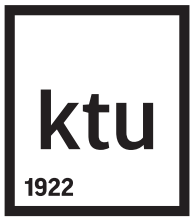


Kauno technologijos universitetas
Informatikos fakultetas

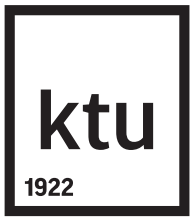
Informacinių technologijų saugos politikos valdymo automatizavimo modelis

Baigiamasis magistro studijų projektas

Viltė Liutkevičiūtė
Projekto autorė

prof. Algimantas Venčkauskas
Vadovas

Kaunas, 2025



Kauno technologijos universitetas
Informatikos fakultetas

Informacinių technologijų saugos politikos valdymo automatizavimo modelis

Baigiamasis magistro studijų projektas
Informacijos ir informacinių technologijų sauga (6211BX008)

Viltė Liutkevičiūtė
Projekto autorė

prof. Algimantas Venčkauskas
Vadovas

prof. Jevgenijus Toldinas
Recenzentas

Kaunas, 2025

Kauno technologijos universitetas
Informatikos fakultetas
Viltė Liutkevičiūtė

Informacinių technologijų saugos politikos valdymo automatizavimo modelis

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;
2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;
3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);
4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Viltė Liutkevičiūtė
Patvirtinta elektroniniu būdu

Viltė Liutkevičiūtė. Informacinių technologijų saugos politikos valdymo automatizavimo modelis. Baigiamasis magistro studijų projektas/ vadovo prof. Algimantas Venčkauskas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Technologijos mokslų studijų sritis.

Reikšminiai žodžiai: IT sauga, Informacijos saugumas, Automatizavimas, IT saugos politika, NLP, XML, „Bash“, Kibernetinė sauga, Komandinis skriptavimas, Politika kaip kodas, Politikos dokumentų interpretavimas.

Kaunas, 2025, 84 p.

Santrauka

Šio magistrinio darbo tikslas – sukurti ir ištirti automatizuotą IT saugos politikos valdymo modelį, tinkantį naudoti šiuolaikinėse organizacijose. Darbe analizuojami esami saugos modeliai ir jų trūkumai, pabrėžiama žmogiškųjų klaidų rizika bei poreikis mažinti rankinį darbą IT politikų įgyvendinimo procese. Tyrimo metu buvo sukurtas modelis, integruojantis natūralios kalbos apdorojimo (NLP) technologiją ir „politika kaip kodas“ (angl. „Policy-as-Code“) principą, siekiant automatiškai interpretuoti organizacijos dokumentuose aprašytas IT politikas ir jas efektyviai įgyvendinti.

Eksperimentinėje darbo dalyje sukurtas ir praktiškai išbandytas modelio prototipas. Rezultatai parodė, kad automatizuotas sprendimas ženkliai pagerino informacijos saugumo lygį, sumažino žmogiškųjų klaidų tikimybę ir užtikrino organizacijos politikos atitiktį su infrastruktūros saugumo nustatymais. IT saugos politikos valdymas tapo automatizuotas, todėl pastebėtas pagerėjęs saugumo valdymo proceso skaidrumas ir efektyvesnis resursų panaudojimas. Tokie rezultatai rodo, kad siūlomas modelis yra pritaikomas praktikoje ir naudingas įmonėms, siekiančioms užtikrinti politikų atitiktį ir didinti savo IT saugumo lygį.

Viltė Liutkevičiūtė. Model for Automated Information Technology Security Policy Management). Master's Final Degree Project/ supervisor prof. Algimantas Venčkauskas; Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Study Field of Technology.

Keywords: IT security, Information security, Automation, IT security policy, NLP, XML, Bash, Cybersecurity, Command scripting, Policy-as-Code, Policy document interpretation.

Kaunas, 2025. 84.

Summary

The objective of this Master's thesis is to develop and investigate an automated IT security policy management model suitable for use in modern organizations. The thesis analyzes existing security models and identifies their shortcomings, emphasizing the risks of human errors and the necessity of reducing manual work in the implementation of IT policies. A new model integrating Natural Language Processing (NLP) technology and the "Policy-as-Code" principle was developed to automatically interpret IT security policies documented within organizations and effectively deploy them.

In the experimental part of the research, a prototype of the proposed model was created and tested in practice. Results indicated that the automated solution significantly improved the level of information security, reduced the likelihood of human errors and ensured compliance of the organization's policies with the infrastructure security settings. IT security policy management became automated, which resulted in improved transparency in security management processes and more efficient resource utilization were observed, demonstrating the practical applicability and benefit of the proposed model for enterprises aiming to ensure policy compliance and to enhance their IT security posture.

Turinys

Paveikslų sąrašas

Lentelių sąrašas

Įvadas

Šiais laikais informacijos saugumas yra kritiškai svarbus kiekvienos organizacijos stabilumui, patikimumui ir veiklos tęstinumui. Organizacijos, siekdamos apsaugoti savo duomenis ir užtikrinti informacijos saugos politikų atitikimą, susiduria su daugybe iššūkių. Vienas svarbiausių iššūkių – tai žmogiškųjų klaidų rizika, kylanti dėl rankinio politikų įgyvendinimo proceso. Rankinis darbas ne tik lėtina saugumo politikų įdiegimą, bet ir didina neatitikčių bei kibernetinių incidentų riziką, kas gali turėti reikšmingų finansinių ir reputacinių pasekmių. Šio tyrimo motyvacija kilo iš poreikio sumažinti žmogiškųjų klaidų tikimybę ir pagerinti saugumo politikų valdymą organizacijose. Esami IT saugos politikų valdymo sprendimai dažnai apsiriboja tik daline automatizacija, todėl reikalingas kompleksinis, visiškai automatizuotas sprendimas, kuris galėtų interpretuoti saugos politikas tiesiogiai iš tekstinių dokumentų ir automatiškai įgyvendinti jas organizacijos IT infrastruktūroje.

Pagrindinis šio tyrimo tikslas – sukurti ir ištirti automatizuotą IT saugos politikos valdymo modelį, kuris remtųsi natūralios kalbos apdorojimo (NLP) technologija ir „politika kaip kodas“ (angl. „Policy-as-Code“) principu. IT saugos politikos valdymas, naudojant NLP metodus, yra nepakankamai ištirta sritis, todėl matoma tyrimų spraga, kurią šis darbas siekia užpildyti. Siūlomas modelis leis organizacijoms automatizuotai interpretuoti tekstu aprašytas IT saugos politikas ir jas efektyviai įgyvendinti praktikoje, taip pasiekti aukštesnį saugumo ir politikų atitikties lygį bei sumažinti rankinio darbo poreikį.

Darbo tikslui pasiekti iškelti šie uždaviniai:

1. išanalizuoti esamus IT saugos politikos valdymo modelius ir identifikuoti jų trūkumus;
2. suprojektuoti naują automatizuotą IT saugos politikos valdymo modelį, kuris naudotų NLP technologiją ir „politika kaip kodas“ principą;
3. sukurti praktiškai veikiančio modelio prototipą;
4. atlikti sukurtos sistemos kokybinį ir kiekybinį vertinimą bei išanalizuoti rezultatus.

Šį darbą sudaro keturi pagrindiniai skyriai. Pirmajame skyriuje atliekama išsami esamų IT saugos politikos valdymo sprendimų analizė, išryškinant esamus trūkumus ir automatizavimo poreikį. Antrajame skyriuje pristatomas sukurtas automatizuotas politikos valdymo modelis, detalizuojama jo koncepcija, struktūra ir veikimo principai. Trečiajame skyriuje pateikiamas siūlomo modelio prototipo realizavimo procesas, jo integracija į parengtą infrastruktūrą. Ketvirtasis skyrius skirtas išsamiam prototipo tyrimui, vertinant modelio efektyvumą, tikslumą ir saugumą.

Šio tyrimo rezultatai bus naudingi organizacijoms, siekiančioms pagerinti savo IT saugos politikų valdymą, sumažinti klaidų riziką, optimizuoti resursų naudojimą ir užtikrinti realią politikų atitiktį infrastruktūroje. Sukurtas automatizuotas sprendimas ne tik didins saugumo politikų taikymo greitį, bet ir užtikrins geresnę politikų atitiktį, taip reikšmingai prisidėdamas prie bendro organizacijos kibernetinio saugumo stiprinimo.

1. IT saugos politikos automatizuoto valdymo analizė

Šiame skyriuje apžvelgiama IT saugos politikų svarba šiuolaikiniame pasaulyje, esami jų taikymo modeliai ir jų trūkumai. Analizuojamos klasikinės prieigos valdymo teorijos, išryškinami jų įgyvendinimo sunkumai ir aptariamos naujos tendencijos: automatizavimas, „politika kaip kodas“ ir NLP technologijų taikymas, leidžiančios modernizuoti saugos politikų valdymą. Šios analizės išvados taps pagrindu tolimesnei modelio kūrimo motyvacijai.

1.1. Informacijos saugos aktualumas

Šiais laikais informacija tapo vienu svarbiausių resursų tiek individualiu, tiek valstybiniu, tiek technologiniu lygmeniu. Kibernetinės atakos tampa vis sudėtingesnės ir dažnesnės, todėl poreikis apsaugoti informaciją tapo dar didesnis. Informacijos saugos tikslas yra užtikrinti, kad duomenys būtų apsaugoti nuo neteisėtos prieigos. Šiame skyrelyje analizuojami informacijos saugos pagrindai, politikų svarba kaip saugumo įgyvendinimo priemonė.

1.1.1. IT saugumo reikšmė organizacijoms

Informacija šiuolaikiniame skaitmeniniame pasaulyje laikoma vienu svarbiausiu organizacijos turtų, todėl jos apsauga tampa esminiu kiekvienos organizacijos tikslu. Nuolat augant technologijų pažangai, o kartu ir kibernetinių grėsmių skaičiui, organizacijos susiduria su būtinybe ne tik taikyti technologines apsaugos priemones, bet ir sistemiškai valdyti saugos procesus. Visų įmonių pagrindas yra duomenys, kuriuos sudaro informacija. Be duomenų negali vykti net pagrindinės organizacijos funkcijos, pavyzdžiui, bankas negali vykdyti operacijų, sveikatos sistemos negali atvaizduoti pacientų ligų išrašų, valstybinės duomenų agentūros negali atnaujinti piliečių informacijos.

Neturint IT saugos strategijos, organizacijai tai gali kainuoti didelius pinigus. Šiais laikais kibernetiniai incidentai organizacijoms tampa vis brangesni, o jų dažnis – vis didesnis. Remiantis atliktu tarptautiniu tyrimu, vien JAV 2021 m. buvo užfiksuota 466 501 kibernetinė ataka – tai didžiausias pažeidimų skaičius pasaulyje (mažiausiai – 419 atakos Japonijoje), o bendra patirta žala vien Kalifornijos valstijoje siekė net 1,228 milijardo JAV dolerių [1]. Be to, didžiausia dalis sukčiavimo atvejų tais metais buvo susijusi su tapatybės suklastojimais (angl. „imposter scams“), kurie sudarė net 48 % visų sukčiavimo atvejų. Šie skaičiai įrodo, kad kibernetiniai incidentai nėra tik techninė problema – tai tiesioginė finansinė našta verslui, kuri gali lemti milžiniškus nuostolius, reputacijos praradimą ir teisines pasekmes. Todėl organizacijoms būtina iš anksto investuoti į prevenciją ir sistemingai diegti IT saugumo priemones.

Organizacijų IT saugumas gali pagerėti, užtikrinus politikų, sertifikatų ir standartų atitikimą. Įmonėse dažnai naudojami informacijos saugos standartai, tokie kaip: GDPR, NIS2, ISO/IEC 27001, reikalauja iš organizacijų dokumentuotų, įgyvendinamų saugos priemonių (politikų atitikimo), kurios padeda valdyti prieigas, incidentus ir duomenis. Tyrėjai F. Adebol’as, M. Viqaruddin’as, W. Ifeoluw’as ir S.

Bunmi'as atliko tyrimą, kuriame įrodė, kad ISO saugumo standartų įgyvendinimas įvairiose industrijose turi reikšmingą įtaką organizacijos kibernetinio saugumo stiprinimui [2]. Tokie kaip ISO, informacijos saugos standartai, padeda organizacijoms sistemingai identifikuoti pažeidžiamumus, įgyvendinti tvirtas saugos priemones, didina klientų ir suinteresuotųjų šalių pasitikėjimą, sustiprintą duomenų apsaugą ir reglamentų atitiktį. Taip pat buvo pastebėta, kad ISO27001 saugumo standarto įdiegimas įmonėje padėjo išvengti „WannaCry" kibernetinės atakos pasekmių [3]. Be to, įmonės, kurios laikosi šių standartų, dažnai fiksuoja sumažėjusį saugumo incidentų skaičių ir geresnį pasirengimą atremti kylančias grėsmes. Tačiau svarbu pažymėti, kad sertifikato gavimas yra tik pirmas žingsnis – organizacijoms būtina nuolat peržiūrėti ir atnaujinti savo saugumo politiką, atsižvelgiant į kintančias kibernetines rizikas bei teisines prievoles. Taigi, saugumo politikos nustatymas, įgyvendinimas ir atitikimas turi būti vieni svarbiausių uždavinių organizacijoje, kadangi tai gali apsaugoti organizacijos duomenis nuo kibernetinių atakų, išvengti reputacinių ir finansinių nuostolių.

1.1.2. CIA triada

Organizacijos, siekdamos užtikrinti efektyvų informacijos saugos valdymą, remiasi universaliais saugumo principais. Vienas iš labiausiai pripažintų ir plačiausiai taikomų modelių yra vadinamoji CIA triada, kuria grindžiami visi pagrindiniai informacijos apsaugos aspektai.

Konfidencialumas, vientisumas ir prieinamumas (angl. „Confidentiality", „Integrity", „Availability") – šie principai apibendrinami vadinamojoje CIA triadoje, kuri plačiai taikoma tiek teoriniuose saugos modeliuose, tiek praktiniuose informacinių sistemų valdymo sprendimuose. CIA triada padeda struktūrizuoti saugos reikalavimus ir kurti organizacijos poreikius atitinkančias IT saugos politikas.

Informacijos konfidencialumas, vientisumas ir prieinamumas – šie trys komponentai yra informacijos saugumo ramsčiai, į kuriuos orientuojasi visos organizacijos, kurdamos ir įgyvendindamos savo saugumo politikas. Viena iš įmonių, kuri savo IT saugos politiką paremia CIA triada, yra „Fortinet" organizacija. Ji savo informaciniame straipsnyje apibrėžia CIA triados sudėtines dalis per praktinius organizacijos procesus [4]:

- **Konfidencialumas.** Tai įtraukia organizaciją stengtis užtikrinti, kad duomenys yra laikomi slaptai ir privačiai. Norint tai įgyvendinti, prieiga prie informacijos turi būti kontroliuojama taip, kad uždraustų neleistinus veiksmus su duomenimis – tyčinius ar netyčinius.
- **Vientisumas.** Tai užtikrina, kad duomenys yra patikimi ir nesuklastojami. Duomenys tokiais tampa tik tada, jei jie yra autentiški, tikslūs ir patikimi.
- **Prieinamumas.** Sistemos, tinklai, programos turi funkcionuoti, kaip yra nurodyta ir kokiu metu. Asmenys su prieiga į specifinę informaciją turi galėti pasiimti šią informaciją, kai jiems jos reikia, tik per atitinkamą, adekvatų laiką.

Nepriklausomai nuo veiklos srities, bet kokia saugos politika turi būti grindžiama šiais tikslais: ji turi nurodyti, kaip konkrečios priemonės ir taisyklės padeda išlaikyti konfidencialumą, užtikrinti vientisumą ir palaikyti prieinamumą. Jei politika bent šiek tiek praranda konfidencialumo, vientisumo ir prieinamumo kaip vienybės idėją, joje galima įžvelgti pažeidžiamumą. CIA triados svarbą aprašė

mokslininkai, tyrinėję informacijos saugumo rizikos valdymo modelius debesijos sprendimuose [5]. Tyrėjai teigia, kad atsižvelgti į CIA triada būtina įgyvendinant saugumo planus ir plačiai taikyti jį politikoje ir modeliuose, nes organizacijų informacijos saugumo infrastruktūra ir informacijos valdymas tampa pagrindiniu reikalavimu įmonėms, kurios turi informacines sistemas bet kurioje aplinkoje. Nepriklausomai nuo to, ar kalbama apie IT saugos politiką, ar apie saugos modelį, CIA triados tikslai turėtų būti išlaikomi kaip pagrindiniai orientyrai saugumo architektūroje.

1.2. Klasikiniai IT saugos politikos valdymo modeliai ir jų trūkumai

IT saugos politikos yra kuriamos pagal tam tikras saugos taisykles. Norint sukurti tinkamą IT saugos politiką, pasitelkiami įvairūs informacijos saugos modeliai, kurie struktūrizuoja vartotojų prieigas, resursų valdymą ir leidimų kontrolę. Šie modeliai padeda užtikrinti, kad politikos būtų įgyvendinamos nuosekliai, atsižvelgiant į saugumo tikslus, organizacijos struktūrą ir rizikos veiksnius.

1.2.1. Egzistuojantys IT saugos modeliai

Šiuolaikiniame IT pasaulyje egzistuoja daug IT saugos modelių, kuriuos galima pritaikyti organizacijose ir pagerinti jų saugumą. Dabar rinkoje esančius saugos modelius plačiai apibendrina CISSP (angl. „Certified Information Systems Security Professional“) – informacijos saugumo sertifikatas, sukurtas tarptautinio informacijos sistemų saugumo sertifikatų konsorciumo [6]. CISSP sertifikatas apibrėžia šiuos pagrindinius IT saugos modelius:

1. „Bell-LaPadula“ modelis;
2. „Biba“ vientisumo modelis;
3. „Clark-Wilson“ vientisumo modelis;
4. „Brewer and Nash“ modelis;
5. imti-suteikti (angl. „Take-Grant“) modelis;
6. netrukdymo (angl. „Noninterference“) modelis;
7. prieigos valdymo matrica.

„Bell-LaPadula“ modelis. Šį modelį išrado mokslininkai David’as Elliot’as Bell’as ir Leonard’as LaPadula, todėl šis modelis vadinamas „Bell-LaPadula“ modeliu. Jis siekia išlaikyti saugumo *konfidencialumą* (vienas iš CIA triados). Šiame modelyje subjektų (naudotojų) ir objektų (failų) klasifikacija organizuojama nediskretiškai, o atsižvelgiant į skirtingus slaptumo sluoksnius. „Bell-LaPadula“ modelis turi tris taisykles:

1. Paprasta konfidencialumo taisyklė – teigia, kad subjektas gali skaityti tik tame pačiame slaptumo lygmenyje ir žemesniame slaptumo lygmenyje esančius failus, bet ne aukštesniame slaptumo lygmenyje, dėl to šią taisyklę galima vadinti – neskaitymo taisykle.

2. Žvaigždės konfidencialumo taisyklė – teigia, kad subjektas gali rašyti failus tik tame pačiame slaptumo lygyje ir aukštesniame slaptumo lygmenyje, bet ne žemesniame slaptumo lygyje, dėl to šią taisyklę galima vadinti – neįrašymo taisykle.

3. Stiprios žvaigždės konfidencialumo taisyklė – labai apsaugota ir tvirčiausia, kuri teigia, kad subjektas gali skaityti ir rašyti failus tik tame pačiame slaptumo lygyje, o ne aukštesniame ar žemesniame slaptumo lygyje, dėl to taisyklę galima vadinti – neskaitymo ir neįrašymo taisykle.

„Biba“ vientisumo modelis. Šį modelį išrado mokslininkas Kenneth'as Biba, todėl vadinamas „Biba“ modeliu. Jis naudojamas siekiant išlaikyti saugumo vientisumą. Šiame modelyje subjektų (naudotojų) ir objektų (failų) klasifikacija organizuojama nediskretiškai, atsižvelgiant į skirtingus slaptumo sluoksnius. Svarbu pabrėžti, kad veikia atvirkščiai nei „Bell-LaPadula“ modelis. Modelis turi tris taisykles:

1. Paprasta vientisumo taisyklė – teigia, kad subjektas gali skaityti tik tame pačiame slaptumo lygyje ir aukštesniame slaptumo lygmenyje esančius failus, bet ne žemesniame slaptumo lygmenyje, dėl to taisyklę galima vadinti – neskaitymo taisykle.

2. Žvaigždės vientisumo taisyklė – teigia, kad subjektas gali rašyti į failus tik tame pačiame slaptumo lygyje ir žemesniame slaptumo lygyje, bet ne aukštesniame slaptumo lygyje, dėl to šią taisyklę galima vadinti – neįrašymo taisykle.

3. Stiprios žvaigždės vientisumo taisyklė – veikia taip pat kaip „Bell-LaPadula“ modelyje.

„Clark-Wilson“ vientisumo modelis. Šis modelis yra labai saugus. Jame egzistuoja šios esybės: subjektas – bet kuris naudotojas, kuris prašo duomenų, apriboti duomenų elementai – subjektas negali jų tiesiogiai pasiekti (juos reikia pasiekti naudojant „Clark-Wilson“ saugos modelį), neriboti duomenų elementai – subjektas gali tiesiogiai juos pasiekti.

„Clark-Wilson“ saugumo modelio komponentai:

- Transformacijos procesas. Jame subjekto užklausa į apribotus duomenų elementus apdorojama transformavimo procese, kuris paverčia jį leidimais ir perduoda integracijos patvirtinimo procesui.

- Integracijos patvirtinimo procesas. Atlieka autentikaciją ir autorizaciją. Jei pavyko sėkmingai, subjektui suteikiama prieiga prie apribotų duomenų elementų.

„Brewer ir Nash“ modelis yra saugumo modelis, orientuotas į abu CIA triados punktus: privatumą ir konfidencialumą. Modelis dar vadinamas Kinijos sienos modeliu – buvo sukurtas siekiant suteikti prieigos valdiklius, kurie gali keistis dinamiškai, priklausomai nuo ankstesnių naudotojo veiksmų. Pagrindinis modelio argumentas yra tas, kad subjektui suteikiama prieiga tik prie duomenų, kurie neprieštarauja jo turimiems, valdomiems duomenims. Modelis stengiasi apsaugoti naudotojus nuo interesų konfliktų dėl siekiamų ir turimų duomenų [7]. Šis modelis grindžiamas dviem principais:

1. Prieiga suteikiama subjektui, jei toks subjektas jau turi prieigą prie kitų objektų duomenų rinkinyje, kur reikalinga nauja prieiga.

2. Prieiga suteikiama, jei objektas, kurį reikia pasiekti, visiškai priklauso kitai interesų konflikto klasei.

Imti-suteikti modelis veikia kaip diagrama, kurios viršūnės yra subjektai arba objektai. Linijos tarp jų yra pažymėti etiketėmis, o kiekviena etiketė nurodo teises į resursą. Kiekviename įgyvendintame modelyje yra dvi teisės: imti ir suteikti. Jos atlieka ypatingą vaidmenį grafiko perrašymo taisyklėse – apibūdina leistinus grafiko pakeitimus. Iš viso modelis turi keturias taisykles:

1. „imti“ taisyklė – leidžia subjektui perimti kito objekto teises (pridėti liniją, kilusią iš subjekto);

2. „suteikti“ taisyklė – leidžia subjektui suteikti savo teises kitam objektui (pridėti liniją, kuri baigiasi ties subjektu);

3. „kurti“ taisyklė – leidžia subjektui kurti naujus objektus (pridėti viršūnę ir liniją nuo subjekto link naujos viršūnės);

4. „pašalinti“ taisyklė – leidžia subjektui pašalinti teises, kurias jis turi kitam objektui (pašalinti liniją, kilusią iš subjekto).

Netrukdymo modelyje (taip pat žinomas kaip „Goguen-Meseguer“ saugumo modelis) dėmesys sutelkiamas į tai, kaip subjekto didesnio jautrumo lygio veiksmai daro įtaką sistemos būsenai arba subjekto žemesnio jautrumo lygio veiksmams (tai bus trukdžiai). Naudotojai (subjektai) yra savo skyriuose ir padalinti į atitinkamas grupes, todėl informacija nenuteka iki kitų skyrių. Taikant netrukdymo saugumo politiką, netrukdymo modelis gali išreikšti kelių lygių saugumą, pajėgumų perdavimą, izoliavimą, suskirstymą, diskretinę prieigą, kelių naudotojų (raktų) prieigą, automatines paskirstymo ir autorizacijos grandines bei pažeminimą. Informacijos srautą kontroliuoja saugumo politika, kuri yra nesikišimo tvirtinimų (t. y. „pajėgumų“) rinkinys.

Prieigos valdymo matricos saugos modelis sudarytas iš lentelės, kuri apibrėžia konkrečių subjektų ir objektų prieigos teises. Matrica yra duomenų struktūra, kuri veikia kaip operacinės sistemos lentelės paieška. Pavyzdžiui, 1.1 lentelė yra matrica, turinti konkrečius naudotojo apibrėžtus prieigos leidimus ir išsamiai nurodo, kokius veiksmus jis gali atlikti. Naudotojas Jonas turi skaitymo ir rašymo prieigą prie duomenų failo, taip pat turi prieigą prie programos paleidimo. Naudotojas Antanas gali skaityti duomenų failą ir turi prieigą prie programos. Naudotoja Albina neturi prieigos prie duomenų pagal šią prieigos matricą.

1.1 lentelė. Lentelės eilutės nurodo kiekvieno subjekto galimybes. Šios eilutės vadinamos galimybių sąrašu. Stulpeliai nurodo prieigos sąrašą kiekvienam objektui ar aplikacijai.

Naudotojas	Duomenų failas	Programa
Jonas	Skaityti ir rašyti	Vykdyti
Antanas	Skaityti	Vykdyti
Albina	-	-

1.2.2. IAM karkasai

Identitetų ir prieigos valdymas (IAM) yra politikų, procesų ir technologijų karkasas, kuris leidžia organizacijoms valdyti skaitmeninius identitetus ir kontroliuoti naudotojų prieigą prie kritinės organizacijos informacijos [8]. IAM sudaro trys komponentai:

1. tapatybių saugojimas – kredencialų atradimas ir saugojimas, autentikacijos paslaugos;
2. tapatybių valdymas – apima funkcijas, susijusias su tapatybės administravimu, atsižvelgiant į gyvavimo ciklą;
3. prieigų valdymas – reiškia procesą ir technologijas, naudojamas norint pasiekti organizacijos specifines programas, informaciją ir išteklius, kartu su pagrindimu dėl teisių ir privilegijų.

Šiuo metu egzistuoja daug IAM modelių, kuriuos organizacijos gali įsidiegti savo tinkle. Apie juos apibendrintai bus aptarta šiame skyriuje:

1. DAC – diskrecinis prieigos valdymo modelis;
2. MAC – imperatyvinis prieigos valdymo modelis;
3. RBAC – rolėmis pagrįstas subjektų valdymo modelis;
4. ABAC – požymiais pagrįstas subjektų valdymo modelis;
5. RuBAC – politikomis pagrįstas prieigos modelis;
6. „Zero Trust“ modelis.

DAC (angl. „Discretionary Access Control“) – diskrecinis prieigos valdymo modelis pasižymi tuo, kad objekto savininkas nustato, kas gali prieiti prie jo turimo objekto. DAC mechanizmai yra valdomi pagal naudotojo identifikavimo duomenis, pvz., naudotojo vardą ir slaptažodį. DAC yra diskrecinis, nes savininkai gali perduoti objektus arba bet kokią autentifikuotą informaciją kitiems naudotojams. Paprasčiau tariant, savininkas gali nustatyti prieigos privilegijas. Pavyzdys – „Linux OS“ teisių priskyrimas failui [9]. DAC požymiai:

- naudotojai gali perduoti savo objekto nuosavybę kitam naudotojui;
- naudotojas gali nustatyti kitų naudotojų prieigos tipą;
- nesėkmingas autorizavimas gali apriboti naudotojo prieigą po kelių nesėkmingų bandymų;
- neįgalioti naudotojai nematys objekto charakteristikų, vadinamų failo dydžiu, katalogo keliu ir failo pavadinimu.

MAC (angl. „Mandatory Access Control“) – imperatyvinis prieigos valdymo modelis, kuriame subjektams suteikiama mažai laisvės sprendžiant, kas gali prieiti prie jų turimų objektų. MAC suteiks prieigą naudotojui pagal jo tapatybę ir duomenis. Norėdamas gauti prieigą, naudotojas turi pateikti savo asmeninę informaciją. Tai labai saugu, nes taisykles ir apribojimus nustato administratorius ir jų tinkle griežtai laikomasi. MAC nustatymų ir politikos valdymas yra talpinamas saugiame tinkle ir juos gali naudoti tik sistemos administratoriai. MAC dažnai galima rasti įgyvendintą vyriausybinėse organizacijose. MAC požymiai:

- MAC politika gali padėti sumažinti sistemos klaidų skaičių;
- ji pasižymi griežtesniu saugumu, nes tik administratorius gali pasiekti ar keisti valdymo priemones;
- MAC turi įdiegtą sistemą, kuri gali žymėti ir atriboti gaunamus taikomųjų programų duomenis.

RBAC (angl. „Role Based Access Control“) – rolėmis pagrįstas subjektų prieigos valdymo modelis, kuriame sprendimai priimami atsižvelgiant į subjektų roles. Šiais laikais yra dažniausiai naudojamas ir matomas padidėjęs organizacijų susidomėjimas įsidiegti RBAC modelį, nes jį lengva valdyti ir jis užtikrina patikimą prieigos kontrolę, kad būtų apsaugotas jų skaitmeninis turtas [10]. Naudojant RBAC, rolė kontroliuoja naudotojo prieigą ir teises naudotis tam tikru išteklių rinkiniu. Todėl keli naudotojai gali turėti vieną rolę, o vienas naudotojas gali turėti kelias roles. Šiame modelyje panaikinti prieigos teises yra paprasta, kaip ir jas suteikti. Norint nustoti naudotis resursais, organizacijai tereikia nuimti konkrečią rolę nuo to naudotojo.

ABAC (angl. „Attribute-Based Access Control“) – atributais pagrįstas prieigos valdymo modelis. Tai gali būti atributai konkretaus asmens, resurso arba aplinkos. Pavyzdžiui, subjekto atributai (asmens ūgis), resurso atributai (programinė įranga, kuri veikia tik tam tikroje operacinėje sistemoje arba interneto svetainėje) arba aplinkos atributai (paros laikas arba laiko trukmė). ABAC privalumas yra tas, kad administratoriai gali kurti kontrolės politiką nežinant naudotojų, o naujų naudotojų įtraukimas yra paprastesnis. Kiekviena operacijos ir objekto pora kiekvienam subjektui ar rolei priskiriama iš karto, o ABAC variklis gali suteikti arba uždrausti leidimą pagal priskirtus atributus, prieigos kontrolės politiką ir aplinkos sąlygas.

RuBAC (angl. „Rule-Based Access Control“) – naudoja užprogramuotą sąlygų arba taisyklių rinkinį (politikų rinkinį), kurį įveda sistemos administratorius. Tuomet mechanizmas, atsižvelgęs į sąlygas, nustato, ar subjektas gali gauti prieigą prie objekto. Tai administratoriaus atsakomybė priskirti kiekvienam subjektui būtinus veiksmus, kuriuos jie gali atlikti su objektu. Kai kurie modeliai, suteikdami leidimą, atsižvelgia tik į subjektą ir objektą, o RuBAC atsižvelgia ir į veiksmą. Galima palyginti su programuotojų naudojamais teiginiais „if-then“ ir „if-then-else“, todėl šis modelis gali naudoti daug sąlygų ir kintamųjų, kad suteiktų arba atsisakytų suteikti leidimą į prieigą [11]. Pavyzdžiui, organizacija turi dokumentus, kuriuos galima būtų prieiti tik tam tikru paros metu arba tam tikroje geografinėje vietovėje – tam gali būti naudingas RuBAC modelis.

„Zero Trust“ saugumo modelis – nulinio pasitikėjimo modelyje prieigos kontrolės centras yra identiteto autentiškumo nustatymas. Pagrindinės modelio atsakomybes apima tapatybės autentifikavimą,

nuolatinį rizikos vertinimą ir dinaminę prieigos kontrolę. Modelis sudarytas iš patikimų agentų, dinaminės prieigos kontrolės variklių ir tapatybės autentiškumo nustatymo variklių. Kai subjekto patikimumas užtikrinamas ir jis pereina per daugelį saugumo sluoksnių, tik tada jis gauna prieiga prie objektų. Pagrindinis principas – „niekada nepasitikėk, visada tikrink“ – yra ZT saugumo modelio aukščiausioji taisyklė [12].

