TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Informaatikainstituut

RÜNDEPUUDE METOODIKA JA SEDA TOETAV TARKVARALINE RAAMISTIK

Magistritöö

Üliõpilane: Alexander Andrusenko
Üliõpilaskood: 081902IAPM
Juhendaja: Aivo Jürgenson

Tallinn
2010

Deklareerin, et käesolev magistritöö, mis on minu iseseisva töö tulemus, on esitatud Tallinna Tehnikaülikooli magistrikraadi taotlemiseks ja et selle alusel ei ole varem taotletud akadeemilist kraadi.

Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud või (avaldamata tööde korral) toodud autorlus välja põhitekstis.

.....

Alexander Andrusenko

Annotatsioon

Turvameetrikate väljatöötamine on viimastel aastakümnetel olnud oluline infoturbe arengusuund, kuigi turvalisuse mõõdetavus ei ole siiani tõestatud. Käesolevas töös kirjeldatakse turvalisuse kvantifitseerimisega seotud probleeme ning selle majanduslikke aspekte. Lugejale tutvustatakse ründepuude metoodikat, mis on samm edasi turvalisuse mõõdetavaks muutmisel.

Ründepuude metoodika kasutab majanduslikku lähenemist süsteemi turvamisele. Töös tehakse ülevaade metoodika kasutusvõimalustest ja eelistest ning antakse praktilised juhised, kuidas hinnata ründepuude turvaparameetreid.

Kuna kasutatavad arvutusvalemid ja algoritmid on üsna keerulised ja ressursimahukad, siis otsustati ehitada tarkvaraline tööriist, mis oluliselt lihtsustab metoodika rakendamist. Töös kirjeldatakse valminud raamistikku ning võrreldakse seda teiste samalaadsete toodetega. Tähelepanu pööratakse peamiselt raamistiku paindlikuks disainitud arhitektuurile, mis tagab selle laiendatavust ning hallatavust.

Abstract

In the last decades, development of security metrics has been an important research area in the field of information security. However, the measurability of security has still not been proven. This master's thesis describes problems related to the quantification of security and its economic aspects. The reader is introduced to attack tree methodology, which is one step closer to the quantification of security.

Attack tree methodology uses an economic approach to securing systems. The thesis gives a brief review of possible applications and advantages of the methodology and offers recommendations on how to estimate security parameters.

As the computation formulae and algorithms are rather complex and resource-consuming, it was decided to develop a software tool that would simplify employment of the methodology. The framework is described and compared with other similar products. Attention is mainly focused on the architecture of the framework, which is designed to be flexible so as to ensure extensibility and maintainability.

Sisukord

Sissejuhatus

Kui turvaline süsteem on? Kui turvaliseks ta saab, kui rakendada teatud vastumeetmed? Milline süsteem valikust on kõige turvalisem? Millised on kõige nõrgemad kohad? Mida kaitsta? Kui palju investeerida kaitsemehhanismidesse? Need on küsimused, millega puutuvad igapäevaselt kokku firmajuhid, turvaanalüütikud, süsteemiadministraatorid ja muud isikud, kelle ülesannetesse kuulub turvalisuse tagamine või oma varade kaitsmine.

Vastamaks ülal toodud küsimustele, peab meil olema kindel metoodika, mis võimaldaks turvalisust modelleerida ja mõõta ning mida saaks kasutada riskianalüüsi läbiviimisel. Sellest, kui täpne ja süstematiseeritud kasutatav metoodika on, sõltub süsteemi turvalisus. Viimastel aastakümnetel on turvameetrikate väljatöötamine olnud tähtis teaduse arengusuund, kuid turvalisuse mõõdetavust ei ole siiani suudetud tõestada. Siiski on tehtud mõned sammud selle kvantifitseerimise poole ning on loodud metoodikad, mis võimaldavad turvalisust mingil määral hinnata.

Antud töö esimene eesmärk on tutvustada kvantitatiivset riskianalüüsi metoodikat, mis põhineb ründepuudel. Töös selgitatakse, kuidas saab ründepuude abil turvalisust mõõdetavamaks muuta, kus on neid võimalik kasutada ja milles seisneb selle metoodika praktiline kasu süsteemide turvamisel.

Selleks, et ründepuude metoodikat saaks kasutada praktikas, tuleb hinnata teatud turvaparameetreid. Senistes teadustöödes on antud aspektile pööratud äärmiselt vähe tähelepanu, kuigi turvaanalüütiku seisukohast on see kriitilise tähtsusega. Käesoleva magistritöö teine eesmärk on anda mõned soovitused turvaparameetrite hindamiseks.

Ründepuude koostamiseks ja arvutamiseks on vajalik tarkvaraline tööriist, mis seda protsessi lihtsustaks. Praeguseks ajaks ei ole autori teada ühtegi programmi, mis võimaldaks arvutada ründepuid, kasutades piisavalt realistlikke semantilisi mudeleid. Antud töö kolmas eesmärk on ehitada ründepuude metoodikat toetav raamistik, mida saaks kasutada praktilises töös.

Töö on organiseeritud järgmiselt: esimeses peatükis tutvustatakse teema tausta ning kirjeldatakse, miks on turvameetrikate väljatöötamine oluline teaduse arengusuund, millised nõuded peavad olema rahuldatud turvalisuse kvantifitseerimiseks ja milliste piirangutega peab arvestama. Samuti selgitatakse infoturbe seoseid teiste valdkondadega, rõhutades turvalisuse majanduslike aspektide olulisust. Lisaks tutvustatakse riskihalduse protsessi ning selle tähtsust turvaprobleemide lahendamisel.

Teises peatükis kirjeldatakse kasutatavat riskianalüüsi metoodikat ja antakse ülevaade ründepuid puudutavatest teadustöödest. Detailsemalt kirjeldatakse kah-

te teoreetilist ründepuude mudelit, mis on antud töö kirjutamise hetkeks värskeimad ja peegeldavad teadaolevatest semantikatest reaalsust kõige täpsemini. Lisaks vaadeldakse mõlema mudeli arvutusvalemeid ja -algoritme.

Kolmandas jaotises selgitatakse, kuidas kasutada ründepuude metoodikat riskianalüüsi läbiviimisel. Samuti antakse praktilised juhised, kust leida informatsiooni ründepuu turvaparameetrite hindamiseks ning milliseid aspekte tuleb seejuures meeles pidada, mis on märkimisväärne panus ründepuude teadussuunale.

Neljas peatükk on pühendatud valminud tarkvaralise raamistiku tutvustamisele ning sisaldab võrdlust teiste samalaadsete toodetega, tuues välja nende olulisemad puudused. Kirjeldatakse arendusvahendite valimise protsessi ning raamistiku paindlikku arhitektuuri. Samuti näidatakse, kuidas antud programmi laiendada ja edasi arendada. Lisana on esitatud raamistiku lähtekood optilisel andmekandjal koos käivitamis- ja kasutusjuhenditega.

1 Turvalisuse mõõtmine

1.1 Milleks mõõta?

Lord Kelvin on üle 100 aasta tagasi öelnud, et kui objekti mingit omadust ei saa mõõta, siis on teadmised sellest omadusest liiga pinnapealsed [1]. Ka turvalisuse olemuse sügavamaks mõistmiseks on vaja meetodit selle mõõtmiseks – turvameetrikat. Turvameetrika mõiste all peame silmas kvantifitseeritavate turvaparameetrite mõõtmistel põhinevat mõõtesüsteemi, mis võimaldab leida, kui hästi vastab süsteem oma turvanõuetele [7].

Wayne Jansen artiklis Directions in Security Metrics Research [5] jaotab turvameetrikate kasutusstsenaariume kolme klassi:

1. Strateegiline tugi. Turvaparameetrite hindamine aitab strateegiliste otsuste vastuvõtmisel, näiteks ressurside planeerimisel, toodete ja teenuste valimisel.

2. Kvaliteedikontroll. Meetrikate kasutamine kogu tarkvaraarenduse elutsükli ajal aitab elimineerida turvanõrkusi, mõõtes turvataset, uurides süsteemi riske, analüüsides leitud turvaauke.

3. Taktikaline ülevaade. Süsteemi turvalisuse monitoorimine annab ülevaate kogu süsteemist, aitab kontrollida süsteemi vastavust turvanõuetele, mõõta kaitsemehhanismide efektiivsust, hallata riske ja identifitseerida parendamist vajavaid protsesse või süsteemi osi.

Seega peab turvameetrika võimaldama üheselt määrata, kui turvaline süsteem on, millised on selle kõige nõrgemad kohad, kui efektiivne on mingi kaitsemeetmete komplekti rakendamine ja kui palju on mõtet investeerida. Täpsete turvameetrikate olemasolu võimaldaks süsteeme paremini kaitsta – võtta vastu ratsionaalseid turvaotsuseid ja investeerida majanduslikult põhjendatud turvameetmetesse, mis annaksid konkreetseid turvagarantiisid.

Turvameetmete efektiivsuse mõõtmist ei tohi alahinnata. Kaitsemehhanismide rakendamisel võib tekkida illusioon, et süsteem muutub turvalisemaks, kuid tegelikult ei ole need mehhanismid efektiivsed. Seda seletab järgmine näide: majaomaniku ukseluku võtmel on 5 muuki, millest igaühel on 10 võimalikku positsiooni. See tähendab, et antud lukul on kokku 100 000 võimalikku võtit. Kui sissemurdja hakkab kõiki võtmeid läbi proovima, kusjuures igale võtmele kulub 5 sekundit, siis võtmete läbiproovimine võtab 69 tundi aega. Siis otsustab majaomanik paigaldada turvalisem lukk, mille võtmel on 7 muuki ja igal muugil 12 erinevat positsiooni. Kõigi võtmete läbiproovimine võtaks aega umbes 3 aastat.

Kas selle luku paigaldamine tõstab maja turvalisust? Kui majal on näiteks kaitsmata aken, siis vastus on eitav, kuna ükski sissemurdja ei hakka kõiki võtmeid läbi proovima, vaid siseneb läbi akna [10, lk 103].

Turvameetrikate väljatöötamine on viimasel ajal olnud oluline teaduse arengusuund. USA INFOSEC Uuringute Nõukogu (*INFOSEC Research Council*) publitseeris 2005. aastal Raskete Probleemide Nimekirja (*Hard Problems List*), kus turvameetrikate arendamine on kuulutatud tähtsaks uuringute suunaks [3]. Informatsiooni Infrastruktuuri Kaitse Instituut (*The Institute for Information Infrastructure Protection*), mis on erinevate infoturbega tegelevate institutsioonide konsortsium, nimetab samuti turvameetrikate loomist üheks neljast prioriteedist järgmiseks 5-10 aastaks [4].

1.2 Miks turvalisuse mõõtmine on raske?

Turvalisuse mõõtmine ei ole triviaalne probleem. Seda raskendab asjaolu, et absoluutset turvalisust ei ole olemas. Iga süsteem on turvaline vaid mingite konkreetsete rünnete või ohtude suhtes. Näiteks planeedi hävimise vastu ei ole kaitstud ükski sellel planeedil füüsiliselt paiknev süsteem. Seega, üritades mõõta süsteemi turvalisust, peab tegema teatud eeldusi võimalike ohtude kohta. Ülejäänud ohtude realiseerumine on jääkrisk, mida ei maandata.

Samuti peab tegema eeldusi potentsiaalse vastase ja tema käitumismudeli kohta. Süsteem on täpselt nii turvaline, kui raske on seda murda mingil konkreetsel ründajal, ja kaitsestrateegia erinevat tüüpi vastaste puhul on tüüpiliselt erinev. Seega tuleb vaadelda süsteemi ja selle kaitsemehhanisme ründaja vaatepunktist. Selleks peab aga kokku leppima, mis tüüpi ründaja vastu süsteemi kaitstakse [6].

Turvagarantiide andmine on raske ülesanne ka siis, kui on olemas täpsed meetrikad ning vajalikud eeldused on paigas. Isegi siis, kui on välja arvutatud, et süsteem on turvaline, jääb alati oht, et valitud eeldused ei olnud realistlikud ning süsteemi võivad ikkagi jääda kriitilised nõrkused. Seega, turvameetrikate olemasolu ei võimaldaks tõestada süsteemi turvalisust üldiselt, vaid ainult mingis kontekstis mingitel tingimustel vastavalt valitud mudelile.

Ka siis, kui teoorias on tõestatud, et süsteem on turvaline mingis mudelis, ei pruugi see praktikas vastata tõele. Teooria töötab kõige paremini ideaalsetes laboritingimustes idealiseeritud mudelitel. Päriselus on tingimused oluliselt keerulisemad, alati võivad avaldada mõju varjatud parameetrid, mida süsteemi modelleerimisel ei arvestatud. Samuti ei käitu tihtipeale ründajad vastavalt valitud mudelile. Seega on naiivne loota, et teoreetiliselt turvaline süsteem on päriselus tingimata turvaline [10, lk 133].

Turvalisuse mõõtmis teeb tihti ebatäpseks ka inimfaktori olemasolu. Paljude

meetrikate rakendamiseks peab eelnevalt hindama teatud parameetreid – näiteks varade väärtusi ja rünnete õnnestumise tõenäosusi. Need eksperthinnangud võivad aga erinevate hindajate puhul oluliselt erineda sõltuvalt spetsialisti kogemustest ja informatsiooni hulgast, mida ta omab antud süsteemi ja selle konteksti kohta. Infojaotus on tihtipeale ebaühtlane ning ei saa eeldada, et erinevatel turvaanalüütikutel on täpselt samad andmed ja teadmised. Seega on paljude meetrikate üks ülesannetest inimfaktori minimeerimine. Ideaalis ei tohiks arvutuste tulemus sõltuda inimesest, kes neid arvutusi teeb.

Informatsiooni asümmeetrilisus leiab aset ka ründajate ja kaitsjate vahel. Ründaja on mingis mõttes eelistatud seisus – ta võib rünnata mida iganes, kuid kaitsja peab kaitsma kõike. Kaitsjal on küll tüüpiliselt rohkem infot süsteemi kohta, kuid teisest küljest ei tea ta, kui suur osa sellest infost on olemas ka ründajal. Otsustusteooriat asümmeetrilise informatsiooni puhul on palju uuritud majandusteaduses, kuid infoturbealal on see veel vähe arenenud [8].

Kuigi turvameetrikate väljatöötamise suunas on viimastel aastakümnetel tehtud palju tööd, siiski pole ülal toodud probleemide ja kitsenduste tõttu siiamaani suudetud tõestada, et turvalisus on otseselt mõõdetav [9]. Vaatamata sellele on olemas teatud metoodikad, mis võimaldavad uurida ja hinnata süsteemi turvalisust. Järgmises alajaotises vaadeldakse majanduslikku lähenemist turvaprobleemide analüüsile.

1.3 Turvalisuse majanduslikud aspektid

Viimasel ajal on leitud, et süsteemi turvamisel on tähtsal kohal majanduslikud aspektid. Sageli soovitakse süsteemi turvalisust väljendada selles, kui palju selle murdmine maksma läheb. Ründe keerukus on väljendatav rahalises ekvivalendis. Turvaaugu leidmiseks võib vastane seda ise otsida või osta see turvanõrkuste turult (vt jaotist 3.4.4) – mõlemad variandid maksavad raha. Ründe teostamiseks on tihti vaja osta teatud vahendeid või tööriistu. Botneti-ründe tellimisel on samuti kindel tunnihind. Lisaks peab vastane arvestama majandusliku kahjuga, kui ta jääb oma tegevusega vahele. Iga tõsisema ründe sooritamiseks peab vastane kulutama ressursse.

Sellest tulenevalt hakati kasutama CTB (*Cost To Break*) meetrikat, mis väljendab vähimat rahalist kulu iga ründaja jaoks turvaaugu leidmiseks ja ärakasutamiseks. Seda on kasutatud näiteks krüptosüsteemide tugevuse hindamiseks – arvutati, kui palju kulub ründajal raha krüptosüsteemi murdmiseks, ostes kõige võimsamaid (ja odavamaid) vahendeid ning kasutades kõige efektiivsemaid teadaolevaid tehnikaid [11].

Ka turvameetmete valimisel on tähtsad majanduslikud tegurid – nende hinna

ja efektiivsuse suhe. Kuna kaitsemehhanismid on tihti kallid, siis nende ratsionaalne valik on firmajuhtide jaoks oluline küsimus. Üldjuhul ei soovita ressursse kulutada meetmetele, millega saavutatav kokkuhoid on väiksem kui tehtud investeeringud [2]. Ei ole ratsionaalne kulutada 10 000 krooni seifi peale, et kaitsta varasid, mille väärtus on 1000 krooni, kuna oodatav kasu on väiksem kui tehtud kulutused. Sellise kaitsemehhanismi rakendamine ei ole majanduslikult põhjendatud.

Andmeturve ja majandus on omavahel tihedalt seotud. Paljud turvaprobleemid on paremini seletatavad mikroökonoomika terminites - moraalirisk, asümmeetriline informatsioon, võrguefekt jne [8]. Moraalirisk infoturbes on situatsioon, kus isikud, kes tegelevad süsteemi turvamisega, ei ole need, kes selle eest vastutavad. Kui nad ei vastuta enda isikliku varaga, siis kiputakse võtma liiga suuri riske või valima kaitsemehhanisme liiga lohakalt. Selle tulemusena valitakse ebaefektiivsed turvameetmed.

Asümmeetrilise informatsiooni teooria, mille tuntuim näide on rämpskaupade turg (*Market For Lemons*), selgitab, miks on turul palju ebaturvalisi tooteid – ostjad ei ole suutelised eristama turvalisi tooteid ebaturvalistest. Kui klient ei ole võimeline hindama toote kvaliteeti, siis ei ole tootja motiveeritud kvaliteedi tõstmises [8].

Võrguefekt on olukord, kus mingi võrgu väärtus kasvab võrdeliselt selle kasutajate arvuga. IT alal kehtib see tarkvaratoodete puhul – suurem ühe firma toodete kasutajate võrk on iga kasutaja jaoks kasulikum. Sellest tulenevalt ei arvestata infoturbetoodete valimisel tihtipeale sellega, kui efektiivne ja kvaliteetne mingi produkt on, vaid sellega, kui palju kliente on teinud sama valiku [12]. See kõik mõjutab süsteemide üldist turvalisust negatiivselt.

