TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Mart Hütt 203970IAPM

KAHE, KONTSEPTI STABIILSUSEL BASEERUVATE, SÜSTEEMI ROLLIDE MINIMEERIMISE ALGORITMIDE TESTIMINE NELJA ERINEVA ANDMESTIKU PEAL

Magistritöö

Juhendaja: Ants Torim,
PhD

Tallinn 2024

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Mart Hütt

05.01.2024

Annotatsioon

Käesoleva lõputöö eesmärk on defineerida RBAC süsteemi kasutajarollide stabiilsus selliselt, et see kirjeldaks kui oluline iga kasutajaroll on süsteemi jaoks ja see võimaldaks tuvastada potentsiaalselt üleliigseid rolle süsteemis. Seejärel defineerida meetodid, mille eesmärgiks on minimeerida RBAC süsteemi kasutajarollide arv kasutades defineeritud stabiilsust. Eelduse, et rollide minimeerimine lihtsustab süsteemi hallatavust.

Kasutajarollide stabiilsuse ning meetodite testimiseks implementeeritakse need töö käigus ning valideeritakse neid kasutades nelja erisuurusega RBAC süsteemi andmestikuga. Mõõtes iga andmestiku puhul sooritatud muudatuste arvu, süsteemi keskmise stabiilsuse muutu, süsteemi minimaalse stabiilsuse muutu ning rollide koguarvu muutu.

Lõputöö tulemuseks on kasutajarollide stabiilsuse definitsioon RBAC süsteemide jaoks. Lisaks stabiilsuse definitsioonile, kaks algoritmi süsteemi rollide arvu minimeerimiseks, mis kasutavad defineeritud stabiilsust ning mille efektiivsus on testitud.

Töö tulemuste põhjal järeldus, et töökäigus loodud algoritmid vähendavad rollide arvu süsteemis ning suurendavad stabiilsust, kuid ei muuda RBAC süsteemi hallatavamaks.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 66 leheküljel, 6 peatükki, 13 joonist, 30 tabelit.

Abstract

Testing of two, concept stability based, role minimizing algorithms on four different datasets

The purpose of this thesis is to define what access role stability is for RBAC systems and use the definition to create two algorithms for minimizing system access roles. The purpose of which is to make systems more manageable.

For testing the definition of stability and the defined algorithms, they will be implemented and tested using four different size RBAC system datasets and their efficiency will be measured by the change of total number of rules, average stability, minimal stability and number of changes maid to the initial access matrix.

The result of this thesis is a definition of role stability for RBAC systems and two algorithms that minimise roles in a given system by utilising the defined stability.

As a result of this work, it was found that the implemented algorithms reduce the number of roles in a RBAC system and increase its stability, but didn't make it more manageable.

The thesis is in Estonian and contains 66 pages of text, 6 chapters, 13 figures, 30 tables.

Lühendite ja mõistete sõnastik

RBAC	*Role-based Access Control*, rollipõhine juurdepääsuõiguste juhtimine
DAC	*Discretionary access control,* diskteetne juurdepääsuõiguste juhtimine
MAC	*Mandatory access control,* kohustuslik juurdepääsuõiguste juhtimine
RE	*Role engineering*, rollitehnika
FCA	*Formal Concept Analysis*, Formaalne kontseptianalüüs
QC	QualityCover, FCA algoritm kontseptide moodustamiseks
GAIN	Kontsepti olulisuse hindamise funktsioon
CMS	Organisatsiooni andmestiku nimetus
AWS	Amazon-i pilveteenuse näidis andmestiku nimetus
NP	*Non-deterministic polynomial-time*, Mitte polünoome
MHS	*Minimal Hitting set*, Minimaalne ühisosa

Sisukord

Jooniste loetelu

Tabelite loetelu

1 Sissejuhatus

1.1 Taust

Kasutajate arv IT süsteemides on drastiliselt kasvanud viimaste aastate jooksul ning see on endaga kaasa toonud uusi probleeme. Selleks, et kõigil kasutajatel oleks olemas kõik vajalikud õigused neile määratud kohustuste täitmiseks, on tarvis efektiivset viis õiguste haldamiseks. Suur hulk erinevaid õiguseid ning kasutajaid ei ole mitte ainult andmehalduse probleemiks, vaid ka suureks turvariskiks. Õiguste valesti määramine võib endaga kaasa tuua andmelekke, nagu seda juhtus 2018. aastal USA haridusministeeriumis [17] või isegi süsteemi täielikku ülevõtmise kolmanda osapoole poolt.

IT maastiku suured ning kiired muutused on peamiseks põhjuseks, miks õiguste ning kasutajate arv on viimastel aastatel nii kiirelt kasvanud. Monoliitsetelt rakendustelt on ülemindud mikroteenustel põhinevale arhitektuurile, milles tehakse iga funktsionaalsuse jaoks eraldi rakendus. Kõik need rakendused omavad endajaoks spetsiifilisi õiguseid, mida on tarvis hallata selleks, et rakendused töötaks korrektselt ning rakendused saaksid omavahel suhelda. Teiseks kiire kasvu põhjustajaks on pilveteenuste, nagu näiteks AWS, laialdane kasutuselevõtmine, milles kogu funktsionaalsus on õiguste põhine [18].

Hetkel on kõige populaarsem raamistik õiguste ja kasutajate sidumuseks rollipõhine juurdepääsuõiguste juhtimine (RBAC) [19]. RBAC süsteemis piiratakse erinevate kasutajate tegevusi rollidega. Ning rollide moodustamiseks RBAC süsteemi jaoks kasutatakse tehnikat nimetusega rollitehnika (*role engineering*) [20]. Isegi kui süsteemi jaoks on juba mingi rollid loodud siis aitab rollitehnika nende korrektsust ning vajalikust valideerida, kuna õigesti tuvastatud ja defineeritud rollid on turvalise infosüsteemi alustalaks [20].

1.2 Eesmärgid

Käesoleval lõputööl on kaks eesmärki, primaarne ja sekundaarne. Sekundaarseks on defineerida, mis on RBAC süsteemi kasutajarollide stabiilsus, võttes aluseks S. O. Kuznetsov'i poolt defineeritud formaalse kontsepti stabiilsuse [6]. Kontsepti stabiilsus on mõõt mille põhjal on võimalik hinnata erinevate hüpoteeside usaldusväärsust, ja seda on kasutatud eelnevalt andmekaeves mustrivalikuks [11]. Selles töös proovitakse defineerida RBAC süsteemi kasutajarollide stabiilsust viisil, mis kirjeldaks kui oluline on iga kasutajaroll süsteemi jaoks ja see aitaks tuvastada potentsiaalselt üleliigseid rolle süsteemis. Kuna süsteemi õiguste maatriksit on võimalik vaadelda kui formaalse kontseptianalüüs (FCA) konteksti, siis sellega on olemas kõik teoreetilised eeldused, et arvutada stabiilsused RBAC süsteemi kasutajarollide jaoks.

Primaarseks eesmärgiks on testida kahte hüpoteetilist meetodit, mille ülesandeks on minimeerida kasutajaterollide arvu süsteemis. Minimeerimine aitaks muuta süsteemi rolle ülevaadlikumaks ning hallatavamaks. Testitavateks meetoditeks on hüpoteetilised viisid, mis selle lõputöö raames välja töötati, et eemaldada üleliigseid kasutajarolle süsteemist kasutades rollide defineeritud stabiilsusest. Peale potentsiaalselt üleliigse kasutajarolli tuvastatust sooritatakse muudatus algses õiguste maatriksis, eemaldades rolliga seotud õigused kasutajatelt. Peale igat muudatust vaadeldakse, kuidas sooritatud muudatus mõjutas süsteemi kasutajarollide koguarvu ning kasutajarollide stabiilsust.

Ärilises mõttes toimuks kasutajaga seotud rollide asendus üksikute õigustega ning seda võimalikult väheste muudatustega õiguste maatriksis. Ideaalsel juhul oleks võimalik asenda üks roll ühe üksik õiguse omistamisega kasutajale. Minimeerides rollide arvu ning asendades osad rollid üksikute õigustega süsteem peaks süsteem muutuma turvalisemaks, kuna turvalisuse mõttes on õiguse valesti omistamise potentsiaalsed tagajärjed väiksemad kui rolli valesti omistamisel.

1.3 Metoodika

Lõputöös kasutakse meetodite uurimiseks nelja erineva andmekogu ning kasutajarollide moodustamiseks QC algoritmi. Kolm vaadeldavat andmekogu, spordikeskus, intranet ning CMS, pärinevad Tanel Õunase magistritööst „Rollikaevandamise meetodite võrdlus kolme organisatsiooni andmete põhjal" [12] ja neljas AWS, on näidis Amazon-i pilveteenuse ligipääsumaatriks [13], millest on võetud 1023 korda 1024 suurune alamvalim. Andmekogud said valitud selliselt, et oleks võimalik võrrelda testitavate meetodit efektiivsust, kasutajarollide stabiilsuse arvutamisel ning üleliigsete kasutajarollide tuvastamisel, erinevate RBAC süsteemide suuruste puhul. QC algoritmi rakendamise kasuks otsustati, kuna tegemist on efektiivse FCA algoritmiga kontseptide loomiseks [8, 12] ning käesoleva töö eesmärk ei ole uurida rollide moodustamist, vaid nende eemaldamist.

Töö käigus on plaanis esmalt moodustada kasutajarollid ning arvutada igale rollile stabiilsus. Seejärel muuta algset õiguste maatriksid erinevate viisidel, tuginedes rollide stabiilsusele, et otsusta milliseid muudatusi sooritada. Peale muudatuste sooritamist moodustatakse kasutajarollid uuesti ja arvutada uute rollide stabiilsused. Saadud tulemuste puhul võrreldakse kuidas muutus süsteem tervikuna, võttes aluseks rollide koguarvu, keskmise rollide stabiilsuse, rollide minimaalse stabiilsus ning sooritatud muudatuste koguarvu. Keskmine rollide stabiilsuse ja minimaalne rollide stabiilsus on olulised mõõdud selleks, et hinnata, kas stabiilsus on RBAC süsteemide jaoks oluline näit, mille põhjal teha otsuseid. Rollide koguarv annab ülevaade kui efektiivne kindel minimeerimise algoritm on ja muudatuste koguarv ütleb kas minimeerimine on äriliselt mõistlik, kuna kõik muudatused õiguste maatriksis lisatakse tagasi kasutajatele üksikõigusena ja kui süsteemi ligipääsu haldamine ei muutu lihtsamaks ei ole algoritmi kasutamine mõistlik.

1.4 Töö struktuur

Lõputöö on jagatud kolmeks erinevaks etapiks. Esmalt vaadeldakse teoreetilisi aluseid, millele lõputöö baseerub. Täpsemalt, tutvustatakse mis on rollipõhine juurdepääsuõiguste juhtimine, rollitehnika ja formaalne kontsepti analüüs ning kuidas need on omavahel seotud. Seejärel defineeritakse, mis on kasutajarolli stabiilsus RBAC süsteemis ning tutvustatakse hüpoteetilisi viis kuidas saaks kasutada rollide stabiilsust potentsiaalselt üleliigsete kasutajarollide tuvastamiseks ja eemaldamiseks. Viimases etapis implementeeritakse ning katsetatakse eelnevas etapis tutvustatud lahendusi ning analüüsime nende efektiivsust.

2 Teoreetilised alused

Selle peatüki eesmärgiks on anda ülevaade teoreetilistest alustest, mida kasutatakse selle lõpputöö raames. Peatükis tutvustatakse, mis on rollipõhine juurdepääsuõiguste juhtimine, rollitehnika, formaalne kontseptianalüüs, kontsepti stabiilsus ja QC algoritm.

2.1 Rollipõhine juurdepääsuõiguste juhtimine

Rollipõhine juurdepääsuõiguste juhtimine (RBAC) on Ameerika Ühendriikide valitsuse poolt välja töötatud raamistik kasutajaõiguste haldamiseks. Vajadus RBAC-i järele tulenes asjaolust, et üha rohkem valitsusasutusi ja ettevõtteid sõltus tugevalt infosüsteemidest ning seoses sellega oli tarvis tagada infosüsteemide terviklikkus, kättesaadavus ja konfidentsiaalsus. Korruptsioon, volitamata avalikustamine või ettevõtte ressursside vargus võivad häirida organisatsiooni tegevust ja tekitada tõsise rahalise, õigusliku, ja või mainelise kahju [1]. Senini kasutatud DAC ja MAC raamistikud ei olnud enam asjakohased. DAC, kus kõigil kasutajatel olid kõik õigused süsteemi jaoks mille eest nad vastutasid, baseerub täielikult kasutajate diskreetsusel, mis on suureks turvariskiks. MAC, kus anti kasutajatel õiguseid ühekaupa, muutus raskesti hallatavaks suurte kasutaja hulkade puhul.

RBAC-i võib vaadata kui MAC-i edasiarendust. Selleasemel, et anda kasutajatele õiguseid üksikult, grupeeritakse õigused kokku rollideks. Õigused grupeeritakse rollideks selliselt, et iga roll sisaldaks ainult neid õigused, mida selle rolli täitja vajab sooritamaks kõiki talle määratud tegevusi.

RBAC jälgib järgnevaid põhimõtteid [1]:

- Iga tegutseja puhul on aktiivne ainult see roll, mida tegutseja hetkel kasutab: *AR(s: subject) = {the active role for subject s}*

- Igal tegutsejal võib olla õigus täita ühte või mitut rolli: *RA(s: subject) = {authorized roles for subject s}*
- Igal rollil võib olla volitused ühe või mitme tehingu tegemiseks: *TA(r: role) = {transactions authorized for role r}*
- Subjektid võivad tehinguid sooritada. Predikaat exec(s, t) on tõene, kui subjekti s saab täita tehing t praegusel ajal, vastasel juhul on see vale: *exec(s: subject, t: tran) = true iff subject s can execute transaction t*

2.2 Rollitehnika

Rollitehnika (RE) on metoodika rollide defineerimiseks ning sidumiseks erinevate tegevusõigustega selliselt, et need oleksid organisatsiooni seisukohast lähtudes ärilises mõttes vajalikud ning loogilised. RE-d võib vaadelda kui RBAC-i eeldust, kuna RE peamiseks kasutusvaldkonnaks on RBAC-i erinevate komponentide defineerimine. RE võimaldab moodustada õigesti rolle, mis on suurte ja või komplektsete organisatsioonide infosüsteemide juhtimise aluseks [2].