Taigi, šie IAM modeliai yra dažniausiai literatūroje ir praktikoje sutinkami modeliai. Jie padeda užtikrinti tinkamą prieigos nustatymą valdomiems identitetais, padaro politiką struktūrizuotą pagal tam tikras taisykles. Tačiau, tobulėjant programišiams ir jų kibernetinėms atakoms, atsiranda vis daugiau trūkumų šiuose modeliuose, kurios šiuolaikinėse įmonėse gali sukelti daug grėsmių.

1.2.3. IT saugos politikų valdymo sprendimai

Organizacijos turi siekti atitikimo nustatytoms IT saugos politikoms. Politikos gali būti sukurtos, naudojant klasikinius saugos ar IAM modelius. Juos įdiegti gali politikų valdymo ir diegimo sprendimai. Tai gali būti paruošti valdymo įrankiai arba rankinė valdymo struktūra. Tokie įvairūs sprendimai, skiriami politikų diegimui, konfigūracijų valdymui bei įrenginių paruošimui. Visgi šie sprendimai dažnai apsiriboja tik konfigūracijų vykdymu, tačiau nesugeba interpretuoti organizacijos dokumentacijos ar automatizuoti viso saugumo taisyklių įgyvendinimo ciklo. Šiuo metu dažniausiai naudojami politikų valdymo sprendimai: rankinis diegimas, automatinio diegimo įrankiai („Ansible“, „Puppet“) bei politikų valdymas per debesijos paslaugas („Microsoft Intune“).

Dažnai galima pastebėti, kad įmonės (ypač mažesnio dydžio) diegia saugumo nustatymus **„rankiniu būdu“**. Tai gali daryti OS komandomis, GPO nustatymais, sistemos konfigūracijomis. Organizacijos, kurios nepasiryžusios išleisti daugiau pinigų saugumui, dažnai palieka IT saugos politikų atitiktį IT administratoriui, kuris rankiniu metodu ir savo nuožiūra stengiasi įgyvendinti saugumo standartus. IT administratoriui tai gali kainuoti daug laiko, sukurti daug klaidų ir, galiausiai, daug pažeidžiamų vietų įmonės infrastruktūroje. Pavyzdžiui, organizacijos kompleksiniuose ir dinaminiuose tinkluose (IoT, debesijos kompiuterijoje) tokie rankiniai procesai gali daug lėčiau aptikti ir ištaisyti problemas, tai gali sukelti klaidas konfigūruojant išteklius ir netvarkingai įgyvendinti saugos politiką, todėl sistemos tampa pažeidžiamos dėl atitikties problemų ir grėsmių [13]. Taigi, rankinis būdas yra dažnas organizacijose, tačiau nerekomenduojamas, jei norima įgyvendinti IT politikų atitiktį kompleksinėje sistemoje.

Norint automatizuoti rankinius darbus, kartais organizacijos pasinaudoja **automatiniais diegimo įrankiais**, kaip „Ansible“, „Puppet“. Šie įrankiai gali vienu metu valdyti didelį kiekį įrenginių ir paleisti konfigūracijas centralizuotai. IT administratoriaus paruoštos užduotys-skriptai (angl. „playbooks“ arba „manifests“) įgyvendinamos automatizuotai, taip sumažinant žmogiškųjų klaidų tikimybę. Tačiau šių įrankių naudojimas reikalauja IT administratoriaus techninių žinių, kadangi užduočių paruošimui naudojamos specialios skriptavimo kalbos: YAML, DSL ir kt. Todėl pasiruošimo procesas gali kainuoti daug laiko, bet, paruošus infrastruktūrą, centralizuotas diegimas taps greitesnis už „rankinį“. Viename tyrime, kuriame buvo naudojamas automatizuoto diegimo įrankis „Ansible“ žiniatinklio programai sukurti, buvo nustatyta, kad šio diegimo valdymo sprendimas pagreitino paslaugų diegimą 5,6 kartais už rankinį diegimą [14]. Taip pat išvadose tyrėjai teigė, kad šis įrankis yra patikimas naudoti sudėtingose užduotyse, IT infrastruktūros valdyme ir netgi saugumo didinimui sistemose. Taigi,

norint sutaupyti įmonės laiką ir resursus, rekomenduojama naudoti automatizuotus įrankius, kurie gali sumažinti žmogiškųjų klaidų riziką ir patikimai įgyvendinti pakeitimus sistemose ar infrastruktūroje.

Dar vienas iš IT saugos politikų valdymo sprendimų yra **debesijos platformos**, kurias dažnai naudoja didesnės organizacijos, turinčios didelį kiekį naudotojų ir įrangos. Viena iš tokių platformų yra „Microsoft Intune“, kuri taip pat orientuota į grafinę sąsają, nuotolinį įrenginių valdymą ir paprastesnį politikų taikymą. Nors ši platforma geriausiai tinka „Windows OS“ naudojančiuose įrenginiuose, konfigūracijų diegimas yra ribotas kitose OS ir vis tiek reikalauja rankiniu būdu suvedamų įmonės nustatytų politikų. „Intune“ įrankis buvo paminėtas straipsnyje apie saugumo procesų supaprastinimą didelėse organizacijose, bet išlaikant reikalingą saugumo lygį [15]. Šiame straipsnyje teigiama, kad „Intune“ tinkamas naudoti įvairiems įrenginiams (telefonams, planšetiniams ir nešiojamiems kompiuteriams), gali kontroliuoti programas pagal tam tikras politikas, užtikrina saugumo nustatymų įgyvendinimą įrenginiuose, taip pat atlieka naudotojų autorizaciją pagal vietovę ar kitus parametrus. Lenkų autorės teigia, kad su „Intune“ galima užtikrinti, kad įrenginiai ir programėlės atitinka nustatytus įmonės saugumo reikalavimus. Tačiau IT administratoriaus įsikišimo reikia ir šiame politikų valdymo sprendime. Administratorius turi išanalizuoti įmonės IT saugos politikas ir reikalavimus, nustatyti ir įdiegti įrenginiuose per „Intune“ platformą atitinkamas konfigūracijas ir tik tada stebėti platformoje politikų atitikimo rezultatus ir statusus. Taigi, tai tikrai sutaupo daug laiko, tačiau reikalauja didelio administratoriaus įdirbio ir žinių, naudojantis tokiomis politikų valdymo sistemomis.

Kiekvienas IT saugos politikų valdymo būdas palyginamas 1.2 lentelėje. Išanalizavus sprendimų galimybes, galima įžvelgti kiekviename sprendime trūkumų.

1.2 lentelė. IT saugos politikų valdymo sprendimų palyginimas

Kriterijus	**Rankinis būdas**	**„Ansible“/„Puppet“**	**„Intune“**
Automatinis diegimas į įrenginius	Nėra	Yra	Yra
Automatinis skripto taikymas	Nėra	Yra	Yra
Politikų interpretavimas iš politikos	Nėra	Nėra	Nėra
Reikalaujamas techninis pasirengimas	Aukštas	Aukštas	Vidutinis
Operacinių sistemų palaikymas	Universalus	„Linux“ / „Windows“	Ribotas („Windows“)
Žmogiškųjų klaidų tikimybė	Aukšta	Vidutinė	Vidutinė
Naudojimo patogumas	Sudėtingas	Vidutinis	Patogus
Konfigūracijų keitimo lankstumas	Lankstus	Labai lankstus	Ribotas
Sąsajos tipas	Komandinė eilutė	Skriptai (YAML, DSL)	Grafinė vartotojo sąsaja

Apibendrinant galima teigti, kad šiuo metu egzistuojantys IT saugos politikų valdymo sprendimai padeda vykdyti politikų įgyvendinimą, tačiau nė vienas jų nesprendžia viso politikos valdymo ciklo – nuo IT politikos reikalavimų įmonės viduje iki atitinkamai sudiegtų nustatymų įmonės galiniuose

įrenginiuose ir sistemose. Galima įžvelgti automatizavimo trūkumą, kuris organizacijoms gali būti ypač svarbus sprendimas. Šiais laikais organizacijos reikalauja kuo daugiau automatizuotų sprendimų. Dėl šios priežasties kyla poreikis integruoti automatizuotą dokumento analizę, natūralios kalbos interpretaciją ir automatinį politikos nustatymų generavimą į vieną sprendimą. Toks požiūris ne tik sumažintų administracinę naštą, bet ir užtikrintų didesnį atitikimą tarp teorinių politikų ir realių saugumo konfigūracijų.

1.2.4. Problemos esamuose IT saugos politikų valdymo sprendimuose

Informacijos saugumo valdymo srityje egzistuoja įvairūs modeliai, karkasai ir įrankiai, bet jų diegimas organizacijose kelia nemažai iššūkių. Literatūroje aptarti klasikiniai modeliai (pvz., „Bell-LaPadula“, „Biba“, „Clark-Wilson“), IAM prieigos valdymo modeliai (pvz., DAC, MAC, RBAC, ABAC, „Zero Trust“) ir praktiniai sprendimai (pvz., „Rankinis darbas“, „Ansible“, „Puppet“, „Intune“) dažnai veikia izoliuotai ir atskirai vienas nuo kito, orientuoti į vieną aspektą arba taikomi labai specifiniuose kontekstuose.

Reikia atkreipti dėmesį, kad ne visi paminėti modeliai tenkina visus tris CIA triados tikslus. Todėl naudoti tik vieną modelį yra rizikinga dėl galimų pažeidžiamų vietų. Pavyzdžiui, „Bell-LaPadula“ neturi vientisumo ir prieinamumo tikslų, tuo pačiu neturi prieigos kontrolės valdymo ir failų dalinimosi taisyklių. Šis modelis, kaip buvo paminėta, užtikrina tik informacijos konfidencialumą. Galima prisiminti ir „Biba“ vientisumo modelį – jis užtikrina tik informacijos vientisumą. Šiuolaikiniame ir technologiniame pasaulyje tik vieno tikslo neužtenka. Be to, jie sukurti statinėms, hierarchinėms sistemoms, todėl sunkiai pritaikomi debesijos, IoT ar kompleksiniuose tinkluose. IAM modeliai yra lankstesni, tačiau jų praktinis diegimas – kompleksinis, reikalaujantis daug laiko, lėšų ir išankstinės infrastruktūros paruošimo.

Nors „Bell-LaPadula“ ir „Biba“ modeliai yra lankstūs, tačiau pritaikymas ir valdymas yra labai sudėtingas – taip teigia tyrėjai, aprašę teigiamas ir neigiamas savybes šiems modeliams [16]. Tyrėjai testavo įvairius modelius, diegiant modelio politikas debesijos kompiuteriuose. Dažnai minima, kad šie modeliai yra labai lankstūs, galima pritaikyti daugumai sistemų ir organizacijų, tačiau iš to atsiranda ir pačio modelio sudėtingumas. Lankstumas dažnai išauga į sudėtingumą, nes laisvė ir daug galimybių nėra tinkamai struktūrizuota pagal taisykles. Kaip tyrime apie modernių tinklų saugumo modelį išdėstoma [17], įgyvendinimas darosi vis sunkesnis naudojant šiuolaikines tinklo technologijas, tokias kaip debesijos sprendimai, daiktų internetas ir programinės įrangos tinklas, kurių komponentai su laiku keičiasi greitai ir dinamiškai. Todėl ypač didelėse organizacijose sudėtinga įgyvendinti tokį „laisvą“ modelį. Tai gali apkrauti IT administratorius sudėtingais procesais, užimti daug laiko ir resursų. O mažos ir vidutinės įmonės iš viso mažiau naudoja saugos modelius, nes negali sau leisti naudoti brangias technologijas, todėl mažos įmonės investuoja daug mažiau į IT servisų infrastruktūrą ir darbo vietą [18]. Tai reiškia, kad modeliai teoriškai atrodo suteikiantys daugiau saugumo organizacijoje, tačiau, priklausomai nuo įmonės dydžio, organizacijos kartais vengia diegti saugumą užtikrinančius modelius dėl pinigų ir žmogiškųjų resursų stokos.

Vieną didžiausią problemą dažni tyrėjai įvardija pačių darbuotojų nekompetentingumą [19], saugumo politikų neišmanymą, naivumą kibernetinių įsilaužėlių atžvilgiu. Žmonės yra silpniausia grandis įmonės saugume – kad ir koks geras modelis, politikos, jei žmonės nėra apmokomi, neturi pakankamai

žinių, viskas gali nueiti veltui. Daugelyje IT saugos politikų valdymo sprendimuose yra būtinas administratoriaus stiprus specialų techninių žinių išmanymas, norint užtikrinti sėkmingą politikų įgyvendinimą organizacijoje. Kadangi aptartuose karkasuose ir modeliuose yra dar vis reikalingas rankinis darbas, tam reikia vykdyti didesnę žmogiškųjų klaidų patikrą ir samdyti daugiau bei stipresnius kandidatus į šią poziciją. Taip pat tai kainuos daug laiko ir pinigų. Tinklų saugumo konfigūracijos tyrime yra teigiama: „jei modeliai turėtų didesnį kiekį automatizacijos procesų, tai pagerintų apsaugą prieš kibernetines atakas, padėtų subalansuoti didėjantį kompiuterių tinklų kompeksiškumą ir dydį, suteiktų dinamikos politikomis grįstam infrastruktūros valdymui“ [20]. Todėl, neturint automatizavimo ar esant mažam jo kiekiui, modelis tampa nesaugus, pažeidžiamas dėl žmogiškųjų klaidų ir netinkamas naudoti šiuolaikinėse įmonėse, kurios yra dinamiškos.

Apibendrinus esamų modelių, karkasų ir sprendimų analizę, galima išskirti pagrindines šioje srityje egzistuojančias problemas:

- modelių saugos ribotumas – neaprėpia visų CIA triados tikslų;
- prastas pritaikomumas dinamiškose, moderniose sistemose;
- sudėtingas, ilgas ir daug resursų reikalaujantis įgyvendinimas;
- pernelyg lankstūs modeliai tampa sunkiai valdomi;
- dažnas rankinių sprendimų naudojimas didina klaidų riziką;
- modeliams įgyvendinti, administratoriams reikia turėti specifines technines žinias;
- per žemas automatizacijos lygis esamuose sprendimuose.

1.3. IT saugos politikų automatizavimas ir NLP taikymo galimybė

Šiame skyrelyje yra išnagrinėjama automatizavimo svarba ir galimybės, naujasis būdas interpretuoti IT politiką – tai naujų technologijų natūralios kalbos apdorojimo (NLP) modeliai ir „politikos kaip kodas“ principas. Išanalizuojamos NLP galimybės ir pritaikymas automatizuotame IT politikos valdyme.

1.3.1. Automatizavimo poreikis IT politikos valdyme

Šiuolaikinėse organizacijose IT saugos politikos valdymas vis dar dažnai vykdomas rankiniu būdu. Tai atlieka administratoriai, interpretuodami politikas savaip ir kurdami konfigūracijas organizacijos įrenginiams ar naudotojams. Nors dokumentuose politikos yra apibrėžtos tiksliai, pagal taisykles ir standartus, tačiau jų įgyvendinimas praktikoje reikalauja žmogaus įsitraukimo. Šis procesas yra ne tik lėtas ir naudojantis daug resursų, bet ir priklausomas nuo žmogaus, kuris atlieka politikų įgyvendinimą organizacijoje. Pagal Kalvin’o Nobles’o tyrimą, žmogiškieji faktoriai, tokie kaip: stresas, perdegimas, nuovargis, daro tiesioginę neigiamą įtaką įmonės kibernetiniam saugumui, įtraukiant ir politikos diegimą [21]. Taip pat straipsnyje kalbama apie padidėjusią kibernetinių atakų tikimybę, jei

šie žmogiškieji faktoriai yra įmonėje, o tai kelia didelę grėsmę. Kitame straipsnyje teigiama, kad apsiskaičiavimas (įprasta žmogiška klaida) iššaukia daugiau nei 80 % kibernetinių incidentų, duomenų pažeidimų ir kenkėjiškų programų atakų [22]. Taip pat teigiama, kad vienas iš svarbiausių žmogiškųjų faktorių yra darbuotojų atsidavimas verslo saugumo politikai, kur žmogaus sąmoningumo stoka gali turėti didelių pasekmių, tokių kaip: kibernetinio saugumo pažeidimai dėl žmogaus išsiblaškymo ir dažnos klaidos saugumo politikose dėl neapdairumo bei nesigilinimo.

Vienas iš būdų, kaip išspręsti žmogiškųjų klaidų problemą, diegiant saugos politikas organizacijoje, yra automatizavimas. Automatizuoti įrankiai leidžia realiuoju laiku stebėti organizacijos saugos būklę ir operatyviai reaguoti į kylančias grėsmes – taip mažinant priklausomybę nuo žmogiškųjų faktorių. Tyrėjai pažymi, kad nuolatinė stebėsena, pagrįsta automatizuotais sprendimais ir reguliariais auditais, yra būtina siekiant sumažinti saugumo pažeidimų riziką ir užtikrinti organizacijos atitiktį teisės aktams bei trečiųjų šalių valdymo reikalavimams [23]. Straipsnyje apie autonomines sistemas „Zero Trust" architektūroje minima, kad šiuolaikinėse ypač dinaminėse, daug konteksto turinčiose ir didelėse įmonėse automatizuoti mechanizmai pagerina ir papildo „Zero Trust" politikų architektūrą bei sprendžia organizacijos greitai kintančias grėsmes [24]. O papildžius automatizuotus sprendimus su dirbtiniu intelektu (DI), sistemos gali padėti pagerinti tinklo saugumą, tuo pačiu metu sumažinant poveikį produktyvumui. Tad žmogiškųjų išteklių pakeitimas į automatizuotas ar DI sistemas, pastiprina įmonės saugumą ir padidina produktyvumą. Ateities tendencijos rodo, kad saugumo politikų valdymas vis dažniau bus grindžiamas automatizavimo ir DI sprendimais, kurie leis dar labiau pagerinti saugos kontrolę [23].

1.3.2. Politika kaip kodas („Policy-as-Code")

Kitas modernus automatizuotas politikų diegimo metodas yra „politika kaip kodas" (angl. „Policy-as-Code" – PAC). Tai saugumo politikų perkėlimas iš žmogui suprantamo formato (teksto) į mašinai suprantamą, struktūrizuotą formatą (pvz., JSON, YAML, HCL). Tai nusistovėjusį tradicinį metodą, t. y. rašytiniai dokumentai, pakeičia į kodą, kuris nurodo ir įgyvendina taisykles bei procedūras. Taip organizacijos politika gali būti konvertuojama į programinį kodą ir plačiai sudiegiama organizacijoje. Politikų valdyme PAC suteikia daug privalumų [25]:

- **Automatizavimas.** Politikos gali būti taikomos automatiškai, taip sumažinant žmogiškųjų klaidų ir nenuoseklumo riziką.
- **Versijų kontrolė.** Politikos pakeitimus galima sekti ir kontroliuoti, užtikrinant, kad visi dirbtų su naujausia versija.
- **Integracija.** PAC galima integruoti su kitomis sistemomis ir įrankiais, pavyzdžiui, automatizavimo sistemomis ir saugumo platformomis.
- **Bendrinimas.** Politikas galima lengvai bendrinti ir diegti tarp vidaus komandų ir įrenginių organizacijoje dėl savo universalumo.
- **Atitiktis.** PAC gali padėti organizacijoms laikytis industrijos reglamentų ir standartų.

PAC poreikis atsirado dėl kelių iššūkių organizacijose. Tradicinės politikos talpinamos dokumentuose (pvz., „MS Word" failai), kuriems reikia žmogaus interpretacijos, dėl kurios atsiranda klaidų. Taip pat svarbu suprasti problemą, kad skirtingi žmonės skirtingai interpretuodavo taisykles. PAC išsprendžia greitą politikų atnaujinimą bei diegimą visoje sistemoje ir veiksmingą politikų auditą ar tikrinimą, įvertinant atitiktį.

PAC svarba buvo nagrinėta tyrime, kuriame įvertinamas debesijos kompiuterių saugumas, atitiktis ir prieigos kontrolė po automatizuotų ir PAC saugumo politikų diegimo [26]. Šie saugumo sprendimai buvo diegiami „be serverių" (angl. „serverless") debesijos kompiuterijoje ir paaiškėjo, kad tokių saugumo sprendimų diegimas ryškiai pagerino politikos vykdymo tikslumą, atitikties tikrinimo našumą ir prieigos kontrolės tikslumą, tuo pačiu metu sumažinant našumo sąnaudas. Išskiriant būtent „politika kaip kodas", šis sprendimas dinamiškai ir nuosekliai sugebėjo įgyvendinti politikas net keliose debesijos platformose, užtikrinant mažiausių privilegijų principą. Tai įrodo, kad šis sprendimas yra dinamiškas ir užtikrinantis atitiktį.

„Politikos kaip kodas" privalumai:

1. greitas politikų diegimas ir atnaujinimas;
2. automatizuotas nuoseklumo užtikrinimas;
3. galimybė atlikti politikų auditą ir testavimą kaip programinį kodą;
4. pagerėja atsekamumas: kas, kada ir kaip keitė politikas.

„Politikos kaip kodas" trūkumai:

1. reikia specifinių techninių žinių (skriptavimo kalbos);
2. pradiniam diegimui reikia daug laiko pasiruošimui;
3. išlieka rizika, kad klaidingai užrašyta politika sukels netinkamą sistemos elgesį.

Apibendrinant, „politika kaip kodas" suteikia galimybę organizacijoms automatizuoti saugumo politikų valdymą, užtikrinant didesnį nuoseklumą, spartesnį reagavimą ir lengvesnį atitikties užtikrinimą. Nepaisant tam tikrų įdiegimo iššūkių, šis metodas tampa būtina šiuolaikinių IT saugumo architektūrų dalimi, ypač debesijos ir „be serverių" aplinkose.

1.3.3. Natūralios kalbos apdorojimo (NLP) veikimo principai

Pastaruosius keletą metų, ypač paspartėjus DI technologijoms, populiarėja DI naudojimas įvairiose srityse. IT saugos politikų valdyme vienas iš automatizuotų sprendimų gali būti šiuo metu populiarus ir plačiai naudojamas natūralios kalbos apdorojimas (angl. „Natural Language Processing" – NLP). Tai yra DI sritis, skirta žmogaus kalbos supratimui, analizei ir generavimui. NLP pritaikymas yra labai platus: teksto analizė, dokumentų klasifikavimas, paieškos sistemos optimizavimas. Vienas iš

pavyzdžių gali būti kenkėjiškų laiškų nuskaitymas su NLP. Toks tyrimas buvo atliktas Saudo Arabijoje, kur tyrėjai eksperimento metu nustatė, kad mašininio mokymo ir natūralaus kalbos apdorojimo sprendimai, panaudoti kenkėjiškų laiškų klasifikavimo įrankiui, sukūrė aukštą tikslumo įvertinimą – net 98,2 % [27]. Galima teigti, kad NLP sprendimai yra patikimi ir juos galima naudoti organizacijos kibernetiniam saugumui gerinti.

Šis automatizavimo būdas su NLP gali paspartinti ir patikslinti IT saugos politikų diegimo procesą organizacijoje. Kadangi dauguma organizacinių dokumentų (tarp jų ir IT saugos politika) yra parašyti natūralia kalba, kurią NLP modeliai gali lengvai suprasti, interpretuoti ir išvesti atsakymą. Net aukšto lygio tinklo saugumo politikos būna parašytos natūralia, žmogiška kalba. Tinklo saugumą nagrinėję tyrėjai teigia, kad NLP naudojimas natūralios kalbos tinklo politikų klausimams išversti į tinklo kodo eilutes pavyko sėkmingai [28]. Tai sumažino žmogiškųjų klaidų skaičių ir pagreitino politikų konfigūravimo procesą. Taigi, NLP leidžia automatizuoti natūralios kalbos dokumentų supratimą, ištraukti svarbiausias sąvokas, identifikuoti taisykles ir parengti struktūrizuotą informaciją. NLP panaudojimas gali stipriai pagerinti organizacijos IT saugos politikos valdymą bei diegimą.

Norint naudoti NLP modelius tekstinių politikų interpretavimui, svarbu suprasti NLP veikimo principus ir etapus:

1. **Teksto segmentavimas (angl. „Tokenization“).** Tekstas yra suskaidomas į mažesnius vienetus – žodžius, sakinius arba pastraipas. Šis etapas būtinas tam, kad sistema galėtų analizuoti kalbą struktūrizuotai ir suprasti jos sudedamąsias dalis.

2. **Kalbos dalių žymėjimas (angl. „Part-of-Speech Tagging“).** Kiekvienam teksto žodžiui priskiriama gramatinė kategorija, tokia kaip daiktavardis, veiksmažodis, būdvardis ar prieveiksmis. Ši informacija leidžia tiksliau suprasti sakinio struktūrą ir žodžių tarpusavio ryšius.

3. **Tikrinių daiktavardžių ir vardinių vienetų atpažinimas (angl. „Named Entity Recognition“).** Sistema identifikuoja tikrinius daiktavardžius, tokius kaip asmenų vardai, vietovės, organizacijos pavadinimai, dokumentų pavadinimai. Tai padeda išskirti esminę informaciją iš didelio kiekio teksto.

4. **Sakinio dalių analizė (angl. „Parsing“).** Atliekama gilesnė sintaksės ir sakinio dalių analizė – nustatomi gramatiniai ryšiai tarp žodžių, pavyzdžiui, kas sakinyje yra veiksnys, tarinys, papildinys, pažyminys.

5. **Semantikos analizė (angl. „Semantic Analysis“).** Bandoma suprasti gilesnę teksto prasmę – nustatyti žodžių ir sakinių reikšmes, aptikti užslėptas reikšmes, ketinimus, intencijas ar kontekstines užuominas.

Tokie NLP etapai padeda sistemai ištraukti struktūrinę prasmę iš laisvos formos teksto, sudėti į atitinkamus parametrus ir perduoti sistemai toliau dirbti su išskirtais parametrais. Toks būdas itin paspartina sistemų procesus, sumažina klaidų riziką (nes viskas vyksta automatizuotai) ir užtikrina kokybišką atsakymą. Nors NLP labai pažengęs, tačiau gali iškilti sunkumų, kai dirbama su nesusistemintu, netvarkingu tekstu. Norint tokių problemų išvengti, reikia nurodyti modeliui tikslią įvesties dokumento

struktūra ir nusistatyti taisykles. Tada NLP modeliui sumažėja rizika nukrypti į kitokias prasmes ar kontekstus.

Pagrindiniai NLP iššūkiai:

- dviprasmybės – žodžiai gali turėti kelias reikšmes;
- konteksto supratimas – sistema turi suprasti viso sakinio, o kartais ir viso dokumento, kontekstą;
- kalbos klaidos – gramatinės klaidos, trumpiniai, šnekamosios kalbos ypatumai;
- kalbos variacijos – sinonimai, antonimai, skirtingi sakinio struktūros variantai

Apibendrinant galima teigti, kad natūralios kalbos apdorojimas (NLP) suteikia galimybę automatizuotai analizuoti ir suprasti žmogaus kalba parašytus dokumentus, įskaitant IT saugos politikas. Tinkamai suprojektuoti NLP procesai leidžia sumažinti žmogiškųjų klaidų riziką, paspartinti dokumentų interpretavimą ir užtikrinti didesnį rezultatų nuoseklumą. Nepaisant iššūkių, susijusių su kalbos dviprasmybėmis ar struktūros netikslumais, NLP taikymas sudaro tvirtą pagrindą tolimesniam politikų automatizavimo sprendimų vystymui.

1.3.4. NLP modeliai

NLP modeliai yra pagrindinė priemonė, leidžianti automatizuotai apdoroti natūralios kalbos tekstą. Skirtingi modeliai pasižymi skirtingu sudėtingumu ir taikymo galimybėmis. Šiame skyrelyje bus aptarti ir palyginti esami NLP modeliai.

„Rule-based“ (taisyklėmis grįstas) modelis. Tai vienas iš pirmųjų NLP modelių, kuris paremtas rankiniu būdu sukurtomis gramatikos, sintaksės ar semantikos taisyklėmis. Veikia pagal iš anksto apibrėžtus šablonus: pavyzdžiui, „jei žodis X yra prieš žodį Y, tai jų ryšys yra toks“. Toks modelis nėra mokomas iš duomenų, bet remiasi žmogaus sukurtais šablonais ir rankinėmis instrukcijomis. Ypač sėkmingai šis modelis veikia finansinių sukčiavimų aptikimui dėl aiškių skaičiavimų ir taisyklių. Pagal tokį atliktą tyrimą, modelis pasiekė 99 % tikslumą eksperimente, kuriame įrodė pranašumą atrandant bankų finansinių sukčiavimo procesus prieš kitus modelius [29].

Privalumai:

- geras tikslumas siauroje srityje (kai tekstas labai standartizuotas);
- lengva paaiškinti, kodėl gautas vienas ar kitas rezultatas.

Trūkumai:

- neefektyvūs, kai kalba įvairi ir sudėtinga (kaip lietuvių kalba);
- sunku prižiūrėti ir išplėsti, reikia nuolat papildyti taisykles;
- senas ir jau retai naudojamas.

Daugiakalbis BERT modelis. BERT (angl. „Bidirectional Encoder Representations from Transformers“) – vienas populiariausių NLP modelių, leidžiantis suprasti žodžių reikšmę kontekste. BERT yra neuroninis transformerių architektūros modelis, išmokytas su daugiau kaip šimto kalbų duomenimis, įskaitant lietuvių kalbą. Modelis naudoja dvipusę kontekstinę informaciją: kiekvieną žodį analizuoja tiek iš ankstesnio, tiek iš vėlesnio teksto konteksto. BERT modeliai geba suprasti žodžių prasmes sakinio kontekste, todėl yra itin geri semantinės analizės užduotims. Atliktame tyrime, buvo nustatyta, kad šlamštinių laiškų aptikimui panaudotas „Google“ įmonės BERT modelis pasiekė 97 % bendrą tikslumą [30]. Tačiau dėl savo universalaus pobūdžio daugiakalbis BERT gali prarasti tikslumą specifinėse lietuviškos kalbos situacijose.

Privalumai:

- gerai supranta bendrinę lietuvių kalbą;
- geba atpažinti kontekstą ir semantiką.

Trūkumai:

- neoptimizuotas konkrečiai lietuvių kalbai – gali suklysti sudėtinguose sakiniuose ar retuose techniniuose terminuose;
- dėl universalumo šiek tiek nukenčia tikslumas specifinėse užduotyse;
- naudoja daug išteklių procesui įvykdyti;
- neturi dažnai naudojamos NLP funkcijos – kalbos dalių žymėjimo.

„SpaCy“ NLP modelis. „SpaCy“ – tai viena populiariausių atvirojo kodo bibliotekų, skirtų greitam ir efektyviam natūralios kalbos apdorojimui. Ji naudoja optimizuotus statistinius modelius ir pažangius algoritmus, kad galėtų atlikti įvairias NLP užduotis, tokias kaip: tikrinių daiktavardžių ir vardinių vienetų atpažinimą, kalbos dalių žymėjimą, priklausomybių analizę, sakinių segmentaciją, teksto klasifikaciją, lematizaciją, morfologinę analizę. Skirtingai nuo didelių neuroninių tinklų, „SpaCy“ buvo kuriama taip, kad būtų ypač greita ir praktiška realaus pasaulio užduotims, todėl ji idealiai tinka, kai reikia didelio greičio ir pakankamai geros tikslumo kombinacijos. „SpaCy“ palaiko 75 kalbas, tarp jų – ir lietuvių kalbą. Dėl šių savybių „SpaCy“ išsiskiria ne tik dideliu greičiu ir efektyvumu, bet ir integracijos galimybėmis su kitomis duomenų apdorojimo sistemomis.

„SpaCy“ lietuvių kalbos modelis – „lt_core_news“, skirtas atlikti pagrindines NLP užduotis – sakinių segmentaciją, žodžių analizę, kalbos dalių žymėjimą bei priklausomybių ištraukimą. Nors lietuvių kalbos modelis dar nėra toks geras kaip anglų ar kitų populiarių kalbų, jis užtikrina bazinį funkcionalumą ir gali būti sėkmingai naudojamas automatizuotam tekstų apdorojimui, ypač kai reikalingas greitas rezultatų gavimas. Tai leidžia efektyviai analizuoti formalią rašytinę lietuvių kalbą ir gauti struktūruotą informaciją apie tekstą. Modelis išmokytas su universaliu priklausomybių korpusu ir yra aktyviai palaikomas bendruomenės.

