Tietoturvallisuusnäkökohtien vaikutus muutostiedostojen hyväksymiseen avoimen lähdekoodin ohjelmistoprojekteissa

Oulun yliopisto
Tietojenkäsittelytieteiden laitos
Pro gradu -tutkielma
Teemu Ahlholm
27.05.2013

Tiivistelmä

Tässä tutkimuksessa analysoitiin tietoturvallisuusnäkökohtien vaikutusta avoimen lähdekoodin ohjelmistokehityksen kannalta olennaisiin muutostiedostoihin. Tutkimus kohdistui tietoturvaan ja sen huomioimiseen Bugzilla-virheidenraportointijärjestelmässä tapahtuvassa, avoimeen ohjelmistoarkkitehtuuriin perustuvan ja salatun tietoliikenteen mahdollistavan OpenSSH-tiedonsiirtoprotokollan muutostiedostojen hyväksymisprosessissa.

Aiheen valintaan vaikutti osaltaan myös aiheen potentiaalisuus avoimen lähdekoodin tietoturvan edistämiseksi ja mahdollinen muutostiedostojen hyväksymisprosessin jatkokehitys. Tutkimuksessa hyödynnettiin Bugzilla-järjestelmässä olleita, viralliseen OpenSSH-ohjelmistoversioon (release) hyväksyttyjä vuoden 2012 muutostiedostoja. Analyysi kohdistui muutostiedostojen hyväksymisprosessiin ja prosessiin liittyneisiin, käyttäjien luomiin tiketteihin ja dialogeihin, joiden sisältämä kommunikaatio käsitteli OpenSSH:n kehitystä ja ylläpitoa. Aineiston analysointi tapahtui sekä hakusanojen että sisällönanalyysin perusteella. Näissä analyyseissä huomioitiin kaikki tietoturvaan liittynyt sisältö.

Avoimen lähdekoodin ohjelmistojen muutostiedostojen hyväksymisprosessia on tutkittu aiemmin mutta ei tietoturvanäkökulmasta. OpenSSH-ohjelmistoon kohdistuvana tämä tutkimus eroaa lähestymistavaltaan aiemmasta tutkimuksesta. Tämä tutkimus kohdistui tietoturvallisuusnäkökohtien vaikutukseen muutostiedostojen hyväksymisprosessissa. Tutkimuksessa olennaista oli selvittää, millainen oli tietoturvallisuusnäkökohtien vaikutus muutostiedostojen hyväksymiseen avoimen lähdekoodin ohjelmistoprojekteissa ja miten tietoturvaan vaikuttaviin mahdollisiin löydöksiin reagoitiin ennen muutostiedoston hyväksymistä ja liittämistä ohjelmiston lopulliseen julkaisuun.

Tämän tutkimuksen kannalta tärkein tulos oli, että vaikka tutkimuksen kohteena olleen ohjelmistotuotteen lopulliseen julkaisuun hyväksytty muutostiedosto liittyi tietoturvaan, ei tietoturvaa huomioitu tai kyseenalaistettu Bugzilla-ympäristössä käydyssä muutostiedostojen kehitysprosessissa tai hyväksymisprosessissa. Bugzillassa tapahtuvan OpenSSH-ohjelmistotuotteen lopullisiin ohjelmistoversioihin liitettyjen muutostiedostojen hyväksymisprosessi ei huomioinut, ottanut kantaa tai kyseenalaistanut muutostiedostojen tietoturvaa.

Koska näiden tikettien mahdollisiin epäkohtiin olisi puututtu usein viimeistään pääkäyttäjän toimesta, todettiin niiden olevan tutkimuksen kannalta analysoitavissa. Ohjelmistokehitystä tapahtuu myös ohjelmiston kehittäjien suljetulla postituslistalla, joka on tarkoitettu ainoastaan OpenSSH-ohjelmiston tietoturvaan liittyvien asioiden kehittämiseen ja toteuttamiseen.

OpenSSH, SSH, avoin lähdekoodi, open source, hyväksymisprosessi, OSS, tietoturva, haavoittuvuus, buffer overflow, memory leak, heap buffer overflow, stack buffer overflow, DoS, Denial of Service, code injection, luottamuksellisuus, confidentiality, eheys, integrity, saatavuus, availability, pääsynvalvonta, access control, osapuolten todentaminen, verification, tunnistaminen, identification, Bugzilla, tiketti.

Esipuhe

Pro gradu -tutkielmani aiheen valintaan vaikutti mielenkiintoni avoimen lähdekoodiin, avoimen lähdekoodin ohjelmistoihin ja tietoturvaan. Tästä syntyi ajatus ja halu toteuttaa tutkimus, joka kohdistuu useisiin avoimen lähdekoodin tietojärjestelmiin, joiden yhdistävänä tekijänä on tietoturva. Teoria-aineisto perustui ensisijaisesti tieteellisesti vertaisarvioituihin tutkimuksiin, artikkeleihin ja alan julkaisuihin. Tutkimus kohdistui OpenSSH-tiedonsiirtoprotokollan Bugzilla-järjestelmässä muutostiedostojen hyväksymisprosessiin ja erityisesti tietoturvan huomioimiseen hyväksymisprosessissa.

Miltei kaksivuotisen projektin lopputulos on valmis. Tästä haluan kiittää useita tahoja.

Kiitokset opinnäytetyöni ansiokkaasta ohjauksesta ohjaajalleni, Oulun yliopiston tietojenkäsittelytieteen laitoksen yliopisto-opettaja Henrik Hedbergille, tutkielmaani opponoineelle lehtori Ari Vesaselle sekä dosentti Raija Haloselle asiantuntevasta eteenpäin auttaneesta tienviitoituksesta ja ohjauksesta. Kannustuksen sanoista haluan kiittää opiskelijakollegoitani, ystäviäni ja aivan erityisesti perhettäni, tytärtäni Ainoa ja poikaani Oulaa.

Oulussa 27.5.2013

_____

Teemu Ahlholm

Sisällys

1. Johdanto

Tietoturvan ja tietoturvallisten ohjelmistotuotteiden merkitys korostuu tietotekniikan kehittyessä. Avoimen lähdekoodin ohjelmistotuotteet muodostavat vaihtoehdon kaupallisille ohjelmistotuotteille. Ohjelmiin ja ohjelmistoihin kohdistuvat tarpeet ja vaatimukset luovat paineita niin avoimen lähdekoodin ohjelmistojen kuin suljetun lähdekoodin ohjelmistojen ohjelmistokehitykselle.

Tässä opinnäytetyössä analysoitiin tietoturvallisuusnäkökohtien vaikutusta avoimen lähdekoodin ohjelmistokehityksen kannalta olennaisiin muutostiedostoihin. Tutkimus kohdistui alkujaan OpenBSD-kehitystiimin toteuttamaan, salatun tiedonsiirron mahdollistavaan OpenSSH-tietoliikenneprotokollaan. OpenSSH on ohjelmisto, joka kehitettiin vuonna 1999 ei-kaupalliseksi vaihtoehdoksi alun perin suomalaisen Tatu Ylösen vuonna 1995 kehittämälle, nykyään kaupalliselle SSH -tietoliikenneohjelmistolle (Bauer, 2001a). OpenSSH-ohjelmisto julkaistiin ensimmäisen kerran OpenBSD 2.6 -käyttöjärjestelmäjulkaisun mukana. SSH puolestaan kehitettiin aikoinaan salaamatonta tiedonsiirtoa käyttävän RSH (Remote SHell) -protokollan korvaajaksi. Tämä tutkimus kohdistui OpenSSH-ohjelmiston (OpenBSD Secure Shell) kehittymisen kannalta olennaisiin muutostiedostoihin, niiden hyväksymisprosessiin kehitys- ja virheidenraportointiympäristönä toimivassa Bugzilla-työkalussa, johon kirjatuissa tiketeissä ja raporteissa käsiteltiin OpenSSH:n kehitystä ja ylläpitoa.

Ohjelmien toiminnalliset ja laadulliset vaatimukset muodostavat haasteita, joihin tietokoneohjelmistojen kehityksen ja myös jatkokehityksen tulisi pystyä vastaamaan mahdollisimman tehokkaasti. Avoimen lähdekoodin ohjelmistojen kehitysympäristöjä ja niiden toiminnan tietoturvaa tulisi voida kehittää entisestään vastaamaan ajanmukaisia tarpeita. Potentiaalisiksi todetut ja havaitut muutostiedostot päätyvät usein osaksi lopullista ohjelmistotuotetta, joten näiden muutostiedostojen tietoturva ja muutostiedostojen kehitysprosessi tulisi tietoturvatuotteissa huomioida vielä vallitsevaa käytäntöä kokonaisvaltaisemmin.

Tämä tutkimus erosi aikaisemmista tutkimuksista lähestymistavaltaan: tässä tutkielmassa tutkittiin ja analysoitiin tietoturvan vaikutusta muutostiedoston hyväksymisprosessiin avoimen lähdekoodin ohjelmistokehityksessä. Olemassa oleva, tieteellisesti vertaisarvioitu tutkimus evidensseineen oli tärkeää taustamateriaalia asioiden ymmärtämisen kannalta, ja siten auttoi muodostamaan näkemyksen siitä, millaista tutkimusta aiheesta, avoimen lähdekoodin muutostiedostojen hyväksymisprosessista oli aiemmin tehty. Tutkimuksen kannalta olennaisimmat käsitteet määriteltiin ja muodostettiin aiemmin tutkitun aineiston pohjalta. Tutkimuksen kannalta olennaisessa roolissa oli erityisesti tutkimuskysymys:

Millainen on tietoturvallisuusnäkökohtien vaikutus muutostiedostojen hyväksymiseen avoimen lähdekoodin ohjelmistoprojektissa?

Vastaavaa tutkimusta ei aiemmin ole tehty OpenSSH-ohjelmistoon ja sen kehitykseen liittyen.

Tutkimuksen fokuksena oli OpenSSH-ohjelmisto ja erityisesti ohjelmistotuotetta kehittävien tahojen näkemykset avoimeen lähdekoodiin perustuvan OpenSSH-ohjelmiston tietoturvasta tietoturvaan liittyviä hakusanoja ja sisällönanalyysia apuna käyttäen. Tutkimus kohdistui Bugzillaan eli virheiden kirjausjärjestelmään kirjattuihin, OpenSSH-ohjelmistoa koskeviin raportteihin eli tiketteihin. Bugzilla on yleisesti käytössä oleva avoimeen lähdekoodiin perustuva avoin ja yleiskäyttöinen tietokoneohjelmien ja ohjelmistojen virheiden raportointijärjestelmä. Bugzillaa käyttäviä tahoja ovat mm. NASA, KDE, GNOME, Apache, Red Hat, Novell ja Wikimedia Foundation. Bugzilla-järjestelmän toiminta perustuu siihen, että jokainen, joka haluaa osallistua tuotteen kehitykseen, voi raportoida virheiden raportointijärjestelmään havaintojaan, kehityspyyntöjä, virheitä tai muuta ohjelmiston kannalta olennaista tietosisältöä. Ohjelmaa käytetään sekä avoimen että suljetun lähdekoodin kehitystyökaluna ja ensisijaisena virheiden raportointityökaluna. Tässä tutkimuksessa jokaisesta järjestelmään syötetystä ilmoituksesta käytettiin termiä tiketti.

Raportointijärjestelmään kirjatut, kehitettävää tai muutettavaa tuotetta koskevat virheet, havainnot, huomiot, kehitysehdotukset tai kommentit voidaan raportoinnin jälkeen osoittaa tehtäväksi tai työpyynnöksi tietylle tai tietyille kehittäjille. Jokaiselle virheelle, havainnolle tai kehitysehdotukselle voidaan tapauksen luonteesta riippuen tehdä useita tilapäivityksiä (status), jonka lisäksi virheen tietoihin voidaan liittää ohjelmakoodia, käyttäjien huomautuksia sekä esimerkkejä havainnoista tai tilanteista, joissa virhe tai kommentoitava ominaisuus tai havainto näkyy. Osa järjestelmään syötetyistä käyttäjien kirjaamista kuvauksista eli tiketeistä suljetaan tarpeettomina, osa puolestaan iteroituu järjestelmän sisällä saaden elinkaarensa aikana mahdollisesti useita toistuviakin tilapäivityksiä. Osa muutostiedostoehdotuksista ei tule päätymään lopulliseen ohjelmistoon (Bugzilla, 2012.)

Avoimen lähdekoodin tietoturvaa voidaan tulkita ja tutkia monin eri tavoin ja monilla eri tasoilla. Aiheen laajuuden vuoksi tämä tutkimus rajattiin koskemaan ainoastaan OpenSSH-ohjelmistoa ja ko. ohjelmiston lopulliseen ohjelmakoodiin päätyneitä muutostiedostoja. Avoimen lähdekoodin syvällinen analyysi kooditasolla jätettiin huomioimatta valittujen tutkimusmenetelmien vuoksi. Teoriaosuus johdettiin tieteellisestä lähdeaineistosta. Tutkimusaineisto rajattiin koskemaan vuotta 2012, joten tutkimuksen kannalta olennaista tietoa olivat kaikki vuonna 2012 lopulliseen koodiin asti päätyneet muutostiedostot. Olennaista oli myös kaikki OpenSSH-ohjelmiston muutostiedostoihin liittyvä, Bugzillassa vuonna 2012 käyty keskustelu.

Tietoturvan arvioinnilla tarkoitetaan yleensä ohjelmiston tietoturvaan vaikuttavien ominaisuuksien järjestelmällistä ja tasapuolista mittaamista ja analysointia. Tämä tutkimus voi osaltaan tarjota jatkokehitysideoita siihen, kuinka OpenSSH:n kaltaisen ohjelmistotuotteen tietoturva tulisi huomioida mahdollisissa Bugzillan tulevissa versioissa. OpenSSH-tuotteelle olemassa olevien sähköpostilistojen lisäksi tulisi olla täysin tietoturvaan keskittynyt virheidenraportointityökalu, jossa analysoitaisiin, käsiteltäisiin ja prosessoitaisiin ainoastaan OpenSSH-tuotteen tietoturvaa ohjelmistotuotteen mahdollisen tietoturvahyödyn maksimoimiseksi. Ainoastaan kehittäjille tarkoitettu, täysin ulkopuolisilta suljettu sähköpostilista ei välttämättä ole riittävän kattava ja kokonaisvaltainen ympäristö OpenSSH:n kaltaisen tietoturvatuotteen kehittämiseksi. Löydetyt tietoturvatapaukset olivat toisistaan täysin poikkeavia, eikä niillä ollut vaikutusta muutostiedoston hyväksymiseen, joten niiden ainoa yhtenevä tutkintalinja oli se, että ne liittyivät Bugzilla-ympäristössä oleviin lopulliseen julkaisuun (release) liitettyihin muutostiedostoihin, ne olivat vuodelta 2012 ja niissä käsiteltiin OpenSSH:n tietoturvaa.

Luvussa 2 käsitellään avoimen lähdekoodin käytännöt, muutostiedostojen hyväksymisprosessi, ja yleisellä tasossa Bugzillan toiminta. Luvussa 3 käsitellään tietoturva kokonaisuutena ja yleisesti, tietoturvan peruskäsitteet, tietoturvaan liittyvät hyökkäykset ja tietoturvan rooli avoimessa lähdekoodissa. Luvussa 4 käsitellään ja kuvataan tutkimuksessa käytetyt tutkimusmenetelmät ja tutkimusympäristö. Luvussa 5 esitellään analysoitava aineisto, tutkimuksessa käytetyt analyysimenetelmät, kuvaus analyysistä, löydökset ja yhteenveto analyysistä. Luvussa 6 käsitellään pohdinta. Luvussa 7 on tukielman yhteenveto.

2. Avoimen lähdekoodin käytännöt

2.1 Avoin lähdekoodi

Avoimesta lähdekoodista voitiin Unixin kohdalla puhua jo 1970-luvulla, koska kyseisen käyttöjärjestelmän lähdekoodi oli jokaisen saatavilla (Russell, 2004). Standardoinnin ja kaupallisten toimenpiteiden seurauksena koodista tuli kauppatavaraa, eikä se ollut enää avoin, vapaasti jaettava (Hansen, 1973). Avoimen lähdekoodin rinnalle muodostui suljettu lähdekoodi, joka oli lisensoitua, salaista, eikä käyttäjällä ole mahdollisuutta saada siihen omistusoikeutta (Arakji & Lang, 2007). Tämän päivän ohjelmistot ovat edelleenkin kalliita rakennettavia ja haastavia ylläpidettäviä, olipa kyseessä avoimeen tai suljettuun perustuva tuote (Raja & Barry, 2005). Viime vuosina onkin havaittu kasvua uudelleen käytettävän lähdekoodin ja ns. suoraan hyllyltä (off-the-shelf) -tyyppisten ohjelmistojen suhteen. Useimmissa tapauksissa tällaisten ohjelmistotuotteiden lähdekoodi ei ole käyttäjien analysoitavissa. Jos ohjelmistotuote ei toimi halutulla tavalla, ovat ohjelmistotuotteeseen liittyvän tietohallinnon johto ja ohjelmistoprojektien vetäjät lähdekoodin omistajan armoilla, voidaanko tai halutaanko koodimuutoksia tehdä vai ei. (Rakic & Medvidovic 2001; Dashofy, Medvidovic & Taylor 1999.)

Raja ja Barry (2005) mainitsevatkin yhdysvaltalaisen tietojenkäsittelijän Frederick Phillips ”Fred” Brooksin kirjoittaneen yli 30 vuotta sitten ihmelääkkeen tarpeellisuudesta auttamaan ohjelmistosuunnittelijoita ja tietoteollisuusalan ammattilaisia nopeuttamaan ja ylläpitämään ohjelmiin tai ohjelmistoihin pohjautuvia tietojärjestelmiä. Avoimen lähdekoodin ohjelmistojen liike (OSS, Open-source software) on tutkimuksen (Raja & Barry, 2005) mukaan muuttanut tapaa kehittää, ylläpitää ja päivittää ohjelmistoja. Avoimen lähdekoodin ohjelmiin ja ohjelmistoihin pohjautuvat tietojärjestelmät ovat yleensä kehitetty vapaasti käytettäviksi ja levitettäviksi ohjelmiksi ja ohjelmistoiksi, ja siten ne ovat saatavilla lähes ilmaiseksi tai ilman kustannuksia (Schryen, 2011). Tapauksesta riippuen avoin lähdekoodi ei kuitenkaan aina ole ilmaista eikä suljettu lähdekoodi ole aina kaupallista tai maksullista (Schryen & Kadura, 2009). Viime aikoina avoimen lähdekoodin ohjelmistoja on kuitenkin yhä useammin käytetty ohjelmistojärjestelmissä. Syy avoimen lähdekoodin ohjelmistojen käyttämiseen ei ole ainoastaan matala ja kustannustehokas kustannustaso tai koodin helppo käytettävyys, vaan jatkuvasti kehittyvä ohjelmistojen laatu (Raja & Barry, 2005.)

Eräs tuoreimmista ohjelmistojen laatua tutkivista tutkimuksista (Raja & Barry, 2005) perustuu empiiriseen tutkimustietoon, jota on saatu kerättyä ohjelmistojärjestelmien kehittämistyössä perinteisin analytiikan ja tiedonhankinnan menetelmin. Eräs yleisimpiä tapoja mitata ohjelmistojen laatua on laskea määrällisesti järjestelmässä ilmenevien vikojen määrää, ja tätä tarkoitusta varten he ovat kehittäneet avoimen lähdekoodin laatua mittaavan mallin, ja siten empiirisesti myös kokeilleet sitä käytännössä. Tällaisia vikoja ovat mm. ohjelman kaatuminen epänormaalilla tavalla ja toisaalta käyttäjän mielestä ohjelman epänormaali käyttäytyminen. Avoin lähdekoodi muuttaakin hieman tätä asetelmaa, ja ohjelmistojen suhteen on tarkasteltava uudelleen niitä mittareita, joita

perinteisesti ja yleisesti on käytetty arvioitaessa ohjelmistojen laatua (Raja & Barry, 2005.)

Tutkimuksessaan Raja ja Barry (2005) toteavatkin avoimen lähdekoodin ohjelmistojen herättäneen viime vuosina runsaasti mielenkiintoa. Tutkimuksessaan he väittävätkin useiden merkittävien avoimeen lähdekoodiin pohjautuvien ohjelmistotuotteiden ohittaneen vastaavat kaupalliset tuotteet sekä markkinaosuudessa että arvioitaessa niiden laatua. Avoimen lähdekoodin tuotteet eivät ainoastaan houkuttele yksittäisiä käyttäjiä, joilla ei ole varaa kalliisiin kaupallisiin versioihin, vaan niistä on tullut varteenotettava vaihtoehto ja ehdokas monien suurten yritysten ja hallitusten ohjelmistojen hankintaprosessissa (Gurbani, Garvert & Herbsleb 2006; Raja & Barry; 2005.) Avoin lähdekoodi voikin olla yksittäiselle kotikäyttäjälle ilmainen, mutta yrityskäyttöön tarkoitettu ohjelmakoodi voi olla maksullista (Russell 2004; Schryen & Kadura 2009), mutta toisaalta avoimen koodin hyödyntämisen ei kuitenkaan tulisi olla mielekästä ainoastaan sen edullisuuden vuoksi (Schryen, 2009). Kansainvälisesti tunnetun ICT-alan julkaisun, CIO Magazinen[1] vuonna 2002 tekemä tutkimus paljasti, että tietotekniikkayhteisö muuttuu ja muovautuu rakenteeltaan ja toimintatavoiltaan mukavammaksi avoimen lähdekoodin kehitysmallin myötä, ja että enemmistö, 64 prosenttia tutkimuksen kohteena olevista yhtiöistä käytti avoimen lähdekoodin ohjelmistoja lähinnä web-palvelimena, palvelimen käyttöjärjestelmänä tai web-kehitysympäristönä (Raja & Barry, 2005).

Tällaiseen voi vaikuttaa myös avoimen lähdekoodin tuotteiden hyväksymiskynnys, joka tutkimuksen (Zhou & Davis, 2005) mukaan on erittäin merkittävä erityisesti palvelinten ja käyttöjärjestelmien suhteen. Kyseisellä toiminta-alueella kaksi avoimen lähdekoodin hallitsevaa ja korkealaatuista tuotetta, Apache ja Linux ovat jo tuotemerkkeinä todistaneet tehokkuutensa (Crowston, Wei, Howison & Wiggins, 2008). Näistä Linux on eräs menestyksekkäimpiä avoimeen lähdekoodiin pohjautuvia projekteja, ja se on käytössä monissa kaupallisissa sovelluksissa (Thomas 2006; Weiss 2008). Toisaalta on kuitenkin vielä muutosvastarinnasta kärsiviä aloja, joilla työskentelevät ihmiset ovat haluttomia siirtymään avoimeen lähdekoodiin perustuvien tuotteiden käyttämiseen (Moreno 2006; Weber 2004).

Forresterin julkaisemien tutkimustulosten mukaan useimmilla eurooppalaisilla yrityksillä on selkeitä suunnitelmia avoimeen lähdekoodiin siirtymiseksi (Zhou & Davis, 2005). Tästä huolimatta erityisesti yritysten liiketoimintaan ja projektien hallinnoimiseen liittyvien ihmisten ja työntekijöiden keskuudessa on edelleen runsaasti pelkoja ja ratkaisemattomia kysymyksiä. Kaikki nämä pelot ja huolenaiheet voidaan jäljittää avoimen lähdekoodin tuotteiden laatuun ja luotettavuuteen (Raja & Barry, 2005). Tähän voi myös vaikuttaa osaltaan se, että avoimen lähdekoodin projektien yhteydessä kehittäjät eivät useinkaan ole työsuhteen sitomia, ja siten henkilöstön vaihtuvuus on suurempaa verrattuna suljetun lähdekoodin ohjelmistokehitykseen (Stewart, Darcy & Daniel, 2005).

Tutkimuksen mukaan (Zhou & Davis, 2005) tällaiseen käyttäytymiseen voi osaltaan vaikuttaa kaksi yleisintä pelkoa, eli yleinen tuen riittämättömyys (lack of formal support) ja muutosten hitaus, jotka myös Oraclen aiempi toimitusjohtaja Ray Lane on maininnut avoimen lähdekoodin konferenssissa 2004. Tutkimukseen mukaan tutkimukseen osallistui vain kahdeksan suosittua, kehitysvaiheessa olevaa avoimen lähdekoodin hanketta. Vaikka tämän vuoksi tuloksista oli mahdollista tehdä ainoastaan

1 http://www.cio.com/magazine

osittaisia johtopäätöksiä, oli tuloksista kuitenkin mahdollista saada havaintoja (Zhou & Davis, 2005.)

Avoimen lähdekoodin tunnuspiirteet on täsmällisesti määritellyt Open Source Initiative (2013), joka on Bruce Perensin ja Eric S. Raymondin vuonna 1998 perustama järjestö (Goldman & Gabriel, 2005). Avoimesta lähdekoodista tai avointa lähdekoodia edustavasta ideologiasta tai jopa filosofiasta ei voida puhua pelkästään mahdollisuutena päästä tutustumaan itse ohjelman tai ohjelmiston lähdekoodiin. Avoimen lähdekoodin edustaminen on ydinajatukseltaan myös juridinen kysymys (Goldman & Gabriel, 2005). Tämä tarkoittaa koko teoksen kattavaa lisenssiä, jonka mukaisesti tekijän on voitava tarjota teostaan, tässä tapauksessa ohjelmakoodia, ohjelmaa tai ohjelmistoa muutoksineen esim. yhteisön tutkittavaksi, käytettäväksi tai muokattavaksi (Ducheneaut, 2005). Ohjelman tai ohjelmiston ohjelmakoodin luojilla on toisin sanoen lupa ja mahdollisuus säilyttää oikeus koodiinsa, mikäli he eivät erikseen luovu niistä. OSI:n ydinajatus ja tavoite on edistää avoimeen lähdekoodiin perustuvien ohjelmien ja ohjelmistojen käyttöä (Open Source Initiative, 2013).