Sellest tulenevalt hakatakse üha rohkem kasutama majandusteadust infoturbes. Riskide haldamisel kasutatakse informatsiooni asümmeetria teooriat. Potentsiaalse ründaja käitumismudeli analüüsid põhinevad tihti mänguteoorial. Paljud spetsialistid hakkavad teadvustama, et infoturve on palju sügavam ja poliitilisem probleem, kui varem arvati. Ainult tehnilistest lahendustest ei piisa, andmeturbe probleemide lahendamiseks peab tegema koostööd majandusteadlaste, juristide, kriminalistide ja poliitikutega [8]. Ka antud töö teises peatükis vaadeldakse mänguteoorial põhinevat riskianalüüsi metoodikat, mille efektiivseks rakendamiseks tuleks kasutada teiste alade spetsialistide kogemusi.

Süsteemi turvalisuse hindamiseks ja selle parendamiseks vajalike kaitsemehhanismide valimiseks kasutatakse riskihalduse protsessi, mille eesmärk on piisava turvataseme tagamine.

1.4 Riskihaldus

Nagu eelnevalt mainitud, absoluutset turvalisust ei ole olemas, kuid see ei pruugi tingimata valmistada probleeme. Näiteks poevargustest tingitud kahjud on USAs hinnanguliselt 10-26 miljardit dollarit aastas, kuid see ei ole põhjus poodide kinnipanemiseks. Supermarketid eksisteerivad edasi ja jäävad ikkagi kasumisse. Krediitkaartide vargustest põhjustatud kahjud on USAs ligi 10 miljardit dollarit, kuid VISA ja MasterCard ei ole turult kadunud [10, lk 383]. On palju süsteeme, mille turvaprobleemid põhjustavad kahjusid, kuid nad töötavad ikkagi piisavalt edukalt.

Turvalisus ei pea olema perfektne, vaid turvariskid peavad olema hallatavad [10, lk 384]. Turvarisk on vaatlusealuse ohu potentsiaal ära kasutada mingi vara nõrkusi ja tekitada organisatsioonile kahju [13, lk 78]. Ohust O põhjustatud turvariski arvutamise valem on järgmine:

$$Risk[O] = \Pr[O] \cdot Kahju[O]\,,$$

kus $\Pr[O]$ on ohu O realiseerumise tõenäosus ning $Kahju[O]$ on ohu O oodatav kahju. Kogu süsteemi turvariski arvutamiseks saab kasutada järgmist valemit:

$$\sum_{O} \Pr[O] \cdot Kahju[O]\,,$$

kus liidetakse kokku kõik riskid, mida määravad kõikvõimalikud ohud [2].

Riskihalduse protsess algab riskianalüüsiga, mis koosneb järgmistest etappidest:

1. Objekti piiritlemine ja varade liigitamine.
2. Varade spetsifitseerimine.
3. Varade hindamine.
4. Ohtude, nõrkuste ja olemasolevate turbevahendite spetsifitseerimine,
5. Turvarikete tõenäosuste hindamine.
6. Oodatavate kahjude hindamine.

Kui riskianalüüs on läbi viidud, siis järgneb sellele ettevalmistus riskide maandamiseks: süsteemi turvatarbe otsustamine, sobivate turvameetmete valimine, nende tasuvuse analüüs, jääkriski suuruse otsustamine. Jääkrisk on niisugune risk, mida

on otstarbekas taluda, püüdmata teda kahandada üha kulukamate kaitsemehhanismidega. Turvaülesanne on oma olemuselt rentaablusülesanne: kaitsemeetmetele tehtud kulutused peavad end tasuma. Riskide maandamiseks rakendatakse valitud meetmed [13, lk 79 ja 106].

Näide riskide haldamisest on suuremate toidupoodide kaalusüsteem, kus puuvilju kaalub klient ise, mitte klienditeenindaja kassas. Seejuures on oht, et klient ei ole aus ja lisab kilekotti puuvilju pärast kaalumist juurde või hoiab kilekotti kaalumise ajal osaliselt käes, et kaal näitaks vähem. Probleemi leevendatakse turvakaameratega, kuid see ei välista rünnet täielikult. On leitud, et kassatöötaja ajakulu vähendamine sellise süsteemi abil tasub end ära ning klientide pettused on aktsepteeritav jääkrisk.

Ka küberturbe alal on olukord samasugune. Serveritel ei ole vaja murdmatuid paroole – on vaja paroole, mis on piisavalt tugevad, et ära hoida suurema osa ründeid, ja mida on kasutajatel samal ajal võimalik meelde jätta. Krediitkaardid ei pea olema perfektselt turvalised, vaid piisavalt turvalised, et hoida rünnete arvu vastuvõetaval tasemel, ja seejuures odavad. Selleks, et äriline tegevus oleks tulus, ei pea vältima kõikvõimalikke riske, vaid nende suurus peab firma jaoks olema optimaalne [10, lk 384]. Optimaalne on seejuures minimaalsete kogukuludele (kahjud + turbekulud) vastav turvatase [13, lk 106].

Nagu ülal mainitud, algab riskide haldamise protsess riskianalüüsiga. Järgnevas vaadeldakse riskianalüüsi metoodikaid.

1.4.1 Riskianalüüsi metoodikad

Riskianalüüsi metoodikad võib tinglikult jagada kaheks: kvalitatiivsed ja kvantitatiivsed. On olemas ka kombineeritud metoodikad ning kaudne riskianalüüs, kuid neid antud töös ei vaadelda. Kvantitatiivse riskianalüüsi käigus hinnatakse varasid, ohte ja muid turvaparameetreid täpsel arvulisel skaalal. Eeldatakse, et igale varale on võimalik leida materiaalne väärtushinnang ning kõik kahjud on väljendatavad rahalises ekvivalendis.

Kvalitatiivne analüüs on mõnevõrra ebatäpsem. Ohtude realiseerimise tõenäosuste, varade aspektväärtuste ja oodatavate kahjude hindamisel kasutatakse jämedat mõneastmelist skaalat. Vajalikele parameetritele määratakse verbaalsed hinnangud, näiteks ”väike”, ”suur”, ”keskmine”, ”oluline”, ”ebaoluline” [13, lk 96]. Sellisel juhul väljenduvad saadud riskide suurused samuti ebatäpsel kvalitatiivsel skaalal.

Kvantitatiivne riskianalüüs on üsna töömahukas ja raskendatud puudulike või ebatäpsete lähteandmete puhul. Õigustatud on ta juhul, kui varade koguväärtus on suur ja kavandatavad turvameetmed kulukad. Vähem turvakriitilistes süsteemides on mõnikord otstarbekam määrata turvariski kvalitatiivselt [13, lk 96].

Vahel osutub, et kvantitatiivse riskianalüüsi käigus ei suudeta mingit parameetrit hinnata täpsemini kui mõneastmelise kvalitatiivse skaala järgi. Selleks, et taolisi andmeid saaks kasutada arvutusvalemites, peab neile kinnitama orienteeruvad arvulised hinnangud. Selleks valitakse hinnangute alam- ja ülempiirid. Näiteks lepitakse kokku, et mõistele "väike" vastab arv 100 ja hinnangule "suur" vastab 1000. Ülejäänud hinnangud valitakse proportsionaalselt, kasutades alam- ja ülempiire.

Selleks, et turvalisus muutuks mõõdetavamaks, on olulisel kohal turvariskide hindamise kvantifitseerimine. Paljude füüsikaliste omaduste hindamine algas läbi kvalitatiivse võrdluse (näiteks "soojem", "külmem") enne, kui sellest sai täpselt defineeritud füüsikaline suurus (näiteks "temperatuur"). Sellist tendentsi on märgata ka turvameetrikate puhul, millest tulenevalt loodetakse, et ka turvalisus muutub tulevikus täpselt mõõdetavaks suuruseks [5].

1.4.2 Ohtude modelleerimine

Riskianalüüsi käigus kasutatakse tihti ohtude modelleerimist. Modelleerimine tähendab keerulisemast objektist lihtsustatud abstraktsioonide (mudelite) loomist, pöörates tähelepanu olulistele detailidele ja jättes kõrvale ebaolulised. Mudelite koostamine on loomulik viis keerulisema probleemi sügavamaks mõistmiseks ja süsteemist ülevaate saamiseks.

Ohtude modelleerimine on tähtis süsteemide turvamisel ja konstrueerimisel. See aitab paremini mõista, kuidas süsteem töötab, ja identifitseerida selle olulisemaid riske. Kui süsteemi ohte ei õnnestu piisavalt täpselt modelleerida, siis ei ole seda võimalik ka õigesti turvata. Paljud turvaprobleemid ongi tingitud sellest, et nende ehitamisel ei suudetud ette näha reaalseid ohte [10, lk 304-305].

Modelleerimist kasutatakse infoturbes üsna laialdaselt. Näiteks on see nõutud Baseli Pangajärelevalve Komitee poolt loodud raamistikus, mis reguleerib pangainstitutsioonide kapitalihaldust [14]. Samuti on see osa Microsoft SDL (*Security Development Lifecycle*) raamistikust [15].

Ohtude modelleerimist alustatakse võimalike ohtude loetlemisega. Seejärel viiakse koostatud nimekiri loetavamale ja ülevaatlikumale kujule. Tavaliselt kasutatakse grupeerimist (näiteks süsteemi osade järgi) või hierarhilisi mudeleid. Üsna loomulik viis on kasutada puu andmestruktuuri, kus komplekssed ohud jaotatakse rekursiivselt lihtsamateks, kuni saadakse piisavalt hoomatavad alamohud. Selle tulemusena tekib kergesti loetav mudel, mis annab süsteemist hea ülevaate.

Üks võimalik lähenemisviis ohtude modelleerimisele on ründajakeskne. Sel juhul vaadeldakse süsteemi ründaja vaatepunktist ning üritatakse hinnata tema ressursse, võimalusi ja eesmärke [15]. Kui kasutada sellist lähenemist koos hierarhilise mudeliga, siis on ohtude modelleerimise tulemuseks ründepuu.

Järgmises peatükis selgitatakse ründepuu olemust ning sellel põhinevat metoodikat. Antakse ülevaade ründepuude evolutsioonist ja selgitatakse selle eeliseid. Detailsemalt kirjeldatakse kahte ründepuude mudelit ning vaadeldakse nendega soetud arvutusvalemeid ja -algoritme.

2 Ründepuude metoodika

2.1 Eeldused

Nagu eelpool mainitud (vt jaotist 1.2), selleks, et saaks analüüsida süsteemi turvalisust, peab eelnevalt tegema eeldusi potentsiaalse ründaja kohta. Vastaseid saab tinglikult jagada nelja klassi: "heatahtlikud", kuulsusest motiveeritud, kasumist motiveeritud ründajad ja küberterroristid.

"Heatahtlikud" (ingl. k. *white hat*) ründajad on need, kes avastades turvaaugu, annavad sellest teada süsteemiadministraatorile. Nemad ründavad peamiselt uudishimust ja naudivad nõrkuste avastamist ja ärakasutamist. Selliste vastaste peamine huvi on leida vigu, dokumenteerimata võimalusi või panna süsteemi tegema midagi, milleks ta ei ole mõeldud. Lõppeesmärgiks on tavaliselt täielik kontroll süsteemi üle, mitte organisatsioonile kahjude tekitamine. Sellised ründajad on süsteemile tihtipeale pigem kasulikud kui ohtlikud, kuigi nad võivad saada ligipääsu konfidentsiaalsetele andmetele.

Kuulsusest motiveeritud ründajad kuuluvad tavaliselt teatud kommuunidesse, kust saavad eduka ründe puhul tunnustust. Selliste vastaste tüüpiline rünne on veebilehtede näostamine, mille käigus jäetakse avalikule leheküljele teate õnnestunud ründest ja oma pseudonüümid. Kuulsusest motiveeritud ründajad tegutsevad tihti gruppides ja võivad tekitada organisatsioonile kahjusid. Samuti on nad kuulsuse nimel mõnikord nõus tegema mõõdukaid kulutusi.

Kasumist motiveeritud vastane on majanduslikus mõttes ratsionaalne. Ta ei ründa süsteemi, kui see on ebaperspektiivne või materiaalselt kasutu. Lisaks valib ta enda jaoks kõige tulusama ründe. Selline ründaja võib olla nii hea- kui ka pahatahtlik olenevalt sellest, milline käitumine on parasjagu sobivam. Tema eesmärk ei ole otseselt kahju tekitamine, vaid oma kasumi saamine. Siiski on see vastane organisatsioonile ohtlik, kuna ründe tagajärjel tekitatakse tavaliselt otseseid või kaudseid kahjusid [16].

Viimane liik on küberterroristid, kes ründavad peamiselt poliitilistel, etnilistel, rassilistel või religioossetel põhjustel. Nende ohvriteks on tavaliselt konkreetsed inimgrupid või terved riigid. Terroristide eesmärk on tekitada ründeobjektile kahju kõikvõimalike meetmetega. Tihtipeale käituvad nad ebaratsionaalselt ja on valmis kulutama märkimisväärseid ressursse või isegi ohverdama oma elu. Sellised ründajad on ilmselt kõige ohtlikumad, ja nende vastu on kõige raskem võidelda, kuna terroristide motiive ja käitumist on raske prognoosida. Nende tüüpiline strateegia on DOS (*Denial of Service*) rünne, mille eesmärk on teenusetõkestus või kogu süsteemi töö häirimine [17].

Antud töö põhineb suures osas kvantitatiivsel riskianalüüsil ning vaatleb kasumist motiveeritud ründajat. Ründajalt ratsionaalse käitumise eeldamine on vii-

masel ajal saanud palju kriitikat. Alessandro Acquisti ja Jens Grossklags [18] toovad välja mõned põhjused, miks selline eeldus ei ole alati mõistlik. Esiteks raskendab ründeotsuseid informatsiooni asümmeetrilisus. Ründajal ei ole tavaliselt täielikku informatsiooni süsteemist ja selle nõrkustest, seetõttu ei ole ta võimeline objektiivselt hindama selle turvaparameetreid. Samuti on mõned parameetrid immateriaalsed ja ründaja ei oska neid kvantifitseerida.

Isegi kui ründajal oleks täielik informatsioon, ei ole ta tavaliselt füüsiliselt võimeline opereerima suurte andmehulkadega. See on tingitud inimese sünnipärasest piiratud ratsionaalsusest ja võimetuses meeles pidada suurt infohulka. Sellest tulenevalt kasutatakse sageli lihtsustatud mudeleid, lähendusi ja heuristikaid.

Kui saaks eeldada, et ründajal on olemas täielik informatsioon ja ta on võimeline opereerima kogu infohulgaga, ei pruugi tema käitumisstrateegia olla ikkagi ratsionaalne. Mõned majanduslikud ja psühholoogilised uuringud on näidanud, et kaod on tüüpilise inimese jaoks absoluutsel skaalal suurema kaaluga kui kasumid. Samuti on inimesel tendents ohverdada pikema perspektiivi suuremaid kasumeid koheselt saavutatava väiksema tulu nimel. Ka selle pärast ei suuda ründaja käituda täiesti ratsionaalselt.

Siiski võib ülal tood väidetele tuua vastuargumente. Näiteks ei tohi unustada fakti, et süsteemi kaitsjad on samasugused inimesed nagu ründajadki, seega on nende mõtlemisviisi mustrid paljuski sarnased. Kui ründaja ei saa olla täielikult ratsionaalne, siis ei saa seda olla ka kaitsja. Kui süsteemi turvaanalüütik suudab end asetada vastase kohale ja mõelda nagu ründaja, siis on tema otsustusstrateegiat võimalik piisavalt hästi prognoosida. Selleks on vaja küll kogemusi, kuid perfektne ratsionaalsus ei ole tingimata vajalik. Tähtis on see, et mõlema osapoole käitumisviis langeks võimalikult palju kokku. Seega võib eeldada, et ründaja on kasumist motiveeritud ja ratsionaalne nii palju, kui inimesel on võimalik.

Antud juhul võib ratsionaalsus olla üsna subjektiivne. Me ei eelda, et vastane valib ründeid alati ainult ootetulu järgi – ta võib vaadelda ka muid parameetreid ja valida neist kõige prioriteetsemad. Ratsionaalsuse all peame edaspidi silmas seda, et ründaja ei käitu juhuslikult, vaid tal on fikseeritud strateegia, millest ta juhindub ründeobjektide valimisel.

Erinevates ründepuude mudelites tehakse erinevaid eeldusi vastase käitumise kohta. Sihitu ründaja juhuslikke otsuseid on praktiliselt võimatu prognoosida, seega on oluline, et tema käitumismudel oleks fikseeritud. Küberterroristide ja teiste ebaratsionaalsete vastaste käitumist on veel liiga raske analüüsida. Edaspidi täpsustame eeldusi ründaja käitumise kohta erinevate mudelite juures eraldi, hetkel aga piisab vaid eeldusest, et ründaja käitumine vastab valitud strateegiale.

Lisaks eeldame, et kõik kasumeid ja kahjusid kirjeldavad suurused on väljendatavad rahalisel skaalal. Mõned allikad vaatlevad lisaks rahalistele ka immateriaal-

seid turvaparameetreid [21]. Selleks aga, et neid saaks kasutada arvutusvalemites, tuleb neid kuidagi kvantifitseerida. Antud töös eeldatakse, et igale parameetrile on võimalik leida rahaline ekvivalent. Jaotises 3.4 antakse praktilised soovitused, kuidas seda teha.

