Kauno technologijos universitetas

Informatikos fakultetas

Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodas

Baigiamasis magistro studijų projektas

Aurimas Kliokys

Projekto autorius

vyresn. lekt. Gedeiminas Činčikas

Vadovas

Kaunas, 2026

Kauno technologijos universitetas

Informatikos fakultetas

Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodas

Baigiamasis magistro studijų projektas

6211BX008

Aurimas Kliokys

Projekto autorius

vyresn. lekt. Gedeiminas Činčikas

Vadovas

dr. Nerijus Šatkauskas

Recenzentas

Kaunas, 2026

Kauno technologijos universitetas

Informatikos fakultetas

Aurimas Kliokys

Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodas

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.

Aurimas Kliokys

Patvirtinta elektroniniu būdu

Kliokys, Aurimas. Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodas. Magistro baigiamasis projektas / vadovas vyresn. lekt. Gedeiminas Činčikas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai, Informacijos ir informacinių technologijų sauga.

Reikšminiai žodžiai: MQTT, ML, lėtos DoS atakos, aptikimas, daiktų internetas, metodas.

Kaunas, 2026. 70 p.

Santrauka

Analizuojant daiktų interneto naudojamus protokolus buvo pastebėta, kad vienas iš dažniausiai naudojamų protokolų yra MQTT dėl jo paprastumo. Tačiau su protokolo paprastumu atsiranda vis didesnė grėsmė kibernetinėms atakoms. Viena iš tokių kibernetinių atakų tipų yra lėtos DoS atakos, kurios išnaudodamos protokolo parametrus gali naudojant mažus tarpininko resursus užblokuoti visas galimas jungtis. Dauguma šiuo metu realizuotų metodų dažniausiai naudoja dirbtinio intelekto algoritmus su jau sugeneruotais duomenimis. Todėl šio darbo tikslas buvo realizuoti metodą, kuris naudodamas hibridinį atpažinimo metodą paremtą taisyklėmis ir mašininiu mokymusi aptiktų kenksmingus klientus tiek realiu metu, tiek jau su turimais duomenimis analizuojant skirtingus laiko intervalus. Norint šį metodą ištestuoti buvo realizuoti kiti papildomi įrankiai, kurie gali generuoti realius klientus ir kenksmingus klientus. Taip pat kitas labai svarbus metodo įrankis yra MQTT protokolo paketų nuskaitymas, kuris buvo realizuotas kaip metodo dalis. Realizavus visus įrankius ir patį metodą, buvo atlikti eksperimentai, kurie patvirtino, kad su naudojamais duomenimis metodas sėkmingai aptiko visus kenksmingus klientus.

Kliokys, Aurimas. Low-Rate Attack Detection Method in MQTT-Based IoT. Master's Final Degree Project / supervisor Senior Lecturer Gedeiminas Činčikas; Informatics Faculty, Kaunas University of Technology.

Study field and area (study field group): Computer Science, Information and Information Technology Security.

Keywords: MQTT, ML, slow DoS attacks, detection, Internet of Things, method.

Kaunas, 2026. 70 pages.

Summary

An analysis of the protocols used in the Internet of Things revealed that MQTT is one of the most used protocols due to its simplicity. However, this simplicity also poses an increasing threat of cyberattacks. One type of such cyberattack is slow DoS attacks, which, by exploiting protocol parameters, can block all possible connections using minimal intermediary resources. Most currently implemented methods primarily use artificial intelligence algorithms with pre-generated data. Therefore, the goal of this work was to implement a method that, using a hybrid detection method based on rules and machine learning, would detect malicious clients both in real time and using existing data by analyzing different time intervals. To test this method, additional tools were developed that can generate both legitimate and malicious clients. Another very important tool of the method is MQTT protocol packet scanning, which was implemented as part of the method. After implementing all the tools and the method itself, experiments were conducted that confirmed that, using the available data, the method successfully detected all malicious clients.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

AMQP (angl. *Advanced Message Queuing Protocol*) – atviro standarto skelbimo ir prenumeratos protokolo architektūra.

CoAP (angl. *Constrained Application Protocol*) – pranešimų perdavimo protokolas, kuris yra pagrįstas REST architektūra.

CVE (angl. *Common Vulnerabilities and Exposures*) – viešai žinomų kibernetinio saugumo spragų ir pažeidžiamumų sąrašas.

DI (angl. *Internet of Things*, *IoT*) – daiktų internetas, kuris be žmogaus įsikišimo gali perduot informacija į internetą.

DL (angl. *Deep Learning*) – gilusis mokymasis, dirbtinio intelekto sritis, naudojanti daugiasluoksnius dirbtinius neuroninius tinklus.

DoS (angl. *Denial of Service*) – kibernetinė ataka, kuria siekiama padaryti sistemą neprieinamą vartotojams.

DTLS (angl. *Datagram Transport Layer Security*) – saugumo protokolas, pritaikytas duomenų perdavimui per UDP srautą be ryšio užmezgimo.

HTTP (angl. *Hypertext Transfer Protocol*) – Hipertekstų persiuntimo protokolas žiniatinklio duomenims persiųsti.

IDS (angl. *Intrusion Detection System*) – įsibrovimų aptikimo sistema stebinti srautą ir įrenginius dėl žinomos kenkėjiškos veiklos.

IETF (angl. *Internet Engineering Task Force*) – atvira bendruomenė, kurianti ir tobulinanti interneto technologijų standartus.

MITM (angl. *Man-in-the-Middle*) kibernetinė ataka, kurios metu įsibrovėlis slapta įsiterpia tarp dviejų komunikuojančių pusių.

ML (angl. *Machine Learning*) – mašininis mokymasis, dirbtinio intelekto sritis, pagrįsta algoritmais, kurie geba patys mokytis iš duomenų.

MQTT (angl. *Message Queue Telemetry Transport*) – pranešimų perdavimo protokolas, naudojant skelbimo ir prenumeratos architektūrą.

NIDS (angl. *Network Intrusion Detection System*) – tinklo įsilaužimų aptikimo sistema, stebinti viso tinklo srautą.

OASIS (angl. *Organization for the Advancement of Structured Information Standards*) – organizacija, kurianti atviro kodo projektus ir standartus.

OSCORE (angl. *Object Security for Constrained RESTful Environments*) – saugumo protokolas, skirtas daiktų interneto įrenginiams naudojantiems CoAP protokolą.

OSI (angl. *Open Systems Interconnection*) – septynių sluoksnių modelis, skirtas paaiškinti tinklo ryšio veikimą.

QoS (angl. *Quality of Service*) – paslaugų kokybės mechanizmas, užtikrinantis prioritetų teikimą tam tikram tinklo srautui.

REST (angl. *Representation State Transfer*) – programinės įrangos architektūros stilius, naudojamas kuriant saityno paslaugas.

SASL (angl. *Simple Authentication and Security Layer*) – karkasas, skirtas autentifikavimo palaikymui interneto protokoluose.

SSL/TLS (angl. *Secure Socket Layer / Transport Layer Security*) – saugaus ryšio technologijos, užtikrinančios duomenų privatumą ir vientisumą.

TCP (angl. *Transport Layer Security*) – transporto lygmens protokolas, skirtas saugumui užtikrinti TCP/IP tinkluose.

TCP/IP (angl. *Transmission Control Protocol/Internet Protocol)* – pagrindinių interneto protokolų rinkinys, reglamentuojantis duomenų persiuntimą tarp įvairių tipų kompiuterių ir operacinių sistemų.

UDP (angl. *User Datagram Protocol*) – greitas transporto sluoksnio protokolas, nenaudojantis patvirtinimo mechanizmų.

XMPP (angl. *Extensible Messaging and Presence Protocol*) – tiesioginių pranešimų protokolas, kuris yra pagrįstas XML.

Įvadas

Daiktų interneto (DI) įrenginiai tapo neatsiejama kasdienio gyvenimo dalimi, naudojami tiek kasdienėms, tiek specialioms užduotims optimizuoti ar automatizuoti. Šios technologijos plėtra atnešė reikšmingų pokyčių įvairiose srityse – nuo išmaniųjų namų iki pramonės automatizavimo. Naudojantis daiktų interneto įrenginiais žmonėms nebereikia rizikuoti gyvybe siekiant surinkti duomenis pavojingose ar sunkiai pasiekiamose vietovėse. Tačiau spartus DI įrenginių tinklų augimas kelia vis daugiau iššūkių, susijusių su tinklo saugumu. Šie saugumo iššūkiai yra ypač svarbūs moderniame pasaulyje, kuriame tinklų patikimumas ir duomenų apsauga tampa kritinėmis užduotimis.

Siekiant užtikrinti DI įrenginių bei perduodamų duomenų saugumą kuriami specializuoti protokolai, kurie ne tik saugo duomenis, bet ir palengvina efektyvų komunikacijos procesą tarp įrenginių. Tačiau daugelis šiuo metu naudojamų DI įrenginių vis dar naudoja senesnius ar paprastesnius protokolus, kurių kūrimo metu į saugumo aspektus nebuvo atsižvelgta. Dėl šios priežasties yra vystomi nauji metodai, skirti padidinti saugumą ir užtikrinti jų integraciją į esamus protokolus.

DI įrenginiai dažnai yra nedideli ir kompaktiški, kad juos būtų patogu nešiotis arba integruoti į mažas ar nepastebimas vietas. Tačiau šie dizaino apribojimai sukuria naujų iššūkių, tokių kaip maža baterijos talpa, ribota atmintis ar procesoriaus našumas, dėl kurių sudėtinga įdiegti daug išteklių reikalaujančius saugumo protokolus.

Augant naujų atakų tipų skaičiui, DI įrenginiams būtina užtikrinti nuolatinį atnaujinimą, tačiau daugeliu atvejų tai nėra įmanoma per tinklą. Todėl kuriant naujus saugumo protokolus ar metodus, svarbu atsižvelgti į jų ateities plėtros galimybes bei pritaikomumą modernioms ir būsimosioms saugumo grėsmėms.

Taip pat, norint apsisaugoti nuo kibernetinių atakų ar jas pastebėti realiu arba analizės metu, reikia realizuoti atakų atpažinimo metodus protokoluose, kad sužinojus apie ataką būtų galima imtis specialių veiksmų, užtikrinančių duomenų saugumą ar tolesnį įrenginių funkcionavimą.

Šio darbo tikslas – realizuoti lėtų atakų aptikimo metodą daiktų interneto tinkluose, naudojančiuose MQTT protokolą.

Norint pasiekti šį tikslą išsikeltos šios užduotys:
1. atlikti daiktų interneto pažeidžiamumo analizę ir jų išvengimo svarbą;
2. išanalizuoti daiktų internete naudojamų protokolų skirtumus ir svarbą;
3. nustatyti atakų priežastis DI tinkluose ir įrenginiuose, kurie naudoja MQTT protokolą;
4. sukurti lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodą;
5. realizuoti lėtų atakų aptikimo prototipą su visais reikiamais papildomais moduliais;
6. atlikti prototipo eksperimentinius tyrimus ir palyginti gautus rezultatus.

Darbas yra sudarytas iš keturių dalių. Pirmąją darbo dalį sudaro analizė, kurioje aptariama, kas yra daiktų internetas, naudojami protokolai pranešimams siųsti ir jų palyginimas. Taip pat įsigilinama į MQTT protokolo atakas ir jų tipus, etapus, protokolo pažeidžiamumus, naudojami metodai, bei realizacijos skirtos apsisaugoti nuo atakų. Antroje dalyje pristatomas atakų atpažinimo metodas ir aprašomas metodo veikimo principas. Trečioje dalyje aprašoma sukurto metodo realizacija ir pati

naudojama sistema. Paskutinėje, ketvirtoje, dalyje atliekamas sukurto metodo eksperimentinis tyrimas.

1. Daiktų interneto tinklų atakų aptikimo problema MQTT protokole

1.1. Problemos apžvalga daiktų interneto įrenginiuose ir tinkluose

Per pastaruosius metus tobulėjant technologijų sektoriui vienas iš labiausiai tobulėjančių ir vis plačiau panaudojamų technologijų yra daiktų internetas (DI) (angl. *Internet of things, IoT*). Patį terminą „Daiktų internetas“ 1999 m. sugalvojo Ashtonas, kuris išpopuliarėjo XX amžiuje. Staigus šuolis naudojant DI atsirado 2011 metais, kai namų automatizavimas ir nešiojami prietaisai buvo pradėti naudoti [1]. Šie įrenginiai gali būti laidinio arba belaidžio tinklo, kuris sujungtas su unikaliai atpažįstamais daiktais, galinčiais apdoroti duomenis ir bendrauti tarpusavyje su žmogaus įsikišimu ar be jo [2]. Didėjant populiarumui ir panaudojimų atvejams „*Strategy Analytics*“ duomenimis, iki 2025 m. pabaigos prijungtų prietaisų skaičius viršys 38 milijardus, o iki 2030 m. planuojama 50 milijardų įrenginių [3]. Žinoma, šie skaičiai dažnai skiriasi pagal naudojamas strategijas ir formules.

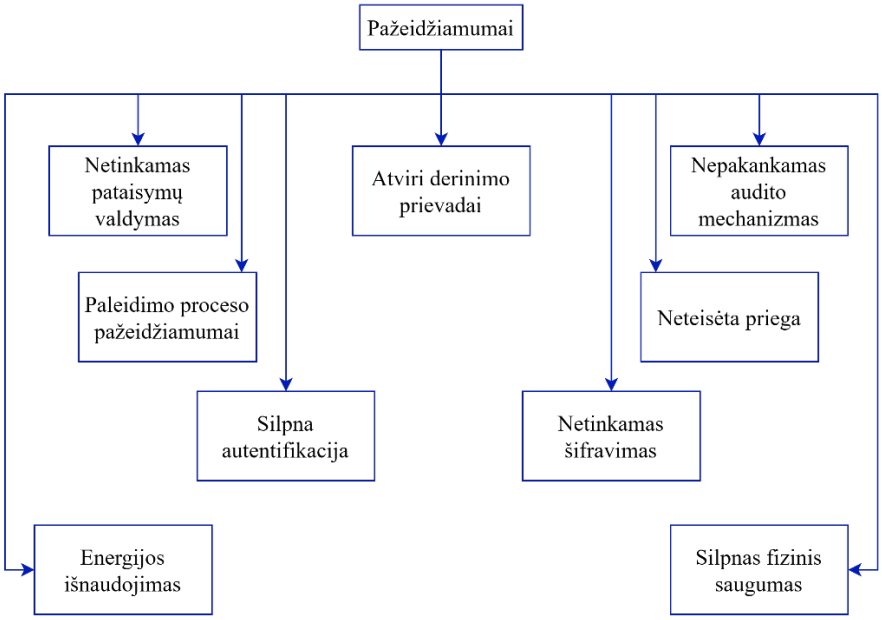
Šių išmaniųjų technologijų pritaikymo galimybės yra itin plačios: nuo medicinos, kur jutikliais stebima žmogaus sveikatos būklė, gamybos procesų bei statistikos monitoringo, žemdirbystės, siekiant įvertinti dirvožemio būklę, iki žmogaus įprasto gyvenimo būdo stebėsenos. Nors šie prietaisai kuriami įvairioms paskirtims, siekiant užtikrinti patikimą daiktų interneto veikimą, esminis prioritetas išlieka energijos vartojimo efektyvumas ir įrenginio saugumas [2].

Energijos tvarumas – didėjantis DI paslaugų naudojimas žmonių gyvenime kelia susirūpinimą dėl energijos suvaržytų DI tinklų tvarumo. Masiškai diegiami DI sensoriai ar kiti prietaisai visose srityse reikalauja nuolatinės energijos tiekimo ilgą laiką dėl naudojimo sudėtingai prieinamose ar kenksmingose vietovėse. Kadangi šių prietaisų dydis mažėja, mažėja ir jų baterijos, tai sutrumpina prietaiso gyvavimą, reikalingą palaikyti prietaisą kuo ilgiau. Taip pat reikia nepamiršti, kad dažniausiai baterijų ar akumuliatorių gyvavimo trukmė yra daug mažesnė nei elektronikos komponentų. Apytiksliai baterijos gyvavimo laikas būna nuo 3–5 metų, kai tuo tarpu elektronikos komponentų apie 10–30 metų. Žinoma, šie skaičiai gali skirtis nuo gamintojo ar aplinkos. Norint prailginti šių prietaisų gyvavimą yra naudojama energijos surinkimas iš supančios aplinkos pasitelkiant radijo dažnių signalus, saulės energiją, vėją, vandenį ar žmogaus kūno šiluminę ar mechaninę energiją [2]. Taip pat daugelio daiktų interneto įrenginių gautus jutiklių duomenis reikia apdoroti ir apskaičiuoti, tačiau skaičiavimai reikalauja labai daug energijos, todėl mažos galios daiktų interneto įrenginiai negali atlikti sudėtingų skaičiavimo užduočių. Be to, nedidelės galios daiktų interneto įrenginiai taip pat turi ribotus skaičiavimo išteklius ir vos gali patenkinti kai kurių užduočių vėlavimo reikalavimus. Siekiant taupyti energiją ir kartu patenkinti vėlavimo reikalavimus, vienas iš metodų yra perduoti visus veiksmus atlikti serveriams [4]. Naudodami šį metodą galime sumažinti įrenginio skaičiavimo galios ir operatyviosios atminties poreikį, tai dar labiau sumažina energijos suvartojimą

Saugumo tvarumas – norint užtikrinti tvarų DI, reikia atsižvelgti į duomenų ir įrenginio saugumą [2]. DI įrenginiai yra atakuojami ne tik kenkėjiškos programos, bet ir techninės įrangos lygmeniu. Kadangi šie daiktų interneto įrenginiai kartais gali būti įdiegti nuotoliniu būdu, įgalioti asmenys gali sukurti tam tikrus pašalinius objektus arba nutekėjimo kanalą daiktų interneto sistemos aparatinėje įrangoje [5]. Todėl norint užtikrinti kuo didesnį sistemos saugumo tvarumą, paties įrenginio komponentuose reikia atlikti nuodugnų testavimą kūrimo etape. Kitas minėtas būdas, reikalaujantis užtikrinti saugumo tvarumą, yra įrenginio duomenų saugumas, kuriam užtikrinti reikia atkreipti dėmesį į šiuos aspektus:

1. konfidencialumas,
2. prieinamumas,
3. autentifikavimo patvirtinimas,
4. autorizavimas,
5. vientisumas,
6. neatšaukiamumas [3].

Išsiaiškinus, kas yra daiktų internetas, jo veikimą, panaudojimo sritis ir tvarumo reikalavimus, galima pereiti prie problemų, su kuriomis susiduria DI įrenginiai ir tinklai. Visais laikais pagrindiniai pažeidžiamumai daiktų interneto įrenginiuose ir tinkluose išlieka panašūs. Šiuos pagrindinius pažeidžiamumus galima suskirstyti ir matyti grafike (žr. **1 pav.**), kuris išvardija pagrindinius pažeidžiamumus.



1 pav. Daiktų interneto pažeidžiamumai [2]

Iš **1 pav.** galime matyti tokius DI pažeidimus kaip:

1. Daiktų interneto mazgų fizinis saugumas (angl. *Physical Security of IoT Nodes*) – DI įrenginiai dažnai yra diegiami atokiose ir nesaugiose vietose, todėl yra lengvai prieinami fizinėms atakoms, tokioms kaip vandalizmas ar vagystė. Mechaninė ar elektroninė žala gali būti žingsnis į kitas atakas. Pavogę daiktų interneto įrangą, įsilaužėliai gali gauti prieigą prie daiktų interneto sistemos funkcionalumo ar įgaliojimų ir tada panaudoti kitus, prijungtus įrenginius, kad padarytų fizinę žalą naudotojams [6]. Siekiant išgauti šią konfidencialią informaciją iš pavogtos techninės įrangos, dažnai pasitelkiama atvirkštinė inžinerija (angl. *reverse engineering*).
2. Atviros derinimo jungtys (angl. *Open Debugging Ports*) – kūrimo metu paliktos atviros jungtys su galimybe skaityti, vykdyti ar keisti sistemos veikimo funkcionalumą taip gaunant prieigą ne tik prie įrenginio, bet ir prie viso tinklo. Šis pažeidžiamumas gali būti dėl netyčinės gamintojo klaidos ar specialiai, kad pats naudotojas galėtų atsinaujinti savo prietaisą, kai tokios galimybės nėra per tinklą automatiškai.
3. Energijos išnaudojimas (angl. *No Energy Harvesting*) – išteklių išeikvojimo atakos gali trikdyti ryšio kanalus ir gali sukelti platų neleistiną daiktų interneto išteklių, pavyzdžiui: pralaidumo, atminties, procesoriaus laiko, disko vietos, naudojimą. Dėl to iškraunama daiktų interneto

įrenginio baterija ir sutrinka paslaugų tiekimas teisėtiems naudotojams, kol nebus vėl pakrauta, kas gali būt sudėtinga tam tikrais atvejais [2].

4. Silpnas autentiškumo nustatymas (angl. *Weak Authentication*) – autentiškumo nustatymo mechanizmai laikomi vienu didžiausių iššūkių daiktų interneto ekosistemoje. Šis procesas, apimantis subjektų tapatybės patikrą, atlieka dvi esmines funkcijas: leidžia patvirtinti viešajame tinkle veikiančių nuotolinių įrenginių teisėtumą bei užkerta kelią neteisėtiems subjektams įsiterpti į saugų ryšį [3]. Šis pažeidžiamumas dažniausiai atsiranda siekiant optimizuoti protokolus ir mažinti jų sąnaudas, dėl to pasirenkami supaprastinti, tačiau nepakankamai saugūs tapatybės patvirtinimo metodai.
5. Netinkamas šifravimas (angl. *Improper Encryption*) – išmaniųjų programų vientisumas tiesiogiai priklauso nuo jutikliais surenkamų duomenų saugumo. Duomenys privalo išlikti saugūs ir patikrinami tiek jų šaltinyje, tiek perdavimo metu [2]. Tačiau dėl ribotos skaičiavimo galios ir įrenginių išteklių apribojimų neįmanoma tikėtis, kad visi DI galės naudoti šifravimo ir (arba) iššifravimo algoritmus, ypač sudėtingus ir reikalaujančius daug skaičiavimų [7].
6. Neteisėta prieiga (angl. *Unauthorized Access*) – daugelis pigių ir paprastų DI įrenginių pasižymi silpnais gamykliniais nustatymas ir prisijungimo duomenimis, kuriuos galima nesunkiai ištestuoti rankiniu būdu arba naudojant automatizuotus metodus. Šių prisijungimo duomenų daugelis naudotojų net nežino arba tiesiog nesureikšmina pakeist, todėl nukenčia pačių saugumas. Išnaudodami šį paprastą žmonių neapdairumą nusikaltėliai gali lengvai, be sunkesnių problemų gauti visą prieigą prie įrenginio duomenų ir kitų tinklo mazgų.
7. Netinkamas audito mechanizmas (angl. *Insufficient Audit Mechanism*) – nėra mechanizmo, kuris leistų išsaugoti daiktų interneto prietaisuose atliktų veiksmų žurnaluose ir laiku juos patikrinti, kad būtų užtikrintas jų saugumas. Tokie įvykiai, kaip programos klaidos, sėkmingi ar nesėkmingi prisijungimo bandymai, autentifikavimo bandymai, autorizavimo bandymai, turėtų būti registruojami užšifruotuose žurnalų failuose [2], kad būtų galima atsekti vartotojo ar atakos atlikus žingsnius.
8. Netinkamas pataisymų valdymas (angl. *Improper patch management*) – nesaugioje aplinkoje esantys neatnaujinti įrenginiai tampa lengvu taikiniu užpuolikams, kurie nuolat ieško silpnų vietų. Papildomą riziką kelia „nulinės dienos“ (angl. *zero-day*) pažeidžiamumai, reikalaujantys itin operatyvios gamintojo reakcijos, tačiau daugeliu atvejų tokio palaikymo DI įrenginių rinkoje trūksta [8] ir gali užtrukti net kelelius metus, kol bus ištaisyta.
9. Paleidimo proceso pažeidžiamumai (angl. *Boot process Vulnerabilities*) – vykstant paleidimo procesui, visi trys pagrindiniai komponentai – mikroprograminė įranga (angl. *firmware*), įkrovos tvarkyklė (angl. *bootloader*) ir paleidimo seka – yra pažeidžiami, todėl gali būti panaudoti kaip įsilaužimo vektorius [2]. Piktavaliai, išbandydami šio proceso veikimo principus, gali nepastebimai įterpti kenkėjišką kodą, o tai leidžia apeiti įrenginio saugumo patikras bei įsilaužimo aptikimo sistemas.

Galime pastebėti, jog saugumas yra esminis daiktų interneto sistemos bruožas, susijęs su unikaliomis saugos priemonėmis, kurios taip pat yra labai svarbios, kad prietaisas būtų patikimas ir turėtų saugumo požymių. Planuojant saugumo strategiją, nuo pat pradžių būtina iš naujo įvertinti konkrečius įrenginius bei ištisus tinklus, atsižvelgiant į nepakankamą jų apsaugą ir netinkamus šifravimo metodus. Milijardai tarpusavyje sujungtų išorinių sistemų įvairiose technologijų srityse rodo, kaip stipriai ši DI aplinka padidino bendrą sistemų sudėtingumą. Saugumo problemų mastas sparčiai auga kartu su prijungtų išmaniųjų įrenginių skaičiumi, todėl saugumo klausimai turi būti sprendžiami kompleksiškai vertinant visą sistemą. Be to, dėl tradicinių apsaugos metodų specifikacijos ir ribotų

DI įrenginių išteklių, kaip ribotos energijos, standartinės saugumo inovacijos ne visada gali būti tiesiogiai pritaikomos. Įvairios grėsmės, įskaitant numatomas ar atsitiktines, kelia pavojų šių įrenginių išlikimui bei apsaugai, todėl įrenginių atsparumas tampa kritiniu iššūkiu. [9].

