Tietoturva – luento kurssille Johdatus ohjelmistotekniikkaan

Jonne Itkonen

Jyväskylän yliopisto, tietotekniikan laitos

draft – 16. joulukuuta 2003

1 HUOMIOI!!!!

Tämä kirjoitus on keskeneräinen ja tulee pysymään sellaisena kunnes toisin mainitaan. Tätä kirjoitusta ei saa käyttää kuin viitteellisenä oheislukemistona kurssilla ITK130 Johdatus ohjelmistotekniikkaan. Tälle kirjoitukselle ei saa perustaa omia tietoturvaratkaisujaan tai toteutuksiaan. Sanoudun irti kaikista virheistä tässä kirjoituksessa sekä kaikista mahdollisista ongelmista mitä aiheutuu, jos vastoin ylläolevia ohjeita joku tätä tietoturvansa toteutukseen käyttää.

Pidän kirjoituksen tietoa kuitenkin validina kurssin tarpeisiin. Jos korjattavaa löytyy, informoikaa siitä, ja korjaan kyseiset kohdat. Kirjoitus tulee kehittymään joskus, aikoja en lupaa.

Kirjoituksen käyttöön muita tarpeita varten on ehdottomasti otettava ensin yhteyttä tekijään. Tämä koskee kaikkea muuta käyttöä, kuin kurssin ITK130 tenttiin valmistautumista.

Eli, jos muuhun kuin tenttiinlukuun tätä kirjoitusta käytät, toimit väärin ja vastuu on yksin sinun.

2 Yleistä

Tietoturva on hankala ja monimutkainen asia. Se ulottuu aivan kaikkeen, mitä ohjelmistokehitykseen ja ohjelmistojen käyttämiseen kuuluu. Jätä tietoturvallisuuden varmistaminen asiantuntijoille. Jos tiedät, mitä on tietoturva, mikä on tietoturvallista, jätät sen varmistamisen asiantuntijoille. Jos haluat tulla tietoturva-asiantuntijaksi, etkä tiedä, mistä aloittaa, suosittelen, että vaihdat haaveesi.

Tämä paperi sen paremmin kuin tämä luento ei tee sinusta tietoturva-asiantuntijaa. Kaikkien tietokoneiden kanssa työskentelevien on kuitenkin syytä tietää jotain tietoturvasta. Toivottavasti tämä paperi auttaa täman tavoitteen saavuttamisessa.

Tietoturva koostuu kahdesta osa-alueesta, tiedon turvaamisesta ja tiedon turvallisesta käsittelystä. Tiedon turvaamisen neljä alalajia ovat tiedon salaaminen, tiedon eheys, aitouden varmentaminen ja tiedon lähettämisen kiistämisen estäminen. Ensimmäinen estää tiedon ymmärtämisen ulkopuolisilta, toinen osoittaa tiedon muuttumattomuuden ja kolmas kertoo viestivien osapuolien olevan juuri keitä he väittävätkin olevansa tai todentaa viestin. Neljäs kohta tarkoittaa sitä, ettei lähettäjä voi jälkeenpäin kiistää lähettäneensä tietoa. Tätä ei enemmälti paperissa, eikä luennolla käsitellä. Tiedon turvallisen käsittelyn mahdollistaa virheettömät ja oikein toimivat sovellukset. Tulee kuitenkin aina muistaa, että järjestelmä on yhtä vahva, kuin sen heikoin lenkki. Heikoin lenkki ei aina ole tietokone, sovellus tai tietoverkko. Yleensä se löytyy tuolin ja näppäimistön välistä.

Tietoturvaa heikentää monesti myös virheellisesti tai huolimattomasti kirjoitettu sovellus. Kenties yleisin huolimattomuus on puskurivuoto: varataan liian vähän muistia käyttöön tai ei valvota kunnolla, ettei ohjelma voi kirjoittaa varatun muistialueen ulkopuolelle. Toinen yleinen huolimattomuus on liian suurien oikeuksien antaminen esimerkiksi sovelluksen sisäisellä juontokielellä kirjoitetuille ohjelmille.

