VILNIAUS UNIVERSITETAS

MATEMATIKOS IR INFORMATIKOS FAKULTETAS

PROGRAMŲ SISTEMŲ STUDIJŲ PROGRAMA

Bakalauro baigiamasis darbas

Varžymosi principais grįstų atakų aptikimas naudojant paaiškinamo dirbtinio intelekto metodą kenkėjiškų programų obfuskacijos atveju

Defense Against Adversarial Malware Obfuscation Attacks Using Explainable Artificial Intelligence

Liudas Kasperavičius

Darbo vadovas : prof. dr. Olga Kurasova

Recenzentas : assoc. prof. Linas Petkevičius

Vilnius
2025

Santrauka

Šiame darbe nagrinėjamos varžymosi principais pagrįstos atakos prieš kenkėjiškų programų detektorius bei tokių atakų aptikimo ir apsisaugojimo nuo jų strategijos. Siekiant patobulinti jau esamus aptikimo metodus bei pritaikyti juos bet kokiems požymių vektoriams, siūloma sujungti *MCA* dimensijų mažinimo metodą ir *LIME* – ML modelių sprendimų paaiškinimo metodą, kurie atitinka mokslinėje literatūroje minimas perspektyviausias varžymosi principais pagrįstų atakų aptikimo strategijas. Atliekamas siūlomo metodo tyrimas nustatant jo tikslumą bei lyginant su prieš tai minėtomis varžymosi principais pagrįstų atakų aptikimo strategijomis. Tyrimui pasitelkiamas *MalGAN* karkasas kaip tokių atakų generatorius, kadangi šio karkaso naudojami dvejetainiai požymių vektoriai turėtų kelti daugiausia sunkumų esamiems apsisaugojimo nuo varžymosi principais pagrįstų atakų metodams. Nustatyta, jog pasiūlytas metodas geba tiek apsaugoti nuo, tiek aptikti varžymosi principais pagrįstas atakas efektyviau nei esami metodai.

Summary

This work analyses *adversarial attacks* against malware detectors as well as strategies of detection and defense against such attacks. With the goal to improve already existing detection methods and adapt them for use with any feature vectors, this work explores combining *MCA* dimensionality reduction method and *LIME* – a method for explaining ML models' decisions as these are implementations of the most perspective strategies of defense against adversarial attacks mentioned in scientific literature. A study is conducted to determine the accuracy of the suggested method by comparing it with the aforementioned strategies of defense against *adversarial attacks*. *MalGAN* framework is used for this study as the generator of such attacks since the binary vectors used by it should cause the most difficulty for existing strategies of defense. Experimental results show that the suggested method is able to both defend against and detect *adversarial attacks* more effectively and consistently than currently used methods.

Turinys

Terminų žodynas

Inercija (*angl. Inertia*). Dispersijos dalis, kurią „paaiškina“ PCA komponentė.

Karkasas (*angl. Framework*). Nurodo specifines technologijas, naudojamus požymius ir perturbacijas, siekiamus tikslus AE generacijai. Skirtas apibrėžti procesą ir įrankius, kuriuos naudojant būtų galima generuoti nurodytų tikslų siekiančius AE.

Maišymo funkcija (*angl. Hash function*). Tai funkcija $f : \{0,1\}^* \to \{0,1\}^m$. Naudojama, kai iš begalinės įvesčių erdvės norima gauti fiksuoto dydžio (m) išvestį.

Nulinės sumos žaidimas (*angl. Zero-Sum Game*). Dviejų žaidėjų žaidimas, kuriame galimas vienas laimėtojas. Laimėtojo laimėta suma yra lygi pralaimėtojo pralaimėtai sumai.

Pėdsakas (*angl. Signature*). Programos struktūros ir požymių santrauka, beveik unikaliai identifikuojanti programą (pvz., maišymo funkcija).

Q-Funkcija (*angl. Q-Function*). $Q : S \times A \to \mathbb{R}$, čia S – galimų būsenų erdvė (*angl. State Space*), A – galimų veiksmų erdvė (*angl. Action Space*).

Sprendimų priėmimo riba (*angl. Decision Boundary*). Paprasčiausiems ML modeliams tai yra kreivė plokštumoje. Sudėtingesniems – daugiadimensiniams modeliams – daugdara (*angl. manifold*).

Strategija (*angl. Policy*). Tai funkcija $\pi : S \times A \to \{0,1\}$, čia S – galimų būsenų erdvė (*angl. State Space*), A – galimų veiksmų erdvė (*angl. Action Space*). Šią funkciją RL modelis „išmoksta“ mokymosi metu.

Surogatinis Modelis (*angl. Surrogate Model*). ML modelis, aproksimuojantis kitą ML modelį, kurio parametrai (svoriai) nėra žinomi.

Varžymosi principais grįstas mokymasis (*angl. Adversarial Retraining*). ML modelio mokymas naudojant AE kaip mokymosi duomenis. Viena iš apsisaugojimo nuo varžymosi principais pagrįstų atakų strategijų.

Varžymosi principais pagrįstos atakos (*angl. Adversarial Attacks*). Tai atakos, pritaikytos „apgauti“ ML klasifikatorius.

„Juodos dėžės“ ataka (*angl. Black-Box Attack*). Varžymosi principais pagrįstos atakos atvejis, kai atakuojamo ML modelio parametrai bei klasifikacijos tikimybiniai įverčiai nėra žinomi.

Santrumpos

AE – varžymosi principais pagrįstomis atakomis obfuskuoti kenkėjiško kodo pavyzdžiai (*angl. Adversarial Examples*).

API – *angl. Application Programming Interface*.

CNN – *angl. Convolutional Neural Network*.

DI – dirbtinis intelektas (*angl. artificial Intelligence*).

DLL – dinamiškai susieta biblioteka (*angl. Dynamic Link Library*).

GA – genetiniais algoritmais pagrįstas ML modelis (*angl. Genetic Algorithm*).

GAN – generatyviniai priešiški tinklai (*angl. Generative Adversarial Networks*).

GBDT – *angl. Gradient Boosted Decision Trees*.

IDS – *angl. Intrusion Detection System*.

KNN – *angl. K-Nearest Neighbours*.

LIME – *angl. Local Interpretable Model-agnostic Explanations* – lokalūs, interpretuojami ML modelių išvesčių paaiškinimai.

MCA – *angl. Multiple Correspondence Analysis*.

ML – mašininis mokymasis (*angl. Machine Learning*).

NLP – skaitmeninis natūraliosios kalbos apdorojimas (*angl. Natural Language Processing*).

PCA – *angl. Principal Component Analysis*.

PE – *angl. Portable Executable*.

RL – skatinamasis mokymasis (*angl. Reinforcement Learning*).

SVM – *angl. Support Vector Machine*.

XAI – paaiškinamas dirbtinis intelektas (*angl. Explainable Artificial Intelligence*).

Įvadas

Pastaraisiais metais kenkėjiškas kodas ir programos kuriamos itin sparčiai (∼450000 kenkėjiškų programų per dieną 2024 m. AV-TEST[1] duomenimis). Kenkėjiško kodo aptikimo programos, kurios tradiciškai remiasi programų pėdsakais, nespėja atnaujinti pėdsakų duomenų bazių pakankamai greitai. Dėl to dirbtinio intelekto (DI), tiksliau mašininio mokymosi (ML), naudojimas kenkėjiškų programų ar kenkėjiško kodo aptikimo srityje tapo itin populiarus [DCB+21]. Tačiau ML modeliai, nors ir geba aptikti kenkėjiškas programas iš naujų, dar nematytų, duomenų, yra pažeidžiami varžymosi principais pagrįstoms atakoms [CSD19; HT17; RSR+18; ZHZ+22]. Šių atakų principas yra ML modelio – klasifikatoriaus – sprendimų priėmimo ribos radimas – žinant šią ribą pakanka pakeisti kenkėjiškos programos veikimą taip, kad ML modelis priimtų sprendimą klasifikuoti ją kaip nekenksmingą [DCB+21]. Nustatyta, jog šią ribą galima rasti tiek žinant klasifikatoriaus parametrus, tiek jų nežinant ir net turint labai ribotą prieigą prie klasifikatoriaus rezultatų (pvz., klasifikacijos rezultatą be tikimybių – tokios sąlygos vadinamos „juodos dėžės“ atvejai) [FWL+19].

Vis tik varžymosi principais pagrįstos atakos nėra neįveikiamos. Nuolat kuriami nauji jų aptikimo metodai, tokie kaip varžymosi principais grįstas mokymasis, gradientų slėpimas ir kt. Kiekvienas metodas turi savų stiprybių ir silpnybių bei dažniausiai remiasi viena iš specifinio ML modelio įgyvendinimo savybių, kitaip tariant, nėra vieno geriausio, tinkamiausio ar teoriškai teisingo varžymosi principais pagrįstų atakų aptikimo metodo. Tiksliau, nėra pačių AE konstravimo teorinio modelio, dėl šio proceso kompleksiškumo, tad jų aptikimo strategijos teorinis modelis taip pat nėra žinomas [CAD+21]. Šiame darbe siekiama generalizuoti AE aptikimą apjungiant panašiame kenkėjiškų programų aptikimo kontekste naudojamą LIME [RSG16] metodą ir kitas mokslinėje literatūroje aprašytas technikas.

Tikslas – pritaikyti LIME metodą sėkmingų varžymosi principais grįstų atakų aptikimui prieš kenkėjiškų programų detektorius vertinant bet kokius požymius.

Uždaviniai:

1. Apžvelgti kenkėjiško kodo obfuskacijos metodus bei apsisaugojimo nuo jų strategijas.
2. Pritaikyti varžymosi principais pagrįstų atakų aptikimą dvejetainius požymių vektorius naudojantiems modeliams taikant dimensijų mažinimo metodus.
3. Sukurti klasifikavimo proceso praplėtimą į jį įtraukiant varžymosi principais pagrįstos atakos aptikimą ir paaiškinimą su LIME .
4. Ištirti praplėsto klasifikavimo proceso tikslumą (*angl. accuracy*).

[1] https://www.av-test.org/en/statistics/malware

1. Literatūros apžvalga

1.1. Naudojami kenkėjiškų programų požymiai

Varžymosi principais pagrįstos atakos taikosi į ML modeliais paremtus kenkėjiškų programų detektorius. Šie detektoriai yra klasifikatoriai – pateiktį (programą) klasifikuoja kaip kenkėjišką (*angl. malicious*) arba nekenkėjišką (*angl. benign*). Kadangi programos nėra fiksuoto dydžio, klasifikatoriai remiasi programų požymiais, kurie gaunami atliekant požymių ištraukimą (*angl. feature extraction*). Laikoma, jog „juodos dėžės“ atvejais sužinoti, kokius tiksliai požymius vertina kenkėjiškų programų detektorius, yra neįmanoma, tad karkasų apibrėžimuose, priklausomai nuo jų specifikos ir tikslų, neretai pateikiami jų vertinami programų požymiai. Šiame poskyryje išskiriami ir klasifikuojami mokslinėje literatūroje minimi požymiai.