Privalumai:

- ypač didelis greitis ir efektyvus resursų naudojimas;
- palaiko visas svarbiausias NLP užduotis vienoje bibliotekoje;
- pritaikytas realių projektų vystymui – lengva integruoti į produktus;
- galimybė plėsti ir adaptuoti modelius individualiems poreikiams;
- specialiai apmokytas lietuvių kalbai su naujienų ir medijų tekstais.

Trūkumai:

- lietuvių kalbos modelis galėtų būti tikslesnis, ypač sudėtingesnėse sintaksinėse konstrukcijose;
- ribotas gebėjimas atlikti gilią semantinę analizę;
- sudėtingiau gali apdoroti laisvos struktūros ar šnekamosios kalbos tekstus.

Šie aptarti modeliai yra palyginami 1.3 lentelėje. Matomas didelis „SpaCy“ pranašumas lietuvių kalbos kokybėje bei integravimo ir naudojimo programoje paprastumas. Galima įžvelgti, kad „Rule-based“ modelis atsilieka nuo kitų modelių dėl savo lankstumo stokos, tačiau tai normalu, nes modelis yra senesnis už kitus. BERT modelis tenkina daugumą savybių, tačiau dėl savo greičio ir didelio resursų naudojimo bei sudėtingumo, praktikoje tampa rečiau naudojamas.

1.3 lentelė. NLP modelių palyginimas

Savybė	**„Rule-based“ modelis**	**BERT**	**„SpaCy“**
Greitis	Vidutinis	Mažas	Didelis
Resursų naudojimas	Mažas	Didelis	Mažas
Tikslumas	Didelis	Didelis	Didelis
Lankstumas	Mažas	Didelis	Didelis
Konteksto supratimas	Mažas	Didelis	Vidutinis
Kalbų kiekis	Mažas	Didelis	Vidutinis
Lietuvių k. kokybė	Prasta	Vidutinė	Gera
Integravimas	Sudėtingas	Sudėtingas	Labai lengvas
Apmokymas	Nereikalingas	Būtinas	Nebūtinas

Apibendrinant galima teigti, kad nors egzistuoja įvairūs NLP modeliai, praktiniam lietuvių kalbos IT saugos politikų automatizuotam apdorojimui geriausiai tinka „SpaCy“ „lt_core_news“ modelis. Jis užtikrina balansą tarp analizės tikslumo, greičio ir lengvos integracijos į realius sprendimus.

1.3.5. Automatizuotas IT saugos politikų interpretavimas su NLP

Atsižvelgiant į NLP technologijų gebėjimą apdoroti natūralia kalba aprašytas taisykles, šios technologijos tampa vis svarbesnės ir IT saugumo politikų valdymo kontekste. Dėl galimybės automatizuotai atpažinti, klasifikuoti ir struktūrizuoti tekstinius duomenis, NLP gali būti integruota į modernius

saugos politikų įgyvendinimo modelius. Tai padėtų išvengti jau aptartų žmogiškųjų klaidų riziką, sumažinti analizės laiką. Todėl NLP pritaikymas įmonės IT saugos politikų interpretavime yra ypač aktualus.

NLP galima pritaikyti apdorojant natūralios kalbos tekstą iš organizacijos IT saugos politikos dokumento ir suskaidyti jį į elementus. Tam būtų panaudojamos NLP funkcijos: kalbos dalių žymėjimas, sintaksės ir sakinio dalių žymėjimas, tikrinių daiktavardžių atpažinimas, teksto skaidymas į dalis. Šios funkcijos suskaido visą dokumento tekstą į elementus: sakinius, žodžius, žymas, kalbos dalis ar sakinio dalis. Iš šių identifikuotų elementų, galima naudoti „politika kaip kodas" ideologija ir kurti atitinkamus struktūrinius įrašus ar komandas pagal politikas dokumente.

Nors šiais laikais NLP yra plačiai taikomas įvairiose tekstų analizės srityse, jo taikymas IT saugos politikų automatizuotam interpretavimui dar nėra išsamiai išnagrinėtas mokslinėje literatūroje. Nebuvo rasta nei vieno mokslinio straipsnio, kuriame būtų nagrinėtos galimybės tokio NLP panaudojimo ir pritaikymo organizacijos IT saugos politikoje. Todėl, šioje srityje pastebimas aiškus tyrimų trūkumas (angl. „research gap"), kuris rodo poreikį plėtoti ir tirti specializuotus sprendimus. Tokiems specializuotiems sprendimams ypač praverstų ne tik NLP teksto interpretavimo panaudojimas, bet ir automatizuoti IT saugos valdymo modeliai, kurie sukurtų visą pilnai automatizuotą ir orkestravimo savybių turintį sprendimą. Tai galėtų padėti organizacijoms pastiprinti savo IT saugą, išvengti žmogiškųjų klaidų, pagreitinti procesus ir užtikrinti atitiktį.

Atsižvelgiant į šį poreikį, šiame darbe siūlomas naujas modelis, kuriuo siekiama automatizuoti politikų interpretavimo procesą ir prisidėti prie šios srities vystymo.

1.4. Išvados

Šiame skyriuje buvo išanalizuoti egzistuojantys IT saugos politikų modeliai, jų trūkumai ir automatizavimo poreikis. Taip pat buvo aptarti šiuolaikiniai sprendimai, tokie kaip „Policy-as-Code" ir natūralios kalbos apdorojimo (NLP) technologijos, kurie gali reikšmingai pagerinti politikų interpretavimo ir diegimo procesus. Po atliktos analizės buvo suformuotos šios išvados:

1. Egzistuojantys IT saugos modeliai yra riboti – dauguma klasikinių modelių („Bell-LaPadula", „Biba" ir kt.) apima tik vieną iš CIA triados tikslų ir sunkiai pritaikomi šiuolaikinėse, dinamiškose sistemose.

2. Dauguma šiandienos sprendimų („Ansible", „Intune") nesprendžia pilno automatizavimo – jie palengvina konfigūracijų taikymą, bet nepajėgia automatiškai interpretuoti organizacijos politikos dokumentų.

3. Rankinis politikų interpretavimas yra lėtas ir rizikingas, nes priklauso nuo žmogaus žinių, nuovargio lygio, patirties, todėl žmogiškųjų klaidų rizika išlieka aukšta.

4. Automatizavimo poreikis akivaizdus, nes šiuolaikinės organizacijos reikalauja greito, patikimo ir nuoseklaus saugos politikų taikymo, mažinant žmogiškąjį faktorių.

5. „Politika kaip kodas” ir NLP technologijos suteikia galimybę automatizuoti procesus – PAC leidžia saugos taisykles įgyvendinti kaip kodą, o NLP leidžia interpretuoti tekstinius dokumentus automatiškai.

6. NLP taikymas IT saugos politikų interpretavimui yra inovatyvi, bet dar neištyrinėta kryptis, todėl būtinas tolimesnis šios srities tyrimas ir praktinių modelių kūrimas.

2. Automatizuotas IT saugos politikos valdymo modelis

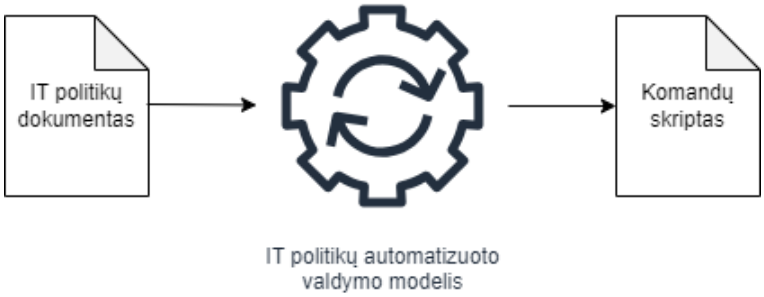
Šiame skyriuje suprojektuojamas automatizuotas IT politikos valdymo modelis, kuris, naudodamas organizacijos politikų dokumentą, sudiegia atitinkamas politikas įmonės įrenginiuose. Siūlomas modelis užtvirtins įrenginių atitiktį organizacijos politikoms ir veiks automatizuotai.

2.1. Tikslas, koncepcija ir uždaviniai

Šio darbo tikslas sukurti modelį, kuris veiktų automatizuotai ir gebėtų nuskaityti IT saugos politikas ir paversti jas į komandas, kurias automatiškai išsiųstų į nurodytus įrenginius. Šis modelis užtikrintų organizacijos politikų įgyvendinimą, naudotų daug mažiau žmogiškųjų išteklių, procesai taptų greitesni, atsekami ir struktūriškai aiškūs.

Siūlomas modelis gali būti naudojamas nepriklausomo dydžio įmonėse, kaip IT saugos politikų diegiklis. Kiekviena organizacija turi tam tikras informacijos ir IT saugos politikas, kurios turi atitikti realią situaciją įmonėse. Deja, ne visada dokumentuose nurodyta informacija atitinka įrenginių, prieigų ir kitų kritinių vietų realią saugumo situaciją. Dažnu atveju saugumo taisykles įgyvendina IT administratoriai, tačiau ne visada galima pasitikėti vienu žmogumi tokiems kritiniams darbams atlikti. Norint išvengti žmogiškojo faktoriaus klaidų, modelis naudos automatizuotus sprendimus.

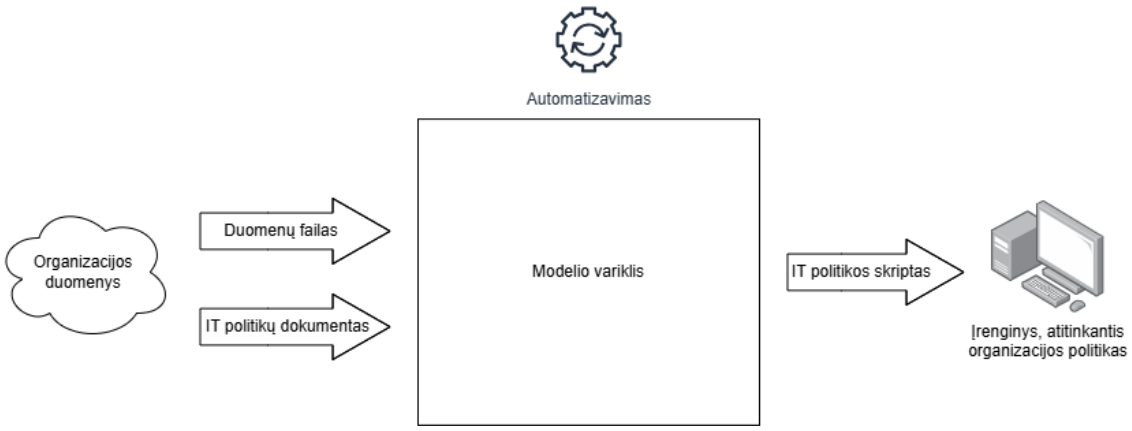
Šiame projekte sukurtas modelis vykdys automatizuotą IT politikų dokumento apdorojimą, informacijos interpretavimą ir pagal tai sugeneruos skriptą, kuris įrenginiui nusiųs komandas, įdiegiančias saugumo konfigūracijas. Modelis užtikrins, kad IT politikų dokumento nurodytos taisyklės ir parametrai yra sudiegti įmonės įrenginiuose. Apibendrintai, šio darbo pagrindiniai elementai bus IT politikos dokumentas, IT politikų automatizuoto valdymo modelis, komandų skriptas – šie elementai pavaizduoti 2.1 paveikslėlyje esančioje blokinėje schemoje.



2.1 pav. Projekto scheminė apžvalga

Pagrindinis siūlomo modelio veikimo principas yra pavadintas „modelio varikliu“. Jame vyks informacijos apdorojimas, skripto automatiniai generavimo procesai. „Modelio variklis“ gaus organizacijos informaciją, kurią sudaro IT politikų dokumentas ir duomenų failas. Duomenų failą sudaro informacija apie įrenginius ir naudotojus. Įvykdžius automatinį skripto generavimą, „modelio variklis“ išves kaip rezultatą IT politikos skriptą, kurį sudarys atitinkamos saugumo komandos įrenginiui.

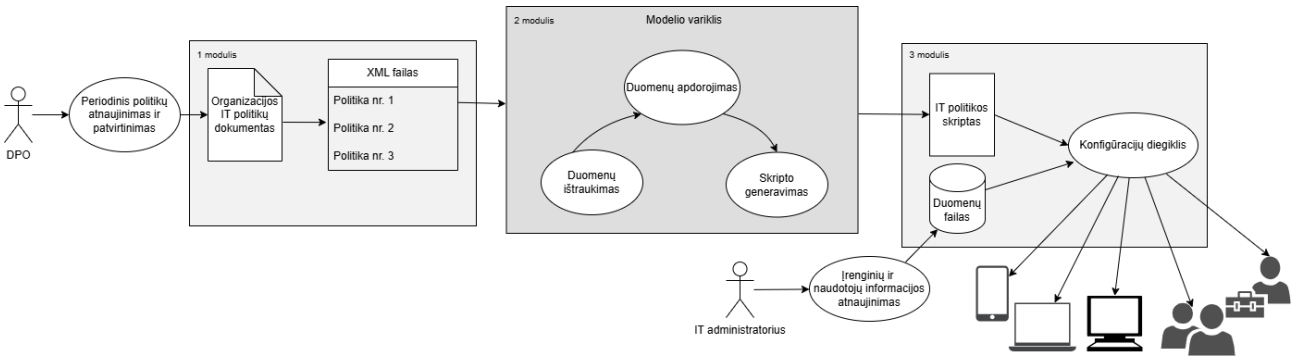
Toliau skriptas bus išsiųstas į atitinkamus įrenginius pagal duomenų failą. 2.2 paveiksliuke atvaizduotas abstraktus projekto kelias link tikslo: įvestys, procesas ir rezultatas.



2.2 pav. IT politikų automatizuoto valdymo modelio tikslas

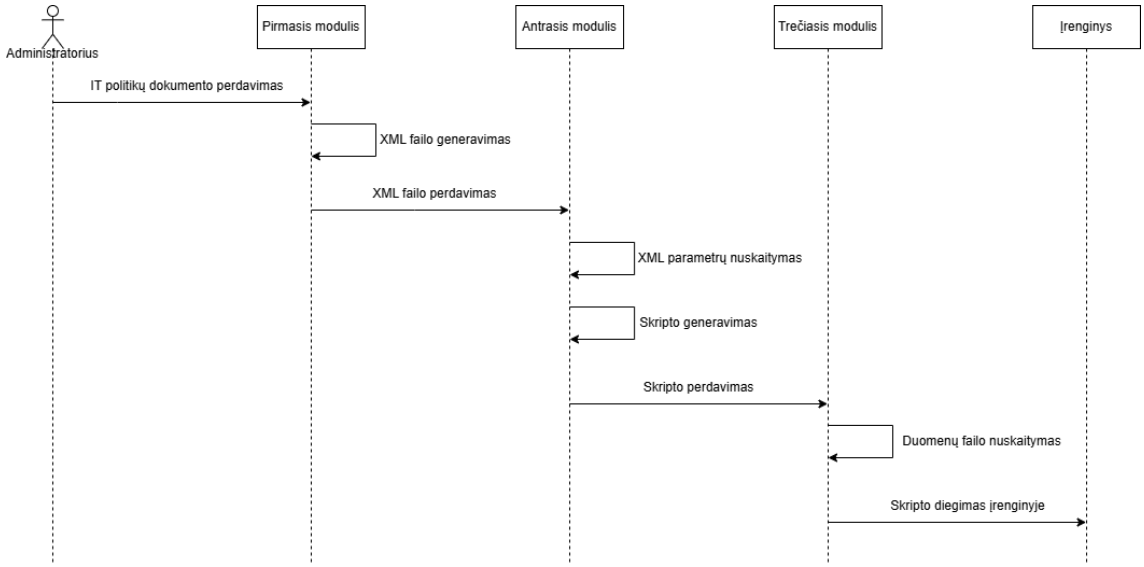
Galima teigti, kad siūlomas modelis atlieka ne tik valdymo procesus, bet ir orkestravimą, kadangi procesai tampa automatizuoti nuo ciklo pradžios iki pabaigos. Modelis orkestruoja visas ciklo sudedamąsias dalis – nuo dokumento iki sukonfigūruotų įrenginių. Galutinis modelio rezultatas turi būti atitinkamai veikiantys įrenginiai pagal nurodytas IT politikas organizacijos dokumente.

Dar daugiau siūlomo modelio detalių pavaizduota 2.3 paveiksle. Šiame paveiksle pavaizduota reali modelio pritaikymo įmonėje schema, kurioje matoma modelio struktūra, sudedamosios dalys ir procesai. Modelį sudaro 3 pagrindiniai moduliai-etapai, kurių kiekvienas atlieka skirtingą paskirtį. Viskas prasideda nuo duomenų apsaugos pareigūno, kuris periodiškai atnaujina ir patvirtina organizacijos IT politikų dokumentą. Dokumentą sudaro IT politikos, kurios nurodo, kokius nustatymus privalo turėti įmonės įrenginiai, kaip turi būti naudojamas įrenginys bei kokius ribojimus turi įgyvendinti. Taip pat nurodytos politikos, kaip valdyti naudotojų saugumą. Turint dokumentą, modelis atlieka savo pirmojo modulio procesą – konvertuoja žmogaus kalba parašytą dokumentą į kompiuteriui įskaitomą XML failą. Toliau pereinama prie antrojo modulio – „modelio variklio“. Jame vyksta pagrindiniai modelio procesai: duomenų bei parametrų ištraukimas iš XML failo ir skripto generavimas. „Modelio variklis“ pagal šablonines komandas sugeneruoja IT saugos politikas atitinkantį komandų skriptą. Trečiasis etapas yra įgyvendinamas organizacijos naudojamais konfigūracijų diegikliais: gautas skriptas yra perduodamas konfigūracijų diegikliui, kuris paleidžia skriptą pagal duomenų faile, kurį pildo ir redaguoja IT administratorius, nurodytus įrenginius ir naudotojus. Galiausiai, skripto komandos pasiekia nurodytus įrenginius ar naudotojus ir jie tampa atitinkantys įmonės IT politikas.



2.3 pav. IT politikų automatizuoto valdymo modelio koncepcija

Patį modelį suvaldyti padeda administratorius. Jis nurodo modeliui įvestis, tačiau toliau modelis dirba savarankiškai. Aiškiau suprasti administratoriaus indelį ir modelio užduotis galima peržiūrėjus 2.4 paveiksle nurodytą sekos diagramą. Diagrama atvaizduoja procesus, kas vyksta kiekviename iš modulių ir kokias įvestis bei išvestis gauna modelis kiekviename etape. Pagrindiniai procesai vyksta atskiruose modeliuose, kadangi tai padeda struktūrizuoti valdymą. Tarp modelių vyksta objektų perdavimai, pavyzdžiui XML failas, skriptas. Galutinis rezultatas, kuris pasiekiamas, atlikus visus veiksmus, yra sudiegtas skriptas įrenginyje.



2.4 pav. Siūlomo modelio sekos diagrama

Šio siūlomo modelio projektavimo pagrindiniai uždaviniai, kurie įgyvendins orkestravimo modelį, yra šie:

1. suprojektuoti IT saugos dokumento, XML ir duomenų failų formatus;
2. sukurti IT politikų dokumento konvertavimo algoritmą į XML failą;
3. sukurti skripto generatorių, kuris iš XML failo parametrų sukurtų atitinkamas komandas kiekvienai politikai;

4. suprojektuoti skripto automatizuoto diegimo įrenginiuose mechanizmą, naudojant orkestravimo įrankį.

2.2. Siūlomo modelio struktūra

Modelį sudaro 3 moduliai, kurie turi skirtingus procesus ir paskirtis. Modulių atlikimas turi eiti eilės tvarka, nes jų rezultatai priklausomi vienas nuo kito. Kiekvienas modulis yra aptartas atskirame skyrelyje, kuriame išaiškinami ir suformuojami kompleksiniai objektai ir procesai. Taip pat pateikiami objektų šablonai, pavyzdžiai ir procesų schemos.

2.2.1. Pirmasis modulis

Pirmąjį modulį sudaro du objektai: IT politikų dokumentas ir XML failas. Taip pat modulyje yra XML failo konvertavimo procesas, kuris IT politikų dokumentą konvertuoja į XML failą.

IT politikų dokumentas

Kiekviena organizacija privalo turėti DPO patvirtintą dokumentą, pagal kokias taisykles organizacija susiplanuoja veikti. Dokumentuoti IT politikas yra ypač svarbu. Tai gali padėti apsaugoti verslo duomenis ir optimizuoti procesų bei pačių darbuotojų efektyvumą.

Gerai sudarytos IT politikos sumažina klaidų pasikartojimą ir kiekį, įgalina darbuotojus ir sustandartizuoja procedūras tarp departamentų bei visoje organizacijoje. Taip pat teisingos įmonės politikos gali užtvirtinti įmonės saugumą, taip atitikti įvairius tarptautinius ar nacionalinius reikalavimus. Tai padės išvengti reglamentų baudų arba kibernetinio saugumo incidentų. Įmonės, turinčios tokį IT politikų dokumentą ir veikiančios pagal jį, gali pretenduoti į saugumo sertifikatų gavimą, pavyzdžiui, ISO27001 – tai būtų įrodymas klientams ir rinkai, kad organizacija veikia sklandžiai, joje duomenys yra saugūs ir patikimai valdomi.

Šio modelio veikimui privaloma naudoti IT politikų dokumentą, kadangi jame yra visos taisyklės, kurios nurodo, kad su jomis įrenginiai bus apsaugoti. Dokumento tekstas yra viena iš įvesčių, todėl nuo jos turi viskas prasidėti. IT politikų dokumentą organizacijose tvirtina duomenų apsaugos pareigūnas, kuris privalo būti kiekvienoje įmonėje, kuri nori atitikti BDAR taisykles. Duomenų apsaugos pareigūnas periodiškai turi peržiūrėti dokumentą ir patvirtinti jį. Patvirtinimas gali būti kas metus, kas puse metų ar kas mėnesį, priklauso nuo įmonės dydžio, pokyčių ir kitų aplinkybių. Patvirtinus dokumentą, dokumentas gali būti naudojamas modelio tolimesniems žingsniams.

Labai svarbu nustatyti vieningą IT politikų dokumento formatą dėl modelio nuskaitymo. Modelis negalės nuskaityti blogo ar netinkamo formato dokumento, todėl kiti modelio etapai neįvyks. Norint perduoti modeliui tinkamą formatą, kurį jis galės perskaityti, reikia atsižvelgti į šias taisykles:

- Dokumentą turi sudaryti skyriai.
- Politikos nurodytos atitinkamuose skyriuose pagal temą.

- Politikos turi būti numeruojamos „x.y.“ formatu, kur „x“ – skyriaus numeris, „y“ – politikos numeris.
- Politikos numerio „y“ numeracija pradedama nuo skyriaus pradžios.
- Politikas turi sudaryti aiškūs sakiniai, kuriuose veiksnys yra objektas (ugniasienė, antivirusinė), tarinys yra veiksmas (išjungti, įjungti, keisti).
- Objektas turi būti daiktavardis arba žodis, išskirtas kabutėmis.
- Veiksmas gali būti veiksmažodis arba esamojo laiko neveikiamosios rūšies dalyvis.
- Politikoje gali būti nurodyti parametrai ir vienetai.
- Jeigu politika nurodo parametrus, parametras turi būti išreikštas skaičiais arba simboline eilute laužtiniuose skliaustuose.
- Jeigu politikoje parametras turi vienetus, jie turi būti parašyti pilnais žodžiais (minutės, valandos, bandymai).

Šios taisyklės nurodo, ką modelis yra įgalus perskaityti ir suprasti. Vadovaujantis šiomis taisyklėmis galima sudaryti tvarkingą IT politikų dokumentą su punktais, taisyklėmis ir struktūra. Organizacijos IT politikų dokumentas gali atrodyti kaip 2.5 pavyzdyje.

Organizacijos įrenginių valdymo IT saugos politikų dokumentas

1. Prisijungimo į įrenginį valdymas

1.1. Prisijungimui į įrenginį įjungiamas slaptažodis.

1.2. Slaptažodžio struktūra turi būti ne mažiau kaip iš 8 įvairių simbolių, iš kurių privalomai nustatomi didžiųjų ir mažųjų raidžių, skaičių ir simbolių.

1.3. Slaptažodžio galiojimo laikas yra 1 metai.

1.4. Prisijungimui į įrenginį įjungiamas „MFA“.

1.5. Įrenginys yra užblokuojamas po 10 nesėkmingų bandymų ir nustatomas 60 sekundžių užblokavimas.

2. Programų ir failų leidimai

2.1. Įjungtas kenkėjiškų failų ir programų blokavimas.

2.5 pav. Organizacijoje patvirtinto IT saugos politikų dokumento pavyzdys

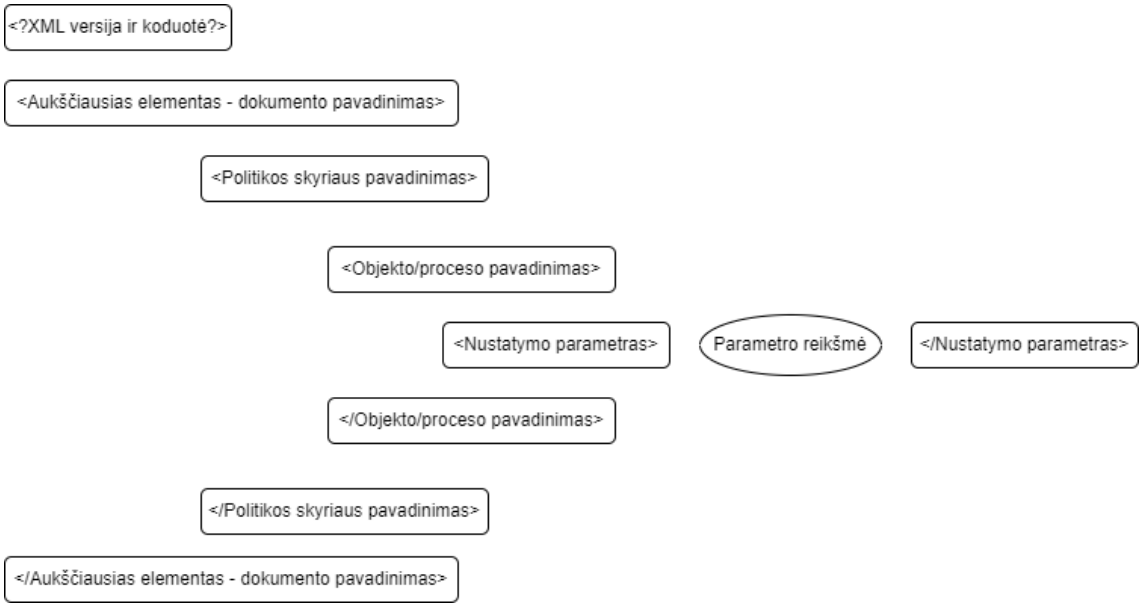
Dokumento sudarymo procesas užtrunka ilgai, kadangi politikas suplanuoti ir užtvirtinti turi duomenų apsaugos pareigūnas. Žinoma, dokumento ilgis priklauso nuo organizacijos dydžio, tačiau, kuo daugiau politikų bus nurodyta dokumente, tuo saugiau organizacija gali veikti. Politikas galima rašyti pagal tarptautines rekomendacijas, pavyzdžiui, NIST. Kai dokumentas paruoštas, jį nuskaitys pirmasis modulis, konvertuos į XML failą ir persiųs į antrąjį modulį.

XML failas

IT politikų dokumente nurodytos taisyklės ir parametrai turi atsidurti konfigūraciniame faile, kuris yra vadinamas XML failu. Pasirinktas XML formatas, nes šis formatas naudojamas duomenų talpinimui ir persiuntimui. Šis failas bus naudojamas kaip tarpinė stotelė tarp dokumento ir skripto. IT politikų dokumentas yra parašytas žmogiškąja kalba, todėl kompiuteriai jos įskaityti negalės. Tam reikia paversti žmogišką kalbą į kompiuterinę kalbą – XML elementus. Kadangi tiesiogiai ištraukti duomenis ir įdėti duomenis į skriptą yra sudėtinga, tam panaudojama laikina struktūrinė duomenų talpykla.

XML faile talpinami politikų nustatymai ir parametrai. Jie nurodo, kokias funkcijas įjungti ar išjungti, kokius parametrus nustatyti tam tikram nustatymui. XML faile nurodytos visos dokumento nustatytos politikos. Kai administratorius nurodo modeliui įvesties failą, „modelio variklis“ nuskaito atitinkamas dokumento dalis ir sudaro politikas atitinkantį skriptą. Todėl modelis sutaupo skaitymo, rašymo ir konvertavimo greitį bei visada turi pačią naujausią DPO patvirtintą informaciją apie politikas.

Aiškiau suprasti XML formatą galima pažvelgus į 2.6 paveiksle pavaizduotą XML kodo blokinę schemą, kuri pavaizduoja, kaip gali atrodyti XML failo vidus. Jį sudaro elementai: šakninis (tėvinis) elementas (aukščiausiai esantis) ir vaikų elementai, kurie yra po šakniniu elementu. Tėvinis ir vaiko elementas pademonstruoja hierarchinį ryšį tarp politikų ir jų nustatymų. Parametro reikšmė gali būti nustatymo išjungimas ar įjungimas arba tam tikras skaičius ar kitokia reikšmė.



2.6 pav. XML failo pseudo schema

Kaip ir organizacijos IT politikų dokumentui, taip ir XML failui reikia apibrėžti jo formatą ir taisykles. XML failas turi atitikti tokias taisykles:

- Nurodyta koduotė ir versija.
- Elementai turi priklausyti hierarchijai: saugumo politika, skyriaus pavadinimas, objekto pavadinimas, politikos parametrai ir nustatymai.
- Kiekvienas elementas turi būti pavadintas pagal reikšmę: „objektas“, „veiksmas“, „parametras“, „vienetas“.

- Elementas „parametras“ turi būti elemento „vienetas“ tėvinis elementas.
- Skaičiai parametruose turi būti nurodomi ne raidėmis, o skaitmenimis.
- Elementų reikšmės turi būti be skyrybos ženklų, mažosiomis raidėmis.
- „Objektas“ ir „veiksmas“ elementų reikšmės turi būti parašytos Vardininko linksniu ir bendratimi, atitinkamai.

1 išeities kodo pavyzdyje nurodyta, kaip turėtų atrodyti hipotetinė slaptažodžių politika XML formatu. Aukščiausias elementas yra saugumo politika, toliau eina politikos grupės – skyriaus pavadinimas ir tada politikų pavadinimai, kuriuose nurodyti parametrai: „Objektas“, „Veiksmas“, „Parametras“, „Vienetas“. Kiekvienas turi atitinkamą reikšmę, kuri paimta iš politikos. Jeigu politikoje buvo parametras, kartu su juo nurodoma ir vieneto reikšmė. Elementų reikšmės yra nurodomos subendrintos dėl tolimesnio etapo (antrojo modulio) ir jo skaitymo taisyklių.

```
<?xml version="1.0" encoding="UTF-8"?>
<saugumo_politika>
    <prisijungimo_valdymas>
        <slaptazodis>
            <Objektas>slaptazodis</Objektas>
            <Veiksmas>ijungti</Veiksmas>
        </slaptazodis>
        <slaptazodzio_struktura>
            <Objektas>slaptazodis</Objektas>
            <Veiksmas>tureti</Veiksmas>
            <Parametras>8
             <Vienetas>simboliai</Vienetas>
            </Parametras>
        </slaptazodzio_struktura>
        <slaptazodzio_galiojimas>
            <Objektas>slaptazodis</Objektas>
            <Veiksmas>galioti</Veiksmas>
            <Parametras>1
             <Vienetas>metai</Vienetas>
            </Parametras>
        </slaptazodzio_galiojimas>
        <mfa>
            <Objektas>mfa</Objektas>
            <Veiksmas>ijungti</Veiksmas>
        </mfa>
    </prisijungimo_valdymas>
</saugumo_politika>
```

1 kodo fragmentas. XML išvesties failo pavyzdys.