Tiedon turvallisuuteen liittyy myös miten tietoa käytetään, ja onko oikea tieto käytössä. Tällaisia ongelmia, esimerkiksi onko aivokuvauslaitteessa juuri oikeat asetukset tai kuinka virukset ja troijalaiset toimivat, kuinka niitä torjutaan, ei tässä kirjoituksessa käsitellä.

Ensin tutustutaan hieman tiedon turvaamiseen, sitten turvallisten tietoa käsittelevien ohjelmien tekemiseen. Lopuksi pohditaan hieman alan ongelmia.

3 Tiedon turvaaminen

Tietoa on salattu ja varmennettu aikojen alusta lähtien. Myös jokainen teistä on salannut ja varmentanut tietoa elämänsä aikana. Syitä ei tässä lähdetä etsimään, sen sijaan käydään läpi joukko erilaisia tapoja tiedon salaamiselle ja varmentamiselle. Salaamisesta enemmän kiinnostuneita kehoitan tutustumaan matematiikan laitoksen kurssiin MAT229 Salakirjoitukset, lisätietoja osoitteesta: `http://www.math.jyu.fi/opiskelu/kurssi_infot/salakirjoitukset.html`

3.1 Lasten leikit

Kopsenalantti konimontti kononttti kohunutpuntti kontinkontti koiltäkintti. Patkutjova povatava jaattaneetsava paksiäkeva paviaoma paliäkiva. Nämä kielet ovat yksi esimerkki tietoturvasta. Tärkeä viesti, jota aikuiset eivät saaneet lukea, on koodattu salakielelle, ja näin pysynyt lasten salaisuutena.

Kontinkielen ja vastaavien lisäksi joku on saattanut tutustua yksinkertaiseen tapaan salata sanoja sotkemalla kirjaimet. Valitaan aakkosille joku satunnainen järjestys, avain,

ja muutetaan viestin a-kirjaimet avaimen ensimmäiseksi kirjaimeksi, b-kirjaimet toiseksi ja niin edelleen. Kolmas vaihtoehto on erikoisten symbolien käyttö kirjainten tilalla, esimerkiksi Morsen aakkosilla tai mielikuvituksellisilla oudoilla häkkyröillä.

Nämä tavat eivät ole kovin turvallisia, sillä ne voidaan murtaa yksinkertaisesti vertailemalla kirjainten esiintymistiheyksiä. Jos tiedetään, että kielessä esiintyy eniten kirjainta k, korvataan salakirjoituksen eniten esiintyvä kirjain kirjaimella k. Pienellä kokeilulla kaikki kirjaimet saadaan selville.

Hankalampi tapa salata kieli on kirjoittaa viesti jollain toisella, vähän käytetyllä kielellä, esimerkiksi intiaaniheimojen kielillä, eskimokielillä tai vaikka suomeksi. Lukijoita ja ymmärtäjiä on taatusti vähemmän maailmassa kuin englanninkieliselllä tekstillä. Tätä tapaa on käytetty melko menestyksekkäästi aiemmissa sodissa.

Yski hukasa ilmiö on htaivatu viime aiionka, nmiätitin jos saenojn släislä olveat kjmeriiat seotkatan, psyyy teksti keikiuntn khitsluloeun lvuataenta. Tihs mghit be bteter sowhn in Egnlsih, as wodrs tned to be a bit shteorr in that laaggune, comreapd to Fnnsiih, for elmpxae.

Meniotniiknelein on möys htniavao, että sjonaa, jedion slläsiä ketniajrin jytsejräs on vttedhiau pskesiatsavniäi, ei oaakeln niin hopplea lekua. Tihs is olny a slaml elpmaxe scnetnee, so you can tset, if the ecefft debircsed is mroe pnesert in Esilgnh lgaugnae.

3.2 Tiedon piilotus

Nämä ylläkuvatut tavat salata tietoa eivät ole kovin tehokkaita. Entäpä jos tiedon salaamisen sijaan tieto piilotettaisiin? Tämäkään ei ole mikään uusi keksintö. Antiikin aikoina viesti saatettiin kirjoittaa viestinviejän paljaaksi ajeltuun päänahkaan. Sitten odotettiin, että viestinviejän hiukset kasvoivat tarpeeksi pitkiksi, ja lähetettiin hänet matkaan. Perillä pää ajeltiin ja viesti luettiin. Joskus maailmassa on ollut aikaa…

Nykyään viesti voidaan piilottaa kuvaan tai äänisignaaliin. Tavallinen digitaalikuva sisältää jo niin paljon informaatiota, pikseleitä, ettei kukaan huomaa, jos siellä täällä joku piste hieman poikkeaa ympäristöstä. Tällaista tiedon piilottamista informaatiovirtaan kutsutaan steganografiaksi. Ei mikään uusi keksintö tämäkään, vanhoista puupiirustuksista on löydetty kosolti piiloviestejä.