Lähenemisviise, kuidas RE-d rakendada rollide moodustamiseks, võib jagada kolmeks: ülalt-alla (inglise keeles: *top-down*), alt-üles (inglise keeles: *bottom-up*) ja hübriid (inglise keeles: *hybrid*). Ülalt-alla lähenemise puhul moodustatakse rollid lähtudes ettevõtte funktsioonide ja eesmärkide analüüsist. Tegemist on enamasti manuaalse protsessiga, mille käigus defineeritakse rollid nullist. Seda protsess sooritatakse enamasti korduvalt ning sõltumatute üksuste poolt. Korduvalt selleks, et tagada rollide põhjalikkust ning piisav granulaarsus. Ühekordne läbimine võib paljastada mitmeid ilmseid rollid, kuid need ei pruugi pikas perspektiivis olla piisavalt täpselt defineeritud ettevõtte optimaalseks tegutsemiseks [2]. Alt-üles lähenemine, mida kutsutakse ka rollikaevandamiseks (inglise keeles: *role mining*), kasutab rollide moodustamiseks olemasolevaid seoseid kasutajate ja õiguste vahel. Sõltuvus olemasolevatest seostest tähendab, et loodud rollid kvaliteet sõltub tugevalt sisendandmete kvaliteedist [2]. Hübriidlähenemise, mis on kombinatsioon esimesest kahest, puhul määratakse rollid esialgu kasutades alt-üles lähenemisviis ja selle tulemust hiljem täiendatakse teabe

põhjal, mis on saadud ülalt-alla lähenemisviisilt [2]. Käesolevas lõputöö võrreldakse meetodeid, mis kasutavad alt-üles lähenemist.

2.3 Formaalne kontseptianalüüs

Formaalne kontseptianalüüs (FCA) on Rudolf Wille poolt 1982. aastal välja töötatud meetod andmeanalüüsi, informatsiooni esitluse ning informatsiooni haldamise jaoks. FCA käsitleb kahte tüüpi elemente: „formaalsed objektid" ja „formaalsed atribuudid". FCA eesmärk on struktureerida, eelnevalt mainitud atribuuti ja objekti hulkade vahel eksisteerivaid seoseid „formaalseteks kontseptideks" [3]. Loodud kontseptid kirjeldavad objekte ja nendele kuuluvaid omadusi. FCA puhul võib paralleele tuua klasterdamisega, mõlemas koondatakse sarnaste omadustega objektid kokku. Paralleele saab ka tuua kasutajate ning rollidega, mis on oluline omadus selle lõputöö jaoks. Näiteks, kasutajad oleks objektid ning õigused oleks omadused. Tänapäeval on FCA kasutuses valdkondades, kus töödeldakse suuri andmehulki nagu näiteks keeleteadus, tarkvaraarendus, masinõppe, psühholoogia ja teised [3].

2.3.1 Formaalne kontekst

FCA formaalsed objektid ja formaalsed atribuudid koos seostega loovad vaadeldava ruumi ehk formaalse konteksti. Omadussõna "formaalne" kasutatakse rõhutamaks, et need on formaalsed mõisted ning need ei pea olema „objektid" ja „atribuudid" mingis konkreetses mõttes. Objektide ja atribuudid vasted reaalses maailmas võivad olla näiteks: koolid ja erialad, loomad ja nende karakteristikud või kasutajad ja õigused. Definitsiooni kohaselt formaalne kontekst K kontseptianalüüsis koosneb kolmikust (G, M, I), kus G on hulk formaalseid objekte, M on hulk formaalseid atribuute ja kontekst I $\subseteq$ G x M on binaarne seos objekti ja atribuudi vahel [4]. Üks levinumaid viise formaalse konteksti esitamiseks on binaarne maatrikstabel, nagu näiteks tabelis 1., milles objektid on veergudes ja atribuudid on ridades.

2.3.2 Formaalne kontsept

Kui formaalne kontekst vaatab kogu formaalsete objektide- ja atribuutide hulka siis formaalne kontsept on paar (A, B), kus $A \subseteq G$ ja $B \subseteq M$, $A' = B$, ja $A' = A$. A on alamhulk formaalsete objektide hulgast, millele on ühised atribuudid ja B on alamhulk formaalsete atribuutide hulgast, mis esinevad teatud objektidel. Alternatiivselt võib sõnastada, et A' kirjeldab atribuutide hulka, mis on ühised kõigile objektidele hulgas A ja B' kirjeldab objektide hulka, mis on ühised kõigile atribuutidele selles atribuutide hulgas B [5].

$$A' := \{m \in M \mid \forall\, g \in A : gIm\}$$

$$B' := \{g \in G \mid \forall\, m \in B : gIm\}$$

Formaalse kontsepti formaalsete objektide alamhulka A nimetatakse ekstentsiooniks ja formaalsete atribuutide hulk B nimetatakse intentsiooniks.

Tabel 1. Formaalse konteksti lihtne õigusmaatriksi näidis

	Juhataja	**Raamatupidaja**	**Laopidaja**	**Müüa**
Müük	**X**	**X**		**X**
Inventuur	**X**	**X**	**X**	
Arved	**X**	**X**	**X**	
Seadistus	**X**			

Võttes aluseks ühe paari (A, B) formaalse konteksti näidisest (Tabel 1.), kus A on objektide hulk {Inventuur, Arved} ja B on atribuutide hulk ehk intentsioon {Juhataja, Raamatupidaja, Laopidaja}, saame nende binaarsete seoste ($I \subseteq G \times M$) abil kätte ühe kontsepti ({Inventuur, Arved}, {Juhataja, Raamatupidaja, Laopidaja}), mida võib vaadelda kui rolli ja selle omistamist.

2.4 Kontsepti stabiilsus

Kontsepti stabiilsuse on mõõt mille põhjal on võimalik hinnata erinevate hüpoteeside usaldusväärsust. Oletame, et eksisteerivad punktid x_1, ..., x_n ruumi R^n jaoks ning kasutades neid punkte on võimalik ehitada polünoom, mille aste ei ole suurem kui n, mis rahuldab kõiki tingimusi. Nüüd oletame, et on võimalik konstrueerida polünoom P punktide alamhulgast $\{x_{i1}, ..., x_{ik}\} \subseteq \{x_1, ..., x_n\}$ nii, et kõik punktid x_1, ..., x_n asuvad P kõveral ning polünoomi aste ei ole suurem kui k. Kontsepti stabiilsuse väitab, et hüpoteesi P korrektsus, mis on antud $\{x_{i1}, ..., x_{ik}\} \subseteq \{x_1, ..., x_n\}$ on lihtsam ja usaldusväärsem hüpoteesina kui polünoom, mis on konstrueeritud kogu hulgast $\{x_1, ..., x_n\}$ [2]. Kontsepti stabiilsuse numbriliseks esinduseks kasutakse stabiilsuse indeksit, mis on suhtearv kõigi ekstenstiooni alamhulkade vahel, mis kinnitavad hüpoteesi ja on vahemikus 2 ... n-1 ning kõigi võimalike sama suurusega alamhulkade vahel.

Formaalse konteksti K = (G, M, I) ja formaalse kontsepti C = (A, B) jaoks on defineeritud stabiilsus järgnevalt:

$$\langle C\rangle_j = \{Y \subset A \mid |Y| = j,\ Y' = B\},\ \langle C\rangle_\Sigma = \bigcup_{j=2}^{n-1} \langle C\rangle_j,$$

$$\gamma_j(C) = |\langle C\rangle_j|,\ \gamma_\Sigma(C) = |\langle C\rangle_\Sigma|,\ n = |A|.$$

Summeerimise piirid tulenevad FCA-st, mille definitsiooni kohaselt ei saa ekstenstiooni suurus olla väiksem kui 2 [9].

Kontsepti stabiilsus indeks j-nda tasenadi jaoks ($2 \leq j \leq n - 1$) on defineeritud kujul:

$$J_j(C) = \frac{\gamma_j(C)}{\binom{n}{j}},$$

ja terve kontsepti stabiilsus indeks:

$$J_{\Sigma}(C) = \frac{\gamma_{\Sigma}(C)}{2^n - n - 2}$$

Kuna kõikide võimalike alamhulkadega, suurusega n − 1, n – 2, n – 3 jne., kasutamine on väga ressursi kulukas, siis kasutatakse kahte alternatiivset lähenemist: noameetod (*jackknife method*) ja *Bootstrap* meetod. Noameetodi puhul leitakse kõik alamvalimid suurusega n – 1 ja i-nda alamvalimi jaoks arvutatakse välja uuritava statistika S väärtuse S_i. Võttes keskmist S_i väärtuste S_*, siis arvutame S_* kõrvalekallete S_i ruutude keskmise. Tulemus (ebaolulise aritmeetilise teisenduse piires) annab hinnangu statistika S. *Bootstrap* meetod hindab dispersiooni statistikat sarnasel viisil, selle erinevusega, et uued valimid, samuti suurusega n, on genereeritud esialgsest proovist n-kordse valikuga koos asendamisega (iga element Esialgse valimi osa võib uues valimis esineda nullist n korda).

Tabel 2. Kontsepti stabiilsuse näide - rehvid

	Mugavus (m)	Lumi (l)	Jää (j)	Vastupidavus (v)
1	1.9	1.4	1.8	2.7
2	2.1	0.8	3.8	2.3
3	1.7	1.9	1.6	3.7
4	1.7	2.0	2.4	3.4

Tabel 3. Kontsepti stabiilsuse näide – grupeerimise väärtused

	Grupp 1. (g1)	Grupp 2. (g2)	Grupp 3. (g3)	Grupp 4. (g4)
Mugavus	≤ 2.1	≤ 2.5	> 2.5	
Lumi	≤ 2.0	> 2.0		
Jää	≤ 2.4	≤ 3.0	≤ 4.0	> 4.0
Vastupidavus	≤ 3.0	≤ 3.7	> 3.7	

Näiteks, püstitame hüpoteesi lähtudes tabeli 4. andmetest ning kasutades tabeli 3. grupeerimise loogikat. Tabelis 4. on kirjeldatud autorehvi eksperdi hinnangud erinevate omaduse kohta, mis koosneb minimaalsetest väärtustest selliselt, et kõik nelja vaadeldavate rehvi jääks samasse grupeerimise vahemikku. Hüpoteesiks mille usaldatavust testida soovime on: rehvide mugavus langeb gruppi 1., lumel sõit gruppi 1, jää gruppi 3. ja vastupidavust gruppi 3. FCA-st lähtudes saame konteksti millel on intentsiooniks {g1, g1, g3, g2} ja ekstensioon {1, 2, 3, 4}. Vaadeldes ekstensiooni alamhulki, suurusega n – 1, {1, 2, 3}, {1, 2, 4}, {1, 3, 4} ja {2, 3, 4} näeme, et {1, 3, 4} ei grupeeru jää näitaja järgi samamoodi nagu algne kontsept. Alamavalmi {1, 3, 4} puhul saaks moodustada minimaalse intentsiooniks {g1, g1, g3, g1}, kuna maksimaalne jää näit on 2.4, mis langeks Grupp 1. Nüüd teeme sama alamhulkadega mille suurus on n-2 ehk 2. Tulemuseks saame, et kuuest võimalikust alamhulgast ainult 2, {2, 3} ja {2, 4}, säilitavad algse. Püstitatud hüpoteesi stabiilsuseks on hüpoteesi kinnitavad alamhulkade kogus jagatud kõigi võimalikega, 5/10 = 0.5.

2.5 QualityCover

QualityCover (QC) on FCA algoritm formaalsete kontseptide loomiseks, mis lahendab üht formaalses kontseptianalüüsi peamist probleemi. FCA-s tekitatakse formaalsetest kontekstidest liiga suurel arvul formaalseid kontsepte. QC algoritm loob minimaalse kontseptid hulga, kasutades ahnet lähenemist, mis katab kogu formaalse konteksti [8]. QC algoritmi on üles ehitatud tsüklile, mille igas iteratsioonis proovitakse moodustada kontsept selliselt, et see kataks kõige suurema võimaliku arvu konteksti elementidest, mis ei ole veel kaetud. Selleks, et hinnata loodud kontseptide headust kasutab QC algoritm GAIN funktsiooni [8].

3 Rollide stabiilsus ja selle kasutus rollide minimeerimisel

Selles peatükis vaatame, mis on RBAC süsteemi kasutajarollide stabiilsus ning tutvustame ideid, kuidas saaks kasutada kasutajarollide stabiilsust selleks, et tuvastada süsteemi jaoks üleliigseid rolle ning see järel need eemalda, et luua lihtsamini hallatav süsteem.

3.1 Süsteemi rollide stabiilsus

RBAC süsteemides on kolm olulist komponenti tegevuste juhtimiseks: õiguste maatriks, rollid ning rollide omistamine kasutajatele. FCA komponendid, mis vastavad nendele kolmele on kontekst, mis on sisult ekvivalentne õiguste maatriksile ning ekstentsiooni ja intentsiooni, mis on vastavalt rollid ning nende omistamine. Sõltuvalt õiguste maatriks orientatsioonist saame ekstentsiooni, mis kirjeldab süsteemi rolle ja intentsioonid, mis defineerib rollide omistamist kasutajatele. Seda juhul kui maatriksi read kirjeldavad õigusi ning tulbad süsteemi kasutajaid. Transponeerides õiguste maatriks, vahatavad intentsioonid ja ekstentsioonid tähenduse.

Lähtudes kontsepti stabiilsuse definitsioonist: stabiilsus on suhtearv ekstentsiooni alamhulkade, mis kinnitavad testitavat hüpoteesi ja kõikvõimalike alamhulkade vahel [6]. Käesoleva töö eesmärgiks on minimeerida rollide arv süsteemis. Selleks, et seda saavutada peaks iga roll katma võimalikult suure ala õiguste maatriksist. Hüpoteesiks, mida stabiilsus testib on: iga kontsept katab kontekstist maksimaalse ala. Selleks, et arvutada stabiilsust proovime kontrollime, kas valitud ekstentsiooni alamhulk ja intentsiooni kontsept katavad maksimaalse ala õiguste maatriksist.

