Dokumendiformaadid ja nende turvaprobleemid

Jan Willemson, Cybernetica

18. oktoober 2000. a.

1 Digitaalsed dokumendid: andmevahetus ja digitaalsignatuurid

Sedamööda, kuidas ülemaailmsed arvutivõrgud oma võidukäiku alustasid, hakkasid arvutid info digitaalse töötlemise kõrval üha rohkem tegelema ka selle info edasikandmisega. Üks esimesi ja lihtsamaid selle valdkonna teenuseid – elektronpost – elab tänaseni, kuid tavalise e-maili poolt pakutav ASCII-teksti vahetamise võimalus ei rahulda kasutajaid enam ammu. Ka Eestis töötatakse iga päev tuhandete tarkvarapakettidega ning produtseeritakse väga erineva formaadiga faile, milles sisalduvat infot kellegi teisega jagada soovitakse.

Siit saavad aga alguse probleemid. Teadupärast liigub digitaalne informatsioon üle võrgu 1-deks ja 0-deks kodeeritult ning seega ei näe dokumendi kättesaaja muud kui vaid bitijada, mida ta peab mingite vahenditega interpreteerima hakkama. Kust peaks dokumendi saaja teadma, mida temani jõudnud 1-d ja 0-d tähendavad?

Vastuse annavad tavaliselt selle bitijada välised vahendid, näiteks failide laiendid või mime-tüübid, mille alusel operatsioonisüsteem valib dokumendi käsitlemiseks vajaliku rakenduse. Teise näitena võib tuua UNIX-süsteemide file-käsu, mis üritab faili sisu järgi tema tüüpi ära mõistatada, laskmata end laiendist häirida. Mõlemal juhul jääb lahtiseks küsimus, kuidas käsitseda postkasti maabunud bitijada, mille kohta ükski rakendus midagi arvata ei oska. Üheks võimalikuks vastuseks on ainult standardsete ja üldtuntud dokumendiformaatide kasutamine digitaalses andmevahetuses; seda lahendust käsitleme täpsemalt jaotises 2.

2000. aasta märtsis võeti Eesti Vabariigis vastu digitaalallkirja seadus [5], mis lubab elektrondokumendi tema paberkolleegiga õiguslikus mõttes samasse

seisusesse tõsta, võimaldades bitikujul esitatud andmeid sellisena ka signeerida. Vastavatest meetoditest ning mõnedest probleemidest võib huvitatud lugeja leida informatsiooni näiteks Ahto Buldase artiklitest [1] ja [2]. Osutub, et digitaaldokumentide signeerimise seisukohast on dokumentide vaatlemine üldiste bitijadadena eeliseks: me ei pea hakkama välja töötama eraldi meetodeid DOC-, PDF-, Bitmap-, WAW-failide ning ka kõigi teiste kunagi eksisteerinud või kunagi tekkida võivate failiformaatide tarvis. Formaatide *á priori* fikseerimatus põhjustab siin aga hoopis suuremaid probleeme kui lihtsad arusaamisraskused. Neid käsitleme pikemalt jaotises 3.

2 Millises formaadis vahetada andmeid?

Kuna tarkvaratööstus on arenenud turumajanduse kõigi hundiseaduste järgi, ei leidu maailmas hetkel universaalset dokumendivahetuseks kasutatavat formaati, millega kõik rahul oleksid. Iga kasutaja ja iga kontor esitab soovitavale formaadile oma nõudmisi, nendest johtuvalt tulekski valida saadaolevaist võimalustest kõige paremini sobiv dokumenditüüp.

Mille alusel otsus langetada? Tiina Tamme käsitleb oma magistritöös [3] elektroonilise dokumendivahetusega seotud probleeme ning toob muuhulgas välja järgmised nõudmised.

1. Elektrooniliseks andmevahetuseks kasutatav formaat peab olema avatud.
2. Ta peab olema sõltumatu ühest kindlast operatsioonisüsteemist, rakendusprogrammist või tootjast.
3. Ta peab lubama kasutada graafikat ja muid multimeedia võimalusi ning tagama dokumentide hea kujunduse.
4. Formaat peab võimaldama dokumendis sisalduva info mugavat kasutamist (otsing, indekseerimine) ja redigeerimist.
5. Fail peab olema mahult kompaktne.
6. Formaati peab saama lihtsalt integreerida olemasolevatesse süsteemidesse ja tööprotsessidesse.