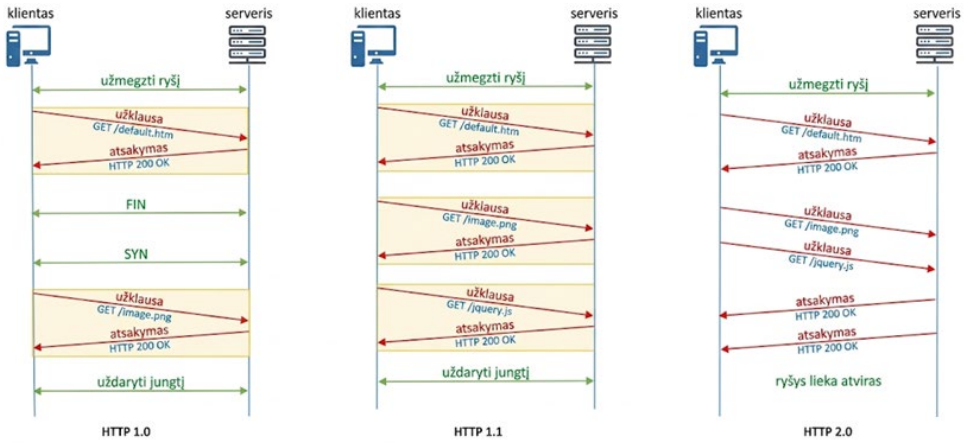
Nepavykus apsaugoti daiktų interneto tinklo nuo esamų ar ateityje atsirandančių pažeidžiamumų, kyla didelė saugumo rizika. Tobulėjant DI technologijoms, sudėtingėja ir piktavalių taikomi metodai. Jie gali pasisavinti asmeninę informaciją, ją klastoti arba sutrikdyti sistemų veikimą taip sukeldami neigiamų pasekmių. Įvykus sėkmingai atakai jos gali lemti milžiniškus finansinius nuostolius ar net kelti tiesioginį pavojų žmonių gyvybei. Norint išspręsti šias problemas yra kuriami nauji ar modifikuojami esami DI komunikacijos protokolai, kurie užtikrintų aukštesnį saugumo lygį tam tikrais aspektais.

1.2. Protokolų apžvalga

Besivystančios daiktų interneto technologijos suteikia didelį potencialą naujiems sprendimams ir pažangesnių programų, sistemų ar struktūrų kūrimui, kurie gali pagerinti visų ar tam tikrų sektorių aspektus. Per pastarąjį dešimtmetį duomenų rinkimą palengvino naujausi komunikacijos tinklų ir protokolų pasiekimai, daugiausia žemesniuose OSI sluoksniuose, tai yra fiziniame, ryšio ir tinklo sluoksniuose. Tačiau be jų veiksmingam duomenų rinkimui ir dalijimuisi labai svarbūs ir viršutinio sluoksnio protokolai. Daiktų interneto kontekste taikomojo lygmens protokolai dažniausiai reiškia apatinę TCP/IP modelio taikomojo lygmens dalį, atitinkančią OSI sesijos lygmenį, ir jie taip pat paprastai vadinami pranešimų perdavimo protokolais. Skiriamos dvi pagrindinės kategorijos: protokolai, kurie veikia pagal užklausos ir atsakymo modelį, ir protokolai, kurie veikia pagal skelbimo ir prenumeratos modelį. Jie gali būti naudojami tik tam tikrose daiktų interneto ryšio architektūros dalyse arba visoje, tai yra, nors daiktų interneto taikomasis protokolas gali būti naudojamas daiktų interneto prietaisų ir daiktų interneto vartų ryšiui palaikyti, kiti protokolai gali būti naudojami tarp vartų ir debesies arba debesies ir galutinio naudotojo [10]. Toliau šiame skyriuje bus aptarti keli taikomojo lygmens protokolai, naudojami informacijai perduoti, jų veikimas, pliusai, minusai ir saugumo aspektai:

- HTTP (angl. *Hypertext Transfer Protocol*),
- MQTT(angl. *Message Queue Telemetry Transport*),
- CoAP (angl. *Constrained Application Protocol*),
- AMQP (angl. *Advanced Message Queuing Protocol*),
- XMPP (angl. *Extensible Messaging and Presence Protocol*);

1.2.1. HTTP protokolas



2 pav. HTTP protokolo veikimo algoritmai [11]

HTTP (angl. *Hypertext Transfer Protocol*) protokolą 1989 m. sukūrė Timas Bernersas-Lee su kitais atstovais. Jis, šiuo metu yra kuruojamas Interneto inžinerijos darbo grupės (angl. *IETF*) [12]. Šis protokolas yra universalus ir nepalaikantis būsenos (angl. *stateless*), tai reiškia, kad kiekviena užklausa apdorojama kaip atskiras, su ankstesniais veiksmais nesusijęs įvykis, kuris dažniausiai naudojamas perduoti duomenis per pasaulinį žiniatinklį. Viena iš pagrindinių HTTP ypatybių yra derybos dėl duomenų pateikimo turinio. Tai leidžia skirtingų sričių įrenginiams, sukurtiems nepriklausomai nuo duomenų, jais dalytis. HTTP yra užklausos atsakymo protokolas, kuriame klientas siunčia užklausos pranešimą, o kompiuteris arba kitas sujungimo dalyvis sukuria atsakymo pranešimą [11].

Iš anksčiau pateikto paveikslėlio (žr. **2 pav.**) galima matyti HTTP protokolo veikimų principus tarp išleistų skirtingų versijų. HTTP 1.0 versija kiekvienai užklausai atidaro naują jungtį, kad gautų užklausą ir atsakymą, o paskui ją uždaro, tai labai apkrauna tinklą. HTTP 1.1 versijoje ši problema buvo išspręsta, atsiradus galimybei vienoje jungtyje siųsti kelias užklausas ir gauti atsakymus, taip padidinant protokolo našumą ir sumažinant jungčių kiekį. Naujausioje HTTP 2.0 versijoje buvo realizuotas lygiagretumas, tai dar labiau optimizavo protokolo efektyvumą, leidžiant siųsti ir gauti kalias užklausas vienu metu vienoje jungtyje.

Pastaruoju metu HTTP dažnai siejama su REST architektūra, siekiant palengvinti įvairių subjektų sąveiką naudojant žiniatinklio paslaugas. Naudojant HTTP ir REST protokolų deriniu leidžiama įrenginiams naudoti standartizuotas CRUD operacijas: kūrimo (angl. *CREATE*), skaitymo (angl. *READ*), atnaujinimo (angl. *UPDATE*) ir naikinimo (angl. *DELETE*). Taip galima sukurti REST modelį, skirtą įvairių sričių daiktų interneto įrenginiams. Norint atvaizduoti duomenis yra naudojamas atviro standarto formatas JSON, kuris yra plačiai paplitęs. Nors toks sprendimas užtikrina patikimą didelių duomenų kiekių perdavimą, jis nėra optimizuotas ribotų išteklių aplinkai. Viena iš pagrindinių problemų yra ta, kad daiktų interneto įrenginiai, turintys ribotus išteklius, nedidelius duomenų kiekius siunčia nereguliariai, o kiekvieną kartą nustatant TCP ryšį atsiranda didelis vėlavimas ir didelės protokolo sąnaudos [10].

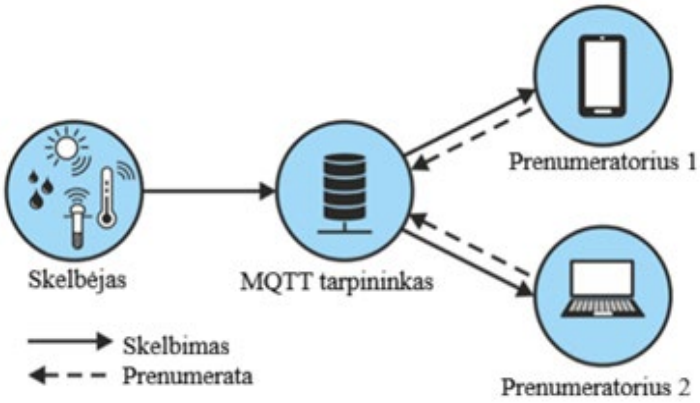
Kalbant apie saugumą, HTTP protokolas šiuo metu naudoja TLS 1.3 protokolą, kuris užtikrinta saugų ir šifruotą ryšio kanalą, žinomą kaip HTTPS. Nors šis HTTPS ir REST protokolų derinys gana dažnai taikomas eksperimentiniuose ar kituose daiktų interneto įgyvendinimuose, kuriems nekeliami dideli

reikalavimai ir apribojimas. Tačiau šie protokolai nėra iki galo tinkami daiktų interneto įrenginiams, bendraujantiems su debesimi, dėl savo sudėtingumo, didesnių antraščių ir didesnių energijos sąnaudų [10].

Naudojamo HTTP REST protokolo aspektai yra šie:

1. Ištekliai – laikomas objektas, turintis susijusių duomenų tipus, ryšį su kitais ištekliais ir su juo veikiančių funkcijų rinkinį. Ištekliais laikoma bet kokia informacija ar duomenys, pavyzdžiui, tekstinis dokumentas, paveikslėlis, laikinė paslauga arba kitų išteklių rinkinys.
2. Užklausos metodai – HTTP protokolo veiksmai, nurodantys, ką serveris turi daryti su duomenimis. Vieni pagrindinių metodų yra CRUD funkcijos.
3. Užklausos antraštės – tai papildoma meta informacija, kurią sudaro duomenų formatas, autorizavimo duomenys, spartinimo nustatymai ir slapukai. Antraštės aprašo siunčiamą turinį, bet pačios duomenų savyje neturi.
4. Užklausos turinys – duomenys, kurie siunčiami kartu su užklausa, kuriuos serveris turi apdoroti.
5. Atsakymo turinys – serverio suformuotas atsakymas, kuriame pateikiami apdoroti duomenys arba informacija apie atliktą veiksmą.
6. Atsakymo būsenos kodas – skaitiniai kodai, nurodantys užklausos atsakymo rezultatą [13].

1.2.2. MQTT protokolas



3 pav. MQTT protokolo veikimo algoritmas [10]

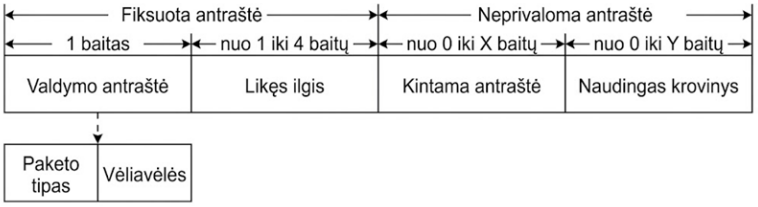
MQTT (angl. *Message Queue Telemetry Transport*) – pranešimų perdavimo protokolas, kurį 1999 m. pristatė Andy Stanfordas-Clarkas („IBM“) ir Arlenas Ninpperis („Arcom“) [10]. Šis protokolas yra naudojamas pranešimų perdavimui, naudojant skelbimo ir prenumeratos architektūrą. MQTT protokolas dažniausiai yra naudojamas informacijai ir matavimams iš nutolusių jutiklių rinkti ir perduoti į servisus. Dėl savo paprastumo ir lengvumo jis ypač tinkamas sistemoms, kuriose duomenų srauto pralaidumas yra ribotas [12].

Kaip pavaizduota **3 pav.**, MQTT protokolas pranešimus perduoda naudojant centrinį įrenginį, kuris yra tarpininkas (angl. *broker*) ir atsakingas už ryšių apdorojimą tarp įrenginių. Kiekvienas įrenginys gali prenumeruoti daugiau nei vieną temą, kuri yra kiekvieno paskelbto pranešimo adresas, veikiantis kaip virtualus kanalas. Taip pat prenumeratorius gaus kiekvieną pranešimą, kuris buvo paskelbtas tam tikroje temoje. Verta paminėti, kad klientai gali skelbti pranešimus šiose temose, tuo pat metu gaudami pranešimus iš tos pačios arba kitos temos, taigi klientas gali būti leidėjas ir abonentas tuo pačiu metu [14]. MQTT protokolas gali veikti keliais QoS (angl. *Quality of Service*) lygiais:

1. nulinis lygis yra žemiausias lygis, tačiau greičiausias, nes jam nereikia gauti gauto pranešimo patvirtinimo;
2. pirmasis lygis užtikrina, kad bent kartą išsiųstas pranešimas bus gautas;
3. antrasis lygis užtikrina pranešimų pristatymą ir kontroliuoja dubliavimąsi.

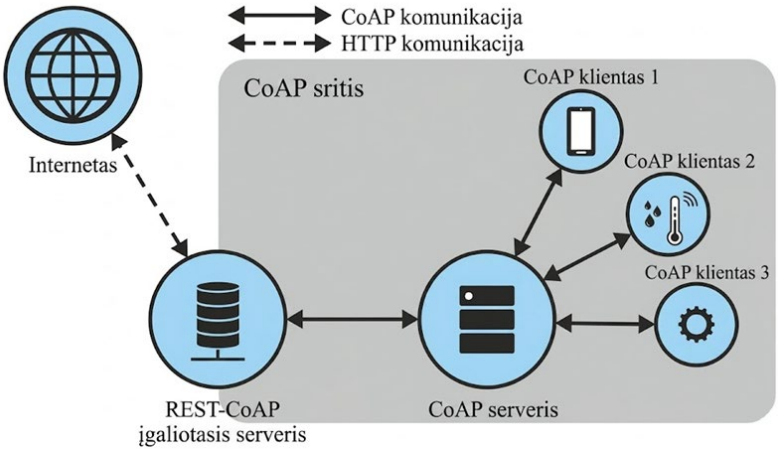
MQTT protokolas buvo sukurtas siekiant kuo labiau sumažinti dažnių juostos plotį ir energijos suvartojimą komunikuojant, tačiau neatsižvelgiant į saugumą. Norėdami bent kažkaip autentifikuoti naudoja vartotojo vardą ir slaptažodį, kartais net ir jų nenaudoja, o dėl papildomo saugumo keli tarpiniai brokeriai realizuoja įvairius papildomus mechanizmus. Todėl MQTT saugumas priklauso nuo pasirinkto tarpininko. Žinoma, reikia paminėti, kad protokolas naudoja SSL/TLS, jei naudotojai tai realizuoja, tačiau jie mažina DI įrenginių našumą [14] ir garantuoja tik nuo taško iki taško šifravimą.

MQTT protokolo siunčiamas paketas susideda iš keturių pagrindinių dalių, kaip pavaizduota **4 pav.** Kiekviename MQTT pranešime yra 2 baitų fiksuota antraštė, o kintama antraštė ir naudingoji apkrova yra neprivalomos. Fiksuotos antraštės pirmasis baitas nurodo valdymo paketo tipą bei atitinkamas vėliavėles. MQTT palaiko 14 rūšių valdymo paketų, tokių kaip, prisijungti (angl. *CONNECT)*, prisijungimo atsakymas (angl. *CONNACK*), paskelbti (angl. *PUBLISH*) ir kitus, skirtus ryšiui užmegzti bei pranešimams tarp klientų ir tarpininko perduoti [15].



4 pav. MQTT paketo formatas [15]

1.2.3. CoAP protokolas



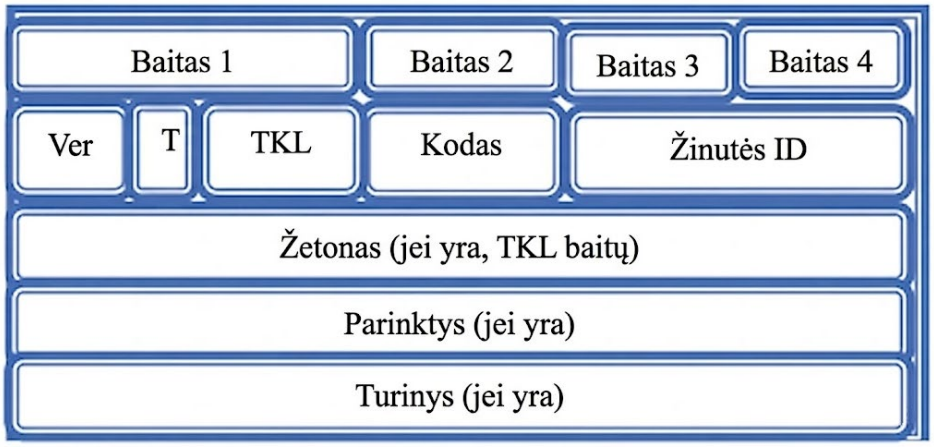
5 pav. CoAP protokolo veikimo algoritmas [10]

CoAP (angl. *Constrained Application Protocol*) – 2014 m. birželio mėn. išleistas pranešimų perdavimo protokolas, kuris yra pagrįstas REST architektūra. Dažnu atveju DI įrenginiuose negalima naudoti HTTP protokolo, dėl atminties ir skaičiavimo pajėgumų ribojimų. Šiai problemai išspręsti

buvo sukurtas šis CoAP protokolas, kuris naudoja modifikuotas HTTP funkcijas, kurios yra labiau pritaikytos daiktų interneto įrenginiams [12]. Tai reiškia, kad protokolą galima lengvai sujungti su jau naudojamu HTTP protokolu [16]. Protokolas sukurtas siekiant užtikrinti patikimumą esant ribotam dažnių juostos pločiui ir didelei perkrovai, nes jis turi mažas tinklo pridėtines išlaidas ir mažas energijos sąnaudas. Saugumui užtikrinti pasirenkami numatytieji DTLS parametrai, atitinkantys 3072 bitų RSA raktą [17].

Kaip protokolas veikia galima matyti **5 pav.**, kurį sudaro CoAP klientai, CoAP serveris su kuriuo komunikuoja klientai ir REST-CoAP įgaliotasis serveris, kuris komunikuoja su internetu ir CoAP serveriu.

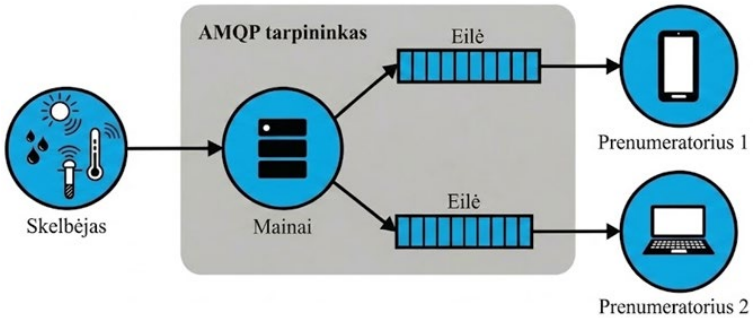
Šis protokolas naudoja UDP, siekiant sumažinti pralaidumo poreikį ir sąnaudas, lyginant su TCP [18]. Kai CoAP protokolo klientas siunčia vieną ar daugiau užklausų į serverį, atitinkamas atsakymas siunčiamas ne per jau egzistuojantį ryšį, o asinchroniškai naudojant CoAP pranešimus. Nors naudojant UDP sumažėja protokolo patikimumas, jį galima padidinti atitinkamai pakoregavus protokolą [10]. Vis dėlto srauto kontrolės trūkumas siunčiant nepatvirtinamus pranešimus gali sukelti tinklo perkrovas. Kitas protokolo trūkumas yra paties protokolo naujumas, nors protokolas ir populiarėja, tačiau vis dar vystosi, o dėl atviro kodo pobūdžio atsiranda daugelis skirtingų versijų, kurios gali būti nesuderinamos tarpusavyje [19]. Kadangi CoAP yra plačiai prieinamas ir veikia per UDP, jis naudoja DTLS sluoksnį (o ne SSL/TLS), dažniausiai palaikant RSA ir AES arba ECC ir AES algoritmus. Kaip paslaugų lygmens protokolas, CoAP taip pat gali palaikyti kitus taikomųjų programų lygmens saugos sprendimus, kaip OSCORE (angl. *Object Security for Constrained RESTful Environments*) [20].



6 pav. CoAP pranešimo formatas [16]

Iš pateikto CoAP pranešimo formato (žr. **6 pav.**) galima matyti, kad pirmasis baitas apima protokolo versiją (*Ver*), tipo lauką (*T*, žetoną) ir žetono ilgį (*TKL*). T yra tipo laukas, kurį sudaro pagrindinė pranešimo tipo informacija. *TKL* reiškia žetono lauko dydį baitais. Toliau yra Kodas laukas kuris apima konkretesnę pranešimo tipo informaciją. Toliau yra žinutės ID laukas. Pranešimo ID laukas yra unikalus ID. Šio unikalaus ID paskirtis – sekti pranešimus ir atskirti tikėtinus pasikartojimus. Paskutinėje CoAP pranešimo struktūros dalyje yra laukas turinys (angl. *Payload*), kuriame yra perduodama informacija [16].

1.2.4. AMQP protokolas



7 pav. AMQP protokolo veikimo algoritmas [10]

AMQP (angl. *Advanced Message Queuing Protocol*) – tai 2003 m. pristatyta atviro standarto skelbimo ir prenumeratos protokolo architektūra, kurią 2011 m. standartizavo OASIS (angl. *Organization for the Advancement of the Structured Information Standards*) [18]. Šis protokolas apima žinučių eilių kūrimą, jų maršrutizavimą, saugumą bei patikimumą aplikacijų lygmenyje. Taip pat šis protokolas pagrįstas dvejetainiu duomenų perdavimo lygiu, todėl leidžia efektyviai keistis pranešimais įvairiose aplinkose [17]. Saugumo aspektu, AMQP palaiko autentiškumo patvirtinimą ir (arba) šifravimą, pagrįstą SASL ir (arba) TLS [11].

Kaip galima matyti iš **7 pav.,** šią protokolo struktūrą sudaro trys pagrindiniai elementai:
1. mainai (angl. *Exchange*) – atsakingas už pranešimų surinkimą iš leidėjų ir jų maršrutizavimą į tinkamas eiles pagal nustatytas taisykles.
2. eilė (angl. *Queue*) – pranešimų eilės, kuriose saugomi pranešimai, kol juos perskaito prenumeratoriai.
3. susiejimas (angl. *Binding*) – ryšys tarp mainai ir eilė, leidžiantis apibrėžti pranešimų prioritetus naudojant įvairias strategijas, tokias kaip tiesioginė (angl. *direct*), teminė (angl. *topic*) ar plačioji (angl. *fanout*) [18].

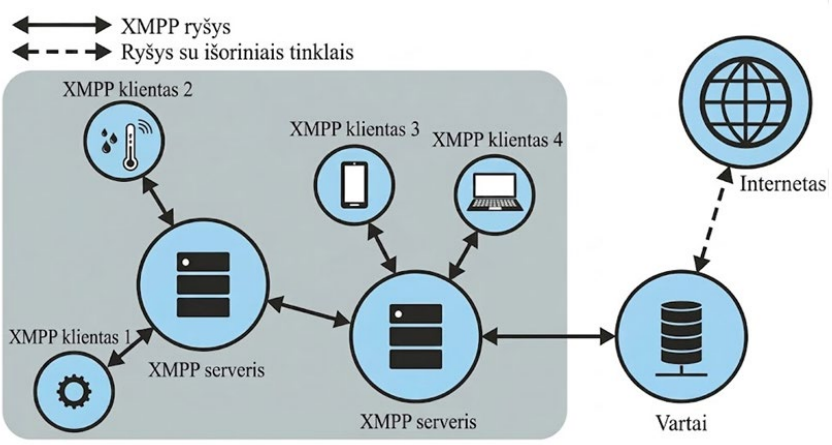
AMQP protokolas naudoja TCP ryšį, kuriame yra naudojamos trys paslaugos kokybės (angl. *Quality of Service*, QoS) lygiai kaip ir prieš tai aptartame MQTT protokole. QoS 0 nereikalauja gauti patvirtinimo iš gavėjo. QoS 1 reikalauja gauti gavimo patvirtinimą. Paskutinis QoS 2 lygis jau garantuoja, kad pranešimas bus pristatytas tik vieną kartą be pasikartojimų.

Fiksuota Antraštė (8 baitai)	Papildoma Antraštė	Krovinys (pasirenkamas ir kintamas)

8 pav. AMQP protokolo paketo struktūra [21]

Patį protokolą sudaro nedidelė struktūra, kurią sudaro fiksuota antraštė, papildoma antraštė ir duomenys, kurie yra atvaizduoti **8 pav.**

1.2.5. XMPP protokolas



9 pav. XMPP protokolo veikimo algoritmas [10]

XMPP (angl. *Extensible Messaging and Presence Protocol*) tiesioginių pranešimų protokolas, pagrįstas XML (angl. *Extensible Markup Language*), kurį 2011 metais išleido organizacija IETF [12]. Šis protokolas turi galimybę palaikyti tiek sinchroninį, tiek asinchroninį skelbti-prenumeruoti modelį, kuris leidžia šį protokolą panaudoti plačiai didelio masto ir realaus laiko sistemose. Be to, jis palaiko per TCP veikiančias sistemas, kaip VoIP signalų, vaizdo įrašų, failo perdavimo ir daugelį kitų [17]. Įrenginiai, naudojantys šį protokolą ryšio užmezgimui, naudoja unikalų kiekvieno mazgo adresą, kurie yra vadinami JID (angl. *Jaber IDentifiers*). XMPP pranešimą sudaro srautas ir eilė, o XML srautas perduoda XML eilutę kaip duomenis [18]. Šiame protokole galima apibrėžti tris strofų tipus XMPP protokole:

1. Buvimo (angl. *presence*) – kontroliuoja ir praneša apie subjekto prieinamumą ir atlieka subjekto registraciją kito subjekto sąraše. Serveris perduoda informaciją visiems subjekto sąrašo įrašams, subjektams, kurie yra užregistruoti šiam subjektui.
2. Pranešimo (angl. *message*) – naudojama duomenims iš subjekto į subjektą siųsti, nurodant pranešimo pavadinimą ir turinį. Pranešimų strofa negauna patvirtinimo iš gaunančiojo subjekto, kuris yra klientas arba serveris.
3. Iq (angl. *info/query*) – sukuria siuntėjų ir gavėjų poras. Ji naudojama informacijai iš serverio gauti arba kai kuriems nustatymams siųsti. Mechanizmas panašus į HTTP GET ir POST metodus. Kiekviena iq strofa turi ID, kuris ją sieja su atsakymo strofa [10].