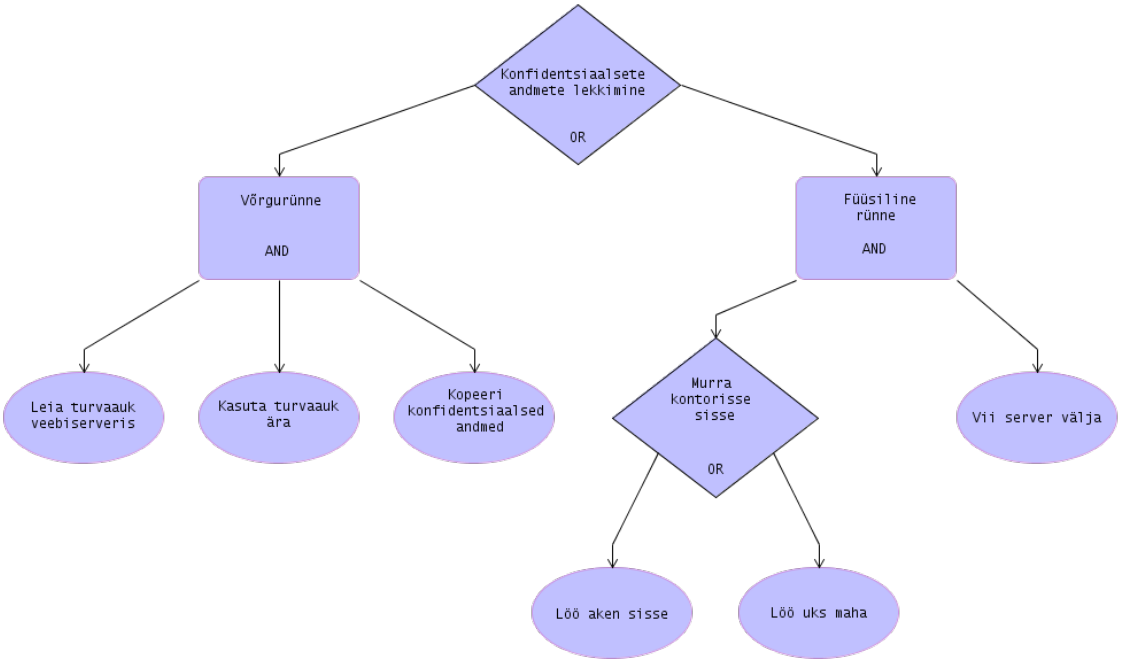
2.2 Ründepuu

Ründepuude riskianalüüsi metoodika pakub formaalset viisi, kuidas kirjeldada süsteemi turvalisust erinevate rünnete vastu. Selle lähenemine on lihtne: analüüs algab primaarse ohu identifitseerimisega, millele järgneb rünnete jaotamine lihtsamateks alamrünneteks [19]. Sõnu "oht" ja "rünne" kasutatakse antud töös sünonüümidena, kuna ründepuu koostamisel peetakse ohtude all silmas ründaja tegutsemisest tingitud ohte e. ründeid. Primaarne oht on niisugune rünne, mis tekitab süsteemile otsest kahju. Näiteks konkureeriva firma poolt tarkvaratoote lähtekoodi varastamine ja turule paiskamine tekitab organisatsioonile otsest kahju. Samas, turvaaugu leidmine veebiserveris ei tekita kahju, seega ei ole ka primaarne rünne.

Jaotamise protsess jätkub, kuni jõutakse alamrünneteni, mille edasine jaotamine ei ole enam võimalik või vajalik. Selliseid alamründeid nimetatakse atomaarseteks või elementaarseteks. Selle protsessi tulemusena tekib puu, mille juur kujutab primaarset ohtu ning lehtedes asuvad elementaarründed. Jaotatavad tipud võivad olla erinevat tüüpi, kuid praktikas kasutatakse tavaliselt AND- ja OR-tippe [19]. OR-tippude alamad on alternatiivsed alamründed, mille abil saavutatakse selle ründe eesmärk. AND- tippude alamad kujutavad erinevaid samme, mis on vajalikud mingi ohu realiseerumiseks [20]. Lühidalt öeldes on ründepuu hierarhiline struktuur, mille iga tipp kujutab mingit rünnet.

Joonisel 1 on kujutatud lihtne ründepuu näide. Juurtipp e. primaarne rünne on märgendiga "Konfidentsiaalsete andmete lekkimine". See on OR-tipp ning tal on kaks alamat: "Võrgurünne" ja "Füüsiline rünne". Seega võivad andmed lekkida kas võrgu- või füüsilise ründe tagajärjel. Võrgurünnet kujutaval tipul on kolm alamohtu: "Leia turvaauk veebiserveris", "Kasuta turvaauk ära" ning "Kopeeri konfidentsiaalsed andmed". Need on selle puu lehed ja kujutavad elementaarründeid. Võrguründe õnnestumiseks peavad realiseeruma kõik kolm alamohtu, kuna antud tipp on AND tüüpi. Füüsilise ründe tipp on samuti AND tüüpi ja tal on 2 alamat: "Murra kontorisse sisse" ning leht "Vii server välja". Kontorisse sissemurdmise tipp on OR tüüpi ning jaotub kaheks: "Löö aken sisse" ja "Löö uks maha". Seega on kontorisse sissemurdmiseks vaja realiseerida vähemalt üks nimetatud alamrünnetest.

Sellisel kujul sobib ründepuu ohtude modelleerimiseks ning annab ülevaate



Joonis 1: Näitepuu

kogu süsteemi turvanõrkustest. Selleks aga, et ründepuude abil saaks mingil moel mõõta või hinnata süsteemi turvalisust, tuleb igale tipule kinnitada mingi hulk turvaparameetreid. Puud, mille tippudel on rohkem kui üks parameeter, nimetatakse mitmeparameetriliseks ründepuuks.

Rünnet saab vaadelda mänguna, mille üks mängija on ründaja. Enne tegutsemist teeb ta valikuid vastavalt fikseeritud strateegiale. Vastane otsustab, kas mängu on mõtet alustada, kasutades turvaparameetrite väärtusi. Vaadeldavateks parameetriteks võivad olla näiteks järgmised [2]:

- ründe õnnestumise tõenäosus p,
- ründe maksumus c,
- oodatav rahaline trahv, kui rünne oli edukas π^+,
- oodatav trahv, kui rünne ei olnud edukas π^-,
- kasum g.

2.3 Lühiülevaade ründepuude ajaloost

Ründepuude idee ei ole sugugi uus. Ohupuid kasutati juba 1980-ndate alguses turvakriitiliste süsteemide uurimisel [22]. 1991. aastal laiendas Weiss ohupuude metoodikat infosüsteemidele [23]. Mõiste "ründepuu" populaariseeris Schneier, avaldades 1999. aastal artikli ajakirjas *Dr. Dobb's Journal* [20]. Schneier soovitas

kirjeldada süsteemi turvalisust läbi võimalike rünnete, analüüsides seda vastase vaatepunktist.

2005. aastal avaldasid Mauw ja Oostdijk formaalse ründepuude semantilise raamistiku, kus nad kirjeldasid mõningaid ründepuude teisendusi. Muuseas tõestasid nad, et ründepuu semantika on kooskõlaline parajasti siis, kui vastava Boole'i valemi viimine ekvivalentsele kujule (näiteks disjunktiivsele normaalkujule) ei muuda selle semantikat ja seega ka ründaja ootetulu [25]. Samasse töörühma kuuluv Opel lõi ka tarkvaralise tööriista ründepuudega opereerimiseks [26]. Paljud autorid mainisid, et päriselus on puu tippudel mitmeid parameetreid, kuid varasemates uurimustes kasutati enamasti näidetes ja arvutustes siiski vaid ühte parameetrit korraga.

Olulise sammu edasi tegid Buldas *et al.*, kes lõid mänguteoreetilise mudeli ründaja otsustusprotsessist, kasutades mitu parameetrit korraga ning tutvustades mõistet "mitmeparameetriline ründepuu". Toodud valemite abil oli võimalik arvutada ründaja ootetulu, kasutades lehtede parameetreid [2]. Oli loodud ka tarkvaraline analüsaator, mis põhines antud mudelil [27].

Ülal mainitud mudeli täiendasid Jürgenson ja Willemson, laiendades parameetrite sisendväärtusi vahemikeks. Selline lähenemine peegeldas reaalsust paremini, kuna turvaanalüütik ei ole tüüpiliselt võimeline hindama parameetreid täpselt. Ebatäpsust on aga võimalik esitada väärtuste vahemike abil. Näiteks võib juhtuda, et analüütik ei suuda anda reaalarvulist hinnangut mingi ründe õnnestumise tõenäosusele, kuid ta teab, et see jääb 0.4 ja 0.6 vahele. Jürgensoni ja Willemsoni mudel võimaldas arvutada ründaja ootetulu, kasutades vahemike abil esitatud sisendeid [24].

Siiski oli Buldas *et al.* mudelil mitmeid puudusi, millest kõige suurem on see, et ta ei ole kooskõlaline Mauw ja Oostdijk raamistikuga. Lisaks kasutati ründaja kasumi parameetrit globaalselt iga alamründe juures, mis tähendas, et iga õnnestunud alamrünne annab vastasele terve kasumi. Samuti eeldati, et ründaja valib OR-tippude alamatest alati ühe, kuigi see ei pruugi tõele vastata, kui alamrünnete riskid ja trahvid on väikesed ning tõenäosused kõrged. 2008. aastal avaldasid Jürgenson ja Willemson ründepuude täpse arvutussemantika, mis vastas Mauw ja Oostdijk raamistikule ega omanud ülal nimetatud kitsendusi [28].

Selle mudeli kasutamine praktikas oli aga raskendatud eksponentsiaalse keerukuse pärast. Esitatud algoritmi optimeeritud versiooni abil sai mõistliku ajaga arvutada kuni 30-lehelisi puid, mida võib praktikas jääda väheks [28]. Samad autorid avaldasid hiljem efektiivse lähendustel põhineva geneetilise algoritmi, mis võimaldas analüüsida ka suuremaid kui 100-lehelisi ründepuid [19].

Praktiliselt kõik eelnevalt mainitud tööd vaatlevad implitsiitselt niinimetatud paralleelmudelit. Eeldatakse, et kõik elementaarründed sooritatakse üheaegselt

ja ründaja valikuid atomaarsete rünnete õnnestumise või ebaõnnestumise põhjal ignoreeritakse. Seetõttu ei ole need mudelid realistlikud. Praktikas on ründajal võimalus muuta plaanitavate tegevuste järjestust või katkestada, kui teatud kriitiline alamhulk rünnetest ebaõnnestub ning edasine tegutsemine ei ole enam mõistlik. Lootusetute katsetega mitte riskimine ilmselgelt vähendab ootetrahve ning suurendab vastase ootetulu [29].

Järgmise sammuna laiendasid samad autorid paralleelmudelit, tuues sisse atomaarsete rünnete järjestuse ja andes vastasele võimaluse samme vahele jätta või katkestada tegevust enne, kui kõik elementaarründed on läbi proovitud. Seda nimetatakse jadamudeliks. Lisaks leiti, et praktikas on puu struktuur liiga kitsendav ning kasulikum on kasutada juurega tsükliteta suunatud graafi, nimetades seda ründegraafiks.

Ründegraafi mõistet on kasutatud ka varem. Swiler, Phillips ja Gaylor uurisid juba 1998. aastal ründegraafide abil võrgu turvanõrkusi, kuid nende ründegraafi semantika erineb allikas [29] kirjeldatud semantikast. Swileri, Phillipsi ja Gaylori mudelis kujutati graafi tippudena süsteemi olekuid, ning servad tähistasid ründaja tegevusi. Samuti vaadeldi vaid ühte parameetrit korraga. Kõige tõenäolisema ründestrateegia leidmiseks kasutati ründe õnnestumise tõenäosust servade kaaluna ja rakendati graafi lühima tee leidmise algoritmi [30]. Käesolevas töös kasutatakse mõistet "ründepuu" ka siis, kui tegemist on graafiga.

Jadamudel on küll paindlikum ja peegeldab reaalsust täpsemini kui eelmised, kuid ka sellel on arenguruumi. Päriselus võib ründaja proovida mingit alamhulka elementaarrünnetest ja olenevalt nende õnnestumisest või ebaõnnestumisest valida järgmisi samme, muuta elementaarrünnete järjekorda või tegutsemise katkestada. Selline täisadaptiivne juht on siiski uurimiseks veel liiga keeruline.

Antud mudel on aga pooladaptiivne, kus eeldatakse, et vastane fikseerib algse järjestuse alamrünnetest ning proovib neid järjest, omades võimalust ründeid ära jätta või tegevust katkestada, kuid algne järjekord siiski säilib. Autorid pakkusid välja ka efektiivse algoritmi ootetulu arvutamiseks fikseeritud alampuu ja rünnete permutatsiooniga, mille halvima juhu ajaline keerukus on $\mathcal{O}(n^2)$, kus n on atomaarsete rünnete arv [29].

2.4 Arvutused ründepuudega

Järgnevas antakse ülevaade ründepuude arvutamiseks kasutatavatest valemitest ja algoritmitest. Vaadeldakse kahte erinevat lähenemist: paralleel- ja jadamudelit. Kuigi paralleelmudeliteks võib nimetada praktiliselt kõiki varasemaid mudeleid, vaadeldakse antud töös ainult artiklis [28] kirjeldatud mudeli valemeid, kuna see on kõige täpsem paralleelmudel ning kooskõlaline Mauw ja Oostdijk raamistikuga.

Jadamudeli arvutusalgoritmid on aga kirjeldatud artiklis [29].

2.4.1 Paralleelmudel

Paralleelmudelis käitub ründaja järgmiselt:

1. Koostab ründepuu ning väärtustab lehtede parameetrid.
2. Vaatleb kõiki ründekomplekte e. alamhulki $\sigma \subseteq \mathcal{X} = \{X_i : i = 1, ..., n\}$, kus X_i on mingi elementaarne rünne. Ründaja arvutab välja oma ootetulu nende komplektide jaoks, mille puhul on primaarne rünne (ründe eesmärk) saavutatav. Neid nimetame kehtestavateks ründekomplektideks.
3. Valib välja ründekomplekti maksimaalse ootetuluga ja teostab sellesse kuuluvad elementaarsed ründed.

Ründepuud võib vaadelda monotoonse Boole'i valemina $\mathcal{F}$ üle muutujate hulga $\mathcal{X}$. Selle valemi kehtestatavad väärtustused $\sigma \subseteq \mathcal{X}$ vastavad ründekomplektidele, mis on piisavad primaarse ründe materialiseerimiseks, st peamine eesmärk on saavutatav. Ründaja ootetulu on siis arvutatav järgmise valemi abil:

$$O = \max \{O_\sigma : \sigma \subseteq \mathcal{X}, \mathcal{F}(\sigma := true) = true\} \,,$$

kus O_σ tähistab ründaja ootetulu, kui ta otsustab proovida ründekomplekti σ, ning $\mathcal{F}(\sigma := true)$ tähistab valemi $\mathcal{F}$ väärtustust, kui kõigi σ muutujate väärtused on $true$ ja ülejäänud on $false$.

Ründekomplekti σ ootetulu O_σ on arvutatav järgmiselt:

$$O_\sigma = p_\sigma \cdot g - \sum_{X_i \in \sigma} E_i \,,$$

kus p_σ on ründekomplekti σ õnnestumise tõenäosus, g on oodatud kasum ning E_i on ootekulu, mis on defineeritud järgmise valemi abil:

$$E_i = c_i + p_i \cdot \pi_i^+ + (1 - p_i) \cdot \pi_i^- \,,$$

kus kasutatud muutujad on defineeritud jaotises 2.2. Ründekomplekti σ õnnestumise tõenäosuse arvutamiseks saab kasutada järgmist valemit:

$$p_\sigma = \sum_{\substack{\rho \subseteq \sigma \\ \mathcal{F}(\rho := \mathsf{true}) = \mathsf{true}}} \prod_{X_i \in \rho} p_i \prod_{X_j \in \sigma \setminus \rho} (1 - p_j) \,.$$

Ründekomplekt σ võib sisaldada liiasust, st võivad eksisteerida alamhulgad $\rho \subseteq \sigma$, mis on samuti piisavad primaarse ründe materialiseerimiseks. Siiski ainult minimaalsetest kehtestavatest ründekomplektidest ei piisa, kuna nad ei pruugi garanteerida maksimaalset õnnestumise tõenöosust, seega peab arvestama kõigi ründekomplektidega. Seetõttu kasutatakse korrutistes tegurit $(1 - p_j)$ nende ründekomplektide puhul, mis kuuluvad alamhulka $\sigma \setminus \rho$.

2.4.2 Jadamudel

Jadamudelis vaadeldakse ründepuu asemel juurega suunatud tsükliteta AND-OR ründegraafi, mis on samuti ekvivalentne monotoonse Boole'i funktsiooniga. Erinevalt paralleelmudelist käitub ründaja seekord järgmiselt:

1. Koostab ründegraafi elementaarrünnete hulgaga $\mathcal{X} = \{X_i : i = 1, ..., n\}$.
2. Valib alamhulga $S \in \mathcal{X}$, mille puhul on primaarne rünne saavutatav, ja vaatleb sellele vastavat alampuud.
3. Valib alamhulga S permutatsiooni α.
4. Arvutab valitud alampuu ja permutatsiooni ootekulu.
5. Maksimeerib ootekulu üle kõigi S ja α valikute.

Antud töös kirjeldatakse ülal toodud nimekirjast täpsemalt ainult neljandat punkti, sest teiste tegevuste jaoks on olemas üldlevinud algoritmid. Kuna antud etapil vaadeldakse ühte alampuud ning ühte alamhulka S, siis üldsust kitsendamata võime eeldada, et $S = \mathcal{X}$. Sel juhul on ründaja käitumine permutatsiooni α puhul järgmine:

Algoritm 1 Teosta rünne

Eeltingimus: Elementaarrünnete hulk $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, permutatsioon $\alpha \in S_n$ ja monotoonne Boole'i valem $\mathcal{F}$

```
for i := 1 to n do
    Võta X_α(i)
    if juurtipu õnnestumine või ebaõnnestumine ei sõltu X_α(i) edukusest then
        Jäta vahele X_α(i)
    else
        Proovi teostada X_α(i)
        if juurtipp õnnestub või ebaõnnestub then
            Peatu
        end if
    end if
end for
```

Ründaja ootetulu saab arvutada järgmise valemi abil:

$$O_\alpha = p_\alpha \cdot g - \sum_{X_i \in \mathcal{X}} p_{\alpha,i} \cdot E_i \,,$$

kus p_α on primaarse ründe õnnestumise tõenäosus ja $p_{\alpha,i}$ tähistab tõenäosust, et rünnet X_i üritatakse teostada Algoritmi 1 käigus. Järgnevas on toodud efektiivne algoritm suuruste p_α ja $p_{\alpha,i}$ arvutamiseks.

Olgu meil ründepuu lehtedega $X_1, \ldots, X_n$, mille õnnestumise tõenäosused on vastavalt $p_i, i = 1, \ldots, n$. Eeldame, et need suurused on sõltumatud ja vaatleme permutatsiooni $\alpha \in S_n$. Defineerime kaks uut parameetrit tipu Y jaoks: $Y.t$ ja $Y.f$, mis näitavad tõenäosust, et on tõestatud, et antud tipp on vastavalt tõene või väär. Iga lehe parameetrid algväärtustatakse $Y.t = p_i$ ja $Y.f = 1 - p_i$ ning teiste tippude parameetrid algväärtustatakse $Y.t = Y.f = 0$. Järgnev algoritm hakkab nende parameetrite väärtusi uuendama ning lõpuks saame $R.t = p_\alpha$, kus R on juurtipp.