Nendest kriteeriumitest lähtudes võrdleb töö autor erinevaid formaate (HTML, XML, PS, PDF, TEX, RTF, ASCII) ning töötab välja oma (võibolla küll subjektiivsed) eelistused kasutatavate dokumenditüüpide kohta. Tema hinnangul sobivad enamikul juhudel rakendamiseks HTML- ja PDF-failid.

Loomulikult ei saa jätta märkimata, et *de facto* kasutatakse hetkel digitaalses dokumendihalduses väga laialdaselt Microsoft Wordi DOC-faile. Põhjuseid, miks nii Tiina Tamme kui käesoleva artikli autor seda formaati ei soovita, on mitmeid.

1. Tegu on salajase definitsiooniga, ühe firma poolt monopoliseeritud formaadiga. See asjaolu muudab kõik DOC-formaadi kasutajad sõltuvaks Microsofti tegevusest, samuti ühest kindlast operatsioonisüsteemist.

2. Ainus legaalne võimalus DOC-formaadi töötlemiseks on osta Microsofti toode (StarOffice'i jt import-eksport-funktsioonid ei saa DOC-i definitsiooni salastatuse tõttu kunagi perfektseteks). Seega sunnib DOC-failide laialisaatja kõiki oma partnereid Microsofti tooteid ostma. Samuti sunnivad oma potentsiaalseid kaastöölisi sellele sammule ajakirjandusväljaanded (sh nt A&A), kes nõuvad kaastöid DOC-formaadis.

3. Word soodustab WYSIWYG-redaktorina dokumentide halba vormistust (tühikutega tabuleerimine, pealkirjade vormistamine mitte *header*ina, vaid käsitsi kirja suurendades jne). Põhjalikumalt võib WYSIWYG-ideoloogia puudustest lugeda Conrad Taylori suurepärasest artiklist [6]. Lisaks saab sealt hea ülevaate arvutitüpograafia arengust üldse.

4. Tänu omadusele kõiki parandusi kaasas kanda võivad DOC-failid väga mahukaks muutuda. Kas Teie, hea lugeja, olete saanud kaheleheküljelise, vaid lausteksti sisaldava Wordi DOC-i suurusega 600 kilobaiti? Käesoleva artikli autoriga on seda juhtunud korduvalt.

5. Microsoft Wordi erinevad versioonid ühilduvad sageli halvasti, nii pole uuema Wordi DOC vanema versiooniga loetav. Microsofti poole pealt vaadates on tegu kavala turupoliitikaga (kui keegi ostab uuema Wordi, peavad ka kõik tema partnerid sama tegema), kuid kasutaja seisukohast kutsub esile tõsiseid prbleeme.

6. Siiani ootab lahendust makroviiruse leviku probleem.

Lisaks selgub, et DOC-fail sisaldab mitmeid turvaauke, mis ohustavad tema kasutamist digitaalselt signeeritava formaadina. Neid probleeme käsitleme jaotises 4.

3 Millist formaati signeerida?

Nagu eespool mainitud, käsitletakse elektrondokumente signeerimisel üldiste bitijadadena ning unustatakse ära nende tähendus. Probleemid bitijada interpreteerimisel dokumendina võimaldavad aga mitmeid rünnakuid.

Vaatleme näiteks situatsiooni, kus panga klient Aadu võtab pangalt suure laenu ning signeerib Corel WordPerfectis kirjutatud võlatähe digitaalselt. Kuna Aadu ei soovi raha tagasi maksta, kaebab pank ta kohtusse. Kohtuistungi ajal kontrollitakse võlgniku signatuuri elektrooniliselt – see klapib. Siis avatakse WordPerfecti abil võlatäht. Selle peale tõuseb Aadu hämmastunud näol püsti ning lausub: “Mina ei tea WordPerfectist midagi! Mina programmeerisin ise graafikaprogrammi ning Teie ees olev bitijada on minu programmiga joonistatud põdrapilt!” Ja nagu sellest veel vähe oleks, demonstreerib Aadu tõepoolest graafikaprogrammi, mis uuritava dokumendi laadimisel kuvab ilusa põhjapõdrapildi. Kohus mõistab Aadu õigeks ning pank jääb pika ninaga.