Definitsioon: Rolli stabiilsus on suhtearv ekstentsiooni alamvalimite, suurusega 2 ... n-1, mille puhul ei ole võimalik intentsiooni laiendada ja kõigi võimalike ekstentsiooni alamvalimite, mis on suurusega 2 ... n-1, vahel.

Mida kasutajarolli stabiilsuse näitab on asjaolu, kui oluline on mingi õigus või kasutaja süsteemi jaoks. Kui kaob madala stabiilsusega rollist, kas teatud kasutaja või õigus siis ei pruugi seda rolli sellisel kujul nagu see hetkel defineeritud on enam tarvis olla. Kuna eksisteerib võimalik roll, mis tagaks samad õigused samadele kasutajate, kuid ei kaotaks tähtsust ühe või kahe elemendi (kasutaja, õigus) kadumisel.

Tabel 4. Näidis õiguste maatriks

	Kasutaja 1	Kasutaja 2	Kasutaja 3	Kasutaja 4
Õigus 1	X	X	X	
Õigus 2	X	X	X	X
Õigus 3	X	X	X	X
Õigus 4	X	X	X	X
Õigus 5	X	X	X	X
Õigus 6		X	X	X

Vaatame näidis süsteemi ligipääsu õiguste maatriksit (Tabel 5.), mille jaoks moodustati kontseptid kasutades QC algoritmi. Tulemuseks saadi 2 kontsepti A = ({k1, k2, k3}, {õ1, õ2, õ3, õ4, õ5}) ja B = ({k2, k3, k4}, {õ2, õ3, õ4, õ5, õ6}).

Tabel 5. Näidis QC kontseptidest, intentsioon ja ekstentsioon

Kontsept	Intentsioon	Ekstentsioon
A	{0, 1, 2}	{0, 1, 2, 3, 4}
B	{1, 2, 3}	{1, 2, 3, 4, 5}

Arvutame definitsiooni järgi stabiilsuse kontsepti A jaoks. Esmalt leiame kõik võimalikud ekstentsiooni alamvalimid ja see järel leiame alamavalimid, mida ei saa laiendada.

Tabel 6. Näidis ekstentsioon alamvalimid

Kõik võimalikud alamavalimid	Alamavalimid, mida ei saa laiendada
{(0, 1), (0, 2), (0, 3), (0, 4), (1, 2), (1, 3), (1, 4), (2, 3), (2, 4), (3, 4), (0, 1, 2), (0, 1, 3), (0, 1, 4), (0, 2, 3), (0, 2, 4), (0, 3, 4), (1, 2, 3), (1, 2, 4), (1, 3, 4), (2, 3, 4), (0, 1, 2, 3), (0, 1, 2, 4), (0, 1, 3, 4), (0, 2, 3, 4), (1, 2, 3, 4)}	{(0, 1), (0, 2), (0, 3), (0, 4), (0, 1, 2), (0, 1, 3), (0, 1, 4), (0, 2, 3), (0, 2, 4), (0, 3, 4), (0, 1, 2, 3), (0, 1, 2, 4), (0, 1, 3, 4), (0, 2, 3, 4)}

Tulemuseks saame stabiilsuse 14/25 = 0.56. Kontsept A puhul tuleb välja, et on võimalik laiendada kõiki alamvalimid, mis ei sisalda ekstentsiooni õ1, kasutaja k4 võrra.

Tabel 7. Näidis kontsepti laiendamisest

	k1	k2	k3	k4
õ1	X	X	X	
õ2	X	X	X	X
õ3	X	X	X	X
õ4	X	X	X	X
õ5	X	X	X	X
õ6		X	X	X

3.2 Rollide minimeerimine võtme elementide eemaldamise abil

Lähtudes hüpoteesist mida testida soovime ning defineeritud süsteemi kasutajarollide stabiilsusest võiks väheneda süsteemi rollide arv ning kasvada rollide stabiilsus kui eemaldada algest õiguste maatriksit sellised kasutajatele omistatud õigused, mis piirasid intentsiooni laiendamist teiste õiguste maatriksi tulpade arvelt. Kuna kasutajarolli stabiilsus on pöördvõrdeliselt seotud laiendamist takistavate ekstentsiooni alamvalimite hulgaga siis oleks mõistlik eemaldada kõige madala stabiilsusega roll, selline lähenemine peaks minimeerima rollid minimaalse muudatuste arvuga.

Tabel 8. Näidis õiguste maatriks koos eemaldatavate elementide märgistusega

	Kasutaja 1	Kasutaja 2	Kasutaja 3	Kasutaja 4
Õigus 1	X	X	X	
Õigus 2	X	X	X	X
Õigus 3	X	X	X	X
Õigus 4	X	X	X	X
Õigus 5	X	X	X	X
Õigus 6		X	X	X

Tabelis 8. on märgitud punasega eemaldatavad kasutaja õigused, kollasega on õigused, mis tuleks eemaldata kui need ei oleks kaetud kontsepti B poolt ja rohelisega on mitte muutuvad kasutajaõigused.

Asjaolu, mis ilmnes näidis õiguste maatriksi rollide stabiilsuse arvutamise oli, et ühine osa kõigil alamvalimitel, mis takistavad kontsepti laiendamist on õigus nr. 1. Lähtudes sellest, ei oleks tarvis eemaldada kõiki õiguseid, vaid minimaalne ühisosa kõigi alamvalimitel seas. Leidud elemente, mida eemaldada, kutsutakse edaspidi võtme elementideks.

3.2.1 Minimaalse ühisosa

Minimaalse ühisosa *(MHS)* probleem on NP keerukusega ning tegeleb küsimusega: kuidas leida minimaalne hulk elemente nii, et igast vaadeldavast hulgast oleks vähemalt üks esindaja. Probleem on levinud arvutiteaduses ning puudutab valdkondi nagu diskreetset matemaatikat, andmekaevet ja teised [14]. Selles lõputöös on kasutatakse MHS implementatsiooni[1], mis kasutab ühisosa leidmiseks *Hitting-Set* puu algoritmi [15, 21].

Kasutades Tabelis 6. loetletud alamhulki, mis piiravad laiendamist annab algoritm minimaalseks tulemuseks 0. Rakendades leitud MHS tulemust algse õigusmaatriksi muutmiseks, tulemuseks on järgnev tabel.

Tabel 9. Näidis võtme elementide eemaldamise tulemusest

	Kasutaja 1	Kasutaja 2	Kasutaja 3	Kasutaja 4
Õigus 1				
Õigus 2	X	X	X	X
Õigus 3	X	X	X	X
Õigus 4	X	X	X	X
Õigus 5	X	X	X	X
Õigus 6		X	X	X

1 https://github.com/TheMatjaz/minihit

3.3 Rollide minimeerimine kasutades rolli unikaalset osa

Vastandiks võtme elementide eemaldamisele oleks kogu rolli eemaldamine, kuid ilma muutmata teiste rollidega seotud osa õiguste maatriksis. Selle eelduseks oleks leida eemaldatavad rolli unikaalne osa õiguste maatriksis, mida on võimalik leida analüüsides õiguste katte maatriksit. Õiguste katte maatriksit näitab, mitu erinevat rolli katab kasutajale õiguse omistamist. Õiguste katte maatriksi leidmiseks kasutatakse intentsioon ja ekstentsiooni binaarmaatriksite korrutist [10]. Kasutades saadud maatriksit ning rollide stabiilsust selleks, et eemaldada kõige madalama stabiilsusega rolli õiguste omistamine sealt kus kõige madalama rolli intentsioon ja ekstentsioon langevad kokku ning kattuvuse maatriksis on 1. See tagab asjaolu, et õigused eemaldatakse ilma muutmata teiste rollidega seotud osa õiguste maatriksis.

Vaadates kuidas sellise lähenemise puhul on seotud muudatuste arv ja rolli stabiilsus on näha tabelites 10, 11 ja 12, et kõige madalama stabiilsusega rolli valimine ei garanteeri minimaalset muudatuste arvu, kuid kui vaadata muudatuste arvu ekstentsiooni suuruse kohta (muudatuste arv / ekstentsiooni suurus) siis kahel juhul kolmest on paremate seas. Muudatuste arv ekstentsiooni kohta näitab kui suurt ala muudatus mõjutab õiguste maatriksis ning käesolevas töös keskendutakse kasutaja rolli stabiilsusele, kui otsustajale.

Tabel 10. Muudatuste arvu ja stabiilsuse seos lähtudes ülekatte maatriksist, andmestik spordikeskus

Rolli nr.	Stabiilsus	Muudatuste arv	Muudatuste arv ekstentsiooni kohta
1	1.0	4	1.0
2	0.8618378	7	0.4375
3	0.56	5	1.0
4	0.77048930	3	0.15789
5	0.3	4	1.0

Tabel 11. Muudatuste arvu ja stabiilsuse seos lähtudes ülekatte maatriksist, andmestik intranet

Rolli nr.	Stabiilsus	Muudatuste arv	Muudatuste arv ekstentsiooni kohta
1	1.0	2	2.0
2	1.0	2	2.0
3	0.3949579	2	0.28571428571428
4	1.0	14	7.0
5	0.3	1	0.25
6	0.66666666	0	0.0
7	0.44	2	0.4
8	0.28	1	0.2
9	0.3	1	0.25

Tabel 12. Muudatuste arvu ja stabiilsuse seos lähtudes ülekatte maatriksist, andmestik CMS

Rolli nr.	Stabiilsus	Muudatuste arv	Muudatuste arv ekstentsiooni kohta
1	0.999963395439	0	0.0
2	0.999963395	0	0.0
3	0.9977929960	15	1.0
4	0.98373983739837	0	0.0
5	0.99671338	13	0.43333
6	0.99410319	11	1.0
7	1.0	2	1.0
8	0.38617886178861	16	2.0
9	0.72139015	8	0.1355932

Kasutades QC algoritmi ja näidis õiguste maatriksit tabelis 3. saame järgnevad intentsioon ja ekstentsiooni binaarmaatriksid.

Tabel 13. QC algoritm näidis ekstensiooni binaarne maatriks

1	0
1	1
1	1
1	1
1	1
0	1

Tabel 14. QC algoritm näidis intentsiooni binaarne maatriks

1	1	1	0
0	1	1	1

Järgmisena leiame intentsioon ja ekstentsiooni binaarmaatriksi korrutise.

Tabel 15. Näidis kontekstide ülekattuvus

1	1	1	0
1	2	2	1
1	2	2	1
1	2	2	1
1	2	2	1
0	1	1	1

Õiguste maatriksi minimaalseks muutmiseks eemaldatakse õiguste maatrikisist kõige madalama stabiilsusega rolli selliste õiguste omistamised, mida muude rollidega ei kaeta. Kontekstide ülekattuvus näite puhul oleks see punane ala, mis eemaldatakse.

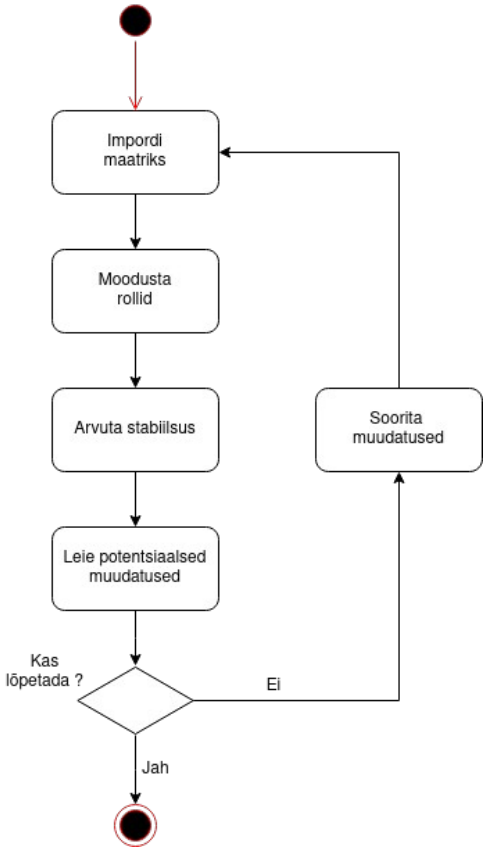
Tabel 16. Näidis konteksti ülekattuvus eemaldamise tulemusest

	Kasutaja 1	Kasutaja 2	Kasutaja 3	Kasutaja 4
Õigus 1				
Õigus 2		X	X	X
Õigus 3		X	X	X
Õigus 4		X	X	X
Õigus 5		X	X	X
Õigus 6		X	X	X

Tulemuseks on õigusmaatriks, mille puhul on võimalik katta kõik õiguste omistamised ühe rolliga.

4 Rollide optimeerimine kasutades stabiilsust

Selles peatükis vaatame eelmises peatükis välja käidud ideede implementatsioone. Implementeeritud rollide optimeerimise algoritmid baseerub lihtsal tsüklil, mis on kirjeldatud joonisel (Joonis 1.). Tsükli esimest kahte sammu, „Loe maatriks" ja „Defineeri rollid", või vaadelda rollide moodustamise sammuna. Rollide moodustamisele järgneb nende stabiilsuse arvutamine ja potentsiaalsete muudatuste leidmine, mis varieerub vastavalt testitavale lähenemisele. Saadud tulemuste põhjal sooritatakse otsus, kas leitud muudatuste sooritamine parandaks süsteemi stabiilsust oluliselt ning sooritada need või lõpetada tsükkel.



Joonis 1. Rollide optimeerimise sammud

4.1 Rollide moodustamine

Rollid moodustatakse kasutades QC algoritmi, mis on implementeeritud *python*[2] programmeerimise keeles, kasutades teeki *quality-covers*[3]. Algoritm saab sisendiks õiguste maatriksi ning väljundiks moodustatakse kolm faili: intentsiooni binaarse maatriksi, ekstentsiooni binaarse maatriksi ja metaandmed.

