kiinnostunut kokemuksistanne ja menestystarinoistanne sovellusten tietoturvallisuuden parantamisessa. Tietoturvallisuuspiireissä saatuja kokemuksia jaetaan yleensä melko varovaisesti. Toivottavasti sovelluskehityksen ammattilaiset voivat vapaammin jakaa keskenään osaamistaan ja kokemuksiaan.

Kirjallisuutta

Ross Anderson: *Security Engineering*
Gary McGraw, John Viega: *Building Secure Software*
Howard, LeBlanc: *Writing Secure Code*
Graff, van Wyk: *Secure Coding: Principles & Practices*
Mark O'Neill et al.: *Web Services Security*
Jay Ramachandran: *Designing Security Architecture Solutions*
Bruce Schneier: *Secrets & Lies*
Whittaker, Thompson: *How to Break Software Security*

Aiheeseen liittyviä linkkejä http://www.iki.fi/japi/security.html#prog

Jari Pirhonen
Tietoturvallisuuspäällikkö ja –konsultti
AtBusiness Communications Oyj
japi@atbusiness.com
http://www.iki.fi/japi/

Kuva 1: Tyypillisiä tietoturvaongelmia

1. Käyttäjän istunnon kaappaus tai mahdollisuus uudelleen ajoon
2. Salasanakontrollien heikkous tai riittämättömyys
3. Puskuriylivuodot
4. Ylimääräiset, haavoittuvat tiedostot ja sovellukset
5. Heikko tai väärintoteutettu salaus tai satunnaislukujen luonti
6. Salasanojen kuuntelu
7. Evästeiden manipulointi
8. Ylläpitomekanismien heikkous tai väärinkäyttömahdollisuus
9. Lokien säilytys tai puute
10. Paljastavat virhekoodit

Lähde: http://www.atstake.com/research/reports/acrobat/atstake_app_unequal.pdf

Kuva 2: Tietoturvaperiaatteita

- Suojaa heikoin kohta ensin
- Rakenna useita puolustuslinjoja
- Turvaa virhetilanteet
- Anna sovellukselle vain välttämättömät oikeudet
- Erottele toiminnallisuudet eri tasoisiin turvatiloihin
- Yksinkertainen on kaunista – ja turvallisempaa (KISS)
- Huolehdi yksityisyyden suojasta
- Muista, että salaisuuksien pitäminen on vaikeaa
- Luota säästeliäästi
- Käytä testattuja välineitä ja komponentteja

Lähde: Gary McGraw, John Viega, *Building Secure Software*