Mida sellise rünnaku (kohalikus folklooris põdrapildirünne, inglise keeli *denial of content attack*) vältimiseks ette võtta? Üks võimalus on kasutatav dokumendiformaat väliste vahenditega (lepingud, eeskirjad, . . .) ära määrata ning leppida kokku, et kogu antud konteksti andmevahetus käib ainult selles formaadis. Loomulikult jääb Aadule sel juhul võimalus eitada mitte dokumendi sisu, vaid konteksti: “See polnud üldse pangale suunatud võlakiri, vaid sünnipäevakaart minu naisele.” Kummatigi sobib see lahendus praktikas üsna hästi suhteliselt väikesele hästi välja kujunenud suhetega kasutajate ringile (ühe firma siseseks või kindlate äripartnerite vaheliseks asjaajamiseks).

Niisuguse lähenemise puhul tekivad aga probleemid siis, kui signeeritava dokumendi potentsiaalne kasutajaskond võib osutuda laiaks nii arvulises kui ajalises mõttes või koguni hetkel mitte teadaolevaks (nt seadus või testament). Siis tuleks kitsas ringis kokku lepitud formaadi asemel kasutada võimalikult levinud, avalikku ja standardset dokumenditüüpi. Parimaks lahenduseks sobib artikli autori hinnangul 7-bitine ASCII formaat, mis on pea ainus kogu maailmas üheselt aru saadavatest formaatidest. Juba 8. biti lisamine tekitab probleeme, sest sümboleid koodidega 128. . . 255 võib interpreteerida erinevalt. Näiteks võib veebilehelt [7] leida erinevad ISO 8859 kooditabelid.

Eelkõige tekkis vajadus paljude kooditabelite järele keelte ning neis kasutatavate sümbolite paljususest. Seda probleemi üritab lahendada ISO standard 10646 (nn Unicode), kehtestades sümbolitele 16-bitise esituse ning võimaldades seega väljendada 65536 erinevat märki, millest peaks piisama kõigile keeltele (sh hieroglüüfe kasutavatele). Samas võib Unicode’i sümbolijadas esineda ka mitmesu-

guseid kontrollsümboleid ning neid võib jälle mitmeti interpreteerida. Tekkivatest probleemidest räägib lähemalt Bruce Schneier artiklis [4].

7-bitise ASCII kooditabeli puuduseks tavakasutaja tarvis jääb tema vähene väljendusvõimsus. Puuduvad elementaarsedki võimalused erinevate kujunduselementide (kirjastiilid, joonised, tabelid jne) loomiseks. Selle probleemi lahendusena on välja töötatud mitmeid kõrgema taseme dokumendikirjelduskeeli, mis kasutavad lähtekoodi loomisel ASCII sümboleid, kuid võimaldavad esitada kuitahes keerulisi kujunduselemente. Näiteks võib tuua juba 15 aastat standardsena püsinud TEXi formaadi, mille kirjelduse esitas TEXi looja Donald E. Knuth monograafias [8].

Sama ideoloogiat kasutab ka viimasel ajal standardiks pürgiv XML (*extensible markup language*) formaat ([9]). HTML-sarnaste juhtsõnadega liigendatakse dokument vastavalt tema sisule (arve number, arve saaja, arve esitaja, arve maht EEKides jne). Loomulikult peavad juhtsõnad olema deklareeritud ja nii käib XML dokumendiga kohustuslikus korras kaasas ka tema tüübidefinitsioon (*document type definition*, DTD). See ideoloogia aitab muuhulgas lahendada põdrapildiründe probleemi, kui nõuda dokumendi signeerimist koos vastava DTDga. Niisiis võiks XML standard (kui ta lõpuks valmis ja valdavaks saab) olla üks tõsisemaid kandidaate signeerimiseks soovitatava dokumendiformaadi kohale.