Algoritmi käigus arvutatakse ülemtippude parameetreid, kasutades alamate parameetreid. Kui meil on AND-tipp A alamatega B ja C, siis kasutame valemeid:

$$A.t = B.t \cdot C.t, \tag{1}$$

$$A.f = B.f + C.f - B.f \cdot C.f \,. \tag{2}$$

OR-tippude puhul kasutatakse valemeid:

$$A.t = B.t + C.t - B.t \cdot C.t, \tag{3}$$

$$A.f = B.f \cdot C.f. \tag{4}$$

Tõenäosuste $p_{\alpha,i}$ arvutamiseks kasutame Algoritmi 2.

Algoritm 2 eeldab, et kasutatav puu on binaarne. Antud eeldus ei ole suur kitsendus, kuna iga n-aarne puu on transformeeritav binaarseks.

Antud algoritm on efektiivne. Selleks, et välja arvutada $n+1$ vajalikku tõenäosust, läbib ta ühe korra kõik lehed ja igal sammul läbib puu lehest juureni kaks korda. Kuna binaarse puu kõrgus ei saa olla suurem kui selle lehtede arv n, siis on halvima juhu ajaline keerukus $\mathcal{O}(n^2)$. Siiski peab arvestama, et Algoritmi 2 kasutamiseks tuleb genereerida kõik $\mathcal{X}$ alamhulgad ja permutatsioonid.

Järgmises peatükis vaadeldakse ründepuude kasutamist praktikas. Antakse ülevaade sellest, kuidas saab selle metoodika abil monitoorida turvalisuse protsessi, hinnata süsteemi riske ja modelleerida turvameetmete rakendamist. Lisaks antakse mõned soovitused, kuidas hinnata tippude parameetreid, ning kirjelda-

Algoritm 2 Tõenäosuste $p_{\alpha,i}$ arvutamine

Eeltingimus: Ründepuu lehtede hulgaga $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ ja permutatsioon $\alpha \in S_n$

Järeltingimus: Tõenöosused $p_{\alpha,i}$: $i = 1, 2, \ldots, n$

1: **for all** $Z \in \{X_1, \ldots, X_n\}$ **do**
2: $Z.t := 0,\ Z.f := 0$
3: **end for**
4: **for** $i := 1$ to n **do**
5: Leia tee $(Y_0, Y_1, \ldots, Y_m)$ juurest $Y_0 = R$ leheni $Y_m = X_{\alpha(i)}$
6: $p_{\alpha,\alpha(i)} := \prod_{j=1}^{m}(1 - Z_j.a)$, kus Z_j on tipu Y_j naaber ja $a = t$, kui Y_{j-1} on OR-tipp, ning $a = f$, kui Y_{j-1} on AND-tipp
7: $X_{\alpha(i)}.t = p_{\alpha(i)}$
8: $X_{\alpha(i)}.f = 1 - p_{\alpha(i)}$
9: Uuenda tippude $Y_{m-1}, Y_{m-2}, \ldots, Y_0$ parameetrid, kasutades valemeid (1)–(4)
10: **end for**

takse ründepuude metoodika võimalikke arengusuundi.

3 Ründepuude kasutamine praktikas

3.1 Riskide hindamine ründepuude abil

Riskide hindamise protsessi esimene etapp on ründepuu koostamine ja selle elementaarrünnete parameetrite väärtustamine. Parameetrite hindamine on täpsemalt kirjeldatud jaotises 3.4. Kui puu on koostatud ja lehtedele on kinnitatud parameetrite väärtused, arvutatakse see ründepuu välja, kasutades jaotises 2.4 esitatud valemeid ja algoritme. Pärast seda algab saadud tulemuste analüüs.

Arvutusrutiinide tulemus koosneb tüüpiliselt kahest komponendist: kriitiline rünne ja vastase maksimaalne ootetulu. Kriitiline rünne on kõige tõenäolisem vaadeldava ründaja strateegia, mis on väljendatav elementaarsete ohtude alamhulga abil. Seega aitab ründepuu arvutamine identifitseerida, milliseid alamründeid on vastasel kõige kasulikum realiseerida. Kui kaitsja teab, milliseid süsteemi osi vastane kõige suurema tõenäosusega ründab, siis on võimalik projekteerida ja valida turvameetmeid.

Maksimaalne ootetulu näitab, milline on vastase oodatav tulu, mida ta saab kriitilise ründe sooritamisest. Ootetulu väärtuse järgi saab otsustada süsteemi turvalisuse üle. Kui see on negatiivne, siis võib väita, et süsteem on praktiliselt turvaline kokkulepitud eeldustel (näiteks ratsionaalse majanduslikult motiveeritud ründaja vastu). See küll ei tähenda, et edukad ründed on välistatud, kuid vaadeldava vastase jaoks ei ole antud süsteemi ründamine majanduslikult otstarbekas. Positiivne ootetulu tähendab, et vastasel on kasulik süsteemi rünnata, seega on vaja rakendada turvameetmeid. Järgnevas vaadeldakse turvameetmete modelleerimise ja valimise protsessi.

3.2 Turvalisuse protsessi jälgimine

Graafiliste mudelite loomine on üsna loomulik viis keerulisemate probleemide sügavamaks mõistmiseks. Ründepuude metoodika üks osa on mugavate abstraktsioonide loomine, mida saab kasutada riskianalüüsi läbiviimisel. Selle abil on võimalik modelleerida riske, mis annab ülevaate kogu süsteemi turvanõrkustest konkreetsel ajahetkel.

Ründepuude metoodika võimaldab teatud mõttes ka mõõta turvalisust. Selle abil on võimalik välja arvutada, kui turvaline on süsteem püstitatud eeldustel – näiteks ratsionaalse kasumist motiveeritud ründaja vastu. Turvalisuse suurust näitab ründaja ootetulu. Seega peegeldab ründepuu süsteemi turvalisuse hetkeseisu ülevaatliku graafilise mudeli abil.

Rangelt võttes ei saa ründepuude metoodikat nimetada turvameetrikaks, kuna selle tulemus sõltub siiski inimesest, kes analüüsi läbi viib, kuid võib öelda,

et see on üks samm edasi turvalisuse kvantifitseerimisele. Ründepuud annavad võimaluse tuletada keeruliste rünnete parameetreid lihtsamate alarünnete parameetritest, mida eelduste kohaselt osatakse enamasti piisava täpsusega hinnata.

Schneier on korduvalt rõhutanud, et turvalisus ei ole toode, vaid pidev protsess [10, lk 84]. Ründepuu on mugav viis selle protsessi pikemaajaliseks dokumenteerimiseks ja jälgimiseks. Niisuguste tõmmiste perioodiline tegemine, salvestamine ja võrdlemine annab ülevaate turvalisuse dünaamikast ning pakub võimaluse selle monitoorimiseks.

3.3 Turvameetmete rakendamise modelleerimine

Paljud süsteemiülemad toetuvad turvameetmete valimisel oma sisetundele, mis kahjuks ei anna garantiisid, et valitud kaitsemehhanismid on efektiivsed ja majanduslikult põhjendatud [2]. Jaotistes 1.1 ja 1.3 on kirjeldatud, miks on tähtis kindel lähenemine vastumeetmete valimisele. Ründepuude metoodika pakub viisi selle protsessi süstematiseerimiseks.

Selleks koostatakse ründepuu, väärtustatakse elementaarrünnete parameetrid ning käivitatakse arvutusalgoritmid, mis näitavad, milline on süsteemi turvalisuse hetkeseis. Pärast seda modelleeritakse turvameetmete rakendamist. Selleks eeldatakse, et iga meetme kohta on teada, kuidas see mõjutab vastava atomaarse ohu turvaparameetreid.

Meetmete rakendamise modelleerimiseks muudetakse ründepuu lehtede parameetreid vastavalt vaadeldavale kaitsemehhanismile. Vaatleme näiteks turvameetmena aknale metalltrellide paigaldamist. Oletame, et niisugune kaitsemehhanism vähendab läbi akna sissemurdmise tõenöosust 0.5 võrra ning suurendab selle ründe maksumust 10 000 võrra. Sel juhul liidame ründepuu vastava lehe maksumuse parameetrile juurde 10 000 ning lahutame tõenäosuse parameetrist 0.5. Kui see on tehtud, siis arvutatakse puu uuesti üle ja võrreldakse tulemusi.

Tulemuseks võib olla see, et arvutatud ootetulu ei vähene. See aga tähendab, et antud turvameetme rakendamine ei ole efektiivne ega tõsta süsteemi turvalisust. Kui ootetulu vähenes, siis võib antud mehhanismi võrrelda teiste meetmetega. Kui meede A vähendab ootetulu rohkem kui meede B, siis saab järeldada, et variandi A rakendamine on efektiivsem ja majanduslikult põhjendatum.

Niisugust lähenemist saab kasutada nii üksikute mehhanismide kui ka terve komplekti rakendamise modelleerimiseks. Kui vaadeldakse mitmest meetmest koosnevat komplekti, siis muudetakse mitme elementaarründe parameetreid korraga. Protsessi lõppeesmärgiks on üldjuhul niisuguse turvameetmete komplekti leidmine, mis viib ründaja ootetulu negatiivseks minimaalsete kulutustega süsteemi kaitsja jaoks.

3.4 Turvaparameetrite hindamine

Paljud riskianalüüsi metoodikad nõuavad kasutajalt teatud turvaparameetrite hindamist. Kui seda ei ole võimalik piisavalt objektiivselt teha, siis võib teoreetiliselt kooskõlaline ja kasulik metoodika osutuda ebapraktiliseks.

Enamik ründepuid käsitlevaid allikaid teeb eelduse, et vajalikke parameetreid on mingil moel võimalik hinnata, seejuures täpsustamata, kuidas seda teha. Siiski on parameetrite väärtustamine kogu protsessi juures üks keerulisemaid tegevusi ning seda probleemi ei saa ignoreerida. Kui juhtub, et keskmine turvaanalüütik ei ole võimeline hindama parameetreid täpsemalt kui mõneastmelise skaala järgi, siis kaob antud kvantitatiivse metoodika eelised kvalitatiivsete üle.

Järgnevas antakse mõned juhised, mis aitavad väheste kogemustega turvaanalüütikutel väärtuste leidmist mõnevõrra lihtsustada. Kirjeldatakse mõned infoallikad ning nimetatakse teatud aspektid, mida on kasulik arvestada. Antud soovitused ei pretendeeri täielikkusele - turvaparameetrite hindamine toetub ikkagi suures osas spetsialisti kogemustele.

3.4.1 Keskkonnaohud

Tõenäosusparameetrite hindamiseks on mugav kasutada statistilisi andmeid. Kui on teada, mitu korda teatud aja jooksul mingi sündmus keskmiselt toimub, siis on võimalik leida suhteline sagedus. Näiteks, kui on teada, et mingis riigis toimub maavärin keskmiselt üks kord 10 aasta jooksul, siis võib kasutada tõenäosusena, et sellel aastal toimub maavärin, 0.1. See on küll väga lihtsustatud lähenemine, kuid see võib tihti olla piisav.

Ründepuude koostamisel on mõnikord vaja arvestada stiihiliste ohtude realiseerumise tõenäosusi. Nendeks võivad olla keskkonnast või objektide kulumisest tingitud ohud, inimvead, riistvaratõrked jne. Keskkonnaohte, nagu näiteks maavärin, torm jms, on ründepuus mõistlik kasutada ainult siis, kui on põhjust arvata, et niisuguse ohu realiseerumine mingil moel aitab ründajat. Näiteks võib maavärin hävitada mõned organisatsiooni füüsilised turvameetmed, mis annab vastasele võimaluse eduka ründe sooritamiseks.

Kui eelnevalt mainiti, et sõnu ”oht” ja ”rünne” kasutatakse antud töös sünonüümidena, siis siinkohal tuleks siiski ründe- ja ohupuul vahet teha. Ründepuu abil modelleeritakse konkreetse vastase käitumist ja valikuid, ohupuu on aga üldiste ohtude kujutamiseks. Ründepuude metoodikas on stiihilisi ohte eraldiseisvate puu lehtedena raske kasutada, kuna eeldatakse, et ründaja valib neid vastavalt oma strateegiale, mis aga stiihiliste ohtude puhul ei ole definitsiooni järgi vastase võimuses.

Kui on siiski plaanis ründepuus kasutada keskkonnaohte, siis on vaja hankida

nende kohta statistilisi andmeid. Kuna looduslikke ohte on jälgitud aastasadu, siis ei ole nende kohta objektiivsete andmete leidmine kuigi raske. Uurida võiks näiteks ÜRO statistilisi andmebaase [31], kuid nad on üsna üldised ja sisaldavad vähe andmeid keskkonnaohtude kohta. Rohkem infot pakub Kolumbia Ülikooli Maa instituut [32], mis ühendab ligi 850 teadlast ja kus tehakse sadu uuringuprojekte. Selle üheks uurimussuunaks on keskkonnast tulenevate riskide ja ohtude vähendamine, mille käigus publitseeritakse ka palju statistilisi andmeid, mida saab kasutada tõenäosuste ja kahjude hindamiseks. Uurida võiks näiteks 2009. aastal välja antud looduskatastroofide riski vähendamist puudutavat aruannet *Global Assessment Report on Disaster Risk Reduction* [33].

Üsna põhjalik on ka Maailmapanga poolt läbi viidud uuring, kus hinnatakse erinevate looduskatastroofide riske erinevates maailmapaikades ning muuseas antakse ka pikaajalised prognoosid [34]. Samuti tasub jälgida MCEER (*Multidisciplinary Center for Earthquake Engineering Research*) Informatsiooniteenust [35], kuhu publitseeritakse igasuguste samateemaliste uuringute tulemusi ja uudiseid. Lisaks leiab CRED (*Centre for Research on the Epidemiology of Disasters*) [36] hädaolukordade andmebaasist statistilisi andmeid bioloogiliste, tehnoloogiliste, hüdroloogiliste, geofüüsiliste ja teiste katastroofide kohta.

3.4.2 Füüsilised ründed

Ründepuude koostamisel tuleb arvestada lisaks elektroonilistele rünnetele ka füüsilistega, kuna tihti juhtub, et vastasel on lihtsam organisatsiooni kontorisse sisse murda kui pääseda serverisse läbi võrgu. Järgnevas vaadeldakse, kuidas hinnata füüsilisi ründeid puudutavaid parameetreid.

Niisuguste rünnete tõenäosuse hindamiseks on kasulik uurida erinevaid rahvusvahelisi kuritegevust puudutavaid uuringuid. Palju andmeid leiab ICVS (*The International Crime Victim Survey*) programmi tulemitest. Üsna põhjalik on näiteks 2008. aastal koostatud aruanne [37] 2004/2005 a. uuringutest, mis sisaldab statistilisi andmeid erinevate kuritegevuse liikide kohta erinevates riikides, k.a. Eestis. Huvi pakuvad varguste, röövide, pettuste ja korruptsiooni statistikat peegeldavad tabelid. Lisaks on uuritud ka inimeste turvatunnet vastavalt kuritegevuse liigile ning nende rahulolu politsei tööga, millest võiks järeldada kurjategijate avastamise ja vahistamise tõenäosusi. Samuti on kasulik tähele panna, kui suureks hindavad inimesed riski langeda kurjategijate ohvriks.

Kasulikku informatsiooni leiab ka Suurbritannia Kuritegevuse Uuringust [38], mis küll sisaldab andmeid ainult Inglismaa ja Walesi kohta, kuid lisaks muule on seal erinevate turvameetmete võrdlused alustatud ning edukate röövikatsete põhjal. Selle järgi on võimalik teha järeldusi teatud kaitsemehhanismide efektiivsuse kohta.

Selleks, et saaks modelleerida turvameetmete rakendamist ühe konkreetse ründe vastu, eeldati, et mingil moel on võimalik hinnata nende parameetreid: kui palju mingi mehhanism vähendab antud ründe õnnestumise tõenäosust, kui palju ta suurendab ründe maksumust jne. Tihti ei ole see aga sugugi lihtne.

Mõnede meetmete kohta on siiski võimalik leida teatud tootjapoolseid turvagarantiisid. Näiteks seifidel on olemas turvatasemed, mis garanteerivad kindlust mingi konkreetse ründe vastu teatud aja jooksul. Ka ukselukkude kohta on olemas kindlad standardid, mis spetsifitseerivad luku turvaparameetreid ja nimetavad ka ründeviise, mille vastu antud lukk kaitstud on.

Kui kaitsemehhanisme on võimalik omavahel võrrelda, kasutades märgitud turvatasemeid või sertifikaate, siis on neile võimalik kinnitada teatud väärtused, mille võrra nad muudavad konkreetse ründe mingit parameetrit. Seda illustreerib järgnevas toodud näide.

Turvaanalüütik vaatleb paberdokumentide varguse elementaarrünnet ning soovib modelleerida kaitsemehhanismide rakendamist, milleks on kolm seifi märgistega TL-15, TL-30 ja TL-60. Nimetatud märgised tähendavad, et antud seifid kannatavad vastavalt 15, 30 ja 60 minutit rünnet tavaliste vahenditega, nagu näiteks portatiivsed mehhaanilised ja elektrilised tööriistad, käiad, kõrgkiiruslikud trellid jms.

Oletame, et turvameetmete mitterakendamisel on ründe õnnestumise tõenäosus 1 ja TL-60 seifi paigaldamisel on see 0.1. Sel juhul on maksimaalse ja minimaalse võimaliku väärtuse vahe 0.9. Kui lõik $[0.1, 0.9]$ jaotada kolmeks võrdseks osaks, siis iga osalõigu pikkuseks tuleb 0.3 ning võib öelda, et TL-30 seif muudab vaadeldava ründe tõenäosust 0.3 võrra rohkem kui TL-15 ning TL-60 seif on omakorda 0.3 võrra "parem" kui TL-30. Seega võib seifide rakendamise modelleerimisel kasutada selliseid andmeid: TL-15, TL-30 ja TL-60 seifid vähendavad dokumentide varguse tõenäosust vastavalt 0.3, 0.6 ja 0.9 võrra.

3.4.3 Küberründed

Küberrünnete parameetrite hindamisel võib samuti kasutada statistilisi uuringuid, kuigi objektiivseid andmeid antud valdkonnas on üsna raske leida. Põhjalikku statistikat on võimalik leida Suurbritannia kohta. Seda käsitleb näiteks Suurbritannia Küberkuritegevuse Aruanne [39], mis näitab üldist tendentsi küberkuritegevuses.