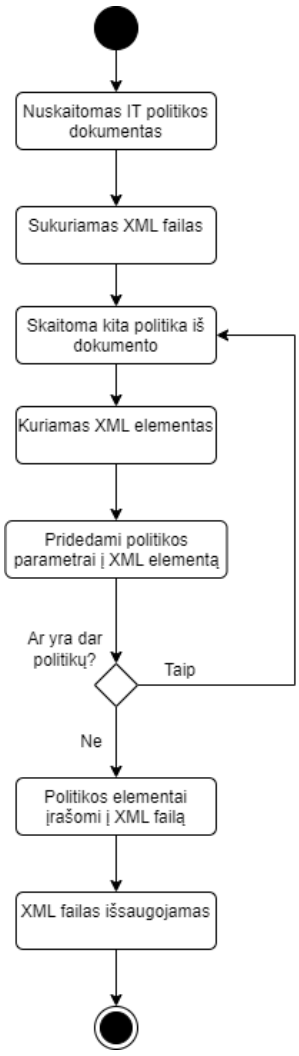
Tokiame formate laikomi parametrai aiškiai atvaizduoja politikų informaciją ir tampa įskaitoma tiek žmogui, tiek kompiuteriui. Turint tokią bendrinę struktūra, galima aiškiai įžiūrėti politikų klaidas (jei

buvo netenkinamos dokumento struktūros taisyklės) ir jas ištaisyti įvesties dokumente.

Duomenų konvertavimas į XML failą

Šis procesas yra vienas svarbiausias šiame automatizuotame modelyje. Konvertavimas iš dokumento į XML failą turi būti vykdomas, naudojant IT politikų dokumento nuskaitymą ir duomenų adaptavimą XML failui. Turint teisingai konvertuotą XML failą, modelis sugebės perskaityti ir sugeneruoti skriptą antrajame modulyje.

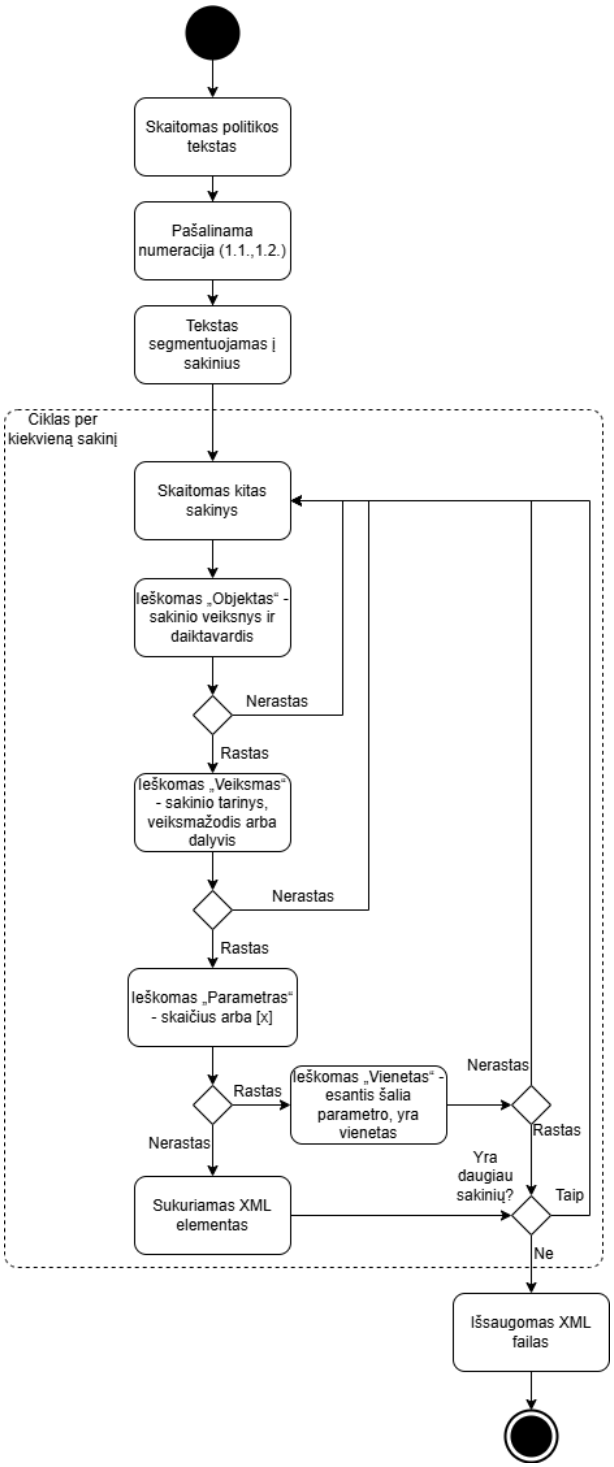
Dokumento konvertavimo algoritmas yra aiškus, nėra kompleksinių funkcijų, kurios apsunkintų arba prailgintų procesą. Algoritmas atvaizduotas veiklos diagramoje, kuri nurodyta 2.7 paveikslėlyje. Procesas prasideda nuo dokumento nuskaitymo, XML failo sukūrimo. Tada vyksta XML elementų kūrimas: kiekviena politika praeina pro pasikartojantį ciklą, kuris tikrina, ar dar yra likusių politikų dokumente. Jeigu politikų nebeliko, ciklas užbaigiamas. Visi elementai surašomi į XML failą. Tada failas išsaugojamas ir jį galima perkelti į antrąjį modulį.



2.7 pav. Dokumento konvertavimo į XML formatą veiklos diagrama

Šiek tiek plačiau išnagrinėjamas XML elementų generavimas 2.8 paveiksle pavaizduotoje veiklos diagramoje. Čia aiškiai matomas algoritmo ciklas, kuriame perskaičius kiekvieną sakinį, algoritmas ieško

sakinyje atitinkamų objektų bei parametrų. Jei kažkurių iš jų neranda, reiškiasi, kad tai ne politikos sakinys ir skaitomas kitas sakinys. Jei buvo rasti visi reikalingi politikos parametrai iš sakinio, tada atitinkamai kiekvienam parametrui yra sukuriamas XML elementas į XML failą. Taip tęsiamas ciklas, kol baigiasi visi sakiniai. Kiekvienas paieškos etapas yra atliekamas pagal nurodytas XML elementų taisykles, kaip turi atrodyti: „Objektas“, „Veiksmas“, „Parametras“, „Vienetas“. Galiausiai, veiksmas baigiasi XML failo sugeneravimu, tai yra konvertavimo pabaiga iš IT saugos politikos dokumento.



2.8 pav. XML elementų generavimo veiklos diagrama

Dokumento konvertavimas gali būti atliktas su įvairiomis programavimo kalbomis, kaip „Python“,

„Java“, „C“ ir panašiomis kalbomis. Vienas geriausių pasirinkimų būtų „Python“ kalba, kadangi ji turi teksto nuskaitymo bibliotekas ir sugeba suskirstyti sakinius į atskirus žodžius daug paprasčiau. Taip galima efektyviai optimizuoti programos veikimą.

Norint dar atidžiau suprasti politikų teksto konvertavimą į XML, galima atsižvelgti į 2 išeities kodą, kuris atvaizduoja konvertavimo į XML failą pseudo kodą. Šis kodas nurodo modelio procesų, konvertavimo ir informacijos surinkimo, įgyvendinimo planą. Iš pradžių, modelis privalo nuskaityti politikų dokumento failo tekstą, tada suskaido tekstą į sakinius. Kai sakiniai jau turimi aibėje, panaudojus ciklinę sąlygą, galima ieškoti reikiamų parametrų sakiniuose. Kai rastos atitiktys, reikšmės ir pavadinimai pridedami į XML failą.

```
START
# Teksto nuskaitymas
tekstas = nuskaityti_teksta()
# Teksto numeracijos pasalinimas
tekstas_be_numeracijos = pasalinti_numeracija(tekstas)
# Isskaidymas i sakinius
sakiniai = isskaidyti_i_sakinius(tekstas)
# Sukuriamas pradinis XML medis
xml_medis = sukurti_xml_medi()

#Sukuriami XML elementai
FOR politika IN sakiniai:
    objektas = ieskoti_objekto(politika)
    veiksmas = ieskoti_veiksmo(politika)
    parametras = ieskoti_parametro(politika)
    vienetas = ieskoti_vieneto(politika)
    xml_elementas = sukurti_elementa(objektas, veiksmas, parametras,
       vienetas)
    prideti_elementa_prie_medzio(xml_medis, xml_elementas)
ENDLOOP

# Issaugojamas XML medis i faila
issaugoti_xml_i_faila(xml_medis, "rezultatas.xml")
END
```

2 kodo fragmentas. IT politikų konvertavimo į XML failą modelio pseudo kodas.

Modelio pirmasis modulis veikia pagal šiuos aprašytus algoritmus ir pseudo kodą: efektyviai ir atitinkamai ištrauks reikalingą informaciją iš skirtingų tipų įvesčių ir sugeneruos aiškų duomenų bloką XML formatu. Galutinis šio proceso rezultatas taps XML failas, kuris perduodamas į antrąjį modulį.

2.2.2. Antrasis modulis

Antrasis modulis yra atsakingas už skripto failo sukūrimą. Šiame modulyje vyksta 3 procesai: politikos struktūrizavimas, komandų generavimas ir skripto formavimas. Į šį modelio etapą yra perduodamas XML failas, o išgaunamas skripto failas, kurį sudaro konfigūracinės politikų komandos. Sugeneruotas skriptas atitinka „politika kaip kodas" principą – kiekviena politika atvaizduojama su komanda.

Politikos struktūrizavimas

Antrasis modulis prasideda nuo XML failo nuskaitymo. „Modelio variklis" priima struktūrizuotą duomenų failą XML formatu ir jį atidaro sistemoje. Šiame faile yra nurodyti visi politikų parametrai, suprantami kompiuteriui. Kiekvienas XML elementas yra skenuojamas iš eilės, nuskaitoma kiekviena elemento reikšmė pagal XML hierarchiją:

1. politikos grupė;
2. objektas;
3. veiksmas;
4. parametras;
5. vienetas.

Kai sistema perskaito duomenis, pradedamas vidinės struktūros užpildymas, tai yra, XML reikšmių konvertavimas į loginį vienetą. Visi nuskaityti duomenys yra perkeliami į programos vidinę struktūrą (programavimo kalbose tai gali būti „Python" žodynas, sąrašas, objektų klasė). Tai pagreitina paiešką ir patogiai susieja objektus. Kiekvienas nuskaitytas XML elementas yra prilyginamas programiniam kintamajam, kuris turi pavadinimą ir reikšmę. Visi kintamieji yra grupuojami į savo politikos grupę, nes politiką sudaro 2 arba 4 kintamieji: „Objektas", „Veiksmas" arba „Objektas", „Veiksmas", „Parametras", „Vienetas".

Sistemoje kiekviena politika atrodo kaip duomenų blokas, kurį sudaro kintamieji ir jų reikšmės. Pavyzdžiui, „Python" turi duomenų struktūrą, kuri yra vadinama žodynu (angl. „Python dictionary"). Šiame žodyne arba duomenų bloke yra pateikiamos visos reikšmės apie politiką. Kaip pavyzdį, galima panagrinėti 3 kodo fragmentą, kuriame pavaizduota IT saugos politika XML formato duomenų bloke. Matomi XML elementai ir specifinė kodo sintaksė.

```
<slaptazodis_politika>
  <Objektas>slaptazodis</Objektas>
  <Veiksmas>nustatyti</Veiksmas>
  <Parametras>
    8
    <Vienetas>simbolis</Vienetas>
  </Parametras>
</slaptazodis_politika>
```

3 kodo fragmentas. IT saugos politikos pavyzdys, atvaizduotas XML formato duomenų bloke

Po XML failo nuskaitymo, duomenys yra išsaugojami į programinę struktūrą – duomenų bloką. 4 kodo fragmente pavaizduota, kaip modelis savo atmintyje išsisaugo iš XML nuskaitytą informaciją apie politiką. Politikos parametrai yra sugrupuojami į bloką. Toliau šis duomenų blokas yra naudojamas susiejimui su veiksmu ir komandų generavimui.

```
politika = {
  "Objektas": "slaptazodis",
  "Veiksmas": "nustatyti",
  "Parametras": "8",
  "Vienetas": "simbolis"
}
```

4 kodo fragmentas. Politikos reikšmės, atvaizduotos struktūrizuotame duomenų bloke.

Komandų generavimas

Šiame etape nuskaityti duomenų blokai yra siejami su realiomis komandų struktūromis. Šio etapo tikslas yra transformuoti politikos parametrus į jų techninius atitikmenis, naudojant iš anksto apibrėžtą žinių bazę – veiksmų žemėlapį. Naudojant bazinių veiksmų žemėlapį, yra konstruojamos komandos kiekvienai politikos duomenų blokui.

Modelis šiame etape intensyviai naudojasi veiksmų žemėlapiu, kuriame yra nurodytos bazinės komandos. Tokios komandos yra tiesiog bazinės komandos be jokių parametrų, pavyzdžiui:

- *sudo passwd -l –minlength {parametras}*;
- *sudo ufw deny from {parametras}*;
- *sudo chage –maxdays {parametras} root*.

Kiekviena komanda, esanti veiksmų žemėlapyje, turi po du atpažinimo šablonus. Šablonai – tai atitikmens taisyklė ar sąlyga, kurią politikos reikšmė turi atitikti, norint susieti politiką su komanda. Tai reiškia, kad kiekvienas politikos ir veiksmo (komandos) susiejimas turi praeiti pro du atpažinimo lygmenis:

1. objekto atpažinimo lygmuo – kuris identifikuoja, apie ką kalbama (slaptažodis, ugniasienė, naudotojas ir t.t.);
2. veiksmo atpažinimo lygmuo – kuris interpretuoja, ką su tuo objektu reikia daryti (įjungti, išjungti, nustatyti, apriboti ir t.t.).

Šie atpažinimo lygiai, kiekvienai komandai turi skirtingus šablonus (atitikmens taisykles). Šie šablonai turi turėti tą patį tikslą kaip ir komanda. Pavyzdžiui: politika „ugniasienė blokuoja 192.168.1.45 adresą“ turi tokius šablonus:

- objektas – „ugniasienė“;
- veiksmas – „blokuoti“.

2.1 lentelėje pavaizduota, kokie šablonai yra nurodyti paminėtai komandai, kuri blokuoja ugniasienėje atitinkamą IP adresą. Taigi, patikrinus 2.1 lentelę, galima teigti, kad aptarta politika tenkintų objekto ir veiksmo šablonus ir būtų susieta su komanda *sudo ufw deny from {parametras}*. Objekto ir veiksmo šablonuose gali būti kelios reikšmės, kadangi yra galimi sinonimai.

2.1 lentelė. Veiksmų žemėlapio loginė struktūra

Objekto šablonas	**Veiksmo šablonas**	**Bazinė komanda**
`slaptažodis`	`turėti, reikalauti`	`sudo passwd -l --minlength {parametras}`
`ugniasienė`	`blokuoti`	`sudo ufw deny from {parametras}`
`slaptažodis`	`galioti`	`sudo chage --maxdays {parametras} root`
`atnaujinimas`	`įjungti`	`sudo apt install unattended-upgrades`

Kai politikos objektas tenkina veiksmų žemėlapio objekto šabloną, tada tikrinamas politikos veiksmas su veiksmo šablonu. Jei politikos objektas tenkino objekto šabloną, bet politikos veiksmas netenkino veiksmo šablono, tada ieškomas kitas veiksmo šablonas, kuris tenkintų politikos veiksmą. Veiksmų šablonų gali būti daugiau nei vienas, tačiau kiekvienas šablonas turi turėti atitinkamą bazinę komandą. Jei politika netenkino nei vieno šablono, politika yra praleidžiama, taip išvengiama klaidingų komandų generavimo.

Po to, kai modeliui pavyksta susieti politiką su komanda, pradedamas parametrų užpildymas. Ne visos komandos turi pildomus parametrus – jei komanda neturi parametrų, tokia komanda ir priskiriama politikai. Tačiau, jei yra trūkstami parametrai, pavyzdžiui, IP adresas ar slaptažodžio amžius, kurį reikia užblokuoti, vykdomas komandų formavimas ir parametrų įterpimas.

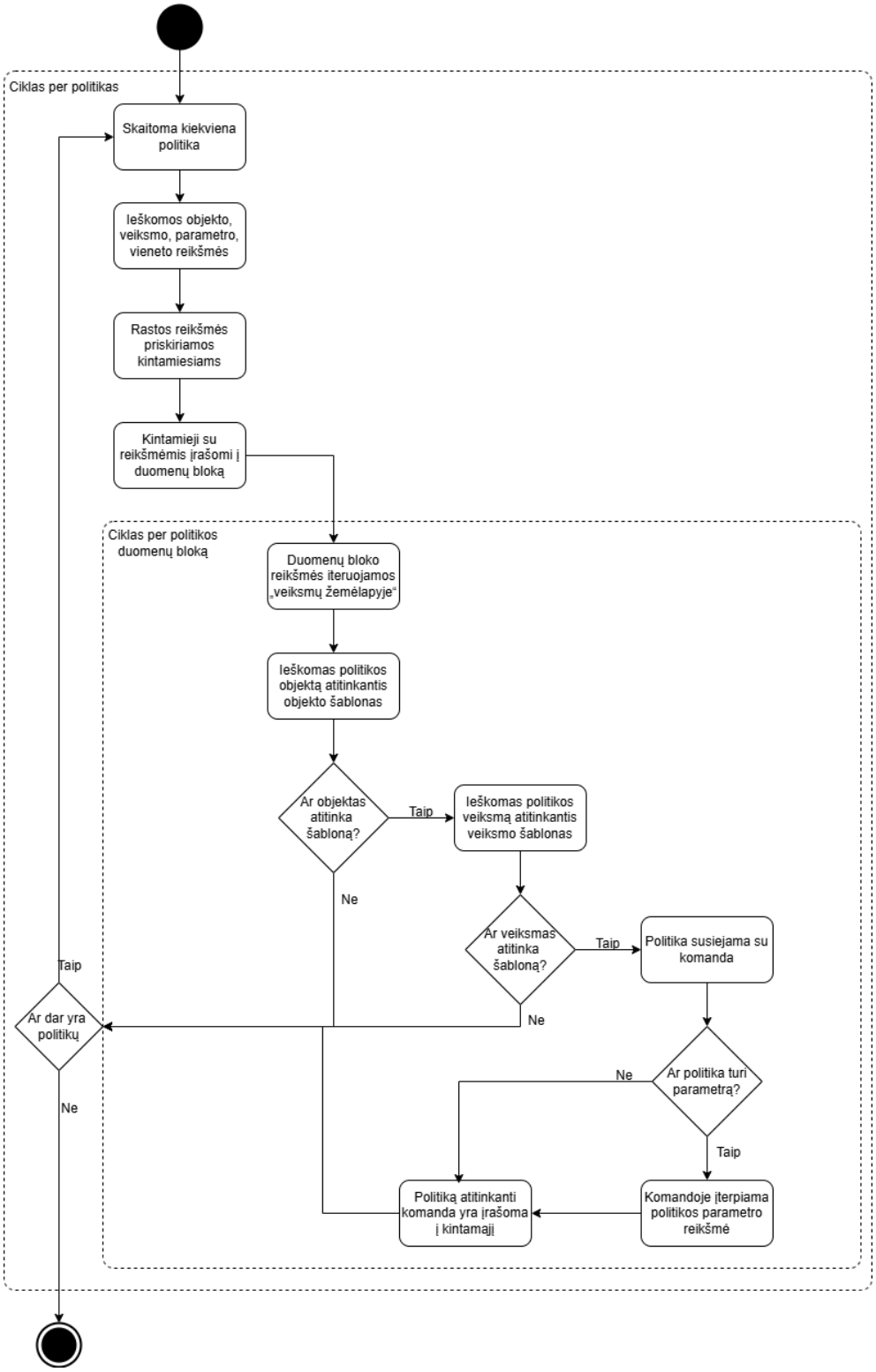
Modelyje yra naudojami du parametrai:

1. parametras – skaitmenys (pvz., 8, 192.168.1.45);
2. vienetas – parametro vienetas (pvz., sekundės, bandymai).

Į komandas įterpiamas tik parametras, tačiau vienetas yra reikšmingas parametro skaičiui. Pavyzdžiui, jei bazinė komanda buvo *sudo faillog -l {parametras}* , tada į parametro vietą yra įterpiamas skaičius iš politikos parametro. Tačiau modelis tikrina vieneto reikšmę, jei tai yra sekundės, jokių papildomų skaičiavimų neatliekama. Jei vienetas yra minutės, tada parametras yra padauginamas 60 kartų, kad būtų išreikštas sekundėmis. Šis veiksmas yra paskutinis komandos generavimo etapas ir jis atliekamas tame pačiame veiksmų žemėlapyje.

Visas komandų generavimo procesas yra pavaizduotas 2.9 paveiksle, komandų generavimo veiklos diagramoje. Matomi du ciklai, kurie skaito kiekvieną politiką ir kiekvienus politikos duomenų bloko parametrus. Jei modelis neranda objekto ar veiksmo, tada išeinama iš duomenų bloko ciklo ir ieškoma kita politika, o esama politika praleidžiama. Jei parametrai buvo rasti, jie įterpiami į komandą, bet jei

politika neturėjo parametrų, paliekama bazinė komanda įrašymui į sistemos atmintį. Toliau komandos bus formuojamos į skriptą.



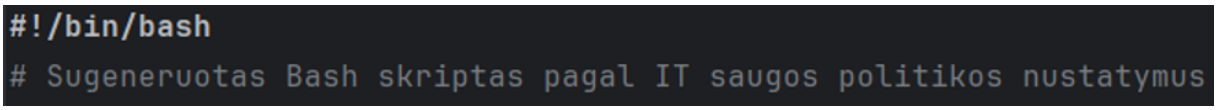
2.9 pav. Komandų generavimo veiklos diagrama

Skripto formavimas

Šiame etape antrojo modulio rezultatas tampa realiai įvykdomų veiksmų sąrašu. Tai tarsi išvesties materializavimas: viskas, kas buvo modelio interpretuota ir generuota, dabar suformuojama į sistemai

suprantamą instrukcijų rinkinį. Šio etapo tikslas yra suformuoti vientisą „Bash“ skriptą, kuriame iš eilės išdėstytos visos interpretavimo procesu gautos komandos, atitinkančios IT saugos politikas.

Visų pirma, sistema sukuria „.sh“ formato failą, kurį atidaro įrašymui. Į failą įrašomas „Bash“ skripto komentaras, nurodantis, kad tai bus skriptas ir kas jame parašyta. Tai gali būti 2.10 paveiksle pavaizduotas kodo komentaras.



2.10 pav. Pavyzdinis skripto komentaras

Kiekviena gauta komanda, kuri buvo sugeneruota generavimo etape, įrašoma į naują eilutę skripte. Taip išlaikoma komandų eilės tvarka ir suteikia patogumo administratoriui, kuris gali peržiūrėti, ar komandos tinkamai buvo sugeneruotos, suskaičiuoti, ar nebuvo praleistų komandų.

Skripto formavimo procesas pavaizduotas 5 kodo fragmente. Pseudo kodas įvykdo pagrindines šio etapo funkcijas: sukurti „.sh“ failą, įterpti antraštę ir komentarą, visų komandų surašymas į naują eilutę ir failo uždarymas.

```
FUNCTION formuoti_bash_skripta(komandu_sarasas, failo_pavadinimas):
    # Sukuriamas failo pradzios tekstas
    skripto_eilutes = []
    skripto_eilutes.append("#!/bin/bash")
    skripto_eilutes.append("# Sugeneruotas pagal IT saugos politikas")

    # Iteracija per visas sugeneruotas komandas
    FOR komanda IN komandu_sarasas:
        IF komanda IS NOT NULL:
            skripto_eilutes.append(komanda)

    # Irasomos skripto eilutes i faila
    failas = atidaryti_faila(failo_pavadinimas)
    FOR eilute IN skripto_eilutes:
        failas.write(eilute + "\n")
    failas.close()
```

5 kodo fragmentas. Skripto generavimas panaudojant XML failą.

Taip sukuriamas pilnai veikiantis, politikas atitinkantis, komandų rinkinys viename faile. Komandų generatorius, vadovaujantis „politika kaip kodas” principu, sukūrė „Bash” tipo failą. Norint failą paleisti esamoje sistemoje, reikia failui pridėti paleidimo teises. Apie patį paleidimą kitose sistemose aptariama trečiajame modulyje.

2.2.3. Trečiasis modulis

Šiame modulyje yra atliekamas galutinis modelio žingsnis – IT saugos politikų įgyvendinimas įmonės įrenginiuose. Šiuo moduliu yra užbaigiamas visas automatizuoto IT saugos politikų valdymo ciklas – nuo dokumento iki realios įrenginio būsenos pakeitimo. Šio modelio tikslas yra užtikrinti, kad sugeneruotos politikos taptų realiomis konfigūracijomis įrenginiuose.

Centralizuotas įrenginių registras – inventorius

Viena iš svarbiausių šio modulio įvesties dalių yra centralizuotas įrenginių registras – inventorius (duomenų failas). Tai yra organizacijos įrenginių aprašas arba sąrašas, kuriame nurodomi egzistuojantys organizacijos įrenginiai, jų informacija. Modeliui būtina žinoti, kuriems konkretiems įrenginiams turi būti taikomos sugeneruotos saugumo politikos.

Kiekvienas toks įrenginių sąrašas (inventorius) atspindi realią organizacijos infrastruktūrą, kokie įrenginiai, kompiuteriai ar serveriai turi būti sukonfigūruoti pagal įmonės saugos politiką. Kiekvienas įrašas šiame sąraše apibūdina:

1. kaip pasiekti įrenginį;
2. kokie yra įrenginio identifikatoriai (pvz., loginiai vardai, IP).

Toks inventorius padeda tiksliai įgyvendinti politikas, nes politikos diegiamos tik į inventoriuje nurodytus įrenginius. Taip pat šio įrenginių registro naudojimas užtikrina, kad įrenginių kiekio pokyčiai taptų paprastesni šioje automatizuotoje sistemoje, nes galima lengvai pridėti naujus įrenginius arba ištrinti senus. Kadangi inventorius yra centralizuotas duomenų failas, užtenka pakeisti tik jame esančią informaciją ir sistema veiks pagal jį, tai yra – nereikia keisti visos sistemos norint diegti saugos politikas vis kituose įrenginiuose.

Šio inventoriaus pavyzdys yra pavaizduotas 2.2 lentelėje, kuriame matoma, kad įrenginiai yra pasiekiami pagal IP adresus, o įrenginių identifikatoriai yra ID numeris, serijinis numeris, tipas. Tokia struktūra gali būti naudojama inventoriaus faile.

2.2 lentelė. Pavyzdinė centralizuoto įrenginių sąrašo struktūra

ID	**S/N**	**IP**	**Tipas**
PC-01	PF223	192.168.1.5	Kompiuteris
PC-02	PJ2134	192.168.1.10	Kompiuteris
PC-03	PN453	192.168.1.25	Kompiuteris
SR-01	PY563	192.168.1.200	Serveris

Toks inventoriaus duomenų failas kartu su sugeneruotu komandų skriptu iš antrojo modulio yra perduodamas į konfigūracijų diegiklį.

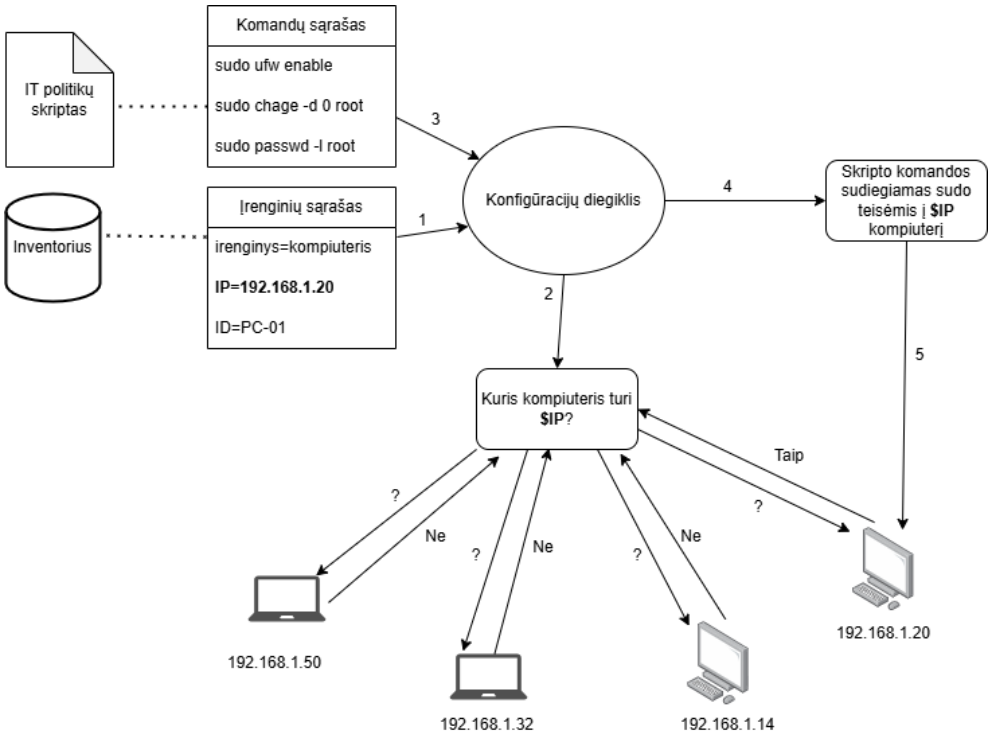
Konfigūracijų diegiklis

Trečiajame modulyje praktinį politikų pritaikymą infrastruktūroje atlieka konfigūracijų diegiklis. Konfigūracijų diegiklis yra esminė orkestravimo grandies dalis, kuri užtikrina perėjimą iš modelio loginio lygmens į fizinį vykdymo lygmenį. Ši dalis yra atsakinga už sugeneruotų komandų įgyvendinimą, sudiegimą nurodytose sistemose. Į konfigūracijų diegiklį yra įvedami du failai:

1. sugeneruotas ir suformuotas komandų „Bash“ skriptas – komandų rinkinys, kurio komandos atitinka kiekvieną IT saugos politiką;
2. inventoriaus failas – centralizuotas įrenginių sąrašas, kuriame nurodyta informacija apie įrenginius, kuriuose turi būti įgyvendintos politikos.

Konfigūracijų diegiklis – tai valdymo įrankis, atliekantis orkestravimo funkcija, kuri be žmogaus įsikišimo parenka įrenginius, perduoda atitinkamas komandas ir vykdo konfigūracijų diegimą bei pateikia rezultato statusą. Administratoriaus paskirtis šiame etape yra tik konfigūracijų diegiklio paleidimas ir inventoriaus informacijos pildymas.

Koncepcinis konfigūracijų diegiklio procesas atvaizduotas 2.11 paveiksle. Pagrindinė kompiuterio identifikacija yra IP adresas, pagal kurį yra identifikuojama, kuriam kompiuteriui bus išsiųstos skripto komandos. Kai įrenginys randamas, užmezgamas ryšys ir išsiunčiamos komandos „sudo” teisėmis.



2.11 pav. Konfigūracijų diegiklio proceso schema

Dažniausiai organizacijos naudoja tam tikrus įrankius, kurie atstoja konfigūracijų diegiklį. Tai gali būti „Ansible“, „Puppet“ įrankiai arba įvairios MDM sistemos. Šios programos leidžia pasiekti įrenginius administratoriaus teisėmis. Modelyje turi būti integruotas toks įrankis, kuris užtikrintų pilną politikų

valdymą ir įrenginiuose. Komandos turi būti siunčiamos per užšifruotą kanalą, kuris gali būti SSH, su privačiais ir viešais raktais.

Kai konfigūracijų diegiklis (automatizuoto valdymo įrankis) atlieka aprašytą diegimo procesą, gaunamas galutinis modelio rezultatas – įrenginiai tampa konfigūruoti pagal organizacijos politiką. Organizacija pasiekia realų informacijos saugumo politikų atitikimą, o ne tik teorinę politikų deklaraciją.