9 pav. pavaizduota decentralizuota XMPP architektūra, pagrįsta kliento-serverio modeliu. Atvaizduotoje diagramoje matoma, kaip skirtingi XMPP klientai (nuo jutiklių iki vartotojų įrenginių) jungiasi prie serverių, kurie gali sąveikauti tarpusavyje (vadinama federacija), o ryšį su išoriniais tinklais užtikrina tinklo sietuvai (angl. *gateways*).

XMPP protokolo viena iš pagrindinių savybių yra saugumas. Skirtingai nuo kitų protokolų, pavyzdžiui, MQTT ir CoAP, kurie remiasi juose neįdiegtu TLS ir DTLS šifravimu, XMPP turi integruotą TLS mechanizmą, užtikrinantį konfidencialumo ir duomenų vientisumo patikimumą. Be to, XMPP naudoja specialų SASL (angl. *Simple Authentication and Security Layer*) profilį subjektams identifikuoti. Tačiau nors protokolas ir užtikrina saugumą, jis turi tam tikrų trūkumų. Vienas iš tokių trūkumų yra XML kalbos naudojimas, dėl kurios siunčiami pranešimai tampa ilgi ir patiria didesnes protokolo sąnaudas [22]. Kita problema yra QoS kontrolės nebuvimas [10].

1.2.6. Protokolų skirtumai saugumo atžvilgiu

Apžvelgus visus protokolus galima padaryti išvadą, kad kiekvienas iš aptartų protokolų turi savitų stipriųjų ir silpnųjų aspektų, priklausomai nuo to, kur ir kaip jie naudojami. Tačiau vienas pagrindinių kriterijų, kuris išsiskiria aptariant šiuos protokolus, yra jų saugumo užtikrinimo būdai. Informacijos perdavimo ir saugojimo kontekste itin svarbu atsižvelgti į potencialias grėsmes bei užtikrinti saugumą. Didžioji dalis aptartų įrenginių naudoja plačiai paplitusį TCP protokolą informacijai perduoti, o UDP perdavimo protokolą naudoja tik CoAP protokolas, kuris sumažina tinklo apkrovą, tačiau negarantuoja, kad išsiųstas pranešimas bus nusiųstas, nes UDP protokolas nenaudoja atsako siuntimo. Vadinasi, CoAP protokolas negarantuoja vientisumo kriterijaus.

Kitas kriterijus, labai svarbus duomenų saugumui yra konfidencialumas, kurį gali užtikrinti duomenų šifravimas. Aptarti protokolai, siekdami išlaikyti nedidelį skaičiavimo sudėtingumą ir mažas išteklių sąnaudas, neturi įdiegtų vidinių saugumo mechanizmų transporto lygmenyje. Todėl dauguma jų remiasi TLS 1.3 kriptografiniu protokolu, išskyrus CoAP protokolas, kuris naudoja DTLS (angl. *Datagram Transport Layer Security*) protokolą. Žinoma, protokolai dažnu atveju nenaudoja tik TLS protokolo, jie naudoja ir kitus papildomus mechanizmus, tokius kaip SASL, kurį naudoja AMQP ar XML šifravimas, kurį naudoja XMPP protokolas.

Kalbant apie autentiškumo patvirtinimą, svarbu užtikrinti, kad perduota informacija gauta iš patikimo šaltinio. Vienas paprasčiausių ir plačiausiai naudojamų metodų – naudotojo vardas ir slaptažodis, kurį palaiko MQTT ir AMQP. Tie protokolai, kurie naudoja TLS 1.3, gali naudotis protokole esančiu TLS sertifikatu, kuris užtikrina autentifikaciją. Dar vienas plačiai paplitęs būdas yra naudoti OAuth 2.0 autorizacijos sistemą, kuri šiuo metu yra viena iš populiariausių būdų užtikrinti autentiškumą ne tik DI įrenginiuose, bet ir daugumoje kitų prie tinklo prijungtų įrenginių.

Nustatyti, kuris iš protokolų yra saugiausias, nėra paprasta, nes jų saugumo lygis priklauso nuo konkrečios konfigūracijos ir taikomų apsaugos priemonių nuo dažniausiai pasitaikančių atakų. Tačiau dažniausios atakos prieš kiekvieną protokolą galima sakyti yra šios: paskirstytoji aptarnavimo perkrovos ataka (angl. *DDoS*), žinutės perėmimo ar neautorizuotos prieigos.

Kalbant apie saugumą reikia atsižvelgti ir į greitai besivystančias kibernetinių atakų grėsmes ir kaip aptarti protokolai yra pasirengę atremti ateityje laukiančius naujus iššūkius. Pavyzdžiui, MQTT ir CoAP susiduria su viename tinkle didėjančiu įrenginių skaičiumi ir sudėtingais atakų modeliais, tokiais kaip botnetas. Algoritmas HTTP yra labai plačiai naudojamas, todėl dažnai gauna atnaujinimus. Likusieji protokolai XMPP ir AMQP, naudojami komunikacijoms bei duomenų mainams, vis dažniau pritaikomi prie debesų kompiuterijos aplinkos, kur atsiranda naujų grėsmių, tokių kaip neteisėta prieiga prie debesų resursų. Todėl norint užtikrinti šių protokolų efektyvumą ateityje, reikia integruoti vis atsinaujinančias autentifikavimo, šifravimo ir anomalijų aptikimo technologijas.

Esančioje **1.1 lentelėje** yra palygini aptarti protokolai pagal jų privalumus, trūkumus, saugumą, energijos resursų sąnaudas, vėlavimo laiką ir antraštės dydį. Toks palyginimas leidžia aiškiau įvertinti, kokiose situacijose kiekvienas protokolas yra tinkamiausias pagal pasirinktus kriterijus.

1.1 lentelė. Protokolų palyginimas [15,18,23,24]

Protokolas	**Privalumas**	**Trūkumas**	**Saugumas**	**Energijos resursų sąnaudos**	**Vėlinimo laikas**	**antraštės dydis (baitais)**
HTTP	Plečiamumas, paprastas, universalus ir patikimas.	Netinka mažiems įrenginiams; padidėja ryšio vėlavimas.	Geras, HTTPS	Aukštos	Didelis	8
MQTT	Lengvas įgyvendinimas; labai tinka įrenginiams su ribotais ištekliais.	Ribotas apimties keitimas dėl tarpininko; nėra šifravimo palaikymo.	Prastas, TLS/ SSL	Mažiausios	Mažas	2
CoAP	Tinka naudoti su ribotais ištekliais.	Ribota QoS; mažiau saugus; nenustatytos tinkamos raktų valdymo procedūros.	Vidutinis, DTLS / TLS / SSL	Vidutinės	Vidutinis	4
AMQP	Siūlomos plačios pranešimų funkcijos; gali būti naudojamas ryšiui tarp heterogeninių prietaisų palaikyti.	Netinka ribotų išteklių taikomosioms programoms; automatinis išteklių nustatymas nepalaikomas.	Puikus, TLS / SSL	Aukštos	Mažas	Kintamas
XMPP	Rekomenduojama pagal IEC 61850; jungia įvairių architektūrų serverius.	Netinka ribotuose įrenginiuose dėl XML; serveriai perkraunami dėl pastovaus žinučių siuntimo.	Aukštas, TLS / SSL	Aukštos	Mažas	Kintamas

1.3. Atakų tipai, etapai ir pažeidžiamumai MQTT protokole

MQTT protokolas yra vienas iš populiariausių pasirinkimų DI aplinkoje dėl savo paprastumo, efektyvaus veikimo, mažo pralaidumo tinkluose ir gebėjimo valdyti didelį skaičių įrenginių. Tačiau dėl šių savybių MQTT tampa patraukliu taikiniu įvairių tipų kibernetinėms atakoms. Nepaisant galimų saugumo priemonių, šiame protokole esantys pažeidžiamumai gali būti išnaudojami įvykdant tiek tradicines, tiek specifines atakas. Todėl šiame poskyryje yra nagrinėjami atakų vykdytojų tipai, dažniausi MQTT protokolo atakų tipai, etapai, priežastys, pažeidžiamumai ir jų poveikis sistemos saugumui.

Visas kibernetines atakas vykdo tam tikri asmenys, kuriuos galima skirstyti pagal vykdomų atakų pagrindą:

- Kenkėjiškas vidaus naudotojas – toks vartotojas, kuris jau priklauso MQTT tinklui ir valdo tam tikrą mazgą (įrenginį);
- Smalsus naudotojas – naudotojas, kuris dėl savo smalsumo ir įdomumo ieško spragų protokole;
- Blogas gamintojas – gamintojas, kuris be leidimo gali prisijungti prie savo gaminio ir įrašyti kenksmingą programą;
- Išorinis užpuolikas – naudotojas, kuris iš išorės bando sustabdyti protokolo tinklų paslaugas, gauti piniginę ar kitokią naudą [25].

Daiktų interneto tinkle yra daug atakų tipų kiekviename lygmenyje, tačiau MQTT protokolas yra taikomojo lygio protokolas, todėl toliau analizuojami pagrindinei šio protokolo atakų tipai:

- DoS (angl. *Denial of Service*) ir DDoS (angl. *Distributed Denial of Service*) – šių atakų tikslas yra išeikvoti tinklo ar įrenginio išteklius, tokius kaip laikas, pralaidumas ar saugojimo vieta, ir taip trukdyti realiems tinklo naudotojams normaliai naudotis sistema. Turėdami omenyje MQTT protokolą, tyrėjai DoS atakas išskyrė į kelias kategorijas. Viena iš jų yra TCP paremta ataka, kadangi MQTT protokolas veikia ant šio protokolo, šios atakos metu yra siunčiamos prisijungimo užklausos, taip sukuriant daug pusiau atidarytų sesijų. Kita ataka yra paremta apkrovimu, šioje atakoje yra išsiunčiamas paketas, kurio dydis viršija 256 MB, maksimalų dydį protokole. Kita kategorija yra paremta QoS, kadangi, kaip anksčiau minėta, MQTT protokolas turi tris skirtingus lygius, siunčiant antro lygio QoS pranešimą ataka užima daugiausia laiko. Viena iš paskutiniųjų kategorijų yra žinutės paskelbimu pagrįsta ataka, kai yra užšifruojamas žinutės turinys, todėl užtrunka norint iššifruoti. Paskutinė kategorija yra „bus“ žinutės grindžiama ataka, kai klientas staiga nutraukia ryšį ir visiems dalyviams yra išsiunčiamas pranešimas [26].
- Žmogus viduryje (angl. *Man-in-the-middle*) – atakos tipas, kai užpuolikas bando perimti arba pakeisti duomenis, patekdamas tarp dviejų ryšio taškų. Sistemose, kuriose naudojamas MQTT protokolas, MQTT klientai gali būti apgaudinėjami naudojant tarpininko suklastotus pranešimus. Vykstant šiai atakai naudotojas daugeliu atvejų net nesupranta, kad ryšys buvo suklastotas [27]. Šio tipo ataka yra populiari dėl to, kad protokole nėra šifravimo algoritmo, kas reiškia, kad duomenys yra siunčiami grynu tekstu kas labai palengvina kenkėjams darbą.
- Grubios jėgos (angl. *Brute force*) – atakos tipas, kai yra bandoma išgauti vartotojų prisijungimus ir jų prieigą prie sistemos naudojant atsitiktines, dažniausiai naudojamas prisijungimų variacijas. Sėkmingai gavę prieigą prie tarpininkų, kenkėjai gauna galimybę prenumeruoti ir skelbti pranešimus. Ši prieiga leidžia jiems išgauti jautrią informaciją arba manipuliuoti įrenginiais, o tai sukelia pasekmes, panašias į tas, kurios pastebimos nesankcionuotos prieigos atvejais [26].
- Neleistina prieiga – vienas iš vyraujančių protokolo saugumo iššūkių. Nors protokolas ir suteikia įvairių būdų autentifikacijai, tokių kaip slaptažodis ar JWT (angl. *JSON Web Token*), tačiau jie iškart nėra realizuoti, ko daugelis žmonių nepadaro dėl nežinojimo. Ši klaida reiškia, kad jei priešininkai gali pasiekti MQTT serverius, jie gali gauti neteisėtą prieigą, kurią išnaudodami pažeidėjai pasiekia asmeninę informaciją ar prenumeruoja norimą temą [26]. Šios atakos išnaudojimui didelį pavojų kelia protokole esantis funkcionalumas, kuris yra vadinamas pakaitiniais simboliais („#“, „+“). Šie simboliai leidžia įsilaužėliams užsiprenumeruoti nežinomas temas. Simbolis #, esantis eilutės pabaigoje, atitinka įvairius temos lygius, o simbolis + atitinka tik vieną lygį.
- Pakartojimo ataka – ataka, kuri remiasi protokolo duomenų vieneto instrukcijų vykdymu. Šiuo atveju aukos skaitytuvo užklausa šia ataka bus siunčiama atskiram piktavaliui ir kuo greičiau jo atsakymas bus atkartotas atgal [25].

MQTT protokolo atakų etapai dažnai atitinka bendrą kibernetinių atakų eigą, kurią sudaro informacijos rinkimas, pažeidžiamumų nustatymas, išnaudojimas ir poveikio įtvirtinimas. Šie etapai gali būti suskirstyti į šiuos žingsnius, pritaikant juos MQTT protokolo specifikai:

1. Informacijos rinkimas – šio etapo metu siekiama surinkti informaciją apie MQTT tarpininką (anlg. *broker*), jo konfigūraciją ar veikiančius klientus;
2. Pažeidžiamumų nustatymas – ieškoma paliktų spragų konfigūracijoje;
3. Prieigos gavimas – bandoma išanalizuotus pažeidimus išnaudoti, kad įgytų prieigą prie DI įrenginių;

4. Atakos vykdymas – gavus prieigą bandoma sutrukdyti protokolo veikimą, stabdant priėjimą prie informacijos, platinant netikrus pranešimus ar kitokius sistemai kenksmingus veiksmus;
5. Poveikio įtvirtinimas – įvykdžius sėkmingą ataką siekiama įtvirtinti jos padarinius;
6. Atakos paslėpimas – įvykdžius ataką siekiama nuslėpti įvykdytus veiksmus, kad nebūtų atsekama.

Šie etapai leidžia sistemiškai analizuoti atakas prieš MQTT protokolą ir padeda identifikuoti prevencines saugumo priemones kiekviename žingsnyje.

Aptarus pagrindinius protokolo atakų tipus bei jų eigą, tikslinga įvertinti priežastis, dėl kurių šie išpuoliai tampa įmanomi. Remiantis atlikta analize, išskiriamos dvi pagrindines saugumo spragų kilmės grupės: protokolo specifikacijos trūkumai ir nepakankamai saugi protokolo realizacija [18].

Silpna protokolo specifikacija atsiranda todėl, kad pradinė protokolo idėja buvo sukurti protokolą, kuris būtų mažos apimties pranešimų perdavimo protokolas, tinkantis įrenginiams, turintiems minimalų pralaidumą, galią ir atminties galimybes. Protokolas turi mažesnę antraštę (2B) ir užtikrina patikimą, mažo vėlavimo ryšį ir mažesnį energijos suvartojimą, tai labai svarbu realaus laiko daiktų interneto taikomosioms programoms. Kadangi siekta, kad protokolas vartotų kuo mažiau resursų, nebuvo protokole integruota šifravimo mechanizmų ar autorizavimo politikos. Šifravimo mechanizmo nebuvimas reikalauja naudoti SSL/TLS protokolus, kad apsaugotų ryšį. Tačiau šių protokolų naudojimas užtikrina tik apsaugą nuo taško iki taško, todėl kenkėjas gali stebėti arba keisti pranešimus perdavimo metu, jei paskutinis taškas nėra užšifruotas. Be to, jis nepriklauso nuo taikomosios programos lygmens naudingosios apkrovos šifravimo ir negali užtikrinti galutinio saugumo. Pavyzdžiui, jei TLS ryšio užmezgimo metu tarpininko autentiškumas nėra tikrinamas, užpuolikas gali apsimesti tarpininku klientams, gauti slaptus duomenis, pavyzdžiui, kliento prisijungimo duomenis, vėliau juos panaudoti imituodamas klientą tarpininkui ir gali sukelti įvairių saugumo ar privatumo pažeidimų. Kaip ir šifravimo mechanizmas taip ir autorizavimo politiką reikia realizuoti atskirai. Didžioji dalis DI tinklo įrenginių gamintojų naudoja paprastą autentifikaciją per vartotojo vardą ir slaptažodį, tačiau, kaip anksčiau buvo aptarta, šis būdas yra lengvai apeinamas. Naudodami šias spragas kenkėjai gali pasiekti norimą tikslą [18].

Kitas protokolo specifikacijos pažeidžiamumas yra netinkamas paketo ilgio patikrinimas. Pats paketas yra pavaizduotas **4 pav.**, kurį sudaro valdymo antraštė, paketo ilgis, kintamos antraštės ir krūvis. Paketo laukų išskyrimo ir identifikavimo procesas vadinamas paketų analizavimu. Nors iš pirmo žvilgsnio analizė atrodo paprasta, dėl pažangių paketų konstravimo metodų užpuolikai gali lengvai išnaudoti protokolo pažeidžiamumus, sukeldami netinkamą analizę, dėl kurios gali kilti tokios atakos kaip buferio perpildymas, paslaugos atsisakymo ar atminties turinio gavimas. Netinkamas ilgio tikrinimas yra aptiktas įvairiuose MQTT įgyvendinimuose ir užfiksuotas CVE (angl. *Common Vulnerabilities and Exposures*) duomenų bazėje. Pavyzdžiui, netikrinamas „*rem_len*“ dydžio, tai leidžia vykdyti neteisingą paketo apdorojimą. Nepakankamas tikrinimas ilgio lauko paskelbtose žinutėse leidžia vykdyti nuotolinį kodą ir sukelti buferio perpildymą. Dėl netinkamo MQTT temų ilgio ir turinio tikrinimo buvo galima perrašyti pagrindinį steko rodiklį, sukeliant DoS ataką. Taip pat MQTT brokeris gali sugesti dėl netinkamo paketo ilgio tikrinimo, kai užpuolikas pateikia neteisingo ilgio paketą [28].

Dar vienas iš pažeidžiamumų yra nepakankamas privalomų laukų tikrinimas. Šie pažeidžiamumai atsiranda dėl privalomų laukų validavimo ignoravimo protokolo įgyvendinimo metu. MQTT

protokole paketo ilgis ir laukai priklauso nuo paketo tipo, todėl būtina aiškiai įgyvendinti privalomų laukų tikrinimą pagal konkretų paketo tipą. Pavyzdžiui, jei paketas turi naudotojo vardo lauką, tada privalo būti ir susijęs slaptažodžio laukas, nes jo nebuvimas gali kelti grėsmę įgyvendinimo saugumui [28].

Loginės klaidos tikrinimo trūkumas MQTT paketo duomenyse ir netinkamas jų apdorojimas yra viena iš dažniausių pažeidžiamumų priežasčių. Šie pažeidžiamumai pasireiškia dėl to, kad realizacijose nėra tinkamai patikrinamos tam tikros protokolo specifikacijos. Dėl to užpuolikai gali išnaudoti šias spragas, siųsdami suformuotus paketus, kurie sukelia įvairias atakas [28].

Silpna protokolo realizacija atsiranda, kai tarpininkas ar naudotojai konfigūruodami MQTT protokolą nesilaiko nustatytų saugumo aspektų ar naudojamojoje bibliotekoje yra blogai realizuotas protokolo veikimas [18]. Dėl šių priežasčių užpuolikai gali pasiekti ar atrasti kitus tinklo pažeidžiamumus taip padarydami dar didesnę žalą naudotojams.

Viena iš tokių atakų yra SlowITe (angl. *The Slow DoS against Internet of Things Environment*) – paslaugos trikdymo ataka, kuri išnaudoja anksčiau aptartus protokolo pažeidžiamumus. Ši ataka yra priskiriama prie lėtos DoS atakų kategorijos, kai naudojamas minimalus atakos pralaidumas ir ištekliai, siekiant nukreipti ataką į tinklo paslaugą, vykdančią atsisakymą teikti paslaugas. SlowITe atakos tikslas yra sukurti daugybę ryšių su MQTT brokeriu, užblokuojant visas prieinamas jungtis, kurias brokeris gali aptarnauti vienu metu. Ataka vykdoma siunčiant prisijungti (angl. *CONNECT*) paketus, kurie inicijuoja ryšį su brokeriu. Šių paketų dydis yra minimalus, todėl apkrova serverio resursams išlieka maža. Po ryšio užmezgimo ataka naudojasi MQTT protokolo galimybe nustatyti „palaikyti gyvą“ (angl. *Keep-Alive*) parametro reikšmę, kuri kontroliuoja, kiek laiko serveris išlaikys ryšį, atidarytą be papildomo duomenų srauto. Užpuolikas gali manipuliuoti šiuo parametru, nustatydamas maksimalų laikotarpį, pavyzdžiui, 65 535 sekundžių, kas prilygsta daugiau nei 27 valandoms. Kadangi brokeris gali valdyti ribotą jungčių skaičių (pavyzdžiui, „*Eclipse Mosquitto*“ pagal nutylėjimą palaiko 1024 jungtis), užpuolikas gali efektyviai blokuoti teisėtų klientų prieigą, nepastebimai išnaudodamas visas jungtis. Šis mažos spartos metodas apsunkina atakos aptikimą, nes serverio procesorius, atmintis ir tinklo pralaidumas beveik neapkraunami. SlowITe pabrėžia MQTT protokolo silpnumus, tokius kaip galimybė manipuliuoti „palaikyti gyvu“ parametru, ir pateikia naują grėsmę DI infrastruktūroms [29].

1.4. DI tinklų apsaugos nuo atakų metodai ir realizacijos MQTT protokole

Šiame skyriuje aptarsime daiktų interneto tinklų apsaugos nuo atakų būdus MQTT protokole, jau realizuotus budus, atakų atpažinimo metodus bei realizacijas. Pagrindinis būdas apsisaugoti nuo atakų yra tinkamas supratimas, kaip protokolas veikia, visas išanalizavimas ir realizavimas įprastų saugumo būdų, dėl kurių nežinojimo dažniausiai ir nukenčiama nuo nusikaltėlių atakų. Kitos technologijos, kaip mašininis mokymasis, blokų grandinė ar dirbtinis intelektas yra perspektyvūs sprendimai ir patenka su kai kuriomis kitomis naujomis technologijomis kaip gelbėjimosi priemonė sprendžiant daiktų interneto saugumo problemą [25].

MQTT protokolas buvo kuriamas siekiant užtikrinti lengvą ir efektyvų veikimą, todėl pradinis dėmesys saugumui nuo kibernetinių atakų nebuvo didelis. Vis dėlto protokole numatytos minimalios apsaugos priemonės, tokios kaip tapatybės patikrinimas, autentifikavimas ir autorizavimas. Šiuolaikinių atakų kontekste šios priemonės dažnai yra lengvai apeinamos, todėl būtina ieškoti papildomų sprendimų.

Mašininio mokymosi technologijos gali būti taikomos dviem pagrindiniais būdais – prižiūrimu (angl. *supervised*) arba neprižiūrimu (angl. *unsupervised*) mokymu. Kadangi DI įrenginiai generuoja daug duomenų, reikia prieš skaičiavimus patikrinti ar nėra dublikatų [25]. Vienas iš pagrindinių mašininio mokymosi privalumų yra gebėjimas apdoroti didelį duomenų kiekį realiuoju laiku, kas DI aplinkoje yra labai svarbu. Įrenginiai naudodami mokymosi modelius gali aptikti DoS atakas, identifikuoti neįprastus duomenų perdavimo modelius ar blokuoti neleistiną veiklą dar prieš jai patenkant į sistemą.

Blokų grandinės technologija yra laikoma inovatyviu sprendimu, galinčiu padidinti DI tinklų saugumą dėl savo paskirstytos ir nekintamos struktūros. Naudojant blokų grandinę, visi tinklo įrenginiai gali dalyvauti saugiai patvirtinant duomenų mainus, taip išvengiant centrinio valdymo taškų pažeidžiamumų. Vienas pagrindinių blokų grandinės privalumų yra galimybė užtikrinti duomenų vientisumą. Kiekvienas pranešimas ar operacija MQTT tinkle gali būti įrašyta į bloką, kuris yra kriptografiškai apsaugotas nuo pakeitimų. Tokia sistema leidžia aptikti ir sustabdyti neteisėtus pakeitimus ar įrenginių bandymus platinti netikrus duomenis.

Viena iš realizacijų yra skirta užtikrinti protokolo autentifikaciją. Šiame metode yra siūloma atsisakyti MQTT protokolo siūlomo autentifikacijos metodo ir naudoti vienkartinio slaptažodžio (OTP) autentifikaciją, naudojant *Ethereum* blokų grandinę. Sprendimas orientuotas į lengvą, energetiškai efektyvų DI įrenginių autentifikavimą be TLS naudojimo transporto lygyje. Šio metodo pagrindas – išmanioji sutartis (angl. *smart contract*), kuri yra atsakinga už autentifikavimo proceso valdymą ir saugų duomenų saugojimą *Ethereum* blokų grandinėje. Pats metodo veikimas sudarytas iš kelių etapų, kaip registracijos kai DI įrenginys registruojamas MQTT brokeriui, kuris susieja vartotojo tapatybę su *Ethereum* adresu. Toliau brokeris paskelbia išmaniąją sutartį *Ethereum* blokų grandinėje, kurioje saugomi įrenginio duomenys, tokie kaip *hash* funkcija apdorotas IPv6 adresas, savininko *Ethereum* adresas, atsitiktinis skaičius ir OTP galiojimo laikas. Įvykus registracijai reikia pereiti prie autentifikacijos, kai DI įrenginys inicijuoja autentifikaciją, brokeris sugeneruoja atsitiktinį OTP. Tuomet vartotojas arba įrenginys perduoda OTP *Ethereum* blokų grandinei, kur išmanioji sutartis patikrina jo galiojimą, atitikimą ir laiką. Jeigu viskas tinka ir pavyksta, brokeris gauna patvirtinimą iš blokų grandinės ir užtikrina, kad autentifikavimo procedūra atlikta teisingai. Tokio metodo naudojimas užtikrina saugumą nuo pasikartotinų prisijungimo duomenų atakų, adresatų privatumo, duomenų vientisumo. Tačiau norint realizuoti tokį metodą neužtenka įprast žinių, reikia būti gerai susipažinus tiek su MQTT protokolu, tiek su *Ethereum* technologija. Kadangi yra naudojama *Ethereum* technologija, duomenų mainai priklauso nuo jo greičio ir stabilumo [30].