Oluliselt detailsem on aga PwC 2010. a. uuring "Information Security Breaches Survey" [40], mis sisaldab andmeid Suurbritannia ettevõtete kohta. Uuringus antakse ülevaade ka kasutatavatest kaitsemehhanismidest ning nendele kulutatud ressurssidest, ettevõtete turvapoliitikast, toime pandud rünnetest ning tekitatud kahjudest.

Sarnast informatsiooni leiab ka CSI uuringutest "Computer Crime and Security Survey" [41]. Oluliselt põhjalikum on aga erafirma Verizon küberkriminalistide meeskonna poolt publitseeritud aruanne [42]. Erinevalt teistest sisaldab see lisaks andmeid rünnatavate infovarade, täpsete ründeviiside ja nende keerukuse, ründajate tüüpide ning nende ettevalmistuse kohta.

Ründe õnnestumise tõenäosuse hindamiseks on kasulik uurida ka turvanõrkuste andmebaase. Populaarsed on näiteks SecurityFocus [45] ja selle BugTraq meililist, CVE [46], Secunia [47] jt. Ohu realiseerumise tõenäosuse hindamiseks võib otsida, kas vaadeldava tarkvaratoote kohta on juba olemas raporteeritud turvanõrkusi, mis antud süsteemis ei ole ära parandatud. Turvanõrkuste andmebaaside kohta publitseeritakse ka statistilisi andmeid. Seega on võimalik teada saada, kui tihti leitakse konkreetsest tarkvaratootest turvaauke, mida võib samuti kasutada ründepuu parameetrite hindamisel.

3.4.4 Ründaja kulud ja trahvid

Füüsilise ründe maksumust vastasele on võrdlemisi kerge hinnata – arvesse tuleb võtta näiteks tööriistade hinda ja ajakulu. Kuidas aga hinnata küberründe maksumust? Järgnevas vaadeldakse mõningaid infoallikaid, mida võib selleks kasutada.

Nii kaua, kui on liikvel olnud vallutusprogramme (ingl. k. *exploit*), on eksisteerinud ka turvaaukude turge – teatud kohti, kus saab turvanõrkusi osta ja müüa. Turvanõrkuste ostmise või müümise all peame silmas tehinguid turvanõrkusi ärakasutavate programmide või andmetega. Algusaegadel toimus niisugune tegevus illegaalselt mustal turul, kuid viimasel ajal on tekkinud täiesti seaduslikke kohti, mis toimivad turvaaukude turgudena [43]. Neid on aga erinevaid tüüpe [44]:

1. Äriettevõtted, mis maksavad raporteeritud tarkvaravigade eest. Mõned neist kasutavad seda informatsiooni mingitel oma eesmärkidel, teised aga vahendavad ja müüvad turvanõrkusi edasi. Viimaseid võib nimetada turvaaukude maaklerfirmadeks.

2. Puukide (programmivigade) leidmise väljakutsed ja võistlused, mille käigus kuulutatakse, et iga leitud puugi eest mingis tarkvaratükis makstakse teatud summa.

3. Turvaaukude oksjonid, kus kasutajad müüvad leitud vigu.

4. Lepingud, mille järgi makstakse raha turvanõrkuste leidmisel mingi perioodi jooksul.

5. Küberkindlustus, mida kasutatakse kindlustamaks oma varasid turvaintsidentide tekitatud kahjude eest.

Turvanõrkuste turge kasutavad tihti ka ründajad uute turvaaukude ostmiseks, et neid mingis süsteemis ära kasutada. Niisugust informatsiooni on paljude ründajate jaoks odavam teistelt osta kui endal hankida nende ebapiisavate kogemuste või liiga suure ajakulu tõttu. Seega on mõnikord mõistlik kasutada turvanõrkuste turuhinda ründe maksumuse hindamiseks.

Vaatamata turgude olemasolule ei ole objektiivse maksumuse teadasaamine alati lihtne, kuna ühtset turuhinda ei ole tavaliselt olemas. Paljud kohad ei avalda, mis hinnaga nad turvaauke ostavad, samas üritavad neid müüa võimalikult kallilt. Kuna selliseid küberturge on võrreldes tavaliste turgudega veel võrdlemisi vähe, siis on madala konkurentsi tõttu hinnad üsna ebaühtlased [43]. Selleks, et teada saada mingit orientiirmaksumust, tuleb uurida informatsiooni võimalikult mitmest allikast.

Buldas *et al.* tööl põhinevad mudelid kasutavad enamasti ka ründaja ootetrahve väljendavaid parameetreid, mis on tavaliselt erinevad olenevalt sellest, kas rünne oli edukas või mitte. Eduka ründe puhul on ootetrahv defineeritud järgmise valemina:

$$\pi^+ = p^+ \cdot k^+ ,$$

kus p^+ ja k^+ on vastavalt vahelejäämise tõenäosus ja vastase materiaalne kahju eduka ründe puhul. Ebaõnnestunud ründe ootetrahv on väljendatav järgmise korrutisena:

$$\pi^- = p^- \cdot k^- ,$$

kus p^-ja k^-on vastavalt vahelejäämise tõenäosus ja vastase kahju ebaõnnestunud ründe puhul.

Ründaja kahjude hindamine ei ole aga lihtne, kuna need on iga ründaja jaoks erinevad ning tuleb arvestada ka immateriaalsete kahjudega, mida on samuti vaja kvantifitseerida. Eeldatakse, et iga kahju on väljendatav materiaalsel skaalal, seega tuleb leida sobiv rahaline ekvivalent. Selle hindamiseks võiks uurida karistusseadustikku, kus on antud võimalike trahvide ja vangistuse pikkuse vahemikud.

Lisaks on võimalik leida andmeid selle kohta, milline on olnud keskmine vahi all hoidmise pikkus erinevat tüüpi kuritegude puhul. Sellist statistikat publitseerib Justiitsministeerium [48], samuti leiab andmeid Statistikaameti andmebaasidest [49]. Samuti tuleks arvestada tingimisi karistuse ning kokkuleppemenetluse võimalusega. Kokkuleppemenetluse puhul võib eeldada oluliselt väiksemat või minimaalset võimalikku karistust antud kuriteoliigi puhul [50].

Võimaliku vangistuse puhul arvestab ründaja kahjudena ilmselt ka raha, mida

ta oleks oma tööga teeninud vahi all olemise perioodi jooksul, moraalset kahju ning reputatsiooni kaotamist. Selliseid kahjusid on äärmiselt raske hinnata ning tuleks kasutada jämedaid hinnanguid – näiteks suurusjärgu täpsusega.

3.5 Ründepuude arengusuunad

Ründepuude metoodika ei ole veel täielikult välja arenenud ja selleks, et ta muutuks praktilisemaks, on mitmeid aspekte, mida tuleks edaspidi uurida. Mõned neist puudutavad ründepuude teooriat ja mõned – antud metoodika kasutamist praktikas. Vaatleme esialgu teoreetilise poole arengusuundi.

Jaotises 2.3 sai mainitud, et praegusel hetkel on teadaolevatest ründepuude mudelitest kõige realistlikum jadamudel. Siiski ei peegelda see veel tegeliku ründaja käitumist täielikult, kuna hetkel vaadeldakse vaid pooladaptiivset juhtu. Lisaks ei arvestada võimalusega, et puus võib olla elementaarründeid, mille ebaõnnestumine blokeerib edasisi tegevusi – näiteks vastane jääb vahele ega saa edasi rünnata. Samuti valmistab probleeme arvutusalgoritmide supereksponentsiaalne ajaline keerukus, mille tõttu ei ole praktikas võimalik arvutada piisavalt suuri ründepuid [29].

Seega on üheks arengusuunaks täisadaptiivse jadamudeli loomine, mis arvestaks blokeerivate elementaarrünnetega. Lisaks tuleb uurida efektiivsemate algoritmide või optimeeringute leidmise võimalusi. Kuna praktikas ei ole tavaliselt esmatähtis teada täpset ründa ootetulu, vaid oluline on, kas see on positiivne või negatiivne, siis arvutuste kiirendamise üheks variandiks on sobilike lähendite leidmine [29].

Lisaks näeb antud töö autor suuremat praktilist kasu niisugusest mudelist, mille sisendid oleksid arvude vahemikud, mitte täpsed väärtused, ja väljund oleks ründaja ootetulu vahemik. Selline lähenemine peegeldaks reaalsust täpsemini ning väljundi täpsus oleks kasutajale kohe näha ja see sõltuks sellest, kui täpselt suudab turvaanalüütik sisendparameetrite väärtusi hinnata. Üks vahemikke kasutav semantika on juba loodud [24], kuid see ei ole kooskõlaline Mauw ja Oostdijk raamistikuga ning on oma olemuselt paralleelmudel. Seega oleks üheks arengusuunaks niisuguse uue lähenemise loomine.

Ründepuude metoodika kasutamine praktikas nõuab samuti teatud lisauuringuid ning analüüsi. Praegusel hetkel ei ole teada, millise täpsusega on keskmine turvaanalüütik võimeline hindama rünnete parameetreid, kuna niisugust analüüsi ei ole veel tehtud. Sellise uuringu läbiviimine on üsna problemaatiline, kuna objektiivseid andmeid on raske leida.

Kui õnnestuks mingil moel hankida objektiivseid turvaparameetrite hinnanguid mingi konkreetse ründepuu jaoks, siis oleks võimalik teha statistilisi uurin-

guid, mille tulemusena selguks, kui palju erinevad keskmise turvaanalüütiku hinnangud objektiivsetest. Kui selliseid andmeid hankida ei õnnestu, siis võib erinevate spetsialistide hinnanguid võrrelda omavahel ning vaadata, kui palju nad keskmiselt erinevad. Ka sellest saaks teha teatud järeldusi parameetrite väärtuste hindamise täpsuse kohta.

Samuti ei ole siiani tehtud sensitiivsuse analüüsi, mis on vajalik selleks, et teada saada, kui palju sõltub väljund sisendi täpsusest. Kui niisugune analüüs oleks olemas, siis oleks võimalik otsustada, milline on aktsepteeritav välja arvutatud ründaja ootetulu viga ning sellest saaks järeldada, millise täpsusega tuleb hinnata parameetreid. On selge, et ebatäpsete andmetega ei ole mõistlik teha täppisarvutusi ilma, et oleks teada, milline on tulemuse täpsusklass. Seega on sensitiivsusanalüüs üsna kriitilise tähtsusega.

Üheks võimaluseks, kuidas lihtsustada väheste kogemustega turvaanalüütikute tööd, on ründepuude andmebaasi koostamine. See oleks tihti esinevatele ja standardsetele alamrünnetele vastavate (alam)puude kogum, mis võiks sisaldada ka rohkemate kogemustega spetsialistide poolt hinnatud parameetrite väärtusi. Teine variant on lasta paljudel spetsialistidel väärtustada salvestatud alampuid ning hoida andmebaasis hinnangute keskmiseid väärtusi. Niisugune ründepuude raamatukogu lihtsustaks oluliselt ründepuude koostamist ja väärtustamist. Samuti aitaks see kaasa ründepuude metoodika parimate praktikate väljakujunemisele.

Üheks ründepuude arvutamisega seotud probleemiks on paljude algoritmide supereksponentsiaalne keerukus, mis takistab piisavalt suurte ründepuude kasutamist praktikas. Nagu ülal mainitud, on üheks võimaluseks probleemi leevendamiseks lähendite ja optimeeringute leidmise võimaluse uurimine. Teiseks võimaluseks on aga hajusarvutuse kasutamine. Selleks tuleb aga uurida olemasolevate algoritmide paralleliseeritavust ja koostada hajusarvutusi toetavaid programme.

4 Ründepuude tarkvaraline raamistik

Peatükis 3 kirjeldatud ründepuude metoodika kasutusvõimalused eeldavad, et neid on võimalik arvutada. Jaotises 2.4 toodud valemid ja algoritmid on üsna keerukad ning töömahukad. On selge, et ilma arvutustehnikat kasutamata on neid võimatu päriselus rakendada. Seega on hädavajalik tarkvaraline tööriist, mis võimaldaks ründepuid mugavalt koostada, arvutada ja tulemusi analüüsida.

Antud töö üheks eesmärgiks oli niisuguse tarkvaralise raamistiku ehitamine. Järgnevas vaadeldakse programmile esitatud nõudeid.

4.1 Nõuded tarkvarale

Kõige tähtsam nõue tarkvarale on selle paindlikkus. Kuna metoodika teoreetiline pool on veel arenemisjärgus, siis võib see olulisel määral muutuda – võib tekkida uusi mudeleid, muutuda ründepuu enda struktuur, ilmselt tekib uusi algoritme jne. Arendatav raamistik peab olema niivõrd paindlik, et ta võimaldaks kasutada erinevaid mudeleid, arvutussemantikaid ja seaks ründepuu struktuurile võimalikult vähe piiranguid. Seda arvestades olid püstitatud järgmised nõuded:

4.1.1 Nõuded kasutajaliidesele

1. Programmil peab olema mugava kasutajaliidesega graafiredaktor, mille abil on võimalik koostada ründegraafe. Graaf peab olema tsükliteta ja suunatud. Juure olemasolu ei ole kohustuslik, kuigi esialgu kasutatakse juurega graafe.

2. Ründepuid peab olema võimalik faili salvestada ja failist lugeda, kusjuures tippude paigutus ekraanil ei tohi selle käigus muutuda, v.a. siis, kui kasutaja soovib, et tippe joonistataks mingi kindla paigutusalgoritmi järgi. Programmi esimeses versioonis paigutusalgoritmide kasutamine ei ole nõutud, kuid peab arvestama võimalusega, et niisugune vajadus võib tulevikus tekkida. Salvestatud ründepuu formaat ei pea olema inimloetav.

3. On nõutud, et tippude välimus (kuju) oleks kasutaja poolt seadistatav. Tipu kuju sõltub selle tüübist. Kasutaja peab saama seadistada, millistele tüüpidele vastavad millised kujud.

4. Kuna ründepuu võib olla üsna suur ega pruugi korraga ekraanile mahtuda, siis peab töölehte saama suumida. Samuti peab olema võimalik alampuid peita ja tagasi laiendada. Peidetud alampuu peab olema visuaalselt eristatav tavalisest tipust.

5. Veaolukordade puhul peavad korrektsed veateated olema nähtavad kasutajaliideses. Veateated peavad olema mittetehnilised ja kasutajale arusaadavad.

6. Arvutusalgoritmide tulemused peavad olema nähtavad kasutajaliideses. Esialgu tuleb kuvada ründaja ootetulu ja kriitilist rünnet, kuid hiljem võib tekkida vajadus mingite teiste väljundite kuvamiseks.

7. Erinevad ründepuude mudelid võivad opereerida erinevate tippude parameetritega. Seega on nõutud, et kasutaja saaks valida, milliseid parameetreid ta soovib kasutada. Programmil peab olema konfigureerimisaken, kus kuvatakse kõik eeldefineeritud parameetrid, millest saab valida komplekti, mida kavatsetakse arvutustes kasutada.

8. Kasutajal peab olema võimalus lehtede parameetreid mugavalt väärtustada. Parameetrid võivad olla erinevat tüüpi, seega peab programm kontrollima, et sisestatud väärtus vastab nõutud kujule – vastasel juhul tuleb kuvada veateade. Uued parameetrid peavad koodi tasemel olema lihtsalt lisatavad. Kasutajaliides ei tohi sõltuda kasutatavate parameetrite arvust ja tüübist ning väärtused ja tüübid peavad olema iga tipu juures nähtavad.

9. Tuleb arvestada, et mõned parameetrid on lokaalsed ja mõned globaalsed. Kui ühe tipu juures muudetakse globaalse parameetri väärtust, siis muutub see ka teistel tippudel. Globaalne on esialgu ainult ründaja kasum, kuid nende lisamine peab olema lihtne.

4.1.2 Nõuded arvutuskihile

1. Programm peab liidestuma erinevate olemasolevate ründepuude arvutusutiliitidega (nn arvutajatega). Uute arvutajate lisamine peab olema piisavalt lihtne. Kasutajal peab olema võimalus valida, millise mudeliga ta soovib töötada.

2. Peab arvestama, et erinevad arvutajad nõuavad erinevaid tippude parameetreid ning sisend- ja väljundformaadid võivad olla erinevad. Kui kasutaja on valinud parameetrite komplekti, mis ei sobi antud arvutajaga, siis kuvatakse talle veateade.

3. Arvutusmootor peab saama kasutajaliidesele edastada veateateid, kuna mõned erindid võivad tekkida ka välise utiliidi töö ajal.

4. Tuleb arvestada, et mõned arvutajad võivad muuta ründepuu struktuuri (näiteks tippe kustutada, lisada või ümber järjestada).

5. Kuna osade arvutajate töö tulemus sõltub tippude järjestusest, siis peab kasutajal olema võimalus tippude järjestust määrata. Arvutusmootorile tuleb tipud ette anda sellises järjestuses, nagu on eelnevalt määratud.

6. Kuna mõned algoritmid on äärmiselt ressursinõudlikud, siis peab arhitektuur toetama arvutusrutiinide välja kutsumist nii sünkroonses kui ka asünkroonses režiimis. Esimesel juhul on kasutaja töö arvutamise ajaks blokeeritud. Teisel juhul saadetakse arvutusloogika kihile päring sisendandmetega, pärast mida saab kasutaja ründepuu kallal edasi tegutseda. Kui arvutaja saab tulemused kätte, saadab ta ise vastuse kasutajaliidese kihile. Raamistiku esimeses versioonis kasutatakse vaid ühte režiimi, kuid teise sisse lülitamine peab olema lihtne.

4.1.3 Üldised ja mittefunktsionaalsed nõuded

Nagu ülal mainitud, peab raamistik olema võimalikult paindlik ning ühendama endas erinevatel ründepuude mudelitel põhinevaid arvutajaid, olles ise neile talitlusloogika ja kasutajaliidese kihiks. Seega on raamistik justkui pakend erinevatele ründepuude utiliitidele.

Sellest tulenevalt peab ta olema kihilise arhitektuuriga, mis tekitaks süsteemi komponentide vahele võimalikult vähe sõltuvusi. Programm peab olema mugavalt laiendatav ja edasiarendatav. Ta peab olema kasutatav teegina teiste sarnaste rakenduste loomiseks. Sellest tulenevalt peab ta olema vabavaraline ja lahtise lähtekoodiga. Kasutaja jaoks peab programm olema piisavalt mugavalt seadistatav.