1.1.1. PE formato programų požymiai

Išskiriami šie pagrindiniai PE formato programų požymiai:

- **DLL vardai (arba API vardai [HT17])** [ZCY+24]. PE faile turi būti nurodyti visi naudojami DLL ir jų API. Prieš pradedant mokyti ML modelį, atliekama visų turimų programų analizė ir nustatoma visų naudojamų DLL ar jų API aibė D. Tarkime $|D| = n$. Tuomet, požymių vektorius programai, naudojančiai $X \subseteq D$ DLL, bus n-matis dvejetainis vektorius, kurio i-asis elementas yra $\begin{cases} 0, \text{ jei } D_i \notin X, \\ 1, \text{ jei } D_i \in X \end{cases}$ čia D_i – i-asis D elementas.
- **PE metaduomenys** [AKF+18]. Tai visi PE formato faile esantys metaduomenys, tokie, kaip sekcijų pavadinimai, sekcijų dydžiai, *ImportTable* ir *ExportTable* metaduomenys ir kt. Formuojant požymių vektorių skaičiuojama metaduomenų maišymo funkcija.

1.1.2. Baitų lygio požymiai

Baitų lygio požymiai gali būti ištraukiami iš bet kokio formato failų. Mokslinėje literatūroje minimi šie pagrindiniai baitų lygio požymiai:

- **Prasmingų žodžių (angl. strings) kiekis** [AKF+18]. Prasmingus žodžius suprantame kaip turinčius prasmę žmogui *(angl. human readable)*. Tai gali būti URL, failų keliai *(angl. file paths)* ar registro raktų pavadinimai. Kadangi prasmingų žodžių kiekis tėra vienas skaičius, požymių vektorius dažniausiai formuojamas prijungiant ir kitus požymius.
- **Baitų/entropijos histograma** [SB15]. Specifinis metodas, užkoduojantis dažniausiai pasikartojančias baitų ir entropijos poras n dimensijų vektoriumi.
- **n-gramos** [ZZY+22]. Dažniausiai sutinkamos skaitmeniniame natūraliosios kalbos apdorojime (NLP). Tai yra n žodžių junginiai, arba, sukompiliuotų programų apdorojimo kontekste, n

baitų junginiai. Nustatant požymių vektorių, visos n-gramos surikiuojamos pagal pasikartojimą programoje mažėjimo tvarka („populiariausios“ viršuje). Iš pirmų m reikšmių sudaromas m-matis vektorius – tai ir yra požymių vektorius.

1.2. Perturbacijos

Perturbacijos – tai pagrindinis obfuskacijos metodas AE kūrimui. Perturbacijų tikslas yra pakeisti kenkėjiškos programos veikimą išsaugant originalų funkcionalumą. Perturbacijos gali būti sudėtingos ir apimti visą programą (pvz., visos programos užšifravimas ir pridėjimas prie kitos programos), semantinės (pvz., tam tikrų mašininio kodo instrukcijų keitimas į ekvivalentų rezultatą pasiekiančias) arba baitų lygio (pvz., nulinių baitų pridėjimas programos gale) [HT17]. Perturbacijų parinkimas įeina į karkaso apibrėžimą. Šiame poskyryje aptariamos mokslinėje literatūroje minimos perturbacijos.

1.2.1. Baitų lygio perturbacijos

Pačias paprasčiausias baitų lygio perturbacijas galima taikyti bet kokio formato failams, tačiau labiau prasmingos perturbacijos taikomos PE formato failams. Išskiriamos šios pagrindinės baitų lygio perturbacijos:

- ***ARBE* (*Append Random Bytes at the End*)** [FWL+19]. PE formato failo gale pridedami atsitiktiniai baitai.
- ***ARI* (*Append Random Import*)** [FWL+19]. PE formato failo *ImportAddressTable* lentelėje pridedama atsitiktinai pavadinta biblioteka su atsitiktinai pavadinta funkcija.
- ***ARS* (*Append Randomly named Section*)** [FWL+19]. PE formato failo *SectionTable* lentelėje pridedamos atsitiktinės sekcijos (sekcijos ir jų tipai yra apibrėžti PE formate).
- ***RS* (*Remove Signature*)** [FWL+19]. Sertifikato pašalinimas iš PE formato failo *Certificate-Table* lentelės.
- **Naujas įeities taškas** [AKF+18]. Prasidėjus programai, iškart peršokama nuo naujo įeities taško į originalųjį.
- ***Header Fields*** [DCB+21]. PE formato failo *PE Header* ir *Optional Header* dalių specifinių laukų keitimas (pvz., sekcijos pavadinimo keitimas [AKF+18]).
- ***Partial DOS*** [DCB+21]. PE formato failo *DOS Header* dalies pirmi 58 baitai po *MZ* skaičiaus yra nenaudojami moderniose operacinėse sistemose, tad juos galima keisti.
- ***Slack Space*** [DCB+21]. Dėl PE formato specifikos, kiekviena nauja sekcija turi prasidėti tam tikro skaičiaus, nurodyto *PE Header* dalyje, kartotiniu nuo pradžios. Kompiliatoriai šį reikalavimą išpildo sekcijų gale pridėdami tiek nulinių baitų, kiek reikia teisingam sulygiavimui pasiekti. Būtent ši nulinių baitų erdvė gali būti keičiama be jokios įtakos originaliai programai.

- ***Padding*** [DCB+21]. Nulinių baitų pridėjimas failo gale.

- ***Full DOS*** [DCB+21]. Perturbacijos esmė tokia pat, kaip ir *Partial DOS*, tik naudojami visi *DOS* dalies baitai, išskyrus *MZ* ir *PE Offset* (*Partial DOS* manipuliacijoms naudoja tik dalį tarp *MZ* ir *PE Offset*).

- ***Extend*** [DCB+21]. Pakeičiama PE formato faile *DOS* dalyje esanti *PE Offset* reikšmė į didesnę[2]. Taip padidinama (išplečiama) visa *DOS* dalis. Tolesnis perturbacijos principas yra toks pat, kaip ir *Full DOS*.

- ***Shift*** [DCB+21]. PE formato failuose kiekvienas sekcijos blokas prasideda su sekcijos vieta nuo pradžios (*angl. offset*). Tarkime ši reikšmė yra S. Sekcijos kodas pradedamas vykdyti tik nuo adreso $P + S$, kur P – programos pradžios adresas. Vadinasi, padidinus[2] S per n, atsiranda n baitų laisvos vietos iki sekcijos pradžios, kurią galima keisti be jokios įtakos programos veikimui.

1.2.2. Semantinės perturbacijos

Semantinių perturbacijų įgyvendinimas taip pat atliekamas baitų lygyje, tačiau šie pokyčiai turi aukštesnio lygio prasmę. Išskiriamos šios semantinės perturbacijos:

- **Nereikalingų DLL/API vardų požymių pridėjimas** [HT17]. PE formato faile *ImportTable* lentelėje pridedami originalios programos nenaudojami DLL/API vardai.

- ***Binary Rewriting*** [DCB+21]. Semantinis instrukcijų perrašymas. Pavyzdžiui, $A + B$ instrukcijos pakeitimas į $A - (-B)$.

[2]šios reikšmės padidinimas reiškia visos failo struktūros keitimą (*DOS* dalis yra failo pradžioje). Būtina pakeisti visų sekcijų vietas nuo pradžios (*angl. offset*) jų metaduomenyse.

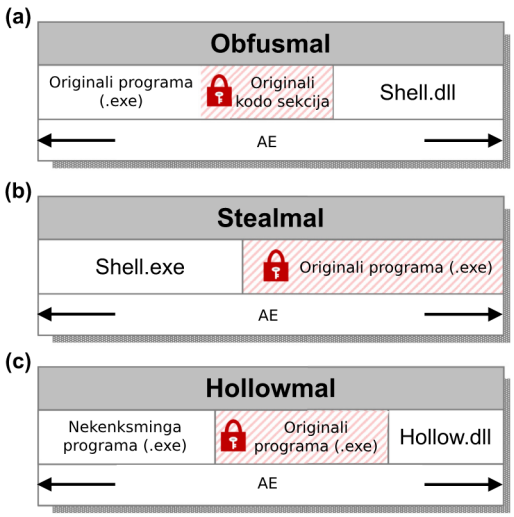
1.2.3. Kompleksinės perturbacijos

Kompleksinės perturbacijos yra pritaikomos tam tikriems tikslams. Obfuskacijos ir varžymosi principais pagrįstų atakų tikslams literatūroje minimos šios kompleksinės perturbacijos:

- ***Obfusmal*** [ZCY+24]. Užšifruojama originalios programos kodo sekcija. Sukuriama ir originalios programos gale pridedama programa *Shell.dll*, kurioje laikomas atšifravimo raktas, originalios programos kodo sekcijos adresas ir dydis. Be to, *Shell.dll* geba atšifruoti originalios programos kodo sekciją ir jai perduoti kontrolę. *Shell.dll* pridedama prie naudojamų DLL, o programos pradžios taškas nustatomas į *Shell.dll* pradžios tašką. Iliustracija pateikiama 1 pav.

- ***Stealmal*** [ZCY+24]. Visa originali programa užšifruojama ir pridedama prie programos *Shell.exe* galo. *Shell.exe* geba atšifruoti originalią programą ir perduoti jai kontrolę. Iliustracija pateikiama 1 pav.

- ***Hollowmal*** [ZCY+24]. Užšifruojama visa originali programa. Ji pridedama prie kurios nors nekenksmingos programos galo. Prie šio junginio galo pridedama *Hollow.dll* programa, kurios veikiamas panašus į *Shell.exe* iš *Stealmal*. Viso junginio pradžios taškas nustatomas į *Hollowmal.dll* pradžios tašką. Iliustracija pateikiama 1 pav.



1 pav. *Obfusmal (a), Stealmal (b) ir Hollowmal (c) perturbacijų veikimo principų iliustracijos. Adaptuota iš [ZHZ+22]*

1.3. GAN tipo modelių karkasai

GAN modelių karkasai paremti generatyviniais priešiškais tinklais (GAN), kurių veikimo principas yra du neuroniniai tinklai (generatorius ir diskriminatorius), žaidžiantys nulinės sumos žaidimą [CDH+16]. Kenkėjiško kodo obfuskacijos kontekste ir ypač „juodos dėžės“ atvejais, diskriminatorius atlieka surogatinio modelio vaidmenį. Bendras GAN modelių mokymosi etapas yra tokia seka [HT17; ZCY+24; ZZY+22]:

1. Generatorius, naudodamas požymių vektorių ir tokios pačios dimensijos „triukšmo“ (*angl. noise*) vektorių, sugeneruoja perturbacijas.
2. Originali kenkėjiška programa modifikuojama pagal perturbacijas (sukuriamas AE).
3. Diskriminatorius klasifikuoja sugeneruotą AE (kenkėjiškas / nekenkėjiškas). Pagal klasifikacijos rezultatą skaičiuojamos diskriminatoriaus ir generatoriaus nuostolių funkcijos reikšmės (diskriminatoriaus nuostolių funkcijos reikšmė priklauso nuo tikro detektoriaus klasifikacijos).
4. Visa seka kartojama nustatytą kiekį kartų.

1.3.1. *MalGAN*

Tai vienas iš pirmųjų ir populiariausių GAN tipo modelių karkasų. „Juodos dėžės“ detektoriai čia apibrėžiami kaip populiarūs ML klasifikatoriai, tokie, kaip MLP (*angl. Multilayer Perceptron*), RF (*angl. Random Forest*), DT (*angl. Decision Tree*), SVM (*angl. Support Vector Machine*). *MalGAN* karkaso [HT17] tikslas ir apibrėžimas pateikiami 1-oje lentelėje.