Tabel 17. Andmestiku sport õiguste maatriks

1	1	1	1	1	0	1	1	0
1	1	1	1	1	0	1	1	0
1	1	1	1	1	0	1	1	0
1	1	1	1	1	0	1	1	0
1	1	1	1	0	0	1	0	0
1	0	1	1	0	0	1	0	0
1	1	1	1	0	0	1	1	0
1	1	1	1	0	0	1	1	0
1	1	1	1	0	0	1	1	0
1	1	0	1	0	0	1	1	0
1	1	0	1	0	0	1	1	1
1	1	1	1	0	0	1	1	1
1	1	1	1	0	0	1	1	1
1	1	1	1	0	0	1	1	1
1	1	1	1	0	1	1	1	1
1	1	0	1	0	0	1	1	0
1	1	1	1	0	1	1	1	0
1	1	0	1	0	1	1	0	0
1	0	1	1	0	1	1	0	0
1	1	1	1	0	0	1	0	0
1	1	1	1	0	0	1	1	0

2 https://www.python.org/

3 https://pypi.org/project/quality-covers/

Kasudes spordikeskus andmestiku õigusmaatriksit (Tabel 10.) moodustati failid järgneva sisuga:

Tabel 18. QC algoritm ekstentsiooni binaarse maatriksi väljund näidis

1	1	1	0	1	0
1	1	1	0	1	0
1	1	1	0	1	0
1	1	1	0	1	0
1	0	0	0	1	0
1	0	0	0	0	0
1	0	1	0	1	0
1	0	1	0	1	0
1	0	1	0	1	0
0	0	1	0	1	0
0	0	1	1	1	0
1	0	1	1	1	0
1	0	1	1	1	0
1	0	1	1	1	0
1	0	1	1	1	1
0	0	1	0	1	0
1	0	1	0	1	1
0	0	0	0	1	1
1	0	0	0	0	1
1	0	0	0	1	0
1	0	1	0	1	0

Tabel 19. QC algoritm intentsiooni binaarse maatriksi väljund näidis

1	0	1	1	0	0	1	0	0
1	1	1	1	1	0	1	1	0
1	1	0	1	0	0	1	1	0
1	1	0	1	0	0	1	1	1
1	1	0	1	0	0	1	0	0
1	0	0	1	0	1	1	0	0

Saadud binaarsed maatriksid, vaadeldes koos, moodustavad kontseptid kui võtta ektentsiooni maatriksit tulp indeksiga x ja intentsiooni maatriksist rida indeksiga x.

Loodud kontseptid on sammuti välja toodud metaandmete failis koos lisa informatsiooniga, mis on huvitav, kuid mitte oluline selle lõputöö jaoks.

Tabel 20. QC algoritm metaandmete väljund näidis

Tähendus	Väärtus

Loodud kontseptid koos kontsepti spetsiifilise informatsiooniga	1,2,3,4,5,6,7,8,9,12,13,14,15,17,19,20,21; 1,3,4,7
	Mandatory#Object
	Uniformity=0.65518; Monocle=504.00000; Frequency=0.80952; Separation=0.57143
	
	1,2,3,4,5,6,7,8,9,12,13,14,15,17,19,20,21; 1,3,4,7
	Mandatory#Object
	Uniformity=0.65518; Monocle=504.00000;
	Frequency=0.80952; Separation=0.57143
	
	1,2,3,4; 1,2,3,4,5,7,8
	Mandatory#Object
	Uniformity=1.00000; Monocle=252.00000;
	Frequency=0.19048; Separation=0.23529
	
	1,2,3,4,7,8,9,10,11,12,13,14,15,16,17,21; 1,2,4,7,8 Mandatory#Object
	Uniformity=0.78162; Monocle=560.00000;
	Frequency=0.76190; Separation=0.65574
	
	11,12,13,14,15; 1,2,4,7,8,9
	Mandatory#Object
	Uniformity=0.86429; Monocle=240.00000;
	Frequency=0.23810; Separation=0.27778
	
	1,2,3,4,5,7,8,9,10,11,12,13,14,15,16,17,18,20,21; 1,2,4,7
	#Object
	Uniformity=0.65288; Monocle=426.00000;
	Frequency=0.90476; Separation=0.60800
	
	15,17,18,19; 1,4,6,7
	#Object
	Uniformity=0.66786; Monocle=135.00000;

Tähendus	Väärtus
	Frequency=0.19048; Separation=0.21053
Mitu kontsepti on kohustuslikud	# Mandatory: 4
Mitu kontsepti on mitte kohustuslikud	# Non-mandatory: 2
Mitu kontsepti on kokku	# Total: 6
Sisend maatriksi ühtede kattuvus	# Coverage: 128/128(100.00000%)
Sisend maatriksi ridade arv	# Number lines: 21
Sisend maatriksi tulpade arv	# Number columns: 9
Keskmine elementide esinemise sagedus kontseptis	# Mean frequency: 0.51587
Keskmine kontsepti ahela sõlme kaal	# Mean monocle: 352.83334
Kontsepti keskmine ühtlus	# Mean object uniformity: 0.77030
Kontsepti keskmine erisus	# Mean separation: 0.42646

Saadud tulemuste seast eemaldati tublikatsed read, kuna kasutatud QC algoritm mingitel juhtudel tekitab tublikatseid kontsepte. Lisaks eemaldati veel kontseptid mille intentsiooni suurus on 1. Tegemist on, kas rolliga millel on ainult 1 õigus, ning see on ekvivalente üksikõiguse omistamisele või rolliga, mis on kasutusel ainult 1 kasutaja poolt, ehk tegemist on kasutajaga kellele on omistatud üksikud õigused. Tegemist on kontekstidega, mis on juba sellisel kujul, mida testitavad lahendus loovad.

4.1.1 AWS andmestiku ettevalmistamine

Rollide moodustamisel oli üks erand juht, milleks oli AWS andmestik. Algne andmestik [13] oli liiga suur testimiseks, 4.8 GB. Selle lihtsustamiseks eemaldati esmalt kõik read milles ei olnud ühtegi „1" väärtus. Selleks kasutati *grep*[4] käsku:

grep "1", amzn-anon-access-samples-2.0.csv > out1.csv

Tulemuseks saadud failis, mis oli 3GB ja ei omanud enam tühiseid ridu. Seejärel transponeeriti csv kasutades tööriista *csvtool*[5] koos *grep* käsuga:

csvtool transpose out1.csv | grep "1", > out2.csv

Teine *grep* koos *csvtool* transponeerimisega eemaldati algsest failist kõik sidumata read ja tulbad ning vähendati suuruse 1.5GB peale. Saadud failist moodustati juhuslik *1024 x 33264 binaarne maatriks. Mõõt 1024 x 33264 tuleneb sellest, et see on libreoffice calc* maksimaalne toetatud suurus. Järgmisena korrati kahte esimest sammu saadud faili puhul ning lõpptulemuseks võeti saadud faili algusest 1023 x 1024 suurusega binaarne maatriks, mis on testimiseks sobilikuma suurusega.

4.2 Rolli stabiilsuse arvutamine

Rollide stabiilsuse arvutamiseks kasutatakse algoritmi, mis on tuletatud kontsepti stabiilsuse algoritmis [6]. Algoritmis leiab esmalt kõik võimalikud alamvalemid iga suurusega 2 ... n-1 ekstentsiooni jaoks ning seejärel testib, kas on võimalik laiendada intentsiooni kasutades valitud ekstentsiooni alamvalimit. Kui maksimaalne intentsioon, mida on võimalik moodustada, ühtib algse intentsiooniga siis loetakse sada alamvalimit kui mitte laiendatavaks. Algoritmi väljundiks on number vahemikus 0 ... 1, mis on konkreetse rolli stabiilsus ja massiiv kõige alamvalimitega, mida ei olnud võimalik laiendada. Tagastatud alavalimite hulk on oluline selleks, et hiljem leida võtme elemendid.

4 https://www.man7.org/linux/man-pages/man1/grep.1.html

5 https://manpages.org/csvtool

4.2.1 Stabiilsuse arvutamise jõudluse parandamine

Suuremate RBAC süsteemide puhul, tekkib stabiilsuse arvutamisel jõudluse probleemid. Alamvalimite hulk kasvas liiga suureks selleks, et kõiki kombinatsioone testida. Üks potentsiaalne lahendus jõudluse probleemile, mida testiti oli paralleelsus ja seda kahel erineval viisil. Esmalt, arvutada iga alamvalim suurusega 2 ... n-1 paralleelselt ja seejärel testida paralleelselt kõiki võimalikke kombinatsioone. Selline lähenemine muutis lahendust leidmise kiiremaks, kuid samas tekitas mälu probleem. Kõiki võimalikke kombinatsioone oli tarvis mälus hoida, mis tähendas seda, et mälu kasutus kasvas ebamõistlikult suureks, ületades 64GB. Teine katse kasutada paralleelsust oli kombineerida alamvalimite leidmine ja stabiilsuse arvutamine. Iga alavalimi suurusega n kombinatsioonide leidmine koos stabiilsuse arvutusega jooksis eraldi protsessina. See lähenemine, ei andnud olulist ajalist võitu kui mälu säästa või kui suurendada paralleelsust, tekkis taas mälu probleem.

Teine lähenemine jõudluse parandamiseks oli hinnata stabiilsust kasutades juhusliku valimit. Alamvalimite puhul mille suurus on N > 10 ei arvutata enam kõik võimalikke kombinatsioone, vaid võetakse juhuslikult 10 * 99999 kombinatsiooni mille põhjal arvutatakse stabiilsus. 10 korda selle pärast, et kombinatsioonide leidmiseks kasutatakse genereerivat funktsiooni, mis saab sisendiks ekstentsiooni ja väljastab kombinatsioone suurusega N. Kuna väljastatavate kombinatsioonide järjekord sõltub sisendist siis sisend ekstentsiooni elementide järjekord muudetakse alati juhuslikult. Lisa jõudlust abistava tegurina on extentsiooni maksimaalseks suuruseks 25. Kui extentsiooni suurus ületab 25 siis võetakse sellest juhuslik valim suurusega 25. Võttes juhuslikult 10*99999 juhuslikku kombinatsiooni suurusega N peaks see andma piisavalt täpse stabiilsuse, et hinnata rolle [16].

Tabel 21. Stabiilsuse arvutamise võrdlus, kõik vs juhuslik valik

Nr.	Ekstentsioon	Stabiilsus, kõigi võimalike puhul	Stabiilsus, pistelise valiku puhul
1	[0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 16, 18, 19, 20]	0.7500629516302565	0.7180167648889052
2	[0, 1, 2, 3, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 20]	0.8749503953112122	0.8618378415253948
3	[0, 1, 2, 3, 4, 5, 6, 7, 8, 11, 12, 13, 14, 16, 19, 20]	0.5001068408681584	0.474519903127878
4	[0, 1, 2, 3, 4, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 19, 20]	0.7502977008335623	0.7704893061252123
5	[0, 1, 2, 3, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 20]	0.7502977008335623	0.7375612605887508

Katsetamisel, mille puhul vähendati N lävend 5 peale ja testitava valiku suurus 10*49999 peale, tulemuseks saadi keskmine delta 0.021. Delta on piisavalt väike, et järjestada rolle stabiilsusel alusel.

4.3 Ekstentsiooni võtme elementide põhine minimeerimise algoritm

Ekstentsiooni võtme elemendil põhinev minimeerimise algoritm leiab kõige madalama stabiilsusega rolli jaoks võtme elemendid. Peale seda kui võtme elemendid on leitud asendatakse algses õiguste maatriksis võtmeelemendi ja intentsiooni ristumise kohad 0 väärtustega.

4.3.1 Võtme elementide leidmine

Võtme elementide leidemiseks kasutati *python*-i implementatsiooni MHS algoritmist nimega *minihit*[6]. Algoritm saab sisendiks kõik rolli stabiilsuse arvutamisel leitud alamvalemid, mille puhul ei olnud võimalik kontsepti laiendada ja tagastab loetelu võimalikest kombinatsioonidest, et katta kõik alamvalemid. Järgmisena leitakse loetelust kõige väiksem kombinatsioon, mis on käeoleva rolli võtme elementideks.

6 https://github.com/TheMatjaz/minihit

4.3.2 Võtme elementide leidmise jõudluse parandamine

Suuremate ekstentsioonide puhul on võimalus, et alamvalimite hulk, mis ei võimalda laiendamist kasvab liiga suureks, rohkem kui 10000. Sellistes olukordades ei suuda MHS implementatsioon mõistliku ajaga tulemusi leida. Jõudluse parandamiseks rakendatakse eel filtreerimist mille loogika on sarnane rolli stabiilsuse arvutmise loogikale. Erinevuseks on asjaolu, et alamvalimid leitakse suureega 1 .. n-1 mitte 2 ... n-1. Lisaks eemaldatkse kõik alamvalimid, mida ei ole võimalik laiendada, algsest ekstentsiooni hulgast enne n-i suurendamist. Tulemuseks saadud kombinatsioon kõik võimalikud kombinatsioonid elementidest, mille eemaldamisel on võimalik rolli intentsiooni laiendada. Peale seda rakendatakse MHS filtreeritud tulemi peal, et leida minimaalne muudatuste arv.

Tabel 22. MHS eel filtreerimise näidis

Piiravad alamvalimid	Eel filtri tulem	MHS tulem
{(16, 17, 18), (17, 18, 14), (18, 17)}	[[17, 18]]	[17], [18]

Eel filtri näitest on näha, et tulemus on [[17, 18]], mis näitab seda, et eemaldada tuleks 17 või 18 indeks ekstentsioonis, mida ka MHS tulem kinnitab. Eel filter vähendab massiivide hulka mida MHS peab töötle, näitas vähenes hulk kolmelt ühele. Selline lähenemine ei anna ajalist võitu väikeste kogude puhul, kuid kui hulk on kümnetes tuhandetes siis juba on.