Tämän salaamisen tehokkuus perustuu siihen, etteivät muut kuin viestin lähettäjä ja vastaanottaja, Alice ja Bob, tiedä miten viesti esille saadaan. Alice ja Bob?

3.3 Saippuaoopperan sankarit

Lienee aika esitellä tämän saippuaoopperan sankarit. Jo pitkään ovat tietoturvateksteissä seikkailleet Alice, Bob, Carol, Eve ja Trent. Tieto, joka tulee turvata, liikkuu Alicelta Bobille tai toisinpäin. Carol on heidän ystävänsä, mutta Eve on ilkeä urkkija, joka pyrkii saamaan selville mitä salaisuuksia Alicella, Bobilla ja Carolilla on keskenään. Trent on kolmas osapuoli, jonka huostaan Alice, Bob ja Carol uskaltavata luovuttaa salaisuuksiaan. Tarpeen vaatiessa muitakin henkilöitä osallistuu näytelmään, mutta emmeköhän pärjää nyt näillä neljällä näyttelijällä. Nimet ovat keksittyjä, eivätkä vastaa tosielämän samankaltaisia henkilöitä ja blaa blaa blaa…

3.4 Salaus

Salauksen tarkoituksena on muuttaa viesti sellaiseen muotoon, ettei alkuperäistä voi tunnistaa tai muodostaa siitä muuten kuin tietämällä jonkin muodosta riippumattoman salaisuuden, avaimen, jolla tieto on salattu. Avain on ohje salausalgoritmille siitä, kuinka sen tulee toimia. Avaimia voi olla yksi tai useita, niistä osa voi olla julkisia, mutta vähintään yhden tulee olla salainen.

Alla salaamista on kuvattu yhtälöllä $E_{K_1}(M) = C$ ja avaamista yhtälöllä $D_{K_2}(C) = M$. Yhtälöissä E on salausfunktio, D avausfunktio, M selkokielinen viesti ja C salattu viesti. Salausavainta merkitään K_1:llä ja avausavainta K_2:llä. Nämä voivat myös olla sama avain. Funktioiden nimet tulevat toimintojen englanninkielisistä nimistä: salata – *encrypt, encipher*, avata – *decrypt, decipher*, viesti – *message*, salattu viesti – *encrypted message, enciphered message* ja avain – *key*. Kryptaus, enkryptaus ja dekryptaus ovat huonoja käännöksiä englannista, kryptaus sana sinänsä jo viittaa huonosti onnistuneeseen elintärkeään salaukseen. Koodaus on viestin esittämistä toisessa muodossa. Esimerkiksi viesti ”ammu ja väistä vasemmalle” voidaan koodata ”kissa istuu”. Koodatessa tulee aina tietää mitä koodataan ja miten. Salatessa mitä tahansa voidaan salata.

Kryptoanalyysi (*cryptanalysis*) on tiede, jonka tarkoituksena on selvittää, kuinka avata salattu viesti ilman tietoa salausavaimesta. Kryptoanalyysin yrittämistä kutsutaan hyökkäykseksi (*attack*). Kryptoanalyysin keinoja ovat mm. salatulla tekstillä, tunnetulla selkotekstillä, valitulla selkotekstillä, mukautuvalla selkotestillä, valitulla salatulla tekstillä ja valitulla avaimella tehty hyökkäys, sekä kumipamppukryptoanalyysi. Tarkempi kuvaus näistä löytyy Schneierin kirjasta Applied Cryptography [4, s. 5–6].

Kryptoanalyysi ja avoimet salausalgoritmit ovat ainut tie turvallisiin ja toimiviin salaualgoritmeihin.