1 lentelė. *MalGAN karkasas*

Tikslas	Efektyviai išvengti AE aptikimo, kai ML kenkėjiškų programų detektoriaus įgyvendinimas nežinomas („juodos dėžės“ atvejis).
Surogatinis modelis	Daugiasluoksnis tiesioginio sklidimo neuroninis tinklas – klasifikatorius. Įvestis – programos požymių vektorius. Išvestis – klasifikacija į kenksmingą arba nekenksmingą. Šis tinklas taip pat naudojamas kaip diskriminatorius GAN architektūroje.
ML modelis	Daugiasluoksnis tiesioginio sklidimo neuroninis tinklas. Įvestis – programos požymių vektorius su prijungtu „triukšmo“ vektoriumi. Išvestis – modifikuotas požymių vektorius. Šis tinklas naudojamas kaip generatorius GAN architektūroje.
Požymiai	*MalGan* straipsnyje [HT17] naudojami tik API vardų požymiai, patenkantys į PE formato programų požymių kategoriją (žr. 1.1.1. skyrelį), tačiau autoriai nurodo, jog gali būti naudojami bet kokie požymiai[3].
Perturbacijos	Semantinės perturbacijos (žr. 1.2.2. skyrelį) – nereikalingų API vardų požymių pridėjimas.

[3]autoriai nagrinėja „juodos dėžės“ atvejį su prielaida, jog detektoriaus naudojami požymiai yra žinomi.

1.3.2. *N-gram MalGAN*

Šis karkasas remiasi *MalGAN* (1.3.1.) karkasu ir siekia jį pagerinti. *N-gram MalGAN* karkaso [ZZY+22] tikslas ir apibrėžimas pateikiami 2-oje lentelėje.

2 lentelė. *N-gram MalGAN karkasas*

Tikslas	Supaprastinti, pagreitinti ir pagerinti varžymosi principais pagrįstas atakas. Pašalinti prielaidas[3] apie detektorių „juodos dėžės“ atvejais.
Surogatinis modelis	Surogatinio modelio veikimas ir architektūra tokia pati, kaip ir *MalGAN* (1.3.1.).
ML modelis	Pagrindinio modelio veikimas ir architektūra labai panašūs į *MalGAN* (1.3.1.), tačiau norėdami stabilizuoti mokymosi procesą, autoriai siūlo nenaudoti „triukšmo“ vektoriaus. Vietoje to, generatoriaus išvestis (n-matis vektorius) modifikuojama nekeičiant pirmų m dimensijų, o kitas $n - m$ pakeičiant nekenksmingų programų požymiais.
Požymiai	Baitų lygio požymiai (žr. 1.1.2. skyrelį) – n-gramos.
Perturbacijos	Autoriai neatliko eksperimentų su perturbuotomis programomis, tačiau pažymi, jog norint gauti sugeneruotus požymių vektorius užtenka pridėti reikiamus baitus programos gale. Tai atitinka 1.2.1. apibrėžtą baitų lygio perturbaciją *ARBE*, tik šiuo atveju pridedami baitai nebūtų atsitiktiniai, o norimos n-gramos.

1.3.3. *MalFox*

MalFox taip pat remiasi *MalGAN* (1.3.1.), tačiau siekia kurti AE realiomis sąlygomis, dėl to atlieka esminius pakeitimus. *MalFox* karkaso [ZCY+24] tikslas ir apibrėžimas pateikiami 3-oje lentelėje.

3 lentelė. *MalFox karkasas*

Tikslas	Generuoti AE, kurių neaptiktų komerciniai detektoriai (prieš tai aptarti karkasai eksperimentams kaip nepriklausomą detektorių naudojo tokius ML modelius, kaip SVM, KNN, GBDT ir kt., bet ne komercinius detektorius). Šio karkaso detektorius yra *VirusTotal* (viešai prieinama paslauga, agreguojanti virš 70 komercinių kenkėjiškų programų detektorių).
Surogatinis modelis	Surogatinis modelis, kaip ir kituose GAN tipo modelių karkasuose, naudojamas kaip diskriminatorius. Įvestis – perturbuota programa. Išvestis – klasifikacija į kenksmingą arba nekenksmingą. Įgyvendinimas – konvoliucinis neuroninis tinklas (CNN).
ML modelis	Standartinis GAN generatorius, požymių vektorių sujungiantis su „triukšmo“ vektoriumi. Įgyvendinimas – konvoliucinis neuroninis tinklas (CNN).

Požymiai	PE formato programų požymiai (žr. 1.1.1. skyrelį) – DLL vardai.
Perturbacijos	Visos kompleksinės perturbacijos (žr. 1.2.3. skyrelį).

1.4. Skatinamojo mokymosi tipo modelių karkasai

Skatinamojo mokymosi (RL) modeliai susideda iš agento ir aplinkos. Aplinka susideda iš informatyvių požymių ištraukimo metodo (*angl. feature extraction*) ir kenkėjiškų programų detektoriaus. Šiuo atveju aplinkos būsenų erdvė S yra požymių vektorių erdvė. Agentas – tai algoritmas ar neuroninis tinklas, kurio tikslas yra surasti optimalią strategiją. Šiuo atveju strategijos veiksmų erdvė A susideda iš perturbacijų (žr. 1.2. poskyrį) [FWL+19]. Bendras RL modelių mokymosi etapas yra tokia seka [FWL+19; ZHZ+22]:

1. Agentas, naudodamas dabartinę aplinkos būseną ir praeito veiksmo atlygį (*angl. reward*), parenka sekantį veiksmą iš galimų veiksmų aibės ir taiko mokymosi algoritmą (algoritmas priklauso nuo agento įgyvendinimo).

2. Atliekamas veiksmas – perturbuojama programa arba požymių vektorius (priklauso nuo karkaso).

3. Gaunami aplinkos kitimo įverčiai – nauja būsena ir atlygis, skaičiuojamas pagal detektoriaus klasifikacijos rezultatą.

4. Seka kartojama tol, kol agentas nelaiko strategijos optimalia arba nustatytą kiekį kartų.

1.4.1. *DQEAF*

Šis karkasas taiko gilųjį skatinamąjį mokymąsi, kai agentas įgyvendinamas kaip gilusis neuroninis tinklas. *DQEAF* karkaso [FWL+19] tikslas ir apibrėžimas pateikiami 4-oje lentelėje.

4 lentelė. *DQEAF karkasas*

Tikslas	Parodyti, jog ML kenkėjiškų programų detektoriai, ypač modeliai, išmokyti prižiūrimu mokymusi, yra pažeidžiami varžymosi principais pagrįstoms atakoms.
Surogatinis modelis	RL karkasuose nenaudojami surogatiniai modeliai. Kaip „juodos dėžės“ detektorius pasirinktas GBDT modelis.
ML modelis	Agentas įgyvendintas kaip gilusis Q-tinklas (CNN praplėtimas, kai tinklas naudojamas kaip Q-funkcijos aproksimacija). Taip pat taikomas prioritetizuotas patirčių pakartojimo metodas (*angl. prioritized experience replay*), kuomet agentas mokomas tik su aukštą atlygį gavusiais perėjimais ($S \times A$).

Požymiai	Požymių vektorius taip pat apibrėžia visų būsenų erdvę S. Šiuo atveju $S = \mathbb{R}^{513}$. Baitų lygio požymiai (žr. 1.1.2. skyrelį) – baitų/entropijos histograma.
Perturbacijos	Perturbacijos apibrėžia visų galimų agento veiksmų erdvę A. Šiuo atveju $A = \{0,1\}^4$. Baitų lygio perturbacijos (žr. 1.2.1. skyrelį): • *ARBE* • *ARI* • *ARS* • *RS*

1.4.2. *MalInfo*

MalInfo remiasi *MalFox* (1.3.3.). *MalInfo* karkaso [ZHZ+22] tikslas ir apibrėžimas pateikiami 5-oje lentelėje.

5 lentelė. *MalInfo karkasas*

Tikslas	Surasti optimalią obfuskacijos strategiją konkrečiai programai, pagal kurią sukurtas AE nebūtų aptiktas komercinių kenkėjiškų programų detektorių.
Surogatinis modelis	RL surogatinis modelis nenaudojamas. „Juodos dėžės“ detektoriumi pasirinkti komerciniai detektoriai (*VirusTotal*).
ML modelis	Agentas įgyvendintas kaip klasikiniai ML algoritmai (konkrečiai dinaminis programavimas ir skirtumų laike (*angl. temporal difference*) algoritmas).
Požymiai	Agentas nėra neuroninis tinklas ir požymių iš programos netraukia. Agentas mokosi tik iš perėjimų, o būsenų erdvė S yra originali programa ir perturbuoti jos variantai. Teoriškai perturbuotų programos variantų galėtų būti be galo daug, tuomet $S = A^\infty, \lvert S\rvert = \aleph_0$, tačiau autoriai nurodo, jog daugiau nei 3 sluoksniai kompleksinių perturbacijų reikšmingai paveikia programos veikimo laiką, o tai gali „sukelti įtarimų“ komerciniams detektoriams. Todėl pasirinkta $S = A^3$.
Perturbacijos	$A = \{0,1,2,3\}$ • *Null* perturbacija – naudinga tik formaliam pilnumui (atitinka nulinį A veiksmą). • Visos kompleksinės perturbacijos (žr. 1.2.3. skyrelį), t. y. tokios pačios, kaip ir *MalFox* (1.3.3.) karkaso.

1.5. Genetinių algoritmų tipo modelių karkasai

Genetinai algoritmai (GA) yra viena seniausių mašininio mokymosi ML apraiškų; jų veikimas paremtas evoliucija [CSD19]. Kenkėjiškų programų obfuskacijai AE generavimas taikant GA yra tokia seka [YPT22]:

1. Sukuriama pradinė populiacija (perturbacijos metodai pradinei populiacijai priklauso nuo karkaso).
2. Atliekamas tinkamumo (*angl. fitness*) vertinimas.
3. Atliekama selekcija – dažniausiai pasirenkami geriausiai įvertinti populiacijos AE, tačiau galimos ir kitos selekcijos strategijos.
4. Atliekamas selekcijos atrinktų AE kryžminimas (po 2) taip sukuriant naują AE, turintį po dalį genų iš abiejų kryžmintų AE.
5. Tam tikrai daliai AE atliekama dalies genų mutacija.
6. Vertinama, ar sugeneruoti AE atitinka kriterijus (vertina detektorius).
7. Jei kriterijai nėra tenkinami, seka kartojama nuo 2-o žingsnio.

1.5.1. *AIMED*

AIMED karkaso [CSD19] tikslas ir apibrėžimas pateikiami 6-oje lentelėje.

6 lentelė. *AIMED karkasas*

Tikslas	AE generavimo greičio padidinimas ir modelių kompleksiškumo sumažinimas, lyginant su GAN ir RL tipo modelių karkasais.
Surogatinis modelis	Surogatinis modelis nenaudojamas. Naudojami „juodos dėžės“ detektoriai yra **3 komerciniai** (*Kaspersky, ESET, Sophos*) ir vienas ML modelis – GBDT.
ML modelis	Klasikinis GA modelis – veikimas visiškai atitinką bendrą seką. Tinkamumo (*angl. fitness*) vertinamas remiasi AE požymių vektoriaus panašumu į originalios programos požymių vektorių (kuo mažiau panašūs, tuo tinkamumo įvertinimas didesnis).
Požymiai	Baitų lygio požymiai (žr. 1.1.2. skyrelį) – atskiras n-gramų atvejis, kai $n = 1$.
Perturbacijos	Baitų lygio perturbacijos[4] (žr. 1.2.1. skyrelį).