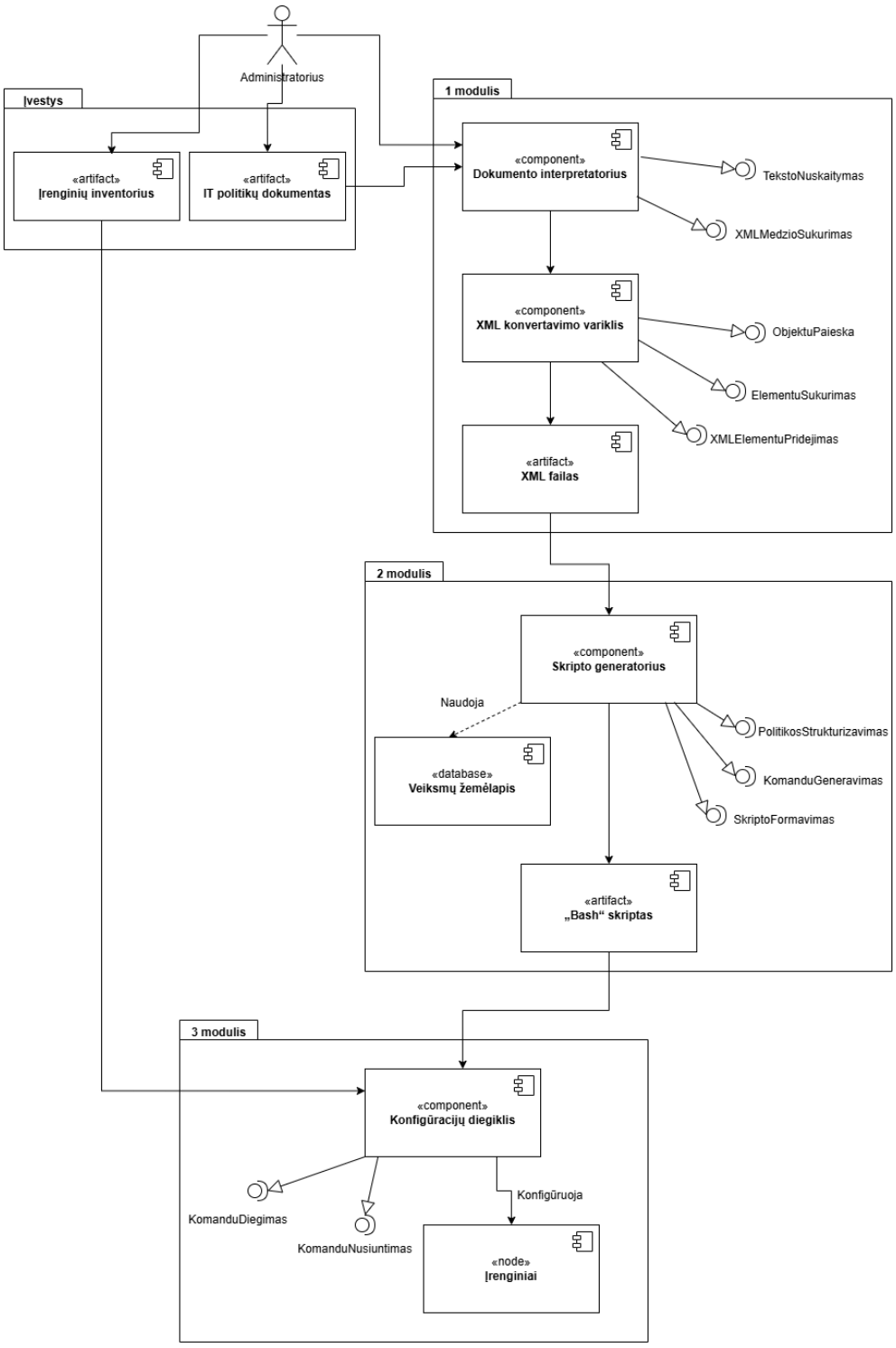
2.3. Apibendrinimas

Suprojektuotas modelis, kurį sudaro trys moduliai, yra skirtas automatizuotam IT politikų interpretavimui, konvertavimui ir įgyvendinimui. Šis modelis veikia kaip pilnas politikų valdymo ciklas – nuo organizacijos dokumento iki pilnai sudiegtų politikų įrenginiuose.

Pagrindiniai duomenų failai, kurie yra perduodami arba sukuriami yra šie:

1. IT saugos politikos dokumentas (tekstinis failas);
2. XML duomenų failas;
3. „Bash“ komandų failas;
4. inventoriaus failas.

Siūlomo modelio architektūrinę struktūrą atspindi 2.12 paveiksle pavaizduota UML komponentų diagrama. Joje pavaizduoti visi esminiai automatizuoto IT saugos politikos valdymo modelio elementai – nuo įvesties artefaktų iki galutinio politikų įgyvendinimo įrenginiuose. Kiekvienas komponentas turi aiškiai apibrėžtas funkcijas, kurios nurodo, kokius veiksmus komponentas atlieka. Tokia struktūra leidžia lengvai keisti ar tobulinti pavienius komponentus neprarandant visos sistemos vientisumo. Administratorius, kaip naudotojas, inicijuoja modelio veikimą pateikdamas dokumentus ir įrenginių informaciją. Modelis automatiškai atlieka analizę, interpretavimą ir politikų pritaikymą, taip užtikrindamas nuoseklų bei klaidoms atsparų procesą.



2.12 pav. Siūlomo modelio UML komponentų diagrama

Kitame skyriuje sukuriamas ir aprašomas siūlomo modelio įgyvendintas prototipas, kuris vadovaujasi šiame skyriuje sukurta architektūra ir veikimo logika.

2.4. Išvados

Apibendrinus siūlomą modelio struktūrą ir veikimo logiką, galima daryti išvadas apie jo funkcionalumą, architektūrą ir praktinį pritaikomumą. Suprojektuoto modelio išvados yra šios:

1. Suprojektuotas modelis automatizuoja visą IT politikų valdymo procesą – nuo dokumento iki įrenginio konfigūracijos sudiegimo.
2. Pirmasis modulis paverčia IT politikų dokumentą į XML failą, kurį sistema gali suprasti ir apdoroti.
3. Antrasis modulis analizuoja XML turinį ir pagal šablonus sugeneruoja tikslias „Bash“ komandas ir skriptą.
4. Trečiasis modulis automatizuotai įdiegia skriptą į įrenginius, naudodamas konfigūracijų diegiklį ir inventorių.
5. Ši trijų modulių struktūra sudaro automatizuotą sistemą, kuri sumažina žmogiškų klaidų riziką, taupo laiką ir užtikrina realų politikų atitikimą organizacijoje.

3. Automatizuoto IT saugos politikos valdymo modelio prototipas

Šiame skyriuje iškeliami funkciniai ir nefunkciniai projekto prototipo reikalavimai, kurie sudėlioja gaires sistemos kūrimui. Taip pat yra pavaizduoti ir aprašyti prototipo pradiniai įvesties failai, paaiškinta jų struktūra. Galiausiai, paaiškintas prototipo modelis, apibrėžtas jo veikimo principas, infrastruktūros dalys ir etapai, kurie įgyvendins norimą rezultatą – automatiškai sudiegiamos IT saugos politikos galiniuose įrenginiuose.

3.1. Sistemos reikalavimai

Prieš kuriant sistemą svarbu apsibrėžti reikalavimus ir žingsnius, kurie padėtų įgyvendinti atitinkamai veikiančią sistemą. Atsižvelgus į funkcinius ir nefunkcinius reikalavimus, sistema įvykdys nustatytą ir apibrėžtą rezultatą.

3.1.1. Funkciniai reikalavimai

1. Sistema turi gebėti priimti IT saugos politikos dokumentus (pvz., tekstinius failus) ir automatiškai konvertuoti juos į XML formatą.
2. Sistema turi patikrinti dokumento struktūrą, kuri turi atitikti organizacijos IT politikos reikalavimus.
3. Sistema turi generuoti tinkamus IT politikos skriptus, atsižvelgiant į iš XML failo gautus duomenis.
4. Sistema turi pritaikyti komandos kintamuosius, tokius kaip nustatymai, saugumo parametrai, įrenginio informacija, pagal ištrauktus duomenis.
5. Sistema turi turėti galimybę automatiškai išsiųsti generuotus skriptus į atitinkamus įrenginius, užtikrinant IT saugos politikos įgyvendinimą.
6. Sistema turi informuoti administratorių apie sėkmingą skripto sugeneravimą.
7. Sistema turi informuoti administratorių apie skriptų diegimo rezultatus (sėkmę arba klaidas).

3.1.2. Nefunkciniai reikalavimai

1. Sistema turi būti be klaidų ir užtikrinanti, kad visos generuojamos IT politikos atitinka organizacijos saugos reikalavimus.
2. Sistema turi būti lanksti, kad ją būtų galima pritaikyti įvairioms IT politikos situacijoms.
3. Skriptų perdavimas ir vykdymas turi būti saugus, užtikrinant, kad konfigūracija negalėtų būti pažeista ar modifikuota.

4. Sistema turi veikti vienodai efektyviai tiek mažose, tiek didelėse organizacijose, nepriklausomai nuo įrenginių ir naudotojų skaičiaus ar padidėjusio IT politikų kiekio.

5. Sistemos dokumentacija turėtų būti aiški ir išsami, kad administratoriui būtų lengva suvokti, kaip atlikti pagrindines užduotis.

3.2. Pradiniai duomenys

Atsižvelgus į projekto reikalavimus, reikia atkreipti dėmesį, kad prototipui reikia turėti įvesties failus. Šių failų vidus turi turėti atitinkamą, taisyklėmis apibrėžtą struktūrą, kad sistema veiktų teisingai ir pagal reikalavimus. Šiame skyrelyje išanalizuojami įvesties failai ir apibrėžtos jų duomenų struktūros ir taisyklės.

3.2.1. Organizacijos IT saugos politikų dokumentas

Vienas iš pagrindinių įvesties failų yra organizacijos IT saugos politikų dokumentas. Šiame dokumente aprašomos įmonės saugumo politikų taisyklės, pagal kurias turi veikti įmonės procesai ir prieigos esamuoju laiku. Tai reiškia, kad įmonė turi apsibrėžti savo taisykles oficialiame dokumente, o tik tada sistema automatiškai įgyvendins aprašytas saugos politikas. Sistemos įvykdytas rezultatas bus toks, kad aprašytos politikos dokumente, esamuoju metu tiksliai atitinka realybėje esančią įmonės situacija. Tai pavers teorines politikas į praktiškai įgyvendintą įmonės infrastruktūrą.

Organizacijos IT saugos politikos dokumentas turi būti užpildytas pagal tam tikrą formuluotę. Sistema, atsižvelgdama į struktūros taisykles, skaitys tekstinį failą ir vers jį į kompiuteriui suprantamą kalbą – XML. Dokumento struktūros taisyklės yra tokios:

1. Dokumentą sudaro skyriai, kuriuose kiekviena politika numeruojama „x.y.“ formatu („x“ – skyriaus numeris, „y“ – politikos numeris).

2. Politikas sudaro aiškūs sakiniai, kuriuose veiksnys yra objektas („ugniasienė“, „antivirusinė“), tarinys yra veiksmas („išjungti“, „įjungti“, „keisti“).

3. Objektas turi būti daiktavardis, vartojamas tinkamu gramatiniu linksniu, atitinkančiu sakinio struktūrą (pvz., „Prisijungimo blokavimas aktyvuojamas“).

4. Veiksmas sakinyje turi būti veiksmažodis (įvairiu laiku) arba dalyvis (veikiamosios arba neveikiamosios rūšies), nurodantis atliktą ar būsimą veiksmą.

5. Nustatymų įjungimui arba išjungimui reikia naudoti atitinkamus dalyvius: „įjungtas“, „išjungtas“.

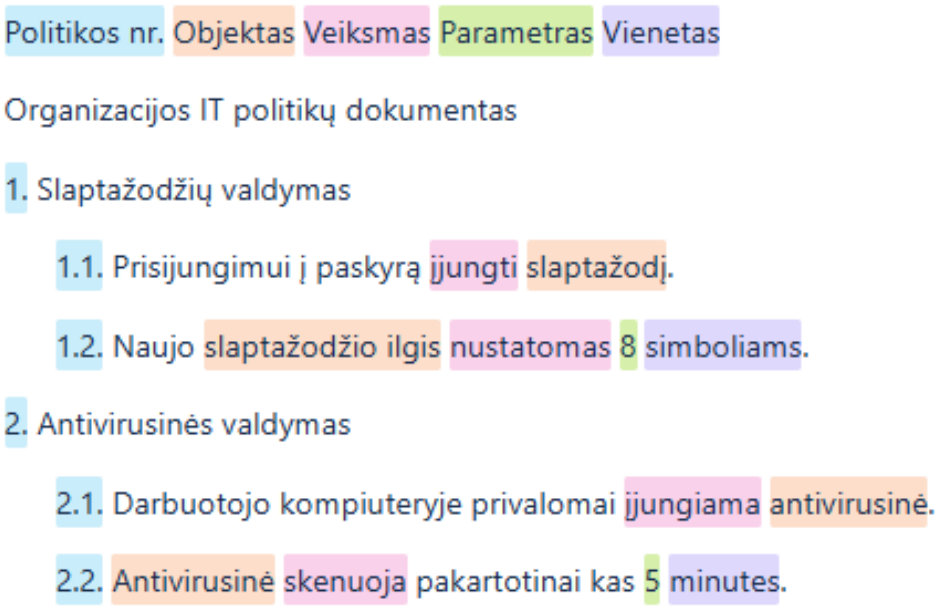
6. Kiekvienas parametras turi būti užrašytas skaitmenimis ir (arba) papildytas vieneto apibrėžimu (pvz., „5 bandymai“, „60 sekundžių“).

7. Skaičiai ir vienetai turi būti pateikti vienareikšmiškai, vengiant neaiškumų (pvz., „10 bandymų“, o ne „10 kartų“).

8. Politikoje reikia naudoti aiškias gramatines konstrukcijas, vengiant sudėtingų ar daugiaprasmių sakinių.

Šios taisyklės apibendrina, kaip turi atrodyti dokumento formatas. Sistema galės perskaityti tokį dokumento formatą ir išgryninti reikšmingus informacijos laukus, išsisaugoti savo atmintyje ir įkelti į XML failą tolimesniems sistemos etapams.

Praktiškesnis taisyklių ir politikų apibrėžimas pavaizduotas 3.1 paveikslėlyje. Matomi apibrėžti laukai, į kuriuos algoritmas kreips dėmesį, skaitant visą tekstą. Todėl būtent šie laukai turi būti tiksliai įvardyti ir panaudoti pagal aukščiau paminėtas taisykles.



3.1 pav. Organizacijos IT politikų dokumento taisyklių pavyzdys

Prototipui pademonstruoti skirtas organizacijos IT saugos dokumentas pateikiamas 1 priede. Šis dokumentas talpinamas projekto kataloge „.txt“ formatu.

3.2.2. IT politikų nustatymų sugeneruotas XML failas

Dar vienas įvesties failas, kurį sukuria pati sistema ir vėliau naudoja kaip įvestį, yra XML failas. Šiame faile yra talpinamos IT saugos politikos kompiuteriui ir žmogui įskaitomu formatu – XML. Kartais administratoriui yra patogiau ir greičiau perskaityti XML formatu paversta organizacijos IT saugos politikų aprašą nei tekstinį failą, taip pat tai padeda išvengti klaidų prototipo veikime.

XML failas yra išvedamas konvertuojant IT saugos politikos dokumentą. Todėl XML failo viduje turi būti nurodytos IT saugos politikos. Šis XML turi svarbią struktūra, kuri yra sukurta, norint teisingai interpretuoti visas politikas ir jų konfigūracijas. XML failo struktūros taisyklės:

1. XML elementai turi būti rašomi hierarchiškai: kiekviena politika yra aukščiausio lygmens ele-

mentas (pvz., *slaptažodis_politika*), o politikų parametrai (pvz., „Objektas“, „Veiksmas“, „Parametras“, „Vienetas“) yra šios politikos subelementai.

2. Politikos elementų pavadinimai turi būti tokie:
    - *objektas* – apibrėžia pagrindinį politikos subjektą;
    - *veiksmas* – apibrėžia politikos veiksmą;
    - *parametras* – apibrėžia skaičių, susijusį su politika;
    - *vienetas* – apibrėžia parametro matavimo vienetą.
3. Elementų reikšmės turi būti:
    - *objektas* – daiktavardis, kuris apibrėžia politikos objektą;
    - *veiksmas* – veiksmažodis, išreikštas bendratimi, kuris apibrėžia veiksmą;
    - *parametras* – skaitinė reikšmė, kuri turi būti papildyta *vienetas* elementu, nurodančiu skaičiaus reikšmės matavimo vienetą.
4. Kiekviena politika turi būti pateikta kaip atskiras XML elementas, kurio pavadinimas sudarytas iš objekto pavadinimo ir priesagos „_politika“ (pvz., *slaptažodis_politika*, *antivirusinė_politika*).
5. Elementų pavadinimai ir reikšmės turi būti logiškai susiję, sudarant aiškias poras:
    - *objektas* ir *veiksmas* turi atitikti sakinio subjektą ir tarinį;
    - *parametras* ir *vienetas* turi tiksliai apibrėžti skaičius ir jų reikšmę.

3.1 paveiksle pavaizduotas tekstinis dokumentas konvertuojamas į pavyzdinį XML failą, pavaizduotą 3.2 paveiksle. Tai yra sutrumpina versija, tačiau aiškiai matoma XML failo norima struktūra.

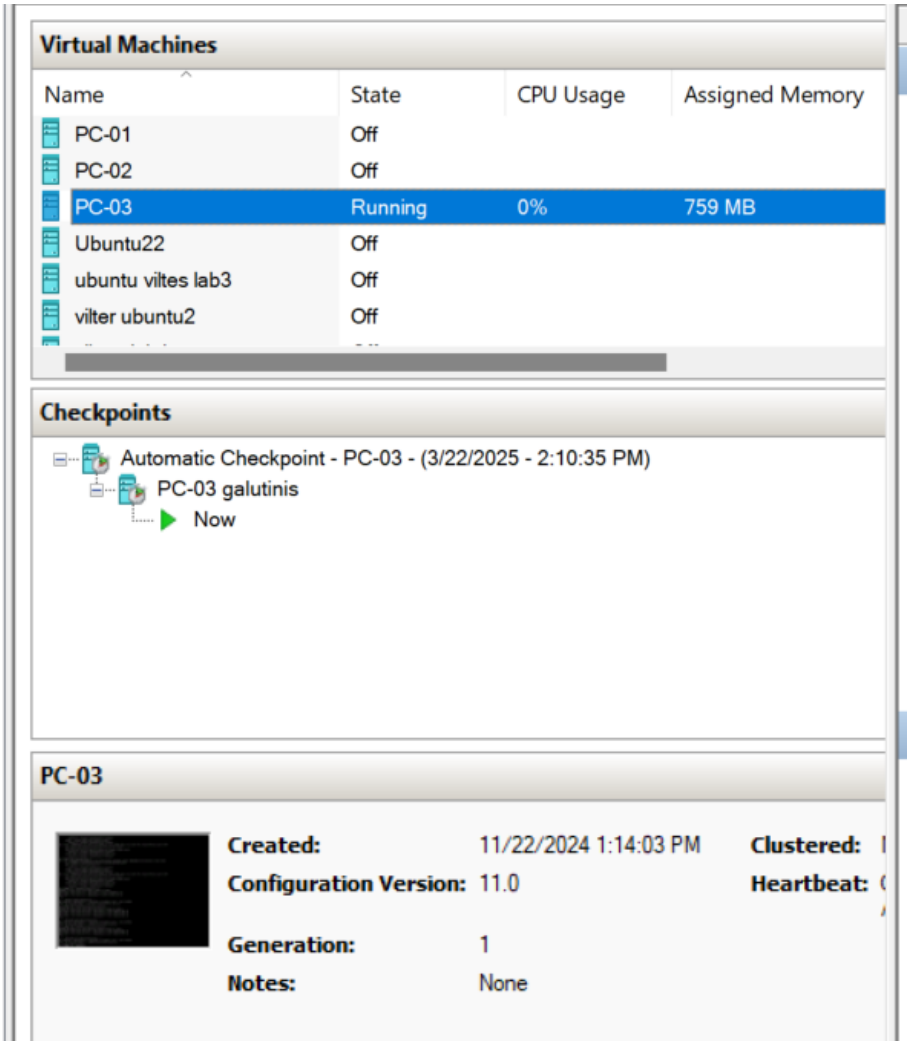
```xml
<?xml version="1.0" ?>
<IT_saugos_politikos>
  <slaptažodis_politika>
    <Objektas>slaptažodis</Objektas>
    <Veiksmas>galioti</Veiksmas>
    <Parametras>
      365
      <Vienetas>diena</Vienetas>
    </Parametras>
  </slaptažodis_politika>
  <prisijungimas_politika>
    <Objektas>paskyra</Objektas>
    <Veiksmas>blokuoti</Veiksmas>
    <Parametras>
      10
      <Vienetas>bandymas</Vienetas>
    </Parametras>
  </prisijungimas_politika>
  <ugniasienė_politika>
    <Objektas>ugniasienė</Objektas>
    <Veiksmas>įjungti</Veiksmas>
</IT_saugos_politikos>
```

3.2 pav. Pavyzdinis XML dokumentas su IT politikomis

3.2.3. Prototipo infrastruktūros aplinka

Norint prototipą įgalinti, būtina turėti įrenginius, į kuriuos bus siunčiamos organizacijos politikos konfigūracinės komandos. Taip pat reikalingas įrenginys, kuriame veikia prototipas ir administratorius gali valdyti jį. Įrenginiai yra fiktyvūs, sukurti testavimo tikslui: išbandyti automatizuotą politikų diegimą.

Šiam tikslui yra sukurti fiktyvūs įrenginiai, kurie yra patalpinti „Hyper-V“ aplinkoje. 3.3 paveiksle pavaizduotos 3 virtualios mašinos, pavadinimais: „PC-01“, „PC-02“, „PC-03“. Siekiant sukurti nepriklausomą ir universaliai veikiantį tinklą tarp kompiuterio ir virtualių mašinų, buvo pasirinktas vidinis virtualus tinklo jungiklis (angl. „internal switch“). Jis leidžia virtualioms mašinoms turėti statinius IP adresus, nepriklausančius nuo fizinio tinklo, ir leidžia prieiti prie interneto per NAT maršrutizavimą.



3.3 pav. „Hyper-V“ aplinkoje patalpinti organizacijos įrenginiai

Kiekviena VM turi „Ubuntu Server 24.04“ operacinę sistemą, po 4GB RAM ir 4 virtualius procesorius. „PC-03“ virtualioje mašinoje yra paleistas veikiantis prototipas, kuris gali pasiekti kitas dvi VM. Šios dvi VM laukia konfigūracinių komandų iš prototipo sistemos.

3.2.4. Inventoriaus failas su įrenginių duomenimis

Projekto sistemai svarbu kaip įvestį turėti įrenginių identifikavimo informaciją. Šiuo atveju naudojami įrenginių IP adresai, per kuriuos siunčiamas komandų skriptas. Įrenginių duomenys yra talpinami */ansible/inventory* faile, kurį „Ansible“ įrankis skaitys kaip įvestį – į kuriuos įrenginius siųsti skriptus. Inventoriaus failas pavaizduotas 3.4 paveiksle.

```
vilte@pc-03:~$ ansible-inventory --list -y
all:
  children:
    pc_group:
      hosts:
        pc-01:
          ansible_host: 192.168.0.50
          ansible_user: ansible_user
        pc-02:
          ansible_host: 192.168.0.51
          ansible_user: ansible_user
vilte@pc-03:~$
```

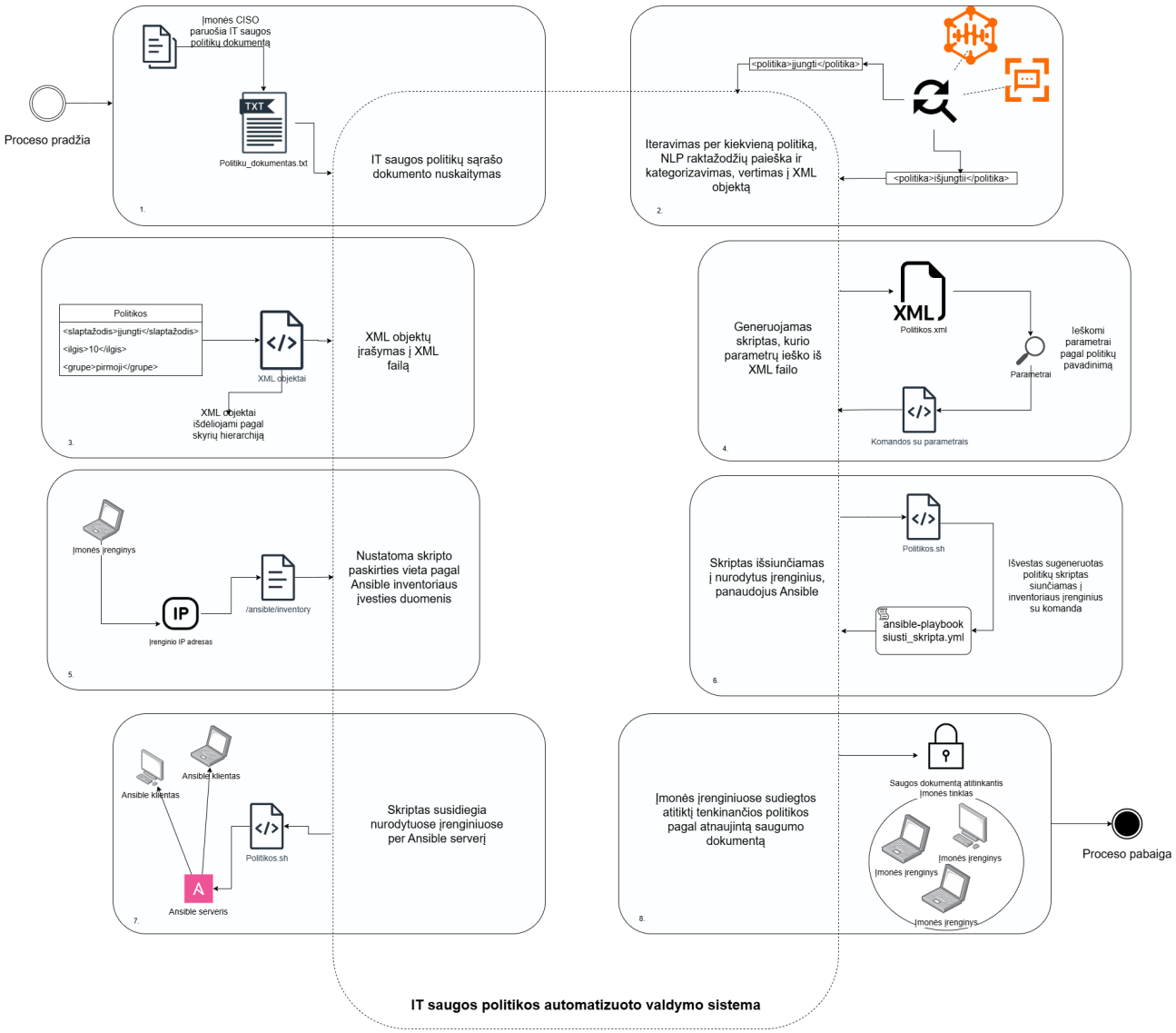
3.4 pav. Inventoriaus failas

Inventoriuje yra nurodoma: įrenginių grupė – tai kompiuterių grupė, įrenginių identifikavimai (pavadinimai), kiekvieno įrenginio IP adresas ir administracinis naudotojas, per kurį yra siunčiamas konfigūracinis skriptas.

3.3. Siūlomo modelio veikimo schema

Šio projekto sistema remiasi pagal antrame skyriuje sudarytą modelio koncepciją. Sistema vykdo daug failų skaitymo, kūrimo ir konvertavimo darbų, duomenų perdavimo bei formatavimo procesų. Visi šie veiksmai, objektai ir duomenys sueina į visumą, kurią galima vadinti siūlomą IT saugos politikos automatizuoto valdymo modeliu. Šio modelio schema yra pavaizduota 3.5 paveiksle.

Prototipo duomenis galima skelti į dvi dalis: įvestis ir išvestis. Kaip buvo minėta, įvestis yra 3 failai: TXT politikų dokumentas, XML failas ir „Ansible“ inventorius. Išvesties pagrindinis failas yra „Bash“ skriptas, kuris yra diegiamas atitinkamuose kompiuteriuose, panaudojus „Ansible“ įrankį.



3.5 pav. Projekto sistemos prototipo schema

Visa sistemos veikimo eiga:

1. sistema nuskaito IT saugos politikų dokumento tekstą;
2. sistema, panaudodama NLP, ieško raktažodžių ir skirsto žodžius į kategorijas;
3. sistema kategorizuotus žodžius konvertuoja į XML elementus;
4. sistema įrašo XML elementus į failą;
5. sistema sugeneruoja „Bash“ komandų skriptą;
6. sistema nuskaito „Ansible“ inventorių;
7. sistema išsiunčia skriptą pagal inventorių į įrenginius;
8. organizacijos politikos atitinkamai susidiegia įrenginiuose.

Šios sistemos eiga realizuoja organizacijos saugos ciklą: organizacijos įrenginiai pilnai atitinka įmonės IT saugos politikų dokumentą, todėl organizacija tampa atitinkanti saugos politikas (angl. „Compliant“).

3.4. Siūlomo modelio prototipo realizacija

Šiam modeliui realizuoti, buvo sukurtas prototipas – automatiškai valdoma politikų diegimo sistema. Sistema sudaryta iš 3 modulių: dokumento konvertavimo modulis, XML konvertavimo modulis, diegimo modulis. Apie juos atskirai aptariama skyreliuose.

3.4.1. Dokumento į XML konvertavimo modulis

Šiame modulyje sistema atlieka dokumento nuskaitymo, teksto konvertavimo į XML ir įrašymo į failą darbus. Modulio schema pavaizduota 3.6 paveiksle. Aiškiai matoma, kad šis sistemos etapas turi įvesti ir išvestį, tai yra: IT politikų tekstinį dokumentą ir politikų XML failą. Procesas prasideda nuo tekstinio failo nuskaitymo, kai failo vidaus tekstas tampa programos kintamuoju. Tada iš teksto pašalinami numeracijos skaičiai, kadangi programai nebereikia politikų punktų. Toliau visi atskirose eilutėse esantys sakiniai yra sujungiami į vieną tekstą, kurį galės skaityti NLP biblioteka „SpaCy“. Ši biblioteka moka skaityti lietuviškus žodžius ir atskirti, kur yra veiksmažodis, veiksnys, skaičius. Šiam moduliui naudojama bibliotekos sudaryta linijinė seka iš specializuotų modelių – „lt_core_news_lg pipeline“, kurį panaudojus, tekstas perskaitomas naudojant žodžio žymas (angl. „tokens“). „lt_core_news_lg pipeline“ turi sukaupusi didelį kiekį lietuviškų naujienų sakinių, žodžių ir jų vektorių, dėl to gali identifikuoti tam tikras lietuviško sakinio ir kalbos dalis.

Ši biblioteka turi specifinius pavadinimus kiekvienai sakino ir kalbos daliai. Šiame prototipe buvo naudojamos tokios žodžių žymos:

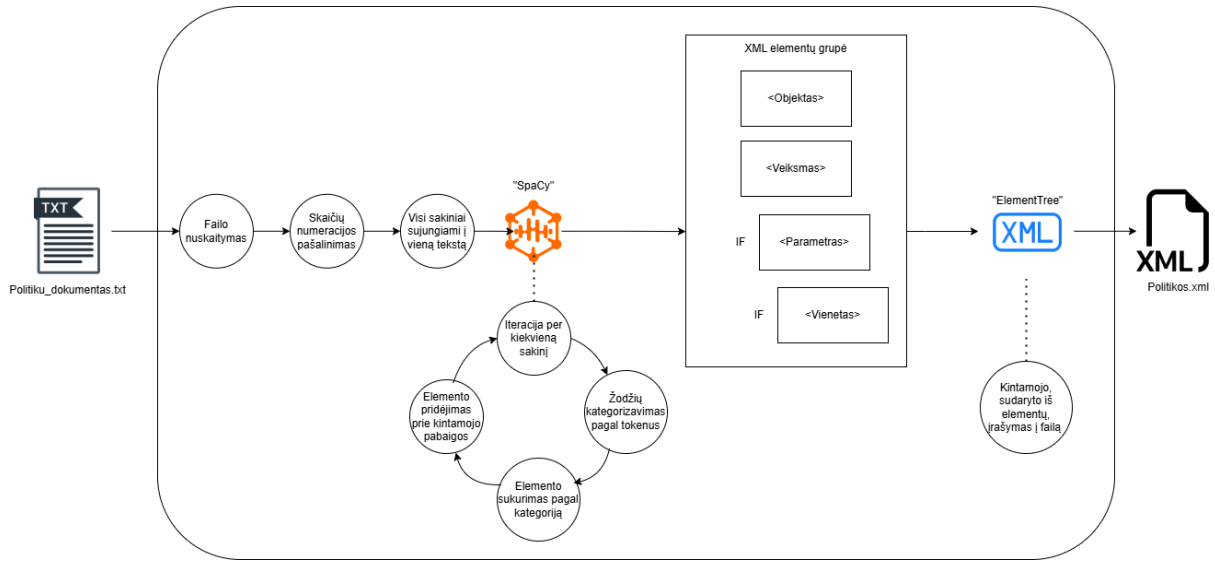
- *nsubj* – veiksnys;
- *dobj* – tiesioginis papildinys;
- *amod* – būdvardinis pažyminys;
- *nmod* – daiktavardinis pažyminys;
- *obl* – netiesioginis papildinys, aplinkybės dalis;
- *advmod* – prieveiksmio aplinkybė;
- *NOUN* – daiktavardis;
- *VERB* – veiksmažodis;
- *AUX* – pagalbinis veiksmažodis;
- *NUM* – skaičius.