3.5 Eri algoritmeja salaukselle

Salausalgoritmit voidaan jakaa käytännössä kahteen ryhmään, symmetrisiin ja julkisen avaimen menetelmiin. Symmetriset menetelmät ovat yleensä yhden avaimen menetelmiä, eli viesti salataan ja avataan samalla avaimella: $D_K(E_K(M)) = M$. Toinen vaihtoehto on, että salausavain voidaan laskea avausavaimesta, sekä toisinpäin.

Julkisen avaimen menetelmässä, jota myös asymmetriseksi menetelmäksi kutsutaan on kaksi avainta. Viesti salataan julkisella avaimella, mutta viestin pystyy avaamaan vain yksityisellä avaimella: $E_{K_j}(M) = C, D_{K_y}(C) = M$. Salaukseen voidaan myös käyttää yksityistä avainta, jolloin avaamiseen tulee käyttää julkista

avainta. Tätä järjestystä käytetään viestin varmentamisessa. Yksityinen avain on vain viestin salaajan hallussa, joten viestin on voinut salata vain hän, vaikka sen kaikki pystyvätkin avaamaan julkisella avaimella.

Kommunikaatio julkisen avaimen menetelmää käyttäen tapahtuu seuraavaan protokollan mukaan [4, s. 31–33]:

1. Alice ja Bob sopivat käyttämästään julkisen avaimen menetelmästä.
2. Bob lähettää Alicelle julkisen avaimensa (turvallisesti).
3. Alice salaa viestin Bobin julkista avainta käyttäen.
4. Bob avaa tiedon yksityistä avaintaan käyttäen.

Valitettavasti julkisen avaimen menetelmät eivät ole täydellisiä. Ne voivat olla jopa tuhat kertaa symmetrisiä menetelmiä hitaampia. Julkisen avaimen menetelmät ovat myös arkoja valitun selkotekstin avulla tapahtuvalle hyökkäykselle. Hyökkääjä keksii palasen viestiä P ja salaa tämän julkisella avaimella: $E_K(P) = C$. Salattua palasta C voidaan sitten verrata salattuun viestiin näin selvittäen viestin sisällön ilman viestin avaukseen tarvittavaa avainta.

Näiden kahden heikkouden tähden on hyvä käyttää hybridijärjestelmää. Luodaan viestikohtainen salainen salausavain, joka salataan sillä salatun viestin kanssa julkisella avaimella:

1. Bob lähettää Alicelle julkisen avaimensa.
2. Alice luo satunnaisen istuntoavaimen K, jonka hän salaa Bobin julkisella avaimella $E_{K_B}(K)$, ja lähettää Bobille.
3. Bob avaa Alicen viestin yksityisellä avaimellaan saaden istuntoavaimen: $D_{K_B}(E_{K_B}(K)) = K$.
4. Molemmat käyttävät salaamiseen istuntoavainta K.

Salausalgoritmit perustuvat yleensä tiedon permutointiin, tekijöihin jakoon, moduloaritmetiikkaan, suuriin lukuihin, alkulukuihin ja bittioperaatioihin, sekä nykyään myös elliptisiin yhtälöihin. Näistä ei tässä yhteydessä tämän enempää, kiinnostuneita kehoitetaan kirjoittamaan suosikkihakukoneeseensa vaikkapa "blowfish encryption algorithm Schneier". Myös kirjaston täti osaa auttaa löytämään hyviä lähteitä. Muita salausalgoritmeja ovat mm. ElGamal, Twofish ja RSA.

Hätäisemmille on kuvassa 3.5 sivulla 3 pureskeltavaa symmetrisen algoritmin RC4 muodossa. Palvelkoon se myös esimerkkinä tästä taiteenlajista, ei kauniista tavasta kirjoittaa ohjelmia.

3.6 Kryptoanalyysistä (ja algoritmien murtamisesta)

Algoritmien murtaminen on jännittävä ja hyvä harrastus. Toisten salaisuuksien selvittäminen ilman lupaa ei ole laillista, eikä älykästäkään monessakaan tapauksessa. Lue edeltävät kaksi lausetta uudestaan ja uudestaan kunnes tajuat niiden viestin.