[4] autoriai rėmėsi perturbacijomis, aprašytomis [AKF+18]

1.5.2. *GAMMA*

GAMMA karkaso [DBL+21] tikslas ir apibrėžimas pateikiami 7-oje lentelėje.

7 lentelė. *GAMMA karkasas*

Tikslas	Efektyvus (neaptikimo šansų didinimas naudojant perturbacijas, paremtas nekenksmingomis programomis) varžymosi principais pagrįstų atakų kūrimas.
Surogatinis modelis	Surogatinis modelis nenaudojamas. GBDT ir *MalConv* pasirinkti kaip „juodos dėžės“ detektoriai.
ML modelis	Pagrindinė modelio idėja yra požymių ištraukimas iš nekenksmingų programų ir jų pridėjimas, naudojant tam pritaikytas perturbacijas, į kenksmingas programas kiekvienos populiacijos generavimo metu. Tinkamumo (*angl. fitness*) ir kriterijų vertinimas atliekamas naudojant detektorių ir pridėtų požymių dydį baitais (norima pridėti kuo mažiau požymių).
Požymiai	• PE formato programų požymiai (žr. 1.1.1. skyrelį). • Kodas sekcijose (nestandartinis požymis).
Perturbacijos	• Visos baitų lygio perturbacijos (žr. 1.2.1. skyrelį), gebančios pridėti baitus. • Autoriai pažymi, jog gali būti naudojama ir DLL / API vardų pridėjimo semantinė perturbacija (žr. 1.2.2. skyrelį).

1.6. Nevalidaus PE formato problema

Andersonas ir kt. [AKF+18], atlikdami eksperimentus su funkcionalumą išlaikančiomis perturbacijomis PE formato failams, pastebėjo, jog ne visais atvejais perturbuotos programos veikia teisingai. Dėl *Windows* operacinės sistemos PE formato failų interpretavimo ir paleidimo specifikos, programas įmanoma parašyti tokiu būdu, jog pakeitus kodo ar kitų sekcijų turinį nekeičiant originalių mašininio kodo instrukcijų, programa neveiktų. Techniškai, programų rašymas tokiu būdu pažeidžia patį PE formato standartą, tačiau šią praktiką neretai naudoja kenkėjiškų programų autoriai.

Norint visiškai išvengti nevalidaus PE formato problemos tenka taikyti perturbacijas, nekeičiančias originalių programų, o taikančias kitokius obfuskacijos metodus. Iš 1.2. poskyryje aptartų perturbacijų, tokias sąlygas atitinka tik 2 kompleksinės perturbacijos (žr. 1.2.3. skyrelį) – *Stealmal* ir *Hollowmal*.

1.7. AE perkeliamumas

Perkeliamumas DI modeliuose dažniausiai suprantamas kaip žinių perkeliamumas (*angl. knowledge transferability*). Tačiau tiriant varžymosi principais pagrįstas atakas buvo pastebėtas ir AE perkeliamumas (*angl. adversarial sample transferability*). Tai reiškia, jog gebant sukurti AE, kuriuos neteisingai klasifikuoja vienas modelis, tikėtina, jog kitas (tos pačios paskirties) modelis taip pat klasifikuos AE neteisingai. Be to, nustatyta, jog AE perkeliamumo savybė galioja skirtingų architektūrų modeliams [DCB+21]. Šia savybe remiasi visos „juodos dėžės" atvejams pritaikytos atakos.

1.8. AE aptikimo strategijos

1.8.1. Varžymosi principais grįstas mokymasis

Varžymosi principais grįstas mokymasis (*angl. adversarial retraining*) – tai papildomas ML modelio mokymo etapas. Tarkime pradinis išmokytas ML modelis yra M_1. Tuomet varžymosi principais grįstas mokymasis yra AE generavimas atakuojant M_1 ir šio modelio papildomas mokymas su sugeneruotais AE. Po mokymo gauname M_2 ($M_1 \xrightarrow{AE} M_2$), kuris bus atsparesnis varžymosi principais pagrįstoms atakoms. Ši strategija nėra atspari „juodos dėžės" atakoms [CAD+21].

1.8.2. Gradientų slėpimas

Gradientų slėpimas (*angl. gradient hiding*) – tai metodas, kai pasirenkama klasifikatoriaus architektūra, kurioje nefigūruoja gradientai, pavyzdžiui, nediferencijuojami modeliai, tokie kaip atsitiktinis miškas (*angl. random forest*). Dėl AE perkeliamumo (žr. 1.7. poskyrį) ši strategija yra neveiksminga prieš „juodos dėžės" atakas [CAD+21].

1.8.3. Kategorijų švelninimas

Kategorijų švelninimas (*angl. label smoothing*) – metodas, leidžiantis geriau klasifikuoti nežinomus duomenis ir taip apsisaugoti nuo varžymosi principais pagrįstų atakų. Kategorijų švelninimas reiškia tiksliai apibrėžtų kategorijų (*angl. hard labels*) pavertimą tikimybiniais vektoriais (*angl. soft labels*). Tuomet ML modelio išvestis (taip pat tikimybinis vektorius) gali būti palyginama su kategorijų vektoriais taikant norimas euristikas, pavyzdžiui, kosinusų panašumą (*angl. cosine similarity*). Dėl AE perkeliamumo (žr. 1.7. poskyrį) ši strategija neveiksminga prieš „juodos dėžės" atakas [CAD+21].

1.8.4. Perkeliamumo blokavimas

Perkeliamumo blokavimas (*angl. transferability blocking*) – tai varžymosi principais grįsto mokymosi (žr. 1.8.1. skyrelį) praplėtimas, įtraukiant naują (*NULL*) kategoriją į galimas klasifikatoriaus išvestis. Ši strategija praplečia klasifikatoriaus mokymą į šiuos etapus:

1. Įprastas klasifikatoriaus mokymas.

2. *NULL* tikimybių funkcijos f skaičiavimas. p_{NULL} suprantama kaip varžymosi principais pagrįstos atakos tikimybė. *NULL* tikimybių ir jų skaičiavimo funkcijos f ryšys pateiktas 1-oje lygtyje.

3. Varžymosi principais grįstas mokymasis, įtraukiant tiek originalias įvestis, tiek perturbuotas (δX).

Ši strategija sprendžia pagrindinį daugelio kitų strategijų trūkumą – AE perkeliamumą (žr. 1.7. poskyrį), todėl yra pakankamai efektyvi [CAD+21].

$$
\begin{gathered}
p_{NULL} = f\left(\frac{\|\delta X\|_0}{\|X\|}\right), \\
\text{čia } \|\delta X\|_0 \sim U[1,\, N_{max}],\ N_{max} = \min n \ni f\left(\frac{n}{\|X\|}\right) = 1
\end{gathered}
\tag{1}
$$

1.8.5. Bazės keitimas, transformacijos

Bazės keitimas, transformacijos (*angl. change of basis, transformations*) – tai strategija transformuojanti duomenis juos projektuojant kitoje koordinačių sistemoje. Pavyzdžiui, PCA metodas keičia duomenų koordinačių sistemos bazę taip, jog pirma komponentė turėtų didžiausią inerciją. Tam tikrais atvejais taikant šią strategiją gali būti prarandama informacija (jei keičiama į mažesnės dimensijos bazę), pavyzdžiui, naudojant *JPEG lossy compression* algoritmą nuotraukų transformacijai. Ši strategija padeda efektyviai apsisaugoti nuo visų tipų varžymosi principais pagrįstų atakų [CAD+21].

1.9. LIME metodas

LIME (*angl. Local Interpretable Model-agnostic Explanations*) – tai lokalūs, interpretuojami ML modelių išvesčių paaiškinimai. ML modeliai dažnai turi būti sudėtingi (dėl to ir sunkiai, ar visiškai neinterpretuojami), nes jie aproksimuoja sudėtingus skirstinius visoje (globalioje) srityje. LIME metodo idėja yra aproksimuoti skirstinį taip pat, kaip ML modelis, bet žymiai mažesnėje – **lokalioje** – srityje (*angl. locally faithful*) aplink mus dominantį tašką (ML modelio įvestį). Tai leidžia LIME naudojamam aproksimacijos modeliui būti ženkliai paprastesniam, daugeliu atvejų – tiesiniam (pvz., tiesinė regresija). LIME veikimo principas yra vieno pavyzdžio (įvesties x_0) perturbavimas taip sukuriant lokalių x_0 artimų įvesčių aibę $\hat{X}$. Kiekvienas $\hat{X}$ elementas pateikiamas originalaus modelio (M) klasifikacijai ir taip gaunama aibė $\hat{Y}$ ($\hat{X} \xrightarrow{M} \hat{Y}$). Abi aibės naudojamos surogatinio (dažniausiai tiesinio) modelio $\hat{M}$ mokymui. Tuomet **interpretuojamas paaiškinimas** yra surikiuotas pagal įtaką galutiniam $\hat{M}$ sprendimui x_0 komponenčių (požymių) sąrašas (pilnas arba dalinis) [RSG16].

Svarbus LIME parametras yra branduolio plotis (*angl. kernel width*) ω. LIME mokymosi etape kiekvienai perturbuotai įvesčiai $\hat{x}_i \in \hat{X}$ priskiria svorį $a_i \propto \exp\left(-\frac{D(x_0,\hat{x}_i)}{\omega^2}\right)$. Taigi, ω turi būti parenkamas atsižvelgiant į skalę, kurioje klasifikuojami klasteriai gali būti atskiriami tiesiškai.

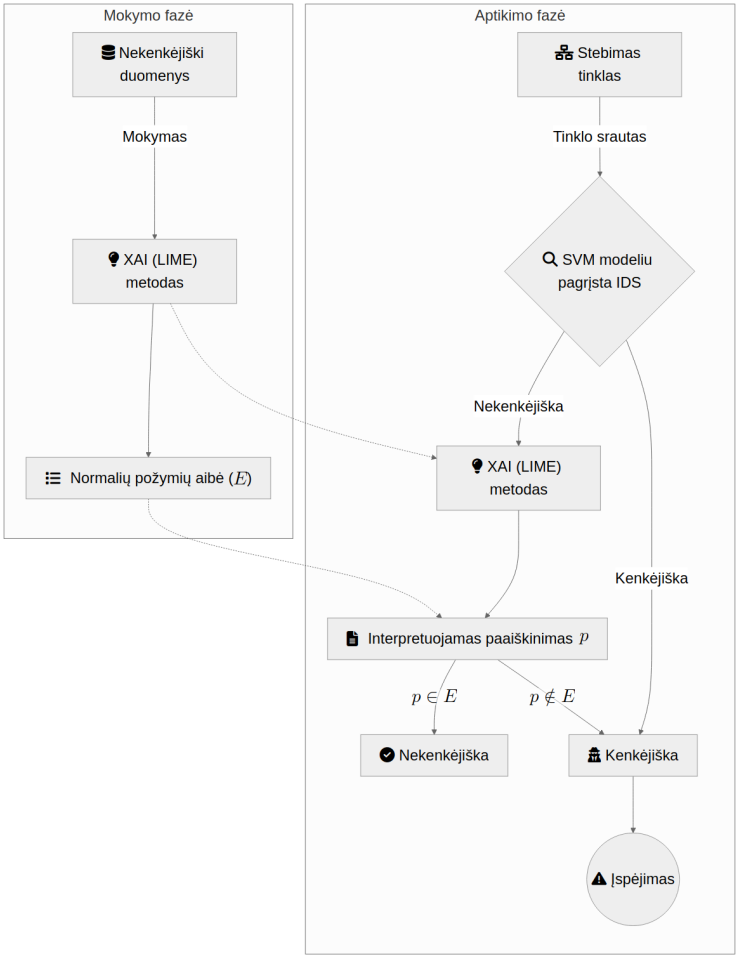
LIME metodas yra paaiškinamo dirbtinio intelekto (XAI) pavyzdys. XAI gali būti naudojamas

kaip AE aptikimo strategija, jei paaiškinimai geba pastebimai atskirti AE nuo tikrų įvesčių (pavyzdžiui, klasifikacijos paaiškinimas statistiškai reikšmingai skiriasi nuo tikrų duomenų paaiškinimų).