Raamistik peab olema platvormist sõltumatu ning kasutaja masinasse paigaldamine peab olema lihtne. See nõue üldjuhul ei kehti väliste arvutajate puhul, kuna enamik neist ei ole antud töö autori poolt tehtud ning neid võib juurde tulla. Raamistik peab töötama sõltumatult väliste utiliitide olemasolust. Selleks aga, et ründepuid saaks arvutada, peab kasutaja masinas olema kompileeritud ja paigaldatud arvutaja, mida ta soovib kasutada.

Raamistiku enda jõudlus ei tohiks arvutamisel olla pudelikaelaks. Programm ise peab olema piisavalt efektiivne ega tekitama keskmisel töömasinal silmnähtavaid viivitusi. Kogu süsteemi jõudlus sõltub aga kasutatavatest välistest arvutajatest.

Järgmises jaotises antakse lühiülevaade olemasolevatest ründepuude analüsaatoritest, mida antud töö autoril on õnnestunud hankida. Samuti selgitatakse, miks nad ei rahuldanud püstitatud nõudeid ning miks otsustati alustada oma raamistiku arendamist.

4.2 Olemasolevad lahendused ja nende puudused

4.2.1 CORAS raamistik

CORAS on mudelipõhine metoodika, mis kirjeldab riskianalüüsi protsessi, kasutades CORAS riskide modelleerimise keelt. See ei ole otseselt ründepuude me-

toodika, kuid kasutab sarnast lähenemist - ohtude modelleerimist, mis põhineb UML keelel. Metoodikas antakse juhised, milliseid tegevusi tuleb teha erinevatel riskihalduse protsessi etappidel, kuhu kuulub ka riskide hindamine. Täpsemat informatsiooni leiab allikast [51].

Siiski keskendub CORAS riskianalüüsi juures kõige rohkem modelleerimisele ning oma olemuselt on ta pigem kvalitatiivne metoodika. Turvaparameetrite hindamiseks kasutatakse enamasti verbaalseid hinnanguid. Mõnikord kasutatakse ka reaalarvude vahemikke, kuid neid kasutavate valemite korrektsuse tõestused puuduvad. Enamasti põhineb analüüs riskimaatriksil, mis sisaldab kvalitatiivseid hinnanguid.

CORAS metoodika autorite poolt on tehtud ka UMLil põhinev raamistik, mida kasutatakse riskide modelleerimiseks. Raamistik võimaldab koostada ründegraafe, kasutades erinevat tüüpi tippe. Eristatakse inimohte, turvanõrkusi, õnnetusi ja intsidente, varasid ning vastumeetmeid. Siiski ei võimalda antud raamistik teha ründepuudega arvutusi ning ei tundu eriti paindlik. Seega CORAS püstitatud nõudeid ei rahulda.

4.2.2 AttackTree+

AttackTree+ on firma Isograph [52] poolt arendatav kommertsiaalne tarkvaratoode, mida saab kasutada ründepuude analüsaatorina. Tasuta on võimalik saada programmi demoversiooni ning vaadata videodemonstratsiooni. AttackTree+ võimaldab koostada ründepuid ning väärtustada lehtede parameetreid. Vaikimisi on tippudega seotud vaid õnnestumise tõenöosuse parameeter, kuid neid on võimalik juurde defineerida. Lisaks saab igale alamründele kinnitada selle võimalikud mõjud süsteemile – näiteks rahalised kahjud.

Programm teeb ka teatud ründepuude analüüsi, mille tulemusena kuvatakse võimalikke minimaalseid ründekomplekte, mis on piisavad primaarse ründe õnnestumiseks. Ründekomplekte on võimalik järjestada nende parameetrite järgi. Samuti on võimalik valitud ründekomplekte ründepuu peal visualiseerida.

Siiski on antud programmil ka olulisi puudusi. AttackTree+ ei võimalda kasutada ründegraafe, vaid ainult puid. Raamistik võimaldab küll defineerida ja väärtustada erinevaid parameetreid, kuid arvutatakse neid üksteisest sõltumatult, kasutades äärmiselt lihtsustatud valemeid. Näiteks uue parameetri lisamisel saab valida, kuidas arvutatakse ülemtipu antud parameetri väärtust alamtippude väärtuste järgi, kusjuures võimalikud on näiteks: korrutamine, liitmine, maksimaalse ja minimaalse valimine. Seega põhineb AttackTree+ propageeruval mudelil, mis ei ole kooskõlaline Mauw ja Oostdijk raamistikuga ega peegelda ründaja käitumist piisavalt realistlikult.

Samuti ei ole AttackTree+ programmi võimalik liidestada teiste arvutusutiliiti-

dega, seega ei ole ta piisavalt paindlik ning seda on raske kohandada arenevate ründepuude mudelitega. Lisaks on AttackTree+ tasuline, kinnise lähtekoodiga ning töötab ainult Windowsi platvormil, seega ei õnnestu seda laiendada ja edasi arendada.

4.2.3 SeaMonster

SeaMonster on SHIELDS projekti raames arendatav turvalisuse modelleerimise tööriist, mis põhineb Eclipse raamistikul. SeaMonster võimaldab koostada ründepuid, süsteemi väärkasutusjuhte, turvalisuse seotud tegevuste graafe jms. Raamistik on vabavaraline ja lahtise lähtekoodiga. Rohkem infot leiab allikast [53].

Selle tööriista kõige suurem puudus on see, et ta ei võimalda ründepuid arvutada. Projekti eesmärk ei olnud turvalisuse kvantifitseerimine, vaid ainult mudelite loomine. Seega on SeaMonsteri praktiline kasu üsna väike ja meie poolt püstitatud nõudeid ta ei rahulda. Lisaks töötab ta hetkel ainult Windowsi operatsioonisüsteemis, kuigi põhineb platvormist sõltumatul ja äärmiselt paindlikul ning laiendataval Eclipse platvormil. Siiski väitsid tööriista autorid, et järgmises versioonis antud puudus kõrvaldatakse ning võib-olla kaalutakse ka ründepuude arvutamise võimaluse tekitamist.

On olemas ka teisi ründepuude raamistikke ja tööriistu, kuid antud töö autori teada on neil ülal kirjeldatud programmidega sarnased puudused. Ükski vaadeldud ründepuude analüsaatoritest ei rahuldanud püstitatud nõudeid. Seega võeti vastu otsus arendada oma raamistikku, millel ei oleks ülal kirjeldatud puudusi. Järgmises jaotises kirjeldatakse kasutatud arendusvahendite valimise protsessi.

4.3 Arendusvahendite valik

Kuna raamistiku paindlikkus ja laiendatavus olid kriitilise tähtsusega nõuded, siis tuli arendusvahendite valikusse suhtuda üsna tõsiselt, sest alguses valesti valitud arendusvahendid võivad mõjutada tervet programmi elutsüklit.

Seega pidi enne arendamise alustamist tegema eeltööd selleks, et otsustada, milliste vahenditega on püstitatud nõuetega raamistikku kõige mõistlikum arendada. Selleks olid erinevate programmeerimiskeelte ja teekide abil koostatud ja läbi testitud prototüübid, mille seast valiti hiljem kõige parem. Järgnevas on toodud proovitud vahendite kirjeldus, puudused ja eelised ning tehtud valiku põhjendus.

4.3.1 C++, Qt, BGL

Alguses oli projekteeritavat programmi plaanis kirjutada programmeerimiskeeles C++ ning kasutajaliidese arendamiseks kasutada Qt teeki [54]. Qt on mitmeplatvormiline raamistik, mida kasutatakse üsna laialdaselt graafiliste programmi-

de ehitamiseks. Lisaks kasutajaliidestele sisaldab see ka mitmelõimelisuse, komponentide suhtluse ja võrgunduse funktsionaalsust. Veel üheks Qt eeliseks on põhjalik ja selge dokumentatsioon ning lai kasutajakond.

Graafi struktuuri ja vajalike algoritmide arendamiseks oli plaanis kasutada BGL (*Boost Graph Library*) teeki, mis on osa populaarsest Boost raamistikust. BGL on väidetavalt äärmiselt paindlik ning lisaks muule sisaldab ka paljude graafialgoritmide realisatsiooni [55].

Siiski ilmnesid prototüübi arendamise käigus mõned BGL teegi puudused. Selgus, et graafi tippude parameetrite salvestamise mehhanism ei ole piisavalt paindlik – nimelt hoiab BGL kõigi parameetrite väärtusi sisemiselt stringi kujul, mis on suur kitsendus.

Lisaks on selle kood üsna keeruline ning ta on kirjutatud, kasutades šabloonklasse ja -funktsioone, mis teeb veateated raskesti loetavaks. Ka dokumentatsioon oli üsna kehv ning paljud võimalused on täiesti dokumenteerimata. Seetõttu ei tundunud raamistiku arendamine BGL abil perspektiivne.

4.3.2 C++, Qt, Graphviz

Järgmises prototüübis võeti graafistruktuuri realisatsiooni aluseks Graphviz teek, millel põhineb samanimeline graafi visualiseerimise tööriist [56]. Kuna plaanis oli raamistiku hilisemates versioonides kasutada tippude paigutusalgoritme, siis tundus Graphviz selleks sobivat.

Graphviz on üsna populaarne ning sisaldab mitmeid graafi visualiseerimise programme. Samuti toetab ta paljusid graafi teksti kujul hoidmise formaate, seetõttu ei pea ise kirjutama parsereid, vaid saab kasutada standardset formaati.

Siiski ilmnesid arendamise käigus ka Graphviziga mõned probleemid, millest kõige suurem on tema piiratud paindlikkus. Teek küll võimaldab uute parameetrite defineerimist, kuid suvalist tüüpi parameetrite lisamist ja faili kirjutamist ta ei toeta. Selleks, et graafi tipuga siduda uut parameetrit, peab selle väärtust hoidma stringi kujul, mis on ebamugav ning ebaefektiivne. Ka faili kirjutamiseks ja failist lugemiseks pidi siiski valmis kirjutama oma parseri.

Graphvizi on C++ ja Qt-ga ka üsna raske integreerida, kuna ta on kirjutatud C keeles. Selleks, et teda ülejäänud objekt-orienteeritud koodiga mugavalt kasutada ning uute võimalustega laiendada, pidi Graphviz koodi ümber kirjutama enda pakendeid, mis on üsna ebamugav ja kohati mõjub ka pärsivalt süsteemi jõudlusele.

4.3.3 Java, JUNG

Järgmisena oli plaanis proovida Java programmeerimiskeele ja JUNG (*Java Universal Network/Graph Framework*) teegi kombinatsiooni. JUNG on Java Swingil põhinev graafide modelleerimiseks, analüüsiks ja visualiseerimiseks mõeldud raamistik [57]. Kuna Java järgib WORE (*write once, run everywhere*) põhimõtet, siis mitmeplatvormilisuse nõue on ilma suurema vaevata täidetud.

JUNG toetab erinevat tüüpi graafe: orienteeritud ja orienteerimata, tsüklitega ja tsükliteta graafe, puid, multigraafe jne. Lisaks sisaldab teek paljude algoritmide realisatsioone ning võimaldab tippe eeldefineeritud paigutusalgoritmide järgi ekraanile visualiseerida.

Vaadeldaval teegil on üsna mugav API-liides ning läbi mõeldud ja paindlikuks disainitud arhitektuur. JUNG realiseerib paljusid tuntud disainimustreid, mis lihtsustab selle projektiga integreerimist. Samuti on antud teeki üsna mugav laiendada ja vastavalt vajadusele muuta.

JUNG lihtsustab oluliselt graafiredaktorite ehitamist ning on varustatud korraliku dokumentatsiooni ja paljude näidetega. Autorid tegelevad teegi arendamisega edasi ning projektil on olemas ka klienditugi. Töö kirjutamise ajaks on viimane väljalase tehtud 24. jaanuaril 2010.

Puudustest võib nimetada vaid mõningaid antud töö autori poolt leitud vigu, kuid need õnnestus üsna lihtsa vaevaga ära parandada ning autoritele raporteerida.

Ülal nimetatud põhjustest tulenevalt otsustati ründepuude raamistikku kirjutada Java keeles, kasutades JUNG teeki. Järgmises jaotises vaadeldakse valminud tarkvara arhitektuuri.

4.4 Tarkvara arhitektuur

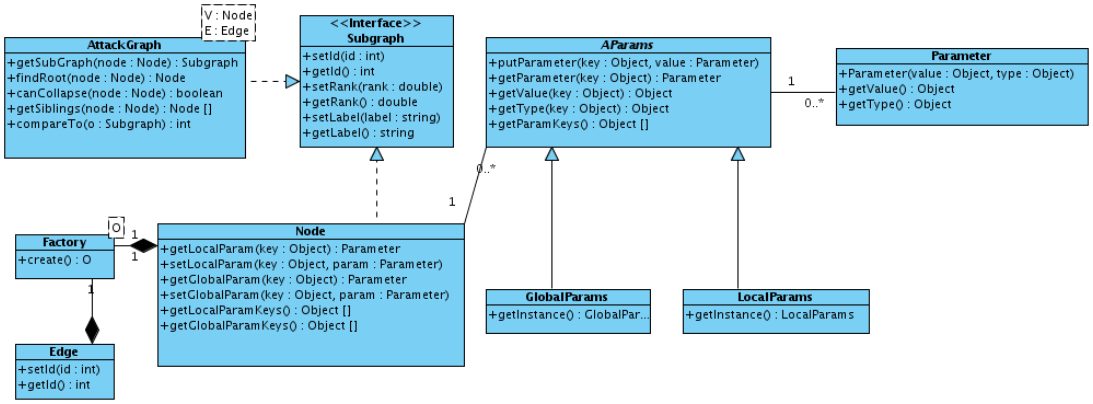
Raamistiku arhitektuur on modulaarne ja koosneb kahest eraldiseisvast kihist: kasutajaliides ja talitlusloogika, mis omakorda jaotuvad alamkihtideks (pakettideks). Järgmistes alajaotistes kirjeldatakse mõlemat kihti täpsemalt. Lisaks vaadeldakse raamistiku ehitussüsteemi.

4.4.1 Talitlusloogika kiht

Talitlusloogika kiht (`bl`) koosneb viiest alampaketist, mis sisaldavad 33 klassi ja umbes 2300 programmirida. Järgnevas vaadeldakse alamkihte detailsemalt. Erilist tähelepanu pööratakse graafi struktuuri, arvutamise ning sisend-väljundiga seotud osadele, mille kirjeldustele on lisatud ka klassiskeemid.

Pakett `model`

Antud pakett sisaldab klasse, mille abil realiseeritakse graafi struktuuri ning sellega seotud lokaalseid ja globaalseid parameetreid. Joonisel 2 on kujutatud selle klassiskeem.



Joonis 2: Paketi `model` klassiskeem

`AttackGraph` on šabloonklass parameetritega `V` ja `E`, mis tähistavad tippu ja serva vastavalt. `AttackGraph` realiseerib graafi struktuuri, laiendades šabloonlassi `DirectedSparseGraph<V, E>`, mis kuulub JUNG teeki ning seetõttu ei ole skeemile kantud. Klassis `AttackGraph` on realiseeritud näiteks alamgraafi, juurtipu ja tipu naabrite hulga leidmise meetodid.

`AttackGraph` realiseerib `Subgraph` liidest, kuna graafis võib olla alamgraafe. `Subgraph` sisaldab meetodeid alamgraafi unikaalse identifikaatori, nime ja järjestuse määramiseks ja küsimiseks. Sama liidest realiseerib ka `Node` klass, mis kujutab graafi tippu. Tipu loomiseks on loodud `Factory` mustrit realiseeriv šabloonklass. Samamoodi luuakse ka graafi servade eksemplare (skeemil kujutatud nimega `Edge`).

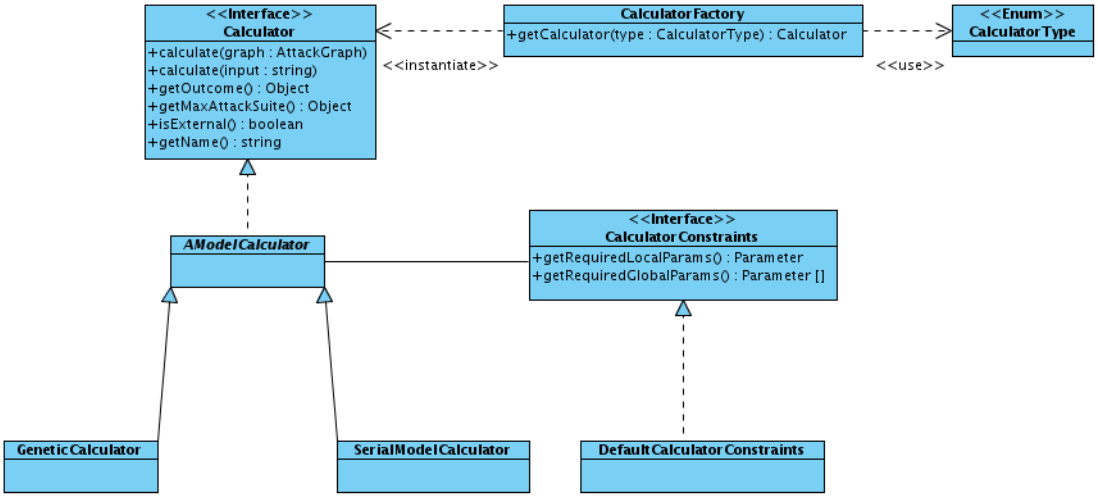
Tipuga on seotud abstraktne klass `AParams`, mis on oma olemuselt räsitabel, mis võimaldab pöörduda salvestatud parameetritele võtme järgi. Võtmena kasutatakse parameetri nime. `AParams` võib sisaldada suvalist arvu parameetreid ning neid on võimalik lisada programmi töö ajal. Väärtuste otsimine räsitabelist on efektiivne. Abstraktsest klassist pärinevad klassid `GlobalParams` ja `LocalParams`, mis erinevad algväärtustuste ja olemasolevate võtmete järgi.

Klass `Parameter` kujutab tipu turvaparameetrit ning sisaldab selle väärtust ja tüüpi. Tüübi salvestamine on vajalik selleks, et saaks kontrollida kasutaja poolt sisestatud väärtuse korrektsust vastavalt parameetri tüübile ning vajadusel kuvada veateadet.