Algoritmien murtamiseen on joukko tunnettuja tekniikoita. Ensimmäinen on niin kutsuttu *brute force* -menetelmä, jossa pyritään selvittämään salattu viesti kokeilemalla kaikkia mahdollisia avaimia. Tämä saattaa viedä hieman aikaa[1], joten kannattanee hankkia hieman enemmän tietoa käytetystä algoritmista. Onko sille heikkoja avaimia, onko sillä algoritmisia heikkouksia?

Algoritmia voidaan yrittää murtaa kokeilemalla erilaisia avaimia ja syöttötietoja vertaillen näitä sitten tuloksiin (tai vaikka tietokoneen muistin sisältöön algoritmin ajon jälkeen). Tällöin algoritmia ei tunneta, mutta sen käytön avulla se voidaan tunnistaa tai mallintaa.

Kun on tiedossa käytetty algoritmi ja sen heikkoudet, voidaan analyysi tehdä näitä hyödyntäen. Myös algoritmin ympäristöä kannattaa tarkkailla. Missä käyttöjärjestelmässä algoritmi pyörii? Kannattaisiko hyödyntää järjestelmän satunnaislukugeneraattorin heikkouksia, tai järjestelmän kellon heikkouksia? Joskus murtamisessa auttaa jo viestin salausajankohdan tietäminen.

Kolmas tapa murtamiseen on matematiikka. Algoritmit perustuvat aina johonkin tiettyyn matemaattisiin totuuksiin joltain matematiikan saralta. Onko tuolla alalla löydetty jotain uusia totuuksia, jotka kyseenalaistavat algoritmin luotettavuuden? Jos ei, kirjat kouraan ja opiskelemaan matematiikkaa, ja löytämään itse nuo heikkoudet.

Aina ei kuitenkaan viestiä tarvitse selvittää päästäkseen selville sen merkityksestä. Jos vaikka sodassa vihollisen alueelta tulee samankaltainen viesti samalta suunnalta ennen pommikoneiden hyökkäystä, voidaan tulevilla kerroilla viestin ilmestyessä laittaa puolustus kuntoon saman tien.

3.7 Turvallinen algoritmi

Ainut tapa saada algoritmista turvallinen on analysoida se läpikotaisin. Ainut tapa tehdä tämä on julkaista algoritmi ja antaa se kaikkien maailman salausalgoritmeista kiinnostuneiden tutkittavaksi. Algoritmit ovat todella helppoja toteuttaa ja käyttää, mutta äärimmäisen monimutkaisia analysoida. Niiden turvallisuus ei perustu niiden salaamiseen, vaan niissä käytetyn avaimen laatuun ja salaisuuteen.

Salainen algoritmi ei koskaan ole turvallinen, ei koskaan. Tämä on helppo ymmärtää käänteisesti: mistä voit tietää, että salainen algoritmisi, jota käytät vaikket sen toiminnasta mitään tiedä, ei sisällä takaporttia[2] tai algoritmillista virhettä? On olemassa joukoittain julkisia algoritmeja, jotka, vaikka ovatkin olleet kaikkien nähtävillä, pystyvät silti salaamaan tiedon niin, ettei sitä ilman avainta voi avata. Miksi ei käyttäisi näitä, sillä niiden tiedetään olevan virheettömiä?

3.8 Eheys

Eheys tarkoittaa viestin muuttumattomuutta. Viesti ei ole muuttunut, eikä sitä olla muutettu lähetyksen aikana. Eheyttä voidaan myös käyttää viestin varmentamiseen.

Kenties helpoin tapa tiedon eheyden varmistamiseen, jos mahdollista, on salata tieto. Salausalgoritmien tulisi olla

1. Aikaa ja energiaa saatetaan itseasiassa tarvita enemmän kuin maailmankaikkeudessa on tarjolla.
2. Takaportti on algoritmiin kehitetty ominaisuus, jota hyödyntämällä voidaan käyttää algoritmia ilman oikeaa avainta.

```
#!/usr/local/bin/python
from sys import*; from string import *;
t,x,y,j,s,a=range(256),0,0,0,1,argv[1]
k=(map(lambda b:atoi(a[b:b+2],16), range(0,len(a),2))*256)[:256]
for i in t[:]:
  j=(k[i]+t[i]+j)%256; t[i],t[j]=t[j],t[i]
  while(s):
    s=stdin.read(1); l,x=len(s),(x+1)%256;y,c=(y+t[x])%256,l and ord(s);
    (t[x],t[y])=t[y],t[x]; stdout.write(chr(c^t[(t[x]+t[y])%256])[:l])
```

Kuva 1: RC4 salausalgoritmi Pythonilla.

toiminnaltaan sellaisia, että jos yksikin bitti salatussa tiedossa vaihtaa tilaa, ei viestiä pystytä enää avaamaan.