Kita realizacija, skirta apsaugoti MQTT tarpininką nuo atakų buvo naudojant laiko pagrindu generuojamą vienkartinį slaptažodį (TOTP) ir šifruoti duomenis naudojant AES (angl. *Advanced Encryption standard*). Pirmas etapas naudoja TOTP sugeneruotą slaptažodį, kuris išsiunčiamas naudotojui į el. paštą. Sėkmingai patvirtinus naudotojo tapatybę, pereinama prie antrojo etapo, kuriame autentifikacija atliekama pagal kliento ID, slaptažodį, prievado numerį ir IP adresą. Šio proceso metu papildomai naudojama juodųjų ir baltųjų sąrašų sistema, kuri leidžia valdyti naudotojų prieigą prie brokerio [31]. Toks metodas padeda užtikrinti duomenų konfidencialumą, vientisumą ir sumažina neteisėto prisijungimo riziką.

Tačiau vienas iš pagrindinių žingsnių prieš bandant apsisaugoti nuo kibernetinių atakų yra gebėjimas jas atpažinti naudojant įvairius metodus, kas yra didelis iššūkis dėl atakų užklausų mažo skirtumo nuo realių užklausų. Tolimesniame sąraše yra aprašomi galimi atakų atpažinimo būdai:

- Mašininiu mokymusi pagrįsti aptikimo mechanizmai. Tokio tipo mechanizmas naudojamas tuomet, kai norima aptikti anomalijas užklausose, tokias kaip padidėjęs duomenų srautas, duomenų kiekio pasikeitimas, prisijungimo trukmė ar daug kitų atsitinkančių anomalijų. Šis mechanizmas gali būti skirtas srauto analizei, sesijos analizei, anomalijomis paremtoms IDS. Nors kai kurie darbai siūlo mašininio mokymosi metodus, jų veikimo analizė yra ribota. Dauguma tyrimų sutelkia dėmesį tik į vienos rūšies ataką arba testuoja metodus su ribotais duomenų rinkiniais. IDS veiksmingumas skirtingose tinklo konfigūracijose ir DI įrenginiuose yra menkai ištirtas. Dažniausiai naudojami atraminių vektorių klasifikatorius (SVM), atsitiktinis miškas (RF), sprendimų medis (DT), naivusis Bajesas (NB), o giliojo mokymosi (DL) metoduose dominuoja ilgalaikė trumpalaikė atmintis (LSTM) ir konvoliucinis neuroninis tinklas (CNN). Didžioji dalis darbų remiasi iš anksto paruoštais duomenų rinkiniais, o realių tęstinių aplinkų naudojimas yra labai retas. Be to, vertinant ML metodus, dažniausiai analizuojama tik tikslumo metrika, o kiti rodikliai yra nagrinėjami rečiau [32].
- Medžiu pagrįsti aptikimo mechanizmai yra pažangus metodas, naudojamas identifikuoti anomalijas MQTT protokole, analizuojant įvykių žurnalus ir stebint duomenų srautus. Šis metodas remiasi hierarchine medžio struktūra, kuri leidžia modeliuoti MQTT tinklo veiklą ir aptikti galimus nukrypimus nuo įprastų veikimo modelių. Kadangi MQTT įvykių žurnalai apima prisijungimo (angl. *CONNECT*), publikavimo (angl. *PUBLISH*), prenumeratos (angl. *SUBSCRIBE*) ir kitus pranešimus su įvairiais parametrais, pavyzdžiui, *QoS*, *retain* ir *will*, šių duomenų analizė suteikia galimybę atpažinti įtartinas veiklas. Tačiau vien tik įvykių eiliškumas ne visada yra pakankamas norint tiksliai nustatyti anomalijas, todėl papildomai taikoma procesų kasyba (angl. *process mining*), kuri leidžia aptikti paslėptas duomenų sąsajas ir struktūras. Procesų kasybos metu sugeneruojami procesų medžiai, kurie atspindi įprastą MQTT veiklą, o iš jų išskiriami submedžiai, leidžiantys atskirti normalias ir įtartinas veiklas. Palyginus šiuos submedžius, galima nustatyti veiksmų sekas, kurios reikšmingai nukrypsta nuo standartinių procedūrų, ir identifikuoti galimas atakas.
- Aptikimas pagal požymius yra metodas, kai atakos nustatomos remiantis iš anksto žinomais jų požymiais ar modeliais. Šis metodas dažnai naudojamas įsilaužimų aptikimo sistemose (angl. *Intrusion Detection System*), kurios analizuoja tinklo srautą ir lygina jį su žinomų grėsmių duomenų baze. Šis metodas yra veiksmingas aptinkant žinomas atakas, tokias kaip DoS, MITM ar žinomų pažeidžiamumų išnaudojimą, tačiau jo trūkumas yra nesugebėjimas identifikuoti naujų, anksčiau neaptiktų grėsmių.
- Taisyklėmis pagrįstas aptikimo metodas, naudojamas tuomet, kai yra iš anksto aprašytos saugumo taisyklės, kad apsisaugotų nuo tokių atakų, kaip dubliuotų klientų, įtartinų prisijungimų ar netinkamų pranešimų. Šiame mechanizme įprasta naudoti paketų analizę taip išanalizuojant, ar nėra pažeidimų pagal nustatytas taisykles ar dar vienas metodas paremtas *fuzzy* taisyklių logika [15].

Žiūrint į **1.2 lentelę** galima matyti išvardintų aptikimo metodų palyginimą pagal jų privalumus ir trūkumus, kurie išryškina metodų stipriąsias ir silpnąsias vietas.

1.2 lentelė. Aptikimo metodų palyginimas

Metodas	**Privalumai**	**Trūkumai**
Mašininiu mokymusi pagrįsti aptikimo mechanizmai	Aukštas tikslumas aptinkant nežinomas atakas, pritaikomas aptinkant naujas grėsmes	Reikalingi dideli duomenų rinkiniai apmokyti modelius, mažas kiekis tęstinių aplinkų

Medžiu pagrįsti aptikimo mechanizmai	Efektyvus atpažįstant nukrypimus nuo įprastų veikimo modelių, gali aptikti paslėptas sąsajas	Polinkis į pertreniruotumą (angl. *overfitting*), gali būti mažiau tikslūs su labai dinamiškais duomenimis
Aptikimo pagal požymius mechanizmai	Efektyvus prieš žinomas atakas (DoS, MITM), greitas aptikimas	Negali aptikti naujų, nežinomų atakų, priklausomas nuo nuolat atnaujinamų duomenų bazių
Taisyklėmis pagrįsti aptikimo mechanizmai	Greitas veikimas, tinkamas mažų resursų įrenginiams, leidžia aptikti aiškiai apibrėžtas grėsmes	Ribotas prieš naujas ir sudėtingas atakas, reikalauja nuolatinio taisyklių atnaujinimo

Aptikti atakas MQTT pagrindu veikiančiuose DI tinkluose yra sudėtinga, nes jos gali paveikti tiek brokerius, tiek klientus. Dauguma tyrimų daugiausia dėmesio skiria DoS atakoms, nors MITM, tapatybės klastojimas ir duomenų nutekėjimas kelia didelę grėsmę. Be to, dauguma mechanizmų reikalauja didelių skaičiavimo resursų ir orientuojasi tik į MQTT tarpininkus. Reikalingi paprastesni aptikimo mechanizmai, skirti ir DI įrenginiams. Taip pat būtina įvertinti aptikimo sistemų veikimą įvairiuose DI tinkluose, kad būtų užtikrintas jų efektyvumas realiuoju laiku [15].

Viena iš realizacijų, kuri įgyvendina papildomą MQTT protokolo apsaugą, yra „*MISSION*“ (angl. *Mining Semantics of IoT Networks*). Šio metodo tikslas – analizuoti ir suprasti DI procesus taip, kad jie būtų paaiškinami ir naudingi tinklo įsilaužimų aptikimo sistemoms (angl. *Network Intrusion Detection Systems*, NIDS), nepakeičiant paties MQTT standarto. Tai autorių atžvilgiu yra sudėtinga duomenų analizės problema. Siekdami užtikrinti nuoseklumą, skaidrumą ir atsekamumą, autoriai taiko CRISP-DM (angl. *Cross Industry Standard Process for Data Mining*) metodiką, kuri yra suderinta su problemos sudėtingumu ir ypač gerai tinka projektuojant daug duomenų reikalaujančias sistemas informacinių sistemų srityje. Vienas iš pagrindinių metodo privalumų yra gebėjimas suteikti aukštą suprantamumą ir tikslumą, nes procesų modeliai leidžia ne tik aptikti anomalijas, bet ir suprasti jų kilmę. Be to, šis metodas yra pritaikytas veikti realiuoju laiku, todėl leidžia greitai reaguoti į galimas grėsmes. Tačiau pagrindinis trūkumas yra didelis duomenų apdorojimo sudėtingumas, kuris gali riboti metodo efektyvumą labai didelio masto tinkluose [33].

Kitame darbe aprašomas metodas, skirtas DoS atakų aptikimui hibridiniuose DI tinkluose, kuriuose naudojami MQTT ir CoAP protokolai. Pasiūlytas metodas remiasi mašininio mokymosi algoritmų, tokių kaip SVM (angl. *Support Vector Machine*), naivusis Bajesas (NB), sprendimų medis (DT) ir KNN (angl. *K-Nearest Neighbor*), integravimu į SDN (angl. *Software-Defined Networking*) architektūrą. Šis metodas leidžia efektyviai aptikti DoS atakas, pasitelkiant duomenų klasifikaciją ir tinklo stebėjimą realiuoju laiku. Metodas naudoja DoS atakų duomenų rinkinį treniruoti mašininio mokymosi modelius, kurie vėliau taikomi SDN kontroleryje stebėti tinklo srautą. Klasifikatoriai analizuoja tokias metrikas, kaip duomenų perdavimo greitis, tinklo užlaikymas ir aktyvių įrenginių skaičius, siekdami aptikti anomalijas. SDN kontroleris priima sprendimus, pagrįstus klasifikatorių pateiktais rezultatais, ir gali dinaminiu būdu perskirstyti tinklo išteklius arba blokuoti įtartinas užklausas. Toks metodas gali aptikti DoS atakas pasiekdamas 84 % tikslumą, 92 % jautrumą ir 94 % specifiškumą. Tačiau toks metodas turi ir trūkumų, kaip papildomas tinklo užlaikymas dėl analizuojamų duomenų ir reikalingas nuolatinis modelio atnaujinimas, kad būtų atsparus naujoms atakoms.[34]

Dar vienoje iš realizacijų pasirinktas šiek tiek kitoks priėjimas sprendžiant kibernetinių atakų atpažinimo problemą pasinaudojant sudėtingų įvykių apdorojimo (angl. *Complex Event Processing*, CEP) programinę įrangą, skirtą aptikti dominuojančias situacijas analizuojant ir koreliuojant

didžiulius duomenų kiekius. Tačiau CEP variklis negali aptikti tam tikro elgesio, kurio anksčiau nemodeliavo ekspertas. Norint išspręsti šią problemą buvo siūloma realizuoti mašininio mokymosi algoritmą, kuriame naudojama pagrindinių komponenčių analizė (angl. *Principal component analysis*, PCA). Norint tai įgyvendinti buvo atlikti reikalingi žingsniai. Pirmiausia buvo apdoroti ir suskirstyti į kategorijas gauti duomenys, kad būtų galima apmokyti ML. Antrajame siūlomos sistemos etape pateikiamas naujas metodas, kaip atskirti elementus skirtingose kategorijose, nustatant klasifikavimo slenksčius, pagrįstus statistiniais rodikliais. Jis apima kiekvieno komponento vidurkio, standartinio nuokrypio ir dispersijos dalies apskaičiavimą, kad būtų sudaryta lygtis, pagal kurią nustatoma priklausomybė kategorijai, o tikslumui patikslinti pasirinktinai naudojamas korekcinis elementas. Paskutiniame etape klasifikavimo lygtis paverčiama CEP modeliu pasirenkant EPL. Tuomet modelis apibrėžiamas išverčiant lygtį į EPL sintaksę, įtraukiant pagrindinius parametrus. Taigi šis algoritmas leidžia automatiškai generuoti CEP modelius, sumažinant tinklo pralaidumo naudojimą ir naudojant duomenų rinkinius, kurie neturi būti itin dideli. Be to, šis pasiūlymas gali būti įgyvendinamas naudojant mažo našumo įrenginius [35].

Išanalizavus šiuos ir daugelį kitų protokolo metodų galima matyti, kad didžioji dalis naudojamų atpažinimų metodų yra pagrįsti mašininiu mokymusi arba įsiskverbimo aptikimo sistemomis (IDS). Šiame darbe kuriamas atakų aptikimo metodas daiktų interneto tinkluose, naudojančiuose MQTT 5.0 protokolą, būtų orientuotas į lėtas DoS (angl. *Denial of Service*) atakas. Lėtos DoS atakos skiriasi nuo tradicinių staigių DoS ar DDoS tuo, kad pažeidėjas nenaudoja didelio staigaus srauto, vietoj to, jis planuotai ir lėtai išnaudoja ir užima tarpininko jungtis, kurios imituoja tikrų MQTT klientų veikimą. Pavyzdžiui, pažeidėjas gali atidaryti daug ilgų MQTT jungčių, siųsti retus *PING* paketus ar labai mažus pranešimus, nepabaigti užklausų arba manipuliuoti *keep-alive/clean session* nustatymais taip, kad tarpininko jungtys būtų užimtos ir kiekviena atskira sesija atrodytų legali ir nekelianti įtarimo.

Šio tipo atakas svarbu aptikti, nes pasekmės yra ne tik laikinas sistemos nepasiekiamumas tikriems MQTT klientams, bet ir sistemos klaidingas veikimas ar visiškas sustojimas dėl negautų ar netinkamų duomenų. Todėl realizuojamo metodo idėja yra ne tik stebėti tinklo srautą MQTT protokole, bet ir analizuoti visas tarpininko jungtis. Analizuojant aktyvių jungčių skaičių, jų trukmę, klientų ID ir IP neatitikimus, dažnai pasikartojančius ne iki galo užbaigtus sujungimus, mažų ar retų laiko tarpsnių pranešimų ir QoS nustatymų netipinius modelius. Tokiu būdu galima atpažinti elgsenos anomalijas, kurios nėra matomos tradiciniu srauto metu.

1.5. Išvados ir tolimesni darbo uždaviniai

1. Išanalizavus daiktų internetą, nustatyta, kad didėjantys tinklai ir mažėjantys įrenginiai reikalauja atkreipti vis didesnį dėmesį į saugumą;
2. Atlikus DI naudojamų protokolų analizę ir palyginimą, nustatyta, kad kiekvienas protokolas turi tiek minusų, tiek pliusų, kurie priklauso nuo planuojamo panaudojimo;
3. Analizės metu buvo atliktas atakų prieš MQTT protokolą analizė, bei nustatyta, kad pagrindiniai atakų tipai yra paslaugos trikdymo, žmogus viduryje, grubios jėgos, neleistinos prieigos ir pakartojimo atakos;
4. Išanalizavus MQTT protokolo pažeidžiamumus, nustatyta, kad protokole trūksta saugumo aspektų, klaidų tikrinimo ar antraštės ilgio tikrinimo;
5. Analizuojant egzistuojančių sprendimų metu nustatyta, kad dauguma iš jų naudoja IDS ir mašininį mokymąsi aptikti atakas.
6. Išanalizavus atakų pavyzdžius buvo sugalvotas metodas, turintis padidinti atakų atpažinimą.

Tolimesni darbo uždaviniai yra:

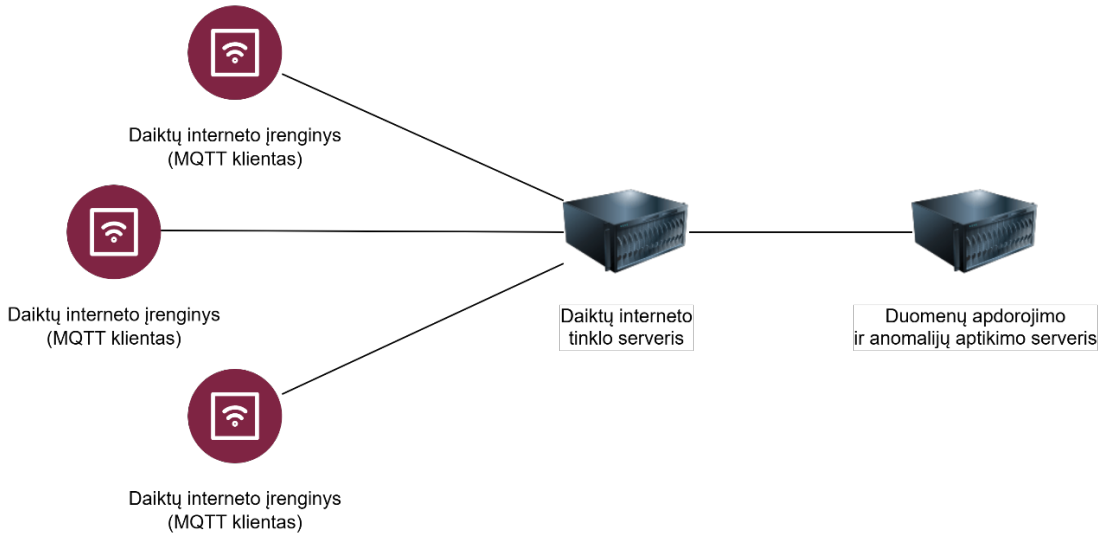
1. sukurti atakų aptikimo metodą, kuris aptiks atakas, panašias į SlowITe ataką;
2. realizuoti atakų aptikimo prototipą pagal sukurtą metodą;
3. atlikti sukurto prototipo eksperimentinius tyrimus ir palyginti gautus rezultatus su kitais atakų atpažinimo metodais.

2. Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodo projektas

2.1. Darbo tikslas

Projektinės dalies tikslas – suprojektuoti metodą, kuris aptiktų lėtas DOS atakas (angl. *slow DoS attack*), kuris sukelia tinklo paslaugos neprieinamumą, perimant visus sujungimus, su kuriais tarpininkas vienu metu gali komunikuoti. Šis metodas bus skirtas MQTT 5.0 protokolo tarpininko (angl. *broker*) apsaugos užtikrinimui nuo tokio tipo atakų stebint visas siekiamas ir esamas jungtis ir jų srautą. Turint šią informaciją apie paketus bus analizuojamos visos jungtys bandant aptikti įtartinas jungtis. Lėtos DoS atakos vyksta ne kaip įprastos atakos staigiai vienu metu, tačiau trunka per ilgą laiką, kuris būna labai panašus į įprasto kliento veikimą. Taip sujungę daug atakuojančių MQTT klientų gauname lėtos DoS atakos pavyzdį.

2.2. Architektūra

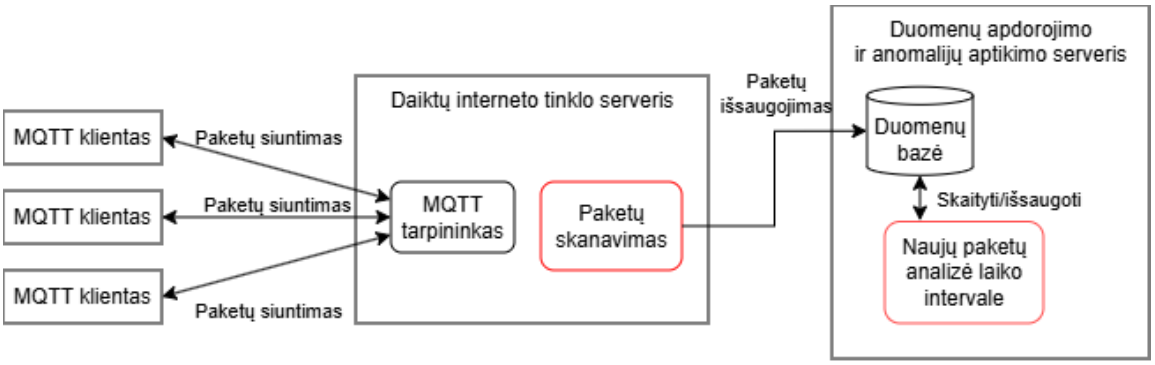


10 pav. Sistemos įrenginių išsamus infrastruktūros sudėtis

Sistemos įrenginių išsamus infrastruktūros vaizdas (žr. **10 pav.**) susideda iš šių atributų:

- **Daiktų interneto įrenginys (MQTT klientas):** šį elementą sudarys įvairūs daiktų interneto įrenginiai, palaikantys MQTT 5.0 protokolą. Iš šių įrenginių bus simuliuojami realių klientų paketai ir lėtojo DoS (angl. *Low-rate DoS*) atakos paketai, kurias mano siūlomas metodas stengsis atpažinti ir išskirti nuo įprastų jungčių. Aptikus ataką, metodas automatiškai užblokuos įtartiną jungtį, taip atlaisvindamas MQTT tarpininko išteklius.
- **Daiktų interneto tinklo serveris:** šiam metodo elementui reikės įrenginio, kurį sudarys du pagrindiniai moduliai – MQTT protokolo tarpininkas (angl. *broker*), kuris privalomas palaikyti komunikaciją su klientais, ir paketų skanavimo modulis. Paketų skanavimo modulis skanuos MQTT protokolo prievado (angl. *port*) priimamus ir siunčiamus paketus *raw* formatu, kuriuos konvertavęs perduos į **duomenų apdorojimo bei anomalijų aptikimo serverį.**
- **Duomenų apdorojimo ir anomalijų aptikimo serveris:** šis įrenginys bus atsakingas už gautų paketų saugojimą duomenų bazėje bei laiko serijos analizę ir anomalijų aptikimą naudojant mašininį mokymąsi iš naujų paketų. Aptikus kenksmingus klientus juos išsaugoti duomenų bazėje ir išspausdinti pranešimų faile.

2.3. Architektūrinis sistemos veikimas

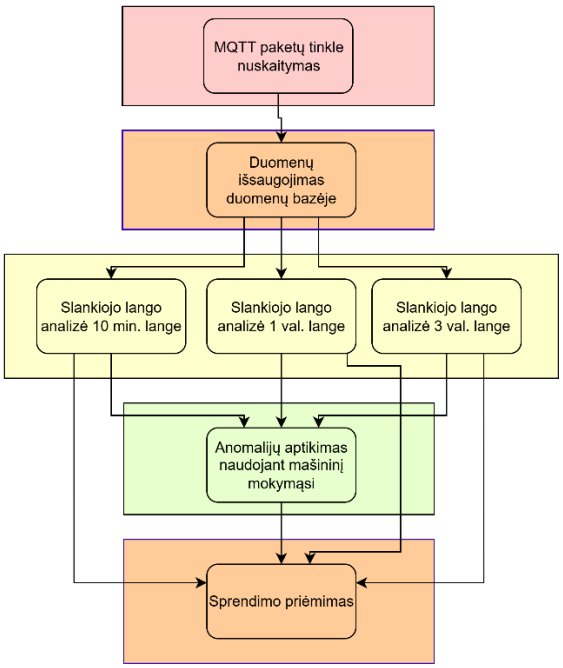


11 pav. Sistemos modulių architektūrinis modelis

Žiūrint iš **11 pav.** grafiko architektūrinis Lėtų DoS atakų atpažinimo MQTT protokole metodą sudarys:

1. MQTT klientas ir MQTT tarpininkas siunčia vienas kitam valdymo paketus, kurie yra išvardyti **2.1 lentelėje**;
2. Paketų skanavimo modulis realiu metu skanuoja MQTT protokolo prievadus;
3. Gautus paketus išfiltruoja pagal pasirinktus paketus ir konvertuoja iš *raw* formato į *json*;
4. Kiekvieno paketo nustatyta informacija yra išsaugojama duomenų bazėje tolimesnei analizei;
5. Kas nustatytą intervalą nuskaitomi nauji paketai atlikti laiko analizę pagal nustatytus intervalus ir atributus;
6. Atlikus atributų skaičiavimą yra pritaikomos taisyklės gauti svorius;
7. Tuomet naudojamas mašininio mokymosi algoritmas aptikti anomalijas;
8. Gautus rezultatus išsaugojame duomenų bazėje tolesniems palyginimams;

Žiūrint į **11 pav.** matyti visos sistemos apibendrintas veikimas su MQTT klientais, tačiau norint atsižvelgti tik į realizuojamą metodą **12 pav.** diagramoje, pavaizduota tikslesni metodo atsakingo už kenksmingų MQTT klientų aptikimą žingsniai.



12 pav. Metodo veikimo principas

Realizuojamame metode pagrindinis dėmesys bus skirtas į prisijungimo (angl. *Connect*) paketus, kurie yra išvardinti **2.1 lentelėje**. Šie paketai yra atsakingi už įrenginių sujungimą, atjungimą ar jungties palaikymą. Lėto tipo DOS atakos pagrinde ir išnaudoja šiuos paketus, kurių sudėtis priklausomai nuo paketo tipo skiriasi.

2.1 lentelė. MQTT protokolo paketo valdymo paketai

Connect	***Publish***	***Subscribe***
CONNECT	*PUBLISH*	*SUBSCRIBE*
CONNACK	*PUBACK*	*SUBACK*
DISCONNECT	*PUBREC*	*UNSUBSCRIBE*
AUTH	*PUBREL*	*UNSUBACK*
PINGREQ	*PUBCOMP*	
PINGRESP		

2.4. Paketų skanavimo veikimas

Realizuojamo metodo paketų skanavimo modulis yra atsakingas už MQTT paketų analizę. Siekiant užtikrinti lankstumą, jis bus sukurtas kaip atskiras komponentas, nepriklausomas nuo konkretaus MQTT tarpininko. Tai yra daroma todėl, kad daugelis tarpininkų pateikia paketus žmogui suprantamu formatu, paslėpdami dalį techninės informacijos, kuri yra būtina kibernetinių atakų analizei. Be to, skirtingų tarpininkų konfigūracijos bei funkcionalumo skirtumai tarp nemokamų ir mokamų versijų apsunkina vieningo sprendimo pritaikymą. Sukūrus nepriklausomą modulį, užtikrinamas suderinamumas su įvairiais MQTT tarpininkais ir išvengiama priklausomybės nuo konkrečios platformos.