Avoimeen lähdekoodin sisältyy tyypillisesti myös lisenssi, jonka taustalla oleviin periaatteisiin kuuluu oikeus käyttää ohjelmaa, ohjelmistoa tai ohjelmakoodia haluamallaan tavalla. Lisensointi mahdollistaa sekä alkuperäisen että editoidun koodiversion kopioimisen että levittämisen. Käytännössä vapaat ohjelmistot ja avoin lähdekoodi ovat käsitteinä lähes identtiset, ja vapaista ohjelmistoista puhuttaessa ohjelmistokehityksen eettiset ja moraaliset kysymykset ovatkin olennaisessa roolissa (Russell, 2004.)

Avoimen lähdekoodin kannalta olennainen ydinajatus kiinnittyy Bretthauerin (2002) mukaan usein vapauden ja avoimuuden ideologiaan ja filosofiaan. Internetin suomenkielinen ja vapaa Linux-käyttöön kehitetty ohjesivusto (Linux.fi, 2011) väittää ja luonnehtii avoimeksi lähdekoodiksi määritellyn koodin, ohjelman tai ohjelmistojen tutkimisen, käyttämisen, kopioimisen ja muokkaamisen jokaiselle avoimena mahdollisuutena, josta seuraa myös käytännön hyötyjä. Avoimen lähdekoodin ohjelmia ja ohjelmistoja voidaankin luonnehtia vallankumouksellisiksi ohjelmistojen kehityksen ja jakelun uudeksi malliksi, joka pohjautuu ideologiaan ohjelmistoista, jotka ovat joko vapaita (free) tai avoimia (open) (Bretthauer, 2002).

Avoimen lähdekoodin ohjelmistot ovat toisaalta myös ristiriidassa ohjelmistojen perinteisen jakelumallin kanssa, jossa ohjelmistotuotteen ostajalle myydään ainoastaan käyttöoikeus valmiiksi käännettyyn ohjelmaan tai ohjelmistoon, ilman käyttöoikeutta kyseessä olevan ohjelman tai ohjelmiston lähdekoodiin (Bretthauer, 2002). Toisaalta, tutkimuksessaan Bretthauer (2002) väittää, että avoimen lähdekoodin ohjelmistojen lisenssit takaavat vapaan pääsyn lähdekoodiin, jolloin ohjelmointitaitoiset käyttäjät voivat itsenäisesti korjata ohjelmakoodissa olevia virheitä tai tehdä koodimuutoksia omien näkemystensä pohjalta. Suljetun lähdekoodin ohjelman ja ohjelmakoodin muokkaaminen on käytännössä mahdotonta, koska muokattavaa lähdekoodia ei ole saatavilla. Avoimen lähdekoodin ohjelmistojen kehitystyössä komponenttien valinta tapahtuu aina tilanteissa, joissa tällaisille tilanteille ominaiset rajoitukset ovat huomattavasti tärkeämmässä roolissa kuin ne kriteerit, joita on ehdotettu yleisissä arviointimalleissa. Avoimen lähdekoodin ohjelmistojen kehittäjät myös käyttävät periaatetta ”first fit” -periaatteen ”best fit” -periaatteen sijaan, jälkimmäisen ollessa useimpien normatiiviseen valintaan pohjautuvien menetelmien suositus. (Hauge, Osterlie, Sorensen & Gerea, 2009.)

Avoimen lähdekoodin ohjelmistojen ja suljetun lähdekoodin ohjelmistojen tietoturvakeskustelut ovat pääsääntöisesti luonteeltaan uskomuksia ja arvauksia, väittää Schryen (2011) tutkimuksensa johtopäätöksessä. Tietoon pohjautuvat oivallukset tuovat uutta tietoa tietoturvakysymysten kaltaisiin kysymyksiin. Tutkimuksessa tuodaan esille se tieto, ettei ole muodostunut tai olemassa mitään empiiristä todistetta siitä, että ohjelmistokehityksen jokin tietty osa-alue tai erityinen tyyppi veisi tietoturvan kehittymistä eteenpäin. Tärkeintä on tarjota taloudellisia kannustimia ohjelmien tai ohjelmistojen tuottajille, jotta ohjelmistoista tehtäisiin haavoittumattomampia. Olennaista myös olisi, että tarvittavien tietoturva-aukkoja korjaavien paikkojen julkistamisesta muodostuisi ja tulisi olemaan pysyvä käytäntö (Schryen, 2009.)

Avoimen lähdekoodin ohjelmistojen kehittäjät ovat arvioineet yksittäisiä avoimen lähdekoodin komponentteja peräkkäin, sen sijaan että ensin tunnistaisivat joukon komponentteja ennen niiden arvioimista (Hauge et al., 2009). Avoimen lähdekoodin ohjelmistojen (OSS) toimittajat voisivat kerätä ja koostaa komponenttien kannalta hyödyllistä tietoa. Tällaista tietoa olisi mm. luettelo komponentin ominaisuuksista, suunnitelma komponentin hyödyntämisestä tulevaisuudessa, luettelo tunnetuista ongelmista ja riippuvuuksista, dokumentaation luettelo, määrittelyt käyttöönoton ohjauksesta, julkaisusykleistä, sekä muusta tarpeellisesta tiedosta, joka olisi helposti saatavilla, ja joka auttaisi avoimen lähdekoodin ohjelmien tai ohjelmistojen kehittäjiä ohjelmiston kannalta tarpeellisten osien valitsemisessa. Tällainen koodin ja sen ympärille rakentunut avoimuus mahdollistaa ohjelmakoodin muokkaamisen ja tutkimisen, jolloin ohjelman tai ohjelmiston toiminnan tarvittaessa kokonaisvaltainen tutkiminen on mahdollista. Näin voidaan varmistua siitä, ettei esim. ohjelmaan lisätty ominaisuus seuraa käyttäjän toimia (Raja & Barry, 2005.)

Toiminta avoimen lähdekoodin ympärillä kiihtyi dramaattisesti 1990-luvun alussa Internetin yleistymisen myötä, ja nykyisissä muodossaan se on merkittävä vaikuttaja koskien ohjelmistoteollisuutta ja koko yhteiskuntaa (Arakji & Lang, 2007). Muutamien avoimeen lähdekoodiin perustuvien ohjelmien ja ohjelmistojen, kuten Linuxin, Apachen tai Mozilla Firefoxin laajaa käyttäjäkuntaa ovat Rajan & Barryn 2005 ja Crowstonin et al. 2008 mukaan yksityisten käyttäjien lisäksi julkishallinnolliset tahot kuten kaupalliset yritykset ja valtion laitokset. Artikkelissaan Aaron Weiss (Weiss, 2008, s. 40) puhuukin määrällisesti sadoista erilaisista Linux-jakelupaketeista, joiden käyttötarkoitus vaihtelee, ja joista jokaisella ohjelmistotuotteen kehitykseen ja toteutukseen osallistuvalla taholla on käyttäjäsegmenteistään oma näkemyksensä.

Linux-käyttöjärjestelmästä on muodostunut yksi menestyksekkäimmistä avoimen lähdekoodin hankkeista, jonka päälle on rakennettu lukuisia kaupallisia sovelluksia (Raja & Barry, 2005). Tutkimuksen pohjalta on myös (Alhazmi, Malaiya & Ray, 2007) voitu osoittaa, että usein avoimen lähdekoodin ohjelmistokehitys myös poikkeaa perinteisestä ohjelmistokehityksestä. Xu ja Jones (Xu & Jones, 2010) puolestaan ovat tutkimuksessaan todenneet, että avoimen lähdekoodin projektiyhteisöä voidaan kuvailla rakenteeltaan yhteiskuntajärjestelmäksi, jossa projektiyhteisön sosiaalinen pääoma (social capital) on tärkeä voimavara avoimen lähdekoodin projektin onnistumiselle.

Tietokoneohjelmien ja -ohjelmistojen tuottamisprosessia on (Työ- ja elinkeinoministeriö, 2011) perinteisesti koordinoitu yritysten toimesta. Tällaiseen käytäntöön sisältyvät tyypillisesti suljettu lähdekoodi ja yrityksen ydinliiketoimintaan tai julkishallintoon keskeisesti liittyvät, tarkasti valvotut liikesalaisuudet ja uudet innovaatiot. Avointa lähdekoodia sitä vastoin pidetään taloudellisesti tehokkaana mallina uusien teknisten innovaatioiden kaupallistamiselle. Siirtyminen avoimeen tai suljettuun lähdekoodiin on syytä arvioida ohjelmien tai ohjelmistojen

käyttötarkoituksen perusteella. Ohjelmistojen käyttämisestä syntynyt käyttökokemus on todistanut, että koodin avoimuuteen ja muokattavuuteen on haluttu vaikuttaa (Arakji & Lang, 2007.)

Avoimen ja suljetun lähdekoodin empiirisesti vertailevassa tutkimuksessa (Schryen, 2011) on verrattu 17 tunnettua ja laajasti käytössä olevaa ohjelmistotuotetta niiden julkaistujen haavoittuvuuksien ja ohjelmien jälleenmyyjien julkaisemien muutostiedostojen käyttäytymisen suhteen. Riittävän perusteellisella diagnosoinnilla ja analytiikalla onkin pystytty osoittamaan, että suljetun lähdekoodin ja avoimen lähdekoodin ohjelmistot eivät olennaisesti eroa toisistaan haavoittuvuuksien vakavuuden (severity of vulnerability), ajallisesti tapahtuvan haavoittuvuuden paljastumistyypin (the type of vulnerability disclosure over time), haavoittuvuuden tyypin kehittyneisyyden tai ohjelmistojakelijan korjauskäyttäytymisen suhteen (vendors ´ patching behavior) (Schryen, 2011). Luottamus (reliance) ja luottamuksellisuus (confidentiality) sen sijaan korostuvat riippumattomien asiantuntijoiden päästessä vapaasti tutkimaan ja analysoimaan ohjelman rakennetta ja toimintaa, ja siten avoimesti kertomaan löydöksistään, joka puolestaan mahdollistaa sen, ettei ohjelman tai ohjelmistojen käyttäjien tarvitse luottaa ainoastaan ohjelman, ohjelmiston tai ohjelmakoodin luoneen osapuolen sanaan (Linux.fi 2011; Alhazmi, Malaiya & Ray 2007).

Vaikkakin Schryen (2011) tutkimuksessaan toteaa, että avoimen lähdekoodin ohjelmistojen kehittyminen näyttäisi ehkäisevän ”erittäin huonoa” paikkaamiskäyttäytymistä (patching behavior), ei yleisesti ottaen ole muodostunut minkäänlaista empiiristä todistetta siitä, että jokin erityinen ohjelmistokehityksen tyyppi, olipa kyseessä avoin tai suljettu lähdekoodi, ensisijaisesti ohjaisi tietoturvaa millään tavoin. Pikemminkin ohjelmistojakelijan politiikka on avainasemassa vaikuttamaan muutostiedostojen avulla tehtävään korjauspäivityskäyttäytymiseen (Schryen, 2011).

Tutkimustulostensa pohjalta Schryen ja Rich (2010) ehdottavatkin olennaisena menettelynä vahvojen taloudellisten kannustimien tarjoamista ohjelmistojakelijoille, jotta nämä toimittaisivat korjauksia ainakin siihen mennessä julkaistuihin tai toistaiseksi vielä julkaisemattomiin haavoittuvuuksiin. Avoimen lähdekoodin sosiaalisena ympäristönä on usein virtuaalinen online-yhteisö (Wasko ja Faraj, 2005.)

2.2 Muutostiedostot

Muutostiedostojen eli ohjelmistopäivitysten hyväksymisprosessia on tutkittu, ja aiheesta on saatavilla myös alan tieteellisiä julkaisuja. Muutostiedostolla tarkoitetaan kahden eri tiedostoversion välistä, tietyssä formaatissa olevaa muutosta, jolla pyritään välttämään kokonaisten tiedostoversioiden kopiointi. Birdin ja kumppanien (2007) tutkimuksesta käy ilmi, että avoimen lähdekoodin ohjelmistojen (OSS) suosio on täysin riippuvainen työstä, jota vapaaehtoiset edistävät resurssien, aikataulujen ja kykyjensä kautta ja kustannuksella. Tutkimuksessaan Weißgerber kumppaneineen (Weißgerber, Neu & Diehl, 2008) tutki kahden suuren avoimen lähdekoodin projektin sähköpostiarkistoja ja etsi mm. vastausta kysymyksiin a) kuinka moni sähköposti sisältää muutostiedoston, b) kauanko muutostiedoston hyväksyminen kestää, c) vaikuttaako muutostiedoston koko sen hyväksymismahdollisuuksiin ja siihen d) kauanko aikaa kuluu sen hyväksymiseen. Vastaukset näihin tutkimuskysymyksiin olivat osin ennakoimattomia ja yllättäviä, joten net voisivat tulevaisuuden skenaarioissa olla avuksi muutostiedostojen hallinnassa (Weißgerber, Neu & Diehl, 2008).

Asundi ja Jayant (2007) ovat tutkineet muutostiedoston katselmoinnin prosessia OSS -projektien koodin katselmoinnin laajuuden edustajana. Tutkimuksen kannalta huomioitavaa oli muutostiedostojen hyväksyttäväksi jättämisen muodostuminen merkittäväksi tavaksi, jonka kautta hankkeen ydinryhmän ulkopuoliset tahot pystyivät antamaan panoksensa kyseiseen hankkeeseen. Bird ryhmineen (Bird et al., 2007) onkin todennut, että muutostiedostojen ehdotus- ja hyväksyttämisprosessi osaksi lopullista lähdekoodikokoelmaa on avoimen lähdekoodin palapelin tärkeä ja olennainen palanen, ja että muutostiedostoihin liittyvän datan käyttö voisi suurella todennäköisyydellä olla hyödyllistä tietoa ymmärrettäessä OSS-hankkeiden toimintaa. Siinä missä olemassa olevat muutostiedostojen katselmointiprosessien kuvaukset usein perustuvat yksilöiden kuvauksiin, Asundi ja Jayant (2007) ovat analysoineet systemaattisesti viiden OSS-projektin sähköpostiarkistot tämän tyyppisen prosessin kuvaamiseksi. Tällaisen menettelyn avulla he ovat samalla pystyneet tällaisen prosessin avulla analysoimaan eroavaisuuksia ydinryhmän työpanosten, kuten muutostiedostojen tai katselmointien kommenttien ja ydinryhmän rooliin siirtyneiden satunnaisten avustajien välille (Asundi & Jayant, 2007).

Bird ja kumppanit (2007) puolestaan esittelevät menetelmänsä muutostiedostojen esittämisessä, tunnistamisessa ja hyväksynnässä, ja näin esittävät tutkimuksessaan tuloksia sekä arviointia edellä mainituista menetelmistä sovellettaessa niitä Apache webserverin, Python-tulkin, Postgres SQL -tietokannan ja rajoitetusti myös MySQL-tietokannan hankkeiden suhteen. Tähän liittyen Bird kumppaneineen (2007) esittää arvokkaita tapoja, mahdollisia käytäntöjä, joissa tällaista dataa on käytetty ja jossa sitä voidaan tulevaisuuden skenaariot huomioiden hyödyntää ja käyttää. Asundi ja Jayant (2007) ovat myös tutkineet aihetta ja näin on voitu osoittaa, että siinä missä muutostiedoston katselmointiprosessit eivät ole täysin identtisiä tyypillisten OSS-projektien kesken, kaikkien projektien ydinjäsenet painottavat portinvartijan olennaista ja aktiivista roolia, jotta hyväksyttyjen muutostiedostojen riittävän korkeatasoinen katselmoinnin laatu voitaisiin taata.

Yksi tärkeimmistä periaatteista avoimen lähdekoodin ideologian ja filosofian mukaan on se, että kuka tahansa voi haluamallaan tavalla päättää osallistumisestaan ja osallisuudestaan tiettyyn, omia lähtökohtia parhaiten vastaavaan hankkeeseen, tässä tapauksessa OSS-tyyppiseen projektiin (Bird et al., 2007). Tällaisten OSS-hankkeiden menestys on huomattavan riippuvainen erityisesti uusista tulokkaista, jotka ovat kykeneviä ja halukkaita osallistumaan hankkeeseen, usein monipuolisestikin. Tyypillisimmillään tällainen osallistuminen näkyy dokumentoinnissa, koodin ohjelmistovirheiden korjaamisessa, uusien ominaisuuksien lisäämisessä, teknisen asiantuntemuksen jakamisessa ja tarvittavan tuen jakamisessa käyttäjille. Vaikka jokaisessa OSS-projektissa onkin kehittäjien ydinryhmä lähdekoodin (ja joissain tapauksissa myös dokumentaation) kirjoitusoikeuksineen, mukaan lukien kyseisen ohjelman tai ohjelmistohankkeen koodivarastot, voivat myös tulokkaat ilman tätä valtuutusta osallistua hankkeeseen omalla, lähinnä henkilökohtaisia vahvuuksiaan vastaavalla osuudellaan. Nämä osuudet ovat tyypillisimmillään ja pääosin tehty toimittamalla muutostiedostoja projektin postituslistoille (Asundi & Jayant, 2007.)

Ohjelmistoprojektin postituslistalle (mailing list) ja usein myös itse projektiin osallistuvien joukko on yleensä suurempi kuin sisäpiirissä toimivien kehittäjien joukko. Kuitenkin vain osa hankkeeseen osallistuvista listaa käyttävistä osapuolista todella jättää aikaansaannoksensa käsiteltäväksi muutostiedostona. Tämän vuoksi muutostiedostojen esittämisen ja hyväksymisen prosessit ovat ratkaisevan tärkeitä OSS-yhteisöille ja siten tutkimuksen arvoisia. (Bird et al., 2007.) Tutkijat esittävätkin erään tähän mennessä lähtökohtaisesti unohtuneen tavan kerätä tietoa muutostiedostojen

jättämisestä ja hyväksymisestä. Louhimalla (data mining) aikaan saatu data voi antaa Birdin ja kumppanien (2007) näkemyksen mukaan tutkijoille yksityiskohtaisempia ja hienommiksi partikkeleiksi jauhettuja näkemyksiä OSS-yhteisöistä, joka pääasiassa on jakautunut kehittäjien (contributors), kehittäjien dev-postituslistan osanottajien ja käyttäjien rooleihin. Bird ja kumppanit (2007) ovat kehittäneet menetelmän joka sekä havaitsee muutostiedoston esittämisen projektin postituslistoille tutkien samalla, onko toimitettu ja hyväksytty muutostiedosto liitetty projektin koodihakemistossa olevaan lopulliseen lähdekoodikokoelmaan (codebase).

Lähdekoodin tarkastamisen (source code inspections) käsite syntyi yli kolmekymmentä vuotta sitten kustannustehokkaaksi tekniikaksi ja metodiksi etsimään ja tunnistamaan tietokoneohjelmistojen ohjelmointivirheitä ja vikoja. Tällainen ohjelmien puhdistamisprosessi määriteltiin suoritettavaksi katselmointien avulla (Fagan, 1976.) Aiheeseen liittyen on kirjoitettu ja julkaistu runsaasti myös tieteellisesti arvioitua kirjallisuutta, jossa on käsitelty ja tutkittu näitä prosesseja ja komponentteja (ihmisiä, dokumentteja ja aiheeseen liittyviä artefakteja), jotka liittyvät koodintarkastukseen. Tähän liittyen, vertaisarvioinnin tehokkuuden ja tärkeyden ovat OSS-ohjelmistoprojektien suhteen hahmotelleet ja todenneet myös Feller ja Fitzgerald (2002) tutkimuksessaan. Bird kumppaneineen (2007) esittelee puolestaan tutkimuksensa havaintoja sekä alustavan arvioinnin, soveltaen näitä menetelmiä usean ohjelmistotuotteen, Apachen, Pythonin, PostgreSQL'n ja MySQL:n kaltaisiin projekteihin ja hankkeisiin.

Asundi ja Jayant (2007) ovat tutkimuksessaan analysoineet myös avoimeen lähdekoodiin perustuvia projekteja. Tutkimuksen kohteena olevia projekteja ja hankkeita ovat olleet mm. Apache, GCC, GDB, Mplayer ja Gaim. Tällaiset projektit ovat metodologialtaan monimuotoinen sekoitus kooltaan monimuotoisia tietojärjestelmiä, kohdennettuja loppukäyttäjätyyppejä, sovellusten monimuotoisia ympäristöjä ja vaihtelevia tapoja käyttää tietojärjestelmiä, joten tämän vuoksi ne eivät edusta ominaisuuksiltaan keskenään saman aikakauden ohjelmistotuotteita (Asundi & Jayant, 2007). Edellä maininut projektit poikkeavat toisistaan myös siten, että siinä missä Apache, GCC ja GDB edustavat ns. infrastruktuuriltaan ns. kriittisiä sovelluksia, Gaim ja Mplayer voidaan puolestaan luonnehtia ja luokitella ns. viihteellisiksi, ei-kriittisiksi sovelluksiksi (Bird et al., 2007). Neljän sovelluksen ja ohjelmistotuotteen kohdalla projektia avustavia tahoja neuvotaan toimittamaan muutostiedosto ohjelman tai ohjelmistotuotteen kehittäjien mahdolliselle postituslistalle (Asundi & Jayant, 2007). Tämä menettely helpottaa ja yksinkertaistaa muutostiedoston kommentointia. Toisaalta keskimääräisen Apache-loppukäyttäjän vuorovaikutus käyttämänsä sovelluksen suhteen ei ole läheskään niin intensiivistä tai kokonaisvaltaista kuin loppukäyttäjillä, jotka käyttävät GCC-, GDB-, Gaim- ja Mplayer -sovelluksia, toteavat Asundi ja Jayant (2007) tutkimuksessaan.

Asundi ja Jayant (2007) analysoivat tutkimuksessaan neljän sovelluksen sähköpostilistan arkistot kunkin projektin tai hankkeen Internet-sivuilta. Arkiston sähköpostien tutkimuksen kannalta olennaisten tietojen, kuten lähettäjän henkilöllisyyden, viestin päiväyksen, otsikkotietojen ja muiden olennaisten tunnisteiden poimimiseen tarvittavien hakukriteerien pohjalta käytettiin Perliä (Practical Extraction and Report Language). Tämän jälkeen otsikkotiedot jäsenneltiin säännönmukaisiksi lausekkeiksi, joita oli käytetty ja edelleen käytettiin tunnisteina muutostiedostojen hyväksynnän yhteydessä. Huolimatta siitä, että muutostiedostojen kehittäjäyhteisöä oli opastettu sisällyttämään asiasana ”[PATCH]” jätettävän muutostiedoston otsikkokenttään, käyttivät monien muiden projektien jäsenet vapaamuotoisesti samankaltaista mutta säännönmukaista tunnistetietoa helpottamaan ja

yksinkertaistamaan muutostiedoston jättämisprosessia. Muutostiedoston esittämistä seuraa tämän tyyppisessä prosessissa usein sähköpostikeskustelulle ja uutisryhmälle tyypillinen keskustelusäikeen (thread) muodostuminen (Asundi & Jayant, 2007.) Asundin ja Jayantin (2007) tutkimuksessa käytettiin viestin otsikkoriviä ja numeroa kytkemään ja yhdistämään säikeen viestejä loogiseksi kokonaisen keskustelusäikeen kattavaksi kokonaisuudeksi, jonka jälkeen säikeen vastausten määrä oli mahdollista laskea. Useimmissa projekteissa ohjelmiston tai ohjelmistoprojektin kehittäjien sähköpostilista on vain yksi funktio, media jonka avulla muutostiedostoja esitetään katselmoitavaksi ja liitettäväksi projektiin. Muutostiedostot on luotu ja luodaan saman tai samojen tiedostojen alkuperäisestä ja modifioidusta tiedostosta käyttäen yleensä tarkoituksen sopivaa Unixin diff-apuohjelmaa. Tämä on yksi syy, minkä vuoksi muutostiedostot yleensä esiintyvät useissa eri formaateissa (Bird et al., 2007.)

Tutkimuksessaan Bird kumppaneineen (2007) on kiinnostunut erityisesti niistä muutostiedostoista, jotka varsinaisesti on hyväksytty ja tullaan hyväksymään ohjelmiston lopulliseen lähdekoodikokoelmaan. Ylemmällä tasolla tällainen prosessi vaikuttaisi suhteellisen merkityksettömältä. Tietyillä muutostiedostojen asennuksen yhteydessä tehtävillä parametreilla ja komennoilla on mahdollista suorittaa muutostiedoston käänteinen päivitys. Oman ongelmansa tähän lähestymistapaan tuo se, että muutostiedostoihin kohdistuvia tiedostoja on usein modifioitu ennen kuin ne on päätetty sijoittaa kyseisen ohjelmistohankkeen tai projektin koodiarkistoon. Jos kehittäjä lisää ja liittää hyväksytyn muutostiedoston ja samalla muuttaa koodia mm. siirtämällä aaltosulkeet rivin lopusta seuraavan rivin alkuun, ei kyseisen alkuperäisen muutostiedoston käänteinen suorittaminen enää onnistu, ja tämän seurauksena muodostuu väärä negatiivinen tulos (false negative). Tällaisessa tapauksessa ongelmana on se, että muutostiedostot ja niiden rakenne muodostuu yksinomaan tietokoneohjelman riveistä (Bird et al., 2007.) Muita Birdin (Bird et al., 2007) ja kumppanien tutkimuksessaan huomioimia havaintoja olivat mm. väärät negatiiviset tulokset, jotka muodostuivat koodin lisätyistä, koodin muutoksiin, lisäyksiin ja poistoihin kantaa ottavista kommenttiriveistä. Nämä kohdistuivat muutostiedostoihin ennen tiedostoihin liittyvien muutostiedostojen suorittamista, ja ne koskivat myös koodissa olevia, uudelleen nimettyjä muuttujia (Bird et al., 2007). Lähdekoodin tutkimista ja tarkastamista on aina pidetty tehokkaana ohjelmistojen virheiden havaitsemisessa. Muutostiedostojen tarkastelunprosessin voidaan todeta vastaavan OSS -ohjelmistokehitysprosessien vastaavaa tarkasteluprosessia suljetuissa kaupallisissa projekteissa tapahtuville koodin läpikäynneille ja tarkastuksille. (Asundi ja Jayant, 2007.) Asundi ja Jayant (2007) havaitsivat tutkimuksessaan myös, että OSS-projekteissa muutostiedostojen katselmoinnit ja katselmointiin liittyvät ominaisuudet vaihtelivat. Eräs mahdollinen selitys toimenpiteiden määrälliselle vaihtelulle voisi selittyä projektiin osallistuvien tahojen kulttuuristen taustojen eroavaisuuksilla (Asundi ja Jayant, 2007).