Joskus salaus ei kuitenkaan ole mahdollista. Tällöin voidaan viestistä laskea tiiviste (*message digest, hash value*), joka on käytännössä koko viesti pakattuna yhteen sitä kuvaavaan tavurykelmään, joka on kooltaan huomattavasti pienempi kuin itse viesti. Jälleen, jos yksikin bitti viestissä vaihtuu, on tiivisteen oltava tyystin erilainen. Selvää on, että voi olla olemassa kaksi eri viestiä, joiden tiiviste on sama. Tiivistysfunktiot on kuitenkin tarkkaan kehitetty sellaisiksi, että saman tiivisteen tuottavan kahden erillaisen viestin mahdollisuus on minimaalinen. Esimerkkeinä tällaisista algoritmeista mainittakoon MD5 ja SHA.

3.9 Aitouden varmentaminen

Salaisen viestin vastaanottajan on hyvä tietää, keneltä viesti tulee. Myös avaimen saaja halunnee varmistua, että avain on juuri oikean henkilön avain. Varmentamisen avulla tämä onnistuu. Varmentaminen on myös digitaalista allekirjoittamista. Lähettäjä merkitsee lähettämänsä viestin niin, että siinä on hänen oma leimansa. Jos tuota leimaa pyritään muuttamaan, tai se poistamaan, osoittaa varmentaminen vilpillisyyden. Arvatenkin varmentamisessa käytetään apuna salaamista ja tiivistettä.

Yksinkertainen tunnistusmuoto on salainen **salasana**. Jos salasana pysyy salaisena ja vain asianosaisten hallussa, tiedetään aina viestin avaamisen onnistuessa, että viesti on tullut joltain avaimen haltijalta.

Jos käytetään asymmetrista salausta, on tämä jo kohtalaisen varma tapa tunnistaa lähettäjä, koska eihän kukaan muu kuin yksityisen avaimen haltija voi lähettää sillä salattua viestiä. Yksityinen avain pysyy vielä vain salaajan hallussa, joten avaimen jakamisesta ei tule tietoturvariskiä, kuten aiemmassa tapauksessa.

Salasanan käytön toinen sovitus on **jaettu salaisuus**. Vaikka jokaisella asianosaisella olisi omat salausavaimet, on heillä yksi yhteinen jaettu salaisuus. Jos tuo salaisuus esiintyy avatussa viestissä, on lähettäjän henkilöllisyys taas astetta varmempi.

Viestin aitouden varmentaminen, ts. viestin todentaminen, voidaan toteuttaa digitaalisen allekirjoituksen avulla seuraavasti: Alice laskee viestistään tiivisteen, jonka hän salaa yksityisellä avaimellaan. Hän lähettää viestin ja salatun tiivisteen Bobille, joka aukaisee Alicen julkisella avaimella tiivisteen, laskee viestistä oman tiivisteen ja vertaa näitä kahta. Jos ne ovat samat, on viesti Alicelta, eikä se ole muuttunut matkalla.

4 Turvalliset ohjelmistot

Seuraavassa käydään läpi yleisimpiä ohjelmistossa tietoturvattomuutta aiheuttavia virheitä, sekä pohditaan toimintojen käyttöoikeuden rajoittamista.

4.1 Yleisiä virheitä

Kun ohjelmien turvallisuus kyseenalaistuu, taustalla on yleensä puskuriylivuoto, muistiin jääneet salaisuudet tai liian tehokkaat ohjauskielen ominaisuudet. Puskuriylivuoto on näistä varmaankin yleisin. Siinä puskurin ylittymistä ei tarkkailla, joten pahantahtoinen koodi pääsee muuttamaan pinossa puskurin lähellä olevaa rutiinin paluuosoitetta mieleisekseen. Aiheesta tarkemmin osoitteessa `http://www.linuxjournal.com/article.php?sid=6701`.