Nors realizuojamas metodas priims visus ir išsiunčiamus, ir gaunamus paketus tolimesnei analizei, jų visų nereikia, todėl bus apdorojami tik pasirinkti paketai iš praeitame skyriuje esančios **2.1 lentelės**.

MQTT protokolo paketai susideda iš trijų dalių:
1. Fiksuota antraštė (angl. *Fixed Header*) – privaloma visuose MQTT valdymo paketuose;
2. Kintamųjų antraštė (angl. *Variable Header*) – privaloma tik kai kuriuose MQTT valdymo paketuose;
3. Papildomi parametrai (angl. *Payload*) – privaloma tik kai kuriuose MQTT valdymo paketuose.

Turėdami paketo sudėtį, galime nuspręsti, kuriuos paketus naudosime analizei ir kokią informaciją iš šių paketų reikia išsaugoti. Visų naudojamų paketų fiksuota antraštė bus išsaugoma, kad būtų galima nustatyti, koks paketas yra naudojamas ir kaip jį apdoroti. Kintamosios antraštės ir papildomų parametrų išsaugoma informacija priklausys nuo paketo, kad nebūtų saugoma perteklinė informacija taip apsunkinant duomenų agregavimą.

Fiksuotos antraštės sudėtį galima matyti **2.2 lentelėje**.

2.2 lentelė. Fiksuotos antraštės sudėtis

Bitas	7	6	5	4	3	2	1	0
Baitas 1	MQTT valdymo paketo tipas				Kiekvienam MQTT valdymo paketo tipui būdingos žymės			
Baitas 2	Likęs paketo ilgis (Kintamųjų antraštės ir papildomų parametrų)							

Be MQTT paketo atributų taip pat bus saugojami ir pasirinkti tinklo parametrai su kiekvienu paketu:
- siuntėjo (MQTT kliento) IP adresas;
- siuntėjo (MQTT kliento) prievadas;
- gavėjo (MQTT tarpininko) IP adresas;
- gavėjo (MQTT tarpininko) prievadas;
- serverio laikas.

Modulio veikimas paleidžiamas kaip atskira programa, skanuojanti MQTT protokolo įprastus prievadus (1883 ir 8883) bei pagrindinę serveryje esančią tinklo plokštę (eth0). Esant poreikiui, šiuos parametrus galima pakeisti naudojant kitus prievadus ar tinklo sąsajas nurodant paleidus programą, taip užtikrinant sprendimo universalumą ir išvengiant programos kodo keitimo. Paleidus programą, ji veikia kaip atskiras procesas, realiuoju laiku priimantis gaunamus ir siunčiamus paketus raw formatu (nepakeistus žemo lygio duomenis tiesiai iš tinklo srauto) prieš bet kokį tarpinį apdorojimą. Iš viso srauto išfiltruojami tik MQTT protokolo paketai, kurie toliau yra apdorojami išskiriant į MQTT protokolo paketo atskiras dalis. Kadangi fiksuotos antraštės sudėtis yra ta pati visiems paketams, ji dekoduojama visur taip pat, o likusios dalys (kintamoji antraštė ir papildomi parametrai) pagal esamą paketą. Dekodavus visą paketą jis yra išsaugojamas duomenų bazėje, kad galėtų būti apdorojamas kito metodo modulio.

Visuose saugomuose paketų atributuose bus saugoma fiksuota antraštė bei šie atributai priklausomai nuo paketo:
- *CONNECT* – paketas, kuris privalo būti pats pirmas išsiųstas užmezgus tinklo ryšį su serveriu. Šio paketo pagrindiniai atributai, kurie bus saugomi yra:
    - Kintama antraštė:

        - *CONNECT* vėliavėles:
            - *Clean Start* – nurodantis, vyksta nauja sesija ar sena;
            - *Keep Alive* – nurodantis laiko intervalą sekundėmis, kaip dažnai MQTT klientas turi išsiųsti paketą, kad palaikytų jungtį su tarpininku;
            - *User Name Flag* – nurodantis kliento prisijungimo vardo buvimą pakete
            - *Password Flag* – nurodantis kliento slaptažodžio buvimą pakete;
            - *CONNECT* parametrai:
                - *Session Expiry Interval* – nurodantis, kiek laiko tarpininkas turi palaikyti sesiją klientui atsijungus nuo tarpininko.
    - Papildomi parametrai:
        - *ClientID* – skirtas nurodyti kliento ID, nenurodžius tarpininkas sukuria specialų id;
        - *User Name* – jei *User Name Flag* nurodytas, tuomet nurodo vartotojo vardą;
        - *Password* - jei *Password Flag* nurodytas, tuomet nurodo vartotojo slaptažodį;
- *CONNACK* – tarpininko atsakymas į *CONNECT* paketą, patvirtinantis sėkmingą prisijungimą su tarpininku. Šio paketo pagrindiniai atributai kurie bus saugomi, yra:
    - Kintama antraštė:
        - *CONNACK* vėliavėlės:
            - *Session Present* – nurodantis sesijos statusą, ar ji yra nauja ar sena;
            - *Connect Reason Code* – nurodantis priežastį jungties patvirtinimo;
            - *CONNACK* properties:
                - *Session Expiry Interval* – nurodantis serverio nustatytą, kiek laiko tarpininkas turi palaikyti sesiją klientui atsijungus nuo tarpininko
                - *Assigned Client Identifier* – nurodantis tarpininko nustatytą kliento ID
                - *Server Keep Alive* – nurodantis serverio nustatytą laiko intervalą sekundėmis, kaip dažnai klientas turi siųsti paketus;
- *PUBLISH* – kliento siunčiama informacija. Šio paketo pagrindiniai atributai kurie bus saugomi, yra:
    - Kintama antraštė:
        - *Topic Name* – nurodantis temą, kuriai buvo išsiunčiama informacija;
- *SUBSCRIBE* – temą, kurią klientas užprenumeravo, kad gautų informaciją iš nurodytos temos. Šio paketo pagrindiniai atributai, kurie bus saugomi, yra:
    - Papildomi parametrai – nurodantis temų sąrašą, iš kurių klientas nori gauti informaciją;
- *PINGREQ* – skirtas indikuoti tarpininką, kad klientas yra gyvas, kad patikrintų, ar tarpininkas yra prieinamas;
- *PINGRESP* – skirtas atsakyti į *PINGREQ* paketą patvirtinant, kad jungtis su serveriu yra aktyvi;
- *DISCONNECT* – skirtas indikuoti, kad jungtis buvo nutraukta.

2.5. Duomenų bazės veikimas

Duomenų bazės modulis yra vienas iš pagrindinių realizuojamo metodo komponentų, atsakingas už surinktų MQTT paketų, jų metaduomenų ir analizės rezultatų saugojimą. Šio modulio tikslas – užtikrinti efektyvų ir struktūruotą duomenų srautų valdymą, kad vėlesnės analizės (slankiojo laiko ir anomalijų aptikimo) galėtų greitai pasiekti reikiamą informaciją be perteklinio apdorojimo.

Visi duomenys, gauti iš paketų skanavimo modulio (**2.4** skyrius), išsaugomi duomenų bazėje kaip atskiri įrašai. Kiekvienas įrašas atitinka vieną MQTT valdymo paketą, kuriame fiksuojama:

- MQTT protokolo techninė informacija (antraštės duomenys);
- tinklo parametrai (IP adresai, prievadai);
- serverio laiko žyma, pagal kurią vėliau formuojami laiko intervalai.

Be žemo lygmens paketų saugojimo duomenų bazėje kaupiami ir apdoroti duomenys, gauti iš laiko analizės bei anomalijų aptikimo modulių ir aptiktų kenkėjų. Duomenys iš kiekvieno laiko lango intervalo saugomi atskirai, kad būtų galima atlikti analizę ir palyginti kliento elgesį tarp skirtingų laiko langų. Tai leidžia nustatyti ilgalaikius pokyčius ir tendencijas.

Kiekvieno laiko lango įrašą sudarys šie pagrindiniai elementai:

- bendri atributų rezultatai visame laiko intervale;
- lango tipas;
- intervalo pradžios ir pabaigos laikas.

Kiekvieno kliento tame lango įraše sudarys šie pagrindiniai elementai:

- kliento identifikacijos numeris arba ip;
- bendri atributų rezultatai tame laiko intervale;
- taisyklių gautas rezultatas;
- mašininio mokymosi gautas rezultatas;
- galutinis rezultatas;
- galutinis sprendimas („*normal*“, „*attacker*“);
- suaktyvintos taisyklės;
- papildomi parametrai.

Kiekvienos atpažintos atakos įrašą sudarys šie pagrindiniai elementai:

- atakuotojo tipas;
- atakuotojo identifikacijos numeris;
- aptikimo laikas;
- priimtas sprendimas.

Tokiu būdu duomenų bazė veikia kaip centrinis elemento ryšys tarp paketų skanavimo, atributų analizės ir anomalijų aptikimo modulių, leidžiantis tiek realaus laiko, tiek ilgalaikę lėtų DoS atakų stebėseną.

2.6. Lėtų DoS atakų aptikimo veikimas

Realizuojamo metodo lėtų DoS atakų aptikimo modulis susidės iš dviejų dalių:

1. Kelių etapų laiko slankiojo lango analizė pagal nustatytus parametrus ir taisykles – naudodami šį metodą galėsime stebėti MQTT klientų veikimą skirtingais laiko intervalais ir aptikti

kenksmingus klientus pagal konkrečius rodiklius. Naudodamiesi šiuo būdu galime apskaičiuoti ir aptikti klientus, kurie prieštarauja iš anksto nustatytiems elgesio modeliams ir taisyklėmis.
2. Anomalijų aptikimą naudojant mašininį mokymąsi – naudodami šį būda padidiname šansus aptikti kenksmingus klientus, kurie veikia neįprastu būdą, kurių elgesys nepažeidžia jokių fiksuotų taisyklių. Tam taikysime vienklasinius modelius, kurie bus apmokomi, kai duomenų laikotarpyje nevyko jokių atakų.

Abi šios dalys veikia kartu – taisyklėmis paremta analizė leidžia aptikti žinomas ir interpretuojamas atakų charakteristikas, o mašininio mokymosi modelis papildo sistemą gebėjimu identifikuoti netikėtas ar naujas anomalijas.

2.7. Slankiojo laiko analizės veikimas

Ši dalis susidarys iš trijų skirtingų laiko serijų intervalų:
1. Trumpas langas (10 min.) – skirtas aptikti trumpalaikes anomalijas ir neįprastus jungimosi modelius (pvz., dažnus trumpalaikius prisijungimus ir trumpalaikį jungčių šuolį), kurie gali būti pradinių etapų indikatorius.
2. Vidutinis langas (1 val.) – skirtas aptikti vidutinio laikotarpio lėtas DoS taktikas, pvz., palaipsniui didėjančią jungčių koncentraciją arba klientų ID/IP sukimąsi per ilgesnį periodą.
3. Ilgas langas (3 val.) – skirtas aptikti ilgalaikes lėtas DoS atakas, kurios vystosi kelias valandas ar ilgiau ir kurių tikslas – užimti brokerio jungčių išteklius be akivaizdžių momentinių srauto pikų.

Šie intervalai buvo pasirinkti dėl to, nes lėtoms DoS atakoms būdinga tai, kad jų poveikis kaupiasi per ilgą laiką – užpuolikai siekia išnaudoti brokerio jungčių arba sesijų talpą palaipsniui, todėl būtina turėti ir trumpalaikį, ir ilgalaikį stebėjimą. Trumpesni langai leidžia greitai užfiksuoti pradines anomalijas, o vidutiniai ir ilgi langai suteikia galimybę patvirtinti ir sumažinant klaidingų signalų kiekį.

Visi langai skaičiuojami persidengiančiai su 50 % persidengimu (kiekvienas naujas langas prasideda po pusės ankstesnio lango trukmės). Pavyzdžiui:
- 10 min. langai: 17:30 – 17:40, 17:35 – 17:45;
- 1 val. langai: 17:30 – 18:30, 18:00 – 19:00;
- 3 val. langai: 17:30 – 20:30, 19:00 – 22:00.

Tokia strategija suteikia dažnesnius ir tikslesnius rezultatus, ilgesnį stabilumą, galimybę pritaikyti daugiasluoksnes taisykles.

Visi minėti laiko langai yra bendri visam tinklo srautui, vadinasi, tuo pačiu metu analizuojami visi gauti MQTT paketai iš visų klientų. Kiekvienas paketas priskiriamas atitinkamam laiko intervalui pagal jo gavimo laiką, todėl visi klientai yra vertinami bendrame laiko kontekste. Nors laiko langai yra bendri visiems MQTT klientams, atributai apskaičiuojami kiekvienam klientui atskirai. Tai reiškia, kad kiekviename lange sistema grupuoja duomenis pagal ClientID arba IP adresą ir kiekvienam jų nustato atskiras metrikas.

Kadangi jau paketų skanavimo modulis apdoroja renkamus paketus, papildomai jų nereikia konvertuoti, nebent papildomai pridėti tuščias reikšmes, nes kiekvienas paketas turi skirtingus parametrus ir jų kiekį.

Pagrindinių skaičiuojamų atributų sąrašas:

1. Naujų sesijų skaičius (*CONNACK*) kiekviename laiko intervale leidžia įvertinti, kiek kartų klientas ar IP sėkmingai prisijungė prie tarpininko laiko intervale. Šio atributo skaičiavimas padeda nustatyti neįprastus jungimosi šuolius, kurie gali reikšti tarpininko jungčių užėmimo bandymą.
2. Tylių sesijų skaičius ir santykis parodo, kiek sesijų po sėkmingo prisijungimo (*CONNACK*) per laiką, normalizuotą pagal derybose nustatytą *Keep Alive*, nesiunčia jokio kliento srauto (pvz., *PUBLISH*, *SUBSCRIBE*, *PINGREQ* ar *DISCONNECT*). Šis atributas padeda identifikuoti jungtis, kurios laikomos aktyviomis be realios veiklos.
3. *PINGREQ* dažnis, normalizuotas pagal *Keep Alive*, parodo, kaip dažnai klientas palaiko sesiją lyginant su leidžiamu neveiklumo laiku. Tai leidžia palyginti skirtingus klientus net tada, kai jų *Keep Alive* nustatymai skiriasi.
4. Paketų *PUBLISH* ir *SUBSCRIBE* dažnis leidžia nustatyti, ar klientas realiai vykdo duomenų perdavimą / prenumeravimą, ar tik palaiko jungtį. Šis atributas svarbus atskiriant mažo aktyvumo normalius klientus nuo tyčia palaikomų jungčių.
5. Pakartotinių prisijungimų (*CONNACK*) dažnis per tą patį *ClientID* arba IP adresą leidžia aptikti klientus, kurie dažnai persijungia, o ypač – kai persijungimai kartojami be realios veiklos.
6. Sesijų koncentracijos indeksas (HHI) apskaičiuojamas siekiant įvertinti, ar vienam IP adresui tenka neproporcingai didelė naujų sesijų dalis laiko intervale.
7. *Keep Alive* reikšmės nuokrypis nuo bazinio lygio leidžia nustatyti, ar klientas naudoja neįprastai dideles brokerio patvirtintas *Keep Alive* reikšmes. Šis atributas pats savaime nėra pakankamas atakai nustatyti, tačiau sustiprina įtarimą, kai kartu stebimas tylių sesijų elgesys.

Kiekvienas apskaičiuotas atributas bus naudojamas ne atskirai, o kartu su kitais rodikliais, kad būtų galima sudaryti bendrą kliento elgesio vaizdą. Visi atributai bus kaupiami pagal laiko intervalus (10 min., 1 val., 3 val.,) ir bus vertinami tiek lokaliai, tiek globaliai. Tokia analizė leidžia nustatyti tiek pavienius nenormalius klientus, tiek bendrą tinklo būklės pokytį. Kad užtikrintume tikslumą ir išvengtume klaidingų signalų, kiekvienas atributas bus lyginimas su ankstesnio lango reikšmėmis ir baziniu normaliu elgesiu, kuris bus nustatytas stebint sistemą kurioje nevyksta jokios atakos. Reikšmingi skirtumai tarp langų ir nuokrypiai nuo bazinės reikšmės padės įvertinti ar MQTT klientų elgesys veikia įtartinai.

Turėdami atributus ir laiko intervalus galime naudoti tikslias taisykles kiekvienam atributui:

1. (IP lygmeniu) Jeigu per tam tikrą laiko langą vienam IP adresui priskiriamų naujų sesijų skaičius reikšmingai viršija kitų IP adresų reikšmes, tokia elgsena laikoma įtartina. Ši taisyklė vertinama dviem aspektais:

   – IP adresas laikomas išskirtiniu, jei jo sesijų skaičius ženkliai viršija bendrą sesijų pasiskirstymo medianą, o nuokrypis vertinamas pagal medianą ir medianinį absoliutų nuokrypį, taikant z-reikšmės slenkstį 3,0;
   – papildomai tikrinamas augimas lyginant su ankstesniu to paties tipo laiko langu, kai sesijų skaičius padidėja daugiau nei 30 %, esant sąlygai, kad ankstesniame lange buvo bent 3 sesijos. Taisyklė laikoma suaktyvinta, kai rizikos balas viršija 0,7 ir sesijų skaičius yra ne mažesnis nei 3;

2. (IP lygmeniu) Jeigu tylių sesijų santykis vienam IP adresui yra ne mažesnis nei 0,8 ir per laiko langą nustatyta bent 2 sesijos, toks elgesys laikomas būdingu lėtoms DoS atakoms. Siekiant išvengti klaidingų teigiamų signalų dėl natūralių pavienių ryšio klaidų, taikomas tolydinis

pasitikėjimo koeficientas. Pilna rizika priskiriama tik tada, kai IP adresas sukuria 3 ar daugiau sesijų, o, esant mažesniam sesijų skaičiui, rizikos balas proporcingai sumažinamas;
3. (IP lygmeniu) Jeigu didelė dalis IP adreso sesijų naudoja neįprastai dideles *Keep Alive* reikšmes ir tuo pačiu metu yra tylios, apskaičiuojamas santykis tarp tylių sesijų su dideliu *Keep Alive* ir bendro sesijų skaičiaus. Taisyklė naudoja tokį patį pasitikėjimo koeficientą kaip ir 2 taisyklė bei laikoma suaktyvinta, kai šis santykis siekia ne mažiau nei 0,5 ir sesijų skaičius yra bent 2;
4. (*ClientID* lygmeniu) Jeigu MQTT klientui priskirtų sesijų, kuriose nesiunčiami *PUBLISH* ar *SUBSCRIBE* paketai, tačiau ryšys su tarpininku išlaikomas neįprastai ilgą laiką (pavyzdžiui, viršija 1,5 karto derybose nustatyto *Keep Alive* laiko), toks klientas laikomas įtartinu;
5. (*ClientID* lygmeniu) Jeigu tas pats MQTT klientas tyliai užima tarpininko resursus be jokio naudingo srauto, apskaičiuojama, kokią procentinę laiko lango dalį (pvz., iš 10 minučių) šis klientas laikė atidarytą pasyvią sesiją. Taisyklė laikoma suaktyvinta, jei klientas pasyviai išnaudoja daugiau nei 80 % viso laiko lango trukmės;
6. (IP lygmeniu) Jeigu tylių sesijų koncentracijos indeksas (HHI) viršija 0,3 arba per vieną laiko langą padidėja daugiau nei 0,1 lyginant su ankstesniu langu, laikoma, kad didelė dalis jungčių yra sutelkta viename IP adresų segmente, kas gali indikuoti centralizuotą lėtą ataką.

Kiekviena taisyklė sukuria atskirą rizikos balą intervale [0; 1]. Bendras rizikos balas apskaičiuojamas taip, kad kiekvienos taisyklės indėlis sumažina „normalaus elgesio“ tikimybę, o galutinis rezultatas parodo bendrą elgsenos riziką.

$$R_{rules} = 1 - \prod_i (1 - s_i) \qquad (1)$$

s_i – atskiros taisyklės sugeneruotas rizikos balas.

Apskaičiavus bendrus rizikos balus, metodas papildomai įtraukia rizikos paveldėjimo mechanizmą, skirtą tinklo ir kliento lygmens stebėjimo rezultatams susieti. Šis mechanizmas remiasi principu, kad jeigu IP adresas pagal nustatytas taisykles įvertinamas kaip įtartinas, kiekvienas iš šio adreso prisijungęs MQTT klientas automatiškai paveldi 50 % suminio IP adreso rizikos balo. Paveldėtas dydis klientui priskiriamas kaip papildomas rizikos veiksnys, kuris sujungiamas su individualiomis kliento (*ClientID*) taisyklėmis nustatytu balu naudojant bendrąją rizikos jungties išraišką. Tokia strategija leidžia identifikuoti lėtąsias atakas, kurių metu užpuolikas siekia išvengti detekcijos kurdamas po vieną pasyvią sesiją iš to paties IP adreso ir taip paskirstydamas kenkėjišką veiklą tarp skirtingų identifikatorių, kurie pavieniui neviršytų nustatytų elgsenos slenksčių.

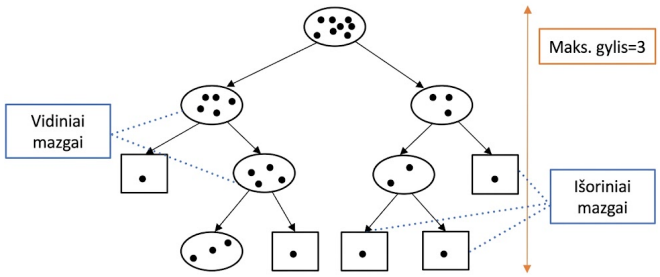
2.8. Anomalijų aptikimas naudojant mašininį mokymąsi

Be taisyklių kiekvienam laiko langui sugeneruojamas atributų vektorius, kuris susideda iš šių reikšmių: kliento pradėtų sesijų kiekis, kliento tylių sesijų santykis, kliento gautų paskelbimo ar prenumeratos santykis su laiko intervalu sekundėmis, vidutinis normalizuotas PING dažnis kliento sesijų metu, tylių sesijų su aukštu *keep_alive* parametru kiekis, lango dydis minutėmis. Turėdami šiuos vektorius juos paduodame į anomalijų aptikimo modelį, paremtą izoliuoto miško (angl. *Isolation Forest*) algoritmu.

Modelis mokomas naudojant tik tuos duomenis, kurie surinkti iš laikotarpių, kai tinkle nevyko atakų – vadinasi, jis išmoksta reprezentuoti „normalų“ MQTT klientų elgesį. Dėl to modelis geba identifikuoti klientus, kurių elgesys žymiai skiriasi nuo įprasto, net jei tie skirtumai nėra aprašyti

jokiose taisyklėse. Tuomet modelio rezultatas normalizuojamas į intervalą [0, 1]. Kuo reikšmė artimesnė 1, tuo didesnė tikimybė, kad elgesys neatitinka įprasto modelio.

Izoliuotojo miško algoritmas yra priskiriamas prie neprižiūrimų mašininio mokymosi metodų, nes jam nereikia iš anksto pažymėtų duomenų pagal tai, kokio tipo yra įrašas. Vietoje to, algoritmas pats formuoja normalaus duomenų pasiskirstymo vaizdą ir identifikuoja įrašus, kurie nuo šio modelio reikšmingai nukrypsta. Tokia savybė ypač naudinga siekiant aptikti žemo intensyvumo DoS atakas, kurios gali likti nepastebėtos tradicinėmis priemonėmis.



13 pav. Izoliuoto miško (angl. *Isolation Forest*) veikimo principas [36]

13 pav. atvaizduotas Izoliuoto miško (angl. *Isolation Forest*) veikimo principas, parodantis, kaip duomenys yra atsitiktinai dalijami į poskyrius, kol atskiriami anomalūs įrašai, kuriems pasiekti reikia mažiau dalijimų nei įprastiems duomenims.

Modelis mokomas naudojant tik tuos duomenis, kurie surinkti iš laikotarpių, kai tinkle nevyko jokių atakų. Tokiu būdu modelis išmoksta reprezentuoti „normalų“ MQTT klientų elgesį, atsižvelgdamas į įprastą paros bei apkrovos ciklų dinamiką. Kad modelį būtų galima tinkamai apmokyti, naudojamas pakankamas duomenų kiekis, kad būtų atvaizduoti tipiniai tinklo elgsenos modeliai.

Po apmokymo nauji laiko langų vektoriai perduodami modelio vertinimui. Izoliuoto medžio algoritmas apskaičiuoja kiekvieno vektoriaus „izoliacijos gylį“, t. y., kiek dalijimų reikia, kol konkretus duomenų taškas yra izoliuojamas nuo kitų. Kuo mažiau dalijimų reikia — tuo labiau tikėtina, kad šis taškas yra anomalus.

Tokiu būdu, net jei elgesio anomalijos nėra aprašytos jokiose taisyklėse, modelis gali aptikti klientus, kurių veikla išsiskiria nuo įprasto tinklo fono. Tai leidžia identifikuoti ilgalaikes, mažo intensyvumo atakas, kurios dažnai nepastebimos tradicinėmis heuristinėmis ar ribinėmis priemonėmis. Be to, dėl modelio neprižiūrimo pobūdžio jis gali būti periodiškai atnaujinamas ir adaptuojamas prie kintančių tinklo sąlygų, užtikrinant aukštą aptikimo tikslumą ilgalaikėje perspektyvoje.