Samas on XMLi lähtekood käsitsi kirjutamiseks liiga ebamugav ja tema laiema leviku eelduseks saab suuremal või vähemal määral WYSIWYG ideoloogial baseeruvate XML-editoride ilmumine. Siit võivad aga tekkida uued turvaprobleemid. Vaatleme näiteks HTML-formaadis (mis kujutab endast XMLi erijuhtu) esitatud dokumenti, mille lepingupartner Aadu Teile signeerimiseks saadab. HTML-brauseris näeb asi kena välja ning Te otsustate lepingut oma digitaalallkirjaga kinnitada. Dokumendi lähtekoodis sisaldub aga rida

```
<!-- Kingin oma maja Aadule -->,
```

mis koos muu tekstiga signeeritud sai. Hiljem võib Aadu väita, et allakirjutaja võis lähtekoodi uurida ning seal kirjas seisev väljendab Teie vaba tahet. Moraal dokumendiformaadi kohta: kui otsustate kasutada midagi keerulisemat kui 7-bitist ASCIId, tehke võimalikud turvariskid endale varakult selgeks!

4 Millist formaati mitte signeerida?

Peatükis 2 nägime, et elektroonilises dokumendihalduses kasutatav formaat peab rahuldama küllalt kõrgeid nõudmisi: olema redigeeritav, võimaldama graafika ja

multimeedia lisamist ning integreeruma olemasolevatesse süsteemidesse. Ilmselt ei saa laialdasi võimalusi pakkuv dokumendiformaat olla väga lihtne. Juba erinevate kujunduselementide ja kirjastiilide esitamine nõuab üsna kõrgetasemelisi kirjeldusvahendeid, mis praktikas kipuvad endast sageli kujutama tõsiseltvõetavaid programmeerimiskeeli. Nii näiteks ületavad PostScripti võimalused tunduvalt lihtsamate arenduskeskkondade omi ([10]).

Samas tähendab süsteemi keerukuse kasv automaatselt tõsisemate turvaprobleemide teket, sest võimalike rünnakuobjektide arv kasvab. Retooriliselt õhates: kui juba nii lihtne formaat nagu Unicode võib endas riski kätkeda, mis siis veel oluliselt keerulisematest rääkida!

Tõepoolest, osutub, et paljud tänapäeval praktikas kasutatavad dokumendiformaadid võivad sisaldada ohtusid, millest tavaliselt eriti ei räägita, kuid mida tuleks turvalise dokumentide signeerimise keskkonna loomisel arvesse võtta. Järgnevas vaatleme näitena veelkord Microsoft Wordi DOC-formaati, tuues nüüd välja potentsiaalsed ohud juhul, kui seda tahetakse kasutada nt lepingute elektroonilisel allkirjastamisel.

1. Digitaalallkirja tehnoloogia nõuab, et kord signeeritud dokumendist ei tohi hiljem bittigi muuta, vastasel juhul ei õnnestu dokumendi signatuuri hiljem verifitseerida. DOC-faili võidakse aga muuta kasutajalt midagi küsimata, nt rehkendades avamisel ümber dokumendi sisukorra või muutes trükkimisel kellaaja välja. Seega on dokumendi hoidjal kaks võimalust: muuta fail mittekirjutatavaks või seda üldse mitte lugeda.

2. DOC-fail võib sisaldada palju mittenähtavat infot, näiteks salvestatakse *allow fast saves* valiku sisselülitamisel kõik faili kunagi tehtud parandused. Kujutagem korraks ette, et Teie äripartner kasutab üht kunagi valmis tehtud lepingut *template*'ina ning uuendab selles iga kord vaid nimesid, hindu ja kaubaartikleid. Nii võite Teile allkirjastamiseks saadetud lepingu faili binaareditoriga uurides nii mõndagi huvitavat teada saada ... Lahendus – kiireid inkrementaalseid salvestusi mitte lubada.

3. Tänu võimalusele lülitada dokumendi koosseisu Visual Basicus kirjutatud skripte, võib pahatahtlik partner saata Teile dokumendi, mis muudab oma välist kuju vastavalt arvuti kella näidule. Nii signeerite omateada võlakirja 30 dollari peale, kuid järgmisel päeval saab sellest summast 3 miljonit. Selle rünnaku vastu tuleks pooltevahelise lepinguga sätestada skriptide-makrode mittekasutamine ning Wordis neid lubavad valikud välja lülitada.