Muistiin jääneet salaisuudet voivat aiheutua paitsi puskuriylivuodosta, myös siitä, ettei ohjelmoija ymmärrä sotkea arkaluonteista tietoa sisältänyttä muistialuetta. Ohjelmointikielten muistinvapautusfunktiot eivät yleensä puhdista käyttämäänsä muistialuetta, vaan ohjelmoijan on itse muistettava tehdä tämä, vaikkapa kirjoittamalla satunnaista dataa varattu muistialue täyteen ennen sen vapauttamista.

Liian tehokkaista ohjauskielistä on monilla varmasti ikäviä kokemuksia. Varsinkin eräs amerikkalainen suuri ohjelmisto- ja käyttöjärjestelmätoimittaja on aikojen saatossa tullut kuuluisaksi liian tehokkaista ohjauskielistä toimisto- ja sähköpostiohjelmissaan. Varovainen on myös syytä olla ohjelmien kanssa, jotka liian kevyin perustein antavat käyttäjän kirjoittaa käyttöjärjestelmäkomentoja ohjelman suoritettavaksi. Java-ohjelmoijatkin voisivat harkita useampaan kertaan ennen `java.lang.Runtime`-olion metodien käyttöä.

Miten voi sitten tietää, mikä ohjelma on tietoturvallinen ja mikä ei? Yksi keino on seurata ohjelmistojen tietoturvallisuutta valvovien yhteisöjen tiedotteita, esimerkkinä suomalainen CERT-FI `http://www.ficora.fi/suomi/tietoturva/index.htm` ja kansainvälinen CERT `http://www.cert.org/`. Toinen hyvä keino on luottaa avoimeen ohjelmistoon. Avoimet ohjelmistot ovat kaikkien tutkittavissa ja maailmasta löytyy yllättävän paljon ihmisiä, jotka haluavat olla ehdottoman varmoja käyttämiensä ohjelmien turvallisuudesta.

Viimeaikoina on tapahtunut kolme eri avoimen ohjelmiston turvallisuuden kyseenalaistumista, mutta näistä on selvitty säikähdyksellä, osiltaan juuri ohjelmistojen avoimuuden takia. Tapaukset ovat salausohjelmisto GnuPG:n ElGamal-algoritmin toteutuksen ongelmat [1], sekä Linux-jakelupaketti Debianin palvelimiin tapahtuneen hyökkäyksen [2] yhteydessä paljastunut virhe Linuxin kernelissä [3], virhe, joka on jo korjattu.

4.2 Käyttöoikeuden rajoittaminen

KESKEN

Tietoturvaa on myös käyttäjän käyttöoikeuksien rajaaminen vain tälle sallittuihin operaatioihin. Yksi tapa rajata käyttöoikeuksia on **käyttöoikeuslistat** (*access control list*). Ennen operaation suorittamista tarkistetaan käyttöoikeuslistasta, onko käyttäjällä oikeus operaatioon.

Käyttöoikeuslistojen ongelmana on jatkuva varmentaminen. Aina ennen operaation suorittamista on varmistettava, että asiakas todella saa suorittaa operaation. Tälle vaihtoehtoinen muoto on **kykenevyys** (*capability*), jolloin asiakas voi yksinkertaisesti suorittaa vain niitä operaatioita, joita hän kykenee suorittamaan. Hieman ontuva analogia on auton avain: jos Alice antaa autonsa avaimet Bobille, voi Bob ajaa autoa, mutta Eve ei, koska hänellä ei ole Alicen auton avaimia. Kykenevyys on vanha teknologia, mutta valitettavasti jäänyt käyttöoikeuslistojen ja vastaavien teknologioiden varjoon.

5 Tietoturvan ongelmat

Tietoturva ei ole ongelmatonta. Sen ylläpito kuluttaa aikaa ja resursseja. Saattaapa jopa käydä niin, että tärkeä tieto pysyy salattuna vain sen takia, että salausavain on aikojen kuluessa hukattu. Näiden ongelmien lisäksi on itse turvaongelma: onko tieto todella turvassa tietoturvan soveltamisen jälkeen?