4.4 Minimeerimine kasutades rolli unikaalseid elemente

Ülekatte põhise minimeerimise algoritmi puhul arvutatakse QC tulemustest saadud intentsiooni binaarse maatriksi ja ekstentsiooni binaarse maatriksi korrutis. Saadud tulemus annab meile kattuvuse maatriksi. Seejärel võetakse kõige madalama stabiilsusega roll ning asendatakse algses õiguste maatriksis nullidega kohad, kus madalaime rolli intentsioon ristub ekstentsiooniga ja kattuvuse maatriksi vastavas kohas on 1.

4.5 Optimeerimise protsessi lõpetamine

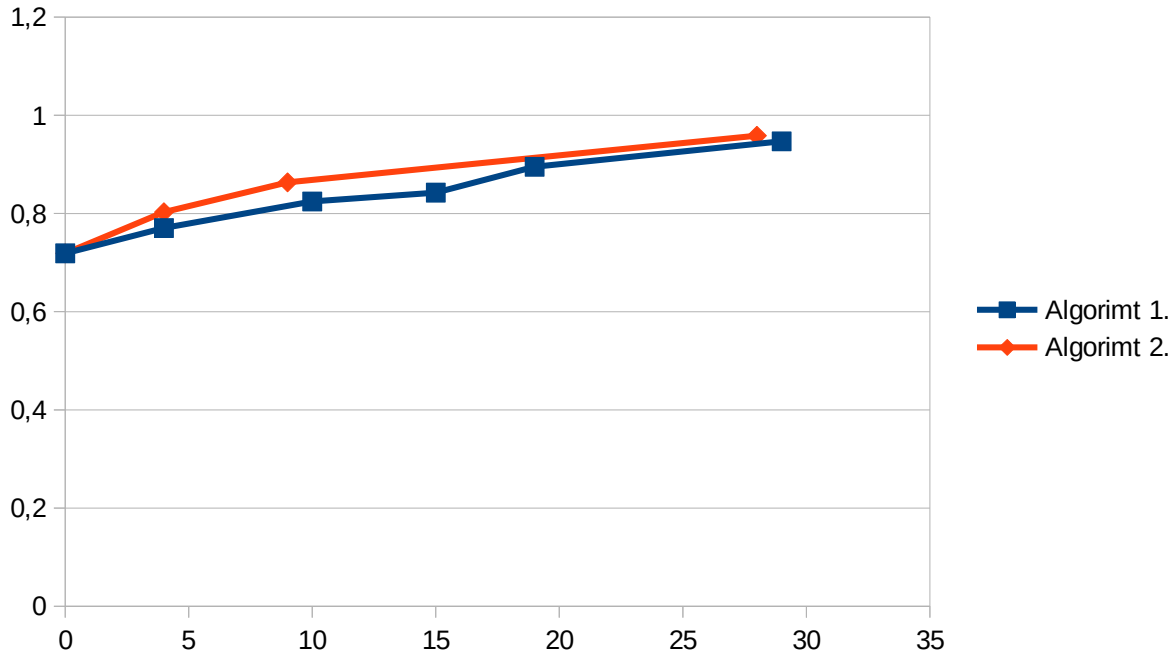
Otsus protsessi lõpetamiseks baseerub stabiilsusel, kuna mõlemad lahendused kasutavad kasutajarolli stabiilsust selleks, et otsustada kuidas muuta algset õiguste maatriksit. Lõpetamiseks on kaks kontrolli. Esimene kontrollib, kas arvutatud rollide keskmine stabiilsus on suurem kui 0.95. Kui süsteem saavutab keskmise stabiilsuse suurema kui 0.95 siis on piisavalt stabiilne, et järgnevad muudatuse ei õigusta ennast enam. Teiseks lõpetamise tingimuseks on olukord kui tsükkel on läbinud vähemalt neli tsüklit ning viimane keskmise stabiilsuse muut on väiksem kui eelmise ja üleeelmises tsüklis.

5 Tulemuste analüüs

Selleks peatükis võrdleme kuidas algoritmide poolt sooritatud muudatused RBAC süsteemi õiguste maatriksis mõjutasid süsteemi keskmist stabiilsust, minimaalset stabiilsust ning rollide koguarvu. Tulemused on kujutatud graafidel mille x-teljel on muudatuste kogu arv ja y-teljel vaadeldava mõõdu muut. Algoritm 1. on võtme elemendil baseeruv lahendus ja algoritm 2. on unikeelsetel elementidel baseeruv. Lisaks vaatame ka algoritmide jooksutamise ajakulu.

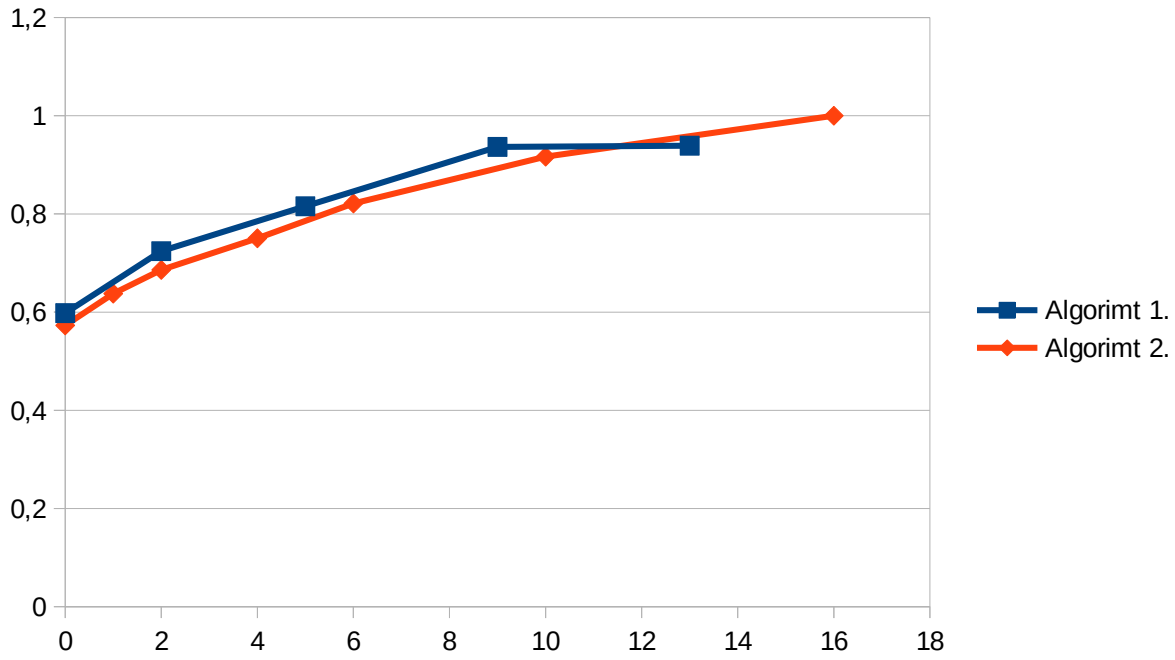
5.1 Rollide keskmise stabiilsuse muut

Võrreldes kuidas mõlemad algoritmid, unikeelsetel elementidel baseeruv ja võtme elemendil baseeruv, muudavad RBAC süsteemi keskmist stabiilsuts, kasutades kõiki neljad andmestiku, saame järgnevad joonised.



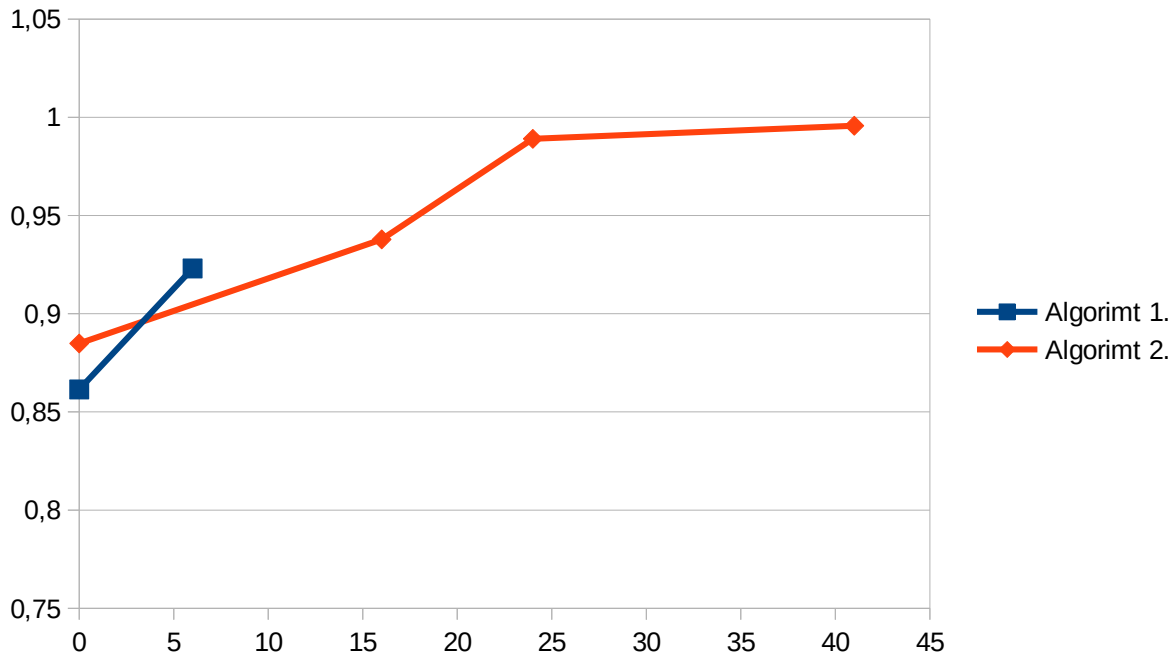
Joonis 2. Sportkeskuse andmestikus rollide keskmise stabiilsuse muut

Spordikeskuse andmestiku puhul jõudsid mõlemad algoritmid väga sarnase keskmise rollide stabiilsuse juurde muudatuste arvuga, mis on peaaegu võrdsed. Esimese algoritmi kasutamisel saadi lõpuks keskmine stabiilsuse väärtuseks 0.9469434 ning selleks oli tarvis sooritada 29 muudatust. Teise algoritmiga saavutati keskmine stabiilsuse väärtuseks 0.9582865 ja seda 28 muudatusega. Mis paistab silma on asjaolu, et keskmise stabiilsuse kasv toimub esimese algoritmi puhul ühtlasemalt, teise algoritmi puhul on näha suurt muudatuste arvu kasv liikudes kolmandalt iteratsioonilt neljandale.



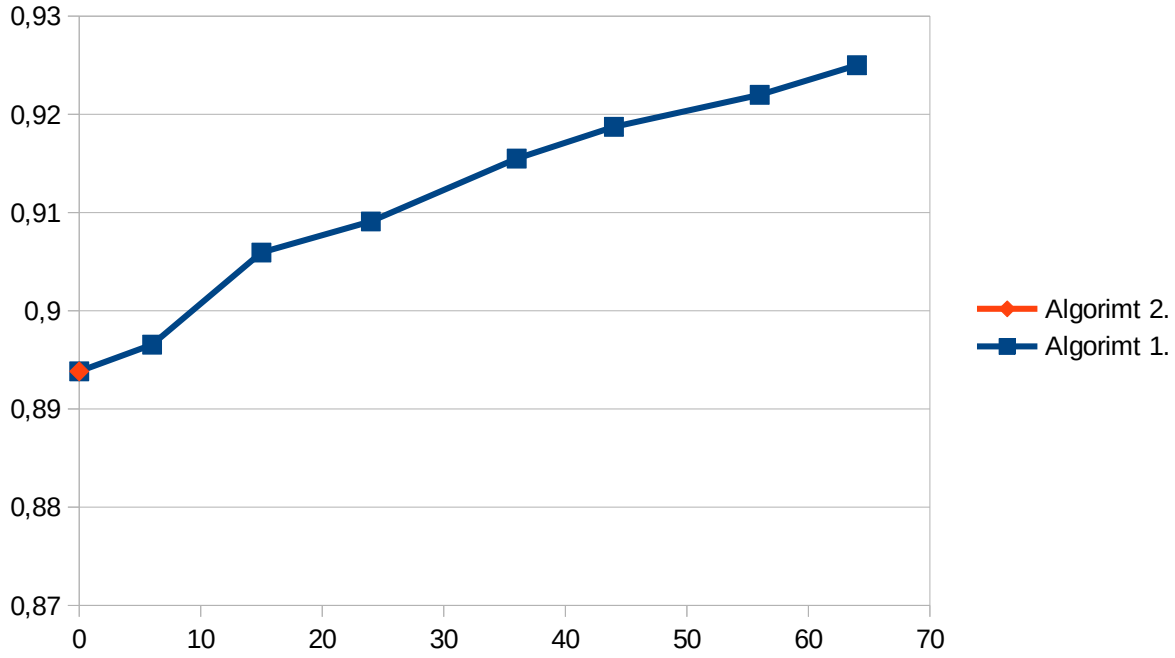
Joonis 3. Intraneti andmestikus rollide keskmise stabiilsuse muut

Intraneti andmestiku puhul ilmnes sama, mis paistis välja ka spordikeskuse andmestiku puhul. Mõlemad algoritmid saavutasid väga sarnase keskmise stabiilsuse minimaalse erinevusega muudatuste arvus. Selle andmestiku puhul on muutuste arvu ja stabiilsuse kasv ühtlasem, aga kuna unikeelsetel elementidel baseeruva algoritmi keskmise stabiilsuse kasv on stabiilsem sooritab see rohkem iteratsiooni ning lõpptulemuseks saavutas keskmiseks stabiilsuseks väärtuse 1.



Joonis 4. CMS andmestikus rollide keskmise stabiilsuse muut

CMS andmestiku puhul tekkis võtme elemendil baseeruvat algoritmi kasutades nii suur hulk alamvalimeid, mis takistasid laiendamist, et algoritm ei suutnud peale nelja tundi teist tsüklit lõpetada ning selles kohas otsustati test pooleli jätta. Üks muudatus mille võtme elemendil baseeruv algoritm sooritas suutis tõstis kogu süsteemi keskmist stabiilsust oluliselt paremini muudatuste arvu kohta kui unikeelsetel elementidel baseeruv, kuid lõpp tulemus mille unikeelsetel elementidel baseeruv algoritm saavutas oli parem ning unikeelsetel elementidel baseeruv algoritm suutis iseseisvalt lõpuni jõuda.