Projekteissa Apache HTTPD, GCC ja GDB ilmeni tutkittaessa tiettyjä erityispiirteitä kuten prosentuaalisesti korkeaa ydinryhmän osallistumista ja katselmointia ja ohjelmakoodin katselmointiin osallistumista. Tämä voi selittyä sillä, että nämä avoimen lähdekoodin projektit ovat suhteellisen vanhoja, ja niihin on saatu mukaan useita (suhteellisen) vanhoja kehittäjiä, jotka noudattavat tiettyjä, hyviksi havaittuja formaaleja prosesseja. Vaihtelut voivat johtua myös siitä, että tuotteen monimutkaisuus voi vaatia normaalia tiukempaa tarkastelua. Muita tekijöitä jotka vaikuttavat ei-ydinryhmän jäsenen eli ryhmän ulkopuolisen tahon osallistumiseen prosessin katselmointiprosessissa voivat olla dokumentaation saatavuus, ohjelmointityylit ja projektin suosio (Asundi & Jayant, 2007.) Tutkimuksessaan Weißgerber kumppaneineen (Weißgerber et al., 2008) sovelsi Birdin ja kumppaneiden (Bird et al., 2007) uudelleen

implementoimaa tekniikkaa purkaakseen korjaustiedostot sähköposteista ja löytääkseen niiden sovellukset projektin arkistosta. Lisäksi tutkijat määrittelivät sen milloin muutostiedostoa voidaan pitää hyväksyttynä. Weißgerber ryhmineen (Weißgerber et al., 2008) sovelsi kuitenkin edellä mainittuja tutkimustekniikoita kahden avoimen lähdekoodin projektin postituslistoille ja CVS-arkistoihin.

Tulevaisuuden suunnitelmissaan tutkimusryhmän (Bird et al., 2007) tavoitteena on yhteistyössä muutostiedostoja käsittelevien tahojen kanssa etsiä ja jalostaa parempia työkaluja muutostiedostojen analysointia varten. Tärkeimmät näiden projektien osalta saadut tulokset Weißgerberin ja kumppanien (Weißgerber et al., 2008) tutkimuksessa tehtyyn tapaustutkimukseen liittyen ovat seuraavat: molemmissa hankkeissa noin 3-4 % postituslistalle lähetetyistä sähköposteista sisälsi muutostiedostoja. Vain noin 6 % tiedostoista ovat koskeneet ainakin kerran hyväksyttyä muutostiedostoa. Kummassakin tutkimuksen kohteena olleessa projektissa muutostiedoston hyväksymisen todennäköisyys on noin 40 %. Mikäli muutostiedostot ovat hyväksyttyjä, hyväksyminen tehdään yleensä nopeasti. FLAC:ssa[2], joka on audiotiedostojen työstämiseen kehitetty työkalu ja jonka kehitys tapahtuu SourceForgen tarjoamassa kehitysympäristössä, puolet hyväksytyistä muutostiedostoista on tehty yhden viikon aikana. Weißgerberin ja kumppanien tutkimuksessa (Weißgerber et al., 2008) mainitussa projektissa OpenAFS[3], joka puolestaan on avoimeen lähdekoodiin perustuva tiedostojärjestelmä, jopa 61 % muutostiedostoista hyväksyttiin vain kolmessa päivässä.

Asundin ja Jayantin (2007) tutkimus osaltaan edistää olemassa olevaa empiiristä OSS-projekteihin liittyvää kirjallisuutta ja siten osaltaan vahvistaa käsitystä siitä, että heidän tutkimuksensa on ensimmäisten OSS-projektien vertaisarviointeihin perustuvien empiiristen tutkimusten joukossa. Asundi ja Jayant (2007) ovatkin luonnehtineet useita innovatiivisia toimenpiteitä, joita voidaan käyttää OSS-projekteihin liittyvien laajojen vertaisarviointien ja projekteihin osallistuvien yhteisöjen seuraamiseen. He ovat tutkimuksensa perusteella pystyneet todistamaan, että nämä määrälliset mittaustulokset osaltaan auttavat muodostamaan perustan OSS-projektien paremmalle ymmärtämiselle ja antavat samalla keinoja seurata ja kontrolloida ohjelmistoprojektien kehityksen laatua myös tietoturvan kannalta. Asundin ja Jayantin (2007) tutkimus osaltaan johdattaa useiden mielenkiintoisten tutkimuskysymysten tekijän pariin, joka määrää muutostiedostojen katselmoinnin laajuuden OSS-projekteissa. Ovatko nämä tunnistettavat piirteet sellaisia, jotka todennäköisesti johtuvat korkeasta (tai matalasta) katselmoinnin ja tarkastelun tasosta, millainen määrä katselmoimatonta koodia on sellaista, joka kelpuutetaan osaksi lähdekoodikokoelmaa (codebase), miten katselmoinnin laajuus vaikuttaa OSS projektiin kokonaisuudessaan ja onko niillä merkittäviä vaikutuksia tuotteen laatuun, projektin onnistumiseen ja/tai muihin tuotteen tekijöihin tai mittaustuloksiin (Asundi & Jayant, 2007.)

Pienten muutostiedostojen (tiedostot, joiden koodista enintään 4 riviä on muutettu) mahdollisuudet tulla hyväksytyksi ovat (Weißgerber et al., 2008) tutkimuksen perusteella keskimääräistä korkeammat. Hyvin suurten muutostiedostojen suhteen mahdollisuudet puolestaan ovat merkittävästi keskimääräistä alhaisempia, joten muutostiedoston koolla on vaikutusta mahdollisuuteen saada muutostiedosto hyväksytyksi. Molemmissa Weißgerberin ja kumppanien (Weißgerber et al., 2008) tutkimuksessa tutkituista projekteissa tekevien (committers) osapuolten määrä oli alhainen verrattuna muutosta ehdottaneiden osapuolten (submitters) määrään (suhde noin 0,1-0,2). Vaikka ehdottavien osapuolten määrä on suuri, molemmissa projekteissa

2http://flac.sourceforge.net

3http://www.openafs.org

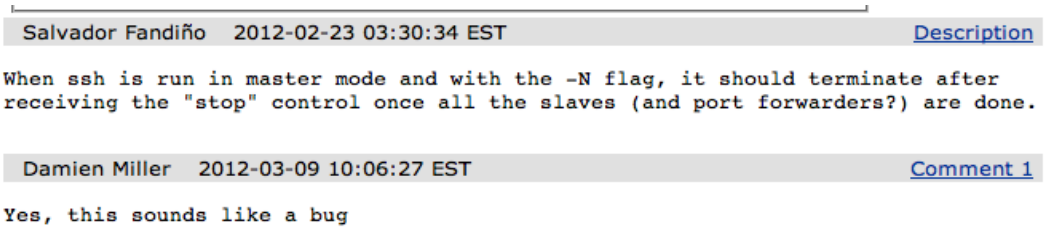
on olemassa pieni 4-5 muutosta ehdottavien osapuolten ryhmä, jotka tekevät erityisesti paljon muutostiedostojen jättämisiä ja ehdotuksia. Kaikki projektin postituslistalle (mailing list) lähetetyt korjaukset eivät välttämättä ole kyseiseen projektiin liittyviä muutostiedostoja, vaan niitä voidaan myös osoittaa liittyviin projekteihin (esim. yhteensopivuuden varmistaminen tai parantaminen) (Weißgerber et al., 2008.)

2.3 Bugzilla

Tutkimuksessa olennaisessa roolissa on Mozilla Foundation'in[4] kehittämä ja hallinnoima Bugzilla[5] http://bugzilla.mindrot.org, joka on yleiskäyttöinen tietokoneohjelmien virheiden raportointiin kehitetty, avoimeen lähdekoodiin perustuva web-pohjainen testaus- ja raportointityökaluohjelmisto (ks. kuva 1) (Serrano & Ciordia, 2005).

Kuva 1. Bugzilla-järjestelmän päänäkymä (https://bugzilla.mindrot.org).

Bugzillan suosiosta kertoo se, että sen on valinnut työkalukseen niin useat avoimen lähdekoodin projektit kuin kaupalliset projektit. Bugzillan avoimuudesta kertoo se, että kuka tahansa voi halutessaan raportoida havaitsemistaan virheistä (ks. kuva 2) järjestelmään, josta ne voidaan ohjata tietyille kehittäjille (Serrano & Ciordia, 2005).



Kuva 2. Uusi virheilmoitus (https://bugzilla.mindrot.org/show_bug.cgi?id=1985).

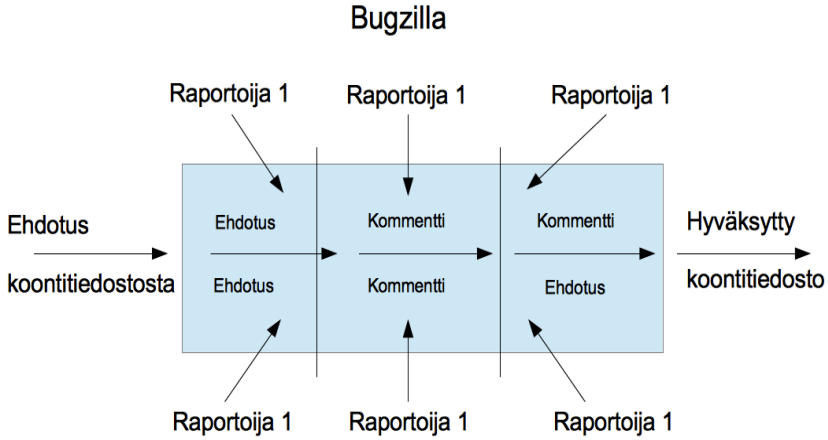
4http://www.mozilla.org/foundation/

5http://bugzilla.mindrot.org

Raportoinnin seurauksena Bugzilla (Serrano & Ciordia 2005; Bugzilla 2012) toimittaa raportoidut virheet oikeisiin osoitteisiin, tässä tapauksessa kyseisen ohjelmiston kehittäjille.

Bugzillan ominaisuuksiin kuuluvat myös virhekohtaiset tilapäivitykset, huomautusten lisääminen sekä esimerkkitilanteita. Ohjelmistoa voidaan soveltaa myös ohjelmistokohtaisten ominaisuuspyyntöjen keräämiseen (Serrano & Ciordia, 2005).

Virheen tietoihin ja tilapäivityksiin voidaan (Serrano & Ciordia 2005; Remillard 2005) liittää virheen kannalta olennaista tietoa, kuten tilapäivitykset, käyttäjän huomautukset ja virheen kannalta olennaiset virheen ilmenemistilanteet.



Kuva 3. Yksinkertaistettu kaavio Bugzillan toiminnasta.

Virheiden raportoinnin lisäksi Bugzillaa voidaan hyödyntää myös ohjelman kannalta olennaisten ominaisuuspyyntöjen keräämistyökaluna (Serrano & Ciordia, 2005). Bugzillaan kirjatun ohjelmistovirheen elinkaari on upotettu (ks. kuva 3) järjestelmään sisälle, ja se käsittää useita eri tilapäivityksiä, jotka kertovat vallitsevasta tilanteesta ohjelmistovirheen elinkaaressa (Remillard, 2005).

Tikettien tilapäivitysten seuraaminen ei välttämättä ole riippuvainen Bugzillan käytöstä. Tikettien tilapäivityksiä eli statuksia koskevia muutoksia voidaan usein seurata myös sähköpostin kautta liittymällä sitä varten perustetulle sähköpostilistalle.

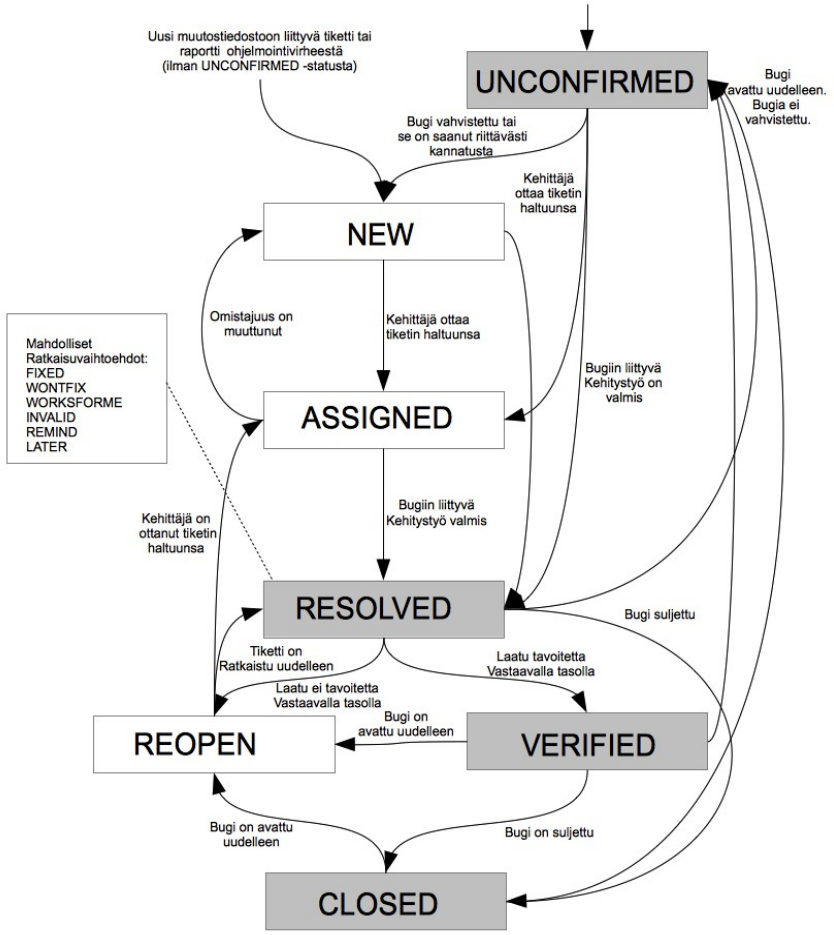
Bugzilla-ympäristössä avointen tikettien status voi olla UNCONFIRMED (tiketti on uusi, eli se on juuri lisätty järjestelmään), NEW (tiketti on lisätty tietylle virhelistalle ja se tulee käsitellä), ASSIGNED (tikettiä ei ole ratkaistu ja se on osoitettu sisällön kannalta oikealle henkilölle), REOPENED (tiketti on ratkaistu, ne on avattu uudelleen, koska ongelma ei ole ratkennut) tai READY (tiketistä on tarpeeksi tietoa jotta kehittäjä voi aloittaa sen korjaamisen).

Suljettujen tikettien osalta status voi olla RESOLVED (ongelma on ratkaistu ja tiketti odottaa seuraavaa vaihetta eli laaduntarkkailua) tai VERIFIED (Laadunvalvonta on tutkinut tiketin ja ongelman ja hyväksyy ongelman ratkaisun riittävällä tarkkuudella).

Muita suljettujen tikettien tilapäivityksiä ovat FIXED (tiketin ongelman korjaus on tarkistettu ja testattu), INVALID (kuvattu ongelma ei ole ohjelmistovirhe), WONTFIX

(kuvatun kaltaista ongelmaa ei voida korjata), DUPLICATE (ilmoitettu ongelma on toisen ongelman duplikaatti), WORKSFORME (ongelman ei saatu toistettua tai koodin analysointi ei tuottanut haluttua ratkaisua) ja INCOMPLETE (tiketillä oleva ongelma on epämääräinen tai puutteellinen) (Serrano & Ciordia 2005; Remillard 2005).

Tämän tutkimuksen kannalta olennainen aineisto on statukseltaan (ks. kuva 4) joko RESOLVED, VERIFIED tai CLOSED (tiketti on suljettu).



Kuva 4. Kirjatun tiketin elinkaari (http://www.bugzilla.org/docs/3.0/html/lifecycle.html).

Apuohjelman avulla luotu diff-tiedosto on hyvin usein lisätty Bugzillan tikettiin. Tietoturvan kannalta on olennaista, että koodi on tässä muodossa nähtävillä, mutta järjestelmän kannalta olennaiset aikaleimat pitävät yllä eheyttä eli liitetiedostossa olevan koodin manipulointi ei ole mahdollista ilman, että toimenpiteestä jää jonkinlaisia merkintöjä tai sormenjälkiä.

Tyypillisesti Bugzillassa oleviin muutostiedostoihin liittyvässä kommunikoinnissa käytetty kieli lyhenteineen ja termeineen luo oman, persoonallisen merkityksensä järjestelmän sisällä käytyjen keskustelujen sisällölle. Keskusteluihin ja muutostiedoston kehittyessä muodostuneeseen dialogiin liittyy myös usein eri ammatti- ja harrastelija- tai muulle pienryhmälle ominainen puhetapa, slangi, joka on muotoutunut vallitsevan kulttuurin pohjalta ja siten käsittää jokaiselle järjestelmän käyttäjälle ainakin jossain

määrin tuttuja ns. uudissanoja. Bugzillassa käydyistä dialogeista välittyykin perustelu sille, että lyhenteiden ja termien merkitys on oletusarvoisesti avattu käyttäjille, eli niiden käyttö on perusteltua.

Bugzillan muutostiedostoihin liittyvien dialogien lauserakenteet eivät yleensä ole monimutkaisia eivätkä ilmaisut välttämättä ole moniselitteisiä, joten esimerkiksi tottuneelle uutisryhmien käyttäjälle muutostiedoston sisältö, olennainen asia todennäköisesti selviää kertalukemalla. Asian selkeyden kannalta turhia täytesanoja ei käytetä, ja Bugzillaan muodostuneen aiheen kannalta turha asia painottuu lähinnä dialogiin kirjoittavien tahojen sähköpostiohjelmissa tai dialogissaan käyttämiin tekstisisältöihin. Sisällön kannalta turhaa tiivistämistä ei erikseen tarvita.

Bugzillan sisältämien tikettien kommunikaatiossa käytetty kieli on luonteeltaan lähinnä teknistä muttei välttämättä virheetöntä, joka puolestaan voi kertoa kielellisten taitojen lisäksi järjestelmässä tai sähköpostilistalla käsitellyn asian kiireellisyydestä, eli kiireelliseen kysymykseen tai ongelmaan toivotaan kiireellistä vastausta tai ratkaisua. Epäselvissä tapauksissa osapuolet pyytävät lisätietoja keskustelun muilta osapuolilta kyseisen järjestelmän tai sähköpostilistan kautta. Vastaus voi tulla myös täysin ennakoimattomalta taholta, eli keskustelu voi kehittyä spontaanisti aikojen kuluessa.

Perinteiseen sähköpostitse tapahtuvaan kommunikointiin verrattuna Bugzillassa tapahtuva, dialogimainen ja yksinomaan englanninkielisenä tapahtuva kommunikointi on nopeaa. Muutostiedosto on eräs monista Bugzillan dialogin tyypeistä. Dialogit voivat kohdistua muihin ohjelmistotuotteisiin tai aiheen ympärille. Bugzillan kaltaisen järjestelmän etu on, että Bugzillassa olevan ohjelman tai ohjelmiston kehittymistä seuraa suhteellisen kattava ohjelmistokehittäjien joukko, ja usein tästä joukosta löytyykin joku, joka voi auttaa ilmaiseksi, olipa kyse millaisesta ongelmasta (vianmäärityksestä, asetusten konfiguroinnista, asennusavusta) tahansa. Olennaista tällaisissa pääasiassa ohjelmistoteknisten asioiden ympärille rakentuneessa dialogissa tai keskustelusäikeissä on, että turhan tietoaineksen tai kielioppivirheiden sijaan voidaan tiketin ympärille rakentuneen dialogin säikeessä keskittyä olennaiseen.

Bugzilla on olennaisessa roolissa työkaluna ja aktiivisessa käytössä ohjelmistotuotteiden ohjelmistovirheiden raportoinnissa ja ohjelmistokehityksessä. Keskustelevia osapuolia tarkasteltavassa Bugzillaan muodostuneen dialogin jokaiselle osapuolelle muutostiedostossa käsiteltävä aihe on vähintään jollakin tavoin entuudestaan tuttu. Vastausten lähestymistapa ja muutosehdotukset ovat luonteeltaan hyvin samankaltaisia, ja ne mukailevat sekä täydentävät toisia vastauksia että dialogin alulle saattanutta kysymystä. Keskustelusäikeessä voi esiintyä myös täysin keskusteluun liittymätöntä kommentointia tai aiheen kannalta epäolennaista vastausten sisältöä.

3. Tietoturva

3.1 Tietoturvallisuus

Tietoturva (information security) tai tietoturvallisuus on käsitteenä hyvin laaja, moniin eri osa-alueisiin jakaantunut kokonaisuus, ja se on määritelty prosesseiksi ja käytännöiksi, joiden mukaisesti tietojärjestelmät, niihin liittyvät tiedot, palvelut ja tietoliikenne pyritään suojaamaan kaikilta mahdollisilta sisäisiltä ja ulkoisilta uhilta (Layton, 2007). Tietoturvaan keskeisesti liittyvistä tietoturvauhista muodostuikin 2000-luvun vaihteen ja erityisesti ICT-alan median keskeinen huomion kohde. Tämä osaltaan lisäsi virustorjuntaohjelmistojen ja niitä myyvien yritysten määrää (COSS, 2011.)

Tietoturvallisuuden synnyttämiä tai tietoturvaan liittyviä uhkina ovat Laytonin (2007) ja Allenin (Allen, 2001) näkemysten mukaan esimerkiksi sähköpostien välityksellä leviävät huijausyritykset (hoax), yksityisyyden loukkaaminen, toisten käyttäjien käyttäjätunnusten tai salasanojen luvaton käyttäminen tai hyödyntäminen, sähköpostin lähettäminen tekaistuilla nimillä tai valeprofiililla, tietoverkossa tai verkkoympäristöissä ilman lupaa tapahtuva Internetsivujen tai sivustojen muokkaaminen ja Internet-hakukoneiden avulla tehtävien Internethakujen ohjaaminen vääriin osoitteisiin (Whitman & Mattord 2008; Suomen Internetopas 2011). Globaaleiksi ongelmiksi hyvin yleisesti koetaan Internetin postipalvelimia toisinaan raskaastikin kuormittava ja toisinaan jopa palvelinten tietoturvaa koetteleva roskaposti (spam), tietoteollisuuteen erityisesti liittyvä, laiton teollisuusvakoilu (industrial espionage), musiikki- ja ohjelmistoteollisuuden kanssa kamppaileva laittomien tallenteiden ympärille muodostunut, tekijänoikeuksia loukkaava ja tuoteväärennöksiä suosiva piratismi (Layton, 2007.) Keskeisiä uhkia tietoturvan kannalta ovat myös Dhillonin (Dhillon, 2007) ja Allenin (2001) tutkimuksissaan luettelemat haittaohjelmat, joista tunnetuimpana tietokonevirukset, terrorismiksi luokiteltava verkkoterrorismi ja sotateollisuuden ohjelmistovirheistä modernin informaatiosodankäynnin (information warfare) muodoksi luettava elektroninen sodankäynti, ELSO. Näiden voidaankin perustellusti sanoa olevan yksinomaan tietotekniikan mukanaan tuomia tietoturvariskejä ja haavoittuvuuksia (Dhillon, 2007.)

Peltier'n (Peltier, 2002) ja Tietoyhteiskunnan kehittämiskeskuksen (TIEKE, 2011) näkemysten mukaan tietoturvan ensisijainen ja optimaalinen tavoite olisi saavuttaa taso, jossa tietoturvan kustannukset olisivat vähintäänkin tasapainossa tietojärjestelmään olennaisesti liittyvän tiedon arvoon ja siihen kohdistuvan tietomurron aiheuttamiin tappioihin verrattuna. Allenin (2001) näkemyksen mukaan turvallisuus onkin riittävällä tasolla silloin, kun turvaratkaisuista on huolehdittu asianmukaisella tavalla ja niitä ylläpidetään. Waldenin ja kumppanien tutkimus (Walden, Doyle, Welch & Whelan, 2009) toisaalta todistaa, että ihmisten asioidessa verkkopankissa, kommunikoidessaan tai tehdessään ostoksia verkossa, tulevat he riippuvaisiksi näihin tarkoituksiin käytettävien ohjelmien ja ohjelmistojen turvallisuudesta. Identiteettivarkaudet, haittaohjelmien tartunnat ja muu tietokonerikollisuus muodostavat tutkimuksen mukaan (Peltier, 2002) miljardien eurojen ja dollarien kustannuksia kuluttajille ja yrityksille. Samalla murenee se luottamus, jota ihmiset tarvitsevat tehdäkseen liiketoimintaa ja

hankintapäätöksiään verkossa, Walden kumppaneineen (Walden et al., 2009) väittää tutkimuksensa yhteenvedossa.

Tutkimusten mukaan (Layton, 2007) ainoastaan murto-osa tietojärjestelmiä ja niiden toimintaympäristöjä uhkaavista uhista muodostuu ulkoisista uhista.

Vakavammaksi ongelmaksi ovatkin muodostuneet tietokoneohjelmien ohjelmointivirheet, jollaisia on kaikissa ohjelmistoissa, ja joista nykyään käytetään termiä haavoittuvuus (Allen, 2001). Haavoittuvuudet ovat usein ns. tietoturva-aukkoja, joita apuna käyttäen krakkerit eli hyökkääjät voivat hyödyntää omien tarkoitusperiensä ja päämääriensä saavuttamiseksi. Siksi onkin riittävän tietoturvan tason saavuttamisen kannalta merkittävää, että tällaisiin haavoittuvuuksiin, eli ohjelmistoissa ja ohjelmissa oleviin virheisiin voidaan puuttua mahdollisimman varhaisessa vaiheessa. (Alhazmi, Malaiya & Ray 2007; Jakubowska & Penczek 2007.)

3.2 Tietoturvan peruskäsitteet

Luottamuksellisuus (confidentiality)

Luottamuksellisuus (confidentiality) ja luottamuksellisuuden säilyminen ja säilyttäminen ovat Drevinin ja kumppanien (2007) tutkimuksen mukaan osa nykyaikaista tietoturva-ajattelua ja tietoturvan prosessimallia. Muita olennaisia peruskäsitteitä ovat eheys (integrity) ja saatavuus (availability). Luottamuksellisuutta ei kuitenkaan pidä sekoittaa toiseen tietoturvan ja turvallisuuden yhteydessä hyvin yleisesti käytettyyn termiin, luotettavuuteen (reliability), joka puolestaan tarkoittaa tietojärjestelmän moitteetonta toimintaa (Abbot, 1990.)