1.9.1. Varžymosi principais pagrįstų atakų aptikimas IDS naudojant LIME

IDS (*angl. Intrusion Detection System*) – tai sistema, veikianti uždarame tinkle ir nuolat analizuojanti tinklo srautą. Vienas iš IDS įgyvendinimo būdų yra pasitelkti ML modelius, tad šios sistemos taip pat yra pažeidžiamos varžymosi principais pagrįstoms atakoms. Šiame kontekste Tcydenova ir kt. pritaikė LIME kaip AE aptikimo strategiją. Jų siūlomas metodas (žr. 2 pav.) AE aptikimui susideda iš dviejų pagrindinių dalių [TKL+21]:

1. Mokymo fazė.
   - Mokomas ML modelis tinklo srauto klasifikavimui.
   - Kiekvienam nekenkėjiškų mokymo duomenų pavyzdžiui pritaikomas LIME metodas – gaunama paaiškinimų aibė E, kur kiekvienas elementas yra n-matis vektorius, turintis n svarbiausių **interpretuojamų paaiškinimų**.
   - E laikoma normalių požymių aibe.
2. Aptikimo fazė.
   - Išmokytas ML modelis klasifikuoja tinklo srautą.
   - Jei ML modelis nustato, jog įvestis yra kenksminga – ji tokia ir laikoma.
   - Jei ML modelis nustato, jog įvestis nėra kenksminga – taikomas LIME metodas ir gaunamas klasifikavimo paaiškinimas.
   - Jei klasifikavimo paaiškinime figūruoja bent vienas požymis, nepriklausantis E aibei – laikoma, jog įvestis yra kenksminga.
   - Jei visi klasifikavimo paaiškinime figūruojantys požymiai priklauso aibei E – laikoma, jog įvestis nekenksminga.



2 pav. *LIME pritaikymas AE aptikimui IDS (adaptuota iš [TKL+21])*

LIME metodo taikymas varžymosi principais pagrįstų atakų aptikimui yra artimas **perkeliamumo blokavimo** (žr. 1.8.4. skyrelį) strategijai, tačiau šiuo atveju LIME veikia kaip atskiras nuo pagrindinio klasifikatoriaus komponentas, tad pagrindinis ML modelis lieka pažeidžiamas varžymosi principais pagrįstoms atakoms.

1.10. Dimensijų mažinimo metodai

1.10.1. Principinių komponenčių analizė

Principinių komponenčių analizė (PCA) – tai daugiamatės statistinės analizės metodas, naudojamas išskirti svarbiausią informaciją iš turimų duomenų. Svarbiausios informacijos apibrėžimui naudojama inercijos metrika. Inercija – tai dispersijos dalis, kuri yra paaiškinama viena komponente. Iš esmės PCA yra bazės keitimo metodas (žr. 1.8.5. skyrelį), kurio tikslas yra parinkti bazę, tenkinančią nelygybę $\forall n : I_n \geq I_{n+1}$, kur I_n – n-osios komponentės inercija [AW10]. Tai leidžia sumažinti naudojamų komponenčių skaičių pasirenkant pakankamą sukauptos inercijos vertę. Pavyzdžiui, jei pirmosios komponentės inercija yra 50 %, antrosios – 25 % ($\rightarrow$ likusių komponenčių inercija lygi 25 %) ir pakankama inercijos vertė yra 70 %, tuomet užtenka palikti pirmas dvi komponentes, kurių sukaupta inercija (*angl. cummulative inertia*) yra 75 %.

1.10.2. Daugialypės korespondencijos analizė

Daugialypės korespondencijos analizė (MCA) – tai dar vienas daugiamatės statistinės analizės metodas, tik šis skirtas kategoriniams duomenims. Šio metodo veikimo principas pagrįstas standartine korespondencijos analize (*angl. correspondence analysis*) iš kategorinių duomenų sukonstruotai indikatorių matricai. MCA gali būti laikomas PCA praplėtimu kategoriniams duomenims [AV07], tad ir dimensijų mažinimo procesas toks pat – pasirenkama pakankama inercijos vertė ε ir pasirenkami pirmi m stulpelių taip, kad $\sum_{i=1}^{m} I_i \geq \varepsilon$

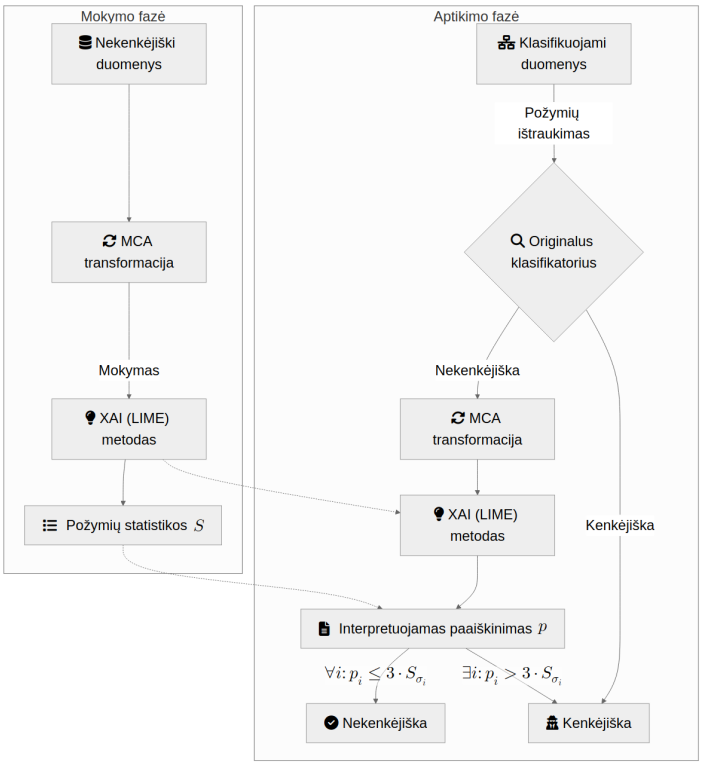
MCA metodas gali būti naudojamas kaip išmokomas statistinis modelis. Kadangi nauja bazė, į kurią transformuojami duomenys, yra glaudžiai susijusi su duomenų rinkinio statistikomis, galima naudoti mokymo duomenų aibę šios bazės nustatymui, o tolimesnius duomenis transformuoti kaip projekcijas į šią bazę. Tolesniuose šio darbo skyriuose MCA naudojamas būtent taip.

2. MCA ir LIME AE aptikimo metodų sintezė

Šio darbo autoriaus siūlomas AE aptikimo metodas yra apjungti MCA dimensijų mažinimo metodą bei LIME pritaikymą AE aptikimui su tam tikromis modifikacijomis (žr. 2.1. poskyrį). Kadangi MCA ir LIME atitinka bazės keitimo (žr. 1.8.5. skyrelį) ir perkeliamumo blokavimo (žr. 1.8.4. skyrelį) AE aptikimo strategijas, kurios iš aptartų yra perspektyviausios – jų sinteze tikimasi gauti dar tikslesnį AE aptikimo metodą (žr. 3 pav.).

2.1. LIME pritaikymo AE aptikimui metodo modifikacijos

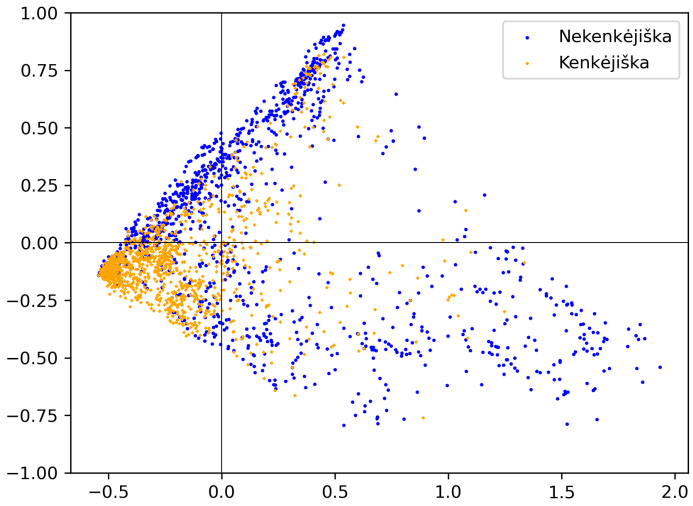
1.9.1. skyriuje siūlomas LIME pritaikymas AE aptikimui remiasi svarbiausių (didžiausią įtaką ML modelio sprendimų priėmimui turinčių) požymių analize. Svarbiausi požymiai laikomi pirmieji 10 [TKL+21]. Nors autoriai neaprašo kaip pasirenkama tokia konstanta, akivaizdu, jog ji nėra tinkama visiems atvejams. Pavyzdžiui, turint žymiai daugiau požymių, ši konstanta gali būti per maža. Tokia problema ypač aktuali kai analizuojami kategoriniai požymiai, turintys daug kategorijų (dažniausiai koduojami kaip dvejetainiai vektoriai). Dėl šios priežasties, autoriaus siūlymas yra „svarbiais“ laikyti visus požymius ir mokymo fazės pabaigoje apskaičiuoti šių požymių įtakos ML modelio sprendimo priėmimui statistikas (vidurkį ir standartinį nuokrypį). Tuomet aptikimo fazėje naudoti 3σ taisyklę [Puk94], t. y., jei bent vieno požymio įtaka ML sprendimo priėmimui nukrypsta per 3 standartinius nuokrypius nuo duotojo požymio vidurkio – laikyti, jog analizuojama įvestis buvo obfuskuota.



3 pav. *LIME ir MCA sintezės AE aptikimo metodo iliustracija*

2.2. LIME branduolio pločio pasirinkimas

LIME branduolio plotį (žr. 1.9. poskyrį) gali padėti nustatyti duomenų vizualizacijos ar kita duomenų analizė. Tai atlikti nėra sudėtinga, kai požymių vektoriaus dimensija nedidelė arba požymiai yra skaitiniai, tačiau turint kategorinius požymius ir atsižvelgiant į jų kodavimą dvejetainiais vektoriais, parinkti tinkamą LIME branduolio plotį gali būti sudėtinga. Taikant MCA transformaciją neretai užtenka branduolio pločio vertę parinkti pagal pirmų dviejų komponenčių skalę, kadangi šios dvi komponentės apibūdina didžiausią inerciją. Pavyzdžiui, 4 pav. vaizduojamu atveju, tinkamos ω vertės turėtų būti vienetų skalėje (tiksli vertė nustatoma eksperimentiškai).