Apmokant izoliuoto miško modelį, išskiriami trys pagrindiniai parametrai, turintys didžiausią įtaką gaunamiems rezultatams. Pirmasis yra naudojamas medžių kiekis, nurodantis, kiek atsitiktinių izoliavimo medžių bus sukurta. Didesnis medžių kiekis suteikia stabilesnius rezultatus ir mažina atsitiktiniu nukrypimų tikimybę, tačiau modelio apmokymo bei sprendimo priėmimo trukmė padidėja. Priešingu atveju pasirinkus per mažą medžių kiekį, rezultatai gali būti netikslūs. Antrasis parametras apibrėžia duomenų užterštumo dalį, kuri nurodo, kokią dalį stebėjimų modelis turėtų laikyti anomalijomis. Paskutinis parametras, naudojamas siekiant rezultatų atkuriamumo, kuris užtikrina, kad atsitiktiniai duomenų padalinimai kiekvieną kartą vyktų vienodai.

2.9. Gautų rezultatų palyginimas

Gauti rezultatai iš slankiojo laiko analizės paremto taisyklėmis (žr. **2.7**) ir anomalijų aptikimo naudojant mašininį mokymąsi (žr. **2.8**) tam pačiam laiko intervalui yra sudedami pagal alfa svorio reikšmę. Tokiu būdu galima tiksliau identifikuoti galimai kenksmingus MQTT klientus ir tiksliau įvertinti jų elgesio riziką. Kiekvienam klientui skaičiuojamas kombinuotas rezultatas, susidedantis iš dviejų dalių:

- taisyklių analizės balas (R_{rules});
- izoliuoto miško modelio balas (R_{ml}).

Norint įvertinti bendrą riziką, šie du rodikliai sujungiami į vieną kombinuotą reikšmę, apskaičiuojamą pagal išraišką:

$$R_{komb} = \alpha * R_{rules} + (1 - \alpha) * R_{ml} \quad (2)$$

Kur α – svorio koeficientas, leidžiantis pritaikyti modelį pagal norimą jautrumo ir tikslumo balansą. Toks kombinuotas taisyklių ir mašininio mokymosi modelio derinys leidžia pasiekti aukštesnį aptikimo tikslumą ir sumažinti klaidingų teigiamų signalų skaičių.

2.10. Projekto dalies išvados

Sukūrę lėtų DoS atakų atpažinimo MQTT protokole metodo modelį galima padaryti šias išvadas:

1. Realizuojant atskirą paketų skanavimo įrankį, metodas nepriklauso nuo MQTT tarpininko ir galima metodą naudoti ant skirtingų tarpininkų;
2. Kadangi serveris ant kurio MQTT tarpininkas yra paleidžiamas dažnu atveju būna riboto resursų, todėl visą aptikimo logiką reikia iškelti į atskirą serverį, kuris turi daugiau resursų;
3. Naudojant dviejų dalių lėtų DoS atakų aptikimą padididėja tikimybė aptikti kenksmingus MQTT klientus;
4. Realizuota slenkančio laiko serijos analizė su skirtingais laiko intervalais padeda aptikti skirtingai veikiančias lėtas DoS atakas;
5. Realizuota anomalijų aptikimo analizė sugebės aptikti anomalijas iš laiko serijos analizės rezultatų, kurie skiriasi nuo įprasto MQTT tarpininko veikimo.

3. Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodo projekto realizacija ir eksperimentinė aplinka

Šiame realizacijai skirtame skyriuje yra aprašomi prototipo modulių veikimo principai, naudotos technologijos bei naudoti įrenginiai.

3.1. MQTT tarpininko konfigūracija

MQTT 5.0 protokolo tarpininku buvo pasirinkta „*Eclipse Mosquitto*“ dėl jo atvirojo kodo licencijos, paprastos konfigūracijos, plataus funkcionalumo ir galimybės lengvai diegti jį bet kuriame nuosavame serveryje. Šis tarpininkas palaiko naujausias MQTT specifikacijas, yra aktyviai prižiūrimas bendruomenės ir pasižymi stabiliu veikimu tiek mažo, tiek didelio apkrovimo aplinkose. Norint paleisti MQTT tarpininką serveryje, pirmiausia būtina tinkamai sukonfigūruoti „*mosquitto.conf*“ konfigūracijos failą. Šiame faile nurodomi pagrindiniai tarpininko veikimo parametrai: naudojami prievadai, klausymo IP adresai, autentifikacijos metodai, TLS/SSL sertifikatai, leidimų politika bei papildomos funkcijos, tokios kaip „*persistence*“ ar žurnalų (angl. *log*) generavimas. Teisinga konfigūracija yra būtina siekiant užtikrinti saugų brokerio veikimą.

Pagrindinis parametras, į kurį buvo atsižvelgta, yra „*max_keepalive*“. Jis apibrėžia maksimalią leistiną sesijos neaktyvumo trukmę: jei per laikotarpį, lygų 1,5 karto nustatytos reikšmės, tarpininkas (angl. *broker*) negauna jokio pranešimo iš kliento, sesija automatiškai nutraukiama. Nustačius per mažą šio parametro reikšmę, išauga tinklo apkrova dėl dažnų sesijos palaikymo paketų siuntimo *PINGREQ*, o tai eikvoja kliento įrenginio energijos resursus. Kitu atveju pasirinkus itin didelę reikšmę ryšys gali nutrūkti dėl išorinių veiksnių, apie tai nesužinant nei tarpininkui, nei klientui. Nors maksimali protokolo palaikoma reikšmė yra 65 535 sekundės (daugiau nei 18 val.), praktinėje tinklo infrastruktūroje maršruto parinktuvai ir NAT mechanizmai neaktyvias TCP sesijas dažniausiai nutraukia po 30-60 minučių, kai kuriais atvejais net greičiau. Atsižvelgiant į tai, šiame darbe pasirinktas 1000 sekundžių (16 min. 40 sek.) intervalas, užtikrinantis stabilų ryšį be perteklinės apkrovos. Taip pat norint ištestuoti visų leistinų sesijų užėmimą buvo nustatytas „*max_connections*” parametras į 1024, kuris vienu metu leidžia tik tokį klientų kiekį, priešingu atveju klientas išsiuntęs prisijungimo paketą negaus jokio atsakymo.

```
2025-09-30T23:24:38: mosquitto version 2.0.22 starting
2025-09-30T23:24:38: Config loaded from D:\KTU\magistras\MBD\3semestras\mosquitto.conf.
2025-09-30T23:24:38: Opening ipv4 listen socket on port 1883.
2025-09-30T23:24:38: mosquitto version 2.0.22 running
2025-09-30T23:25:48: New connection from 192.168.0.11:38175 on port 1883.
2025-09-30T23:25:48: New client connected from 192.168.0.11:38175 as auto-DA83A196-E770-DC66-61C8-21B751452871 (p2, c1, k10).
2025-09-30T23:25:48: No will message specified.
2025-09-30T23:25:48: Sending CONNACK to auto-DA83A196-E770-DC66-61C8-21B751452871 (0, 0)
2025-09-30T23:25:48: Received SUBSCRIBE from auto-DA83A196-E770-DC66-61C8-21B751452871
2025-09-30T23:25:48:    test (QoS 0)
2025-09-30T23:25:48: auto-DA83A196-E770-DC66-61C8-21B751452871 0 test
2025-09-30T23:25:48: Sending SUBACK to auto-DA83A196-E770-DC66-61C8-21B751452871
2025-09-30T23:25:58: Received PINGREQ from auto-DA83A196-E770-DC66-61C8-21B751452871
2025-09-30T23:25:58: Sending PINGRESP to auto-DA83A196-E770-DC66-61C8-21B751452871
2025-09-30T23:25:59: Client auto-DA83A196-E770-DC66-61C8-21B751452871 closed its connection.
```

14 pav. MQTT protokolo žinučių pavyzdys

14 pav. pavaizduotas MQTT protokolo žinučių pavyzdys. Jame demonstruojamas tarpininko paleidimo procesas bei gautos sistemos žinutės, informuojančios apie naudojamą brokerio versiją, įkeltą konfigūracijos failą, atidarytus prievadus ir MQTT kliento prisijungimo prie tarpininko eigą. Šis žinučių formatas yra pritaikytas žmogaus akiai, todėl aiškiai ir struktūruotai pateikia pagrindinę

informaciją apie vykstančius procesus. Vis dėlto tokios žinutės paslepia daug detalių, esančių žemesnio lygmens tinklo paketuose. Būtent dėl to naudojamas paketų nuskaitymo modulis, kuris leidžia gauti neapdorotus MQTT paketus, analizuoti jų turinį, stebėti protokolo veikimą *byte* lygmeniu ir aptikti galimus anomalijų ar atakų požymius.

Siekiant izoliuoti eksperimentinę aplinką ir užtikrinti jos atkuriamumą, buvo naudojama konteinerizacijos technologija *Docker*. Asmeniniame kompiuteryje buvo sukurtas atskiras *Docker* tinklas pavadinimu „*mqtt_net*“, kuriame paleistas MQTT tarpininkas. Toks sprendimas leido valdyti tinklo parametrus, izoliuoti eksperimentinius srautus nuo pagrindinės sistemos ir lengvai keisti konfigūraciją atliekant skirtingus bandymus.

3.2. MQTT klientų veikimas

Generuoti pakankamai intensyvų ir įvairų MQTT paketų srautą realiomis analizės sąlygomis reikalauja didelio kiekio MQTT klientų. Simuliuoti tokį srautą naudojant fizinius mikrovaldiklius būtų nepraktiška ir labai sudėtinga, nes kiekvieną mikrovaldiklį reikėtų atskirai sukonfigūruoti, prijungti prie tinklo ir pritaikyti taip, kad jis generuotų skirtingo tipo žinutes su skirtingais intervalais. Dėl šių priežasčių buvo pasirinkta naudoti programinį įrankį [37], leidžiantį generuoti MQTT srautą imituojant daugelio įrenginių veikimą per *Python* programą. Šios programos funkcionalumo neužteko visiškai simuliuoti normalių įrenginių veikimo, todėl programinė įranga buvo papildyta pagal reikalavimus, kad galėtų turėti skirtingus *PUBLISH* veikimo principus, tokiu, kad prisijungus laikytų jungtį nuolat atvirą, kad išsiuntus *PUBLISH* paketą iš karto atsijungtų ir kitą kartą, kai norės išsiųsti *PUBLISH* paketą, turės vėl prisijungti. Taip pat buvo pridėtas funkcionalumas, kad būtų galima paketus siųsti pagal tam tikrą intervalą su papildomu nuokrypiu, kad paketų dažnis nebūtų fiksuoto ilgio.

Šiam tikslui buvo sukurta 55 skirtingų virtualių įrenginių konfigūracijų, kurias galima matyti **3.1 lentelėje**:

3.1 lentelė. MQTT klientai

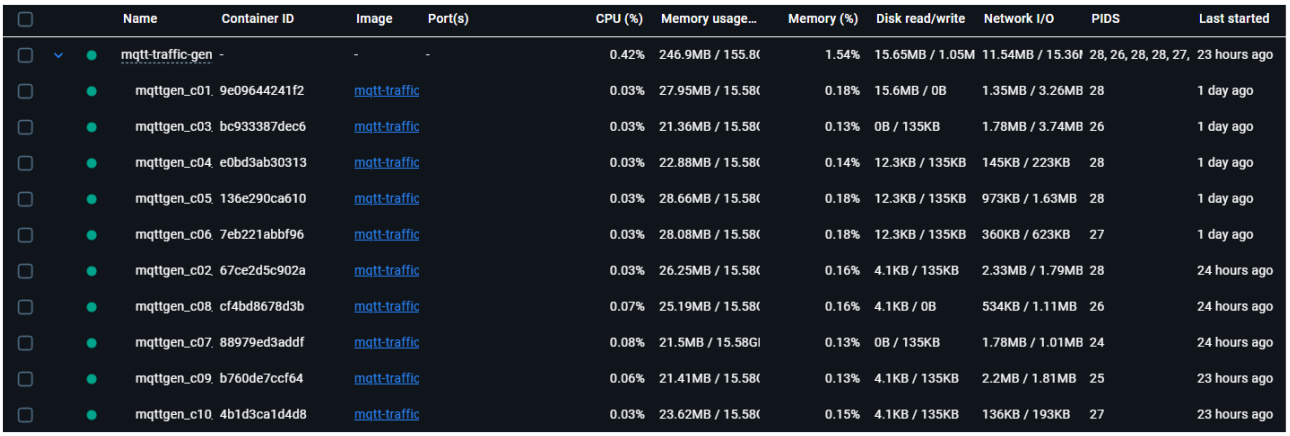
Nr.	Tema	QoS	Dažnis, sec.	Tipas	Veikimas
1.					
1.	*home/thermostat/firmware*	0	43 200	*Publish*	Atsijungia
2.	*home/living/thermostat/humidity*	0	120	*Publish*	Palaiko su nuokrypiu
3.	*home/living/thermostat/humidity*	0	120	*Publish*	Palaiko su nuokrypiu
4.	*home/living/thermostat/temp*	1	10	*Publish*	Palaiko su nuokrypiu
5.	*home/living/thermostat/temp*	1	10	*Publish*	Palaiko su nuokrypiu
6.	*home/thermostat/firmware*	2	30 000	*Publish*	Atsijungia
2.					
7.	*home/door/mainlock*	1	300	*Publish*	Palaiko su nuokrypiu
8.	*home/security/panel/heartbeat*	1	1800	*Publish*	Palaiko su nuokrypiu
9.	*home/bedroom/window*	1	900	*Publish*	Atsijungia
10.	*home/bedroom/window*	1	900	*Publish*	Atsijungia
11.	*home/garage/door*	1	600	*Publish*	Atsijungia
12.	*home/camera/#*	1		*Subscribe*	

3.					
13.	*home/camera/backyard/motion*	2	12	*Publish*	Palaiko su nuokrypiu
14.	*home/camera/frontdoor/motion*	0	10	*Publish*	Palaiko su nuokrypiu
15.	*home/gas/meter*	1	3600	*Publish*	Palaiko su nuokrypiu
16.	*home/power/meter/instant*	1	30	*Publish*	Palaiko su nuokrypiu
17.	*home/water/meter*	0	1800	*Publish*	Palaiko su nuokrypiu
4.					
18.	*home/power/meter/daily*	1	7200	*Publish*	Atsijungia
19.	*home/boiler/status*	0	300	*Publish*	Palaiko su nuokrypiu
20.	*home/solar/inverter/power*	1	900	*Publish*	Palaiko su nuokrypiu
21.	*home/boiler/pressure*	0	1800	*Publish*	Atsijungia
22.	*home/ev/charger/state*	1	600	*Publish*	Palaiko su nuokrypiu
23.	*home/solar/inverter/status*	2	3600	*Publish*	Atsijungia
5.					
24.	*office/floor1/room1/temp*	0	45	*Publish*	Palaiko su nuokrypiu
25.	*office/floor1/room1/temp*	0	45	*Publish*	Palaiko su nuokrypiu
26.	*office/floor1/room2/co2*	1	55	*Publish*	Palaiko su nuokrypiu
27.	*office/floor1/room2/temp*	0	50	*Publish*	Palaiko su nuokrypiu
28.	*office/floor1/room2/temp*	0	50	*Publish*	Palaiko su nuokrypiu
29.	*office/floor1/room1/occupancy*	0	30	*Publish*	Atsijungia
6.					
30.	*office/elevators/panel*	1	300	*Publish*	Palaiko su nuokrypiu
31.	*office/parking/gate*	1	300	*Publish*	Palaiko su nuokrypiu
32.	*office/parking/sensors/free*	0	120	*Publish*	Palaiko su nuokrypiu
33.	*office/entry/turnstile*	1	180	*Publish*	Atsijungia
34.	*office/generator/status*	1	7200	*Publish*	Atsijungia
35.	*office/elevator/#*	1		*Subscribe*	
7.					
36.	*home/boiler/#*	1		*Subscribe*	
37.	*city/weather/#*	1		*Subscribe*	
38.	*plant/pump1/#*	1		*Subscribe*	
39.	*plant/pump1/#*	1		*Subscribe*	
8.					
40.	*plant/chiller1/pressure*	1	600	*Publish*	Palaiko su nuokrypiu
41.	*plant/pump1/status*	1	20	*Publish*	Palaiko su nuokrypiu
42.	*plant/pump1/vibration*	0	120	*Publish*	Palaiko su nuokrypiu
43.	*plant/pump1/vibration*	0	120	*Publish*	Palaiko su nuokrypiu
44.	*plant/chiller1/temp*	1	300	*Publish*	Atsijungia
9.					
45.	*plant/line1/throughput*	0	300	*Publish*	Palaiko be nuokrypio

46.	*plant/line1/power*	1	30	*Publish*	Palaiko su nuokrypiu
47.	*plant/line1/power*	1	30	*Publish*	Palaiko su nuokrypiu
48.	*plant/water/quality*	0	1800	*Publish*	Palaiko su nuokrypiu
49.	*plant/#*	1		*Subscribe*	
10.					
50.	*cloud/firmware/check*	0	21 600	*Publish*	Atsijungia
51.	*cloud/firmware/check*	0	21 600	*Publish*	Atsijungia
52.	*city/weather/station01*	0	1200	*Publish*	Palaiko su nuokrypiu
53.	*fleet/truck01/gps*	0	15	*Publish*	Palaiko be nuokrypio
54.	*fleet/truck01/diagnostics*	1	600	*Publish*	Atsijungia
55.	*fleet/#*	1		*Subscribe*	

Aprašyti MQTT klientai reprezentuoja įvairių sričių įrenginius. Kiekvienas klientas turi skirtingą temą, QoS lygį, pranešimų siuntimo dažnį bei tipą (*Publish*, *Subscribe*). Tokiu būdu sudaroma realistinė, kompleksiška DI aplinka, kurioje vienu metu egzistuoja dažnai atnaujinamų sensorių duomenys, epizodiniai pranešimai, įvykiai realiuoju laiku, periodiniai statuso atnaujinimai ir retai siunčiamos priežiūros ar konfigūracijos žinutės. Šių klientų generuojamas srautas leidžia sistemai veikti ištisomis dienomis ar net savaitėmis. Naudojant šias konfigūracijas per tris dienas buvo sugeneruojama daugiau nei 300 000 individualių MQTT paketų, kurie bus skirti apmokyti mašininio mokymosi algoritmą izoliuotas miškas (angl. *Isolation Forest*).

Kadangi realizuojamame atakų aptikimo algoritme yra specifinės taisyklės skaičiuojančios individualių IP adresų kiekius turėti visus MQTT klientus, veikiančius iš vieno IP adreso būtų netikslinga, nes ir realiame MQTT tinkle klientai veikia iš skirtingų IP adresų. Todėl paleisti šiuos klientus buvo pasinaudota praeitame skyriuje minėta *Docker* technologija išskiriant MQTT klientų konfigūracijas į dešimt atskirų konteinerių, kurie duoda dešimt skirtingų IP adresų, sukuria realistiškesnį MQTT tinklo veikimą. Tai galima matyti **15 pav.**



Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Disk read/write	Network I/O	PIDS	Last started
mqtt-traffic-gen	-	-	-	0.42%	246.9MB / 155.8(	1.54%	15.65MB / 1.05M	11.54MB / 15.36I	28, 26, 28, 28, 27,	23 hours ago
mqttgen_c01	9e09644241f2	mqtt-traffic		0.03%	27.95MB / 15.58(	0.18%	15.6MB / 0B	1.35MB / 3.26MB	28	1 day ago
mqttgen_c03	bc933387dec6	mqtt-traffic		0.03%	21.36MB / 15.58(	0.13%	0B / 135KB	1.78MB / 3.74MB	26	1 day ago
mqttgen_c04	e0bd3ab30313	mqtt-traffic		0.03%	22.88MB / 15.58(	0.14%	12.3KB / 135KB	145KB / 223KB	28	1 day ago
mqttgen_c05	136e290ca610	mqtt-traffic		0.03%	28.66MB / 15.58(	0.18%	12.3KB / 135KB	973KB / 1.63MB	28	1 day ago
mqttgen_c06	7eb221abbf96	mqtt-traffic		0.03%	28.08MB / 15.58(	0.18%	12.3KB / 135KB	360KB / 623KB	27	1 day ago
mqttgen_c02	67ce2d5c902a	mqtt-traffic		0.03%	26.25MB / 15.58(	0.16%	4.1KB / 135KB	2.33MB / 1.79MB	28	24 hours ago
mqttgen_c08	cf4bd8678d3b	mqtt-traffic		0.07%	25.19MB / 15.58(	0.16%	4.1KB / 0B	534KB / 1.11MB	26	24 hours ago
mqttgen_c07	88979ed3addf	mqtt-traffic		0.08%	21.5MB / 15.58GI	0.13%	0B / 135KB	1.78MB / 1.01MB	24	24 hours ago
mqttgen_c09	b760de7ccf64	mqtt-traffic		0.06%	21.41MB / 15.58(	0.13%	4.1KB / 135KB	2.2MB / 1.81MB	25	23 hours ago
mqttgen_c10	4b1d3ca1d4d8	mqtt-traffic		0.03%	23.62MB / 15.58(	0.15%	4.1KB / 135KB	136KB / 193KB	27	23 hours ago

15 pav. MQTT klientų „Docker“ konteineriai

Norint ištestuoti specifines situacijas ir funkcionalumus, papildomai buvo naudojamos „*Eclipse Mosquitto*“ komandinės eilutės priemonės, tokios kaip *mosquitto_pub* ir *mosquitto_sub*. Jos leidžia rankiniu būdu generuoti *Publish* ir *Subscribe* tipo pranešimus, taip pat greitai keisti siuntimo dažnį, temų struktūrą ar siųsti specialiai suformuotus paketų variantus. Tai suteikia galimybę patikrinti, ar

MQTT tarpininkas tinkamai atlieka žinučių priėmimą, QoS valdymą ir ar sukurtas paketų nuskaitymo modulis sugeba aptikti ir teisingai interpretuoti visus gautus paketus.

Norint imituoti lėto veikimo DoS (angl. *Slow DoS*) atakas, kurios dažnai yra naudojamos siekiant išnaudoti brokerio resursus neužgožiant tinklo dideliu srautu, buvo pasitelktas atskiras *Python* įrankis, leidžiantis generuoti lėtai siunčiamus, pagal nustatytą intervalą ir norimą MQTT klientų kiekį bei norimą veikimo principą iš dviejų:

1. *Wait* – laukimo principu laukia, kol MQTT tarpininkas sustabdys sesiją dėl neveiksmingumo viršinant 1.5 *KeepAlive* reikšmės ir iš karto įrankis sukuria naują klientą.

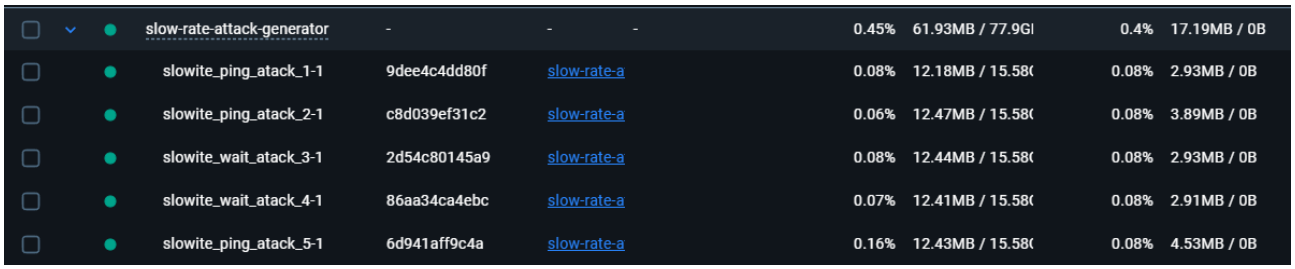
2. *Ping* – miegojimo principu, kol pasiekia 1,4 *KeepAlive* reikšmės ir išsiunčia *PING* paketą MQTT tarpininkui, kad paskutinio paketo laikas būtų pradėtas skaičiuoti iš naujo taip kartojant iki begalybės, nes jungties neveikimo trukmė niekad nepasiekia 1,5 *KeepAlive* veikimo karto.

Norint ištestuoti ar tinkamai veikia taisyklės ir ar mašininio mokymosi modelis tinkamai apsimokė, bei pilnai išnaudoti tarpininko laisvas sesijas taip užblokuojant naujų sesijų sukūrimą buvo naudojama tą pačią *Docker* technologija tam pačiame tinkle kaip ir kiti sistemos konteineriai. Buvo paleistos abiejų veikimų principų konteineriai su skirtingais parametrais, kuriuos galima matyti **3.2 lentelėje**.

3.2 lentelė. Kenksmingų klientų konteinerių parametrai

Nr.	**Veikimo principas**	**Intervalas**	**Klientų kiekis**
1.	*Ping*	60 sek.	200
2.	*Ping*	300 sek.	200
3.	*Wait*	200 sek.	200
4.	*Wait*	100 sek.	300
5.	*Ping*	30 sek.	90

Paleistus konteinerius galime matyti **16 pav.**:



16 pav. MQTT kenkėjų „Docker“ konteineriai

3.3. Paketų skanavimo modulio realizacija

Šio modulio realizacijai buvo pasirinkta naudoti *Python 3* programavimo kalbą su šiomis pagrindinėmis bibliotekomis, pridedančiomis reikiamą funkcionalumą, kad realizuoti metodą:

- ***argparse*:** naudojama komandinės eilutės argumentų apdorojimui, leidžianti vartotojui nurodyti programos parametrus paleidimo metu;
- ***scapy*:** tinklo paketų analizės ir kūrimo biblioteka, leidžianti fiksuoti, kurti bei nagrinėti MQTT ar kitus tinklo protokolų paketus;
- ***os*:** suteikia prieigą prie operacinės sistemos funkcijų, tokių kaip failų keliai, procesai ar sistemos aplinkos kintamieji;
- ***sys*:** leidžia tiesiogiai sąveikauti su *Python* vykdymo aplinka, pvz., nutraukti programą ar skaityti komandinės eilutės argumentus;
- ***dataclasses*:** leidžia paprastai apibrėžti duomenų struktūras (klases) su automatiškai generuojamais metodais;
- ***typing*:** suteikia galimybę naudoti tipų anotacijas, padedančias aiškiau apibrėžti funkcijų argumentų ir grąžinamų reikšmių tipus;
- ***struct*:** naudojama dvejetainių duomenų (pvz., tinklo paketų antraščių) kodavimui ir dekodavimui pagal nurodytą formatą;
- ***enum*:** leidžia apibrėžti vardinius konstantų rinkinius (enumeracijas), kad būtų aiškiau žymimos reikšmės, pvz., paketų tipai;
- ***datetime*:** naudojama laiko žymoms (*timestamp*) tvarkyti ir manipuliuoti datomis bei laikais, svarbu analizuojant paketų seką pagal laiką;
- ***time*:** suteikia priemones laiko matavimui ir uždelsimams (*delay*) įgyvendinti vykdant laiko langų analizę;
- ***abc*:** naudojama apibrėžti abstrakčias klases ir metodus, kad būtų galima kurti modulio struktūrą su paveldimais komponentais;
- ***signal*:** leidžia apdoroti sistemos signalus (pvz., `SIGINT`), užtikrinant tvarkingą programos nutraukimą ar išteklių atlaisvinimą;
- ***json*:** naudojama duomenų saugojimui ir mainams JSON formatu, pvz., eksportuojant paketų analizės rezultatus;
- ***psycopg2*:** suteikia galimybę saugoti surinktus duomenis duomenų bazėje, užtikrinant patogų rezultatų kaupimą ir analizę.