Luottamuksellisuudella tarkoitetaan sitä, että kaikki luottamukselliseksi määritelty tietosisältö, tietojärjestelmät ja niihin liittyvät palvelut ovat tarvittaessa vain niihin ja niiden käyttöön oikeutettujen tahojen ja osapuolten saatavissa ja käytettävissä (Kerievsky, 1976). Luottamuksellisuutta voidaankin parantaa nostamalla luottamukselliseksi määritellyn kohteen salauksen tasoa riittävän korkealle tasolle ja huomioimalla kohteen pääsynhallintaa riittävällä tasolla ja tarkkuudella (Whitman & Mattord, 2008). Tiedon salaaminen ja erilaiset salaustekniikat on kehitetty puolustamaan ja suojaamaan tietojärjestelmissä olevien tiedostojen luvatonta käyttöä ja erityisesti lukemista (Kerievsky, 1976).

Eheys (integrity)

Eheydellä tarkoitetaan (Whitman & Mattord, 2008) tietoturvan yhteydessä sitä, ettei järjestelmää, järjestelmään liittyvää ympäristöä tai järjestelmään liittyvää komponenttia tai osajärjestelmää ole muutettu tai manipuloitu luvatta. Eheydellä tarkoitetaan myös sitä, etteivät tietojärjestelmät, tietojärjestelmään liittyvät tietosisällöt tai järjestelmiin liittyvät palvelut ole muuttuneet mahdollisen ohjelmista, ohjelmistoista tai tietojärjestelmistä johtuvan vikatilanteen, luonnonilmiön tai luvattoman tai oikeudettoman (inhimillisen) toiminnan seurauksena (Kerievsky, 1976).

Jokainen muutos, olipa kyseessä pieni tai iso muutos, tekee ohjelmasta tai ohjelmistosta uuden (Dutta et al., 2006). On huomioitavaa, että pienikin muutos voi ohjelmassa tai ohjelmistossa voi aiheuttaa vakavia seurauksia (Parnas, van Schouwen & Po Kwan, 1990). Tietoturvaa onkin mahdollista eheyden osalta edistää eheyden tarkistamisen avulla, eli lisäämällä tietojärjestelmään, tietojärjestelmän tietoihin ja tietojärjestelmässä

toimiviin palveluihin tarkistuskoodeja, tarkistussummia tai digitaalisia allekirjoituksia (Drevin et al. 2007; Whitman & Mattord 2008).

Saatavuus (availability)

Saatavuudella tarkoitetaan Drevinin ja kumppanien (2007) mukaan palvelua, joka on tarvittaessa varmatoiminen ja esteettömästi hyödynnettävissä. Tietoturvallisuuden yhteydessä olennaista tietosisällön hyödynnettävyyden kannalta on sen saatavuus. Saatavuus on käsite ja ominaisuus, jota mm. tietojärjestelmässä olevat bugit eli virheet voivat heikentää. Saatavuuden merkitsemiskäytäntö vaihtelee, mutta ominaisuuksiin viittaava merkintä 24/7 tarkoittaa tietojärjestelmässä olevan palvelun toimintaa, jolloin tässä yhteydessä voidaan olettaa saatavuuden olevan taattua ja virheetöntä 24 tuntia vuorokaudessa viikon jokaisena päivänä (Drevin et al., 2007). Tieteellisissä julkaisuissa saatavuudella tarkoitetaan myös sitä, että kohteena olevan tietojärjestelmän, tietojärjestelmissä olevien tietosisältöjen ja tietojärjestelmiin liittyvien palvelujen esteetön hyödynnettävyys koskee tarvittaessa jokaista niihin oikeutettua osapuolta tai tahoa (Whitman & Mattord, 2008).

Pääsynvalvonta (access control)

Pääsynvalvonnalla pyritään varmistamaan, että tietoa, resurssia tai tietojärjestelmää ei voi käyttää luvatta (Whitman & Mattord, 2008). Pääsynvalvonta on osa tietojärjestelmän käyttövarmuutta, ja sen tavoitteena on eristää sekä tietojärjestelmät että tietojärjestelmiin liittyvät tietokoneohjelmat sekä niiden toimintaympäristössään että tietojärjestelmiä käyttävien osapuolten ja tahojen suunnalta siten, että ainoastaan välttämätön pääsy järjestelmiin on sallittu siihen oikeutetulta taholta tai osapuolelta (Tanhuamäki 2006; Drevin 2007).

Tunnistaminen (identification)

Tunnistamisella tarkoitetaan tietoturvallisuuden kannalta menettelyä (Henslin, 2001), jolla pyritään tunnistamaan käyttäjän tai tietojärjestelmän identiteetti ja siten yksilöidään tunnistamisen kohde. Tunnistamistoimenpide voi olla joko heikko tai vahva, ja toimenpide voi mahdollistaa luotettavan sähköisen asioinnin, eikä välttämättä edellytä kohteelta suoraa vuorovaikutusta tai toimenpiteitä. Yksinkertaisimmillaan tunnistaminen voi tapahtua esimerkiksi työpaikalla ja se voi kohdistua kollegoihin, henkilökuntaan tai vierailijoihin (Opetustoimen turvallisuusopas, 2013; Whitman & Mattord, 2008.)

Todentaminen (authentication, verification)

Osapuolten todentamisella (autentikointi) tarkoitetaan (Buyn et al., 2006) sitä, että osapuolet ja osapuolten identiteetti (henkilö, palvelu tai tietojärjestelmä) voidaan esimerkiksi pääsynvalvontaan liittyen tunnistaa luotettavasti. Todentaminen voi olla joko yksisuuntainen tai kaksisuuntainen, käyttäjän ja palvelun välinen toimenpide. Tässä yhteydessä olennaista todentamisen kannalta ovat todennettavan kohteen ominaisuudet. Todentamismenettelyssä voidaan esimerkiksi selvittää, mitä todennettava kohde tietää. Useiden modernien tietojärjestelmien tai niiden osajärjestelmien todentamis- ja tunnistamismekanismit on suunniteltu ja toteutettu suorittamaan halutut toimenpiteet samanaikaisesti (Parnas et al., 1990). Käyttäjän suhteen olennaista on käyttäjän identiteetin tunnistaminen, jolloin olennaisessa roolissa ovat sekä käyttäjän tunnistetiedot että niihin liittyvä rooli. Identiteetin todentaminen voidaan myös kohdistaa kaikkeen siihen, mitä todennettavalla taholla on hallussaan, ja se voidaan toteuttaa joko käyttäen tarkoitukseen sopivaa todentamismenetelmää tai

todentamisvälinettä (Buyn et al., 2006). Todennusmenetelmästä on mahdollista koostaan riittävän vahva yhdistelmä. Esimerkki järjestelmään liitetystä valtuutuksesta tai todentamisesta voisi olla kulunvalvontajärjestelmä, jonka toiminnallisuuteen kuuluu todennettavan tahon tai osapuolen kulkukortin lukeminen, henkilötietojen tarkistaminen tietokannasta ja oven avaaminen. Nämä toimenpiteet voidaan tehdä vaiheistettuna tai samanaikaisesti, joko yhtenä tai erillisinä prosesseina (Whitman & Mattord, 2008.)

3.3 Hyökkäykset

Puskurin ylivuotovirhe (buffer overflow)

Puskurin ylivuotovirhe on tekninen ongelma, joka aiheuttaa muutoksia tietokoneohjelman muistialueen indeksissä siten että kyseiseen muistipaikkaa osoittava indeksi ei osoita sille tarkoitettuun muistipaikkaan. Tämä aiheuttaa ohjelmiston prosessissa tarpeettonta muistin varaamista vapauttamatta sitä, joka puolestaan aiheuttaa tietoalkioiden vuotamisen toisiin, ohjelman muuhun toimintaan varattuihin indekseihin ja muistipaikkoihin. Erityisesti Puskurin ylivuotovirhe aiheuttaa tietoturvaongelmia erityisesti matalan tason ohjelmointikielissä. Puskurin ylivuodon seurauksena hyökkääjä voi suorittaa kohdejärjestelmässä haittaohjelmia. Puskurin ylivuotoa hyödynnetään erityisesti palvelunestohyökkäyksissä (Ruwase & Lam, 2004.)

Vaikkakin ohjelmistoista yritetään saada mahdollisimman turvallisia ja toimintavarmoja, ja vaikka niitä hyökkäyksiltä usein (Ruwase & Lam, 2004) sekä manuaalisesti että automaattisesti, löydetään käytettävistä ohjelmista ja ohjelmistoista edelleen puskureiden ylivuotoja (buffer overflow). Toisaalta toisin kuin monilla tutkijoita, jotka tietävät ja tiedostavat puskuriylivuotojen vakavuuden, ei keskimääräisellä ohjelmoijalla ole aiheesta juurikaan tietämystä (Owen, 2007).

Dynaaminen rajojen (bounds) tarkistus löytää virheellisten ohjelmistojen puskureiden ylitykset ennen niiden suorittamista, mikä osaltaan ennalta ehkäisee hyökkäyksiä ja osaltaan parantaa järjestelmän integriteettiä (Ruwase & Lam, 2004). Dynaamisten puskureiden ylivuotojen ilmaisimia ei ole käytössä kovinkaan laajasti, koska ne joko eivät pysty suojaamaan kaikilta puskureiden ylivuotohyökkäyksiltä, olemassa olevan koodin murtamiselta tai eivät reagoi kaikkiin riittävällä tarkkuudella (Cowan, Wagle, Pu, Beattie & Walpole, 2000). Tutkimuksen (Owen, 2007) mukaan lähdekoodien kommenteissa voi olla jopa mainintoja, että puskurien koon tulisi toistaiseksi tuntemattomasta syystä olla erittäin suuri. Tarkemmissa tutkimuksissaan Owen (2007) totesi syyksi puskurin ylivuoto-ongelmat. Ylivuotohaavoittuvuuksia hyödyntävät erityisesti puskurin ylivuotohyökkäykset (buffer overflow attacks) ovat eräs vakavimmista turvallisuusuhista (Ruwase & Lam, 2004). Enemmän kuin puolet tämän päivän laajasti hyödynnettävistä haavoittuvuuksista aiheutuu puskurin ylivuodosta, ja niiden merkitys kasvaa kaiken aikaa (Cowan, Wagle, Pu, Beattie & Walpole, 2000.)

Eräät lähestymistavat on kehitetty tarkistamaan ohjelmien käyttämien puskurien ylivuotojen haavoittuvuuksia (Alhazmi, Malaiya & Ray, 2007). Staattiset tarkastusmenetelmät käyttävät Shaon ja kumppaneiden (Shao, Xue, Zhuge, Sha & Xiao, 2004) haavoittuvuuksien tunnistamisessa tarkastusstrategianaan C-kielisen lähdekoodin ohjelmistotyökaluilla tehtävää analysointia. Tällaisen menetelmän tarjoama suojaus voi olla epätäydellinen ja epätarkka, koska näin voidaan havaita ainoastaan tunnettuja haavoittuvuuksia ja siten yleisen puskuriylivuodon toteamisongelma jää ratkaisematta. Ohjelman testausstrategia tarkastaa puskurin ylivuotohaavoittuvuudet ajamalla ja suorittamalla ohjelmia erityisillä syöttötiedoilla. Vaikkakin käytettäessä tällaista

strategiaa voidaan useimmat haavoittuvuudet saada kiinni, suojaus ei ole kokonaisvaltainen koska haavoittuvuuksien havaitseminen riippuu testidatan aiheuttamista ylivuodoista (Shao et al., 2004.)

Koodin injektointi (Code injection)

Koodi-injektion (code injection, koodin injektointi) kaltaiset haavoittuvuudet jatkavat (Andersson, Clark, Mohay, Schatz & Zimmermann, 2005) vahingollista toimintaansa siitäkin huolimatta, että sovellusten tietoturvaa ja turvallisuutta yritetään parantaa mm. tietoturvallisilla ohjelmointikäytännöillä ja riittävillä muutostiedostojen toimittamisella.

Koodin injektointihyökkäyksessä Anderssonin ja kumppanien (Andersson et al., 2005) tutkimuksen mukaan injektoidaan tiettyä suoritettavaa ohjelmakoodia ns. injektiovektorin välityksellä, joka ilmenee haavoittuvuutena kohteena olevassa prosessissa. Esimerkkejä injektointivektoreita hyödyntävistä hyökkäysstrategioista ovat puskurien ylivuodot ja formaattimerkkijonojen ohjelmointivirheet. Tällaisen prosessin päämääränä on Anderssonin ja kumppanien (Andersson et al., 2005) tutkimuksen mukaan pumpata toteutettavaa koodia ja lisätä ja vahvistaa kohteena olevan prosessin konekielisen käskyn kontrollointia siten, että siihen voidaan hyökkääjän toimesta viitata, ja näin siirtää kyseinen toiminnallisuus hyökkääjän pumppaamaan koodiin. Tämän tyyppisten ylivuotojen kaltaisten ohjelmistoihin ja tietojärjestelmiin kohdistuvien hyökkäysten osuus on noin puolet vuosittain löydetyistä ohjelmistohaavoittuvuuksista (Anderson et al., 2005).

Palvelunestohyökkäykset (Denial-of-Service)

Ruwasen ja Lam'in (2004) tutkimuksen perusteella on olemassa useita erityyppisiä, alhaista kaistanleveyttä hyödyntäviä palvelunestohyökkäyksiä (DoS), joita on mahdollista hyödyntää. Vaikkakin Ruwasen ja Lamin (2004) tutkimuksen mukaan palvelunestohyökkäyksellä usein viitataan vastustajan yritykseen keskeyttää, kumota tai tuhota tietoverkko, se on silti tapahtuma, huomattavan potentiaalinen uhka, joka joka tapauksessa vähentää tai eliminoi verkon kykyä suoriutua oletetuista toiminnoista. Laitteistoviat, ohjelmistovirheet, resurssien loppuminen, ympäristöolosuhteet tai kaikki näiden tekijöiden monimutkaiset vuorovaikutukset toistensa kanssa voivat aiheuttaa palvelunestohyökkäyksen. Vaikkakin hyökkääjät usein palvelunestohyökkäystä tehdessään käyttävät Internetin sallimia ominaisuuksia ohjelmistovirheiden hyödyntämiseen, ensisijaisina kohteina pidetään kuitenkin protokollatason tai suunnittelutason haavoittuvuuksia (Ruwasen ja Lam, 2004).

Tutkimuksessaan Ruwase ja Lam (2004) eivät paneudu erityisesti tunkeutumisen tunnistamisjärjestelmään (Intrusion Prevention System), vaan erityisesti ongelmien päällekkäisyyksiin, ja erityisesti siihen kuinka hyökkäykseen vastataan. Vikasietoisuutta voidaan (Ruwase & Lam, 2004) lieventää jopa solmujen (node) kumoamisella, ja tehokkaat protokollat rajoittavat mahdollisuuteen ilkeämieliselle resurssien tuhlaamiselle.

3.4 Tietoturva avoimessa lähdekoodissa

Avoimen lähdekoodin avoimuus mahdollistaa kaikkien havaittujen epäkohtien, ohjelmointivirheiden, haavoittuvuuksien vapaan korjaamisen. Tietoturva-aspektin avulla avoimen lähdekoodin ohjelmistoista voidaankin puhua vakavasti otettavana vaihtoehtona ja ratkaisuna tietokoneohjelmia vaivaavien tietoturvaongelmien eliminoimiseksi (Layton, 2007). Tutkimuksen (Schneider, 2000) mukaan eräs avoimen

lähdekoodin ohjelmistojen tietoturvaan liittyvä huolenaihe on kuitenkin se, että lähdekoodin ollessa kaikkien vapaasti luettavissa ohjelmistovirheiden aiheuttamat haavoittuvuudet olisivat myös helposti hyödynnettävissä kyseenalaisiin tarkoitusperiin.

Ohjelmiston tietoturva on osa ohjelmiston laatua. Spinellis'in ja kumppanien (Spinellis, Gousios, Karakoidas, Louridas, Adams, Samoladas & Stamelos, 2009) tutkimuksen mukaan ohjelman tai ohjelmiston laadullisten ominaisuuksien tutkiminen tapahtui joko salassa tutkimuksen suorittaneen organisaation sisällä tai se suoritettiin käyttäen tarkoitukseen sopivaa Black box -testaustekniikkaa (Chow 1978; Beizer 1995). Spinellisin ja kumppanien (Spinellis et al., 2009) mukaan avoimen lähdekoodin ohjelmistojen ilmaantuminen on muuttanut tätä kuvaa sallien meidän arvioida sekä ohjelmistotuotteita että prosesseja, jotka tuottavat niitä. Tällä tavoin (Spinellis et al., 2009) versionhallintajärjestelmiin, vikaseurantatietokantoihin, postituslistoihin, ja jopa Wikipediaan sisällytetty ohjelmistojen lähdekoodi ja siihen liittyvä ja tallennettu data sallii tutkijoiden mutta myös kenen tahansa arvioida ohjelmiston ja ohjelmistojen laatua avoimesti ja läpinäkyvästi. Myös suuri määrä avoimen lähdekoodin (usein kilpailevia) projekteja tekee mahdolliseksi verrata samalla verkkoalueella tai alalla olevien vertailukelpoisten tietojärjestelmien laatua (Schneider, 2000). Tutkimuksessa (Spinellis et al., 2009) on saatu näyttöä, että yhdistämällä historiallisen lähdekoodin tilannekuvat tai yhteenvedot merkittäviin tapahtumiin, kuten ohjelmistojen virhelöydöksiin ja ratkaisuihin, on mahdollista uppoutua todellisten ongelmien syihin ja seurauksiin.

Avoimen lähdekoodin tietoturvaa on Dunlapin (Dunlap, 2006) ja Cowanin (Cowan, 2003) mukaan tutkittu ja tutkitaan edelleen laajasti useilla eri tahoilla, ja tämän tutkimuksen tarkoituksena on tuoda käyttäjien ja kehittäjien kokemusperäinen näkökulma asiaan. Suomen kieleen on Wikipedian (2011) artikkelin mukaan muodostunut ja vakiintunut jako tietoturvaan ja tietosuojaan (privacy), joista jälkimmäinen käsitteenä kohdistuu erityisesti henkilötietojen luottamuksellisuuteen, ja ulottuu siten juridisesti myös rikoslakiin ja viime kädessä perustuslakiin ja perustuslaillisiin säännöksiin asti.

Greiner ja muut (Greiner, Boskovic, Brest & Zumer, 2003) ovat tutkimuksessaan vertailleet avoimen ja suljetun lähdekoodin ohjelmistojen tietoturvaa, ja päätyneet lopputuloksessaan siihen, että avoimen lähdekoodin ohjelmistot pyrkivät oletusarvoisesti olemaan tietoturvaltaan parempia kuin suljetun lähdekoodin ohjelmistot, joten vapaan lähdekoodin suosimisen tärkeimmät perusteet liittyvät nimenomaan turvallisuuteen. Cowan (2003) puolestaan on vertaillut tutkimuksessaan eri työkaluja ja menetelmiä avoimen lähdekoodin ohjelmistojen tietoturvan analysoimiseksi. Schryen ja Rich (2010) vertailivat empiirisessä tutkimuksessaan eroja avoimen ja suljetun lähdekoodin ohjelmistojen turvallisuudessa. Tunnetuimpia ja käytetyimpiä avoimeen lähdekoodiin perustuvia ohjelmistoja ovat (Schryen & Kadura, 2009) Firefox-selain, Thunderbird-sähköpostiohjelma, Apache-palvelinohjelma, OpenOffice-toimisto-ohjelmisto, Red Hat ja Debian -Linuxjulkaisut sekä tietokannat MySQL ja PostgreSQL (Allen 2003; Crowston et al., 2008.)

Kirjallisuudessa käydyt keskustelut (Schryen & Kadura, 2009) ovat osoittaneet, että saatavuuden lisääntyminen ja avoimen lähdekoodin ohjelmien ja ohjelmistojen sijoittaminen sekä henkilökohtaisiin että kaupallisiin ympäristöihin ovat johtaneet lähes hurmokselliseen väittelyyn siitä, ovatko avoimen lähdekoodin tietokoneohjelmistojen ja suljetun lähdekoodin tietokoneohjelmistojen tietoturva ja turvallisuus keskenään verrannollisia. Tämän tyyppinen väittely paljastaakin huomattavan kattavan ongelman arvioitaessa turvallisuutta ja tietoturvaa, ja jota perinteisesti vain harvoin on totuttu suorittamaan kvantitatiivisella tasolla (Schryen & Kadura, 2009). Vaikkakin muutamia

metodeja ja metriikoita on Schryenin ja Kaduran (2009) mukaan ehdotettu ja sovellettu empiirisessä tutkimuksessa, on metodologia vasta kehityksensä alkuvaiheessa, ja siten se on hyväksytty ainoastaan luotettavuuden saralla, ilman laajempaa ja huolellisempaa tutkimusta, missä määrin se voitaisiin hyväksyä, tai ylipäätään kysymystä siitä, tullaanko vaatimaan vielä tuoreempia ja uudempia malleja ja metriikkaa.

4. Tutkimusmenetelmä ja aineisto

4.1 Menetelmä

Sisällönanalyysissä on kyse siitä, että aineistoa tarkastellaan eritellen, yhtäläisyyksiä ja eroja etsien ja tiivistäen. Sisällönanalyysi on tekstianalyysin muoto, jonka avulla on mahdollista tarkastella jo valmiiksi tekstiin tai tekstimuotoon pohjautuvia aineistoja. Sisällönanalyysin ohella puhutaan useissa yhteyksissä myös sisällön erittelystä (Tuomi & Sarajärvi, 2002.)

Tuomen ja Sarajärven (2002) mukaan sisällön erittelystä puhuttaessa tarkoitetaan kvantitatiivista eli määrälliseen tutkimukseen perustuvaa dokumenttien analyysia, jossa kuvataan määrällisesti jotakin tekstin tai dokumentin sisältöä. Tutkittavat tekstit voivat olla melkein mitä vain: kirjoja, päiväkirjoja, haastatteluita, puheita ja keskusteluita. Sisällönanalyysin avulla pyritään muodostamaan tutkittavasta ilmiöstä tiivistetty kuvaus, joka kytkee tulokset ilmiön laajempaan kontekstiin ja aihetta koskeviin muihin tutkimustuloksiin. Tutkimusongelmasta riippuen voidaan esimerkiksi laskea tutkimuksen kannalta olennaisten sanojen esiintymistiheyttä tutkittavissa dokumenteissa. Sisällönanalyysilla tarkoitetaan sanallista tekstin sisällön kuvailua. (Tuomi & Sarajärvi, 2002.)

Tutkimuksen kannalta olennaisin vaihe on analysoida kerätty aineisto, tarkastaa aineiston mahdolliset virheet ja puutteet, tehdä aineiston pohjalta riittävän kattavat tulkinnat ja johtopäätökset, sekä kartoittaa tarve mahdolliselle aineiston tietojen täydentämiselle (Hirsjärvi et al., 2001). Tilastollisen tiedon saamiseksi aineistosta muodostettiin muuttujia. Kvantitatiivisessa mielessä ”aineiston järjestäminen tutkimusta varten oli analyysin kannalta työläin vaihe, eikä analysointivaiheen aloittaminen ole aina selvästi määriteltävissä”. (Hirsjärvi et al., 2001, 208 – 209).

Tuomi ja Sarajärvi (2002) luonnehtivat sisällönanalyysia sekä laadullisena sisällönanalyysina että sisällön määrällisenä erittelyprosessina. Molempia voidaan hyödyntää tutkimuksen luonteesta riippuen joko erikseen tai yhdessä samaa aineistoa analysoidessa. Sisällönanalyysia on mahdollista jatkaa tuottamalla mm. sanallisesti kuvatusta aineistosta määrällisiä tuloksia. Tuomen ja Sarajärven (2002, 107-108) mukaan ”sisällön erittelystä puhuttaessa tarkoitetaan kvantitatiivista dokumenttien analyysia, jossa kuvataan määrällisesti jotakin tekstin tai dokumentin sisältöä. Tutkimusongelmasta riippuen voidaan esimerkiksi laskea tiettyjen sanojen esiintymistiheyttä tietyissä dokumenteissa”.

Voidaan todeta (Töttö, 2004), että kaikki tutkimus on loppujen lopuksi itse asiassa hyvinkin pinnallista, usein vain pinnan raapimista. Tämän vuoksi ilmiön tutkiminen ei voi olla syvällistä, ja siten siihen pohjautuvalla tutkimuksella ei voida koskaan saavuttaa ilmiötä kokonaisuudessaan. Perusteellisella tutkimuksella voidaan kuitenkin tavoittaa monipuolista tietoa ilmiön luonteen ominaisuuksista, itse ilmiön luonteesta sekä ilmiöön liittyvistä syy-seuraussuhteista. Tämän vuoksi riittävän hyvin suunnitelluilla ja toteutetuilla tutkimusasetelmilla ja tutkimuksia toistamalla on merkitys tutkimuksen

onnistumisen kannalta. Olennaista on myös lähestyä tutkittavaa ilmiötä useista näkökulmista (Töttö, 2004.)

Laadullinen sisällönanalyysi tarkoittaa ensisijaisesti tutkimusaineiston pirstomista tarpeeksi pieniin osatekijöihin, jonka jälkeen pieniksi pirstotut osat käsitteellistetään. Viimeisenä työvaiheena käsitteellistetyt osatekijät järjestetään uudelleen uudenlaiseksi kokonaisuudeksi. Sisällönanalyysi voidaan tehdä usealla eri tavalla. Yleisimpiä ovat aineistolähtöisyys, teoriaohjaavuus ja teorialähtöisyys. Olennaisimmat näiden väliset erot ovat analyysin ja luokittelun perustuminen joko aineistoon tai valmiiseen teoreettiseen viitekehykseen (Hirsjärvi et al., 2001.)