Kiekvienas įvesties failo sakinys yra patikrinamas su žodžio žymomis ir nustatomi politikai identifikuoti žodžiai. Šiame prototipe įvesties failo sakiniuose yra ieškomi:

1. *objektas* – tai daiktavardis, kuris sakinyje atlieka veiksmą, yra sakinio veiksnys ir daiktavardis;
2. *veiksmas* – tai veiksmažodis arba pagalbinis veiksmažodis, kuris gali būti įvairaus laiko, gali būti modaliniu pagalbiniu veiksmažodžiu („turi“ daryti) arba formuojantis veiksmo atlikimo būseną („uždarytas“) – tai yra veiksmažodžiai ir dalyviai, identifikuojami kaip tariniai;
3. *parametras* – tai yra skaičius, kuris sakinyje išreikštas skaitmenimis;
4. *vienetas* – tai yra netoli skaičiaus esantys papildiniai: būdvardinis, aplinkybės, prieveiksmio, kurie apibūdina, koks yra skaičius.

Radus vieną iš tokių žodžių pagal žodžių žymas, programa sukuria atitinkamą elementą pagal kategoriją, pridedant rastą reikšmę. Rasti veiksmažodžiai yra lematizuojami – iš žodžio šaknies sukuriama bendratis. Visi sukurti elementai įrašomi prie kintamojo pabaigos, todėl kintamasis po kiekvienos iteracijos pasipildo XML elementais. Svarbu paminėti, kad sakiniuose ne visada gali būti parametrai, tada tokio elemento programa neįrašo. Taip pat ne visada ir vienetas yra reikalingas sakinyje, bet jei yra parametras sakinyje, tada ieškomas ir vienetas.

Po elementų sugeneravimo, programa naudoja „ElementTree“ biblioteką, kuri gražiai suformatuoja elementus pagal hierarchiją ir XML failo struktūrą bei įrašo elementus į failą „Politikos.xml“. Ties šia išvestimi užsibaigia šio modulio darbas.

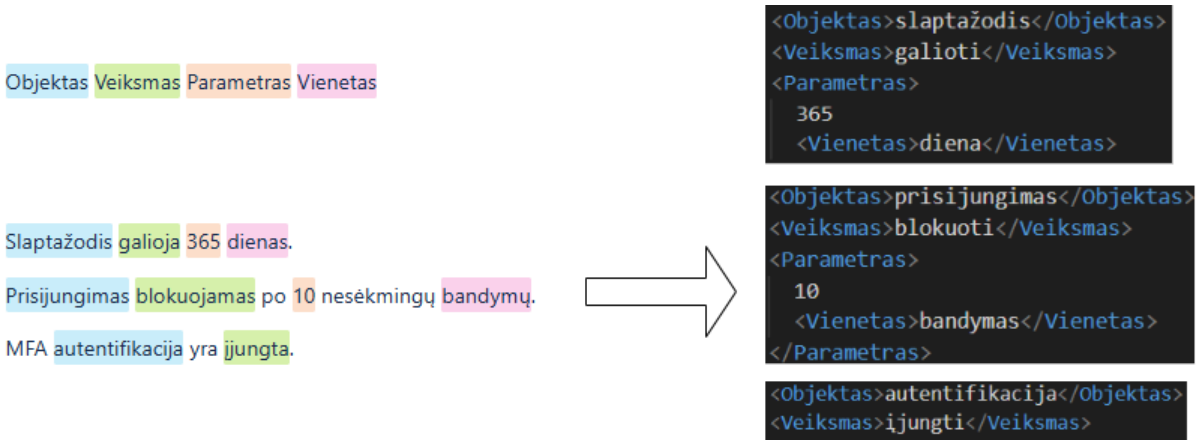


3.6 pav. Dokumento konvertavimo į XML modulio schema

Pagrindinis IT politikų dokumentas yra talpinamas „Politiku_dokumentas.txt“ faile. Šio failo vidus yra nurodytas 1 priede. Taip pat išvesties failas „Politikos.xml“ talpinamas toje pačioje direktorijoje. Šio failo vidus, po sistemos darbo, t. y. įvykus moduliui, yra nurodytas 2 priede.

Panaudojant prieduose pavaizduotus failus, galima panagrinėti šio prototipo pavyzdį. Politikos 1.2., 1.3. ir 1.5. pavaizduotos 3.7 paveiksle, kuriame matoma, kaip sistema identifikuoja, kuriuos žodžius

atrinkti. Kaip buvo paminėta, programa gali iš sakinio išskaityti jam reikalingą objektą, veiksmą ir, jei tokie yra, parametrą ir vienetą. Šias atrinktas reikšmes, sistema įdeda į XML elementą su atitinkamu kategorijos pavadinimu ir įkelia į XML failą. Taip užsibaigia šio modelio procesas, kurio rezultatas yra „Politikos.xml“ failas.



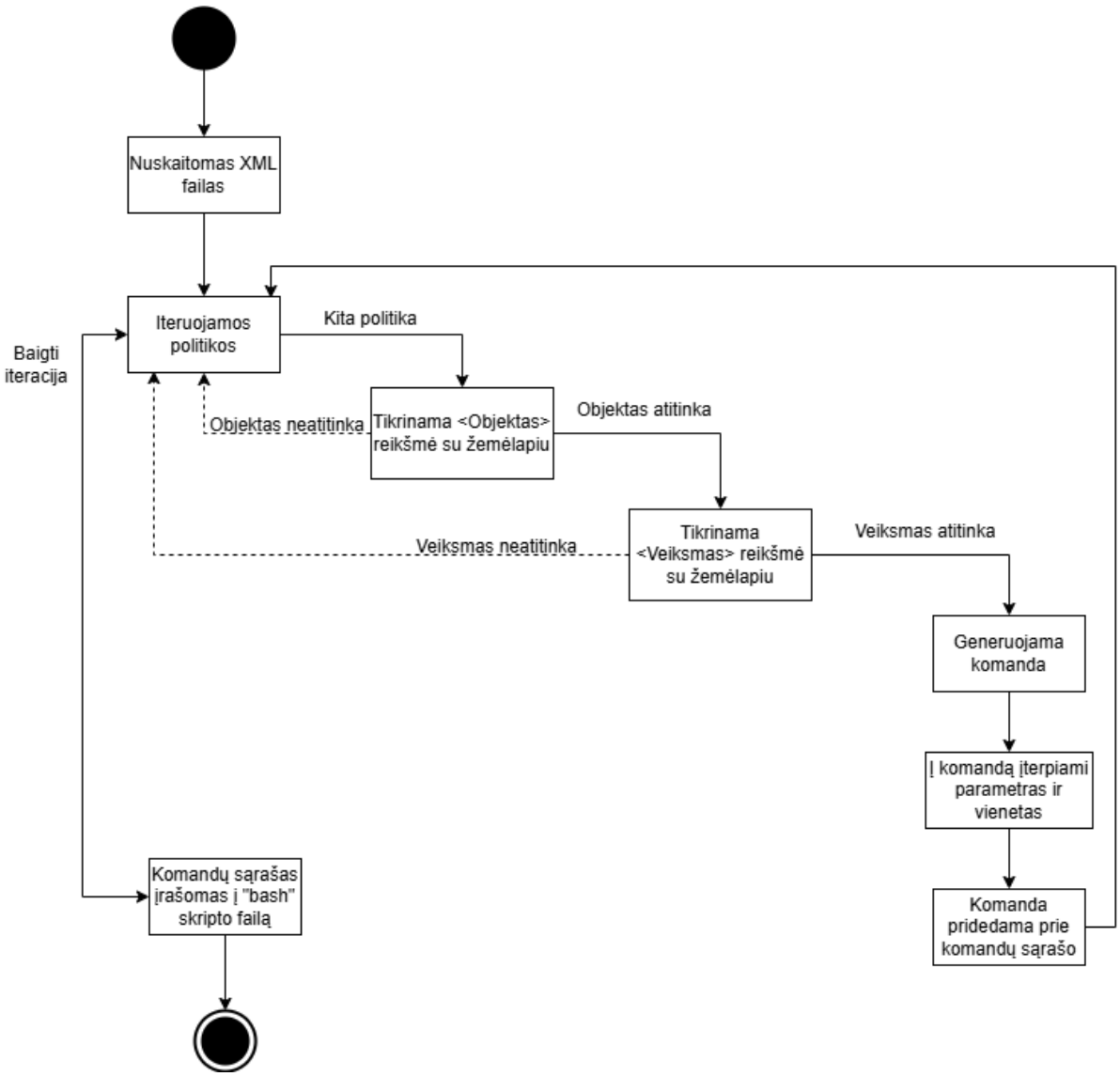
3.7 pav. Reikalingų žodžių sakiniuose konvertavimas į XML elementus

3.4.2. XML failo konvertavimo į skriptą modulis

Šiame modelyje vyksta XML failo išskaitymas, objektų ir veiksmų paieška bei atitinkamų komandų priskyrimas tiems elementams.

Sistema šiame etape sukuria politikos objektų ir veiksmų žemėlapį, kuris padeda programai sutapatinti objektus ir veiksmus kartu su „Bash“ komandomis. Žemėlapis yra kelių lygių: pirmasis lygis – tikrinama objekto reikšmė, antras lygis – tikrinama to objekto veiksmo reikšmė. Prototipo kodo žemėlapyje yra 22 objektų reikšmės ir po 2-5 veiksmus kiekvienam objektui. Svarbu paminėti, žemėlapyje nurodyta, kad didžiųjų ir mažųjų raidžių skirtumus reikia ignoruoti. Taip pat objektų galūnės yra paliekamos įvairioms reikšmėms, tai yra, nenurodoma, kad būtinas yra Vardininko ar koks kitoks linksnis, todėl galima naudoti objektus įvairių linksnių.

3.8 paveiksle pavaizduota šio modulio veiklos diagrama. Procesas prasideda nuo XML failo nuskaitymo, po nuskaitymo faile esančių politikų elementų vidus yra iteruojamas. Kiekvienoje iteracijoje atliekamas objekto reikšmės tikrinimas su žemėlapyje esančiomis objektų reikšmėmis. Jei objekto reikšmė atitinka, tada tikrinama objekto veiklos reikšmė. Jei objekto ar veiklos reikšmė neatitiko, tada grįžtama į iteracijos veiklą ir tikrinama kita politika. Jei tiek objektas tiek veiksmas atitiko reikšmes iš žemėlapio, tada generuojama nurodyta komanda ir pridedami politikos parametrai bei vienetai. Toliau komanda pridedama prie komandų sąrašo kintamojo pabaigos ir toliau tęsiama iteracija. Pasibaigus iteracijoms, komandų sąrašas įrašomas į „Bash“ skripto failą. Todėl šio modulio įvestis yra „Politikos.xml“, o išvestis – „Politikos.sh“



3.8 pav. XML failo konvertavimo į skriptą modulio veiklos diagrama

Veiksmų žemėlapyje nurodytos komandos yra bazinės struktūros, kurios yra papildytos nežinomaisiais kintamaisiais. Jeigu pasirenkama žemėlapyje nurodyta komandą (dėl atitikusių ir susietų objekto ir veiksmo), tada sistema vietoj nežinomųjų kintamųjų įterpia būtent tos politikos parametrus ir vienetus. Jeigu politika neturėjo parametrų ar vienetų, tada paliekama bazinė komanda. Dalį sugeneruoto skripto galima pamatyti 3.9 paveiksle. Būtent su tokia struktūra yra sukuriamos komandos iš bazinių komandų ir taip sukuriamas visas skripto failas.

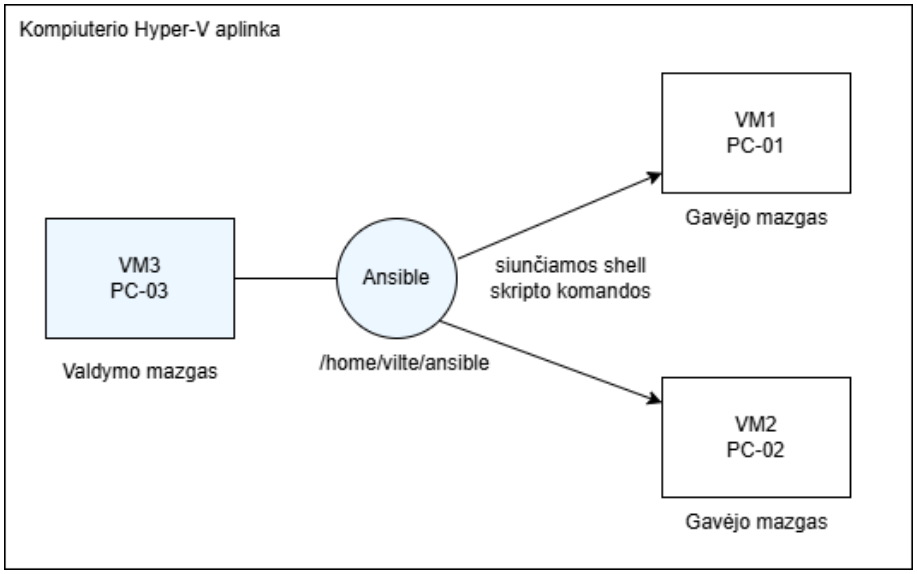
```
#!/bin/bash
# Sugeneruotas Bash skriptas pagal IT saugos politikos nustatymus
sudo passwd -l --minlength 8
for user in $(cut -d: -f1 /etc/passwd); do sudo chage --maxdays 365 $user; done
```

3.9 pav. Dalis IT saugos politikų sugeneruoto „Bash“ skripto

Prototipui nurodžius skaityti pavyzdinį „Politikos.xml“ failą, kuris atvaizduotas 2 priede, prototipas išvedė „Politikos.sh“ failą, kuris atvaizduotas 3 priede.

3.4.3. Automatizuoto skripto diegimo įrenginiuose modulis

Šiam moduliui įgyvendinti buvo sukurtas virtualių mašinų tinklas ir panaudotas „Ansible“ įrenginių orkestravimo įrankis. Suprasti šios infrastruktūros komponentus galima peržiūrėjus 3.10 paveiksle pavaizduotą šio modulio priklausomybės schemą.



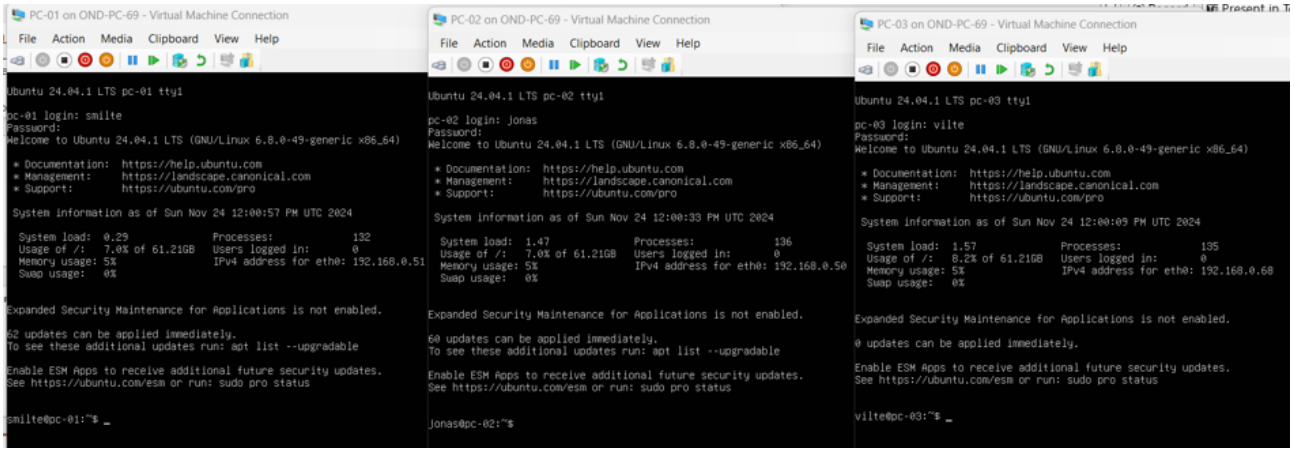
3.10 pav. Virtualių mašinų tinklo ir „Ansible“ priklausomybės schema

Virtualios mašinos, kurių yra 3, paleistos kompiuterio „Hyper-V“ aplinkoje. Visos VM turi sudiegtą „Ubuntu Server 24.04“ operacinę sistemą. Kiekviena VM turi savo naudotoją, kaip galima matyti 3.11 paveiksle. Apibendrintai apie visas VM:

1. VM1 (PC-01) – tai fiktyvios įmonės darbuotojos Smiltės kompiuteris, kurio IP adresas yra 192.168.0.51. Šis kompiuteris neturi saugumo konfigūracijų, todėl jis nurodytas kaip „Ansible“ gavėjo „mazgas“. Kompiuteris turi naudotoją „ansible_user“, per kurį, panaudojant SSH raktus, gali siųsti komandas valdymo „mazgas“.

2. VM2 (PC-02) – tai fiktyvios įmonės darbuotojo Jono kompiuteris, kurio IP adresas yra 192.168.0.50. Šis kompiuteris neturi saugumo konfigūracijų, todėl jis nurodytas kaip „Ansible“ gavėjo „mazgas“. Kompiuteris turi naudotoją „ansible_user“, per kurį, panaudojant SSH raktus, gali siųsti komandas valdymo „mazgas“.

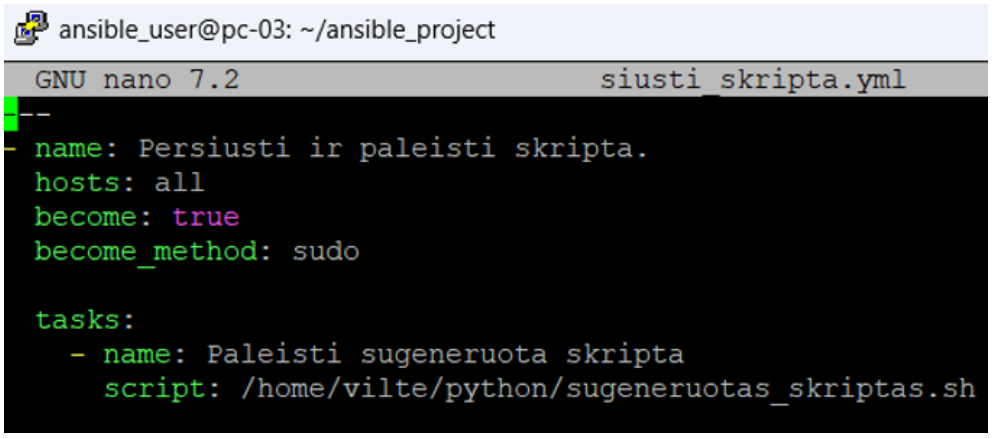
3. VM3 (PC-03) – tai fiktyvios įmonės administratorės Viltės kompiuteris, kurio IP adresas yra 192.168.0.68. Šis kompiuteris yra „Ansible“ valdymo „mazgas“. Kompiuteryje yra sudiegtas „Ansible“ įrankis. Kataloge */home/ansible_user/ansible_project* yra pridėtas jau aptartas įvesties duomenų failas – inventoriaus failas. Iš šio kompiuterio galima prisijungti į kitus du kompiuterius panaudojant SSH raktus.



3.11 pav. Virtualių mašinų tinklas „Hyper-V“ aplinkoje

Kai suveikė du pirmi sistemos moduliai, PC-03 kompiuteryje atsiranda išvesties failas – „sugeneruotas_skriptas.sh“, kuris laikomas */home/vilte/python* kataloge. Turint šį failą, galima pradėti automatizuotą politikų diegimą su „Ansible“ įrankiu.

Automatizuotas politikų diegimas yra paleidžiamas su „Ansible“ valdymo kodu, kuris vadinamas „playbook“. Šiam prototipui buvo sukurtas „siusti_skripta.yml“ valdymo kodo failas, kuris pavaizduotas 3.12 paveiksle. Šiame YAML skripte yra sukurta komanda, pavadinimu „persiųsti ir paleisti skriptą“, kuri į visus inventoriuje esančius kompiuterius siunčia sugeneruotą „Bash“ skriptą su „sudo“ teisėmis.

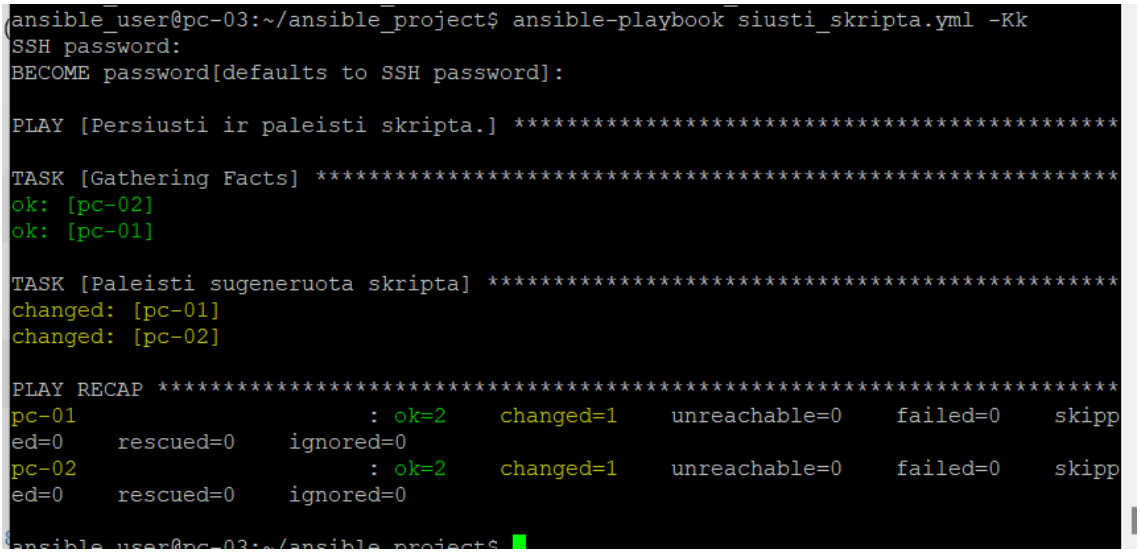


3.12 pav. „siusti_skripta.yml“ failo vidus

Šis valdymo kodas paleidžiamas administratoriaus kompiuteryje su komanda „ansible-playbook siusti_skripta.yml -Kk“. Visas sugeneruotas skriptas yra automatiškai siunčiamas į kiekvieną iš inventoriuje nurodytų kompiuterių. 3.13 paveiksle matomas komandos paleidimas ir užduoties (sugeneruoto skripto paleidimas visuose kompiuteriuose) sėkmingas įgyvendinimas. Atsakyme matomi atitinkami statusai:

- Statusas „OK“ ties kiekvienu kompiuteriu reiškia, kad į kompiuterius buvo sėkmingai prisijungta.

- Statusas „Changed“ ties kiekvienu kompiuteriu reiškia, kad skripte nurodytos komandos įsidiegė kompiuteriuose ir matomi pakeitimai pačiame kompiuteryje. Tai įrodo, kad politikos pakeitė esamus saugos nustatymus.



3.13 pav. Automatiškai sudiegto skripto sėkmingas atsakymas ir rezultatas

Iš tiesų, galima matyti kaip dabar PC-01 kompiuteryje susidiegė skripte nurodytos saugumo politikos. Pavyzdžiui, politika 1.2., kuri nurodo, kad naudotojo slaptažodis turi galioti 365 dienas, sėkmingai susidiegė Smiltės paskyroje. Tai galima patikrinti įvedus *chage -l smilte* komandą, kaip 3.14 paveiksle. Gautas atsakymas nurodo, kad tikrai – slaptažodžio amžius yra 365 dienos, todėl 2024 metų lapkričio 22 dieną sukurtas slaptažodis baigs galioti 2025 metų lapkričio 22 dieną.

3.14 pav. Smiltės slaptažodžio galiojimo laikas PC-01 kompiuteryje

Kitas pavyzdys, tai 1.5. politika, kuri nurodė, kad turi būti įjungta MFA autentifikacija. Kai skriptas baigė savo darbą, 3.15 paveiksle matoma, kad PC-01 kompiuteryje tapo sudiegtas „Google Authenticator“ funkcionalumas. Smiltei, įrašius konfigūravimo komandą *google-authenticator*, kompiuteris pasiūlė susisieti savo telefoną su kompiuteriu ir susikurti verifikacijos kodą. Taigi, Smiltei nereikėjo nieko instaliuoti – tik pasibaigus konfigūraciniam skriptui, Smiltė gali naudotis visomis sudiegtomis funkcijomis.

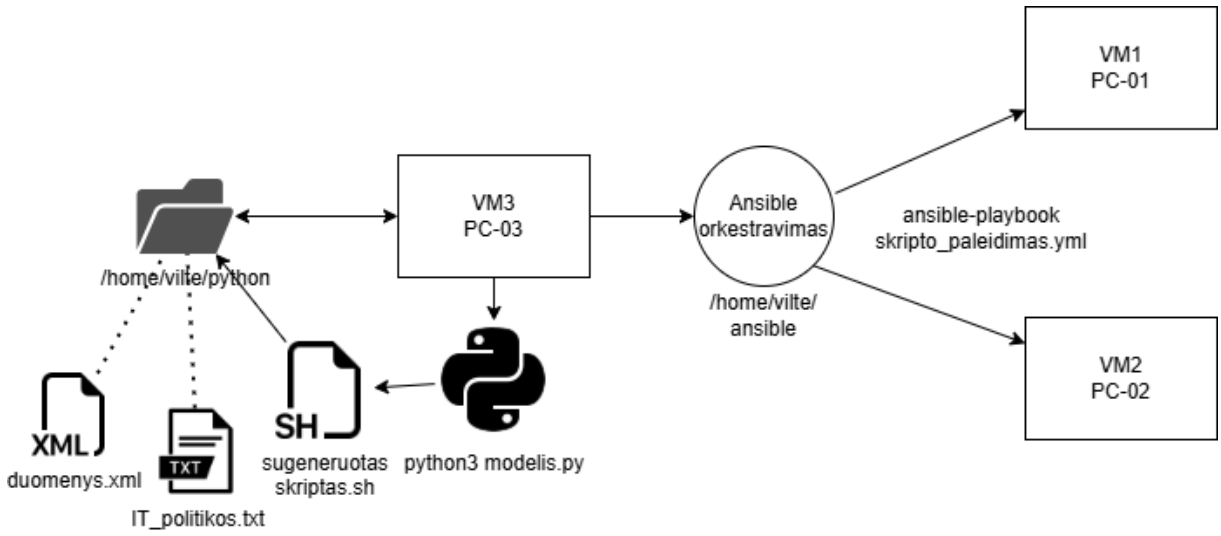
3.15 pav. MFA autentifikacijos registravimas PC-01 kompiuteryje

Taigi, galima patvirtinti, kad politikos, nurodytos komandomis „Bash“ skripto faile, sėkmingai susidiegė inventoriuje nurodytuose kompiuteriuose. Diegimas įvyko automatiškai – administratoriui užteko įvesti vieną komandą, panaudojus „Ansible“ įrankį, o visas diegimas vyko fone. Dabar kompiuteriai atitinka IT įmonės saugos politikas.

3.4.4. Rezultatas

Šios prototipinės sistemos rezultatas yra pilnai sukonfigūruota automatinė sistema, kuri gali iš įmonės tekstinio IT politikų dokumento interpretuoti ir sugeneruoti atitinkamą IT politikų diegimo skriptą bei automatiškai sudiegti šį skriptą nurodytuose įmonės kompiuteriuose.

Visi trys paminėti moduliai buvo sukelti į vieną aplinką – į PC-03 kompiuterį. Sistemos moduliai, kurie atlieka konvertavimus, buvo sujungti į vieną „Python“ failą – „modelis.py“, kuris veikia su „Python3“ paketu. Įvesties ir išvesties failai talpinami */home/vilte/python* direktorijoje. „Ansible“ orkestravimas yra talpinamas */home/vilte/ansible* direktorijoje kartu su paleidimo instrukcijomis „skripto-paleidimas.yml“ ir inventoriaus failu. Visa ši prototipo infrastruktūra pavaizduota 3.16 schemoje.



3.16 pav. Modelio veikimo infrastruktūra PC-03 kompiuteryje

Prototipo veikimo žingsniai yra tokie:

1. Į */home/vilte/python* direktorija įkeliamas IT politikų tekstinis dokumentas.
2. Paleidžiamas „modelis.py“ skriptas.
3. Paleidžiama „ansible-playbook siusti_skripta.yml -Kk“ komanda.

Prototipinės sistemos galutinis rezultatas yra tas, kad įmonės kompiuterių konfigūracijos ir nustatymai pilnai atitinka įmonės IT politikų dokumente įrašytas saugos politikas. Todėl, panaudojus šią sistemą, įmonė tampa pripažinta atitinkanti politikas ir yra apsaugota nuo įmonės nuspėjamų grėsmių. Siūlomo modelio prototipas atlieka visą IT saugos politikų valdymo ciklą – nuo politikos dokumento interpretavimo iki konfigūracijų sudiegimo įrenginiuose.

3.5. Išvados

Sėkmingai įgyvendinus IT saugos politikų automatizuoto valdymo modelio prototipą, galima įžvelgti tokias išvadas:

1. Sukurtas prototipas užtikrina organizacijos IT saugos politikų automatinį įgyvendinimą, nuo teorinio dokumento iki praktinių įrenginių konfigūracijų.
2. Automatizuotas procesas sumažina rankinio darbo poreikį, pagerina tikslumą ir užtikrina nuoseklų saugos politikų taikymą.
3. Sistema efektyviai sujungia įvairias technologijas, tokias kaip NLP, XML, „Bash“, „Python“, „Ansible“, siekiant užtikrinti sklandų duomenų konvertavimą, apdorojimą ir perdavimą.
4. Prototipas sėkmingai ištestuotas realioje infrastruktūroje ir įrodė, kad įrenginiai atitinka nustatytas IT saugos politikas.

4. Automatizuoto IT saugos politikos valdymo modelio prototipo tyrimas

Šiame skyriuje yra atliekamas siūlomo modelio tyrimas, išskiriamos stipriosios ir silpnosios modelio vietos, nurodomos kitos svarbios įžvalgos, kurios buvo nustatytos įvairiais eksperimentiniais metodais.

4.1. Tyrimo tikslas, uždaviniai ir hipotezės

Tyrimo tikslas – įvertinti sukurtą IT saugos politikos automatizuoto valdymo modelį, nustatyti jo efektyvumą, tikslumą, saugumą ir praktiškumą, palyginti su kitais rinkoje esančiais sprendimais.

Tyrimo uždaviniai:

1. atlikti modelio palyginimą su esamais IT politikų valdymo įrankiais;
2. išmatuoti modelio našumą pagal atlikimo laiką, klaidų skaičių ir sistemos resursų naudojimą;
3. patikrinti modelio tikslumą, naudojant skirtingas politikas ir skirtingą jų kiekį;
4. įvertinti kompiuterio saugumo lygį po sudiegtų politikų.

Tyrimo hipotezės:

1. H1: Automatizuotas modelis reikšmingai sutrumpina saugos politikų įgyvendinimo laiką, palyginus su rankiniu diegimu bei esamais įrankiais.
2. H2: Naudojant automatizuotą modelį, sumažėja žmogiškųjų klaidų ir sintaksės klaidų skaičius.
3. H3: Modelio NLP komponentas geba tiksliai interpretuoti suformuluotas politikos taisykles.
4. H4: Automatizuotai sugeneruotas ir įdiegtas skriptas realiai pagerina įrenginio saugumą.

4.2. Siūlomo modelio prototipo kokybinis tyrimas

Šiame skyrelyje kokybinio tyrimo metodu įvertinamas siūlomas modelis tarp egzistuojančių rinkoje modelių.