4 pav. *Pirmųjų dviejų MCA transformacijos komponenčių pavyzdys*

2.3. Panaudos atvejai

1. **Varžymosi principais pagrįstoms atakoms atsparus klasifikatorius.** Šiuo atveju naudojamas praplėstas klasifikavimo procesas, visiškai atitinkantis 3-ią pav. Klasifikatoriaus klasės išlieka tokios pačios.

2. **Varžymosi principais pagrįstų atakų indikatorius.** Taip pat naudojamas praplėstas klasifikavimo procesas, tik šiuo atveju, jei originalaus klasifikatoriaus sprendimas buvo klasifikuoti įvestį kaip *nekenkėjišką*, o tolimesnės analizės rezultatas buvo klasifikuoti įvestį kaip *kenkėjišką* – tuomet įvesčiai priskiriama *obfuskuota* klasė.

2.4. Paaiškinamumo aspektas

Nors LIME metodas skirtas paaiškinti sudėtingų modelių sprendimus taip, kad žmogus gebėtų juos interpretuoti ir suprasti, autoriaus siūlomas metodas šią savybę praranda. Pagrindinė to priežastis yra tai, jog šiame metode LIME paaiškina MCA transformacijos duomenis, kurie savaime nėra interpretuojami. Tuo atveju, kai siūlomas metodas naudojamas kaip **atakos indikatorius**, tai reiškia, jog šis metodas negali pateikti interpretuojamo paaiškinimo, kodėl tam tikra įvestis yra laikoma obfuskuota.

3. Lyginamoji metodų analizė

Lyginamosios analizės tikslas – ištirti kenkėjiškų programų detektorių tikslumą varžymosi principais pagrįstos atakos sąlygomis. Lyginami originalus (nepakeistas) detektorius, autoriaus siūlomas LIME ir MCA sintezės metodas (žr. 2. skyrių) bei atskiros jo sudedamosios dalys: LIME metodo pritaikymas AE aptikimui (žr. 1.9.1. skyrelį) ir MCA transformacijos klasifikatorius (žr. 1.10.2. skyrelį).

3.1. Tyrimo metodika

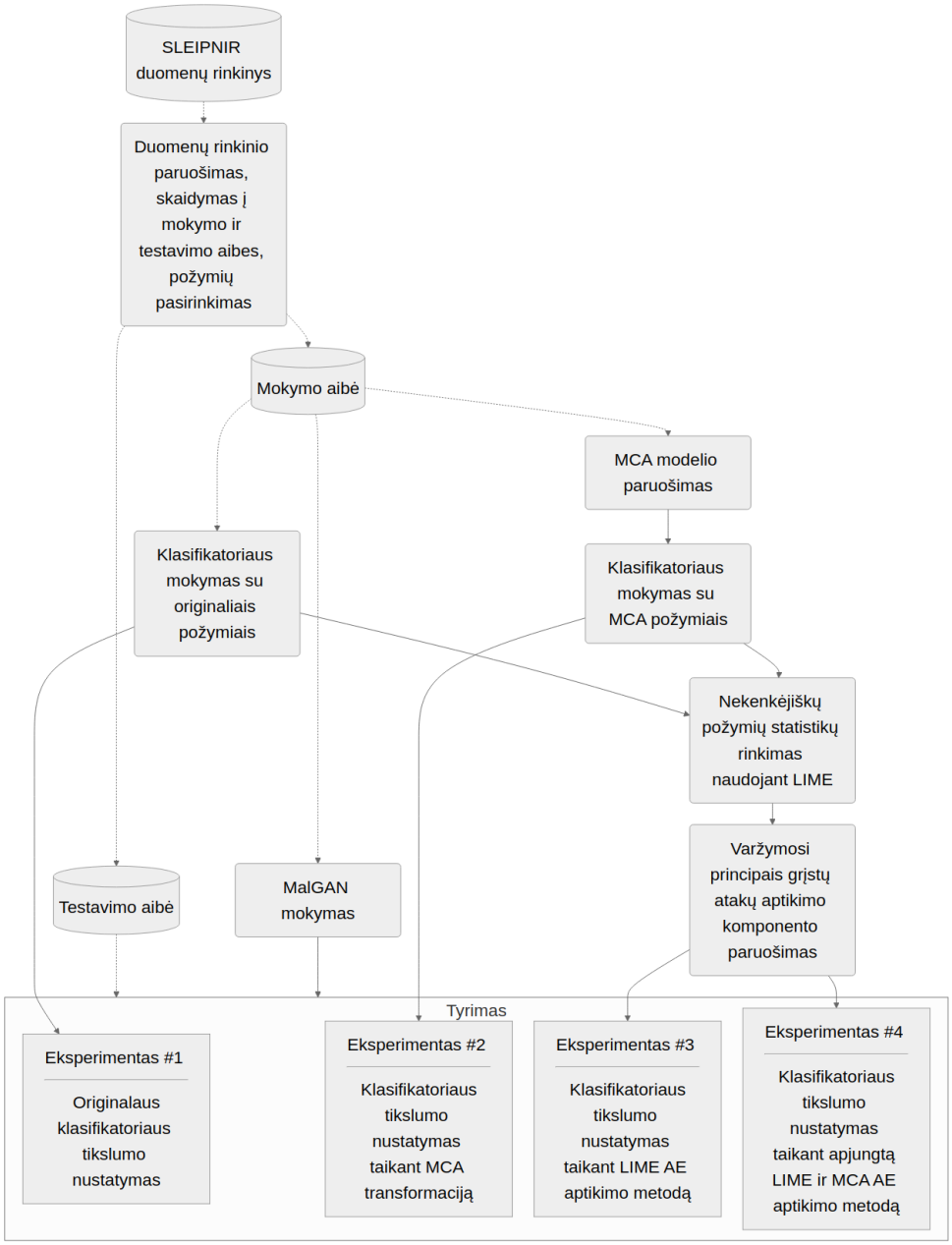
Visa tyrimo metodika pavaizduota 5 pav. Tyrimui pasirinktas *SLEIPNIR* [AHH+18] duomenų rinkinys, kuriame yra 34995 kenkėjiškų programų ir 19696 nekenkėjiškų programų pavyzdžių, užkoduotų 22761-mačiais dvejetainiais vektoriais. Iš šio rinkinio pasirinkta po 1500 unikalių kenkėjiškų ir nekenkėjiškų programų pavyzdžių paliekant pirmas 200 komponenčių – taip gaunamas subalansuotas duomenų rinkinys. Šis rinkinys toliau skeliamas į mokymo ir testavimo aibes santykiu 4 : 1. Eksperimentams atlikti paruošiamos šios priemonės:

1. Testavimo duomenų aibė, iš kurios sudaromas trijų tipų programų požymių srautas: nekenkėjiškos, kenkėjiškos, kenkėjiškos-obfuskuotos.
2. *MalGAN* [HT17] AE generatorius. Pasirinkimo motyvacija pateikiama 3.2. poskyryje.
3. Kenkėjiškų programų detektorius (klasifikatorius). Naudojamas GBDT modelis.
4. Varžymosi principais pagrįstų atakų aptikimo komponentas
    - Naudojantis originalius požymius.
    - Naudojantis MCA transformuotus požymius (žr. 2. skyrių).

Su šiais įrankiais atliekami 4 eksperimentai **aplinkoje su varžymosi principais pagrįstomis atakomis**:

1. Bazinis atvejis – originalaus klasifikatoriaus tikslumo nustatymas.
2. Klasifikatoriaus, naudojančio MCA transformuotus požymius, tikslumo nustatymas.
3. Klasifikatoriaus praplėtimo su modifikuoto LIME metodo pritaikymu varžymosi principais pagrįstų atakų aptikimui tikslumo nustatymas.
4. Klasifikatoriaus praplėtimo su LIME ir MCA metodų sinteze varžymosi principais pagrįstų atakų aptikimui tikslumo nustatymas.

Visų tyrime atliekamų klasifikacijų kokybė matuojama taikant preciziškumo (*angl. precision*), atkūrimo (*angl. recall*) ir F1 metrikas bei vertinamas bendras tikslumas (*angl. accuracy*).



5 pav. *Tyrimo metodika*

3.2. Varžymosi principais pagrįstų atakų karkaso pasirinkimas

Tyrimui pasirinktas *MalGAN* (1.3.1.) karkasas dėl šių priežasčių:

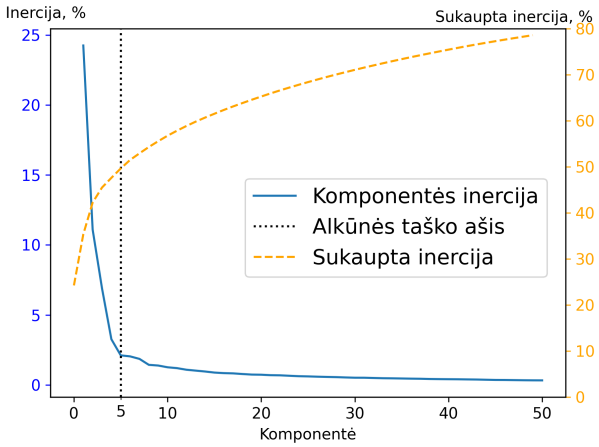
- GAN yra vieni iš perspektyviausių ir efektyviausių ML modelių AE generavimui.
- *MalGAN* yra plačiai naudojamas kaip bazinis modelis tolimesniems tyrimas, yra kelios atviro kodo repozitorijos[5], įgyvendinančios šį karkasą.
- *MalGAN* naudoja dvejetainius požymių vektorius. Būtent tokie požymiai turėtų kelti sunkumų esamiems AE aptikimo metodams (žr. 2.1. poskyrį).

[5] Šiam tyrimui pasirinkta https://github.com/ZaydH/MalwareGAN repozitorija, kaip pradinis *MalGAN* įgyvendinimas

3.3. MCA komponenčių pasirinkimas

Komponenčių pasirinkimas yra 5 pav. minimo MCA modelio paruošimo dalis. Komponenčių pasirinkimui, t. y. jų kiekio pasirinkimui, nėra apibrėžto „teisingo“ metodo [AW10]. Dažniausiai siūlomi metodai yra tik didesnių už 1 tikrinių reikšmių pasirinkimas ir *alkūnės* (*angl. scree / elbow*) metodas. Šie metodai netiko, nes juos taikant paliktos MCA komponentės nepaaiškintų net pusės visos inercijos: visos tikrinės reikšmės analizuojamuose mokymo duomenyse yra < 1, o alkūnės taške sukaupta inercija yra 47,62 % (žr. 6 pav.). Taigi, pasirinkti 2 nestandartiniai MCA komponenčių kiekio kriterijai:

- sukaupta inercija ≥ 75 %,
- išmokyto klasifikatoriaus tikslumas $\hat{a}$ nemažesnis nei originalaus klasifikatoriaus tikslumas a 5 % intervale ($\hat{a} \geq a - 5$ %), kai įvesčių srautas normalus (nėra obfuskuotų programų požymių – AE). Šiuo atveju $\hat{a} \geq 0{,}838$ (žr. 8 lentelę)



6 pav. *Alkūnės analizė MCA inercijai*

8 lentelė. *Originalaus[6] klasifikatoriaus metrikos, kai nevykdoma varžymosi principais pagrįsta ataka*

Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,890	0,887	0,888
Kenkėjiška	0,887	0,890	0,889
Vidurkis	0,888	0,888	0,888
		Tikslumas:	0,888

9 lentelė. *MCA klasifikatoriaus metrikos, kai nevykdoma varžymosi principais pagrįsta ataka*

Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,823	0,870	0,846
Kenkėjiška	0,862	0,813	0,837
Vidurkis	0,843	0,842	0,842
		Tikslumas:	0,842

9-oje lentelėje pavaizduotos MCA transformacijos klasifikatoriaus metrikos gaunamos pasirinkus 50 komponenčių (suspaudimo santykis $4:1$), kurių sukaupta inercija yra 78,57 %. Kadangi $78{,}57\ \% > 75\ \%$ ir $0{,}842 > 0{,}838$, abu MCA komponenčių kiekio kriterijai yra tenkinami.