Šiai modulio realizacijai buvo pasirinkta naudoti objektinio programavimo principus su tokia failų struktūra:

```
MQTT_CAPTURE
├── __init__.py
├── capture_config.py
├── data_classes.py
├── enums.py
├── main.py
├── output/
│   ├── __init__.py
│   ├── database_writer.py
│   ├── json_writer.py
│   └── output_writer.py
├── output_manager.py
├── packet_capture.py
├── packet_parser.py
├── packets/
│   ├── __init__.py
│   ├── base_packet.py
```

```
│   ├── connack.py
│   ├── connect.py
│   ├── disconnect.py
│   ├── pingreq.py
│   ├── pingresp.py
│   ├── publish.py
│   └── subscribe.py
└── protocol_decoder.py
```

Paleidžiant programą galima nurodyti tris specialius argumentus, kurie prideda papildomą funkcionalumą moduliui, taip padidinant programos universalumą:

„*-i*“, „*—interface*“ skirtą nurodyti, kurią interneto jungtį norime stebėti, nenurodžius naudojama „*eth0*“.

„*-o*“, „*—output*“ skirta nurodyti aplankalą, kur saugoti aptiktus MQTT paketus, nenurodžius naudojamas „*mqtt_capture*“.

„*—ports*“ skirta nurodyti prievadus, kuriuos skanuoti, nenurodžius naudojami 1883, 8883 prievadai.

Tuomet sistema pradeda skanuoti nurodyto interneto jungties ir prievadų MQTT protokolo paketus, nuskanuoti paketai yra baitų formatu, todėl juos reikia konvertuoti į tinkamą formatą pagal MQTT protokolo taisykles [38], pirmiausia iššifruojama fiksuota antraštė iš kurios gaunamas paketo tipas, paskui likusių atributų iššifravimas pagal nustatytas taisykles kiekvienam paketui. Turėdami jau iššifruotą paketą viską išsaugome į vieną objektą, kurį sudaro tinklo informacija, paketo fiksuota antraštė, paketo antraštės atributai ir papildomi parametrai. Ir išsaugome gautą paketą į duomenų bazę, kuris bus naudojamas norint išanalizuoti laiko intervalą. Realizuotas modulis veikia pastoviai be jokio sustojimo, kol rankiniu būdu nebus sustabdytas.

CONNACK paketo pavyzdys:

Raw pradinis formatas:

203500003222000a1200296175746f2d32374333374538422d433532362d464233342d374132382d33453142323046384639443321001 4

Apdorotas paketas JSON formatu:


```
{
   "source_ip": "192.168.0.216",
   "dest_ip": "192.168.0.11",
   "source_port": 1883,
   "dest_port": 22489,
   "timestamp": 1762630285.4398313,
   "packet_type": "CONNACK",
   "fixed_header_data": {
      "dup_flag": false,
      "qos_level": 0,
      "retain_flag": false,
      "remaining_length": 53,
      "fixed_header_hex": "2035"
   },
```

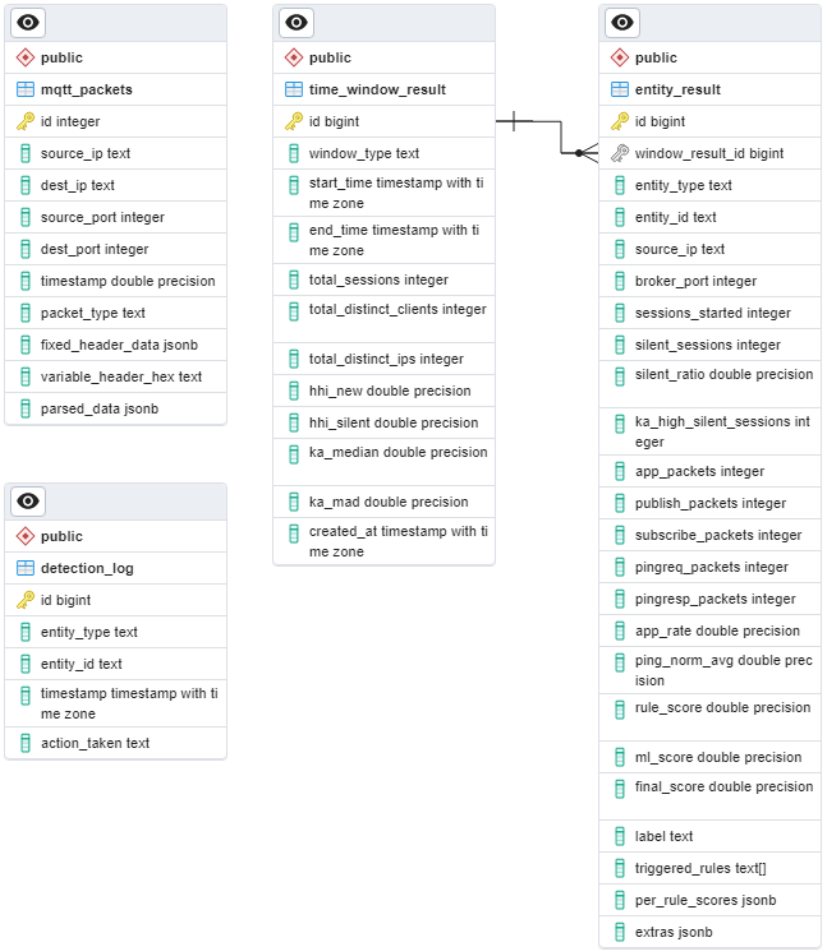
```
  "variable_header_hex":
"00003222000a1200296175746f2d323743333745384 22d433532362d464233342d374132382d3345314232304638463
94433210014",
  "parsed_data": {
    "session_present": false,
    "reason_code": 0,
    "assigned_client_identifier": "auto-27C37E8B-C526-FB34-7A28-3E1B20F8F9D3"
  }
}
```


Išsaugoti paketai duomenų bazėje pavyzdys (žr. **17 pav.**):

19	172.29.103.2...	192.168.0.11	58266	1883	1762804081.7157815	SUBSCRIBE	{"dup_flag": false, "qos_level": 1, "retain_flag": false, "fixed_header_hex": "820d", "remaining_length...	000200
20	192.168.0.11	172.29.103.2...	1883	58278	1762804081.7182138	CONNACK	{"dup_flag": false, "qos_level": 0, "retain_flag": false, "fixed_header_hex": "2009", "remaining_length...	00000622000a210014
21	172.29.103.2...	192.168.0.11	58278	1883	1762804081.7203462	SUBSCRIBE	{"dup_flag": false, "qos_level": 1, "retain_flag": false, "fixed_header_hex": "820d", "remaining_length...	000200
22	172.29.103.2...	192.168.0.11	58264	1883	1762804113.1235566	PINGREQ	{"dup_flag": false, "qos_level": 0, "retain_flag": false, "fixed_header_hex": "c000", "remaining_length...	
23	172.29.103.2...	192.168.0.11	58252	1883	1762804113.1253488	PINGREQ	{"dup_flag": false, "qos_level": 0, "retain_flag": false, "fixed_header_hex": "c000", "remaining_length...	

17 pav. Paketų išsaugojimas duomenų bazėje pavyzdys

3.4. Duomenų bazės realizacija



18 pav. Duomenų bazės lentelės

18 paveikslėlyje galime matyti naudojamos duomenų bazės klasių diagramą, kuri bus realizuota naudojant *PostgreSQL* duomenų bazę. *PostgreSQL* duomenų buvo pasirinkta dėl galimybės palaikyti JSON duomenų tipo, aukšto našumo ir galimybės efektyviai apdoroti didelius duomenų srautus.

- *mqtt_packets* lentelėje saugomi visi nuskaityti MQTT protokolo paketai, gauti iš paketų skanavimo modulio.
- *time_window_result* lentelė saugo apibendrintus atributus pagal skirtingus slenkančio laiko intervalus (10 minučių, 1 valandos ir 3 valandų).
- *entity_result* lentelėje saugomi kiekvieno MQTT kliento analizės rezultatai, apskaičiuoti pagal atskirus laiko langus.
- *detection_log* lentelė registruoja visus aptikimo modulių sprendimus ir sistemos reakcijas.

3.5. Lėtų DoS atakų aptikimo realizacija

Šio modulio realizacijai buvo pasirinkta naudoti *Python* programavimo kalbą su šiomis pagrindinėmis bibliotekomis, kurios prideda reikiamą funkcionalumą realizuoti metodą:

- ***logging*:** naudojamas programos įvykių registravimui, siekiant palengvinti programos veikimo stebėjimą;
- ***time*:** leidžia atlikti veiksmus, susijusius su laiku, pvz.: laikinai pristabdyti programos vykdymą;
- ***dataclasses*:** suteikia patogų būdą kurti klases duomenų saugojimui;
- ***datetime*:** naudojamas datų ir laiko objektams kurti, apdoroti bei formuoti, leidžiantis atlikti tikslias laiko operacijas;
- ***typing*:** naudojamas tipų anotacijoms nurodyti, siekiant padidinti kodo aiškumą ir patikimumą;
- ***psycopg*:** *PostgreSQL* duomenų bazės tvarkyklė;
- ***os*:** suteikia prieigą prie operacinės sistemos funkcijų;
- ***collections*:** pateikia papildomus duomenų struktūrų tipus, kurie papildo standartines *Python* kolekcijas;
- ***json*:** leidžia konvertuoti duomenis tarp *Python* objektų ir JSON formato;
- ***sys*:** leidžia pasiekti programos argumentus, išėjimą ir interpretatoriaus nustatymus;
- ***dotenv*:** įkelia konfigūraciją iš .env failo į aplinkos kintamuosius;
- ***pickle*:** leidžia serializuoti ir deserializuoti *Python* objektus į failus;
- ***pathlib*:** patogus ir saugesnis darbas su failų ir katalogų keliais;
- ***sklearn (IsolationForest)*:** naudojamas mašininio mokymosi anomalijų aptikimui modeliuojant tinklo paketų elgesį.

Šiam modulio realizacijai buvo pasirinkta naudoti objektinio programavimo principus su tokia failų struktūra:

```
MQTT_ANALYZIS
├── constants.py
├── main.py
├── metrics.py
├── ml.py
├── packets.py
├── resolver.py
├── schema.py
├── service.py
├── settings.py
```

└── utils.py

Paleidus modulį pirmiausia yra patikrinama, ar duomenų bazėje visos naudojamos lentelės egzistuoja, tokiu atveju, jeigu jos neegzistuoja, jos yra sukuriamos. Tuomet gaunami paskutiniai laiko intervalai kiekvienam laiko langui – tokiu atveju, jeigu jų nėra, yra pradedama generuoti laiko intervalai nuo pirmo išsaugoto paketo kiekvienam intervalui. Kai ateina laikas naujam laiko intervalui yra nuskaitomi visi paketai tame laiko intervale. Tuomet kiekvienam paketui yra priskiriamas klientas pagal jo *source_ip* ir *source_port*, nes kliento identifikacijos numerį (angl. *id*) turi tik *CONNECT* arba *CONNACK* paketas. Kai jau turime visų paketų klientus, jie yra suskirstomi į grupes pagal klientų identifikaciją ir apskaičiuojamos visos taisyklės, kurios aprašytos **2.7** skyriuje ir palyginamos su prieš tai vykusiais laiko intervalais. Kai apskaičiuojame visas taisykles, patikriname, ar vyksta mašininio mokymosi procesas ar jau sprendimų priėmimas. Kol vyksta mašininio mokymosi apmokymas, *ml_score* yra nustatomas į *null*, priešingu atveju yra gaunama reali mašininio mokymosi gauta reikšmė. Tuomet sudedame *rule_score* ir *ml_score* pagal **2.9** skyriuje esančią 2 formulę ir išsaugome gautus duomenis į duomenų bazę *entity_result* lentelę.

Aptikus kenksmingą klientą taip pat yra išsaugoma ir į *detection_log* lentelę kenksmingo kliento informacija, su gautų įverčių reikšmėmis.

Atlikus modulio realizaciją algoritmas buvo paleistas pirmiausiai apmokyti mašininio mokymosi modelį naudojant 300 000 MQTT paketų. Kitas žingsnis prie replikuojančių realių MQTT klientų prijungti kenksmingus MQTT klientus ir nustatyti, kaip algoritmas veikia ir ar atpažįsta kenksmingus klientus. Kiekvienam laiko intervalui yra išspausdinama kiekvieno kliento rezultatai, kurie sudaro kliento *id*, sprendimą, aktyvuotų taisyklių rezultatą, aktyvuotas taisykles, mašininio algoritmo rezultatus ir galutinį rezultatą **19 pav.**. Taip pat aptikti kenkėjai yra išsaugojami duomenų bazėje **20 pav.**.

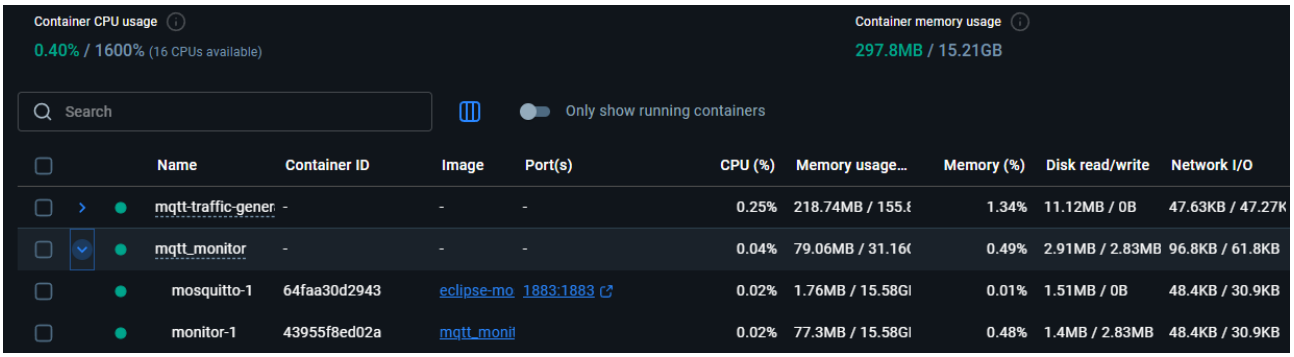
```
=== Window 10m [2026-01-18T19:00:00+00:00 – 2026-01-18T19:10:00+00:00] | sessions=12 | ips=5 | clients=12 |
client:ccfMIzgsAmvgQ_156 | label=normal | rule_score=0.000 | rules=- | ml_score=0.000 | final_score=0.000
client:67G0oYzl15yZ6_311 | label=normal | rule_score=0.000 | rules=- | ml_score=0.000 | final_score=0.000
client:Ftd5DUeydvexx_156 | label=normal | rule_score=0.000 | rules=- | ml_score=0.000 | final_score=0.000
client:9CLw5xkHMY4U4_156 | label=normal | rule_score=0.000 | rules=- | ml_score=0.000 | final_score=0.000
client:67G0oYzl15yZ6_312 | label=normal | rule_score=0.000 | rules=- | ml_score=0.000 | final_score=0.000
client:loadtest-0-cizrmn | label=attacker | rule_score=0.333 | rules=- | ml_score=1.000 | final_score=0.667
client:loadtest-1-cul1hg | label=attacker | rule_score=0.333 | rules=- | ml_score=1.000 | final_score=0.667
client:loadtest-2-353rse | label=attacker | rule_score=0.333 | rules=- | ml_score=1.000 | final_score=0.667
client:BY9h1aTUK5op5_105 | label=normal | rule_score=0.000 | rules=- | ml_score=0.000 | final_score=0.000
client:loadtest-3-pqdof2 | label=attacker | rule_score=0.333 | rules=- | ml_score=1.000 | final_score=0.667
client:loadtest-4-48mhir | label=attacker | rule_score=0.333 | rules=- | ml_score=1.000 | final_score=0.667
client:loadtest-0-inpow0 | label=attacker | rule_score=0.333 | rules=- | ml_score=1.000 | final_score=0.667
```

19 pav. Laiko intervalo rezultatų pavyzdys

	id [PK] bigint	entity_type text	entity_id text	timestamp timestamp with time zone	action_taken text
1	1123	client	loadtest-9-c4j0ve	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
2	1122	client	loadtest-8-730bk5	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
3	1121	client	loadtest-7-lac98c	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
4	1120	client	loadtest-6-pdeqyd	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
5	1119	client	loadtest-5-od9e2x	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
6	1118	client	loadtest-4-3kglqc	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
7	1117	client	loadtest-3-c1u3gm	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
8	1116	client	loadtest-2-9j8u26	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
9	1115	client	loadtest-1-r0i5hc	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m
10	1114	client	loadtest-0-ba5n9u	2026-01-19 21:15:38.31248+02	final_score=0.667 rule_score=0.333 ml_score=1.000 window=10m

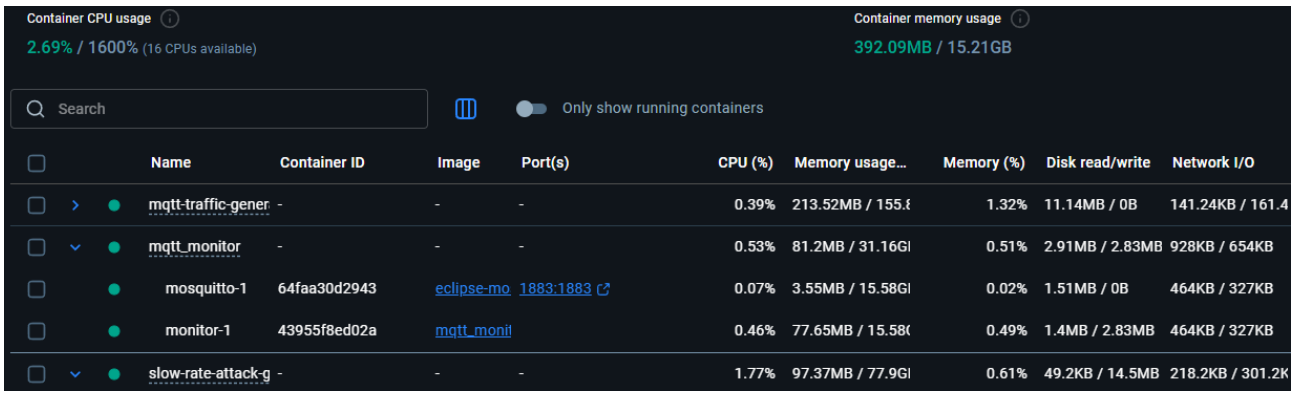
20 pav. Aptikti kenksmingi MQTT klientai duomenų bazėje

Paleidus visus konteinerius (MQTT tarpininką, tinklo skaitymą ir realius klientus) buvo nustatyta, kad visi konteineriai išnaudoja 297,8 MB kompiuterio operatyviosios atminties ir 0,40 % kompiuterio procesoriaus resursų. Tačiau didžioji dalis sąnaudų buvo išnaudota konteinerių, kuriuose simuliuojame klientus, o paties tarpininko sąnaudos siekė tik 1,76 MB operatyviosios atminties ir tik 0,01 % procesoriaus resursų. Šias sąnaudas galime matyti *Docker* programinės įrangos lange, atvaizduojant realiu laiku **21 pav.**.



21 pav. Kompiuterio resursų sąnaudos tik su realiais klientais

Tuo pačiu metu paleidus ir kenksmingus klientus sunaudojami resursai išaugo tiek visų konteinerių iki 392,09 MB operatyviosios atminties ir 2,69 % procesoriaus resursų **22 pav.**. Žinoma, paties tarpininko sąnaudos taip pat buvo tos, kurios pakilo iki 3,55 MB operatyviosios atminties ir 0,02 % procesoriaus, tačiau reikia atsižvelgti kad tarpininko resursai išaugo 2 kartus. Tačiau klientų kiekis, išaugęs daugiau nei 18 kartų nuo apytiksliai 55 klientų iki 1024 klientų, buvo nustatytas kaip maksimalus kiekis vienu metu aktyvių jungčių. Kitas parametras, į kurį galime atkreipti dėmesį, yra informacijos siuntimas (angl. *Output*) arba gavimas (angl. *Input*) naudojant tinklo sąsają (angl. *Network I/O*). Šio parametro parametrai padidėjo apytiksliai 10 kartų nuo 48,4 KB iki 464 KB gavimo ir nuo 30,9 KB iki 327 KB siuntimo. Gavimo parametras yra didesnis vien dėl to, kad tarpininkas gauna daugiau paketų negu išsiunčia.



22 pav. Kompiuterio resursų sąnaudos su kenksmingais klientais

Pasiekus maksimalų kiekį leistinų jungčių tarpininke tarpininkas daugiau nepriima jokių naujų jungčių (žr. **23 pav.**) ir grąžina pranešimą, kad nauja jungtis yra nepriimta dėl pasiekto limito. Tik seniau prisijungę klientai gali siųsti paketus, tačiau, jeigu jie atsijungs jų vietą užims kenksmingas klientas, nes jie bando vis iš naujo prisijungti. Taip gali susidaryti situacija, kai visas jungtis per tam tikrą laiką užims kenksmingi klientai. Žinoma, kadangi mes išsaugome tam tikrus paketus duomenų bazėje, galime matyti *CONNECT* paketų laviną, nes kenksmingi klientai bando be pertraukos prisijungti iš naujo.

```
2026-03-09T15:37:24: Client connection from 172.20.0.16 denied: max_connections exceeded.
2026-03-09T15:37:24: Client connection from 172.20.0.16 denied: max_connections exceeded.
2026-03-09T15:37:24: Client connection from 172.20.0.16 denied: max_connections exceeded.
2026-03-09T15:37:24: Client connection from 172.20.0.16 denied: max_connections exceeded.
2026-03-09T15:37:24: Client connection from 172.20.0.15 denied: max_connections exceeded.
2026-03-09T15:37:24: Received PUBLISH from OdyztxlfvGRgO (d0, q1, r0, m2288, 'plant/line1/power', ... (11 bytes))
2026-03-09T15:37:24: Sending PUBLISH to client_subscriber_ejHEN0mSl39t2 (d0, q1, r0, m8305, 'plant/line1/power', ... (11 bytes))
2026-03-09T15:37:24: Sending PUBACK to OdyztxlfvGRgO (m2288, rc0)
2026-03-09T15:37:24: Received PUBACK from client_subscriber_ejHEN0mSl39t2 (Mid: 8305, RC:0)
2026-03-09T15:37:24: Client connection from 172.20.0.15 denied: max_connections exceeded.
```

23 pav. MQTT tarpininko pranešimai pasiekus maksimalų kiekį klientų

3.6. Realizacijos dalies išvados

Sukūrus lėtų DoS atakų atpažinimo MQTT protokole metodo realizaciją galima padaryti šias išvadas:

1. Norint generuoti duomenis ir juos rinkti priimtas sprendimas naudoti *Docker* konteinerius palengvino MQTT klientų paleidimą ir jų įvairovę;
2. 300 000 MQTT paketų, simuliuojančių realius prietaisus, kiekis pakankamas apmokyti mašininio mokymosi algoritmą, kad būtų aptikti kenksmingi klientai;
3. Realizuota duomenų bazė palengvina duomenų analizavimą ir prireikus ištrinti tam tikrus duomenis, kad patobulintume algoritmą, būtų galima pakartoti skaičiavimus su tais pačiais paketais;
4. Realizuotas lėtas DoS atakas simuliuojanti programa palengvina tikrinimą vertinant atakų aptikimo algoritmą su skirtingais duomenimis;
5. Atakų aptikimo modulis sėkmingai aptinka kenksmingus MQTT klientus ir juos išsaugo duomenų bazėje.

4. Lėtų atakų MQTT protokolu veikiančiame daiktų internete aptikimo metodo eksperimentai

Skyriuje, skirtame šiam eksperimentui yra aprašomi įvykdyti eksperimentai keičiant tam tikrus metodo atributus ir jų svorį.

Prieš vykdant eksperimentą reikia atsižvelgti, kad eksperimento metu buvo iš viso:

Realių klientų: 1446

Kenksmingų klientų: 990

4.1. Kenksmingų klientų klasifikavimo slenksčio nustatymo eksperimentas

Šiame poskyryje atliekama sistemos jautrumo analizė, varijuojant kenksmingo kliento nustatymo slenkstį (angl. *detection threshold*) intervale nuo 0,5 iki 1,0, taikant 0,1 žingsnį. Šis slenkstis yra vienas iš pagrindinių sistemos parametrų, nustatant aiškią ribą tarp normalaus elgsenos modelio ir klasifikavimo kaip kenksmingo kliento. Galutinis kliento įvertinimas, kuris yra apskaičiuojamas pagal (2) formulę, yra lyginimas su šia riba, viršijus slenkstį, klientas identifikuojamas kaip užpuolikas.