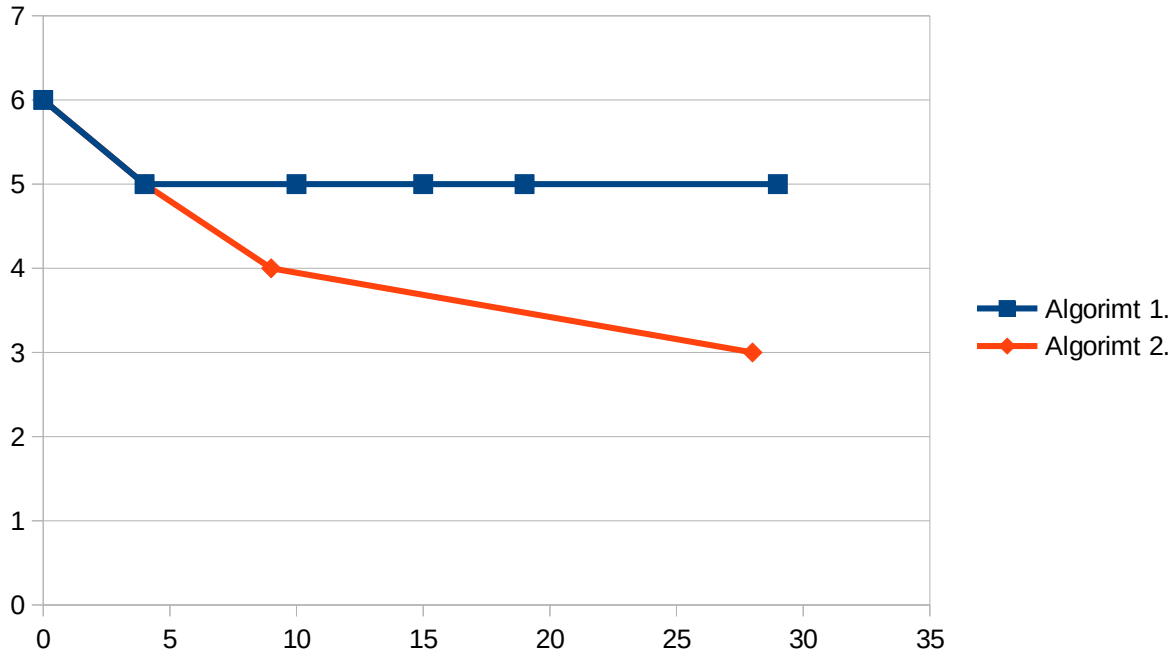


Joonis 5. AWS andmestikus rollide keskmise stabiilsuse muut

AWS andmestiku puhul tekkis probleem unikeelsetel elementidel baseeruva algoritmiga. Kuna kõige madalama stabiilsusega rollil puudus katte maatriksis unikaalne ala, ei sooritanud algoritm muudatusi ning keskmine stabiilsus ei muutunud. Vaadates võtme elemendil baseeruva algoritmi tulemust, ei ole selle puhul ka muut väga silmapaistev. Keskmine stabiilsus kasvas ainult ~0.03 võrra ja seda 64 muudatusega.

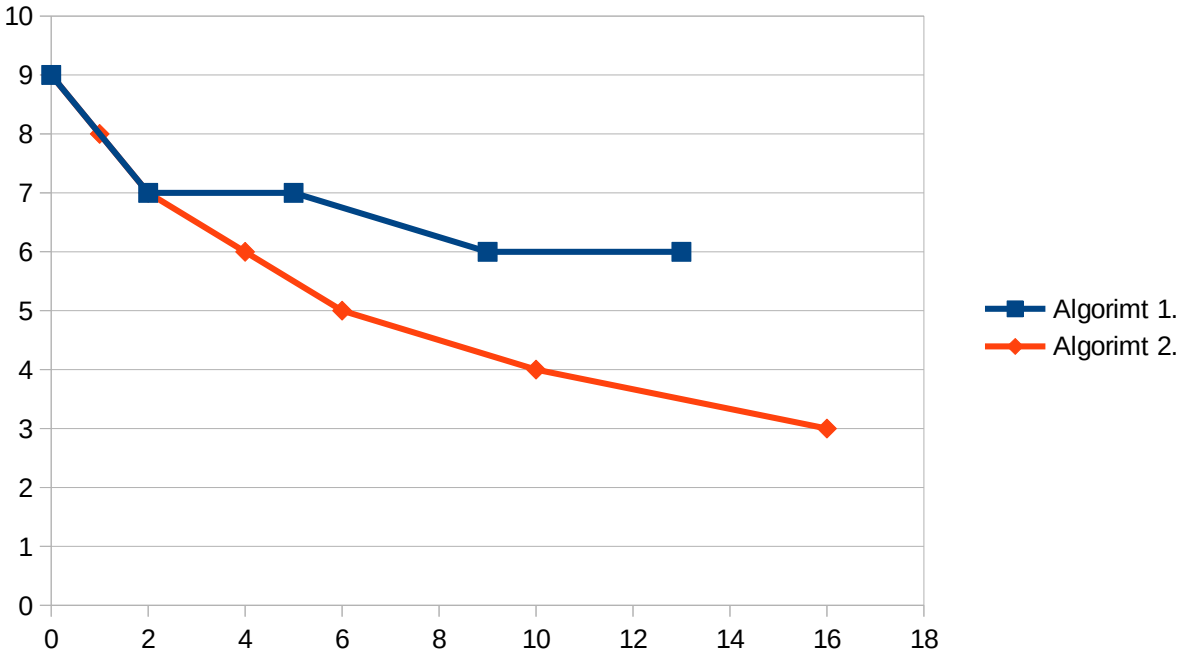
5.2 Rollide koguarvu muut

Vaatame kuidas mõlemad algoritmid muudavad RBAC süsteemi rollide koguarvu. Tulemused kõigi neljad andmestiku jaoks on järgnevatel joonistel.



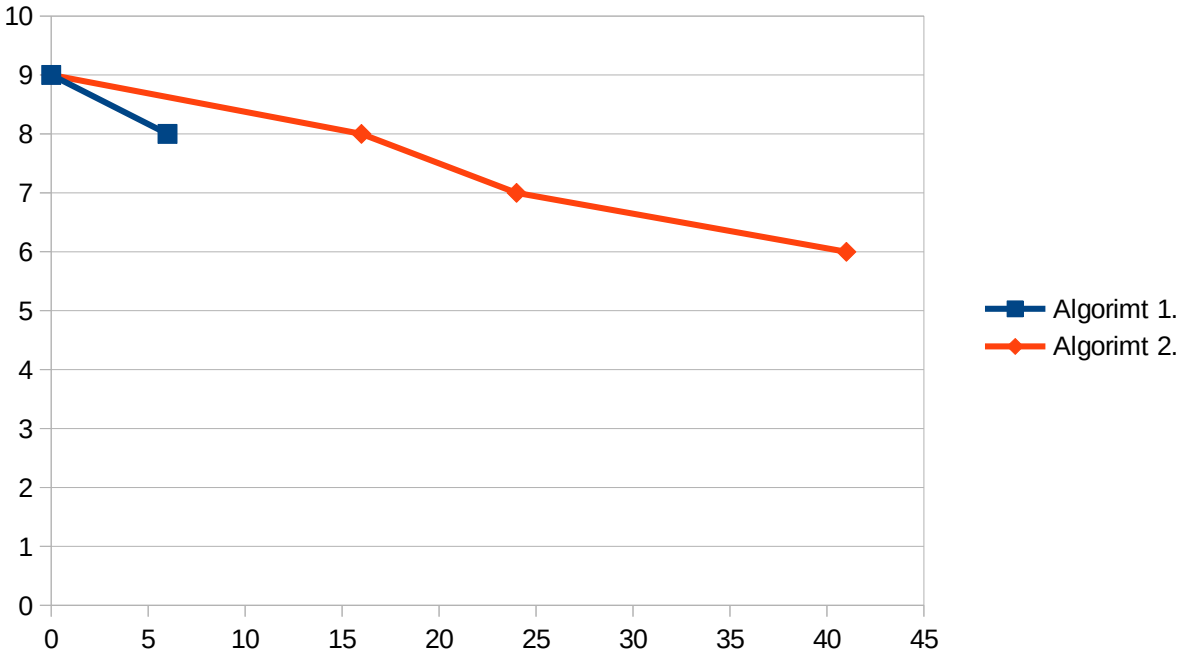
Joonis 6. Sportkeskuse andmestikus rollide koguarvu muut

Rollide koguarv minimeerimise puhul tuleb lahenduste põhimõtteline erinevus välja. Algoritm 2., mis kasutab unikeelseid elemente, väheneb igal iteratsioonil rollide koguarv ühe võrra. Võtme elemendil baseeruv algoritm eemaldab ainult võtme elemendid ning see järel arvutakse kõik rollid uuesti. Tänu sellele võib tekkida sama arv rolle, väiksem arv rolle või isegi suurem hulk kui oli algselt. Teine algoritm eemaldab lihtsalt kõige väiksema stabiilsusega rolli ilma teisi rolle muutmata.



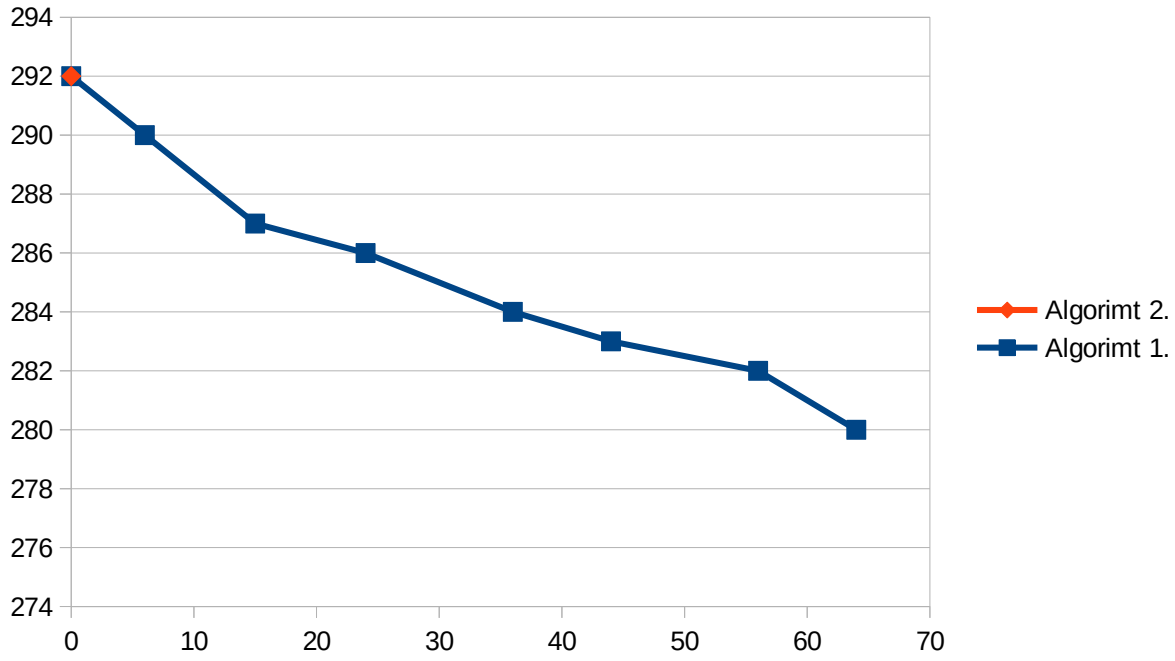
Joonis 7. Intraneti andmestikus rollide koguarvu muut

Intraneti andmestiku puhul on avaneb sama pilt nagu seda oli ka spordikeskuse puhul. Algoritm 2., mis on unikeelsetel elementidel baseeruv eemaldab igal iteratsioonil kõige madalama stabiilsusega rolli ning võtme elemendil baseeruv eemaldab võtme elemendid ning arvutab kõik rollid uuesti, mis võib kaasa tuua rollide arvu mitte muutumise.



Joonis 8. CMS andmestikus rollide koguarvu muut

Nagu juba eelnevalt mainiti siis ei suutnud võtme elemendil baseeruv algoritm oma tööd lõpetada mõistliku ajaga. Ning samamoodi nagu seda oli näha keskmise stabiilsuse puhul, käitus võtme elemendil baseeruv algoritm ka rollide koguarvu muutu juures. Algoritm suutis sooritada sama rollide arvu muudatuse, liikudes üheksalt kaheksale, mida ka unikeelsetel elementidel baseeruv, kuid tegi seda väiksema arvu muudatustega. Nagu ka eelnevate andmestike puhul, eemaldab unikeelsetel elementidel baseeruv igal iteratsioonil ühe rolli.