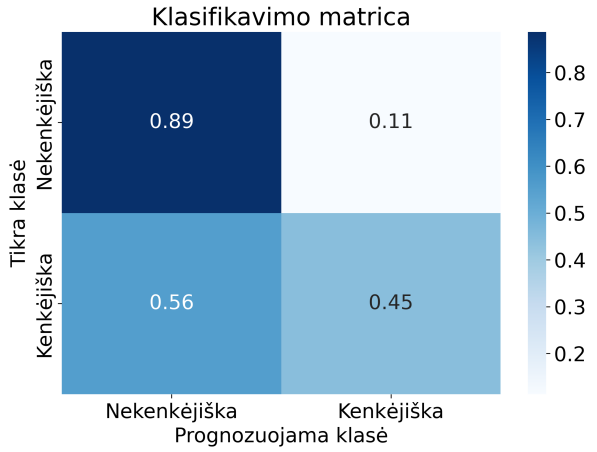
3.4. Eksperimentai

3.4.1. Originalaus klasifikatoriaus tikslumo nustatymas

Kaip ir tolimesniuose eksperimentuose, testavimo duomenų aibė šiam eksperimentui susideda iš 300 kenkėjiškų ir 300 nekenkėjiškų programų požymių. Taip pat pridedami dar 300 obfuskuotų programų požymių (jie gaunami naudojant jau turimus kenkėjiškų programų požymius ir išmokytą *MalGan* modelį).

Kadangi originalus[6] klasifikatorius negali diferencijuoti obfuskuotų programų, turima duomenų aibė nėra subalansuota – turime dvigubai daugiau duomenų, kuriuos klasifikatorius turėtų klasifikuoti kaip kenkėjiškus. Dėl to, klasifikavimo lentelėje (žr. 7 pav.) rodomas prognozuojamos klasės ir visų tai klasei priklausančių duomenų santykis (tokios matricos pagrindinė diagonalė nurodo klasės atkūrimo statistiką (*angl. recall*)).

Eksperimento rezultatai (klasifikavimo metrikos) pateikiami 10-oje lentelėje, bendros metrikos dėl nesubalansuotų klasių skaičiuojamos kaip svertinis vidurkis.



7 pav. *Klasifikavimo matrica*

10 lentelė. *Originalaus klasifikatoriaus metrikos*

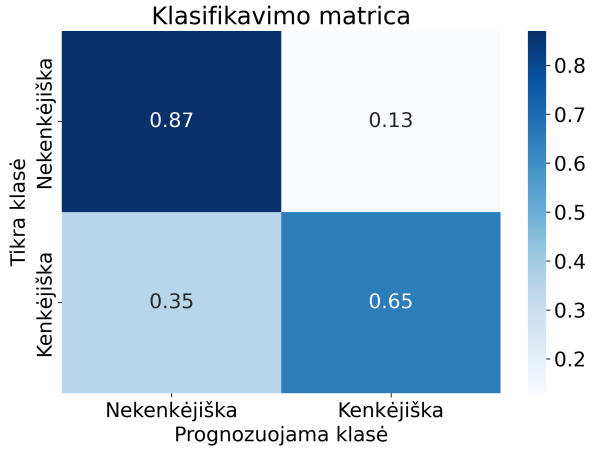
Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,444	0,887	0,592
Kenkėjiška	0,887	0,445	0,593
Svertinis vidurkis	0,739	0,592	0,592
		Tikslumas:	0,592

[6] *Originalus* klasifikatorius yra GBDT modelis (žr. 3.1. poskyrį)

3.4.2. MCA transformacijos klasifikatoriaus tikslumo nustatymas

MCA klasifikatorius yra vienas iš standartinių klasifikatorių (šiuo atveju atsitiktinio miško (*angl. random forest*)), naudojantis pakeistos bazės požymius kaip įvestį. Šioje vietoje naudojamas klasifikatorius gali būti bet koks, tačiau turi būti suderinamas su MCA komponenčių kiekio reikalavimais (žr. 3.3. poskyrį).

Kaip ir originalus klasifikatorius, šis negeba išskirti obfuskuotų programų, tad 8 pav. pateikiamoje klasifikavimo lentelėje rodomas prognozuojamos klasės ir visų tai klasei priklausančių duomenų santykis. Eksperimento rezultatai (klasifikavimo metrikos) pateikiami 11-oje lentelėje.



8 pav. *Klasifikavimo matrica*

11 lentelė. *MCA transformacijos klasifikatoriaus metrikos*

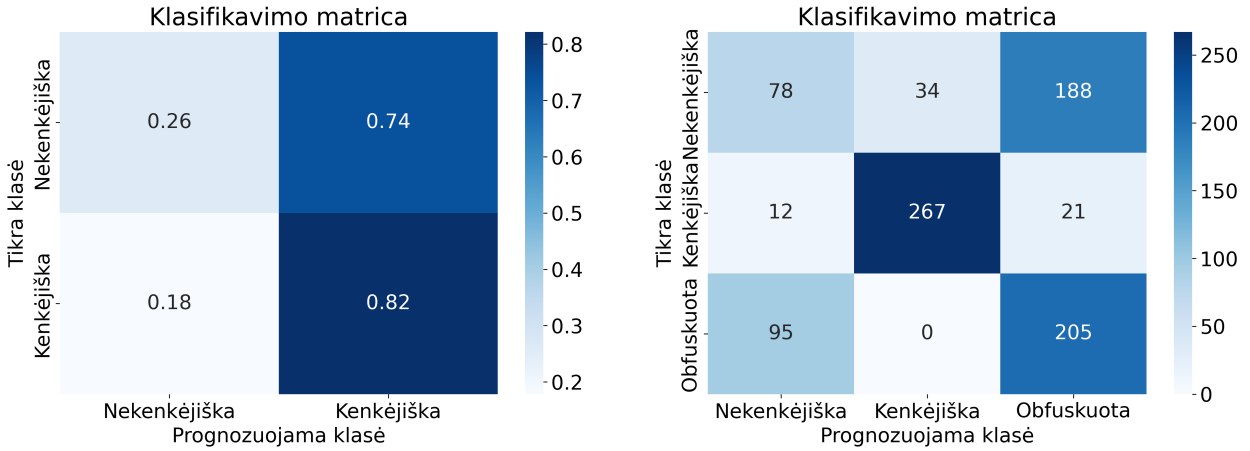
Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,555	0,870	0,678
Kenkėjiška	0,909	0,652	0,759
Svertinis vidurkis	0,791	0,724	0,732
		Tikslumas:	0,724

3.4.3. Modifikuoto LIME pritaikymo AE aptikimui tikslumo nustatymas

Šio eksperimento tikslas yra nustatyti modifikuoto LIME pritaikymo AE aptikimui (žr. 2.1. poskyrį) tikslumą. LIME branduolio plotis paliekamas pagal numatytus nustatymus ($\omega = 0{,}75 \cdot \sqrt{n}$, čia n – požymių skaičius, taigi, $\omega = 0{,}75 \cdot \sqrt{200} \approx 10{,}61$) [RSG16].

Kadangi šis metodas geba aptikti AE – obfuskuotas programas – 9 pav. pateikiamos dvi klasifikavimo lentelės. Pirmoje (9a) – klasifikacijos rezultatai, kai obfuskuota programa laikoma kenkėjiška. Antroje (9b) – išskiriama *obfuskuota* klasė. Kadangi išskyrus šią klasę duomenų rinkinys tampa subalansuotas, klasifikavimo lentelėje pateikiami sveiki skaičiai, atitinkantys modelio prognozuojamų duomenų tai klasei kiekį.

Eksperimento rezultatai (klasifikavimo metrikos) pateikiami 12-oje ir 13-oje lentelėse.



(a) *Neišskiriant obfuskuotų pavyzdžių klasės* ***(b)*** *Išskiriant obfuskuotų pavyzdžių klasę*

9 pav. *LIME pritaikymo AE aptikimui klasifikavimo lentelės*

12 lentelė. *LIME pritaikymo AE aptikimui klasifikatoriaus metrikos (neišskiriant obfuskuotos klasės)*

Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,422	0,260	0,322
Kenkėjiška	0,690	0,822	0,750
Svertinis vidurkis	0,600	0,634	0,607
		Tikslumas:	0,634

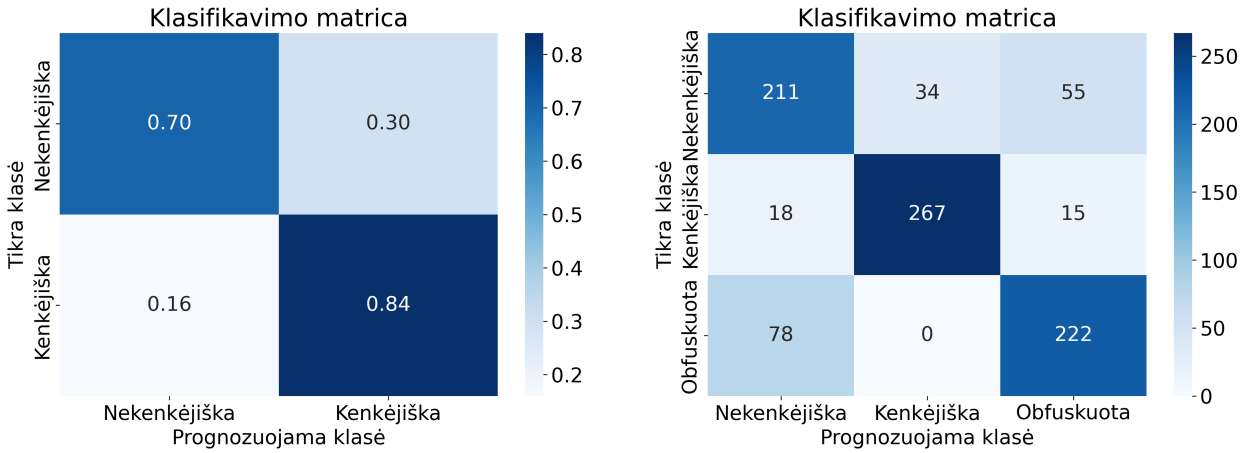
13 lentelė. *LIME pritaikymo AE aptikimui klasifikatoriaus metrikos (išskiriant obfuskuotą klasę)*

Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,422	0,260	0,322
Kenkėjiška	0,887	0,890	0,889
Obfuskuota	0,495	0,683	0,574
Vidurkis	0,601	0,611	0,595
		Tikslumas:	0,611

3.4.4. LIME ir MCA metodų sintezės tikslumo nustatymas

Šio eksperimento tikslas yra nustatyti autoriaus siūlomos modifikuoto LIME ir MCA metodų sintezės varžymosi principais pagrįstų atakų aptikimui tikslumą. Kaip ir praeitame eksperimente, 10 pav. pateikiamos dvi klasifikavimo lentelės – išskiriant ir neišskiriant *obfuskuotos* klasės.