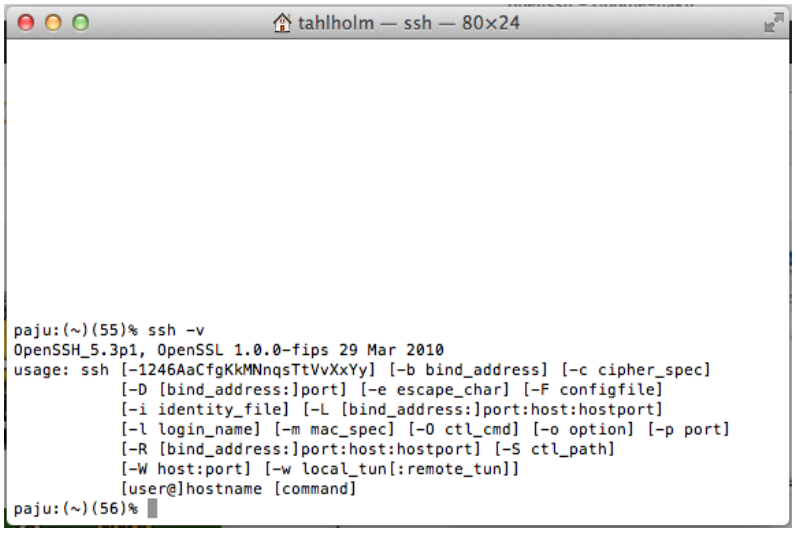
Tutkimusmenetelmänä sisällönanalyysia voidaan luonnehtia väljäksi teoreettiseksi viitekehykseksi sen sijaan, että puhuttaisiin selkeälinjaisesta tutkimusmenetelmästä. (Jokinen, Juhila & Suoninen, 1993). Tutkijan tulee myös olla kriittinen (Hirsjärvi, 2001) käyttämiensä lähteiden ja lähdemateriaalin valinnassa että tulkinnassa ja tulkitsemisessa. Tutkimuksen kannalta olennaista onkin riittävän tarkka esivalmistelu. Nämä seikat tulee Alexanderin ja Taten (Alexander & Tate, 1999) mukaan huomioida jo ennen kuin lähdemateriaaliin perehdytään syvällisemmin.

Kvalitatiivisen tutkimuksen seitsemän tyypillisintä piirrettä ovat: 1) tutkimuksen luonne, joka on kokonaisvaltaista tiedon hankintaa, ja aineiston kokoamista luonnollisissa, todellisissa tilanteissa, 2) ihmisen suosiminen tiedon keruun instrumenttina, 3) induktiivisen analyysin käyttäminen, 4) laadullisten metodien käyttäminen aineiston hankinnassa, 5) kohdejoukon tarkoituksenmukainen valitseminen, jolloin satunnaisotoksen menetelmää ei tulla käyttämään, 6) tutkimussuunnitelman muotoutuminen tutkimuksen edetessä ja 6) tapausten käsitteleminen ainutlaatuisena, jonka mukaisesti aineistoa tullaan tulkitsemaan. (Hirsjärvi, Remes & Sajavaara, 2000, 155.)

4.2 Tutkimusympäristö

Eräs merkittävä ja samalla yksi suosituimmista avoimen lähdekoodin ohjelmistoista, jossa tietoturva on olennaisessa roolissa on Theo de Raadtin johtamassa OpenBSD-käyttöjärjestelmäprojektissa kehitetty, ilmainen SSH-protokollaa käyttävä OpenBSD Secure Shell -ohjelmistopaketti, joka nykyään on saavuttanut monopoliaseman (Bauer 2001a; Weber 2005). OpenSSH on Damien Millerin (2007) tutkimuksen mukaan yleisin Secure Shell'in ilmaisista versioista, ollen samalla osa hyvin yleisesti käytössä olevaa SSH-protokollan implementaatiota (Hoskins, 2006). Se on myös saatavilla lähes kaikkiin UNIX-käyttöjärjestelmiin (Glass, 1999). Se on tietoverkkosovellus, ohjelmisto, joka tukee mm. etäkirjautimista (remote login), komentojen suorittamista (command execution), tiedostojen siirtämistä (file transfer) ja TCP -yhteyksien siirtämistä (TCP forwarding) eteenpäin asiakkaan (client) ja palvelimen (server) välillä (Allen 2003; Venkatachalam, 2007). Se on suunniteltu turvalliseksi käytettäessä ja toimittaessa epäluotettavissa verkoissa ja se sisältää kryptografisen autentikoinnin (cryptographic authentication), luottamuksellisuuden ja eheyden suojaamisen (Venkatachalam, 2007). Lähtien sen julkaisusta, vuodesta 1999, OpenSSH vahvisti nopeasti suosiotaan ja siitä tuli nopeasti Internetin suosituin SSH-implementaatio. OpenSSH onkin kehitetty (Miller, 2007) toimimaan erityisesti Unixin kaltaisissa käyttöjärjestelmissä ja Unix-varianttien kaltaisissa suurkoneympäristöissä (Stuart, 2009). Erityisesti OpenSSH-palvelin, sshd, vaatii pääkäyttäjän (root) oikeudet käyttäjien tunnistamiseksi ja kirjoittaakseen lokitietoa kirjautumisista (Frisch 1997; Venkatachalam 2007).

OpenSSH:n suosio on tehnyt siitä Millerin (2007) mukaan houkuttelevan kohteen sekä tutkimukselle että hyökkäyksille. Miller[6] on toinen Bugzillan OpenSSH-ohjelmiston muutostiedostojen hyväksynnästä vastaava taho. Tämän lisäksi jotkut kirjastoista, joista OpenSSH on riippuvainen, ovat Millerin (2007) mukaan kärsineet vioista, jotka paljastuivat OpenSSH:n käyttämisen yhteydessä ja käytön välityksellä. Alustavista ja edelleen jatkuvista koodien auditoinneista huolimatta OpenSSH on kärsinyt lukemattomista tietoturvahaavoittuvuuksista koko tähänastisen elinkaarensa ajan (CVE, 2013). Tämä on saanut ohjelmiston kehittäjät toteuttamaan useita suojaavia toimenpiteitä, joiden tarkoituksena on vähentää sekä hyödynnettävien virheiden todennäköisyyttä että mahdollisen hyväksikäytön seurauksia, mikäli sellaisia ilmenee. Eräät näistä virheistä on todettu siitäkin huolimatta, että OpenSSH:n historia käsittää useita manuaalisesti suoritettuja tietoturva-auditointeja (Miller, 2007.) Järjestelmänvalvojan näkökulmasta Hoskins (2006) näkeekin tietoturvaan liittyvässä tutkimuksessaan OpenSSH:n olemassaolon siten, että mikäli vastuulla on yksi tai useampia Linux- tai Unix-järjestelmiä, on OpenSSH todennäköisesti yksi tärkeimmistä ”työkaluista” järjestelmänvalvojan omassa ”työkalupakissa”.



Kuva 5. Avoin pääteistunto OpenSSH v5.3p1 käyttämisen mahdollistajana.

OpenSSH:ta voidaan käyttää interaktiivisesti (ks. kuva 5) tai skripteihin upottamalla tarjoamaan turvallista etäkäyttöä tai tiedostojen siirtoja tietojärjestelmien välillä (Allen 2003; Bauer 2001b). On kuitenkin olennaista tiedostaa protokollan käyttöön liityviä riskejä ja vaaroja. Osittain tämän vuoksi OpenSSH:n avulla onkin vaarallisen helppoa mahdollistaa huolimattomasti luotuja tietoliikenneyhteyksiin liittyviä luottamussuhteita tai huolimattoman luottamuksen suhteita. Pahimmillaan käyttäjä jättää itsensä avoimeksi nykyään hyvin yleisille automatisoiduille hyökkäyksille, joten on olennaista tiukentaa ja huomioida pääsynvalvontaa ja käytönvalvontaa eliminoimalla huolimattomasti syntyneet luottamussuhteet. (Hoskins 2006; Venkatachalam 2007.) Myös päivitysten muutostiedostojen pitäminen ajan tasalla on osa kriittisen järjestelmän tietoturvaa (Miller, 2007). Tietojärjestelmän kokonaisvaltainen ja ajantasainen tietoturva edellyttää korostuneen holistista lähestymistapaa, joten pysyäkseen ajan tasalla päivitysten suhteen tulee järjestelmän vastuuhenkilön olla tietoinen päivitysten julkistamisesta (Hoskins, 2006). Mikäli OpenSSH käännetään (build) lähdekoodeista, on syytä liittyä OpenSSH -postituslistalle jotta mahdollisten päivitysten seuraaminen olisi mahdollista (Hoskins 2006; Miller 2007).

6http://www.mindrot.org/~djm/

Hoskinsin (2006) artikkelissaan esittämät menetelmät muodostavat ja luovat perustan modernissa kompleksisessa Linux-järjestelmässä olevan OpenSSH-protokollan turvaamiseksi. Kun järjestelmän vastuuhenkilö uskoo turvanneensa oman järjestelmänsä, on syytä käyttää esim. Nessus Vulnerability Scanner -skannaustyökalua ja esimerkiksi mahdollisten kollegojen vertaisarviointia (peer review) riittävän laadukkaan lopputuloksen tarkistamiseksi ja toteamiseksi (Hoskins, 2006; Botta, Werlinger, Gagné, Beznosov, Iverson, Fels & Fisher 2007). Vahvan kryptografian modernilla aikakaudella olisi epäintuitiivista käyttää vaihtoehtoista, matalan turvallisuustason protokollaa kun on mahdollista parantaa SSH:n turvallisuutta matalin kustannuksin (Bellare, Kohno & Namprempre 2002; Hoskins 2006). Toisaalta Albrect kumppaneineen (Albrecht, Paterson & Watson, 2009) on pohtinut kysymystä, mistä ja miten tiedämme, että muutoksen tekeminen todella korjaa tai parantaa SSH:n suojausta, turvallisuutta tai tietoturvaa. Siihen kuuluvaa SSH-asiakasohjelmaa tai SSH-pääteohjelmaa käytetään työkaluna salatun tietoliikenneyhteyden, tässä tapauksessa telnetin kaltaisen etäyhteyden muodostamiseksi SSH-palvelimelle, tarkoituksena päästä käyttämään toista tietokonetta merkkipohjaisen konsolin kautta. Iyappanin ja kumppanien (Iyappan, Arvind, Geetha & Vanitha, 2009) tutkimuksesta käy ilmi, että SSL/TLS:n ja IPsecin rinnalla SSH on yksi laajimmin käytetyistä tietoturvaprotokollaohjelmistoista Albrect kumppaneineen (Albrecht et al. 2009) puolestaan on sitä mieltä, että eräs monista SSH:n eduista on, että se tarjoaa täysin salatun protokollan tiedonsiirtoon.

Millerin (2007) mukaan tällainen toiminnallisuus ja haavoittuvuuksien esiintymien ohjelman käytölle olennaisissa kirjastoissa ovat aiheuttaneet kehittäjille tarpeen toteuttaa useita ennakoivia toimenpiteitä, joilla pyritään vähentämään hyödynnettävissä olevien virheiden todennäköisyyttä tekemällä hyökkäävän osapuolen toiminta vaikeaksi ja rajoittamalla onnistuneen hyödyntämisen seurauksia. Tällaisia toimenpiteitä ovat vaarallisten API:ien korvaaminen, kompleksisen tai virhealttiin koodin välttäminen käytettävässä kirjastossa, palvelinten käyttöoikeuksien erottaminen toisistaan, protokollamuutokset eliminoimaan esiautentikoinnin (pre-authentication) kompleksisuus ja mekanismi käyttöjärjestelmän tarjoamien hyökkäyksiä lieventävien toimenpiteiden hyötyjen maksimoimiseksi. OpenSSH-ohjelmistossa on (Hoskins, 2006) runsaasti asetuksia, joita voidaan käyttää tehokkaasti mahdollisten riskien eliminointiin tai riskirajan madaltamiseen. Useimmat Linux-jakelujen asetukset on säädetty vastaamaan käyttäjän kannalta neutraaleja asetuksia, mutta joissakin tapauksissa näillä asetuksilla on mahdollista korottaa järjestelmän turvallisuustasoa. OpenSSH (CERT-FI, 2011) on hyvä esimerkki erittäin tärkeästä avoimen lähdekoodin ohjelmistosta, jossa tietoturva tulee ottaa erityisesti huomioon, ja siten se sopi tämän tutkimuksen keskeiseksi tutkimuskohteeksi.

Kansallinen tietoturvaviranomainen CERT-FI on havainnut haavoittuvuuksia lähes kaikissa OpenSSH–tuoteperheen osa-alueissa, ja näistä huomioista ja havainnoista CERT-FI (2011) tiedottaa sivustollaan julkaisemissaan haavoittuvuustiedotteissa. OpenSSH tuoteperheen haavoittuvuuksia on havaittu palvelimissa, palvelinsovelluksissa, työasemissa, loppukäyttäjäsovelluksissa ja verkon aktiivilaitteissa. Hyökkäys voi olla sekä paikallista että tapahtua etäkäytön kautta. Hyväksikäyttö kohdistuu yleensä käyttövaltuuksien laajentamiseen, komentojen mielivaltaiseen suorittamiseen, palvelunestohyökkäyksiin tai luottamuksellisen tiedon hankkimiseen.

Eräs CERT-FI:n vuonna 2002 julkaisema varoitus koski vahingollista ohjelmakoodia sisältävää OpenSSH-jakelupakettia. CERT-FI -varoituksen mukaan eräällä ftp-sivustoilla olleita OpenSSH-jakelupakettien sisältöön oli tehty muutoksia. Toimenpiteen

seurauksena ohjelmapaketteihin oli lisätty vahingollista ohjelmakoodia, joka ajettiin ohjelmaa käännettäessä. Toimenpiteen seurauksena muodostui tietoliikenneyhteys koodiin sisällytetyn ip-osoitteen porttiin muodostaen hyökkäämisen mahdollistavan takaportin, jonka kautta pystyi hallitsemaan kohdejärjestelmää. Ratkaisuna, yleisen toimintamallin mukaisesti, CERT-FI yhdessä OpenSSH:n kanssa tarjosivat joko korjaavan ohjelmistopäivityksen tai ongelman rajoittamisen. Tietoturvaan painottuneista foorumeista voidaan puhua areenoina, joilla käyttäjät keskenään käsittelevät tässä tapauksessa avoimen lähdekoodin tietoturvaa (CERT-FI, 2002a.)

OpenSSH-ohjelmiston ylläpito ja kehitys tapahtuvat kolmessa osoitteessa, jotka ovat bugzilla.mindrot.org, www.openssh.org ja lists.mindrot.org/mailman/listinfo/openssh-unix-dev. Kyseessä on OpenBSD-projektin kehittämä tuote (OpenBSD, 2013). Toisen OpenBSD-tiimin vastuu on tuottaa mahdollisimman puhdasta, yksinkertaista ja turvallista koodia. Tällainen mahdollistaa laadultaan paremman koodin kontrollin ja koodin helpomman katselmoinnin. Toinen tiimi puolestaan muodostaa puhtaasta ohjelmistoversiosta portable-muotoisen version (ns. -p -julkaisu, jollainen on esim. ”OpenSSH 4.0p1”), joka tulisi toimimaan useimmissa erityyppisissä tietojärjestelmissä (OpenSSH, 2011.) Projektin tavoite on yksinkertainen: siinä missä telnetin ja rloginin tietoturva on jo muinoin kyseenalaistettu, voidaan SSH-protokollasta kaikkien tietojärjestelmien osalta puhua nykystandardin mukaisena, tietoturvallisena vaihtoehtona. SSH-protokolla on saatavilla kahtena eri vaihtoehtona: SSH1 ja SSH2. Vanhempi, SSH1-protokolla käsittää kaksi OpenSSH:n tukemaa alemman tason varianttia (protokollat 1.3 ja 1.5). Toinen vaihtoehto, SSH2 on lähtökohtaisesti kehitetty välttämään ja kiertämään RSA:n julkaiseman, nykyään jo rauenneen patentin rajoituksia ja ehtoja. Se myös toisaalta korjaa useita teknisiä ongelmia kuten SSH1-protokollaan vaivanneen CRC-datan eheysongelman. Lähes välittömästi SSH1-protokollan julkaisun myötä useat ei-OpenBSD-ryhmät kiinnostuivat uudesta SSH1-tuotteesta. Damien Miller, Philip Hands ja monet muut, jotka edelleenkin toimivat olennaisissa rooleissa muutostiedostojen hyväksymisprosessissa, aloittivat välittömästi OpenSSH-tuotteen sovittamisen (porttaamisen) Linuxiin ja monin muihin Unixin kaltaisiin tuotteisiin. Alusta alkaen oli projektin kehittäjille muodostunut selkeä käsitys, että jopa alkuperäinen SSH:n toteutus oli liian monimutkainen ja sillä oli liikaa käyttöjärjestelmäriippuvuuksia hoidettavanaan. OpenSSH:n tavoitteena ja lähestymistapana oli ja on tuottaa täysin tietoturvallista ja kivikovaa koodia, jotta tämän kaltaiset eroavuudet voitaisiin välttää. Jotta tämän kaltainen kehitysprosessi olisi helppoa kaikille halukkaille, hyväksi havaittu menetelmä onkin OpenSSH-projektissa ollut pilkkoa tuotekehitysprosessia siten, että siirrettävyyden kehittäminen eriytetään ytimen kehittämisestä (OpenBSD, 2013.)

OpenBSD:n kotisivuilla (OpenBSD, 2013) löytyy listaus, jossa ilmoitetaan sekä release errata (julkaisun virheet) sekä lista muutostiedostoista (patch list). Kyseiseltä sivulta löytyy myös patch branch, joka on lähdekoodeista muodostuva puumainen rakenne (source tree), joka käsittää sekä ohjelmiston kannalta tärkeät muutostiedostot (important patches) että korjaukset (fixes). Käyttäjällä on mahdollisuus valita kolmesta vaihtoehdosta. Hän voi joko ladata tuoreimman julkaisun ja lisätä tarvittavat korjaukset käsin, käyttää patch branch'ia, josta nämä muutostiedostot löytyvät tai käyttää kaikkia tuoreimpia ominaisuuksia sisältävää tuoreinta lähdettä. Tämän vuoksi yleinen käytäntö on, että kaikki errata-merkinnät lisätään muutostiedostojen lopulliseen sijoituspaikkaan (patch branch) 48 tunnin kuluessa erratan julkaisusta. Tämän jälkeen julkaistavat muutostiedostot on mahdollista liittää muutostiedostoille varattuun paikkaan, mutta niihin liittyy useita ehtoja. Muutostiedoston tai useamman muutostiedoston tulee olla yksinkertaisia, lyhyitä ja selkeästi 100 % virheettömiä. Errata-merkinnät tehdään

ohjelmistovirheistä, joilla on merkitystä monille ihmisille. Muut muutostiedostot on mahdollista sulauttaa ns. muutostiedostoja varten kehitettyyn paikkahaaraan (patch branch), mikäli niillä on olennainen vaikutus pienelle käyttäjäryhmälle.

Suuria osajärjestelmiä tai muutostiedostoja ei kyseisessä projektissa suositella sulautettavaksi osaksi patch branch'ia. Tämä johtuu osaltaan resurssienhallinnasta ja muutostiedostoihin liittyvistä ylläpitovaikeuksista (patch tree). Olennaisempaa olisikin kohdistaa resurssit tuleviin julkaisuihin (release), jotta niistä saataisiin entistä parempia. Uusia tai muuttuneita toiminnallisuuksia, laitteiston tukea (hardware support) tai ohjelmointirajapintoihin (API) liittyvää aineistoa ei tämän vuoksi haluta sulauttaa patch branchiin, ja mikäli toiminta vaatisi muutoksia manuaalisivuille (man pages), ei ehdotusta tulla huomioimaan muutoskandidaattina patch branchiin. Poikkeuksena kaikkiin muihin ehtoihin, OpenSSH-ohjelmistojukaisut tullaan sulauttamaan osaksi patch branchia (OpenBSD, 2013.)

OpenSSH:n kuten myös OpenBSD:n virheiden, bugien raportointiin käytetään Bugzilla-raportointityökalua. Kyseessä on web-pohjainen avoin (avoimeen lähdekoodiin) perustuva (Bugzilla, 2012) ohjelma, jonka useat avoimeen lähdekoodin kuin myös kaupalliseen suljettuun koodiin perustuvat projektit ovat valinneet ja käyttävät raportointityökalunaan. Bugzilla kautta kuka tahansa voi raportoida virheistä.

OpenBSD uskoo vahvaan suojaukseen, ja heidän pyrkimyksensä on olla tietoturvateollisuuden ykkönen (OpenBSD, 2013). Omien väittämiensä mukaisesti he voivat mahdollistaa muutoksia joihin muilla tahoilla ei ole mahdollisuutta, ja tämän lisäksi heidän kryptografisen osaamisensa ansiosta heillä on mahdollisuus käyttää kryptografisia lähestymistapoja tietoturvaongelmien korjaamisessa (OpenBSD, 2012). Tämän vuoksi openssh.org-projektissa olennaisessa roolissa onkin sähköposteihin perustuva, postituslistapalvelimelle asennettu postituslista. Toisaalta välitetyt viestit voidaan koostaa kyseiselle postituslistalle kaikkien asianosaisten nähtäville. Koska kyseessä on tietoturvaohjelmisto, tietoturva-aiheinen keskustelu on ainoastaan OpenSSH-kehittäjien luettavissa.

Vapaan ja avoimen lähdekoodin ohjelmistojen suosio on Asundin ja Jayantin (2007) tutkimuksen mukaan saanut alan akateemisten asiantuntijoiden ja tutkijoiden keskuudessa huomattavasti huomiota. Vaikkakin useimmat tutkijoista ovat kvalitatiivisesti kuvailleet useita OSS-prosesseja, on olemassa muutamia kvantitatiiviseen tutkimukseen tai validointiin perustuvia tutkimuksia, joissa tällaisia vapaan ja avoimen lähdekoodin prosesseja on tutkittu.

Suomenkielisillä Linux-sivuilla (Linux.fi, 2011) luottamusta ja luottamuksellisuutta korostettiin riippumattomien asiantuntijoiden päästessä vapaasti tutkimaan ohjelman toimintaa ja kertomaan löydöksistään. Bugzilla mahdollistikin ohjelmakoodin kehittäjille avoimena virheiden raportointiympäristönä (bugtracker) sen, ettei ohjelman tai ohjelmistojen käyttäjien tarvinnut luottaa pelkästään ohjelman tekijän sanaan (Linux.fi, 2011). Aineiston löydökset olivat tyypillisten OpenSSH-käyttäjien havaintoja, joista ohjelmiston muita käyttäjiä ja kehittäjiä haluttiin tiedottaa. Raportoidun ongelman luonne ja vakavuus toimivat kriteereinä sille, johtiko Bugzilla-järjestelmään toimitettu tiedonanto lopulliseen koodin lisättäväksi kokonaiseksi muutostiedostoksi vai muodostuiko se osaksi suurempaa muutostiedostoa tai kokonaisuutta.

Bugzilla-järjestelmässä tapahtuva muutostiedostojen elinkaari noudatteli hyvin yhtenäistä linjaa, jossa tiketin tai raportin lisääjä voi olla kuka tahansa OpenSSH-ohjelmiston toimintaan ja Bugzillan toimintaperiaatteeseen ja tietoturvaan perehtynyt

henkilö. Järjestelmään syötetty muutostiedosto ja muutostiedostoon liittynyt tiketti oli tutkimusaineiston perustella muotoiltu hyvin usein kysymyksen muotoon, joten suoran väittämän tai teesin muotoon asetettuja tikettejä oli tutkittavassa aineistossa vain muutamia. Sekä hakusanoihin pohjautuvan analyysin että sisällönanalyysin perusteella saatujen havaintojen luonteesta ja esitystavasta ei suoranaisesti voinut päätellä, oliko kyseessä avoimeen vai suljettuun lähdekoodiin liittyvä raportti tai tiketti. Ainoastaan OpenSSH-ohjelmiston luonne antoi ymmärtää kyseessä olevan avoimen lähdekoodin ohjelmiston. Tästä voitiin vetää johtopäätös, että avoimen lähdekoodin ohjelmistokehityksen yhteydessä tapahtunut, tikettiin liittyvien tahojen välinen kommunikaatio oli samankaltainen suljettujen ohjelmistojen kehitystyössä tapahtuvan kommunikaation kanssa. Olennaisimmaksi eroavaisuudeksi voitiin todeta suljetun lähdekoodin kommunikaation, joka useinkaan ei ole julkista.

Tyypillisimmillään osallistuminen Bugzillassa olevien, hyväksytyn OpenSSH-muutostiedoston sisältävän tiketin elinkaaren vaiheisiin näkyi c-kielisen lähdekoodiin ohjelmistovirheiden korjaamisessa ja uusien ominaisuuksien lisäämisessä ja tuen jakamisessa käyttäjille. Bugzillassa olevien OpenSSH-tikettien reagointiajat olivat keskimäärin muutostiedostoon liittyvien dialogeissa olevien aikaleimojen perusteella hyvin ajantasaisia, eikä vastausten välillä ollut merkittävää latenssia, viiveitä tai taukoja. Tämä kertoi kompetensseiltaan hyvin monentasoisia kehittäjiä yhdistävästä asenteesta ja intohimosta aiheeseen.

Tutkimus kohdistui avoimeen lähdekoodiin perustuvaan yleiskäyttöiseen tietokoneohjelmien virheiden raportointityökaluun, ja erityisesti järjestelmään syötettyjen muutostiedostojen ja niihin liittyvän, tietoturvaan painottuvan keskustelun analysointiin. Tutkimuksen kannalta olennaista oli aineiston määrittely sekä ajallisesti että koollisesti, mutta kuitenkin siten, että analysoitava aineisto olisi riittävän laaja ja edustaisi tarpeeksi kattavasti muutostiedostoja ja niiden ympärillä käytävää keskustelua.