Yleisin tapa turvata tieto on hälventää se – *security by obscurity*. Uskotaan, että se mistä ei puhuta, pysyy salassa. Viittaan vain aiemmin esitettyyn sotaesimerkkiin kumotakseni uskomuksen. Myös salausalgoritmien salaaminen on esimerkki huonosta turvallisuudesta tietoa piilottamalla.

Kaiken toiminnan takana on kuitenkin aina ihminen. Ja ihminen on juurikin tietoturvan heikoin lenkki, eikä sen katkeamiseen tarvita elokuvamaista juonittelua ja salaliittoutumista. Kuinka moni teistä heittää roskiin kuittinsa, jossa on luottokortin numero kokonaisuudessaan? Kuinka moni säilyttää salasanoja kalenterin takakannessa? Kuinka monen salasana on puolison nimi, tai syntymäaika, tai lemmikin nimi, tai tavallinen sana tai lause? Kuinka moni vie salaiset paperit tietoturvaroskakoriin tai polttaa ne? Niinpä.

Tietoturva-alan eräs suurista kuuluisista miehistä on Bruce Schneier. Hän oli erittäin aktiivinen turvallisten julkisten salausalgoritmien puolestapuhuja siihen aikaan, kun Yhdysvallat vielä kuvitteli kaikkien tietoturvallisten salausalgoritmien olevan vain heidän keksimiään ja hallinnoimiaan. Tuohon aikaan liian tehokkaan algoritmin kuljettaminen Yhdysvaltain rajojen ulkopuolelle oli rinnastettu aseiden salakuljetukseen, riippumatta siitä, oliko algoritmi lähtöisin Yhdysvalloista vai ulkopuolelta sinne tuotu! Tämän takia esimerkiksi ennen Yhdysvalloista poistumista tuli kannettavan tietokoneen kiintolevyltä tuhota kaikki salausohjelmat, vaikka ne sinne olisi ennen tuloaan asentanut Yhdysvaltain ulkopuolella.

Schneier sanoi, vapaasti mukaellen: Oli hienoa, että voitimme tietoturvasodan. Nyt kaikilla on oikeus vahvoihin tietoturva-algoritmeihin. Valitettavasti vain se oli väärä sota.

Oikea sota olisi ollut ihmisten kouluttaminen ja tietoisuuden lisääminen tietoturvan mahdollisuuksista ja mahdottomuuksista. Oikea ratkaisu tietoturvan säilyttämiseen ei ole salausalgoritmeissa, vaan aseistetuissa vartijoissa ja piikkilangassa: `http://www.theatlantic.com/issues/2002/09/mann.htm`

6 Luettavaa

Algoritmien puolelta alan perusteos on Bruce Schneierin Applied Cryptography [4]. Schneier on myös kirjoittanut enemmän tietoturvasta yleisesti kertovia kirjoja ja kirjoituksia, paras viite näihin lienee: `http://www.schneier.com/`

Kevyemmältä puolelta suosittelen Neal Stephensonia, ja hänen romaaniaan Cryptonomicon [5]. Vaikka tarina on kuvitteellinen, ovat ongelmat – ja algoritmit – todellisia. Itseasiassa kirjan juonen kannalta tärkeä algoritmi on Schneierin kehittämä ja ilmeisen vahva, vaikkakin fiktiivisessä romaanissa julkaistu.

http://www.cert.org/

7 Yhteenveto

Tietoturva on monimutkainen asia. Jätä se asiantuntijoille.

Viitteet

[1] ??: .
URL `http://lists.gnupg.org/pipermail/gnupg-announce/2003q4/%000276.html`

[2] ??: .
URL `http://lists.debian.org/debian-announce/debian-announce%-2003/msg00003.html`

[3] "Linux kernel exploit", .
URL `http://lists.debian.org/debian-security-announce/debian%-security-announce-2003/msg00212.html`

[4] Bruce Schneier: *Applied cryptography (2nd ed.): protocols, algorithms, and source code in C*, John Wiley & Sons, Inc., 1995, ISBN 0-471-11709-9.

[5] Neal Stephenson: *Cryptonomicon*, Avon Books, New York, 1999.
URL `http://www.cryptonomicon.com/`