Nagu näha, graafi klass ei sõltu tippude ja servade objektidest, mis teeb disaini paindlikuks ning võimaldab lihtsalt muuta või vahetada tipu või serva realisatsiooni.

Pakett `calc`

Pakett `calc` sisaldab arvutusloogikat realiseerivaid klasse. Joonisel 3 on kujutatud selle klassiskeem.



Joonis 3: Paketi `calc` klassiskeem

Hetkel on realiseeritud kaks arvutajat – `GeneticCalculator`, mis on paralleelmudeli geneetiline algoritm, ning `SerialModelCalculator`, mis on jadamudeli täppisarvutaja. Mõlemad klassid pärinevad abstraktsest klassist `AModelCalculator`, kuhu on tõstetud ühine kood, et vältida dubleerimist.

Mõlemad arvutajate klassid tegelikult arvutusloogikat ei sisalda – see on realiseeritud eraldi kompileeritud programmides. Klasside ülesanne on valmistada ette sisend, käivitada vajalik programm ning lugeda selle väljundit. Sisendi koostamiseks ja väljundi lugemiseks kasutatakse `io` paketis sisalduvaid klasse.

Kõik arvutajad peavad realiseerima `Calculator` liidest, mis sisaldab meetodeid arvutusrutiinide käivitamiseks ja väljundi lugemiseks. Samuti on võimalik kontrollida, kas mingi arvutaja on väline või sisemine. Väline arvutaja on niisugune, mis on realiseeritud eraldi programmina ning mida käivitatakse teise protsessina. Sisemised arvutajad realiseeritakse sama projekti koodis.

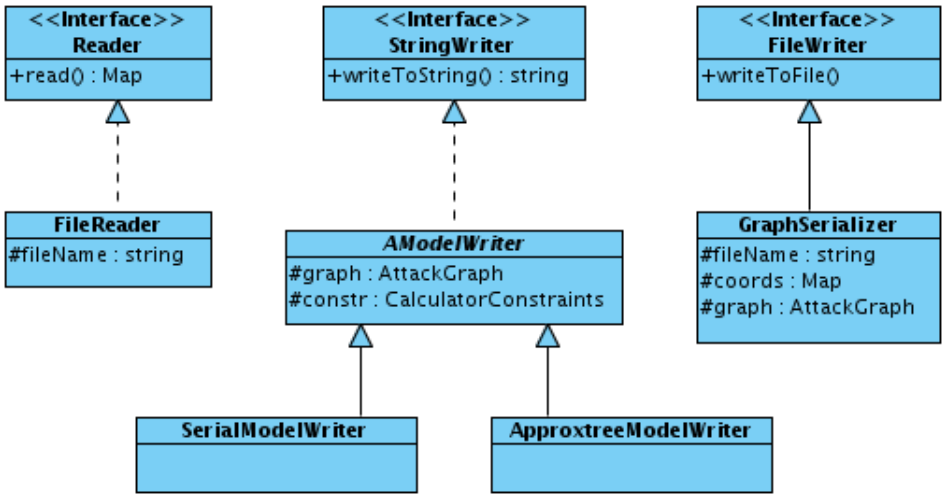
Praeguseks hetkeks on raamistik liidestatud kahe välise arvutajaga, millest üks on Aivo Jürgensoni poolt kirjutatud geneetiline arvutaja, mis põhineb paralleelmudelil ning on kirjeldatud allikas [19]. Teine on jadamudeli arvutaja, mille alged on kirjutatud Tiina Turbani poolt ning mis on edasi arendatud ja täiendatud antud töö autori poolt. Selle algoritmide selgitused on toodud allikas [29].

Iga arvutajaga on seotud `CalculatorConstraints` liidest realiseeriv klass, kus hoitakse antud arvutajaga seotud kitsendusi. Hetkel on kitsendusteks märgitud lokaalsete ja globaalsete parameetrite hulgad, mis peavad olema väärtustatud antud arvutaja kasutamiseks.

`Calculator` tüüpi objektide loomiseks on realiseeritud `CalculatorFactory` klass. See sisaldab meetodit `getCalculator`, mis võtab parameetriks arvutaja tüübi ja tagastab vastava arvutaja eksemplari. Võimalike tüüpide hoidmiseks kasutatakse klassi `CalculatorType`, mis on oma olemuselt klassifikaator.

Pakett `io`

Pakett `io` sisaldab sisend-väljundit realiseerivaid klasse. Selle klassiskeem on toodud Joonisel 4.



Joonis 4: Paketi `io` klassiskeem

Pakett sisaldab kolme liidest - `Reader`, `StringWriter` ja `FileWriter`. Liides `StringWriter` sisaldab meetodit `writeToString`, mille abil kirjutatakse graaf string tüüpi objekti. Liidest realiseerib abstraktne klass `AModelWriter`.

Kuna kaks kasutatud arvutajat on kirjutatud erinevatel aegadel ja erinevate inimeste poolt, siis nende sisend- ja väljundformaadid ei ole identsed. Seega oli kirjutatud kaks erinevat klassi – `SerialModelWriter` ja `ApproxtreeModelWriter`, mis pärinevad abstraktsest klassist `AModelWriter`. Nende abil kirjutatakse ründegraaf formaati, mis sobib kahele ülal nimetatud arvutajale.

Liides `FileWriter` sisaldab meetodit `writeToFile`. Liidest realiseerib klass `GraphSerializer`, mis kirjutab graafi faili, kasutades standardset `Java Serialization` mehhanismi. Graaf ja selle tippude koordinaatide tabel salvestatakse eraldi. See on vajalik selleks, et ründegraafi failist laadimisel oleksid selle tipud paigutatud täpselt nii nagu enne salvestamist.

Liides `Reader` sisaldab meetodit `read`, mille abil teostatakse lugemine. Seda realiseerib klass `FileReader`, mis laadib serialiseeritud graafi failist mällu. Meetod `read` tagastab räsitabeli, mis sisaldab graafi ennast ja selle koordinaate.

Pakett `util`

Pakett `util` sisaldab abiklasse ja utiliite, mida kasutatakse teiste moodulite ja klasside poolt. Nendest tähtsamad on `IdManager`, `RegexChecker` ja `Utils`. Klass `IdManager` realiseerib `Singleton` disainimustrit ning seda kasutatakse unikaalsete identifikaatorite genereerimiseks, millega on varustatud kõik graafid, alamgraafid, tipud ja servad.

`RegexChecker` sisaldab regulaaravaldisi ja meetodeid, mis võimaldavad veenduda, et mingi väärtus vastab etteantud tüübile. Selle klassi abil kontrollitakse kasutaja poolt sisestatud turvaparameetrite väärtuste vastavust nende tüübile. Klassis `Utils` hoitakse hulka kasulikke meetodeid, mis ei sobinud ühegi teise klassi konteksti.

Pakett `conf`

Paketi `conf` tähtsaim klass on `Config`, mille abil on realiseeritud raamistiku konfiguratsioonihaldus. `Config` realiseerib `Singleton` mustrit ning kasutab sisemiselt `Apache Commons Configuration` teeki [58]. Antud klass sisaldab hetkel kasutusel olevate võtmete hulka, mis vastavad erinevatele konfigureeritavatele omadustele. Selle abil on võimalik salvestada ka suvalisi atribuute, kuid see ei ole segaduste vältimiseks soovitatav.

Konfiguratsiooni hoitakse kasutaja kodukataloogis spetsiaalses kaustas. Linux operatsioonisüsteemi puhul on see näiteks `~/.aforest/aforest.properties`. Kõigi seadistatavate omaduste jaoks on määratud nende algväärtused. Programmi esmakordsel käivitamisel luuakse konfiguratsioonifail ning täidetakse see vaikimisi seadistustega.

4.4.2 Kasutajaliidese kiht

Kasutajaliidese kiht (`ui`) koosneb neljast paketist: `components`, `main`, `util` ja `visualization`, mis omakorda sisaldavad 30 klassi ja umbes 3300 programmirida. Järgnevas on detailsemalt kirjeldatud kasutajaliidese kihi paketid.

Pakett `components`

Pakett `components` sisaldab programmi kasutajaliidese komponentide realisatsioone. Sinna kuulub kaks alampaketti – `panels` ja `dialogs`, mis sisaldavad vastavalt kasutatud paneelide ja dialoogiakende klasse. Lisaks sisaldab see menüüelementide ja erinevate nuppude realisatsioone.

Mõned nupud on sellised, et nende olek mõjutab teiste olekut. Näiteks ühele nupule vajutamine deaktiveerib teised nupud. Sellise võimaluse realiseerimiseks

on tehtud klass `ToggleableButtons`, mis hoiab kõigi kasutusel olevate nuppude eksemplare ning võimaldab juhtida nende käitumist.

Dialoogiakendest on olemas parameetrite väärtuste muutmise (`NodeParams-EditingDialog`) ja konfigureerimise (`SettingsDialog`) aknad. Klass `NodeParams-EditingDialog` kuvab kasutusel olevate parameetrite nimesid ning lahtreid, kuhu on võimalik sisestada vastavaid väärtusi. Lokaalsed ja globaalsed parameetreid kuvatakse eraldi.

`NodeParamsEditingDialog` on äärmiselt paindlik – selle töö ei sõltu kasutusel olevate parameetrite arvust, nimedest ja tüüpidest. Kui lisatakse või valitakse parameetreid juurde, siis tekivad õiged lahtrid dialoogiaknasse automaatselt, seega ei ole kasutajaliidest vaja muuta. Samuti kontrollitakse sisestatud väärtuste korrektsust ilma kasutajaliidese koodi parameetrite tüüpe sisse kirjutamata. Kui mingi parameeter on klassifikaator, siis kuvatakse seda liitboksina, kust on hiirega võimalik valida selle väärtust.

`SettingsDialog` võimaldab seadistada raamistiku käitumist ja välimust. Selle aken koosneb kolmest sakist – `Calculators`, `Parameters` ja `Appearance`. `Calculators` sakil on võimalik seadistada kasutatavaid arvutajaid – valida liitboksi abil üks arvutaja eeldefineeritud hulgast ning määrata sellele vastava kompileeritud programmi asukoht failisüsteemis.

`Parameters` sakk võimaldab valida kasutatavaid parameetreid. Seal kuvatakse kõiki globaalsete ja lokaalsete parameetrite nimesid, mille seast saab linnukestega valida sobivaid. `Appearance` sakil asuvad programmi välimusega soetud seadistused.

Alampaketis `panels` on realiseeritud erinevad graafilised paneelid – näiteks `CalcPanel`, `ZoomPanel`, `ModeToolsPanel`, `NodeCreationToolsPanel` jt. `Calc-Panel` hoiab arvutamise käivitamiseks vajalikke nuppe. `ZoomPanel` sisaldab suumimise nuppe. `ModeToolsPanel` paneelile paigutatakse hiire režiimide muutmiseks vajalikke nuppe. Kasutusel on järgmised režiimid: `Transforming`, `Editing` ja `Picking`. Erinevates režiimides on hiirega võimalik teha erinevaid tegevusi.

`Transforming` režiimis saab tervet graafi mööda ekraani liigutada. `Editing` režiimis on võimalik luua ja kustutada tippe ning servi, ahendada ja laiendada alampuid ja avada parameetrite väärtuste muutmise dialoogiakent. `Picking` režiimis saab ühte tippu või alamgraafi liigutada mööda ekraani ning topeltklõpsuga muuta tippude nimesid.

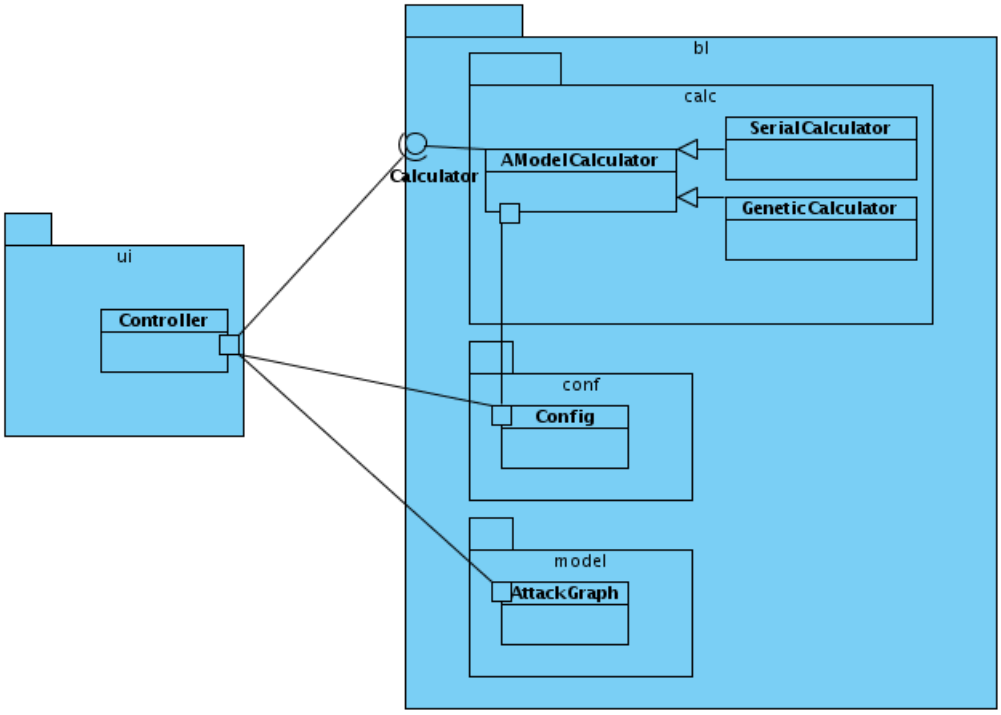
`NodeCreationToolsPanel` sisaldab nuppe erinevat tüüpi tippude loomiseks, milleks on hetkel `AND`, `OR` ja `LEAF`. Antud paneel on äärmiselt paindlik ega sõltu defineeritud tüüpide arvust ega nimedest. Kui raamistikule lisatakse uus parameetrite tüüp, siis uus nupp tekib ekraanile automaatselt. Nõutud on vaid see, et ikoonide kataloogis oleks sellele nupule vastav ikoon ning selle faili nimi oleks

kujul `TÜÜP.png`.

Pakett `main`

Antud pakett sisaldab graafiredaktori klassi `AGraphEditor` ning kontrollerklassi `MainController`. `AGraphEditor` on see, mis sisaldab `main` meetodit ning mida käivitatakse esimesena. `MainController` realiseerib `Controller` GRASP (*General Responsibility Assignment Software Patterns*) disainimustrit. Läbi selle suhtleb talitlusloogika kasutajaliidesega. Niisugune suhtlus on vajalik igasuguste andmete ja teadete saatmiseks madalamatest kihtidest kõrgematesse.

Sõltuvused kontrolleri ja talitlusloogika kihi vahel on kujutatud Joonisel 5. Kontroller suhtleb arvutusloogikaga läbi `Calculator` liidese. Lisaks kasutab ta arvutajate seadistuste lugemiseks `Config` klassi ning annab arvutajatele ette graafi klassist `AttackGraph`.



Joonis 5: Kontrolleri sõltuvused

Pakett `util`

Pakett `util` koosneb kahest klassist - `UiUtils` ning `MyGraphCollapser`. `UiUtils` eesmärk on sarnane talitlusloogika kihi klassiga `Utils`, kuid see sisaldab ainult kasutajaliidesele vajalikke utiliite. `MyGraphCollapser` laiendab ja täiendab JUNG teegi klassi `GraphCollapser`. Selle abil on realiseeritud alampuude ahendamine (peitmine) ja laiendamine.

Pakett `visualization`

Antud pakett sisaldab graafi visualiseerimisega seotud klasse, mis on suures osas JUNG teegi võimaluste laiendused. JUNG toetab uute pistikute (ingl. k. *plug-in*) lisamist, mille abil on mugav juhtida teegi käitumist ning lisada sinna uut funktsionaalsust.

Pakett `visualization` sisaldab erinevaid pistikuid, mida kasutab šabloonklass `MyModalGraphMouse`, mis omakorda realiseerib mitmerežiimilise hiire funktsionaalsust. Visualiseerimise parameetrite seadistamiseks on klass `MyPluggableRenderContext` ning visualiseerimise enda eest vastutab klass `MyVisualizationViewer`.

Graafi välimuse ja käitumise muutmiseks kasutab JUNG teek `Decorator` disainimustrit. Paketis *visualization* on realiseeritud tippude kuju ja suuruse muutmise kujundajad `MyVertexShapeTransformer` ja `MyVertexSizeTransformer`.

4.4.3 Ehitussüsteem

Raamistiku ehitussüsteem kasutab Apache Maven tööriista, mille tööd juhitakse projekti juurkataloogis asuva faili `pom.xml` abil. Selles failis on märgitud projekti sõltuvuste nimekiri, ning Maven laadib teekide hoidlast automaatselt vajalikud paketid alla, kui arendaja masinas neid veel ei ole.

Raamistiku ehitamise tulemusena koostatakse käivitatav `jar` pakk, mis sisaldab kõiki programmi tööks vajalikke teeke ning raamistiku kompileeritud koodi. See võimaldab vältida *Classpath*iga seotud probleeme, kuna paki asukoht failisüsteemis ei ole sel juhul oluline.

Raamistiku väljalaske tegemiseks on keeles Python realiseeritud skript `make-release.py`, mis loob `tar.gz` arhiivi ning paigutab sinna kompileeritud koodi, litsentsitingimused, kasutusjuhendid ning Linux operatsioonisüsteemi jaoks valmis kompileeritud arvutajad. Neid arvutajaid kasutatakse vaikimisi seadistustega.

4.5 Raamistiku laiendamine ja edasiarendamine

Antud jaotises kirjeldatakse valminud raamistiku laiendamise protsessi, et demonstreerida selle paindlikkust. Järgnevas vaadeldakse uue parameetri lisamist, tipu tüübi defineerimist ning liidestamist uue välise arvutajaga.

4.5.1 Uue parameetri lisamine

Oletame, et raamistikule soovitakse lisada uus lokaalne parameeter `skills`, mis tähistab ründaja oskuste hinnangut. Esiteks tuleb klassi LocalParams lisada uus

võti. Selleks defineeritakse staatiline `String` tüüpi konstant:
`public static final String SKILLS = "skills";`

Samuti tuleb konstant `SKILLS` lisada ka võtmete massiivi `allParamKeys`, mis on defineeritud samas klassis. Lisaks on meetodis `initDefaultParams` vaja määrata vaikimisi väärtus koos parameetri tüübiga. Kui soovitakse, et uue parameetri väärtus oleks reaalarv, siis selleks sobib järgmine rida:
`putParameter(SKILLS, new Parameter(null, Double.class));`

Pärast seda tekib parameeter `skills` automaatselt konfigureerimise dialoogiakna `Parameters` sakile, kust seda on võimalik valida. Kui see on valitud, siis saab tipu parameetrite redigeerimise aknast sisestada selle väärtust, kusjuures sisestamisel kontrollitakse, et antud väärtus oleks reaalarv. Selleks aga, et uut parameetrit arvestataks ründepuu arvutamisel, tuleb realiseerida uus arvutaja, mis on kirjeldatud alajaotises 4.5.3.