5. Analyysi

5.1 Aineisto

Pro gradu -tutkimuksen tutkimusdata kerättiin analysoimalla OpenSSH-ohjelmiston kehitysympäristössä käytyjä, käyttäjien ja kehittäjien välisiä keskusteluja, ja tutkimalla heidän tapaansa käsitellä tietoturvaa. Tutkimuksen kannalta olennaista oli karsia tutkimusdatasta tutkimuksen ja analyysin kannalta epäolennainen aines. Tämä tehtiin sekä käyttämällä Bugzillan tarjoamaa hakuominaisuutta että käymällä analysoitava aineisto läpi manuaalisesti. Tutkittava aineisto rajattiin koskemaan tuotetta OpenSSH ja vuotta 2012 eli tapauksia, joiden luontipäivämäärä on välillä 1.1.2012 ja 31.12.2012.

Tutkimuksen kannalta oli olennaista että haku kohdistui OpenSSH-tuotteen kaikkiin komponentteihin ja että tutkittavien tapausten status oli RESOLVED, VERIFIED tai CLOSED. Tähän käytettiin Bugzilla-järjestelmän mahdollistamaa hakutoimintoa. Tutkittavista Bugzilla-kehitysympäristössä olevista, OpenSSH-ohjelmistoa koskevista tapauksista valittiin ainoastaan muutostiedoston sisältävät tapahtumat.

Hakuprosessin tuloksena löytyi 54 OpenSSH:n muutostiedostoihin liittyvää tapausta. Osa aineistossa olevien muutostiedostojen hyväksymisestä siirtyi vuodelle 2013 kohdistuen OpenSSH-ohjelmiston versioon 6.2 (OpenSSH-6.2 release[7]), joten tämä toimi myös aineistoa rajaavana kriteerinä. Muutamien tikettien osalta kriteerinä rajaamiselle oli tiketin elinkaari, joka jatkui vuodelle 2013. Kaikki tiketit eivät sisältäneet muutostiedostoa, joka myös toimi aineiston rajausperusteena. Saatu data analysoitiin sekä hakusanapohjaisesti että sisällönanalyysin kautta, jotka olivat tutkimuksen kannalta sopivimmat vaihtoehdot. Tutkimusaineistoon kohdistuneet haut tehtiin sekä koneellisesti (mekaaninen hakuprosessi) että analysoimalla aineiston sisältö käyttäen analyysimenetelmänä sisällönanalyysia (subjektiivinen, laadullinen analyysi).

Analyyseissä käytettynä viitekehyksenä hyödynnettiin kirjallisuuskatsausta (review), joka muodostettiin etsimällä ja tutkimalla tietoturvaan keskeisesti liittyviä käsitteitä, hakusanoja ja niiden eri muotoja. Tähän tutkimukseen otettiin mukaan yhdeksän tietoturvaan keskeisesti liittyvää käsitettä. Viitekehyksen muodostamiseen käytettiin Nelli-portaalista, Google Scholarista ja tietokannoista löytyneitä tietellisiä artikkeleita.

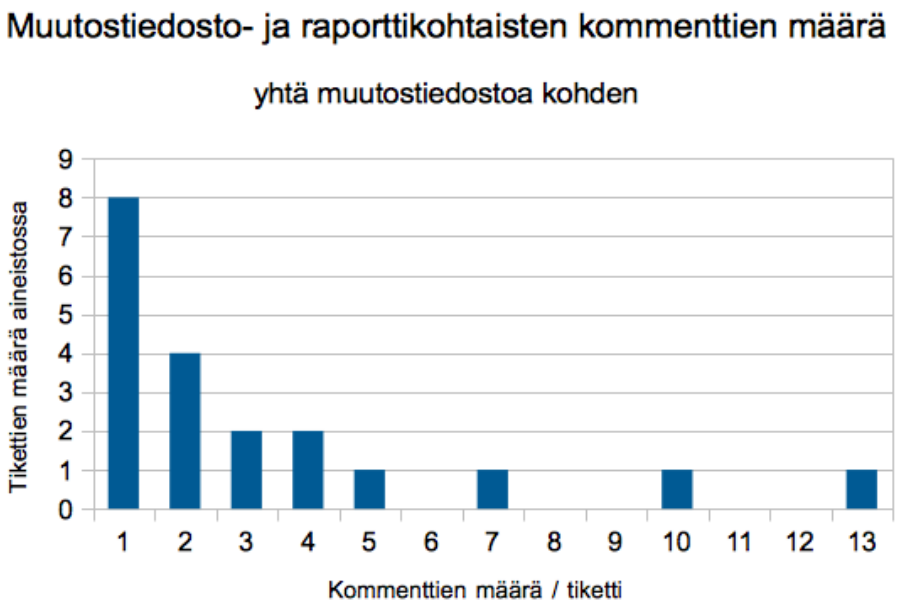
Tutkimusaineiston työstämisessä kuten tutkittavan tiedon jalostamisessa olennaisessa roolissa olivat aputaulukot, joihin kerättiin tutkimuksen kannalta olennainen tieto. Tutkimuksen ensimmäinen vaihe oli etsiä ja kerätä aineisto, joka kokonaisuudessaan löytyi Internetistä.

Osa raportoiduista havainnoista, korjaus- tai muutosehdotuksista tai virheraporteista ei päätynyt valmiiksi muutostiedostoksi tai lopulliseen versioon lisättäväksi koodiksi asti. Tämä tutkimus kohdistui tiketteihin, joissa oli lopulliseen julkaisuun ohjelmakoodiksi päätynyt muutostiedosto.

7http://www.openssh.org/txt/release-6.2

Tutkimuksen kannalta oli olennaista kiinnittää huomiota tiketin elinkaaren aikana käytyyn keskusteluun (ks. taulukko 1) ja erityisesti analysoida keskusteluissa käytyä tietoturvaa ja sen määrää. Tikettikohtaisten kommenttien (ks. taulukko 1) määrä vaihteli välillä yhden (1) ja kolmentoista (13) välillä. Tyypillisesti tiketin asettajan luoman tiketin ja tiketin kuvauksen lisäksi tiketin elinkaareen liittyviä kommentteja oli 1-2. Vain muutamien tikettien kommenttien lukumäärä oli suurempi kuin 2.

Taulukko 1. Muutostiedosto- ja raporttikohtaisten kommenttien määrä yhtä muutostiedostoa kohden.



Tutkimuksen aineisto oli OpenSSH-ohjelmiston kehitystä varten olevan ja muutostiedostojen kehityskaaren kannalta olennaisessa Bugzilla-järjestelmässä sijainneet muutostiedostot, joista jokainen sai hyväksynnän ja pääsi lopulliseksi koodiksi asti. Analyysin kannalta olennaisen, tietoturvaan liittyvien tikettien hakuprosessin tuloksena syntynyt, hypertekstiä sisältänyt lista oli mahdollista kopioida Bugzillasta taulukkolaskentaohjelmaan. Olennaista oli myös poimia kaikesta aineistosta tutkimuksen kannalta merkitykselliset tapaukset ja erityisesti merkityksellinen sisältö.

Vuoden 2012 OpenSSH-tuotteen Bugzillassa olevien, hyväksytyn muutostiedoston sisältävien tikettien yhteenlaskettu lukumäärä oli 21. Raportti käsitti statustietojen lisäksi muutostiedostoon tyypillisesti liittyvän kehitysehdotuksen, muutosehdotuksen tai muun huomion. Bugzilla mahdollistaa liitetiedoston liittämisen raporttiin. Liitetiedoston nimeämiskäytäntö noudatteli hyvin tarkasti muutostiedoston ja siihen liittyvän raportin teemaa. Liitetiedoston nimessä oli usein tarkenteena ”patch”, joka on muutostiedoston yleisesti käytössä oleva englanninkielinen nimitys. Vuoden 2012 raporteissa liitetiedosto oli yksinomaan valmis, diff-työkalulla muodostettu muutostiedosto, jonka koko pienimmillään oli 596 tavua (bytes) ja suurimmillaan 16,88 kilotavua (KB).

Bugzillan OpenSSH-muutostiedostoon liittyvässä keskustelussa useimmat dialogiin kantaa ottaneet tahot olivat kokeneet hyvin samankaltaisen ongelmatilanteen hyvin samantyyppisen laitekokoonpanon kanssa. Käytetyt ohjelmat ja erityisesti OpenSSH olivat yleisesti tunnettuja myös toimintaperiaatteeltaan ja forumilla kirjoitteluun osallistuvat osapuolet puhuivat niistä tunnistettavilla ja yleisesti käytetyillä nimillä,

joten tarkastelun kohteena olleen tietojärjestelmän, Bugzillan OpenSSH-ohjelmiston muutostiedostoihin kohdistuneissa dialogeissa käsiteltiin OpenSSH-ohjelmistoa ja ohjelmiston käyttöön liittyneitä ongelmia, havaintoja, kehitysehdotuksia, uusia ideoita tai raportteja virheistä. Olennaisia havaintoja olivat myös muutostiedoston toimintaan tai ominaisuuksiin liittyneet tiedustelut.

Koska kyseessä oli OpenSSH-ohjelmistoon ja ohjelmiston muutostiedostojen ympärille rakentunut dialogipohjainen foorumin kaltainen järjestelmä, oli ohjelmiston muutostiedostoihin liittynyt tekninen toteutus olennaisessa roolissa ja sen dialogien jokainen osapuoli tiedosti. Yleisellä tasolla voitiin todeta, että Bugzillan tiketit muodostuivat yksinkertaisimmillaan yhdestä kysymyksestä (description) kommentista, kehitysehdotuksesta ja yhdestä tai useammasta vastauksesta (comment).

Tutkija yritti liittyä OpenSSH-ohjelmiston kehittäjien suljetulle sähköpostilstalle, mutta ulkopuolisen tahon liittyminen (subscribe) kehittäjille tarkoitetulle postituslistalle estettiin pääkäyttäjän (Darren Tucker) toimesta:

Postituslistalle (openssh@openssh.com) tutkijan lähettämä liittymispyyntö:

> *”Please could you subscribe me to this list openssh@openssh.com”.*

Pääkäyttäjän vastaus liittymispyyntöön:

> *“Sorry but no, it's a private list for the developers only.”.*

OpenSSH-ohjelmistoprojektin postituslista openssh@openssh.com oli suljettu foorumi (Developers private list, read-only), jossa openssh.org -sivuston ohjeistuksen perustella käsiteltiin OpenSSH-ohjelmiston tietoturvalliseksi luokiteltavaa ohjelmistokehitystä. Postituslistan ohjeistuksessa ohjeistettiin käyttämään vaihtoehtoista www.openssh.org -sivustolla mainittua postituslistaa.

Postituslistan ohjeistuksessa mainittiin, että ainoastaan OpenSSH-ohjelmiston kehittäjät voivat seurata (read-only) suljetulla foorumilla käsiteltäviä asioita lähettämällä tietoturvaan liittyvät kysymykset osoitteeseen openssh@openssh.com.

Ohjeistuksessa ei ollut mainintaa muutostiedostoista. Tämän listan suhdetta tai merkitystä Bugzillan OpenSSH-kehitysympäristöön ja siellä käsiteltyihin hyväksyttyjen muutostiedostojen tietoturvaan ei tässä tutkimuksessa pystytty osoittamaan tai todistamaan.

5.2 Analyysin kuvaus

Tutkimusaineisto oli täysin tekstipohjainen ja sisälsi ainoastaan ASCII-merkkejä (American Standard Code for Information Interchange) koostuen ainoastaan kirjain-merkeistä. Sekä Bugzillan että OpenSSH:n muutostiedostojen kieli oli englanti.

Hyvin usein tikettiin lisättiin usein käsiteltävänä olevaan muutostiedostoon liittyvää tekstimuotoista c-koodia. Aineiston merkkien kokonaismäärä oli 49061. Sanojen määrä oli 7816. Aineistossa ohjelmistotuote oli nimetty muotoon OpenSSH.

Aineiston rajaamisessa ohjelman nimessä käytettiin tiketin informaatiokentässä mainittua tarkennetta Portable. Alustatyypeiltään (platform) analysoidut muutostiedostot

kohdistuivat ja perustuivat suurimmaksi osaksi alustariippumattomaan ja prosessoririippumattomaan ohjelmakoodiin.

Alustariippumattomia (All All) muutostiedostoja oli määrällisesti 14 kpl. Neljän tiketin tarkenteena oli All Linux tai Other Linux. Yksittäisinä tarkenteina esiintyi myös All OpenBSD ja ix86 Solar.

Tutkimusaineiston muutostiedostojen versiot vaihtelivat välillä 5.9p1-6.1p1, ja ne koskivat erityisesti OpenSSH:n versiota 5.9p1 (12 kpl), mutta aineistossa olleet muutostiedostot koskivat myös sekä versionumeroltaan suurempia että pienempiä OpenSSH:n versioita: OpenSSH 6.1p1 (1 kpl) ja OpenSSH 5.9p1 (6 kpl). Kahden muutostiedoston versiomerkintä oli muodossa -current (2 kpl), jonka perusteella ohjelmakoodin voi olettaa toimivan senhetkisen ohjelmaversion kanssa. Muutostiedostoihin liittyviin tiketteihin oli mahdollista määritellä erityyppisiä, muutostiedostojen ja raporttien kannalta merkityksellisiä kenttiä ja niiden sisältöjä. Eräs muutostiedostojen kannalta olennainen kenttä oli muutoksiin liittyvien raporttien tärkeysaste (importance). Vuoden 2012 aineistossa se oli minor, normal, major, critical, trivial ja enhancement. Myös prioriteetin määrittäminen oli mahdollista, ja vuoden 2012 aineistossa sen vaihteluväli oli välillä P2 (16 kpl) ja P5 (4 kpl).

Raporttiin (ks. kuva 6) oli mahdollista statustiedon ja resoluutiotiedon lisäksi kirjata raportin tai muutostiedoston kannalta olennaista tietoa ja yksityiskohtia, joiden voitiin olettaa olevan ratkaisun kannalta merkityksellistä tietoa.

Kuva 6. Bugzillassa olevan OpenSSH-tiketin statustiedot (https://bugzilla.mindrot.org/show_bug.cgi?id=2018).

Aineiston tikettien statukset olivat suurimmaksi osaksi muotoa ”RESOLVED FIXED” (16 kpl).

Kahden tiketin statusarvo oli ”RESOLVED WONTFIX” (2 kpl). Yhden tiketin status oli ”RESOLVED WORKSFORME”. Yhden tiketin status oli ”RESOLVED DUPLICATE of bug 1765”.

Tällä tarkoitettiin kyseisten muutostiedostojen tai muutostiedostossa mainitun ominaisuuden tulkitsemista kaksoiskappaleeksi jo olemassa olevan muutostiedoston tai muutostiedostossa olevan ominaisuuden kanssa, mutta kyseinen tiketti oli silti ratkaistu onnistuneesti.

Aineiston muutostiedostojen dialogissa olevien kommenttien (dialogin aloittaja mukaan lukien) lukumäärä oli 83, (vaihteluvälin ollessa 2-14). Tutkimusaineiston hyväksyttyjen muutostiedostojen statukset jakautuivat kolmeen (ks. taulukko 2) luokkaan.

Taulukko 2. Vuoden 2012 hyväksytyiksi päätyneiden muutostiedostojen statukset.

Status	lkm
RESOLVED FIXED	16
RESOLVED WONTFIX	2
RESOLVED DUPLICATE	1

Hakusanoihin perustuva analyysi

Ensimmäisessä tutkimuksessa käytiin läpi tietoturvaan keskeisesti liittyä hakusanoja apuna käyttäen.

Toisessa tutkimuksessa tutkittava aineisto käytiin läpi sisältöä analysoimalla.

Vuoden 2012 aineisto tuotti kirjallisiin lähteisiin perustuneessa viitekehyksessä mainittujen hakusanojen perusteella kaksi osumaa, ”verification” ja ”authentication”, joiden perusteella kyseessä oli OpenSSH -tuotteeseen kohdistuneet, tietoturvaa käsittävät tapaukset (tiketit 1987 ja 2027).

Hakusanat liittyivät sekä tietojärjestelmän implementointiin että ylläpitoon.

Sisällönanalyysi

Analyysin toisessa vaiheessa jokainen Bugzillaan lisätty vuoden 2012 muutostiedostoksi hyväksytty tapaus analysoitiin sisältöanalyysin avulla mahdollisten tietoturvaan liittyvien tai tietoturvaa käsittelevien dialogien löytämiseksi.

Näin löytyi kuusi tapausta, joissa on viitteitä OpenSSH-tuotteen tietoturvaan. Viitteet tietoturvasta koskivat joko tiketin otsikkoa tai muutostiedoston elinkaaren aikana käytyä dialogia. Tiketin pituus tai dialogissa olevien kommenttien lukumäärä ei ole verrannollinen tiketissä käsitellyn tietoturvan suhteen.

Analysoitavan aineiston rajaaminen

Vuoden 2012 aineisto koostui kokonaisuudessaan 54 muutostiedostoa koskeneesta tiketistä, josta tutkimuksen lähdeaineistoksi rajattiin yhteensä 19 Bugzillassa olevaa, hyväksytyn muutostiedoston sisältänyttä ja siten lopulliseen julkaisuun hyväksyttyä muutostiedostoa.

Analyysia varten nämä tapaukset koostettiin yhteen taulukkoon.

Analyysi tehtiin käyttäen seuraavia hakusanoja (ks. taulukko 3), jotka arvioitiin keskeisten tietoturvatermien joukkoon.

Taulukko 3. Hakusanaan perustuneen analyysin hakusanat.

Hakusanaan perustuvassa analyysissä käytetyt hakusanat	**Termin englanninkielinen muoto**	**lkm**
Luottamuksellisuus	*confidentiality*	0
Eheys	*integrity*	0
Saatavuus	*availability*	0
Pääsynvalvonta	*access control*	0
Osapuolten todentaminen	*verification*	3
Tunnistaminen	*identification*	0
Palvelunestohyökkäykset	*Denial-of-Service*	0
Koodin injektointi	*code injection*	0
Puskurin ylivuoto	*buffer overflow*	0

Olennaista oli aineiston rajaaminen koskemaan ainoastaan lopulliseen ohjelmakoodiin päätyneitä muutostiedostoja. Rajaus tehtiin analysoimalla jokaiseen muutostiedostoon liittyvä dialogi.

Tässä tutkimuksessa haettiin tietoturvaan liittyvän termistön esiintymistiheyttä tutkittavissa dokumenteissa.

Ensimmäisessä vaiheessa tutkittava aineisto käytiin pinnallisesti läpi. Toisessa vaiheessa aputaulukoita apuna käyttäen muodostettiin tutkittavista tiketeistä lopullinen tutkittava aineisto tarkempaa analyysia varten.

5.3 Löydökset

Muutostiedoston lopullisesta hyväksymisestä OpenSSH-tuotteen kehitystyössä vastaavat OpenSSH-tuotteen kehittäjät Damien Miller ja Darren Tucker. Bugzillassa OpenSSH-ohjelmistokehityksestä kommunikoineet tahot olivat profiloituneet muutamaa ammattimaista kehittäjää lukuun ottamatta ohjelmointitaitoisiksi käyttäjiksi.

Dialogien keskustelu on sävyltään vapaamuotoista mutta asiallista. Kaikkien analysoitujen tikettien kaikki kommentit olivat luonteeltaan informatiivisia ja liittyivät tiketissä käsiteltyyn aiheeseen.

Usein kysymyksen lähtökohtana oli OpenSSH-ohjelmiston muutostiedosto. Mielikuvat olivat avoimia, kysymyksiin tulleet vastaukset olivat informatiivisia. Vastauksen takana oleva käyttäjä oli selkeästi valveutunut, kokenut ja uskalsi esittää näkökantansa aiheen tiimoilta.

Muutostiedostoihin liittyvä keskustelu oli luonteeltaan ja asenteeltaan hyvin avointa ja sävyltään erittäin avuliasta ja kiinnostavaa. Myös tahojen välinen vuorovaikutteisuus näkyi tikettien dialogeissa.

Esim. tiketin 2025 kommentissa 3 todetaan:

> ”*Wow, that was quick. Thanks!*”.

Esim. tiketin 2001 kommentissa 2 todetaan:

> ”*Thank you :-)*”.

Käyttäjät ja kehittäjät tiedostivat tietoturvan merkityksen avoimessa lähdekoodissa, sortumatta kuitenkaan liikaan mainostamiseen. Asenne oli positiivinen, tietoa ei pantattu eikä disinformaatioon sorruttu. Koodia kehittäneet ohjelmistokehittäjät tarjosivat jopa useita vaihtoehtoja, jotka usein edustivat kyseisen tahon suositusta. Näistä vaihtoehdoista valinnan vapaus jäi muutostiedoston hyväksyjälle.

Aineiston otanta, 54 tikettiä

Vuoden 2012 otanta käsitti yhteensä 54 tikettiä, joista 19 sisälsi hyväksytyn muutostiedoston. Näistä tiketeistä löytyi kuusi tikettiä (tiketit 1978, 2002, 2003, 2022, 2023 ja 2027), jotka viittaavat tietoturvaan. Näissä tiketeissä käsiteltiin salausalgoritmia (1978), muistin ylivuotoa (2002 ja 2003), verkkoasetusten yhteydessä ilmenevää virhettä (2022) ja spesifikaation muutosta (2023), (2027).

Ylivuotovirheet

Kaksi tikettiä (tiketit 2002 ja 2003) viittasivat selkeästi tyypillisiin C-ohjelmointikielen ohjelmointivirheestä johtuneeseen muistin ylivuotoon, ja tietoturvan kannalta ylivuotovirheiden hyväksikäyttö on perinteinen tapa hyökätä.

Mahdolliset tietoturvaongelmat

Mahdollisiksi tietoturvaongelmiksi luokiteltavien ylivuotojen aiheuttaja tai taustatekijät eivät käyneet ilmi tässä analyysissa. Kahden tiketin (1987 ja 2053) sisältämää muutostiedostoa ei hyväksytty. Päätös hylkäämisistä ei tapahtunut tietoturvanäkökohtien perusteella, vaikka hylkäämisperusteen saanut tiketti (1987) liittyi sähköiseen allekirjoitukseen ja siten tietoturvaan. Tämä tiketti tuotti hakusanoihin perustuvassa haussa osuman hakusanalla "verification".

Muutostiedostojen hylkäämisperusteita

Olennaisia hylkäämisperusteita olivat puutteellinen tuki (tiketti 1987), jonka muutostiedoston hyväksymisestä päättävä pääkäyttäjä kommentissaan esittää tiketin asettajalle. Tässä tapauksessa tiketin asettaja on sama taho kuin kyseisen muutostiedoston esittäjä. Toisessa tiketissä (2053) hylkäämisperuste oli pääkäyttäjän kommentissaaan esittämä ohjeistus, joka teki muutostiedostosta tarpeettoman.

Muutostiedostot, joissa suora viite tietoturvaan

Tietoturvan kannalta olennaiset tiketit liittyivät puskurin ylivuotovirheisiin (2002 ja 2003) ja salausalgoritmiin (1978, 2023). Yksi tiketti liittyi epäselviin virheilmoitusteksteihin (2027) ja tiettyjen verkkoasetusten yhteydessä ilmenneeseen OpenSSH:n yleiseen virheeseen (2022). Kolme aineiston tikettiä (tiketit 1978, 2023 ja 1987) liittyvät OpenSSH:n salaus- tai allekirjoitusteknisiin toteutuksiin, ja siten niiden voidaan katsoa liittyvän tietoturvaan muutenkin kuin pelkän SSH-yhteyden takia. Tässä analyysissa huomioitiin ainoastaan tiketit jotka sisälsivät muutostiedoston mutta joiden

suhteen ei muutostiedoston elinkaaren aikana käyty tietoturvaan liittyvää keskustelua. Tikettien pituuden vaihteluväli oli 2-14 kommenttia, tiketin asettajan kuvaus mukaan lukien. Useissa tiketeissä muutostiedoston hyväksyminen lopulliseen ohjelmaversioon tapahtui ilman osapuolten välistä keskustelua.

Tietoturva otsikkotiedoissa

Erityisesti tikettien 2002 ja 2003 kohdalla tietoturvan kannalta olennaista on molempien otsikkotiedoissa mainittu muistivuoto (memory leak), jonka tunnetuin väärinkäyttö (Wikipedia, 2013) on Denial-of-Service (DoS) eli palvelunestohyökkäys. Molemmissa muistivuotoon liittyviessä tiketeissä asettajana on sama taho (Bert Wesarg). Molempien muutostiedostojen hyväksymisen suoritti pääkäyttäjä Damien Miller.

Tiketti 2002: [mux.c] fix memory leak of control path if bind() fails

> *“Patch applied, will be in openssh-6.1. Thanks!”.*

Tiketti 2003: [mux.c] fix memory leaks when new session message is malformed

> *“patch applied – thanks.”.*

Kummankaan muutostiedoston hyväksymisprosessissa ei pääkäyttäjän hyväksyvää kommenttia lukuun ottamatta otettu kantaa tietoturvaan, vaikka kyseessä oli selkeästi OpenSSH:n tietoturvaongelman korjaavat muutostiedostot. Hyväksyntä ja kommentti muutostiedoston lisäämisestä seuraavaan versioon on rakenteeltaan lähes identtinen muiden vuoden 2012 hyväksyttyjen muutostiedostojen hyväksynnän ja hyväksyntään liittyvien kommenttien kanssa. Molempien muutostiedostojen tiketillä käyty keskustelu rakentui tiketin asettajan (description) ja muutostiedoston hyväksyneen pääkäyttäjän kommenttien varaan (comment).

Muutostiedoston dialogissa hyväksymisestä kertova kuittaus

Tyypillinen loppukäyttäjän kommentti, jolla muutostiedoston ja muutostiedoston prosessiin liittyviä osapuolia informoidaan muutostiedoston liittämisestä lopulliseen versioon on yksinkertaisimmillaan yhden yhden tai kahden virkkeen ja alle kymmenen sanan pituinen kuittaus. Esimerkki tällaisesta on tikettien 1995, 1978, 2007 ja 2013 dialogin viimeinen kommentti, jossa tietoturvaa ei maninita, mutta jossa hyväksytty muutostiedosto kuitataan liitettäväksi seuraavaan (openssh-6.1) julkaistavaan versioon.

> *"patch applied. this will be in openssh-6.1 – thanks!".*

Koska analysoidussa aineistossa kuvatun kaltainen, muutostiedoston hyväksynnästä kertova kommentti oli lähes jokaisessa tapauksessa dialogin viimeinen kommentti, voitiin sen todeta kirjoitusasun vaihteluista ja variaatioiden määrästä huolimatta välittävän perimmäisen ajatuksen kyseisen tiketin dialogiin osallistuneille tahoille.