4.2.1. Esamų sprendimų palyginimas

Šiuo metu IT saugos politikų automatizuotas diegimas yra dažna tema, todėl rinkoje yra bent keletas sprendimų, kurie gali išspręsti saugos klausimus. Priklausomai nuo įmonės tipo, dydžio, verslo pobūdžio, saugumo lygio nustatymo, naudojami skirtingi sprendimai diegiant IT saugos politikas. Deja, ne visos įmonės turi įsidiegusios automatizuotus sprendimus.

Kai kurios įmonės neteikia prioriteto IT politikų diegimo automatizavimui, todėl nenaudoja jokių įrankių IT politikų įgalinimui įrenginiuose ar sistemose. Tai atlieka administratorius **„rankiniu būdu“**. Sistemų administratorius pats skaito įmonės IT saugos politikų dokumentą ir pagal jį kuria komandas kiekvienam kompiuteriui. Komandos nėra siunčiamos automatizuotai, jas administratorius įrašo į kiekvieno kompiuterio terminalą ir taip politika įsidiegia. Taip pat verta paminėti, kad administratoriui reikia daug galvoti, ieškotis informacijos ir skaityti komandų dokumentaciją, norint įdiegti nurodytą politiką su visais atitinkamais parametrais. Kadangi šios operacijos atliekamos žmogaus, yra didelė tikimybė žmogiškosioms klaidoms. Todėl **„rankinis“** sprendimas yra:

- lėtas;
- daug resursų eikvojantis (laikas, darbo užmokestis);
- didina žmogiškųjų klaidų riziką;
- neefektyvus.

Dar vienas iš sprendimų, norint įgyvendinti IT saugos politikų diegimą organizacijoje yra naudojant **konfigūravimo valdymo įrankius**. Tai gali būti „Ansible“ arba „Puppet“. Tokie įrankiai gali automatizuotai siųsti YAML arba „Puppet DSL“ skriptą į nurodytus įrenginius. Tačiau skriptų paruošimui reikalingas gerai įrankį ir atitinkamą skripto kalbą išmanantis administratorius, kuris, panaudojant įrankių dokumentacijas, galėtų sukurti įmonės saugos politikų skriptą. Deja, tai užtrunka daug laiko, nes politikos yra specifinės. Skriptams sukurti reikia dirbti su dokumentaciją bei pritaikyti politikas atitinkamoms YAML ar PP failo užduotims, nes konfigūravimo valdymo įrankiai negali perskaityti tekstinio organizacijos politikų dokumentų. Administratoriui tektų pačiam analizuotis politikas ir išsigryninti jų parametrus. **Tik „Ansible“ ar „Puppet“ įrankio** naudojimas yra:

1. lėtas;
2. didina žmogiškųjų klaidų riziką;
3. tikėtinos skriptų sintaksės klaidos;
4. sudėtingas diegimas ir administravimas;
5. reikalauja papildomų žinių.

Vienas naujesnių sprendimų yra **„Intune“**, skirtas paruoštų įrenginių konfigūracijų diegimui įmonės įrenginiuose. Šis įrankis yra valdomas paprasčiau nei „Ansible“ ar „Puppet“, tačiau saugumo politikų konfigūracijų diegimui reikia didelio administratoriaus įsitraukimo. „Intune“ turi patogią grafinę sąsają, nereikia pačiam rašyti kodo ar skripto, tačiau politikų dokumento nėra galimybės perskaityti. Administratorius pats turi skaityti dokumentaciją ir pritaikyti konfigūracijas pagal įmonės politikas. **„Intune“** sprendimas:

- didina žmogiškųjų klaidų riziką;

- sudėtingas diegimas ir administravimas;
- reikalauja administratoriaus laiko ir žinių.

Taigi, šiuos aptartus sprendimus lyginant su šiame darbe aprašytu IT saugos politikos automatizuoto diegimo modeliu, galima įžvelgti daug skirtumų. 4.1 lentelėje matoma, kad didžiausias skirtumas yra tas, kad kiti sprendimai nesugeba perskaityti lietuviško teksto ir analizuoti jame aprašytas politikas. Taip pat siūlomas modelis yra daugiausiai automatizuotas, įgyvendinamas ir administruojamas greičiausiai, kainuoja mažiausiai kaštų (darbuotojų algos ir laikas, žmogiškųjų klaidų taisymas).

4.1 lentelė. Modelio palyginimas su esamais sprendimais

Kriterijus	**Modelis**	**„Ansible“/„Puppet“**	**„Intune“**	**Rankinis būdas**
Lietuviškų tekstų interpretacija	Taip	Ne	Ne	Ne
Automatinis komandų generavimas	Taip	Ne	Ne	Ne
Automatinis politikų diegimas	Taip	Taip	Taip	Ne
Politikų taikymo greitis	Labai greitas	Greitas	Lėtas	Labai lėtas
Administravimo sudėtingumas	Lengvas	Sudėtingas	Vidutinis	Labai sudėtingas
Žmogiškųjų klaidų tikimybė	Labai maža	Vidutinė	Vidutinė	Aukšta
Universalumas (OS palaikymas)	„Linux“ (galima plėtra)	„Linux“, „Windows“	Tik „Windows“	Priklausomai
Naudojimo patogumas	Lengvas	Sudėtingas	Vidutinis	Sudėtingas
Administratoriaus darbo laikas	Minimalus	Vidutinis	Vidutinis	Didelis

Apibendrinant, šiuo metu rinkoje nėra tinkamai automatizuoto bei efektyvaus sprendimo IT saugos politikų diegimui, todėl šiame darbe sukurtas automatizuotas modelio prototipas praverstų įmonėms įsigyvendinti jų IT saugos politikas organizacijos tinkle.

4.2.2. Siūlomo modelio vertinimas ir išvados

Šis IT saugos politikos automatizuoto diegimo modelis turi daug pranašumu prieš kitus šiuo metu esančius sprendimus. Tačiau pagrindiniai modelio privalumai yra tokie:

1. automatinis dokumentų interpretavimas – nereikia rankiniu būdu ieškoti, kaip įgyvendinti politiką;
2. mažesnė klaidų tikimybė – išvengiama rankinio konfigūravimo klaidų;
3. paprastesnis administravimas – nereikia YAML, „Linux“ komandų ar „PowerShell“ žinių;
4. greitesnis politikų įdiegimas – vienu metu gali būti pritaikytos kelios politikos;
5. saugumo atitikties užtikrinimas – dokumente aprašytos saugumo politikos atitinka konfigūracijas, užtikrinant jų atitiktį organizacijos reikalavimams ir standartams (ISO 27001, BDAR, NIST);
6. lengva plėtra – galima pridėti papildomas politikų taisykles arba išplėsti į „Windows“ ar „MacOS“ sistemas.

Kokybiniu metodu įvertinus modelį, galima įžvelgti ir keletą trūkumų:

1. kol kas orientuotas tik į „Linux“ – jei organizacija naudoja „Windows“, reikia papildomos plėtros;
2. priklausomybė nuo „Ansible“ – vykdymui vis tiek reikia „Ansible“, kuris turi būti tinkamai sukonfigūruotas;
3. ribotas GUI – jei modelis neturi grafinės sąsajos, gali būti sunkiau naudoti mažiau techniniams administratoriams;
4. saugumas priklauso nuo politikų – jei politikos dokumente bus suformuluotos netinkamai, saugumo politikos gali nesusidiegti įrenginiuose.

Apibendrinant, modelis yra tinkamas naudoti įmonėse, kurios nori atitikti saugumo standartams ir apsisaugoti nuo kibernetinių atakų bei įsilaužimų. Tolimesniam tyrimui reikia atlikti kiekybinę analizė, kuri įvertins modelio našumą, efektyvumą, greitį ir universalumą.

4.3. Siūlomo modelio prototipo kiekybinis tyrimas

Šiame darbe aptarto modelio tikslumui įvertinti, atliekamas kiekybinis tyrimas, kuriame vertinamas tikslumas, klaidos, greitis, resursų panaudojimas ir saugumo užtikrinimas.

4.3.1. Modelio kokybės įvertinimas, naudojant skirtingus politikų įvesties dokumentus

Šis modelis yra paleistas „Hyper-V“ aplinkoje sukurtoje VM. Kaip buvo paminėta praeitame skyriuje, visą infrastruktūrą sudaro 3 VM, iš kurių dvi – gaunančios skriptus, o viena iš VM – generuojanti skriptus. Todėl pirmasis tyrimas, kuris vertina generavimo kokybę, yra vykdomas PC-03 virtualioje mašinoje. Prisijungimui prie VM naudojamas SSH įrankis.

PC-03 parametrai:

- 4 GB RAM atminties;
- 4 virtualūs procesoriai;
- „Ubuntu 22.04“ operacinė sistema;
- IP adresas - 192.168.0.52.

Šiam eksperimentui tokių parametrų užtenka, kadangi „lt_core_new_lg“ modelio biblioteka užima apie 541 MB disko vietos, o praktiškai RAM užims apie 1,5-2 GB RAM atminties. Taip pat naudojami nedideli įvesties bandymų dokumentai, kurie neapkraus sistemos resursų.

Siekiant įvertinti modelio gebėjimą generuoti tinkamus XML elementus ir „Bash“ komandas, buvo atlikti 15 bandymų su skirtingais įvesties IT politikų dokumentais. Modelio generavimo kokybei įvertinti buvo atlikti 2 tipų testai:

- **10 bazinių testų**, kurių struktūra atitinka nurodytas modelio įvesties taisykles, bet skiriasi apimtimi, sritimi ir pateikimu:
  - politikų kiekis: nuo 3 iki 16;
  - skirtingos politikos ir parametrai;
  - orientuotos į įvairias sritis.
- **5 kritiniai testai**, kurių struktūra neatitinka nurodytas įvesties taisykles ir specialiai skirti tikrinti modelio atsparumą, silpnąsias vietas, naudojant:
  - dviprasmybes;
  - nenuoseklumus;
  - prieštaringas politikas;
  - įvairius, struktūros neatitinkančius, formatus;
  - ilgus ir sudėtingus sakinius.

Eksperimentas buvo vykdomas tokia tvarka: kiekvienas bandymas paleidžiamas su skirtingu įvesties failu, o modeliui suveikus, sukuriami du failai – XML ir „Bash“. Todėl kiekvieną bandymą sudaro trys failai: įvesties tekstinis dokumentas, išvesties XML ir „Bash“ failai. Kadangi modelyje generavimą sudaro du etapai: iš tekstinio failo į XML failą ir iš XML failo į „Bash“ failą, kiekviename etape apskaičiuojamas tinkamai sugeneruotų politikų skaičius. Tai yra, jei visos tekstiniame faile surašytos politikos buvo tinkamai sugeneruotos XML ir „Bash“ failuose, tada šis bandymas pavyko 100 %. Tačiau gali būti, kad XML etape politikos buvo gerai sugeneruotos, tačiau „Bash“ sugeneravo blogai, todėl atitinkamai kokybė matuojama per kiekvieną generavimo etapą, o ne kaip galutinis variantas.

Modelio generavimo vertinimo metodas yra toks: lyginama kiekvieno etapo išvestis su tikslinėmis, rankiniu būdu paruoštomis teisingomis versijomis. Kiekviena politika yra įvertinama, ar ji buvo: atpažinta (NLP nuskaitymas), teisingai interpretuota į XML (objektas, veiksmas, parametras), korektiškai konvertuota į „Bash“ komandą (veiksmų žemėlapio, angl. „Action Map“, funkcionalumas). Tikslumas yra matuojamas politikų skaičiumi, kadangi įvesties failą sudaro atitinkamas skaičius politikų – toks pats skaičius turi atitikti „Bash“ komandų skaičių. 4.2 lentelėje pateikti eksperimento rezultatai (absoliutūs skaičiai), kiek kiekvienu etapu buvo sugeneruota faktiškai teisingų politikų. Ši lentelė rodo modelio produktyvumą – kiek jis sugeneravo tinkamų politikų iš visų politikų.

4.2 lentelė. Sugeneruotų XML ir „Bash“ politikų kiekiai kiekviename dokumente

Dokumentas	Testo tipas	XML politikos	„Bash“ politikos	Visos politikos
B1	Bazinis	6	5	6
B2	Bazinis	8	7	9
B3	Bazinis	4	4	5
B4	Bazinis	5	2	5
B5	Bazinis	3	3	3
B6	Bazinis	4	4	4
B7	Bazinis	6	6	6
B8	Bazinis	5	5	5
B9	Bazinis	15	14	16
B10	Bazinis	13	13	13
K1	Kritinis	4	0	4
K2	Kritinis	3	2	3
K3	Kritinis	3	1	3
K4	Kritinis	0	0	2
K5	Kritinis	3	1	4

Kiekvienas iš 15 įvesties dokumentų (10 bazinių ir 5 kritinių) buvo vertinamas XML lygmenyje: ar buvo teisingai suformuota politikos struktūra (veiksmas, objektas, parametras), taip pat ir „Bash“ lygmenyje: ar buvo sugeneruota teisinga komanda, atitinkanti politiką. Priskaičiuotos morfologinės klaidos (pvz., „bluokuojti“, „sekundis“) buvo vertinamos kaip dalinai teisingos, bet įtrauktos į teisingų atvejų skaičių, kadangi iš esmės išlaikytas semantinis tikslumas.

Apibendrinant 4.2 lentelėje pateiktus rezultatus, galima teigti, kad:

1. Modelis labai tiksliai sugeneravo XML struktūras – daugumoje dokumentų XML politikų skaičius lygus bendram politikų skaičiui (pvz., B1: 6/6, B4: 5/5).
2. „Bash“ komandų generavimas buvo šiek tiek mažesnio tikslumo, tai rodo kad XML NLP interpretacija yra gana tiksli, bet komandų konversijos logika (veiksmų žemėlapis) dar turi netikslumų (pvz., B1: 5/6, B2: 7/9).
3. Kritiniuose testuose modelis praranda gebėjimą interpretuoti dviprasmiškas ar prieštaringas taisykles komandų lygyje, nors NLP žingsnis veikia pakankamai gerai (pvz., K3: 3/2/5).

Modelio tikslumui įvertinti, taip pat svarbu įvertinti modelio kokybę. Šiame tyrime buvo apskaičiuoti ir santykiniai rezultatai. Tam buvo taikytos klasikinės klasifikavimo užduotims skirtos metrikos: „Recall“, „Precision“, „F1-score“, kurios leidžia įvertinti modelio gebėjimą teisingai ir išsamiai interpretuoti IT saugos politikų dokumentus bei sugeneruoti tinkamą komandą. Kadangi tyrimo tikslas – automatizuoti saugos politikų taikymą realioje sistemoje, svarbu ne tik tai, ar modelis pataikė į teisingą komandą, bet ir tai, ar nepraleido nė vienos svarbios politikos taisyklės. Todėl buvo pasirinktos metrikos, kurios leidžia išmatuoti šiuos aspektus:

1. padengimą („Recall“) – kiek iš visų taisyklių modelis sugebėjo atpažinti (XML struktūroje arba „Bash“ komandoje);
2. tikslumą („Precision“) – kiek iš visų modelio sugeneruotų komandų (galutinės išvesties) buvo tikrai teisingos;
3. „F1-score“ – kaip bendrą kokybės įvertį, kai svarbūs abu kriterijai, „Precision“ ir „Recall“ vidurkis.

„Recall“ metrika apskaičiuojama su šia formule:

$$\text{Recall} = \frac{TP}{TP + FN}$$

čia:

- TP – teisingai sugeneruotos taisyklės (angl. „True Positives“);
- FN – taisyklės, kurias modelis praleido (angl. „False Negatives“).

„Precision“ metrika apskaičiuojama su šia formule:

$$\text{Precision} = \frac{TP}{TP + FP}$$

čia:

- TP – teisingai sugeneruotos taisyklės;
- FP – klaidingai arba perteklinai sugeneruotos komandos (angl. „False Negatives“.

„F1-score“ metrika ypač naudinga tuo atveju, kai „Precision“ ir „Recall“ reikšmės yra skirtingos, nes „F1-score“ jautriai reaguoja į vienos metrikos sumažėjimą ir parodo realų modelio veikimo balansą. Ši metrika apskaičiuojama su formule:

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Šių metrikų naudojimas leidžia ne tik kiekybiškai įvertinti generavimo kokybę, bet ir tiksliai identifikuoti, kur modelio veikimo grandinėje atsiranda klaidos – ar NLP analizėje (XML), ar komandų generavimo žingsnyje („Bash“).

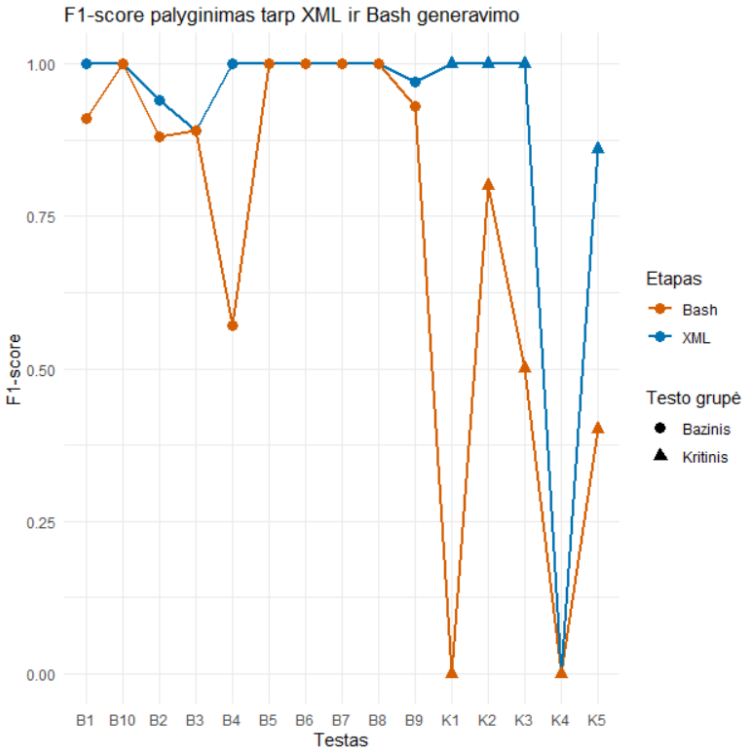
4.3 lentelėje pateikiami santykiniai rezultatai, kurie nurodo modelio kokybės įvertinimą, panaudojant „Recall“, „Precision“, „F1-score“ metrikas. Ši lentelė parodo modelio kokybę – ar jis dirbo kokybiškai, kiek teisingų politikų sudarė procentinę dalį visų dokumente esančių politikų.

4.3 lentelė. Modelio kokybės metrikos: „Precision“, „Recall“ ir „F1-score“ XML ir „Bash“ generavimo etapuose

NR	XML Precision	XML Recall	XML F1	BASH Precision	BASH Recall	BASH F1
B1	1,0	1,0	1,0	1,0	0,83	0,91
B2	1,0	0,89	0,94	1,0	0,78	0,88
B3	1,0	0,8	0,89	1,0	0,8	0,89
B4	1,0	1,0	1,0	1,0	0,4	0,57
B5	1,0	1,0	1,0	1,0	1,0	1,0
B6	1,0	1,0	1,0	1,0	1,0	1,0
B7	1,0	1,0	1,0	1,0	1,0	1,0
B8	1,0	1,0	1,0	1,0	1,0	1,0
B9	1,0	0,94	0,97	1,0	0,88	0,93
B10	1,0	1,0	1,0	1,0	1,0	1,0
K1	1,0	1,0	1,0	0,0	0,0	0,0
K2	1,0	1,0	1,0	1,0	0,67	0,8
K3	1,0	1,0	1,0	1,0	0,33	0,5
K4	0,0	0,0	0,0	0,0	0,0	0,0
K5	1,0	0,75	0,86	1,0	0,25	0,4

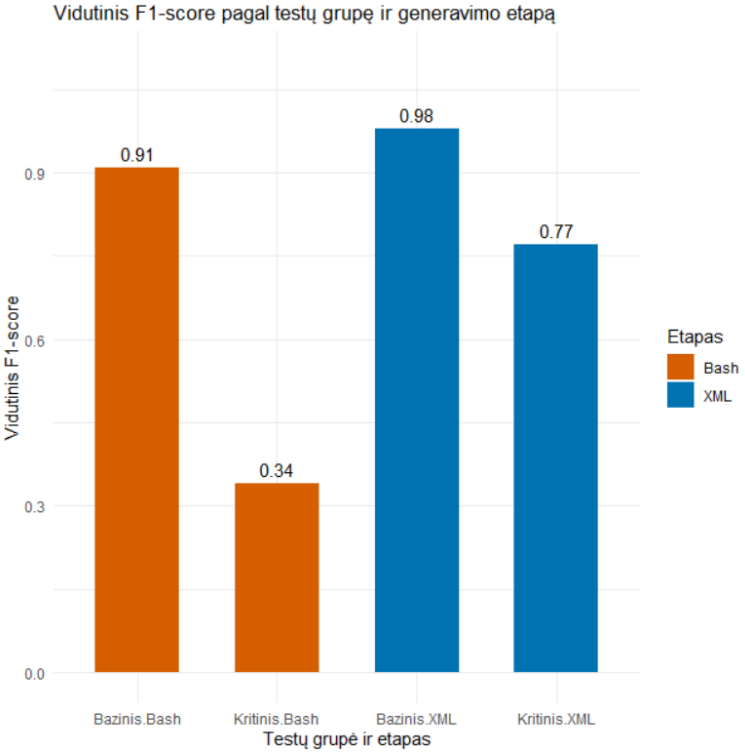
Remiantis 4.3 lentelėje esamomis metrikomis, galima daryti išvadą, kad modelio kokybė XML generavimo etape buvo itin aukšta – daugumoje testų XML „Precision“ ir „Recall“ reikšmės siekė 1,0, o tai rodo, kad modelis sugebėjo tiksliai atpažinti visas politikos taisykles. Visgi „Bash“ komandų generavimo etape rezultatai buvo mažiau stabilūs, ypač kritiniuose testuose. Nors baziniuose testuose „Bash F1-score“ dažniausiai viršijo 0,85, kritiniuose dokumentuose reikšmės svyravo tarp 0 ir 0,5, o tai parodė, kad konversija iš XML į „Bash“ komandas buvo jautresnė dviprasmybėms, neatitikimams įvestyje. Šis skirtumas tarp XML ir „Bash“ metrikų leidžia teigti, kad pagrindinės modelio silpnybės slypi ne NLP analizėje, bet būtent „veiksmų žemėlapio“ (angl. „action map“) žingsnyje, todėl tolimesniam tikslumo gerinimui būtina koncentruotis į XML-„Bash“ konversijos algoritmo tobulinimą.

Taip pat XML ir „Bash“ etapų skirtumą galima įžvelgti 4.1 paveiksle. Grafikas rodo „F1-score“ reikšmes XML ir „Bash“ generavimo etapuose kiekviename teste. Aiškiai matyti, kad XML analizės rezultatai išlieka aukšti visų testų metu, o „Bash“ generavimas tampa itin netikslus kritiniuose testuose (K1–K5). Šis skirtumas leidžia teigti, kad pagrindinės modelio problemos slypi ne NLP interpretacijoje, bet konversijos į veiksmų žemėlapį žingsnyje.



4.1 pav. „F1-score“ pasiskirstymas, priklausomai nuo testo grupės ir etapo

Svarbu pamatyti modelio tikslumo vidutinį pasiskirstymą kiekviename etape – taip galima įvertinti bendrą modelio tikslumą. Tai atvaizduota 4.2 paveiksle. Šis grafikas leidžia apibendrinti modelio rezultatus ir aiškiai matyti, kad XML generavimas yra stabilus tiek baziniuose, tiek kritiniuose testuose, o „Bash“ generavimo kokybė ženkliai suprastėja kritiniuose dokumentuose.



4.2 pav. Vidutinis „F1-score“ pasiskirstymas pagal testo grupes

Apibendrintai, šio viso modelio bendras tikslumas abiejose testų grupėse:

- Baziniuose testuose modelio tikslumas siekia 0,88, t. y. modelis teisingai sugeneravo 88 % visų politikų kaip galutines „Bash“ komandas (63 iš 72);
- Kritiniuose testuose tikslumas smuko iki 0,25, t. y. tik 25 % politikų buvo interpretuota teisingai (4 iš 16).

4.3.2. Modelio klaidų analizė, nagrinėjant įvairius bandymus ir rezultatus

Atlikus kiekybinį tyrimą, buvo atrastos tam tikros klaidos, kurios identifikuoja modelio silpnąsias vietas. Kiekviena klaida buvo išanalizuota ir surinkta į atitinkamų klaidų grupes. Grupes nurodo, kuris modelio etapas sutrikdė gauti tinkamą atsakymą. Grupes buvo suskirstytos į šias sritis:

- gramatika – kai modelis konvertuoja žodžius į neegzistuojančius žoždius dėl lematizacijos, taip pat kabučių buvimas aplink ieškomus žodžius;
- sakinių segmentacija – kai modelis netinkamai padalija tekstą į sakinius arba sujungia antraštes su sakinio pradžia, dėl ko iškraipomas gramatinis kontekstas;
- interpretacija – kai modelis netiksliai supranta sakinio prasmę dėl dviprasmybių, neaiškių formuluočių, neapibrėžtos struktūros ar prieštaravimų tekste ir negali identifikuoti reikalingų žodžių (pvz., parenka ne tą objektą ar veiksmą);
- komandų generavimas – kai modelis negali suformuoti scenarijaus veiksmo dėl veiksmų žemėlapio ribotumo ar dėl to, kad veiksmas neatitinka realių sistemos galimybių.

Klaidos buvo išanalizuotos ir struktūriškai pateiktos 4.4 lentelėje. Kiekviena klaida turi nurodytą savo testo numerį, klaidos grupę ir paaiškinimą.

4.4 lentelė. Modelio klaidų klasifikacija pagal grupes

Klaidos aprašymas	Klaidos grupė	Testas
XML elemento reikšmę nurodo „blokuojti“, o ne „blokuoti“	Gramatika	B1
MFA be kabučių todėl ima „authentication“	Gramatika	B1
„Antivirusinė“ ignoruojama, ima „saugumas“ iš antraštės	Sakinių segmentacija	B2
XML elemento reikšmę nurodo „blokuojti“, o ne „blokuoti“	Gramatika	B3
Ignoruojamas objekto būdvardis „įeinantis/išeinantis“ prievadas, nes ieško tik parametro	Interpretacija	B3
Vietoj objekto „kritiniai atnaujinimai“, identifikuoja objektą: „kas“ 7 dienas	Interpretacija	B3
Nėra veiksmų žemėlapyje komandų apie žurnalus	Komandų generavimas	B4
XML elemento reikšmę nurodo „blokuojti“, o ne „blokuoti“, „sekundžių“ paverčia į „sekundis“	Gramatika	B9
Identifikuoja kaip objektą „ugniasienėje“, o ne „prievadas“	Interpretacija	B9
Veiksmažodis „būti“ nėra įterptas į veiksmų žemėlapį	Komandų generavimas	K1
Vietoj „bandymų“ identifikuoja objektą „tai“ – „tas“	Interpretacija	K1
Veiksmažodis „galėti“ nėra įterptas į veiksmų žemėlapį	Komandų generavimas	K2
Iš antraštės žodį „Saugumas“ ima kaip objektą vietoje „prievadai“	Sakinių segmentacija	K4
Kai du veiksmažodžiai sakinyje, ima antrą: vietoje „atidaryti“ ima „išskyrus“	Interpretacija	K4
Kai du skaičiai sakinyje: „22 ir 80“, ima antrą – 80	Interpretacija	K4
Iš antraštės žodį „Politika“ ima kaip objektą vietoje „Slaptažodis“ (dėl kablelių)	Sakinių segmentacija	K5
Veiksmažodis „būti“ nėra įterptas į veiksmų žemėlapį	Komandų generavimas	K5

Dažniausias gramatinių klaidų pavyzdys buvo rastas B1, B5, B9 testuose. Juose buvo nustatyta aiški sąsaja veiksmo ir objekto (sakinio dalių – veiksnio ir tarinio), kaip veiksmo reikšmė priklauso nuo suformuluoto objekto. Jei objektas yra aiškus daiktavardis „paskyra“, tada veiksmas „blokuojama“ yra interpretuojamas teisingai – jis paverčiamas į XML elemento reikšmę „blokuoti“. Tačiau, jei objektas yra „prisijungimas“, kas modeliui gali atrodyti kaip abstraktus veiksmas, tada veiksmą „blokuojamas“ jis veiksmo elementą paverčia į neegzistuojantį žodį „blokuojti“. Tai įrodo, kad modelis gali turėti lematizacijos klaidų, jei sakinyje objektas panašesnis į veiksmą, nei į sakinio gramatinį veiksnį (naudojama žodžio žyma „nsubj“) .

Pagrindinės sakinių segmentacijos klaidos, kai modelis netinkamai apsibrėžė sakinio ribas ir ieškojo veiksmo bei objekto už sakinio ribų. Tai buvo pastebėta B2, B3 testuose. NLP modelis skaito visą tekstą ir tik tada segmentuoja į sakinius, todėl, jei neranda aiškaus objekto, sujungia su antraštes eilute.

Taip pat buvo pastebėta, kad jei žodis „antivirusinė“ yra ne pirmas sakinio žodis, tada modelis gerai nuskaito sakinį. Norint išvengti tokių klaidų, vertėtų modeliui nurodyti analizuoti kiekvieną sakinį atskirai, o ne visą failo tekstą kartu ir tik tada segmentuoti.

Dažna klaida yra sakinio objektų, veiksmų ir parametrų netikslus interpretavimas. Tai buvo pastebėta B2, B3, B9, K1, K4 testuose. Netinkamas interpretavimas gali atsitikti dėl taisyklių neatitinkančios sakinio struktūros, blogai išreikštų žodžių, todėl NLP gali klaidingai parinkti žodžius objektui, veiksmui ar parametrui. Pavyzdžiui, sakinyje: „Kritiniai atnaujinimai diegiami kas 7 dienas“, modelis interpretuoja žodį „kas“ kaip objektą vietoj „atnaujinimai“. Galima šias klaidas analizuoti taip: laiko intervalas nurodytas neįprasta sakinio forma ir daugiskaitinis junginys „kritiniai atnaujinimai“ NLP modeliui atrodo labiau kaip papildinys ar atributas, o ne pagrindinis sakinio objektas. Norint išvengti tokių klaidų, reiktų teisingai kalbiškai užrašyti politikas sakiniuose ir naudoti paprastus žodžius, trumpus sakinius.

Galima teigti, kad šio modelio silpnoji vieta yra veiksmų žemėlapio funkcionalumas, tai yra komandų generavimo etapas. Šiame etape nustatyta ganėtinai daug klaidų – galima matyti iš klaidų analizės ir kiekybinio tyrimo. Testai B4, K1, K2, K5 nesulaukė teisingo rezultato, nes veiksmų žemėlapyje nebuvo nurodytos politikoms atitinkančios komandos arba nėra surašyti sinoniminiai žodžiai, kurie galėtų taip pat įeiti prie komandų aprašymų. Taigi, kaip didžiausias klaidas galima išskirti esančias komandų generavimo grupėje, nes dėl šių klaidų nėra gautas galutinis rezultatas daugelyje testų. Kitas klaidas galima išspręsti su teisingai suformuluota įvestimi, tačiau būtent komandų generavimo etapas ir veiksmų žemėlapis turi būti taisomas modelio viduje.

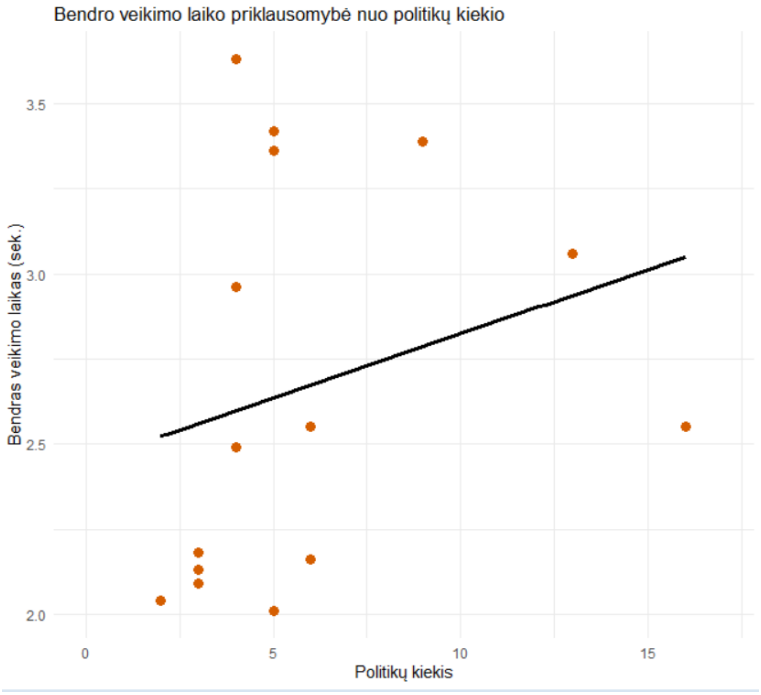
4.3.3. Modelio greičio ir resursų naudojimo matavimas kiekvienu bandymu

Šiame skyriuje analizuojamas modelio veikimo greitis ir kompiuterinių resursų panaudojimas kiekvieno testavimo metu. Kiekvienam bandymui buvo išmatuoti: vykdymo laikas, procesoriaus apkrova ir atminties sąnaudos, naudojant *time -v* komandą. Matavimai buvo atlikti tiek bazinių, tiek kritinių testų metu.