4. Arvutiomanik saab Wordi kasutamiseks endale ise uusi kirjatüüpe luua. Oletame nüüd, et pahatahtlik kasutaja on koostanud kirja nimega *Times New Roman* ning vahetanud selles tavalisega võrreldes mõned sümbolid (nt plussi miinusega). Lepingupartnerini jõudes on faili binaarkuju küll sama, kuid näeb standardkirjas ekraanil välja sootuks teistsugune. Kohtuvaidluse tekkides võib kurikael protsessile oma arvuti tuua ning selle abil kohtule endale meelepärast dokumenti näidata. Selle rünnaku vältimiseks tuleks nõuda, et iga dokumendiga pandaks kaasa ka kõik kasutatud kirjad.

Loomulikult pole väljatoodud (ja väljatoomata) probleemid omased ainult Wordi DOC-formaadile. Nii näiteks võivad ka StarWriter- ja PostScript-failid sisaldada dokumedi kuju muutvaid skripte, kirjastiilide mitmetimõistetavuse probleem tekib samamoodi PDF-failide puhul. Nagu ülalpool juba mainitud, on sellelaadsed mured veidigi võimekama formaadi puhul paratamatud ning nende lahendamiseks tuleb tarvitada signeerimiskeskkonna väliseid meetodeid.

5 Kokkuvõtteks

Juba esimeste turvaprobleemidega tegelevate tarkvaraproduktide loojad põrkasid kokku ebameeldiva vastuoluga. Ühest küljest nõuavad kliendid tarkvaralt võimalikult suurt lihtsust ja kasutusmugavust, teisest küljest toob aga turvaprobleemide vastu võitlemine endaga paratamatult kaasa mõningase käideldavuse languse. Eks ole me ju kõik kirunud paroole, mis kuidagi meelde jääda ei taha!

Ka dokumendikäitlustarkvara loojad tahavad oma töö tulemusele võimalikult suurt turgu ning on seetõttu sunnitud pingutama loodavate vahendite lihtsuse nimel. Probleemid tekivad siis, kui turvalisus lihtsusele ohvriks tuuakse. Nii peaks iga signeerimisvõimelise tarkvarapaketi ostja end kurssi viima sellega, mil määral tootja turvaprobleemidele mõelnud on ja ega näiliselt võimas süsteem tagauksi ei sisalda.

Kuna Eestis digitaalsignatuuri seadus veel ei tööta, siis järelikult vastavat tarkvara nõudvaid kliente veel ei ole. Kui seadus aga käesoleva aasta 15. detsembril jõustub, ei saa meist enam keegi ilma signeerida oskavate programmideta oma igapäevatööd teha ja siis võib dokumendiformaadi õige valik meid paljudest ohtudest päästa.

Viited

[1] Allkirjad elektroonilistel dokumentidel, Ahto Buldas, A&A 6-1999

[2] Allkirjad elektroonilistel dokumentidel: avalikud võtmed ja nende haldus, Ahto Buldas, A&A 1-2000

[3] Elektrooniline dokumendivahetus: standardid ja probleemid, Tiina Tamme magistritöö, 2000, http://www.cs.ut.ee/~tiina/MT.pdf

[4] Security risks of Unicode, Bruce Shneier, Crypto-Gram, juuli 1999, http://www.counterpane.com/crypto-gram-0007.html#9

[5] Eesti Vabariigi Digitaalallkirja seadus, http://www.tsm.ee/digitaalallkiri.html

[6] What has WYSIWYG done to us?, Conrad Taylor, http://www.ideography.co.uk/library/seybold/WYS_intro.html

[7] The ISO 8859 Alphabet Soup, http://czyborra.com/charsets/iso8859.html

[8] The TEXbook, Donald E. Knuth, Addison-Wesley, 1984

[9] XMLi koduleheküg http://www.xml.org

[10] PostScript language Tutorial and Cookbook, Adobe Systems Incorporated, Addison-Wesley, 1985