Eksperimento rezultatai (klasifikavimo metrikos) pateikiami 14-oje ir 15-oje lentelėse.



(a) *Neišskiriant obfuskuotų pavyzdžių klasės* ***(b)*** *Išskiriant obfuskuotų pavyzdžių klasę*

10 pav. *LIME ir MCA metodų sintezės klasifikavimo lentelės*

14 lentelė. *LIME ir MCA metodų sintezės klasifikatoriaus metrikos (neišskiriant obfuskuotos klasės)*

Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,687	0,703	0,695
Kenkėjiška	0,850	0,840	0,845
Svertinis vidurkis	0,796	0,794	0,795
		Tikslumas:	0,794

15 lentelė. *LIME ir MCA metodų sintezės klasifikatoriaus metrikos (išskiriant obfuskuotą klasę)*

Klasė	**Preciziškumas**	**Atkūrimas**	**F1**
Nekenkėjiška	0,687	0,703	0,695
Kenkėjiška	0,887	0,890	0,889
Obfuskuota	0,760	0,740	0,750
Vidurkis	0,778	0,778	0,778
		Tikslumas:	0,778

4. Rezultatai ir išvados

Rezultatai

Atlikus tyrimą, gauti šie rezultatai:

1. Apžvelgti ir pritaikyti mokslinėje literatūroje minimi kodo obfuskacijos bei varžymosi principais pagrįstų atakų – AE generavimo – metodai, jų aptikimo strategijos (žr. 1. skyrių).

2. Pasiūlytas naujas AE aptikimo metodas, pritaikomas bet kokiam kenkėjiškų programų detektoriui bei gebantis apdoroti dvejetainius požymius, apjungiantis LIME pritaikymo AE aptikimui ir MCA transformacijos idėjas (žr. 2. skyrių).

3. Atliktas AE aptikimo metodų lyginamosios analizės tyrimas (žr. 3. skyrių) ir nustatytas pasiūlyto AE aptikimo metodo (žr. 2. skyrių) efektyvumas (matuojant modelio tikslumą ir atsižvelgiant į kitas klasifikacijos metrikas), lyginant pasiūlytą metodą su jo sudedamosiomis dalimis (žr. 16 lentelę).

16 lentelė. *Eksperimentų rezultatų (tikslumo metrikų) suvestinė*

Eksperimentas	**Tikslumas (K/N)**[7]	**Tikslumas (K/N/O)**[8]	Δa[9], %
3.4.1. Bazinis	0,592	-	29,6
3.4.2. MCA	0,724	-	16,4
3.4.3. LIME	0,634	0,611	25,4
3.4.4. LIME + MCA	0,794	0,778	9,4

Išvados

1. Autoriaus siūlomas AE aptikimo metodas lyginamosios analizės tyrime pasiekė geriausią rezultatą (didžiausią tikslumą: 79,4 % (**K/N**) ir 77,8 % (**K/N/O**)). Taip pat buvo pranašesnis už likusius modelius visomis kitomis lyginamomis metrikomis (preciziškumu, atkūrimu ir F1).

2. Autoriaus siūlomo metodo Δa yra mažiausia, taigi, tiek varžymosi principais pagrįstų atakų aptikimo logika, tiek pačios varžymosi principais pagrįstos atakos turi mažą įtaką šio klasifikatoriaus tikslumui.

3. Autoriaus siūlomas metodas aptinka 3 iš 4 varžymosi principais pagrįstų atakų su 76 % preciziškumu. Lyginant su LIME metodo pritaikymu AE aptikimui (vieninteliu kitu metodu iš nagrinėtų, gebančiu aptikti *obfuskuotą* klasę), atakų aptikimo santykis yra panašus, tačiau LIME metodo pritaikymo preciziškumas – tik 49,5 %.

[7] Kenkėjiška / Nekenkėjiška

[8] Kenkėjiška / Nekenkėjiška / Obfuskuota

[9] Skirtumas tarp originalaus klasifikatoriaus tikslumo aplinkoje be varžymosi principais pagrįstų atakų (žr. 8 lentelę) ir **tikslumo (K/N)** aplinkoje su varžymosi principais pagrįstomis atakomis

Literatūra ir šaltiniai

[AHH+18] A. Al-Dujaili, A. Huang, E. Hemberg, U.-M. O'Reilly. *Adversarial Deep Learning for Robust Detection of Binary Encoded Malware*. 2018. https://doi.org/10.48550/arXiv.1801.02950. (Žiūrėta 2025-02-28).

[AKF+18] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, P. Roth. *Learning to Evade Static PE Machine Learning Malware Models via Reinforcement Learning*. 2018. https://doi.org/10.48550/arXiv.1801.08917. (Žiūrėta 2024-09-30).

[AV07] H. Abdi, D. Valentin. „Multiple Correspondence Analysis". Iš: *Encyclopedia of measurement and statistics* 2.4 (2007), puslapiai 651–657.

[AW10] H. Abdi, L. J. Williams. „Principal Component Analysis". Iš: *WIREs Computational Statistics* 2.4 (2010), puslapiai 433–459. ISSN: 1939-0068. https://doi.org/10.1002/wics.101. (Žiūrėta 2025-04-17).

[CAD+21] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, D. Mukhopadhyay. „A Survey on Adversarial Attacks and Defences". Iš: *CAAI Transactions on Intelligence Technology* 6.1 (2021), puslapiai 25–45. ISSN: 2468-2322. https://doi.org/10.1049/cit2.12028. (Žiūrėta 2025-04-07).

[CDH+16] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, P. Abbeel. *InfoGAN: Interpretable Representation Learning by Information Maximizing Generative Adversarial Nets*. 2016. https://doi.org/10.48550/arXiv.1606.03657. (Žiūrėta 2024-10-07).

[CSD19] R. L. Castro, C. Schmitt, G. Dreo. „AIMED: Evolving Malware with Genetic Programming to Evade Detection". Iš: *2019 18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications/13th IEEE International Conference On Big Data Science And Engineering (TrustCom/BigDataSE)*. 2019, puslapiai 240–247. https://doi.org/10.1109/TrustCom/BigDataSE.2019.00040. (Žiūrėta 2024-09-23).

[DBL+21] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, A. Armando. „Functionality-Preserving Black-Box Optimization of Adversarial Windows Malware". Iš: *IEEE Transactions on Information Forensics and Security* 16 (2021), puslapiai 3469–3478. ISSN: 1556-6021. https://doi.org/10.1109/TIFS.2021.3082330. (Žiūrėta 2024-10-14).

[DCB+21] L. Demetrio, S. E. Coull, B. Biggio, G. Lagorio, A. Armando, F. Roli. „Adversarial EXEmples: A Survey and Experimental Evaluation of Practical Attacks on Machine Learning for Windows Malware Detection". Iš: *ACM Trans. Priv. Secur.* 24.4 (2021), 27:1–27:31. ISSN: 2471-2566. https://doi.org/10.1145/3473039. (Žiūrėta 2024-09-30).

[FWL+19] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, H. Huang. „Evading Anti-Malware Engines With Deep Reinforcement Learning". Iš: *IEEE Access* 7 (2019), puslapiai 48867–48879. ISSN: 2169-3536. https://doi.org/10.1109/ACCESS.2019.2908033. (Žiūrėta 2024-09-18).

[HT17] W. Hu, Y. Tan. *Generating Adversarial Malware Examples for Black-Box Attacks Based on GAN*. 2017. (Žiūrėta 2024-09-18).

[YPT22] J. Yuste, E. G. Pardo, J. Tapiador. „Optimization of Code Caves in Malware Binaries to Evade Machine Learning Detectors“. Iš: *Computers & Security* 116 (2022), puslapis 102643. ISSN: 0167-4048. https://doi.org/10.1016/j.cose.2022.102643. (Žiūrėta 2024-10-07).

[Puk94] F. Pukelsheim. „The Three Sigma Rule“. Iš: *The American Statistician* 48.2 (1994), puslapiai 88–91. ISSN: 0003-1305. https://doi.org/10.1080/00031305.1994.10476030. (Žiūrėta 2025-04-18).

[RSG16] M. T. Ribeiro, S. Singh, C. Guestrin. „"Why Should I Trust You?": Explaining the Predictions of Any Classifier“. Iš: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. San Francisco California USA: ACM, 2016, puslapiai 1135–1144. ISBN: 978-1-4503-4232-2. https://doi.org/10.1145/2939672.2939778. (Žiūrėta 2025-02-28).

[RSR+18] I. Rosenberg, A. Shabtai, L. Rokach, Y. Elovici. „Generic Black-Box End-to-End Attack Against State of the Art API Call Based Malware Classifiers“. Iš: *Research in Attacks, Intrusions, and Defenses*. Sudarė M. Bailey, T. Holz, M. Stamatogiannakis, S. Ioannidis. Cham: Springer International Publishing, 2018, puslapiai 490–510. ISBN: 978-3-030-00470-5. https://doi.org/10.1007/978-3-030-00470-5_23.

[SB15] J. Saxe, K. Berlin. „Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features“. Iš: *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)*. 2015, puslapiai 11–20. https://doi.org/10.1109/MALWARE.2015.7413680. (Žiūrėta 2024-11-04).

[TKL+21] E. Tcydenova, T. W. Kim, C. Lee, J. H. Park. „Detection of Adversarial Attacks in AI-Based Intrusion Detection Systems Using Explainable AI“. Iš: *Human-centric Computing and Information Sciences* 11.0 (2021), puslapiai 1–1. https://doi.org/10.22967/HCIS.2021.11.035. (Žiūrėta 2025-02-28).

[ZCY+24] F. Zhong, X. Cheng, D. Yu, B. Gong, S. Song, J. Yu. „MalFox: Camouflaged Adversarial Malware Example Generation Based on Conv-GANs Against Black-Box Detectors“. Iš: *IEEE Transactions on Computers* 73.4 (2024), puslapiai 980–993. ISSN: 1557-9956. https://doi.org/10.1109/TC.2023.3236901. (Žiūrėta 2024-09-15).

[ZHZ+22] F. Zhong, P. Hu, G. Zhang, H. Li, X. Cheng. „Reinforcement Learning Based Adversarial Malware Example Generation against Black-Box Detectors“. Iš: *Computers & Security* 121 (2022), puslapis 102869. ISSN: 0167-4048. https://doi.org/10.1016/j.cose.2022.102869. (Žiūrėta 2024-09-14).

[ZZY+22] E. Zhu, J. Zhang, J. Yan, K. Chen, C. Gao. „N-Gram MalGAN: Evading Machine Learning Detection via Feature n-Gram“. Iš: *Digital Communications and Networks* 8.4 (2022), puslapiai 485–491. ISSN: 2352-8648. https://doi.org/10.1016/j.dcan.2021.11.007. (Žiūrėta 2024-09-23).