Muutostiedoston hyväksynnän kuittaavaa pääkäyttäjän kommenttia edelsi tikettikohtaisesti käyty dialogi, joka käsiteltävästä aiheesta riippuen sisälsi prosessiin osallistuneiden tahojen välistä kommentointia tai muuta dialogi-tyyppistä vuorovaikutusta.

Kuvatun kaltainen loppukommentti toimi evidenssinä siitä, että muutostiedosto tullaan lisäämään ohjelman seuraavaan mahdolliseen julkaisuun. Otannasta voi vetää

johtopäätöksen, että tällainen menettely on OpenSSH-ohjelmiston muutostiedoston hyväksymisprosessissa hyvin yleinen ja tyypillinen käytäntö.

Muutostiedoston dialogin yksinkertaisin rakenteellinen muoto

Rakenteellaan yksinkertaisimmassa muodossaan lopulliseen ohjelmakoodin päätyneen muutostiedoston sisältänyt tiketti käsitti tiketin asettajan kuvauksen (description) lisäksi pääkäyttäjän kuittauksen (comment) hyväksynnästä. Rakenteeltaan tällaisia tikettejä löytyi 6 kpl (tiketit 2002, 2003, 2023, 1996, 2010 ja 2013), ja niistä kolmen aiheena oli tietoturva.

Muutostiedostot joissa ei käsitellä tietoturvaa

Aineistossa oli myös tikettejä ja niihin liittyviä dialogeja, joissa **ei käsitellä tietoturvaongelmia eikä niissä ole mainintaa tietoturvasta**. Tällaisia tikettejä löytyi analysoinnissa 14 kpl (tiketit 1968, 1991, 1995, 1996, 2001, 2004, 2007, 2010, 2011, 2013, 2022, 2025, 2033 ja 2053).

Kooltaan ja rakenteeltaan suurimmat muutostiedostojen dialogit

Tiketin tai muutostiedoston aiheella ei ollut vaikutusta tiketin elinkaaren aikana sen prosessointiin osallistuneiden tahojen lukumäärään. Laajimmillaan hyväksytyn muutostiedoston käsittelyyn osallistui tiketin asettajan lisäksi viisi muuta tahoa (tiketti 2011), ja tiketin dialogissa olevien kommenttien määrä oli 14 mukaan lukien tiketitin kuvaus.

Toisaalta, tiketin prosessointiin osallistuneiden tahojen lukumäärä tai tahojen käymä keskustelu ei tuonut muutosprosessiin tietoturvan kannalta olennaista lisäarvoa. Kooltaan laajoja tikettejä olivat myös tiketit 1968, 1978, 1991 ja 2022. Näiden tikettien aiheet olivat yksittäisiä, toisistaan poikkeavia. Näissä tiketeissä käsiteltiin mm. versiointia, dokumentointia, konfigurointia, yhteyden katkeamista ja ohjelmoinnin tyyliohjeita, ja ne kaikki tiketit sisälsivät muutostiedoston.

Yhteensopivuusongelma/puutteellinen tuki muutostiedoston hylkäämisperusteena

Esimerkki muutostiedoston sisältäneestä tiketistä (1987), joka ei saanut hyväksyntää eikä johtanut lopulliseen ohjelmaversioon. Tiketissä ei keskusteltu tietoturvasta, vaikka muutostiedoston aihe liittyi sähköiseen allekirjoituksen yhteensopivuusongelmaan, joka voidaan tulkita tietoturvaongelmaksi. Hylkäämisperusteena oli yhteensopivuusongelma eli puutteellinen tuki.

Tiketin asettaja ”kape” asetti otsikon muotoon ”FIPS signature verification incompatibility with openssl versions > 0.9.8q”.

> *”When building openssh with openssl library with FIPS (specifically versions newer than openssl 0.9.8q), there is an issue if FIPS mode is active for openssl. In ssh-rsa.c on line 243 RSA_public_decrypt is called, which is disallowed now in openssl (if in FIPS mode). The library requires applications to use the EVP API if running in FIPS mode so it can disallow certain cipher suites and hash algorithms that are not considered FIPS compliant. The user experience is that the scp/ssh client fails because RSA_public_decrypt just returns null if FIPS mode is active in openssl > 0.9.8q.”.*
>
> *”The reference below states that there is a patch, but I cannot find it so I am submitting my own for review.”.*

Tämä tiketti koski FIPS (Federal Information Processing Standard) -formaatissa olevan allekirjoituksen yhteensopivuutta, jonka kirjoittaja (kape) kokee ongelmaksi ja selkeäksi riskiksi:

> *"Unfortunately, this approach disables our custom RSA signature-verification code that is designed to save a substantial amount of pre-authentication attack surface from sshd".*

Raportoija oli lisännyt viestiinsä c-ohjelmointikielellä luodun liitetiedoston (suggested patch). Referenssiksi[8] on lisätty linkki mail-archive -postituslistalle, jossa samantyyppinen ongelma on ollut esillä vuonna 2011. Pääkäyttäjä Damien Miller vastaa kysymykseen, ja toteaa etteivät OpenSSH ja FIPS OpenSSL ole keskenään yhteensopivat:

> *"For this reason it is not going to be accepted for regular OpenSSH,".*

OpenSSL-muutostiedostoja ei ainakaan toistaiseksi hyväksytä OpenSSH:n muutostiedostojen yhteydessä. Hän opastaa pääkäyttäjän roolissaan ottamaan yhteyttä FIPS muutostiedostoista vastaaviin kehittäjiin tai kehittäjätahoihin:

> *"OpenSSH doesn't (yet) have support for FIPS OpenSSL. We might one day, but in the meantime you should address this to the developers of one of the FIPS patchsets".*

Pääkäyttäjän perustelu muutostiedoston tarpeettomuudesta hylkäämisperusteena

Esimerkki muutostiedoston sisältäneestä tiketistä (2053), joka ei saanut hyväksyntää eikä siten johtanut lopulliseen ohjelmaversioon. Tiketissä ei keskusteltu tietoturvasta, eikä tietoturva ollut muutostiedoston hylkäämisperuste. Stanislaw Pituchan asettama tiketti ja tiketin sisältämä muutostiedosto liittyi asennukseen liittyvän option ohittamiseen:

> *"Sometimes the login banner is annoying when the only allowed way to script access to the host is via the ssh client. The attached patch adds a "skipbanner" option, which will ignore the "input_userauth_banner" stage on connection (if set to yes).".*

Muutostiedoston hylkäämisperusteena oli pääkäyttäjän (Damien Miller) perustelu sille, ettei muutostiedoston mahdollistamalle toiminnallisuudelle ole toistaiseksi tarvetta:

> *"use "ssh -q" or LogLevel=quiet. I don't think we need an additional option just for this, sorry".*

5.4 Yhteenveto

Aineiston otannan riittävän tarkassa määrittelemisessä hyödynnettiin Bugzillan mahdollistamaa hakutoimintoa. Olennaista oli käyttää riittäviä tarkenteita ja hakukriteerejä. Aineiston ajallinen rajaaminen tehtiin koskemaan ainoastaan vuotta 2012. Eräs hakukriteereistä oli hakea Bugzillan OpenSSH:n muutostiedostoja koskeneesta aineistosta kaikki tuon ajankohdan rajoissa tapahtuneet muutostiedostot, sekä kohdistaa haku erityisesti OpenSSH-tuotteeseen. Analysoinnissa olennaista oli käyttää useita ns. aputaulukoita, joilla Bugzillasta saatua informaatiota oli mahdollista koostaa haluttuun muotoon sisällön siitä kärsimättä. Tutkimuksen ensimmäisessä vaiheessa tietoturvaa koskevien keskustelujen löytäminen tehtiin käyttäen taulukkolaskennan sallimaa hakuominaisuutta, jolla tarvittavien eksaktien, tarkkoihin hakusanoihin perustuvien löydösten hakeminen oli mahdollista. Toinen tutkimuksen

8http://www.mail-archive.com/openssl-users@openssl.org/msg63512.html

vaihe käsitti aineiston läpikäyntiä ja analysointia, eli aineistosta pyrittiin löytämään kaikki muutostiedostoihin ja erityisesti niiden tietoturvaan liittyvät keskustelut.

Tutkimusaineiston löydökset olivat tietoturvaan ja tietoturvariskeihin (1978, 2002, 2003, 2022, 2023, 2027), konfigurointiin, (2011, 2025, 2013, 2033) ja yleisesti OpenSSH:n toimintaan, dokumentaatioon ja virheilmoituksiin (1968, 1991, 1995, 1996, 2001, 2004, 2007 ja 2010) liittyviä tapauksia. Muutostiedostojen hyväksyntäprosessissa hylättyjä mutta tutkimuksessa analysoituja muutostiedostoja olivat tiketeillä 1987 ja 2053.

Tutkitussa aineistossa tapaukset (tiketti 2002 ja 2003) liittyivät olennaisesti tietoturvaan. Tapauksia analysoinut ja muutostiedostojen hyväksymisestä vastannut pääkäyttäjä ei kyseenalaistanut tai ottanut kommenteissaan kantaa muutostiedostoon tai muutostiedostoa esittäneen käyttäjän esitykseen, vaikka kyseessä oli otsikkotietojen perusteella muistivuotoon (memory leak) liittyneitä tapauksia. Näiden muutostiedostojen hyväksymisestä kertova pääkäyttäjän kommentti oli hyvin samankaltainen lähes jokaisen tässä tutkimuksessa analysoidun, hyväksytystä muutostiedostosta kertovan kommentin kanssa. Pääkäyttäjän lisäämä, hyväksytystä muutostiedostosta kertova kommentti ei sisältänyt kysymyksiä, kyseenalaistamisia tai kannanottoja tietoturvaan. Tämä voitiin todistaa vertaamalla tietoturvaan liittyvien muutostiedostojen dialogeja ja hyväksynnän sisältäviä pääkäyttäjän kommentteja muutostiedostoihin, joissa ei käsitelty tietoturvaa.

Bugzilla-järjestelmän sijaan todennäköisin foorumi OpenSSH-ohjelmiston tietoturvan kehittämiselle on OpenSSH-ohjelmistoprojektin kehittäjille tarkoitettu, suljettu postituslista. Openssh.org -sivuston ohjeistuksen perustella listalla käsiteltiin ohjelmiston tietoturvalliseksi luokiteltavaa ohjelmistokehitystä, vaikkakaan listan ohjeistuksessa ei ollut mainintaa muutostiedostoista tai niiden tietoturvasta. Tämän postituslistan suhdetta tai merkitystä Bugzillan OpenSSH-kehitysympäristöön tai siellä käsiteltyihin hyväksyttyjen muutostiedostojen tietoturvaan ei tässä tutkimuksessa pystytty osoittamaan tai todistamaan.

6. Pohdinta

Tutkimuksen tarkoituksena oli analysoida tietoturvallisuusnäkökohtien vaikutusta muutostiedostojen hyväksymiseen avoimen lähdekoodin ohjelmistoprojekteissa.

Avoimen lähdekoodin ohjelmistojen tietoturva on ajankohtainen aihe avoimen lähdekoodin vuosi vuodelta kasvavan suosion ja suosion mukanaan tuomien uhkien vuoksi. Schryen (Schryen, 2011) väittää avoimeen lähdekoodiin ja tietoturvaan liittyvässä tutkimuksessaan, että kaikki tietojärjestelmässä oleva avoin lähdekoodi on tietoturvan kannalta yhtä turvallista tai turvatonta kuin vastaava, suljettu lähdekoodi. Tätä tutkimustulosta voidaankin peilata Morenon (2006) monitieteelliseen, koodin tietoturvaa käsittelevään tutkimukseen, joka osittain tukee Schryenin (2011) tutkimusta, ja jonka pohjalta avoimen lähdekoodin katsotaan olevan tietoturvan kannalta vahvuus, koska toisaalta avoin koodi (Bretthauer, 2002) oli avoimesti kaikkien nähtävillä ja saatavilla. Tämän vuoksi esim. haitallisen koodin upottaminen muutostiedostoon ei ollut Bugzillan toiminnallisuuden vuoksi mahdollista (Bird et al., 2007). Muutostiedostojen hyväksymisessä monilla asioilla on merkityksensä, eivätkä ne useinkaan itseisarvollisesti kohdistu tietoturvaan (Miller, 2007).

Tutkimusaineiston löydökset liittyivät tietoturvariskeihin, konfigurointiin ja yhteensopimattomuuteen. Tutkimusaineistossa oli myös useita OpenSSH:n yleisellä tasolla liittyviä ongelmia, joita muutostiedostoilla pyrittiin korjaamaan. Yhdenkään tässä tutkimuksessa analysoidun muutostiedostoon liittyvän tiketin dialogissa ei kyseenalaistettu tietoturvaa. Myöskään hyväksymisestä vastaava pääkäyttäjä ei tiketin dialogin missään vaiheessa kyseenalaistanut lopulliseen versioon hyväksymiensä muutostiedostojen tietoturvaa. Tikettien dialogeissa ei ollut evidenssiä siitä, että pääkäyttäjä olisi tarkastanut koodin tietoturvaa tai puuttunut hyväksyttävän koodin tietoturvaan. Tässä yhteydessä dialogilla tarkoitettiin Bugzilla-järjestelmässä OpenSSH-muutostiedoston elinkaaren aikana syntynyttä, tiketin kuvauksen ja sitä seuranneiden kommenttien muodostamaa kokonaisuutta. Analysoitujen muutostiedostojen dialogien perusteella pääkäyttäjä ainoastaan kommentoi muutostiedostojen teknisiä yksityiskohtia. Ohjelman toimintaa analysoineet asiantuntijatasoiset käyttäjät pystyivät Bugzilla-järjestelmän OpenSSH-tiketteihin linkitetyillä, OpenSSH:ta käsittelevillä Internet-sivuilla ja Bugzillan mahdollistaman OpenSSH:n kehitykseen liittyvällä sähköpostilistalla korostamaan löydöstensä, käyttäjien tiedottamien havaintojen luottamusta ja luottamuksellisuutta. Bugzillassa käydyllä OpenSSH:n muutostiedostoon liittyvän dialogin pituudella ei ole suoraa yhteyttä muutostiedoston hyväksymiseen: dialogin pituus tai lyhyys ei vaikuta siihen hyväksyttiinkö muutostiedosto seuraavaan mahdolliseen OpenSSH-julkaisuun (release).

Bugzillan kaltainen avoimen ohjelmakoodin kehitysjärjestelmä (Serrano & Ciordia 2005; Remillard 2005) toisin sanoen mahdollisti sen, ettei ohjelman tai ohjelmistojen käyttäjien tarvinnut luottaa pelkästään ohjelman tekijän sanaan. Ongelman vakavuus ja luonne määrittivät lähtökohtaisesti kysymyksiin pohjautuvien muutostiedostojen elinkaaren Bugzilla-järjestelmään lisätyistä tiketeistä aina osaksi lopullista koodia. Suoria väittämiä sisältäviä tikettejä oli aineistossa vain muutamia, ja tämän perusteella havaintojen luonteesta ja esitystavasta ei suoranaisesti voinut päätellä, oliko kyseessä

avoimeen vai suljettuun lähdekoodiin liittyvä raportti tai tiketti. Ainoastaan järjestelmän luonne kertoi kyseessä olevan avoimen lähdekoodin ohjelmiston.

Järjestelmän käyttäjältä vaadittiin hyväksytty profiili, eli identifiointi ja käyttäjätunnus sisällön tuottamiseksi Bugzillaan. Bugzillan salliman avoimuuden ja myös koodin avoimuuden todettiin olevan tietoturvan kannalta vahvuus, koska vaikka tikettien sisältämä avoin koodi olikin avoimesti kaikkien nähtävillä ja saatavilla, ei esim. haitallisen koodin upottaminen muutostiedostoon ollut kuitenkaan Bugzillan toiminnallisuuden vuoksi mahdollista. Versionhallintajärjestelmän moderaattori oli peruskäyttäjää vahvempien oikeuksiensa ja toimivaltansa puitteissa perustellun valveutunut OpenSSH:n muutostiedostojen sisällön tarkastelun suhteen, ja sen lisäksi hänen vastuullaan oli puuttua järjestelmän ja tietoturvan kannalta olennaiseen ja tarvittaessa kyseenalaiseen toimintaan kyseenalaistamalla, tarkentamalla, editoimalla tai muuten moderoimalla käytyä keskustelua kokonaisuudessaan. Tässäkin terveen järjen ja kokemuksen soveltamisella on jo lähtökohtaisesti merkityksensä (Bird et al., 2007). Tämä osaltaan tarkoitti muutostiedoston elinkaaren tarvittaessa kaikkien vaiheiden kokonaisvaltaista hallintaa.

Tietoturvan kannalta tarpeeton avoimuus johtaisi koodin kyseenalaiseen soveltamiseen ja mahdollisiin väärinkäytöksiin, joten OpenSSH:n kestävän toteutuksen kannalta luottamuksellinen kehitystyö, jollaiseksi luokitellaan mm. salausalgoritmeihin liittyvä kehitys ja kehitykseen liittyvä keskustelu, käytiin suljetulla ja ainoastaan kehittäjille tarkoitetulla openssh.com -postituslistalla. Postituslistan ohjeistuksessa ei ollut mainintaa muutostiedostojen käsittelystä, joten tämän listan suhdetta tai merkitystä Bugzillan OpenSSH-ohjelmiston hyväksyttyjen muutostiedostojen tietoturvaan ei tässä tutkimuksessa voitu osoittaa tai todistaa. Bugzillassa olevan ohjelmakoodin soveltaminen tietoturvan kannalta kyseenalaiseen ja tarkoitusperältään arveluttavaan käyttöön oli mahdollista, joten käytännössä kuka tahansa pystyi avaamaan ja kopioimaan liitetiedoston koodeineen suoraan muutostiedoston säikeestä, ja siten hyödyntämään sitä haluamallaan tavalla. Tutkimuksen (Schneider, 2000) mukaan eräs avoimen lähdekoodin ohjelmistojen tietoturvaan liittyvä huoli on nimenomaan vapaa pääsy ohjelman tai ohjelmiston lähdekoodiin, jolloin ohjelmistovirheiden aiheuttamat haavoittuvuudet olisivat myös helposti hyödynnettävissä kyseenalaisiin tarkoitusperiin. Lähdekoodin tutkimista ja tarkastamista on aina pidetty tehokkaana ohjelmistojen virheiden havaitsemisessa, joten tämä tarjoaa tietoturvan kannalta myös vastakkaisen näkemyksen (Asundi & Jayant, 2007). Tämän tutkimuksen kannalta mahdollinen jatkokehityskohde voisikin olla tutkimus, jossa selvitetään tällaisen ominaisuuden hyödyntäminen porsaanreikänä kiistanalaisen ohjelmakoodin luomisessa ja käytössä.

Hyväksymiskäytäntö jakautui Bugzillan OpenSSH:n osalta hyvin yksiselitteisesti ainakin kahden kokeneen avainhenkilön vastuulle. Avoimeen lähdekoodiin kohdistuneessa tutkimuksessaan Asundin ja Jayant'in (2007) näkemyksen mukaan tämän tyyppisten ohjelmien ja prosessien ohjelmistokehityksen osalta on yleisesti olemassa vain rajoittunut käsitys ja niukasti ymmärrystä, vaikka Bird kumppaineineen (Bird et al., 2007) on tutkimuksessaan todennut, että muutostiedostojen ehdotus- ja hyväksyttämisprosessi osaksi lopullista lähdekoodikokoelmaa on avoimen lähdekoodin elinkaaren ja palapelin tärkeä ja olennainen palanen. Asundi ja Jayant (2007) ovat tutkimuksessaan voineet osoittaa, että siinä missä muutostiedoston katselmointiprosessit eivät ole täysin identtisiä tyypillisten OSS-projektien kesken, siinä kaikkien projektien ydinjäsenet, OpenSSH:n tapauksessa kaksi avainhenkilöä, painottavat moderaattorin, operaattorin tai portinvartijan olennaista ja aktiivista roolia, jotta hyväksyttyjen muutostiedostojen korkeatasoinen katselmointi voitaisiin taata.

Monissa OpenSSH:n kaltaisissa projekteissa ohjelmiston tai ohjelmistoprojektin kehittäjien sähköpostilista oli tietyllä tavalla määritely media, jonka avulla muutostiedostoja esitettiin katselmoitaviksi ja liitettäviksi projektin arkistoon. Ensimmäinen askel muutostiedoston jättämisessä ja näiden muutostiedostojen analysoinnissa jatkotutkimusta varten käsitti Bugzillan muutostiedostot ja sähköpostilistan kaikkien viestien kokonaisvaltaisen analyysin. Tämän vuoksi muutostiedostojen esittämisen ja hyväksymisen prosessit ovat olennaisia OSS-yhteisöille ja siten mahdollinen jatkotutkimuskohde (Bird et al., 2007).

Tässä tutkimuksessa ei voitu muodostaa minkäänlaista empiiristä todistetta siitä, että avoin tai suljettu lähdekoodi tiettynä ohjelmistokehityksen tyyppinä olisi kanavoinut tietoturvaa millään tavoin, mutta toisaalta tietoturva-aspekti tulee huomioida. Avointa lähdekoodia pidetään taloudellisesti tehokkaana mallina uusien innovaatioiden kaupallistamiselle (Arakji & Lang, 2007). Eräänä tärkeimmistä periaatteista avoimen lähdekoodin filosofiassa onkin, että kuka tahansa voi haluamallaan tavallaan päättää osallistumisestaan ja osallisuudestaan tiettyyn, omia lähtökohtiaan vastaavaan hankkeeseen.

29	2055	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	channel_setup_fwd_listener return value used incorrectly	02.01.13	Batch	Status:	RESOL
30	1965	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	IPQoS option ignored for AF_INET since 5.9p1-1	14.10.12		Status:	RESOL
31	1985	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	-N and -O stop	07.09.12		Status:	RESOL
32	1815	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	RemoteCommand and PseudoTTY config options	07.09.12		Status:	RESOL
33	1999	Portable	ssh	djm@mindrot.org	RESO	FIXE	When speaking v2, send client version first to avoid long delay with some proxies	17.08.12		Status:	RESOL
34	2022	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	ssh segfaults when using ldns, SSHFP, a DNSSEC-enabled resolver and a CNAME	20.07.12		Status:	RESOL
35	1978	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	ECDSA & SHA256 support in SSHFP DNS records	17.07.12		Status:	RESOL
36	1995	Portable	ssh	djm@mindrot.org	RESO	FIXE	RequestTTY=no in config doesn't work if stdin is not a tty	06.07.12		Status:	RESOL
37	2003	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	[mux.c] fix memory leaks when new session message is malformed	06.07.12		Status:	RESOL
38	2002	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	[mux.c] fix memory leak of control path if bind() fails	01.06.12		Status:	RESOL
39	1961	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	ECDSA memory leak	14.02.12		Status:	RESOL
40	1943	Portable	ssh	unassigned-bugs@mindrot.or	RESO	FIXE	[PATCH] ssh -W opens two connections when ControlPersist is enabled.	07.01.12		Status:	RESOL
41	2026	Portable	ssh	unassigned-bugs@mindrot.or	RESO	INVA	OpenSSH client leaks username to server	20.07.12		Status:	RESOL
42	2000	Portable	ssh	unassigned-bugs@mindrot.or	RESO	INVA	when using ssh with ControlMaster/ControlPersist, one may get zombie processes	16.05.12		Status:	RESOL
43	2006	Portable	ssh	unassigned-bugs@mindrot.or	RESO	INVA	AIX 5.2 /32 bit - a windows Putty session will not connect to AIX box	10.05.12		Status:	RESOL
44	2053	Portable	ssh	unassigned-bugs@mindrot.or	RESO	WONT	Add option to allow skipping userauth_banner (patch)	21.12.12		Status:	RESOL
45											

Kuva 7. Tutkimusaineiston taulukointia aputaulukon avulla.

Analysointivaiheessa käytettiin useita aputaulukoita (ks. kuva 7), joita apuna käyttäen Bugzillan käsittelemätön raakadata oli mahdollista työstää haluttuun muotoon sisällön siitä kärsimättä. Tutkimuksen ensimmäisessä vaiheessa tietoturvaa koskevat tiketit, muutostiedostot ja niihin liittyvien keskustelujen löytäminen tehtiin käyttäen apuna taulukkolaskentaa ja hakuominaisuutta, joilla löydösten hakeminen oli mahdollista. Toinen tutkimuksen vaihe käsitti aineiston sisällöllisen analysoinnin, eli aineistosta pyrittiin löytämään kaikki muutostiedostoihin ja erityisesti niiden tietoturvaan liittyvät keskustelut.

7. Yhteenveto

Tämän tutkimuksen tarkoituksena oli selvittää tietoturvallisuusnäkökohtien vaikutus muutostiedostojen hyväksymiseen avoimen lähdekoodin ohjelmistoprojekteissa.

Tutkimuksen tutkimuskohteeksi valikoitui OpenSSH (OpenBSD Secure Shell), joka on kokoelma avoimeen lähdekoodiin perustuvia tietoliikenneohjelmia. Näiden ohjelmien ensisijainen tarkoitus on tarjota ja mahdollistaa tietoturvallinen tiedonsiirto ja etäyhteyksien muodostaminen tietojärjestelmien välille (Venkatachalam, 2007). Tässä tutkimuksessa tutkittiin tietoturvan vaikutusta muutostiedoston hyväksymisprosessissa. Muutostiedostojen hyväksymisprosessi tapahtuu kokonaisuudessaan avoimeen lähdekoodiin perustuvassa Bugzilla-virheidenraportointijärjestelmässä (Serrano & Ciordia 2005; Remillard 2005; Bird et al 2007).

Tutkimuksen kohteena olevan aineiston analyysissä käytettiin kahta toisistaan poikkeavaa analyysimenetelmää, jotka olivat hakusanoihin perustuva analyysi ja sisällönanalyysi. Molemmat analyysit kohdistuivat Bugzillassa olevan tiketin elinkaaren jokaiseen vaiheeseen alkaen muutostiedostosta ja päättyen osaksi lopullista julkaisua (release).