Pagrindinės metrikos, naudojamos įvertinti resursų panaudojimą:

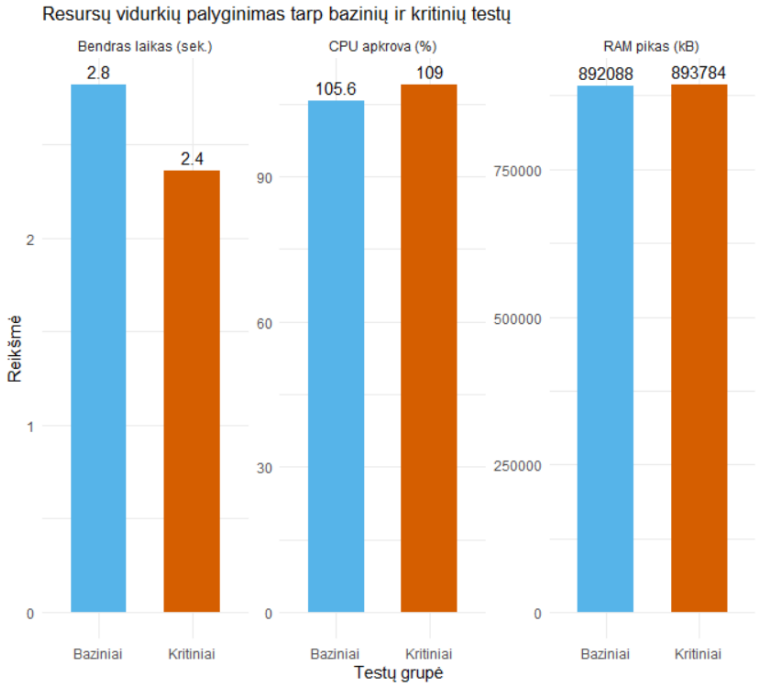
1. bendras modelio veikimo laikas, t. y. vartotojo patiriamas laikas;
2. CPU apkrovos trukmė – apdorojimo skaičiavimų intensyvumas;
3. naudota RAM – atminties panaudojimo pikas;
4. apdorojamų politikų kiekis.

Resursų sąnaudos atvaizduojamos grafikais. Tai padeda įžvelgti priklausomybes ir koreliacijas tarp tam tikrų metrikų. Pavyzdžiui, 4.3 paveiksle pavaizduota, kaip politikų skaičius koreliuoja su veikimo laiku.



4.3 pav. Politikų skaičiaus koreliacija nuo veikimo laiko

Šis sklaidos grafikas su regresine linija parodo, kaip bendras veikimo laikas (sek.) kinta priklausomai nuo politikų skaičiaus. Nors taškai nėra griežtai išsidėstę pagal tendenciją, matyti švelni didėjimo tendencija – kuo daugiau politikų, tuo ilgesnis bendras modelio veikimo laikas, t. y. modelis dirba šiek tiek ilgiau, kai reikia apdoroti daugiau taisyklių. Taip pat 4.4 paveiksle galima įžvelgti resursų sąnaudos skirtumus tarp bazinių testų ir kritinių testų.



4.4 pav. Laiko, RAM, CPU resursų sąnaudų vidurkiai pagal testų grupes

Šis stulpelinis grafikas palygina trijų resursų rodiklių vidurkius tarp bazinių ir kritinių testų grupių:

bendro vykdymo laiko, CPU apkrovos ir RAM piko. Nors kritinių testų politikos buvo sudėtingesnės, matyti, kad jų apdorojimas užtruko net trumpiau nei bazinių testų, o CPU apkrova buvo šiek tiek aukštesnė. RAM sunaudojimas abiejose grupėse išliko stabilus ir beveik nesiskyrė. Tai reiškia, kad modelis geba efektyviai paskirstyti resursus, greičiau apdorodamas net sudėtingas politikos struktūras be didesnės atminties apkrovos. Nėra ryškaus resursų panaudojimo skirtumo tarp bazinių ir kritinių testų.

Apibendrinant galima teigti, kad modelis efektyviai interpretuoja tiek bazines, tiek sudėtingesnes (kritines) politikas. Nepaisant sudėtingumo, kritiniai testai buvo įvykdyti greičiau, nors pareikalavo didesnės CPU apkrovos. Tai rodo, kad modelis geba optimizuoti laiką, mainais į intensyvesnį resursų naudojimą. RAM sunaudojimas išliko stabilus visų testų metu, o veikimo laikas šiek tiek augo didėjant politikų kiekiui, tačiau augimas buvo nedidelis, todėl didesnis politikų skaičius nedaro didelės įtakos modelio veikimo laikui.

4.3.4. Modelio rezultato saugumo patikrinimas realiomis sąlygomis

Šiame eksperimente buvo analizuojamas praktinis NLP modelio sugeneruotų saugumo politikų efektyvumas. Buvo iškelta hipotezė, kad modelio sugeneruotas saugumo politikų skriptas, užtikrina realią kompiuterio apsaugą nuo tipinių grėsmių. Saugumo vertinimui buvo panaudoti du įrankiai:

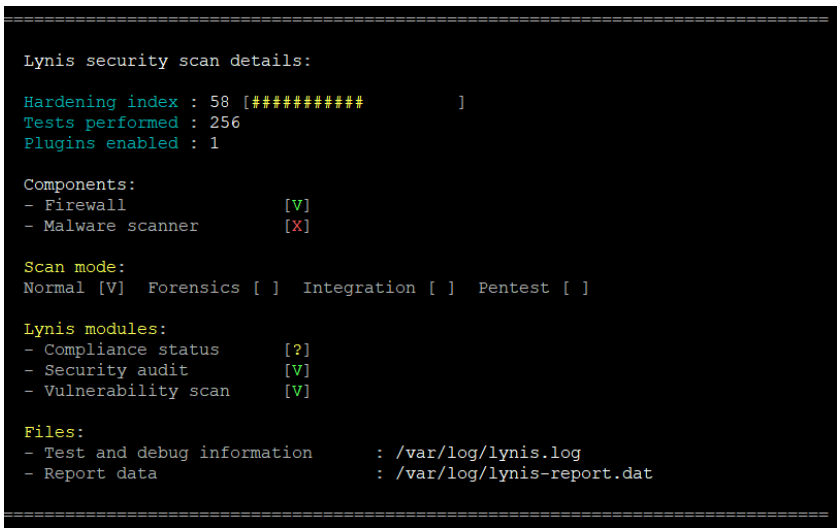
1. „Lynis“ – saugos audito įrankis;
2. „Nmap“ – tinklo saugos analizavimo įrankis.

Su šiais įrankiais tikrinama, ar įdiegti nustatymai nepalieka saugumo spragų, ar paslaugos tinkamai ribojamos, ar sistema atitinka bazinius saugos reikalavimus.

Tyrimas buvo atliktas PC-02 virtualioje mašinoje („Ubuntu Server 24.04 OS“, 4GB RAM, 4 virtualūs procesoriai), į kurią buvo su „Ansible“ sudiegtas saugos skriptas (žr. 3 priedą) pagal testinį IT saugos politikų dokumentą (žr. 1 priedą).

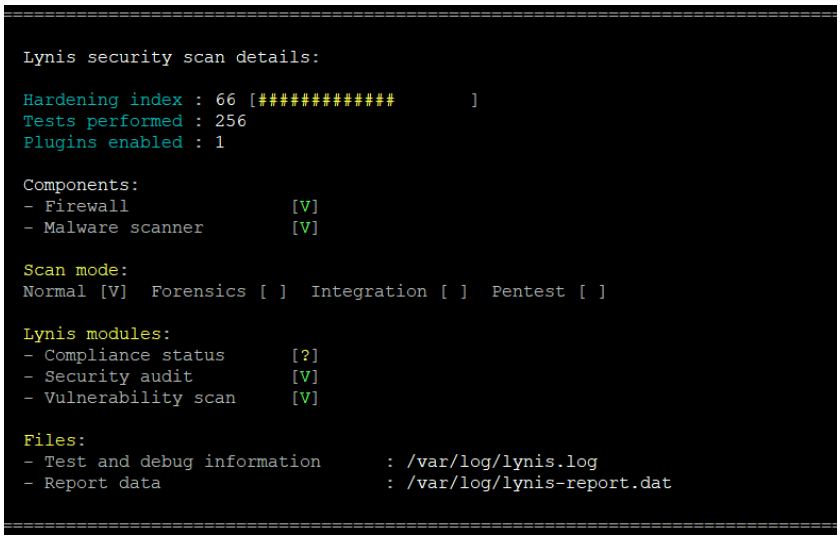
Pirmasis testas buvo panaudojant „Lynis“ saugos auditavimo įrankį, kuris įvertina sistemą (0-100) balais pagal saugumo kriterijus, tokius kaip: vartotojų teisių valdymas, paslaugų ribojimas, slaptažodžių politika. Šis įrankis atlieka 256 testus, kuriuose tikrina įvairias konfigūracijas, tačiau normalu, kad įrenginys neatitinka visų konfigūracijų, nes ne visos yra kritinės.

Prieš įdiegiant saugos skriptą, švariame PC-02 kompiuteryje „Lynis“, atlikęs testus, įvertino sistemos saugumą 58 balais. Rezultatas pavaizduotas 4.5 paveiksliuke.



4.5 pav. „Lynis“ saugumo įvertinimas prieš saugumo politikų įdiegimą

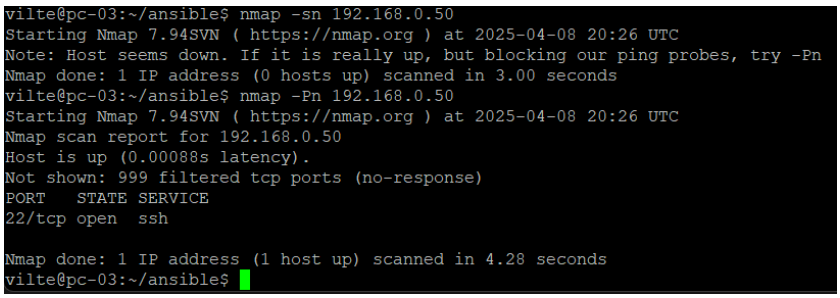
Po saugos skripto paleidimo ir sudiegimo, PC-02 kompiuteris buvo įvertintas 66 balais. Rezultatas pavaizduotas 4.6 paveiksle. Tai reiškia, kad sistemos saugumas pakilo 8 balais.



4.6 pav. „Lynis“ saugumo įvertinimas po saugumo politikų įdiegimo

Taigi, sugeneruoto skripto sudiegimas kompiuteryje tikrai pagerino sistemos saugumą. Tačiau, norint aukštesnio saugumo lygio, reiktų įdiegti daugiau politikų, kurios dar labiau pakeltų saugumo įvertį.

Antrasis tyrimas buvo atliekamas naudojant „Nmap“ įrankį, su kuriuo buvo vertinamas tinklo saugumas po sugeneruoto skripto sudiegimo. 4.7 paveiksle pavaizduota, kaip suveikė ugniasienės nustatymai: ribojimas ICMP užklausoms, blokuojamos visos užklausos, išskyrus 22 prievado SSH paslauga. Kompiuteris dabar tapo atsparus PING užklausoms ir atvirų prievadų pažeidžiamumui.



4.7 pav. „Nmap“ tinklo skenavimo rezultatai

Apibendrinant galima teigti, kad sudiegus siūlomo modelio sugeneruotą skriptą, kuriame sudiegiamos nurodytos IT saugos politikos, kompiuteris iš tiesų tampa saugesnis, atsparesnis kibernetinėms atakoms. Įrenginio saugumas priklauso nuo nurodytų IT saugos politikų, todėl svarbu atsižvelgti į skripto komandų kiekį.

4.4. Tyrimo rezultatai

Atliktas tyrimas parodė, kad siūlomas automatizuotas IT saugos politikos diegimo modelis pasižymi aukštu efektyvumu, ypač aiškiai struktūrizuotų politikų atveju. Kiekybiniai duomenys patvirtino, kad modelis ženkliai sumažina įgyvendinimo laiką (H1) ir žmogiškųjų klaidų tikimybę (H2), o NLP analizės tikslumas XML lygmenyje buvo itin aukštas (H3). Visgi, kritiniuose dokumentuose pastebėta, kad komandų generavimo etapas (veiksmų žemėlapis) yra pagrindinė silpnoji vieta, kuri lemia sumažėjusį galutinį tikslumą (dalinis H3 nepasitvirtinimas). Saugumo testai, atlikti su „Lynis“ ir „Nmap“, parodė, kad modelio sugeneruoti skriptai realiai sustiprina sistemų saugą (H4). Apibendrinant, hipotezės H1, H2 ir H4 buvo patvirtintos, o H3 pasitvirtino iš dalies – tai rodo tolimesnės veiksmų žemėlapio logikos plėtros būtinybę.

4.5. Išvados

Atlikus automatizuoto IT saugos politikos diegimo modelio kokybinį ir kiekybinį tyrimą, pateikiamos išvados, kurios atskleidžia modelio tikslumą, efektyvumą, stipriąsias puses bei tobulintinas sritis:

1. Baziniuose testuose modelis pasiekė 88 % tikslumą, teisingai sugeneruodamas 63 iš 72 galutinių „Bash“ komandų. Tai rodo, kad NLP analizė ir konvertavimo logika veikia tinkamai, kai įvesties failas yra struktūriškai teisingas.
2. Kritiniuose testuose tikslumas pasiekė tik 25 % (4 iš 16 politikų buvo teisingos), kas rodo, kad modelis yra jautrus dviprasmybėms, prieštaravimams sakinyje ir netinkamai dokumento struktūrai.
3. XML generavimo etapas veikia tiksliau nei „Bash“ komandų generavimo etapas – reikia tobulinti veiksmų žemėlapio funkciją ir komandų generavimo algoritmą.
4. Po automatinio politikų įdiegimo įrenginių saugumo balas („Lynis“ vertinimu) pagerėjo nuo 58 iki 66 – tai įrodo, kad modelio sugeneruoti skriptai realiai sustiprino įmonės įrenginio saugumą.

5. Dažniausios modelio klaidos buvo netaisyklinga gramatika, netinkama sakinių segmentacija, klaidinga interpretacija ir komandų generavimo funkcionalumas (veiksmų žemėlapio ribotumas).

6. Modelio veikimui reikalingi techniniai resursai buvo stabilūs – CPU ir RAM sąnaudos išliko nedidelės, pastovios ir nestipriai priklausančios nuo politikų kiekio.

Išvados

Šiame darbe buvo nagrinėjama aktuali IT saugos politikos automatizuoto valdymo problema, kuri yra ypač svarbi šiuolaikinėms organizacijoms, siekiančioms užtikrinti informacijos saugą. Atlikus analizę, nustatyta, kad daugelis tradicinių modelių sunkiai pritaikomi dinamiškoje IT architektūros aplinkoje, o rankinis politikų diegimas yra lėtas, neefektyvus ir dažnai sukuria žmogiškąsias klaidas, dėl kurių reali organizacijos situacija ne visada atitinka nustatytas politikas.

Remiantis atlikta analize, buvo suprojektuotas ir sukurtas automatizuotas IT saugos politikų valdymo modelis, integruojantis natūralios kalbos apdorojimo (NLP) technologijas ir „politika kaip kodas" (angl. „Policy-as-Code") principą. NLP taikymas IT saugos politikų valdyme yra inovatyvi ir dar išsamiai neištyrinėta sritis, todėl šis tyrimas atskleidė naujas galimybes sumažinti žmogiškųjų klaidų riziką bei optimizuoti politikų įgyvendinimo procesus. Sukurtas modelis geba perskaityti natūralia lietuvių kalba parašytus IT saugos politikos dokumentus, automatiškai juos konvertuoti į komandų skriptus ir centralizuotai juos sudiegti organizacijos įrenginiuose, panaudojant „Ansible" įrankį. Visa tai užtikrina pilną organizacijos politikos atitikties ciklą – nuo IT saugos politikos dokumento iki sukonfigūruotų saugių įrenginių.

Atlikus siūlomo modelio prototipo testavimą, nustatyta, kad šis modelis ženkliai sumažina žmogiškųjų klaidų tikimybę, pagreitina saugos politikų diegimą ir užtikrina atitiktį organizacijos politikoms. Eksperimento rezultatai parodė pakankamai aukštą siūlomo modelio tikslumo lygį (88%), panaudojant įvairius IT saugos politikos dokumentus. Taip pat eksperimentinėje aplinkoje buvo nustatyta, kad įrenginių saugumo lygis padidėja, sudiegus siūlomo modelio išvesties skriptus.

Vis dėlto buvo pastebėti tam tikri iššūkiai: NLP algoritmų sunkumai, interpretuojant sudėtingesnius ar taisyklių neatitinkančius sakinius, ir ribotas skripto generavimo etapas, priklausantis nuo iš anksto nustatyto bazinių komandų rinkinio. Tolimesniam modelio tobulinimui rekomenduojama giliau tirti NLP algoritmus ir plėsti jų taikymą kuo įvairesniems įvestiems dokumentams, taip pat pildyti bazinių komandų rinkinį, siekiant užtikrinti modelio universalumą.

Apibendrinant galima teigti, kad sukurtas automatizuotas IT saugos politikų valdymo modelis yra perspektyvus sprendimas šiuolaikinėms organizacijoms, siekiančioms užtikrinti aukštą informacijos saugumo lygį, efektyviai panaudojant šiuolaikines technologijas ir mažinant žmogiškųjų klaidų riziką. Modelis gali būti sėkmingai pritaikomas įvairiose organizacijose, o tolimesni tyrimai galėtų dar labiau pagerinti jo patikimumą ir lankstumą.

Literatūra

[1] Md Haris Uddin Sharif and Mehmood Ali Mohammed. A literature review of financial losses statistics for cyber security and future trend. World Journal of Advanced Research and Reviews, 2022.
https://wjarr.com/sites/default/files/WJARR-2022-0573.pdf.

[2] Adebola Folorunso, Viqaruddin Mohammed, Ifeoluwa Wada, and Bunmi Samue. The impact of iso security standards on enhancing cybersecurity posture in organizations. World Journal of Advanced Research and Reviews, 2024.
https://www.researchgate.net/profile/Ifeoluwa-Wada/publication/385427490_The_impact_of_ISO_security_standards_on_enhancing_cybersecurity_posture_in_organizations/links/67a5b926461fb56424ccb620/The-impact-of-ISO-security-standards-on-enhancing-cybersecurity-posture-in-org
pdf.

[3] Kholoud Mahmoud. Wannacry impact and how adoption of cyber crime measures such as iso 27001 can be beneficial. Union Nikola Tesla University, 2024.
https://www.researchgate.net/publication/381305236_WannaCry_Impact_and_How_Adoption_of_Cyber_Crime_Measures_Such_AS_ISO_27001_Can_Be_Beneficial.

[4] Inc Fortinet. Cia triad. Fortinet.
https://www.fortinet.com/resources/cyberglossary/cia-triad.

[5] Anas Irsheida, Ahmad Murada, Mohammad AlNajdawia, and Abdullah Qusef. Information security risk management models for cloud hosted systems: A comparative study. Procedia Computer Science, 2022.
https://www.sciencedirect.com/science/article/pii/S1877050922007633.

[6] CISSP. Chapter 5: Security models and architecture. CISSP Certification All-in-One Exam Guide.
https://media.techtarget.com/searchSecurity/downloads/29667C05.pdf.

[7] Edison Wazoel Lubua. A discussion of information security models and their application. Uganda Technology and Management University, 2022.
https://www.researchgate.net/profile/Edison-Lubua/publication/359761118_A_Discussion_of_Information_Security_Models_and_their_application/links/624d7e14d726197cfd3f9393/A-Discussion-of-Information-Security-Models-and-their-application.pdf.

[8] Sanket Devlekar and Vidyavati Ramteke. Identity and access management: High-level conceptual framework. Revista Geintec, 2021.
https://revistageintec.net/old/wp-content/uploads/2022/03/2511.pdf.

[9] Maria Penelova. Access control models. Cybernetics and Information Technologies, 2021. https://sciendo.com/article/10.2478/cait-2021-0044.

[10] Shakti Dubey and Dr. P. K. Rai. A review of cloud service security with various access control methods. International Journal of Computer Science and Mobile Computing, 2021. https://d1wqtxts1xzle7.cloudfront.net/66152512/V10I3202106-libre.pdf?1617222877=&response-content-disposition=inline%3B+filename%3DA_Review_of_Cloud_Service_Security_with.pdf&Expires=1705410764&Signature=fvILLGsCR8Bpkg9aADZGVhRDwzbXxSw3DyKJqsHQVL65rhDVUEfhEMta-rwXzbkUYh-6oTiwClnn0k2_&Key-Pair-Id=APKAJLOHF5GGSLRBV4ZA.

[11] Joseph C. Brickley and Kutub Thakur. Policy of least privilege and segregation of duties, their deployment, application and effectiveness. International Journal of Cyber-Security and Digital Forensics, 2021. https://www.researchgate.net/profile/Joseph_Brickley/publication/360492447_Policy_of_Least_Privilege_and_Segregation_of_Duties_their_Deployment_Application_Effectiveness/links/627a67b7973bbb29cc721b3c/Policy-of-Least-Privilege-and-Segregation-of-Duties-their-Deployment-Applicati pdf.

[12] Qigui Yao, Qi Wang, Xiaojian Zhang, and Jiaxuan Fei. Dynamic access control and authorization system based on zero-trust architecture. Association for Computing Machinery, 2021. https://dl.acm.org/doi/abs/10.1145/3437802.3437824.

[13] Noe M. Yungaicela-Naula, Cesar Vargas-Rosales, Jesús Arturo Pérez-Díaz, and Mahdi Zareei. Towards security automation in software defined networks. Computer Communications Journal, 2022. https://www.sciencedirect.com/science/article/abs/pii/S0140366421004436.

[14] Mohammed Daffalla Elradi. Ansible: A reliable tool for automation. Sanderman Publishing House, 2023. https://www.sandermanpub.net/uploads/20230801/1bed84d124606aaf9539846a92290c1d.pdf.

[15] Aleksandra Pyrkosz and Sabina Szymoniak. Simplifying security processes in large organizations while maintaining an appropriate level of security. Centrum Rzeczoznawstwa Budowlanego, 2024. https://doi.org/10.37105/iboa.186.

[16] Hayet Brabra, Achraf Mtibaa, Walid Gaaloul, and Boualem Benatallah. Toward higher-level abstractions based on state machine for cloud resources elasticity. Information Systems, 2020. https://www.sciencedirect.com/science/article/abs/pii/S0306437919305022.

[17] Simon Yusuf Enoch, Jin B. Hong, and Dong Seong Kim. Security modelling and assessment of modern networks using time independent graphical security models. Science Direct, 2019. https://www.sciencedirect.com/science/article/pii/S108480451930308X.

[18] Vaishak Sreejith, Pudukaram Urjith Reddy, Bandari Harshith Rao, and Sakthi Abirami Balakrishnan. Hybrid network security model for small and mid sized enterprises. 2022 Third International Conference on Intelligent Computing Instrumentation and Control Technologies (ICICICT), 2022.
https://ieeexplore.ieee.org/abstract/document/9917896.

[19] Felix Chukwuma Aguboshim, Joy Ebere Ezeife, and Ifeyinwa Nkemdilim Obiokafor. Managing organization information security systems, conflicts, and integrity for sustainable africa transformation. World Journal of Advanced Research and Reviews, 2022.
https://wjarr.com/sites/default/files/WJARR-2022-0425.pdf.

[20] Daniele Bringhenti, Guido Marchetto, Riccardo Sisto, and Fulvio Valenza. Automation for network security configuration: State of the art and research trends. Association for Computing Machinery, 2023.
https://doi.org/10.1145/3616401.

[21] Calvin Nobles. Stress, burnout, and security fatigue in cybersecurity: A human factors problem. Holistica Journal of Business and Public Administration, 2022.
https://sciendo.com/pdf/10.2478/hjbpa-2022-0003.

[22] William J. Triplett. Addressing human factors in cybersecurity leadership. Journal of Cybersecurity and Privacy, 2022.
https://doi.org/10.3390/jcp2030029.

[23] Oluwatosin Ilori, Nelly Tochi Nwosu, and Henry Nwapali Ndidi Naiho. Third-party vendor risks in it security: A comprehensive audit review and mitigation strategies. World Journal of Advanced Research and Reviews, 2024.
https://wjarr.com/sites/default/files/WJARR-2024-1727.pdf.

[24] Sina Ahmadi. Autonomous identity-based threat segmentation in zero trust architectures. The National Coalition of Independent Scholars, 2025.
https://arxiv.org/pdf/2501.06281.

[25] Daniel Foo. Policy as code. Medium, 2023.
https://danielfoo.medium.com/policy-as-code-fa7cba74d128.

[26] Charan Shankar Kummarapurugu. Enhancing serverless computing security in multi-cloud environments: Integrating policy-ascode, automated compliance, and dynamic access controls. International Journal of Innovative Research in Engineering and Multidisciplinary Physical Sciences, 2022.
https://www.researchgate.net/publication/389747418_Enhancing_Serverless_Computing_Security_in_Multi-Cloud_Environments_Integrating_Policy-as-_Code_Automated_Compliance_and_Dynamic_Access_Controls.

[27] Areej Alhogail and Afrah Alsabih. Applying machine learning and natural language processing to detect phishing email. Computers and Security, 2021.
https://www.sciencedirect.com/science/article/abs/pii/S0167404821002388.

[28] Pinyi Shi, Yongwook Song, Zongming Fei, and James Griffioen. Checking network security policy violations via natural language questions. International Conference on Computer Communications and Networks (ICCCN), 2021.
https://ieeexplore.ieee.org/abstract/document/9522325.

[29] Saiful Islam, Md. Mokammel Haque, and Abu Naser Mohammad Rezaul Karim. A rule-based machine learning model for financial fraud detection. International Journal of Electrical and Computer Engineering (IJECE), 2024.
https://core.ac.uk/download/pdf/591354087.pdf.

[30] Vijay Srinivas Tida and Sonya Hsu. Universal spam detection using transfer learning of bert model. Cornell University, 2022.
https://arxiv.org/abs/2202.03480.

Priedai

1 priedas. Pavyzdinis organizacijos IT saugos politikos dokumentas

1. Prisijungimo valdymas:
1.1. Slaptažodis turi 8 simbolių ilgį.
1.2. Slaptažodis galioja 365 dienas.
1.3. Paskyra yra blokuojama po 10 nesėkmingų bandymų.
1.4. MFA autentifikacija yra įjungta.
1.5. Paskyrą turi blokuoti 60 sekundėms.
2. Tinklo apsauga:
2.1. Reikia blokuoti užklausas iš adreso [192.168.0.30].
2.2. Ugniasienėje visi įeinantys prievadai yra išjungti.
2.3. Prievadas 22 yra įjungtas ugniasienėje.
2.4. Reikia blokuoti „ICMP" užklausas.
2.5. Ugniasienė yra įjungta.
3. Darbo vietos apsauga:
3.1. Kompiuteris automatiškai užrakinamas po 10 minučių.
3.2. USB laikmenos yra išjungtos.
3.3. Automatiniai sistemos atnaujinimai yra įjungti.
3.4. Antivirusinė yra įjungta.

2 priedas. Konvertuotas pavyzdinis politikų dokumentas į XML failą

```
<?xml version="1.0" ?>
<IT_saugos_politikos>
  <slaptažodis_politika>
    <Objektas>slaptažodis</Objektas>
    <Veiksmas>turėti</Veiksmas>
    <Parametras>
      8
      <Vienetas>ilgis</Vienetas>
    </Parametras>
  </slaptažodis_politika>
  <slaptažodis_politika>
    <Objektas>slaptažodis</Objektas>
    <Veiksmas>galioti</Veiksmas>
    <Parametras>
      365
      <Vienetas>diena</Vienetas>
    </Parametras>
  </slaptažodis_politika>
  <paskyra_politika>
    <Objektas>paskyra</Objektas>
    <Veiksmas>blokuoti</Veiksmas>
    <Parametras>
      10
      <Vienetas>bandymas</Vienetas>
    </Parametras>
  </paskyra_politika>
  <autentifikacija_politika>
    <Objektas>autentifikacija</Objektas>
    <Veiksmas>įjungti</Veiksmas>
  </autentifikacija_politika>
  <paskyra_politika>
    <Objektas>paskyra</Objektas>
    <Veiksmas>blokuoti</Veiksmas>
    <Parametras>
      60
      <Vienetas>sekundė</Vienetas>
    </Parametras>
  </paskyra_politika>
  <užklausa_politika>
```

```
  <Objektas>užklausa</Objektas>
  <Veiksmas>blokuoti</Veiksmas>
  <Parametras>
    192.168.0.30
    <Vienetas>adresas</Vienetas>
  </Parametras>
</užklausa_politika>
<prievadas_politika>
  <Objektas>prievadas</Objektas>
  <Veiksmas>išjungti</Veiksmas>
</prievadas_politika>
<prievadas_politika>
  <Objektas>prievadas</Objektas>
  <Veiksmas>įjungti</Veiksmas>
  <Parametras>22</Parametras>
</prievadas_politika>
<icmp_politika>
  <Objektas>icmp</Objektas>
  <Veiksmas>blokuoti</Veiksmas>
</icmp_politika>
<ugniasienė_politika>
  <Objektas>ugniasienė</Objektas>
  <Veiksmas>įjungti</Veiksmas>
</ugniasienė_politika>
<kompiuteris_politika>
  <Objektas>kompiuteris</Objektas>
  <Veiksmas>užrakinti</Veiksmas>
  <Parametras>
    10
    <Vienetas>minutė</Vienetas>
  </Parametras>
</kompiuteris_politika>
<usb_politika>
  <Objektas>usb</Objektas>
  <Veiksmas>išjungti</Veiksmas>
</usb_politika>
<atnaujinimas_politika>
  <Objektas>atnaujinimas</Objektas>
  <Veiksmas>įjungti</Veiksmas>
</atnaujinimas_politika>
<antivirusinė_politika>
  <Objektas>antivirusinė</Objektas>
  <Veiksmas>įjungti</Veiksmas>
</antivirusinė_politika>
```

```
</IT_saugos_politikos>
```

3 priedas. Sugeneruotas komandų skriptas pagal organizacijos politikų dokumentą

```
#!/bin/bash
# Sugeneruotas Bash skriptas pagal IT saugos politikos dokumenta
sudo sed -i 's/^PASS_MIN_LEN.*/PASS_MIN_LEN   8/' /etc/login.defs
for user in $(awk -F: '$3 >= 1000 {print $1}' /etc/passwd); do sudo chage
    --maxdays 365 $user; done
echo 'deny = 10' | sudo tee -a /etc/security/faillock.conf
sudo apt update && sudo apt install -y libpam-google-authenticator &&
   echo 'auth required pam_google_authenticator.so nullok' | sudo tee -a
   /etc/pam.d/sshd
echo 'fail_interval = 60' | sudo tee -a /etc/security/faillock.conf
sudo ufw deny from 192.168.0.30
sudo ufw default deny incoming
sudo ufw allow 22
sudo iptables -A INPUT -p icmp --icmp-type echo-request -j DROP
sudo ufw --force enable
echo "TMOUT=600" | sudo tee -a /etc/profile
echo 'blacklist usb-storage' | sudo tee /etc/modprobe.d/usb-storage.conf
   && sudo update-initramfs -u
sudo DEBIAN_FRONTEND=noninteractive apt update -y && sudo DEBIAN_FRONTEND
   =noninteractive apt install -y unattended-upgrades && sudo
   DEBIAN_FRONTEND=noninteractive dpkg-reconfigure unattended-upgrades
sudo apt install -y clamav && sudo systemctl start clamav-freshclam
```