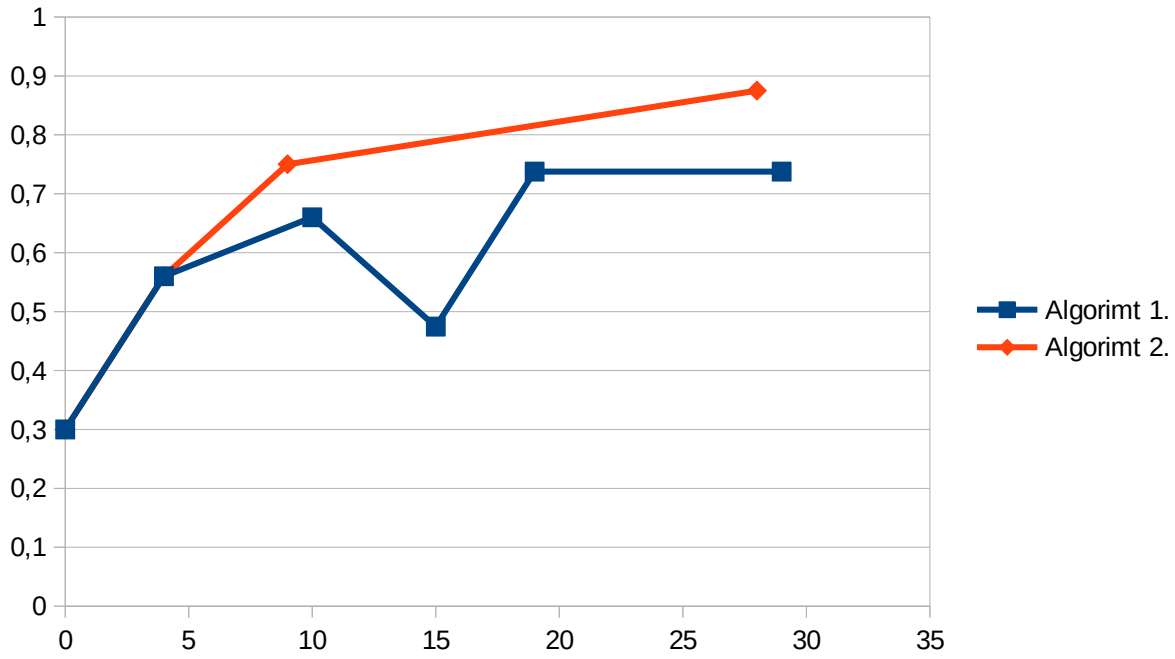
Joonis 9. AWS andmestikus rollide koguarvu muut

AWS andmestiku vähendas võtme elemendil baseeruv algoritm rolle 12 võrra sooritades 64 muudatust ning maatriksi korrutisel baseeruv ei suutnud sooritada ühtki muudatust.

Asjaolu, et ühe rolli eemaldamiseks on tarvis lisada rohkem kui 2 üksikõigust on viide sellele, et minimeerimise algoritmid ei ole äriliselt mõistlikud, kuna selle asemel, et süsteem muutuks hallatavamaks muutub haldamine hoopis keerulisemaks. Iga eemaldatud roll tähendab enamasti nelja või enama üksikõiguse lisamist kasutajale, mida peab haldama. Ainukene mõistlik andmestik, kus algoritme rakendada on intranet. Intranet puhul toimub iteratsioonis ühe rolli asendus ühe üksikõigusega kasutas unikeelsetel elementidel baseeruvat algoritmi ning ühe rolli asendamine kahe õigusega kasutades võtme elemendil baseeruvat algoritmi.

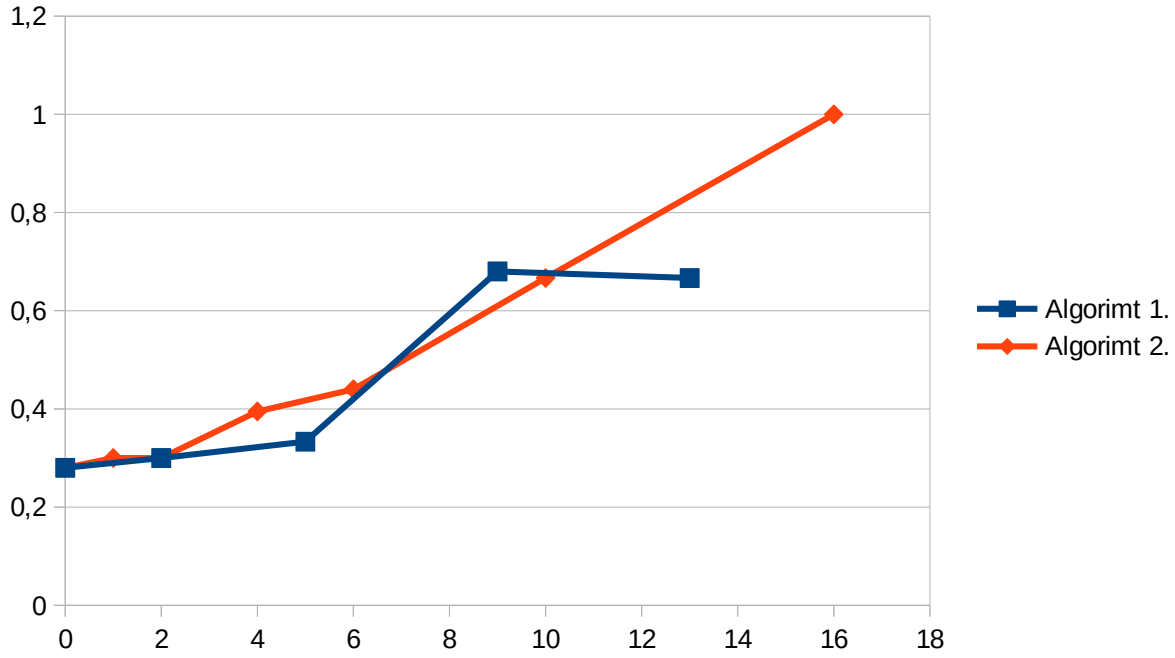
5.3 Rollide minimaalse stabiilsuse muut

Järgnevatel joonistel on näha kuidas mõlemad algoritmid muutsid RBAC süsteemi kasutajarollide minimaalset stabiilsust.



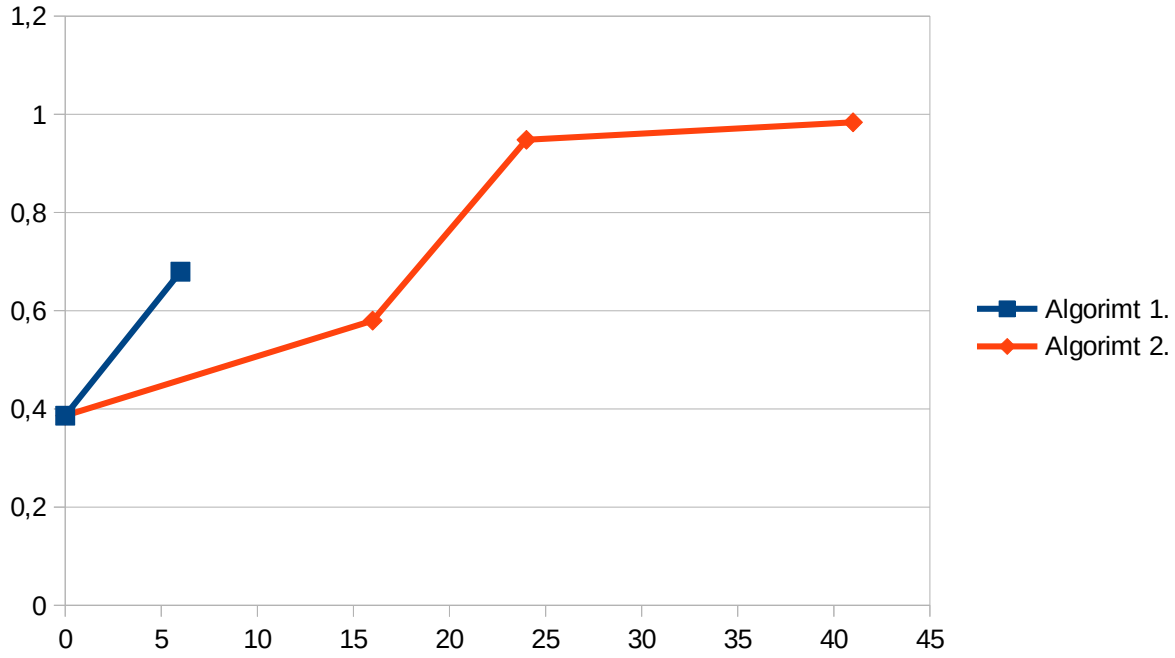
Joonis 10. Sportkeskuse andmestikus rollide minimaalse stabiilsuse muut

Võtme elemendil baseeruva algoritmi puhul ilmneb minimaalse stabiilsuse muudu puhul sama asi, mis ilmnes eelnevalt rollide koguarvu muudu puhul. Kuna algoritm arvutab kõik rollide uuesti siis ei ole tagatud asjaolu, et peale igat muudatust minimaalne stabiilsus kasvab. Kuna unikeelsetel elementidel baseeruv algoritm eemaldab alti minimaalse stabiilsusega rolli siis on alati tagatud, et minimaalne stabiilsus kasvab.



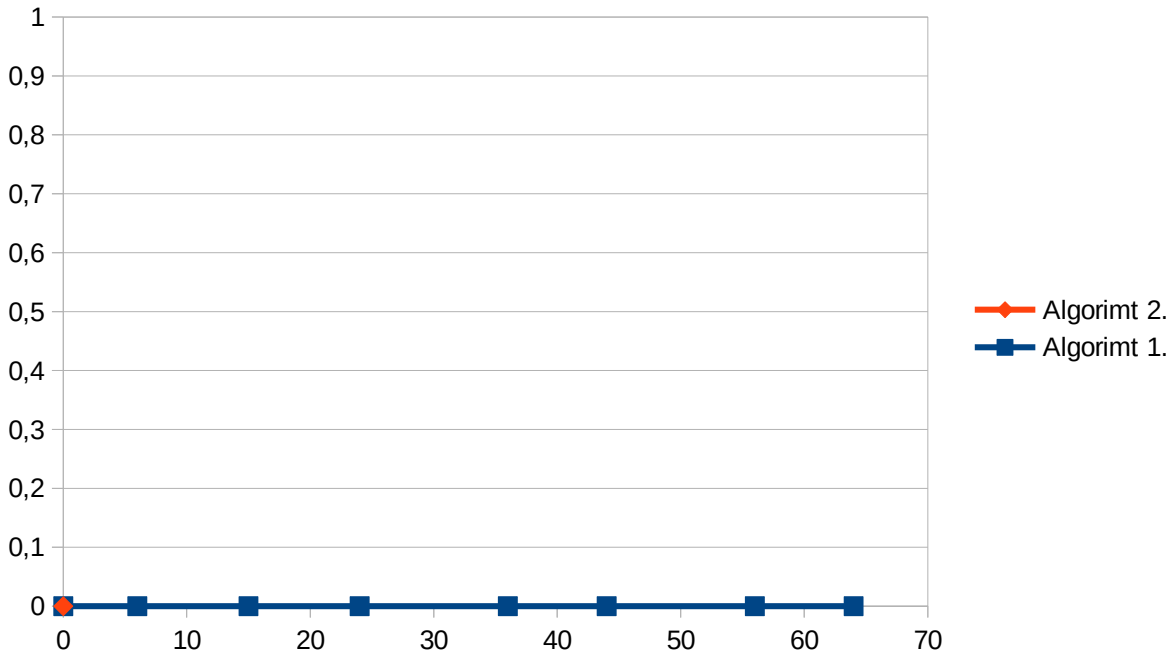
Joonis 11. Intraneti andmestikus rollide minimaalse stabiilsuse muut

Intraneti andmestiku puhul on näha, mida tähendab võtme elemendil baseeruva algoritmi puhul rollide ümber arvutamine. Peale üht muudatust võib minimaalne stabiilsus hüppeliselt kasvada ning peale järgmist muudatust uuesti langeda.



Joonis 12. CMS andmestikus rollide minimaalse stabiilsuse muut

CMS andmestiku puhul käitub rollide minimaalse stabiilsus muut samamoodi nagu kõigi teiste mõõtude puhul. Võtme elemendil baseeruv algoritm teeb hüppelise muutuse suhteliselt väheste muudatustega, aga kuna algoritm ei viinud oma tööd lõpuni ei saa selle põhjal järeldusi teha. Unikeelsetel elementidel baseeruv algoritm käitus nagu eeldatud, igas iteratsioonis kasvas minimaalne stabiilsus.



Joonis 13. AWS andmestikus rollide minimaalse stabiilsuse muut

AWS andmestiku puhul on näha kuidas muutub süsteemi rollide minimaalne stabiilsus suurte süsteemide puhul. Kuna selle süsteemis on rollide arv nii suur, et keskmise stabiilsuse kasv suutis enne pidurduda kui kõik 0 stabiilsusega rollide said eemaldatud, võtme elemendil baseeruva algoritmi puhul. Unikeelsetel elementidel baseeruv algoritm ei suutnud sooritada ühtki muudatust AWS andmestiku puhul eelnevalt mainitud põhjusel.

5.4 Algoritmide ajakulu

Kõik katsetused jooksutati kasutades arvutit milles on AMD Ryzen 9 5900X ja 64 GB DDR4 mälu. Algoritmide testimisel tuli välja asjaolu, et väiksemate andmestike puhul on võtme elementidel baseeruv algoritm kiirem kui unikeelsetel elementidel baseeruv algoritm. Mõlema algoritmi puhul on kasutuses sama meetod stabiilsuse arvutamiseks, sellest erinevust tekkida ei saa. Kus tuleb erinevus sisse on kriteerium mille järgi otsustatakse kuidas muuta õiguste maatriksit. Võtme elementidel baseeruva algoritmi puhul on selleks võtme elementide leidmine, mis on väikse hulga puhul kiire, kuid ajakulu kasvab kiirelt kuni selle punktini, et algoritm ei suuda oma tööd lõpetada. Unikeelsetel elementidel baseeruva algoritmi puhul on selleks maatriks korrutis, et leida unikaalsed elemendid. Maatriks korrutuse ajakulu kasv on lineaarse.

Tabel 23. Algoritmide jooksutamise ajakulu sekundites

Andmekogu	**Võtme elemendil baseeruv algoritm**	**Maatriksite korrutisel baseeruv algoritm**
Spordi keskus	21.474484	40.12545
Intranet	0.37428	0.56660
CMS	Ei lõpetanud	893.19321
AWS	164.79386	94.452808

6 Kokkuvõte

Käesolevas lõputöös defineeriti, mis on kasutajarolli stabiilsus RBAC süsteemis ning kasutati defineeritud stabiilsust selleks, et moodustada kaks algoritmi RBAC süsteemi rollide minimeerimiseks. Üks defineeritud algoritmidest kasutas rolli stabiilsust ning ekstentsiooni alavalmite hulka selleks, et otsustada kuidas muuta süsteemi õiguste maatriksit. Teine kasutas rolli stabiilsust ning rolli unikaalset osa, et otsustada kuidas muuta süsteemi õiguste maatriksit. Seejärel implementeeriti algoritmid ning testiti nende efektiivsust kasutades nelja erinevat andmestikku: spordikeskus, intranet, CMS ja AWS. Efektiivsuse hindamiseks kasutati nelja mõõtu: keskmine süsteemi stabiilsus, minimaalne rolli stabiilsus süsteemis, süsteemi rollide koguarv ning sooritatud muudatuste arv algses õiguste maatriksis. Eesmärgiks oli defineerida RBAC süsteemi kasutaja rollide stabiilsus sellist, et see võimaldaks öelda iga kasutajarolli kohta kui oluline see on RBAC süsteemi jaoks ning seda sama näitajat kasutada töökäigus moodustatud algoritmides, et otsustada millist rolli eemaldada rollide minimeerimiseks.