4.5.2 Uue tipu tüübi defineerimine

Oletame, et raamistikule soovitakse lisada uus tipu tüüp `XOR` ning tekitada paneelile nupp, mille abil saab luua `XOR`-tippe. Samuti tahetakse, et uut tüüpi tippude kuju oleks seadistatav teistest tippudest sõltumatult.

Selleks tuleb loendisse `NodeType`, mis on defineeritud failis `NodeType.java`, lisada element `XOR`. Lisaks tuleb luua uue nupu jaoks ikoon, mis on pildifail laiendusega `.png` ning suurusega 44x16 pikslit. Faili nimeks peab olema `XOR.png` ning see tuleb paigutada kataloogi `modules/ui/resources/images`.

Selleks, et uut tüüpi tipu kuju oleks seadistatav, tuleb klassi Config lisada konstant kujul
`public static final String XOR_SHAPE = "XOR_shape";`
ning meetodisse `setDefaultProperties` tuleb lisada vaikimisi seadistus, näiteks:
`conf.setProperty(XOR_SHAPE, ShapeType.SQUARE + "");`

Pärast seda tekib paneelile automaatselt vastav nupp, millega saab luua `XOR` tüüpi tippe. Samuti tekib parameetrite redigeerimise aknas tipu tüüpide loendisse uus element ning uut tüüpi tippude jaoks saab konfigureerimisaknast seadistada selle kuju. Uut tüüpi tippude arvutamiseks tuleb realiseerida uus arvutaja. Järgmises alajaotises vaadeldakse raamistiku liidestamist uue välise arvutusutiliidiga.

4.5.3 Raamistiku liidestamine uue välise arvutajaga

Oletame, et raamistikus tahetakse kasutusele võtta uus väline arvutaja, mille kompileeritud programm asub kuskil failisüsteemis. Esiteks tuleb paketti `calc` tekitada vastav klass, mis realiseerib `Calculator` liidest, seejuures võib taaskasutada `AModelCalculator` klassi koodi, mis vastutab välise programmi käi-

vitamise eest.

Uue klassi meetod `getName` peab tagastama mingit unikaalset nime. Kuna tegemist on välise arvutajaga, siis peab meetod `isExternal` tagastama `true`. `CalculatorType.java` failis asuvasse loendisse tuleb lisada uus element ning klassi `CalculatorFactory` peab juurde kirjutama vastava objekti loomise koodi.

Kui uue arvutaja sisendformaat erineb olemasolevatest, siis tuleb `io` paketti lisada liidest `StringWriter` realiseeriv klass. Selle jaoks võib taaskasutada klassi `AModelWriter` koodi.

Kui see on tehtud, siis saab uue arvutaja binaarobjekti asukohta teha seadistatavaks. Selleks tuleb `Config` klassi lisada konstant kujul:

```
public static final String NEW_CALC_PATH = "new_calculator_path";
```

ning meetodisse `setDefaultProperties` võib lisada selle vaikimisi pöördustee, näiteks:

```
conf.setProperty(NEW_CALCULATOR_PATH, "bin/new_calculator");
```

Selle tulemusena tekib konfigureerimisakna `Calculators` sakile uue arvutaja valimise võimalus ning samast kohast saab määrata selle asukohta failisüsteemis. Kui uus arvutaja on valitud, siis ründepuu arvutamisel käivitatakse värskelt liidestatud arvutaja.

Kokkuvõte

Selleks, et tagada süsteemi turvalisus, on oluline teada, a) kui kaitstud ta mingil hetkel on, b) kui turvaliseks ta saab, kui rakendada teatud vastumeetmed, c) kuidas valida kõige efektiivsemad kaitsemehhanismid ning d) kui palju on nende rakendamisele otstarbekas ressursse kulutada. Vastamaks nendele küsimustele peab meil olema kindel meetrika, mille abil saaks mõõta süsteemi turvalisust ning mida saaks kasutada riskianalüüsi läbiviimisel.

Niisuguste turvameetrikate väljatöötamine on olnud oluline infoturbe arengusuund, kuid täpseid turvameetrikaid ei ole veel loodud. Seda raskendab asjaolu, et absoluutset turvalisust ei ole olemas – süsteem saab olla turvaline vaid mingi konkreetse ründaja vastu mingitel kindlatel eeldustel.

Viimasel ajal on leitud, et kasumist motiveeritud ründaja vastu kaitsmiseks on kasulik majanduslik lähenemine infoturbele. Süsteemi võib pidada turvaliseks siis, kui selle murdmine ei ole vastase jaoks materiaalselt ratsionaalne. Antud töös anti ülevaade ründepuude metoodikast, mis pöörab tähelepanu turvalisuse majanduslikele aspektidele. Seda ei saa veel nimetada turvameetrikaks selle mõiste rangema definitsiooni järgi, kuna metoodika tulemus sõltub eksperthinnangutest, kuid siiski võib öelda, et see on samm edasi turvalisuse kvantifitseerimisele.

Ründepuude kasutamiseks tuleb hinnata lihtsamate elementaarrünnete turvaparameetreid. Enamikes kirjandusallikates eeldatakse, et turvaanalüütik on mingil moel võimeline seda tegema, kuid ei täpsustata, kuidas seda teha. Siiski näitab praktika, et parameetrite hindamine on riskihalduse protsessi juures turvaanalüütikute jaoks üks keerulisemaid etappe. Käesoleva töö üks tähtsamaid panuseid on kolmandas peatükis antud praktilised juhised, mis aitavad leida vajalikku informatsiooni ning selgitavad, milliseid aspekte tuleb arvestada turvaparameetrite hindamisel. Samuti esitas antud töö autor oma nägemuse ründepuude metoodika arengusuundadest.

Töö käigus valmis tarkvaraline raamistik, mis võimaldab mugavalt koostada ja arvutada ründepuid. Kuna teoreetilised ründepuude mudelid on veel arenemisjärgus, siis oli raamistik disainitud võimalikult paindlikuks, mis lihtsustab selle kohandamist järjest areneva teooriaga ning võimaldab seda tulevikus edasi arendada. Töös on detailsemalt kirjeldatud raamistiku arhitektuur ning selle laiendamise võimalused.

Praeguseks hetkeks on see ainuke tarkvaratoode, mis võimaldab arvutada ründepuid, kasutades antud teadussuuna värskeimaid teoreetilisi tulemusi. Raamistik lihtsustab olulisel määral turvaanalüütiku tööd ning teeb ründepuude metoodika praktikas kasutatavamaks. Seega on valminud tarkvaraline tööriist tähtis panus ründepuude alale.

Summary

To be able to ensure the security of a system it is essential to know a) how secure it currently is, b) how secure it would be if some countermeasures were employed, c) how to choose the most effective security measures and d) how much is rational to spend on security. Therefore, we need some security metrics that could be used to measure the security of a system and to conduct a risk analysis.

The development of such security metrics has been an important research area of information security, although exact security metrics have still not been created. This is hampered by the fact that absolute security does not exist – a system can only be secure against some concrete attacker under certain circumstances.

Lately, it has been realized that an economic approach to security is beneficial against gain-oriented attackers. A system is considered secure if breaking it is economically inexpedient for the adversary. This master's thesis provides an overview of attack tree methodology, focusing on the economic aspects of security. The methodology is currently not a security metrics in the strict sense due to its dependence on expert estimates, but it is one step closer to the quantification of security.

To be able to use attack trees one needs to estimate the security parameters of simpler elementary attacks. Most research assumes that security experts are capable of doing this, but does not specify exactly how it is done. However, estimating the values of security parameters is one of the hardest steps in the risk management process in practice. One of the main contributions of this master's thesis is a set of practical recommendations on information sources and on aspects that need to be kept in mind when estimating parameters. Also, the author of this thesis presented his own vision on future research areas and evolution of the attack tree methodology.

One of the results of the work was the development of a software framework that allows us to create and compute attack trees. As theoretical models of attack trees are still rather immature, the framework was designed as flexible as possible, simplifying its adaption to evolving theory and allowing further development in future. The thesis provides a detailed description of the architecture and the possibilities of extending the framework.

To date, it is the only software product that allows to compute attack trees using the freshest theoretical results in that area of research. The framework significantly simplifies the work of security experts and makes attack tree methodology more usable in practice. Therefore, the developed software tool is an important contribution to the research in the area of attack trees.

Viited

[1] William Thomson (Kelvin), Electrical units of measurement, Institution of Civil Engineers, 3.5.1883, The practical applications of electricity, London, 1884, p 149.

[2] Ahto Buldas, Peeter Laud, Jaan Priisalu, Märt Saarepera, Jan Willemson, Rational Choice of Security Measures via Multi-Parameter Attack Trees, in Critical Information Infrastructures Security First International Workshop, CRITIS 2006, Samos Island, Greece, August 31 - September 1, 2006. LNCS 4347, lk. 235-248.

[3] Hard Problem List, INFOSEC Research Council, November 2005, `http://www.cyber.st.dhs.gov/docs/IRC_Hard_Problem_List.pdf`

[4] National Cyber Security Research and Development Challenges Related to Economics, Physical Infrastructure and Human Behavior: An Industry, Academic and Government Perspective, The Institute for Information Infrastructure Protection (I3P), 2009, `http://www.thei3p.org/docs/publications/i3pnationalcybersecurity.pdf`

[5] Wayne A. Jansen, Directions in security metrics research, National Institute of Standards and Technology, Gaithersburg, MD, March 2009.

[6] Stuart E. Schechter, Toward Econometric Models of the Security Risk from Remote Attack, IEEE Security and Privacy, vol. 3, no. 1, pp. 40-44, Jan. 2005, doi:10.1109/MSP.2005.30.

[7] Shirley C. Payne, A Guide to Security Metrics, SANS Security Essentials GSEC Practical Assignment, Version 1.2e, June 19, 2006.

[8] R. Anderson, Why Information Security is Hard-An Economic Perspective, Proceedings of the 17th Annual Computer Security Applications Conference, p.358, December 10-14, 2001.

[9] V. Verendel. Quantified security is a weak hypothesis: a critical survey of re- sults and assumptions. In NSPW '09: Proceedings of the 2009 workshop on New security paradigms workshop, pages 37–50, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-845-2. doi: `http://doi.acm.org/10.1145/1719030.1719036`.

[10] Bruce Schneier, Secrets and Lies: Digital Security in a Networked World, John Wiley & Sons, 2004.

[11] Stuart E. Schechter. Quantitatively differentiating system security. In The First Workshop on Economics and Information Security, May 16-17, 2002.

[12] Ross Anderson, Tyler Moore. The Economics of Information Security: A Survey and Open Questions. Fourth bi-annual Conference on the Economics of the Software and Internet Industries, January 19-20, 2007, Toulouse, France.

[13] Vello Hanson, Märt Laur, Monika Oit, Kristjan Alliksoo. Infosüsteemide turve. 1. osa: Turvarisk. Tallinn, AS Cybernetica, 2009.

[14] Basel II: Revised international capital framework (BCBS), `http://www.bis.org/publ/bcbsca.htm`

[15] Adam Shostack, Experiences Threat Modeling at Microsoft, In Proceedings of MODSEC workshop in, association with MODELS'08. 2008. Toulouse, France.

[16] P. T. Leeson, C. J. Coyne. The Economics of Computer Hacking. Journal of Law, Economics and Policy, 2006.

[17] J. Rollins, C. Wilson. CRS Report for Congress. Terrorist Capabilities for Cyberattack: Overview and Policy Issues, 2007.

[18] Alessandro Acquisti, Jens Grossklags, Privacy and Rationality in Individual Decision Making, IEEE Security and Privacy, v.3 n.1, p.26-33, January 2005.

[19] A. Jürgenson, J. Willemson, On fast and approximate attack tree computations. In Proc. of 6th Int. Conf. on Security Practice and Experience Conf., ISPEC 2010 (Seoul, May 2010), Lect. Notes in Comput. Sci., Springer, to appear.

[20] Bruce Schneier. Attack trees: Modeling security threats. Dr. Dobb's Journal, 24(12): 21-29, Detsember 1999.

[21] Peng Liu, Wanyu Zang, and Meng Yu. Incentive-Based Modeling and Inference of Attacker Intent, Objectives and Strategies. ACM Transactions on Information and Systems Security, 8(1):78 –118, 2005.

[22] Vesely, W., Goldberg, F., Roberts, N., Haasl, D.: Fault Tree Handbook. US Gov- ernment Printing Office (January 1981) Systems and Reliability Research, Office of Nuclear Regulatory Research, U.S. Nuclear Regulatory Commission.

[23] Weiss, J.D.: A system security engineering process. In: Proceedings of the 14th National Computer Security Conference. (1991) 572–581.

[24] Jürgenson, A., Willemson, J.: Processing multi-parameter attacktrees with estimated parameter values. In Miyaji, A., Kikuchi, H., Rannenberg, K., eds.: Advances in Information and Computer Security, Second International Workshop on Security, IWSEC. Volume 4752 of LNCS., Springer (2007) 308–319.

[25] Mauw, S., Oostdijk, M.: Foundations of attack trees. In Won, D., Kim, S., eds.: International Conference on Information Security and Cryptology – ICISC 2005. Volume 3935 of LNCS., Springer (2005) 186–198.

[26] Alexander Opel. Design and implementation of a support tool for attack trees. Technical report, Otto-von-Guericke University, March 2005. Internship Thesis.

[27] Alexander Andrusenko. Mitmeparameetriliste ründepuude analüüsitarkvara. Bakalaureusetöö, Tartu Ülikool, 2008.

[28] Jürgenson, A., Willemson, J.: Computing exact outcomes of multi-parameter attack trees. In: On the Move to Meaningful Internet Systems: OTM 2008. Volume 5332 of LNCS., Springer (2008) 1036–1051.

[29] Andrusenko, A., Jürgenson, A., Willemson, J.: Serial Model for Attack Tree Computations. In: KSII Transactions On Internet And Information Systems, 2010, to appear.

[30] L. P. Swiler, C. Philips, T. Gaylor. A Graph-Based Network Vulnerability Analysis System. SandiaReport SAND97-3010/1, Sandia National Laboratories, January 1998.

[31] United Nations Statistics Division, `http://data.un.org/`

[32] The Columbia Earth Institute of Columbia University, `http://www.earth.columbia.edu`

[33] Global Assessment Report on Disaster Risk Reduction. United Nations, Geneva, Switzerland, 2009.

[34] Arnold, Margaret, and World Bank. Hazard Management Unit. Natural disaster hotspots case studies / edited by Margaret Arnold ... [et al.] World Bank Hazard Management Unit; Palgrave [distributor], Washington, D.C. : Basingstoke : 2006

[35] The MCEER (Multidisciplinary Center for Earthquake Engineering Research) Information Service, `http://mceer.buffalo.edu/infoservice`

[36] The Internation Disaster Database, CRED (Centre for Research on the Epidemiology of Disasters), http://www.emdat.be/

[37] Van Dijk, J.J.M., van Kesteren, J.N. & Smit, P. (2008). Criminal Victimisation in International Perspective, Key findings from the 2004-2005 ICVS and EU ICS. The Hague, Boom Legal Publishers.

[38] Walker, A., Flatley, J., Kershaw, C, Moon, D. Crime in England and Wales 2008/09, Volume 1. Home Office Statistical Bulletin 11/09. London: Home Office.

[39] Fafinski, S., Minassian, N., UK Cybercrime Report 2009, New York, Garlik, 2009.

[40] Information Security Breaches Survey 2010: Technical Report. Department for Business Enterprise & Regulatory Reform (BERR), http://uk.sitestat.com/pwc/uk/s?ukws.eng_publications.pdf.isbs_survey_2010.technical_report&ns_type=pdf

[41] R. Richardson. CSI Computer Crime and Security Survey. The 12th Annual Computer Crime and Security Survey, Computer Security Institute, 2008.

[42] Verizon Data Breach Investigations Report, http://verizonbusiness.com/resources/security/reports/2009_databreach_rp.pdf

[43] Miller, C. The legitimate vulnerability market: the secretive world of 0-day exploit sales, WEIS 2007 - Sixth Workshop on Economics of Information Security, Pittsburgh PA, 7-8 June 2007.

[44] Böhme, R. Cyber-insurance revisited. In Workshop on the Economics of Information Security (WEIS), Cambridge, MA, 2005.

[45] SecurityFocus Vulnerability Database, http://www.securityfocus.com/vulnerabilities

[46] Common Vulnerabilities and Exposures (CVE), http://cve.mitre.org

[47] Secunia Vulnerability Advisories, http://secunia.com/advisories/

[48] Klopets, U., Vahistamise analüüs, Justiitsministeerium, Kriminaalpoliitika osakond, Tallinn, 2009.

[49] Statistika andmebaas, http://pub.stat.ee/px-web.2001/Dialog/statfile2.asp

[50] Sillaots, M., Kokkuleppemenetlus kriminaalmenetluses, doktoriväitekiri, Tartu, 2004.

[51] Folker den Braber, Ida Hogganvik, Mass Soldal Lund, Ketil Stølen, Fredrik Vraalsen. Model-based security analysis in seven steps – a guided tour to the CORAS method. BT Technology Journal, 25(1):101-117, January 2007.

[52] Isograph kodulehekülg, http://www.isograph-software.com/

[53] Meland P., Spampinato D., Hagen E., Baadshaug E., Krister K., Velle K. (2008). SeaMonster: Providing tool support for security modeling. NISK 2008. National Conference on Information Security. November 2008.

[54] Qt framework, http://qt.nokia.com/

[55] Boost Graph Library (BGL), http://www.boost.org/doc/libs/1_43_0/libs/graph/doc

[56] Graphviz, http://www.graphviz.org/

[57] Java Universal Network/Graph Framework (JUNG), http://jung.sourceforge.net/

[58] Apache Commons Configuration, http://commons.apache.org/configuration/

Lisa 1

Raamistiku lähtekood CD-l