Tutkimuksessa tutkimusmenetelmänä käytetty hakusanapohjainen analyysi kohdistettiin muutostiedoston sisältävien tikettien otsikkoon ja rakenteisiin. Tällainen analyysi tehtiin koneellisesti ja osittain käsin, hakusanojen perusteella analysoitavaa tekstisisältöä mekaanisesti läpikäymällä. Tavoitteena oli yrittää löytää tutkimuksen kannalta olennaisia, kuvaavia ja merkityksellisiä sanoja tai termejä. Hakusanoihin perustuvassa analyysissä sovellettiin englanninkielisiä variaatioita tietoturvaan liittyvistä peruskäsitteistä ja hyökkäämisessä käytettävistä termeistä (palvelunestohyökkäys, koodin injektointi ja puskuriylivuoto, luottamuksellisuus, eheys, saatavuus, pääsynvalvonta, osapuolten todentaminen ja tunnistaminen). Tämän analyysin tuloksena löytyivät löydökset ”verification” ja ”authentication”, joiden suomenkieliset vastineet ovat ”osapuolten todentaminen” ja ”todennus”.

Sisällönanalyysissä tutkittava aineisto käydään läpi erittelemällä, etsimällä yhtäläisyyksiä tai eroja. Sisällönanalyysissä löytyi 19 muutostiedostoihin ja tietoturvaan liittyvää, keskenään samankaltaista ja vertailukelpoista tikettiä. Näiden tikettien määrä, joka kattaa kaikki vuoden 2012 hyväksytyt Bugzilla-järjestelmässä olevat OpenSSH-ohjelmiston muutostiedostot on otannaltaan riittävä ja niistä oli mahdollista johtaa tutkimuksen lopputulokset.

Bugzilla-järjestelmässä tapahtuvassa OpenSSH-ohjelmiston muutostiedostojen hyväksymisprosessissa ei huomioida tietoturvaa. Tietoturvaan liittyvä yhteydenpito ja ohjelmistokehitys tapahtuu ohjelmiston kehittäjien suljetulla postituslistalla.

Schryen (Schryen, 2011) väittää avoimeen lähdekoodiin ja tietoturvaan liittyvässä tutkimuksessaan, että kaikki tietojärjestelmässä oleva avoin lähdekoodi on tietoturvan kannalta yhtä turvallista tai turvatonta kuin vastaava, suljettu lähdekoodi. Morenon (2006) monitieteellisen, tietoturvaa käsittelevän tutkimuksen pohjalta avoimen lähdekoodin avoimuuden voidaan todeta olevan tietoturvan kannalta vahvuus, koska

avoin lähdekoodi on avoimesti kaikkien nähtävillä ja saatavilla. Bretthauerin (2002) tutkimustulosta voidaankin näin peilata myös Schryenin (2011) avoimen lähdekoodin tietoturvaan kohdistuneeseen tutkimukseen.

Tämän ominaisuuden vuoksi esim. haitallisen koodin upottaminen OpenSSH-ohjelmiston muutostiedostoon ei ollut Bugzillan toiminnallisuuden vuoksi mahdollista (Bird et al., 2007). Muutostiedostojen hyväksymisessä monilla asioilla on merkityksensä, eivätkä ne useinkaan itseisarvollisesti kohdistu tietoturvaan (Miller, 2007).

Tässä tutkimuksessa käytettyjä, viitekehyksen muodostaneet tietoturvan käsitteet olivat puskurin ylivuoto, palvelunestohyökkäys, koodin injektointi. Pienten, enintään neljän koodirivin verran editoitujen muutostiedostojen mahdollisuudet tulla hyväksytyksi ovat (Weißgerber et al., 2008) aiemman tutkimuksen perusteella keskimääräistä korkeammat. Tämän tutkimuksen, tutkielman ja tutkielmaan liittyvän tutkimusdatan pohjalta voisi mahdollisena jatkotutkimusehdotuksena olla tietoturvaan, avoimeen lähdekoodiin ja OpenSSH-näkökulmaan kantaa ottavan ohjeistuksen koostaminen.

Mahdollinen jatkokehityskohde voisi olla avoimen lähdekoodin tietoturvatoiminnallisuuksien vertaaminen suljetun lähdekoodin ratkaisuihin, ja erityisesti se, miten suljetun lähdekoodin ratkaisuja olisi mahdollista hyödyntää avoimen lähdekoodin kehittämisessä ja kehitykseen liittyvissä ratkaisuissa.

Toisena jatkokehityskohteena voisi olla Bugzilla-järjestelmää kehittyneempien työkalujen miettiminen, ja erityisesti työkalujen analytiikan jatkokehittäminen. Tällaisilla työkaluilla tietoturvaan ja tietoturvallisuuteen vaikuttavien ohjelmointivirheiden luokitteleminen voisi olla tehokkaampaa. Lähdekoodin tutkimista ja tarkastamista on aina pidetty tehokkaana ohjelmistojen virheiden havaitsemisessa, joten tämä tarjoaa tietoturvan kannalta myös vastakkaisen näkemyksen. Tämän tutkimuksen kannalta mahdollinen, neljäs jatkokehityskohde voisikin olla tutkimus, jossa selvitetään tällaisen ominaisuuden hyödyntäminen porsaanreikänä kiistanalaisen ohjelmakoodin luomisessa ja käytössä.

Kolmantena jatkokehityskohteena voisi olla tietoturvaominaisuuden lisääminen Bugzilla-järjestelmään ja erityisesti OpenSSH-ohjelmiston kehitysprosessiin. Tällä hetkellä muutostiedostojen tietoturvan huomioiminen Bugzilla-järjestelmässä on muutostiedostot hyväksyvän pääkäyttäjän vastuulla.

Lähteet

Abbot, R. J. (1990). *Resourceful Systems for Fault Tolerance, Reliability, and Safety*. ACM Computing Surveys 22, 1 (March 1990), 35-68. DOI=10.1145/78949.78951 http://doi.acm.org/10.1145/78949.78951

Albrecht, M. R., Paterson, K. G. & Watson, G. J. (2009). *Plaintext Recovery Attacks against SSH,* Symposium on Security and Privacy, 2009 30th IEEE Symposium, IEEE Computer Society Press, 2009. ss. 16-26.

Alexander, J. E. & Tate, M. A. (1999). *Web Wisdom. How to Evaluate and Create Information Quality on the Web.* Wolfgram Memorial Library. Widener University. New Jersey: Lawrence Erlbaum Associates.

Alhazmi, O., Malaiya, Y., Ray, I. (2007). *Measuring, analyzing and predicting security vulnerabilities in software systems. Computers & Security 26,* 3 (2007).

Allen D. R. (2003). *Eleven SSH tricks. Linux J. 2003*, 112 (August 2003), 5-.

Allen, J. (2001). *The CERT Guide to System and Network Security Practices. Addison Wesley Longman Publishing Co., Inc.,* Redwood City, CA, USA. ISBN 0-201-73723-X.

Andersson, S., Clark, A., Mohay, G., Schatz, B. & Zimmermann, J. (2005). *A Framework for Detecting Network-based Code Injection Attacks Targeting Windows and UNIX.* In Proceedings of the 21st Annual Computer Security Applications Conference (ACSAC '05). IEEE Computer Society, Washington, DC, USA, 49-58. http://dx.doi.org/10.1109/CSAC.2005.5

Arakji, R. Y. & Lang, K. R. (2007). *The virtual cathedral and the virtual bazaar, Database for Advances in Information Systems,* SIGMIS Database 38, 4 (October 2007), ss. 32-39. DOI=10.1145/1314234.1314242 http://doi.acm.org/10.1145/1314234.1314242

Asundi, J. & Jayant, R. (2007). *Patch Review Processes in Open Source Software Development Communities: A Comparative Case Study.* In Proceedings of the 40th Annual Hawaii International Conference on System Sciences (HICSS '07). IEEE Computer Society, Washington, DC, USA, 166c-. DOI=10.1109/HICSS.2007.426 http://dx.doi.org/10.1109/HICSS.2007.426

Bauer, M. (2001a). *Paranoid Penguin: The 101 Uses of OpenSSH: Part I*. Linux J. 2001, 81es, Article 15 (January 2001).

Bauer, M. (2001b). *Paranoid Penguin: The 101 Uses of OpenSSH: Part II of II*. Linux J. 2001, 82es, Article 18 (February 2001).

Beizer, B. (1995). *Black-Box Testing: Techniques for Functional Testing of Software and Systems*. Wiley 1995, ISBN 978-0-471-12094-0. ss. I-XXV, 1-294.

Bellare, M., Kohno, T., & Namprempre, C. (2002). *Authenticated encryption in SSH: provably fixing the SSH binary packet protocol.* In Proceedings of the 9th ACM conference on Computer and communications security (CCS '02), Vijay Atluri

(Ed.). ACM, New York, NY, USA, 1-11. DOI=10.1145/586110.586112 http://doi.acm.org/10.1145/586110.586112

Berry, D. (1998). *The Safety Requirements Engineering Dilemma.* In Proceedings of the 9th international workshop on Software specification and design (IWSSD '98). IEEE Computer Society, Washington, DC, USA, 147-.

Bird, C., Gourley, A., Devanbu, P., Gertz, M. & Swaminathan A. (2006). *Mining email social networks.* In Proceedings of the 2006 international workshop on Mining software repositories (MSR '06). ACM, New York, NY, USA. ss. 137–143. DOI=10.1145/1137983.1138016 http://doi.acm.org/10.1145/1137983.1138016

Bird, C., Gourley, A. & Devanbu P. (2007). *Detecting Patch Submission and Acceptance in OSS Projects.* In Proceedings of the Fourth International Workshop on Mining Software Repositories (MSR '07). IEEE Computer Society, Washington, DC, USA, 26-. DOI=10.1109/MSR.2007.6 http://dx.doi.org/10.1109/MSR.2007.6

Botta, D., Werlinger, R., Gagné, A., Beznosov, K., Iverson, L., Fels, S. & Fisher, B. (2007). *Towards understanding IT security professionals and their tools.* In Proceedings of the 3rd symposium on Usable privacy and security (SOUPS '07). ACM, New York, NY, USA, 100-111. DOI=10.1145/1280680.1280693 http://doi.acm.org/10.1145/1280680.1280693

Bretthauer, D. (2002). *Open source software: A history, Information Technology and Libraries*, 21(1), ss. 3-10.

Bugzilla. (2012). *The Bugzilla Guide – Documentation. Administrators & End Users*. Viitattu 20.5.2012. http://www.bugzilla.org/docs/2.16/html/glossary.html

Bugzilla. (2013). *Bugzilla – User's guide.* Viitattu 12.1.2013. https://bugzilla.mindrot.org/docs/en/html/using.html

Byun J., Sohn, Y. & Bertino E. (2006). *Systematic control and management of data integrity.* In Proceedings of the eleventh ACM symposium on Access control models and technologies (SACMAT '06). ACM, New York, NY, USA, 101-110. http://doi.acm.org/10.1145/1133058.1133074.

CERT-FI. (2002a). *CERT-FI haavoittuvuustiedote 014/2002: Haavoittuvuus OpenSSH-ohjelmistossa.* Viitattu 24.3.2011 . http://www.cert.fi/haavoittuvuudet/2002/varoitus-2002-14.html

CERT-FI. (2011). *CERT-FI Viestintävirastossa toimiva kansallinen tietoturvaviranomainen. Finnish national computer emergency response team.* Viitattu 26.3.2011. https://www.cert.fi

CERT-FI. (2013). *CERT-FI Varoitus Internet Explorer 8:ssa. Finnish national computer emergency response team.* Viitattu 7.5.2013. http://www.cert.fi/haavoittuvuudet/2013/haavoittuvuus-2013-067.html.

Chow, T. (1978). *Testing Software Design Modelled by Finite-State Machines,* IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, May 1978.

COSS. (2011). *Suomen avoimen lähdekoodin keskus: Tietoa COSSista.* Viitattu 9.6.2011.
http://www.coss.fi/tietoa-cossista

Cowan, C. (2003). *Software security for open-source systems, Security & Privacy*, IEEE, vol.1, no.1, ss. 38- 45.

Cowan, C., Wagle, P., Pu, C., Beattie, S. & Walpole, J. (2000). *Buffer overflows: Attacks and defenses for vulnerability of the decade.* In Proceedings of DARPA Information Survivability Conference and Exposition.

Crowston, K., Wei, W., Howison, J. & Wiggins, A. (2008). *Free/Libre open-source software development: What we know and what we do not know.* ACM Comput. Surv. 44, 2, Article 7 (March 2008), 35 pages. DOI=10.1145/2089125.2089127 http://doi.acm.org/10.1145/2089125.2089127

CVE. *The ultimate security vulnerability datasource.* (2013). Viitattu 19.5.2013. http://www.cvedetails.com/vulnerability-list/vendor_id-97/product_id-585/Openbsd-Openssh.html

Dashofy, E. M., Medvidovic, N., & Taylor, R. N. (1999). *Using off-the-shelf middleware to implement connectors in distributed software architectures.* In Proceedings of the 21st international conference on Software engineering (ICSE '99). ACM, New York, NY, USA, 3-12. DOI=10.1145/302405.302407 http://doi.acm.org/10.1145/302405.302407

Dhillon, G. (2007). *Principles of Information Systems Security: text and cases.* NY: John Wiley & Sons. ISBN 978-0-471-45056-6.

Dor, N., Rodeh, M. & Sagiv, M. (2003). *Cssv. Towards a realistic tool for statically detecting all buffer overflows in c.* In Proceedings of the ACM SIGPLAN 2003 Conference on Programming Language Design and Implementation.

Drevin L., Kruger H. & Steyn. T., (2007). *Value-focused assessment of ICT security awareness in an academic environment.* Computers & Security. Elsevier. 2007.

Ducheneaut, N. (2005). *Socialization in an Open Source Software Community: A Socio-Technical Analysis.* Computer Supported Co-operative Work (CSCW), 14(4):323–368, 2005.

Dunlap I. (2006). *Open Source Database Driven Web Development.* Oxford: Chandos. ISBN 1-84334-161-1.

Dutta, P., Hui, J., Chu, D. & Culler, D. (2006). *Securing the deluge Network programming system.* In Proceedings of the 5th international conference on Information processing in sensor networks (IPSN '06). ACM, New York, NY, USA, 326-333. http://doi.acm.org/10.1145/1127777.1127826.

Fagan, M. E. (1976). *Design and Code inspections to reduce errors in program development.* IBM System Journal 15.

Feller, J. & Fitzgerald, B. (2002). *Understanding Open Source Software Development*, Addison Wesley, London, England.

Frisch, Æ. (1997). *UNIX tehokäyttäjän opas*, 1997. Suomen Atk-kustannus Oy.

Glass, G. (1999). *King Ables: UNIX for programmers and users*, 1999. Prentice Hall.

Goldman, R. & Gabriel, R. (2005). *Innovation Happens Elsewhere*. Richard P. Gabriel. ISBN 1-55860-889-3.

Greiner, S., Boskovic, B., Brest, J. & Zumer, V. (2003). *Security issues in information systems based on open-source technologies,* EUROCON 2003. Computer as a Tool. The IEEE Region 8 , vol.2, no., ss. 12- 15 vol.2, ss. 22-24.

Gurbani, V. K., Garvert, A. & Herbsleb, J. D. (2006). *A case study of a corporate open source development model.* In Proceedings of the 28th international conference on Software engineering (ICSE '06). ACM, New York, NY, USA, 472-481. DOI=10.1145/1134285.1134352 http://doi.acm.org/10.1145/1134285.1134352

Hansen, P. B. (1973). *Operating System Principles.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

Hauge, O., Osterlie, T., Sorensen, C-F. & Gerea, M. (2009). *An empirical study on selection of Open Source Software - Preliminary results.* In Proceedings of the 2009 ICSE Workshop on Emerging Trends in Free/Libre/Open Source Software Research and Development (FLOSS '09). IEEE Computer Society, Washington, DC, USA, 42-47. http://dx.doi.org/10.1109/FLOSS.2009.5071359.

Henslin, J. M. (2001). *Essentials of Sociology. A Down-toEarth Approach.* Taylor & Francis. ISBN 0-536-94185-8.

Hirsjärvi, S., Remes, P. & Sajavaara, P. (2001). *Tutki ja kirjoita. (6. - 7. painos)*. Helsinki: Tammi. (In Finnish).

Hoskins, M. (2006). *Securing openSSH*. Linux J. 2006, 147 (July 2006), 9-.

Iyappan, P., Arvind, K. S., Geetha, N. & Vanitha, S. (2009). *Pluggable Encryption Algorithm In Secure Shell (SSH) Protocol, Emerging Trends in Engineering & Technology,* In Proceedings of the International Conference on, pp. 808-813, 2009 Second International Conference on Emerging Trends in Engineering & Technology, 2009.

Jakubowska, G. & Penczek, W. (2007). *Is your security protocol on time?*. In Proceedings of the 2007 international conference on Fundamentals of software engineering (FSEN'07), Farhad Arbab and Marjan Sirjani (Eds.). Springer-Verlag, Berlin, Heidelberg, 65-80.

Jokinen, A., Juhila, K. & Suoninen E. (1993). *Diskurssianalyysin aakkoset.* Tampere: Vastapaino.

Järvinen, P. (2001). *On research methods*. Tampere: Opinpajan kirja.

Järvinen, P., & Järvinen, A. (2004). *Tutkimustyön metodeista (Uud. p.).* Tampere: Opinpajan kirja. (In Finnish).

Kerievsky, B. (1976). *Security and confidentiality in a university computer network*. SIGUCCS Newsl. 6, 3 (September 1976), 9-11.

Layton, T. (2007). *Information Security: Design, Implementation, Measurement, and Compliance*. Boca Raton, FL: Auerbach publications. ISBN 978-0-8493-7087-8.2011

Linux.fi. (2011). *Linux.fi-wiki: Avoin lähdekoodi.* Viitattu 9.6.2011. http://linux.fi/wiki/Avoin_lähdekoodi

Linux.fi. (2011). *Linux.fi-wiki: OpenSSH.* Viitattu 13.6.2011. http://linux.fi/wiki/OpenSSH

Miller D. (2007). *Security Measures in OpenSSH.* In the proceedings of the AsiaBSDCon 2007, Usenix, March 2007.

Moreno, M. (2006). *Open Source: A Multidisciplinary Approach*. Imperial College Press. ISBN 1-86094-665-8.

OpenBSD. (2012). OpenBSD. Viitattu 27.4.2012. http://www.openbsd.org/security.html

OpenBSD. (2013). OpenBSD. Viitattu 18.5.2013. http://www.openbsd.org/

OpenSSH. (2011). OpenSSH. Viitattu 27.3.2011. http://www.openssh.com/

Open Source Initiative. (2013). *The Open Source Initiative (OSI)*. Viitattu 21.4.2013. http://opensource.org/

Owen, C. (2007). *Tracking system bugs: why are buffer overruns still around?*. In Proceedings of the 35th annual ACM SIGUCCS fall conference (SIGUCCS '07). ACM, New York, NY, USA, 280-283. DOI=10.1145/1294046.1294112 http://doi.acm.org/10.1145/1294046.1294112.

Parnas, D., van Schouwen, J. & Po Kwan, S. (1990). *Evaluation of safety-critical Software.* Communications of the ACM 33, 6 (June 1990) 636-648.

Peltier, T. (2002). *Information Security Policies, Procedures, and Standards: guidelines for effective information security management.* Boca Raton, FL: Auerbach publications. ISBN 0-8493-1137-3.

Raja, U. & Barry, E. (2005). *Investigating quality in large-scale Open Source Software.* In Proceedings of the fifth workshop on Open source software engineering (5-WOSSE). ACM, New York, NY, USA, ss. 1-4.

Rakic, M. & Medvidovic N. (2001). *Increasing the confidence in off-the-shelf components: a software connector-based approach.* SIGSOFT Softw. Eng. Notes 26, 3 (May 2001), 11-18. DOI=10.1145/379377.375228 http://doi.acm.org/10.1145/379377.375228

Remillard, J. (2005). *Source code review systems.* Software, IEEE , vol.22, no.1, pp.74,77, Jan.-Feb. 2005 doi: 10.1109/MS.2005.20

Russell, J. (2004). *A Conversation with Sam Leffler. Queue* 2, 3 (May. 2004), s.16-22.

Ruwase, O. & Lam, M. (2004). *A Practical Dynamic Buffer Overflow Detector.* In the proceedings of the 11th Annual Network and Distributed System Security Symposium (NDSS '04) February 2004.

Schneider, F. (2000). *Open Source in Security: Visiting the Bizarre*. ss .0126, 2000 IEEE Symposium on Security and Privacy (S&P 2000).

Schryen, G. (2009). *Security of open source and closed source software: An empirical comparison of published vulnerabilities.* In 15th Americas Conference on Information Systems, August 6 - 9, 2009, San Francisco, California.

Schryen, G. (2011). *Is open source security a myth?* Communications of the ACM, Vol 54, Issue 5, May 2011. ss. 130-140.

Schryen G. & Kadura, R. (2009). *Open source vs. closed source software: towards measuring security.* In Proceedings of the 2009 ACM symposium on Applied Computing (SAC '09). ACM, New York, NY, USA, 2016-2023. http://doi.acm.org/10.1145/1529282.1529731

Schryen, G. & Rich, E. (2010). *Increasing Software Security through Open Source or Closed Source Development? Empirics Suggest that We have Asked the Wrong Question,* System Sciences (HICSS), 2010 43rd Hawaii International Conference on , vol., no., ss.1-10.

Serrano, N., Ciordia, I. (2005). *Bugzilla, ITracker, and other bug trackers.* Software, IEEE , vol.22, no.2, pp.11,13, March-April 2005 doi: 10.1109/MS.2005.32

Shao, Z., Xue, C., Zhuge, Q., Sha, E., & Xiao, B. (2004). *Security protection and checking in embedded system integration against buffer overflow attacks, Information Technology: Coding and Computing,* 2004. In Proceedings of the ITCC 2004. International Conference on , vol.1, no., pp.409,413 Vol.1, 5-7 April 2004, doi: 10.1109/ITCC.2004.1286489.

Spinellis D., Gousios G., Karakoidas V., Louridas P., Adams P. J., Samoladas I. & Stamelos I. (2008). *Evaluating the Quality of Open Source Software, Electronic Notes in Theoretical Computer Science*, Volume 233. In Proceedings of the International Workshop on Software Quality and Maintainability (SQM 2008), 27 March 2009, ss. 5-28, ISSN 1571-0661.

Stewart, K., Darcy, D. & Daniel, S. (2005). *Observations on patterns of development in open source software projects.* SIGSOFT Softw. Eng. Notes 30, 4 (May 2005), 1-5. http://doi.acm.org/10.1145/1082983.1083272.

Stuart, B. (2009). *Principles of operating systems: design & applications*. Boston, Massachusetts: Thompson Learning. ISBN 1-4188-3769-5.

Suomen Internetopas. (2011). *Suomen Internetopas: Mitä on tietoturva.* Viitattu 9.6.2011.
http://www.internetopas.com/yleistietoa/tietoturva/.

Tanhuanmäki, H. (2006). *Kriittisten järjestelmien muutoksen hallinta.* (Pro gradu -tutkielma, Tietojenkäsittelytieteiden laitos, Tampereen yliopisto).

Thomas, K. (2006). *Chapter 10: personalizing ubuntu: getting everything just right.* Linux J. 2006, 150 (October 2006), 6-.

TIEKE. (2011). *TIEKE Tietoyhteiskunnan kehittämiskeskus ry: Tietoturva.* Viitattu 9.6.2011. http://www.tieke.fi/julkaisut/oppaat_yrityksille/sahkoisen_kaupankaynnin_aapinen/tietoturva/

Tuomi, J. & Sarajärvi, A. (2002). *Laadullinen tutkimus ja sisällönanalyysi.* Helsinki: Tammi.

Työ- ja elinkeinoministeriö. (2011). *Työ- ja elinkeinoministeriön selvitys: Avoimen lähdekoodin ohjelmien merkitys kasvaa.* Viitattu 9.6.2011. http://www.tem.fi/index.phtml?101881_m=99270&s=4265

Töttö, P. (2004). *Syvällistä ja pinnallista. Teoria, empiria ja kausaalisuus sosiaalitutkimuksessa.* Tampere: Vastapaino.

Venkatachalam, G. (2007). *The OpenSSH protocol under the hood.* Linux J. 2007, 156 (April 2007), 6-.

Walden J., Doyle M., Welch G. A. & Whelan. M. (2009). *Security of open source web applications.* In Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement (ESEM '09). IEEE Computer Society, Washington, DC, USA, ss. 545-553. DOI=10.1109/ESEM.2009.5314215 http://dx.doi.org/10.1109/ESEM.2009.5314215

Wasko, M. M. & Faraj, S. (2005). *Why should I share? Examining social capital and knowledge contribution in electronic networks of practice.* MIS Quarterly, 29(1) (March 2005), ss. 35-57.

Weber, S. (2005). *The Success of Open Source.* Harvard University Press, Cambridge, MA, USA. ISBN 978-0-674-01292-9.

Weiss, A. (2008). *Linux Ready for the Desktop? Yes, No, and Maybe.* NetWorker, 12(2), s. 40.

Weißgerber, P., Neu, D. & Diehl, S. (2008). *Small patches get in!.* In Proceedings of the 2008 international working conference on Mining software repositories (MSR '08). ACM, New York, NY, USA, 67-76. DOI=10.1145/1370750.1370767 http://doi.acm.org/10.1145/1370750.1370767

Whitman, M. E. & Mattord, H. J. (2008). *Principles of Information Security, 3rd ed. 2008 Course Technology,* Boston, MA, ISBN 1-4239-0177-0

Wikipedia. (2011) . *Tietoturva*, Viitattu 1.6.2011. http://fi.wikipedia.org/wiki/Tietoturva

Xu, B. & Jones, D. R. (2010). *Volunteers' participation in open source software development: a study from the social-relational perspective.* SIGMIS Database 41, 3 (August 2010), ss. 69-84. DOI=10.1145/1851175.1851180 http://doi.acm.org/10.1145/1851175.1851180

Zhou, Y. & Davis, J. (2005). *Open source software reliability model: an empirical approach.* SIGSOFT Softw. Eng. Notes 30, 4 (May 2005), 1-6. http://doi.acm.org/10.1145/1082983.1083273.