Töökäigus defineeritud stabiilsus, mis ütleb, et kasutajarolli stabiilsus on suhtearv ekstentsiooni alamvalimite vahel, mille puhul ei ole võimalik intentsiooni laiendada ja kõigi võimalike ekstentsiooni alamvalimite vahel. Loodud definitsioon võimaldab leida rolle RBAC süsteemis, mis kaotaksid väärtuse kui rollist eemaldatakse võtme õigus või teatud kasutajalt võetakse roll. Ning algoritmide implementatsioonid, mis loodi kasutades defineeritud rolli stabiilsust vähendasid rollide arvu süsteemis ning suurendasid süsteemi rollide keskmist stabiilsust, kuid neid ei saa edukaks lugeda.

Töökäigus loodud rollide minimeerimise algoritm, mis kastab lisaks rollide stabiilsusele veel võtme elementi ei olnud võimeline toime tuleme keerulisemate RBAC süsteemidega, kuna võtme elementide leidmise lahenduse jõudlus ei olnud piisavalt hea. Isegi kui jõudlus oleks olnud parem oli efektiivsuse analüüsis näha, et muudatuste arv, mis tuleks sooritada süsteemi õiguste maatriksis ühe rolli eemaldamiseks ei õigusta

algoritmi kasutamist süsteemis. Implementeeritud minimeerimise algoritm, mis kasutas rollide unikaalset osaks lisaks stabiilsusele oli jõudluse poolest parem, eriti suuremate süsteemida puhul, kuid sellel ilmnes üks suure probleem. Algoritm ei ole võimeline sooritama minimeerimist kui kõige madalama stabiilsusega rollil puudub unikaalne osa.

Algoritmi omavahel võrdlemisel ilmnes, et rollide unikaalsel osal baseeruv algoritm andis enamustel juhtudel parema tulemuse kuid kumbki algoritm ei ole sobilik süsteemi rollide minimeerimiseks selliselt, et süsteem muutuks hallatavaks. Muudatuste arv, mida oli tarvis teha rollide eemaldamiseks ei õigusta kumbagi algoritmi kasutamist.

6.1 Võimalikud edasiarendused

Kuigi RBAC süsteemi kasutajarollide stabiilsuse definitsioonil baseeruvad algoritmid ei õigustanud enda kasutamist, tegi stabiilsus ise seda mille jaoks see defineeriti. Stabiilsus võimaldas leida rolle, mis on süsteemi jaoks vähem olulised ning teisi stabiilsuse potentsiaalseid kasutusviise võiks edasi uurida.

Teine teema, mida võiks edasi uurida, on rollide minimeerimine kasutades unikeelsetel elementidel baseeruvat algoritmi. Kuid mõõduks, mille alusel otsustada millist rolli eemaldada, kasutada mitte stabiilsust, vaid minimaalset muudatuste arvu. Nagu sai kolmandas peatükis uuritud, ei ole stabiilsuse ja muudatuste arvu vahel tugevat korrelatsiooni. Kuidas käituks see algoritm kui kasutada stabiilsuse asemel muudatuste arvu valik kriteeriumina.

Kasutatud kirjandus

[1] Ferraiolo, D., Kuhn, Role-Based Access Controls, Proceedings of the 15th National Computer Security Conference, pp. 554-563

[2] Ferraiolo, D., Kuhn, D., Chandramouli, R., Role-Based Access Control, Second Edition. ARTECH HOUSE, INC (2007).

[3] Priss, U. 2004. Formal Concept Analysis in Information Science [15.09.2023]

[4] Ganter, B., Wille, R., Formal Concept Analysis: Mathematical Foundations. Springer (1999).

[5] Torim, A. 2012. A Visual Model of the CRUD Matrix [15.09.2023]

[6] Kuznetsov, S., O., On stability of a formal concept, Ann Math Artif Intell 49, pp. 101–115

[7] A. Albano, B. Chornomaz, Why concept lattices are large extremal theory for the number of minimal generators and formal concepts, Proceedings of the 12th International Conference on Concept Lattices and Their Applications (CLA'2015), Clermont-Ferrand, France (2015), pp. 73-86

[8] Amira Mouakher, Sadok Ben Yahia, QualityCover: Efficient binary relation coverage guided by induced knowledge quality, Information Sciences, Volumes 355–356, 2016, pp. 58-73

[9] Ganter, B., Kuznetsov, S.O., 2000, Formalizing hypotheses with concepts. In: Mineau, G., Ganter, B. (eds.) Proc. 8th International Conference on Conceptual Structures, ICCS'00. Lecture Notes in Artificial Intelligence, vol. 1867, pp. 342–356

[10] H. Lu, J. Vaidya and V. Atluri, "Optimal Boolean Matrix Decomposition: Application to Role Engineering," *2008 IEEE 24th International Conference on Data Engineering*, Cancun, Mexico, 2008, pp. 297-306, doi: 10.1109/ICDE.2008.4497438.

[11] Aleksey Buzmakov, Sergei O. Kuznetsov, Amedeo Napoli, Is Concept Stability a Measure for Pattern Selection?, Procedia Computer Science, Volume 31, 2014, pp 918-927

[12] T. Õunase, (2021), Rollikaevandamise meetodite võrdlus kolme organisatsiooni andmete põhjal, [Võrgumaterjal] https://digikogu.taltech.ee/et/Item/e3cd756d-dd05-4a9d-ae4e-9e55da4fd187 [Kasutatud 11.09.2023]

[13] Montanez, Ken. (2011) Amazon Access Samples. UCI Machine Learning Repository. [Võrgumaterjal] https://doi.org/10.24432/C5JW2K. [Kasutatud 12.10.2023]

[14] Mézard M.,Tarzia M. (2007), Statistical mechanics of the hitting set problem, Phys. Rev. E 76, 041124

[15] R. Reiter, A theory of diagnosis from first principles, Artificial Intelligence 32 (1987) 57-95

[16] Mohamed-Hamza Ibrahim, Rokia Missaoui, (2020) Approximating concept stability using variance reduction techniques, Discrete Applied Mathematics, Volume 273, pp 117-135

[17] Data breach put 360,000 Pa. teachers, education department staffers' personal information at risk [Võrgumaterjal] https://www.pennlive.com/politics/2018/03/data_breach_put_360000_pa_teac.html [Kasutatud 26.12.2023]

[18] Roles terms and concepts [Võrgumaterjal] https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_terms-and-concepts.html [Kasutatud 22.12.2023]

[19] Ravi S. Sandhu, Edward J. Coyne, Hal L. Feinstein, and Charles E. Youman, "Role- Based Access Control," IEEE Computer, 2912, February 1996, 38-47

[20] Zhang, D., Ramamohanarao, K., Ebringer, T. Juuni 2007. Role Engineering using Graph Optimisation [Võrgumaterjal] https://dl.acm.org/doi/abs/10.1145/1266840.1266862 [Kasutatud 22.10.2023]

[21] R. Greiner, B. A. Smith and R. W. Wilkerson, A Correction to the Algorithm in Reiter's Theory of Diagnosis, Artificial Intelligence 41 (1989/90) 79-88

Lisa 1– Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks[7]

Mina, Mart Hütt

1 Annan Tallinna Tehnikaülikoolile tasuta loa (lihtlitsentsi) enda loodud teose „Kahe, kontsepti stabiilsusel baseeruvate, süsteemi rollide minimeerimise algoritmide testimine nelja erineva andmestiku peal" mille juhendaja on Ants Torim

1.1 reprodutseerimiseks lõputöö säilitamise ja elektroonse avaldamise eesmärgil, sh Tallinna Tehnikaülikooli raamatukogu digikogusse lisamise eesmärgil, kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

1.2 üldsusele kättesaadavaks tegemiseks Tallinna Tehnikaülikooli veebikeskkonna kaudu, sealhulgas Tallinna Tehnikaülikooli raamatukogu digikogu kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2 Olen teadlik, et käesoleva lihtlitsentsi punktis 1 nimetatud õigused jäävad alles ka autorile.

3 Kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest ning muudest õigusaktidest tulenevaid õigusi.

05.01.2024

7 Lihtlitsents ei kehti juurdepääsupiirangu kehtivuse ajal vastavalt üliõpilase taotlusele lõputööle juurdepääsupiirangu kehtestamiseks, mis on allkirjastatud teaduskonna dekaani poolt, välja arvatud ülikooli õigus lõputööd reprodutseerida üksnes säilitamise eesmärgil. Kui lõputöö on loonud kaks või enam isikut oma ühise loomingulise tegevusega ning lõputöö kaas- või ühisautor(id) ei ole andnud lõputööd kaitsvale üliõpilasele kindlaksmääratud tähtajaks nõusolekut lõputöö reprodutseerimiseks ja avalikustamiseks vastavalt lihtlitsentsi punktidele 1.1. ja 1.2, siis lihtlitsents nimetatud tähtaja jooksul ei kehti.

Lisa 2 – Intraneti andmestiku binaarne õiguste maatriks

0	0	0	0	1	0	0	1	1	0	0	1	1	1	0	0	0	1	0
0	0	1	0	0	0	0	1	0	0	1	0	0	0	0	1	1	1	0
0	0	1	0	1	0	1	0	0	0	1	1	0	0	0	1	1	0	0
1	1	1	1	0	1	0	1	0	1	1	1	0	0	1	1	1	0	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0
1	1	1	1	0	1	0	1	0	1	1	0	1	0	1	1	0	1	1
0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	1	0	1	0
0	0	1	0	0	0	0	1	1	0	1	0	1	0	0	1	0	1	0
0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0

Tabel 24. Intraneti andmestiku binaarne õiguste maatriks

Lisa 3 – CMS andmestiku binaarne õiguste maatriks

1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	1	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	1	1	1	0	0	1	0	0	1	0	1	0
1	1	1	0	1	1	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1
1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	1	0	1	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	1	1	0	0	0	1	0	0	1	0	0	1	0	1	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
1	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1
1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	0	1	1	1	0
1	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0
1	0	0	0	1	1	1	0	0	0	0	0	1	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0

1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	0	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	0	0	0	1	0	0	0	0	0	1	1	1	0
1	0	0	0	0	0	1	1	1	1	0	0	1	0	1	0
1	0	0	0	0	0	1	1	1	1	0	0	1	0	1	0
1	0	0	1	0	0	1	1	1	1	1	0	1	0	1	0
1	0	0	1	0	0	1	1	1	1	1	0	1	0	1	0
1	0	0	1	0	0	1	1	1	1	1	0	1	0	1	0
1	0	0	1	0	0	1	1	1	1	1	0	1	0	1	0
1	0	0	1	0	0	1	1	1	1	1	0	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0
1	0	0	1	0	0	1	0	0	0	0	1	1	0	1	0
0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	1
0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0
0	0	0	0	0	0	1	0	0	0	1	0	1	0	1	0
1	0	0	1	0	0	1	1	1	0	0	0	1	0	1	1

Tabel 25. CMS andmestiku binaarne õiguste maatriks

Lisa 4 – Spordikeskuse andmestiku võrdluse andmed

Võtme elemendil baseeruv algoritm	Unikeelsetel elementidel baseeruv algoritm
Keskmine stabiilsus	
0,72	0,72
0,77	0,8
0,82	0,86
0,84	0,96
0,9	0,96
0,95	0,96
Minimaalne stabiilsus	
0,3	0,3
0,56	0,56
0,66	0,750086
0,47	0,88
0,74	0,88
0,74	0,88
Rollide koguarv	
6	6
5	5
5	4
5	3
5	3
5	3
Muudatuste arv	
0	0
4	4
10	9
15	28
19	28
29	28

Tabel 26. Spordikeskuse andmestiku võrdluse tulemused

Lisa 5 – Intraneti andmestiku võrdluse andmed

Võtme elemendil baseeruv algoritm	Unikeelsetel elementidel baseeruv algoritm
Keskmine stabiilsus	
0,6	0,572977
0,72	0,64
0,82	0,69
0,94	0,75
0,94	0,82
	0,92
	1
Minimaalne stabiilsus	
0,28	0,28
0,3	0,3
0,33	0,3
0,68	0,395
0,67	0,44
	0,67
	1,000
Rollide koguarv	
9	9
7	8
7	7
6	6
6	5
	4
	3
Muudatuste arv	
0	0
2	1
5	2
9	4
13	6
	10
	16

Tabel 27. Intraneti andmestiku võrdluse tulemused

Lisa 6 – CMS andmestiku võrdluse andmed

Võtme elemendil baseeruv algoritm	Unikeelsetel elementidel baseeruv algoritm
Keskmine stabiilsus	
0,86	0,88
0,92	0,94
	0,99
	1
Minimaalne stabiilsus	
0,39	0,39
0,68	0,58
	0,95
	0,98
Rollide koguarv	
9	9
8	8
	7
	6
Muudatuste arv	
0	0
6	16
	24
	41

Tabel 28. CMS andmestiku võrdluse tulemused

Lisa 7 – AWS andmestiku võrdluse andmed

Võtme elemendil baseeruv algoritm	Unikeelsetel elementidel baseeruv algoritm
Keskmine stabiilsus	
0,89	0,890000
0,9	
0,91	
0,91	
0,92	
0,92	
0,92	
0,93	
Minimaalne stabiilsus	
0	0
0	
0	
0	
0	
0	
0	
0	
Rollide koguarv	
292	292
290	
287	
286	
284	
283	
282	
280	
Muudatuste arv	
0	0
6	
15	
24	
36	
44	
56	
64	

Tabel 29. AWS andmestiku võrdluse tulemused