Šio slenksčio parinkimas lemia sistemos klaidų pasiskirstymą, kur žemesnė riba užtikrina aukštą saugumo lygį, tačiau padidina I tipo klaidų (angl. *False Positives*) tikimybę, kai dėl nedidelių elgsenos nuokrypių realūs MQTT protokolo klientai klasifikuojami kaip grėsmingi. Kitu atveju aukštas slenkstis padidina sistemos toleranciją, tačiau sukuria galimybę II tipo klaidoms (angl. *False Negative*), kai lėtos atakos, pasižyminčios mažu intensyvumu, lieka nepastebėtos. Eksperimento tikslas yra surasti optimalią reikšmę, minimizuojančią abiejų tipų klaidas ir užtikrinančią didžiausią F1 rodiklio reikšmę. Būtent šis rodiklis padeda surasti geriausią reikšmę, kai sistema ne tik patikimai pagauna kenksmingus klientus, bet ir be reikalo netrukdo realiems daiktų interneto įrenginiams.

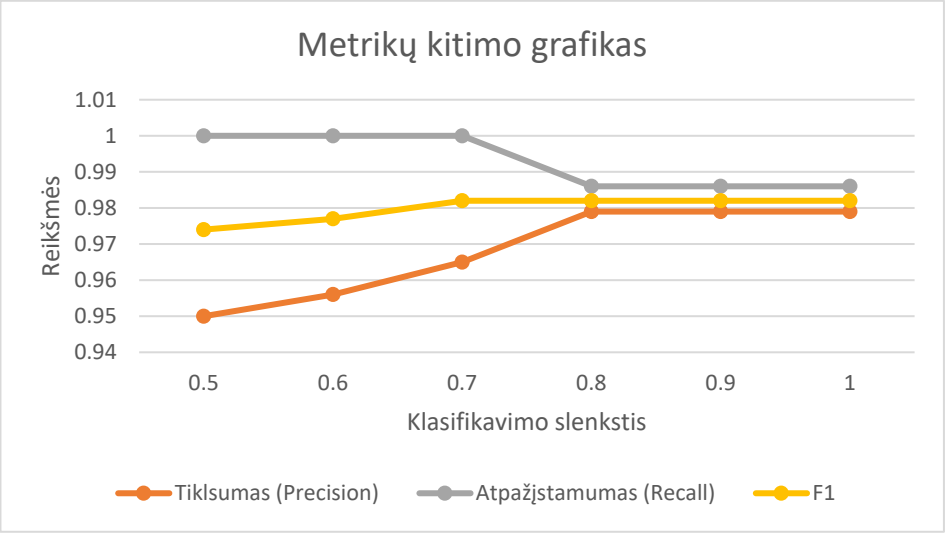
Atliktus eksperimentą iš **4.1 lentelės**, **24 pav.** ir **25 pav.** grafikų galima matyti, kad geriausi rezultatai gaunami tarp 0,7 ir 0,8 slenksčių. Kritinis taškas, ties kuriuo galime pastebėti didžiausius pokyčius, yra 0,8, iki šios ribos sistema išlaiko maksimalų jautrumą (Atpažįstamumas = 1), tačiau, peržengus šią ribą, atsiranda praleistų atakų (FN = 14), nors sistemos specifiškumas ir padidėja. Kitas parametras yra F1, kuris nuo 0,7 reikšmės nustoja kilti ir lieka ties 0,982, kas indikuoja sistemos atsparumą nedideliems parametrų pokyčiams. Kadangi specifiškumas visais slenksčiais išlieka aukštas, galime teigti, kad pasirinktas hibridinis metodas užtikrina minimalią tikimybę klasifikuoti realų klientą kaip kenksmingą.

4.1 lentelė. Bendri slenksčių rezultatai

Slenkstis	**TP**	**FP**	**FN**	**TN**	**Tikslumas (angl. *Precision*)**	**Atpažįstamumas (angl. *Recall*)**	**F1**	**Bendras Tikslumas (angl. *Accuracy*)**	**Specifiškumas (angl. *Specificity*)**
0.5	990	52	0	1394	0.95	1	0.974	0.979	0.964
0.6	990	46	0	1400	0.956	1	0.977	0.981	0.968
0.7	990	36	0	1410	0.965	1	0.982	0.985	0.975
0.8	976	21	14	1425	0.979	0.986	0.982	0.986	0.985

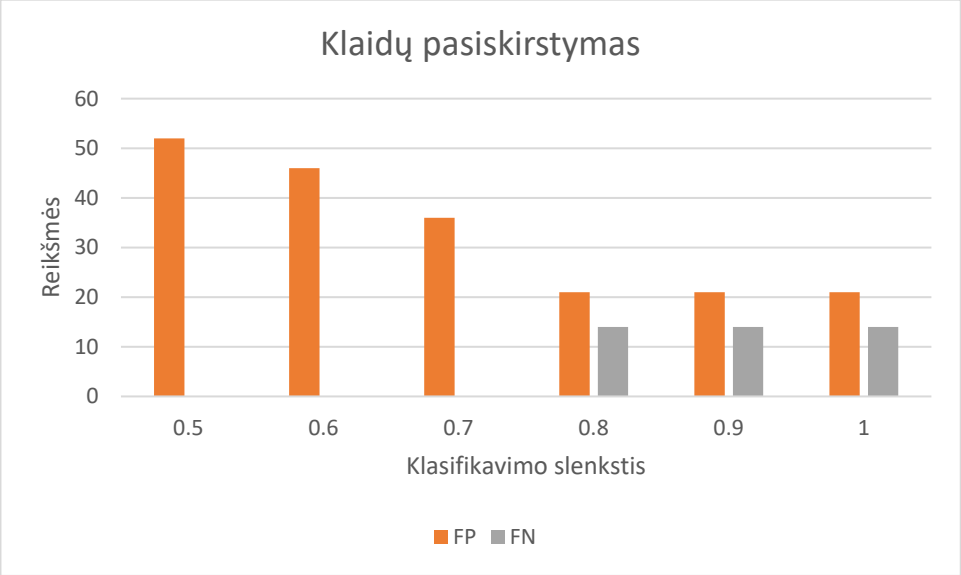
0.9	976	21	14	1425	0.979	0.986	0.982	0.986	0.985
1	976	21	14	1425	0.979	0.986	0.982	0.986	0.985

Grafike **24 pav.** galima matyti tris pagrindines reikšmes, kurios padeda nuspręsti tinkamiausią kritinį slenkstį: Tikslumas, Atpažįstamumas ir F1.



24 pav. Pagrindinių reikšmių pokytis pagal slenksčius

Grafike **25 pav.** galima matyti, kad nors ties 0,7 slenksčiu realių klientų klasifikuojam daugiau kaip kenksmingų, tačiau nepraleidžiam jokių kenksmingų klientų. Tuo atveju, kai slenkstis pakyla iki 0,8, pradedame kenksmingus klientus praleisti kaip gerus, nors ir sumažėja FP kiekis.



25 pav. Klaidų pasiskirstymas pagal slenksčius

Iš **4.2 lentelės** galima nuspręsti, kuris slenkstis yra tinkamiausias, tačiau kadangi realizuotas metodas veikia per kelis laiko intervalus, reikia atsižvelgt ir į pačių intervalų gautus rezultatus. **4.2 lentelėje** galime matyti 0,7 slenksčio skirtingų langų rezultatus – nors didėjant laiko intervalui rezultatai

prastėja ir aptinkame mažiau kenksmingų klientų, tačiau taip pat gauname mažiau ir FP klientų. Todėl vis tiek naudinga turėti kelis laiko intervalus.

4.2 lentelė. 0,7 Slenksčio langų rezultatai

Slenkstis	**TP**	**FP**	**FN**	**TN**	**Tikslumas (angl. *Precision*)**	**Atpažįstamumas (angl. *Recall*)**	**F1**	**Bendras Tikslumas (angl. *Accuracy*)**	**Specifiškumas (angl. *Specificity*)**
10m	990	29	0	1417	0.972	1	0.986	0.988	0.98
1h	868	22	122	1424	0.975	0.877	0.923	0.941	0.985
3h	608	15	382	1431	0.976	0.614	0.754	0.837	0.99
Bendras	990	36	0	1410	0.965	1	0.982	0.985	0.975

Šio eksperimento metu nustatėme, kad tinkamiausia slenksčio reikšmė yra 0,7 todėl kitiems eksperimentams ši reikšmė ir bus pritaikyta.

4.2. Alfa svorio variacijų poveikis klasifikavimo rodikliams

Kito eksperimento pagrindinis tikslas yra nustatyti optimalią hibridinio algoritmo svorio koeficiento α reikšmę, kuri lemia galutinio kliento rezultatą pagal (2) formulę. Koeficiento α dydis apibrėžia kiekvieno komponento įtaką galutiniam kliento elgsenos vertinimui. Didėjant koeficiento reikšmei, didesnis svoris suteikiamas taisyklėmis paremtam aptikimui, kuris pasižymi didesniu aiškumu, tačiau yra mažiau lankstus kintant atakų pobūdžiui. Mažėjant reikšmei, prioritetas teikiamas izoliuoto miško modelio generuojamam anomalijos balui, kuris geba identifikuoti sudėtingesnius, netiesioginius ryšius tarp parametrų, kuriuos naudojame.

Eksperimento metu α koeficientas varijuojamas intervale nuo 0,0 iki 1,0 taikant 0,2 žingsnį, išskyrus 0,5, tarp kurio buvo taikomas 0,1 žingsnis. Toks intervalo žingsnis pasirinktas gauti optimaliausią hibridinio veikimo tašką. Taip siekiama įrodyti, kad naudojamas hibridinis metodas pranašesnis už vieną metodą.

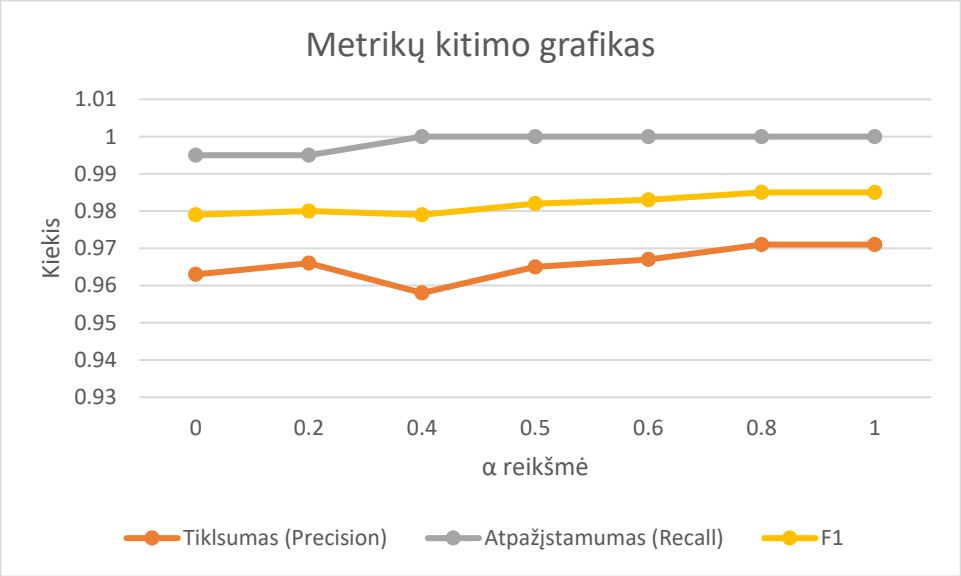
Atlikus eksperimentą iš **4.3 lentelės**, **26 pav.** ir **27 pav.** grafikų galima nustatyti, kad optimaliausias rezultatas gaunamas naudojant 0,8 α koeficientą, kuris reiškia, kad metodas pagrinde remiasi taisyklėmis paremtu aptikimu, tačiau išlieka lankstus ir gali atpažinti naujas anomalijas, kurios nėra apibrėžtos taisyklėmis. Nors nuo 0,4 reikšmės aptinkame visus kenksmingus klientus, vis tiek svarbu kuo mažiau klasifikuoti klaidingai realius klientus, o ties 0,8 FP nustoja mažėti ir sustoja ties 30.

4.3 lentelė. Bendri α koeficiento rezultatai

Slenkstis	**TP**	**FP**	**FN**	**TN**	**Tikslumas (angl. *Precision*)**	**Atpažįstamumas (angl. *Recall*)**	**F1**	**Bendras Tikslumas (angl. *Accuracy*)**	**Specifiškumas (angl. *Specificity*)**
0	985	38	5	1408	0.963	0.995	0.979	0.982	0.974
0.2	985	35	5	1411	0.966	0.995	0.98	0.984	0.976
0.4	990	43	0	1403	0.958	1	0.979	0.982	0.97
0.5	990	36	0	1410	0.965	1	0.982	0.985	0.975
0.6	990	34	0	1412	0.967	1	0.983	0.986	0.976

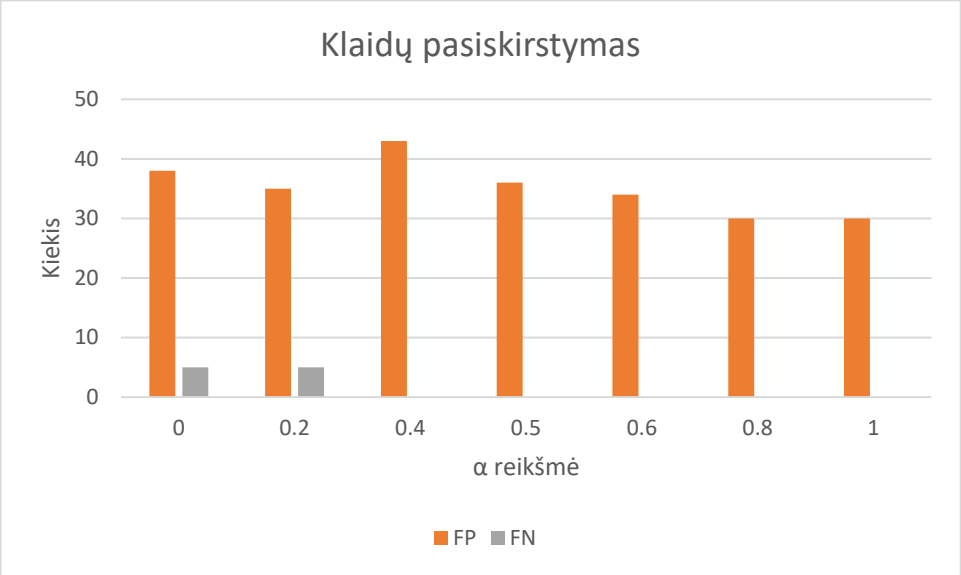
0.8	990	30	0	1416	0.971	1	0.985	0.988	0.979
1	990	30	0	1416	0.971	1	0.985	0.988	0.979

Grafike **26 pav.** galima matyti tris pagrindines reikšmes, kurios padeda nuspręsti tinkamiausią kritinį slenkstį: Tikslumas, Atpažįstamumas ir F1. Atpažįstamumas pasiekia maksimalią reikšmę 1 ties 0,4 koeficientu, tačiau kiti du argumentai didinant taisyklių aptikimu paremtą aptikimą didėja, nors ir nedaug.



26 pav. Pagrindinių reikšmių pokytis pagal α koeficientą

Grafike **27 pav.** galima matyti, kad ties 0,4 α koeficientu metodas visus kenksmingus klientus klasifikuoja taisyklingai, tačiau realių klientų klasifikavimas išlieka vis tiek aukštas. Kai pasiekiame 0,8 koeficientą, sistema nustoja neteisingai klasifikuoti realius klientus.



27 pav. Klaidų pasiskirstymas pagal α koeficientą

Iš **4.4 lentelės** galima matyti 0,8 α koeficiento skirtingų langų rezultatus. Lyginant šiuos rezultatus su **4.2 lentelės** gautais rezultatais galima matyti, kad TP reikšmė visuose laiko intervaluose išliko aukšta, kai **4.2 lentelėje** su kiekvienu laiko intervalu TP prastėjo. Kita reikšmė, kuri minimaliai sumažėjo buvo FP iš 36 iki 30 taip sumažinant klaidingai nustatytų realių klientų. Tarp likusių reikšmių galima irgi pastebėti minimalius pakilimus taip gerinant metodo veikimą.

4.4 lentelė. 0,8 α koeficiento langų rezultatai

Langas	**TP**	**FP**	**FN**	**TN**	**Tikslumas (angl. *Precision*)**	**Atpažįstamumas (angl. *Recall*)**	**F1**	**Bendras Tikslumas (angl. *Accuracy*)**	**Specifiškumas (angl. *Specificity*)**
10 min.	990	29	0	1417	0.972	1	0.986	0.988	0.98
1 val.	990	29	0	1417	0.972	1	0.986	0.988	0.98
3 val.	990	29	0	1417	0.972	1	0.986	0.988	0.98
Bendras	990	30	0	1416	0.971	1	0.985	0.988	0.979

Šio eksperimento metu nustatėme, kad tinkamiausias α koeficientas yra 0,8, kai sistema pagrinde remiasi taisyklėmis paremtu aptikimu, tačiau vis tiek atsižvelgia į mašininio mokymosi modelio rezultatus. Ši reikšmė bus naudojama kitiems eksperimentams.

4.3. Izoliuoto miško parametrų įvertinimas

Šiame poskyryje pagrindinis dėmesys kreipiamas į metodo mašininio modelio dalies optimizavimą ir skirtumus naudojant skirtingas reikšmes. Tam pasiekti nurodoma ta pati pirmame eksperimente (**4.1**) nurodytas kritinis slenkstis 0,7. Nors praeitame skyriuje nustatėme, kad optimaliausia α koeficiento reikšmė yra 0,8, kuri reiškia, kad mašininio mokymosi dalis sudaro tik 20 % galutinio sprendimo, šiame skyriuje ši reikšmė nustatyta ties 1.0, kad taisyklėmis paremtas aptikimas neturėtų jokio svorio galutiniam rezultatui.

Izoliuoto miško modelis pagrinde remiasi dviem parametrais: užterštumu (angl. *contamination*) ir medžių kiekiu, kurie šiame eksperimente ir bus keičiami. Užterštumo parametras svarbus tada, kai turimas duomenų modelis, naudojamas modelio apmokymui, turi netinkamų įrašų. Kitas parametras yra medžių kiekis, kuris padeda stabilizuoti gaunamus rezultatus. Kiti parametrai, kurie buvo naudojami, tačiau šio eksperimento metu nekeisti buvo atsitiktinė būsena (angl. *random state*), kuris skirtas atkartoti tuos pačius rezultatus testuojant modelį, nes priešingu atveju gauti rezultatai tame pačiame modelyje skirtųsi. Paskutinis parametras buvo skirtas lygiagrečiam skaičiavimui kontroliuoti, kurį nustatėme kaip -1, kad naudotų visus kompiuterio procesorius taip pagreitinant visų medžių generavimą.

Užterštumui naudotos reikšmės buvo: 0,01; 0,03; 0,05; 0,07; 0,1;

Medžių kiekio naudotos reikšmės buvo: 50; 100; 250; 500;

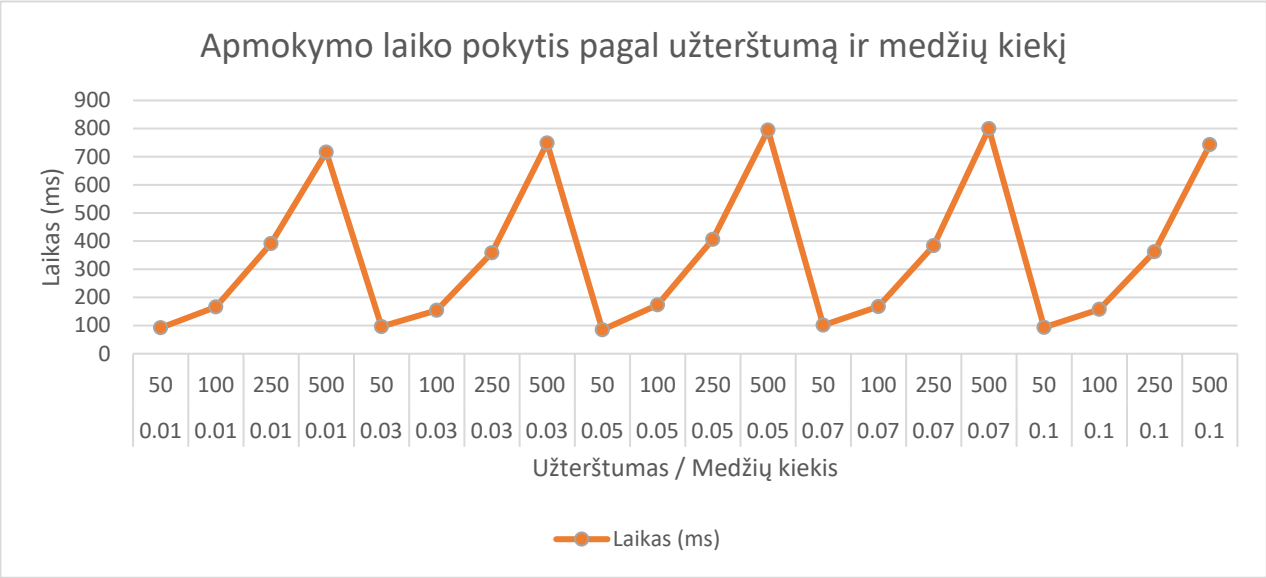
Visų modelių apmokymui buvo naudojami 16 345 įrašai.

Atlikus eksperimentą iš **4.5 lentelės** ir **28 pav.** galima matyti, kad keičiantis užterštumo reikšmei gautas apmokymo laikas minimaliai padidėja, tačiau didžiausią laiko šuolį galima matyti keičiant medžių kiekį, kuris apytiksliai dvigubai didėjo su kiekvienu pasikeitimu.

4.5 lentelė. Mašininio modelio apmokymo trukmė pagal skirtingas reikšmes

Užterštumas	Medžių kiekis	Laikas (ms.)
0.01	50	92.16
0.01	100	166.15
0.01	250	391.46
0.01	500	716.52
0.03	50	96.32
0.03	100	154
0.03	250	358.66
0.03	500	748.66
0.05	50	84.79
0.05	100	173.43
0.05	250	406.4
0.05	500	794.77
0.07	50	101.29
0.07	100	167.62
0.07	250	383.92
0.07	500	800.03
0.1	50	93.54
0.1	100	157.67
0.1	250	362.66
0.1	500	742.97

Gautus **4.5 lentelės** rezultatus buvo nuspręsta atvaizduoti naudojant linijinį grafiką, kad keičiantis parametrams ryškiau matytųsi pokytis, kurį galime matyti **28 pav.** grafike.



28 pav. Mašininio modelio apmokymo trukmė pagal skirtingas reikšmes

Atlikus eksperimentą su modelių apmokymo trukme svarbiausia išsiaiškinti, ar sugeneruotų modelių grąžinami rezultatai skiriasi ar išlieka panašūs. Iš **4.6 lentelės** galima matyti, kad visų parametrų rezultatai išlieka identiški išskyrus pasirinkus 250 medžių kiekius, kai per viena padidėja FP visuose užterštumo variantuose.

4.6 lentelė. Mašininio modelio rezultatai pagal skirtingus parametrus

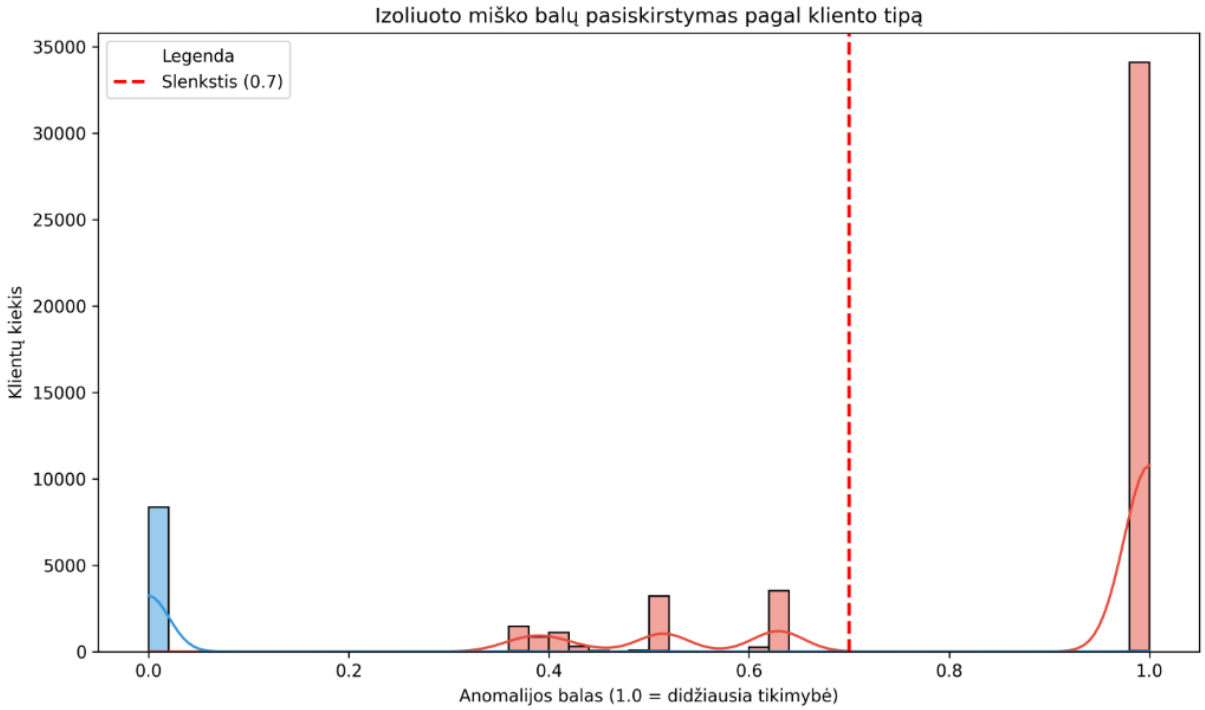
Užterštumas	**Medžių kiekis**	**TP**	**FP**	**FN**	**TN**	**Tikslumas (angl. *Precision*)**	**Atpažįstamumas (angl. *Recall*)**	**F1**	**Bendras Tikslumas (angl. *Accuracy*)**
0.01	50	990	44	0	1401	0.9574	1	0.9783	0.9696
0.01	100	990	44	0	1401	0.9574	1	0.9783	0.9696
0.01	250	990	45	0	1400	0.9565	1	0.9778	0.9689
0.01	500	990	44	0	1401	0.9574	1	0.9783	0.9696
0.03	50	990	44	0	1401	0.9574	1	0.9783	0.9696
0.03	100	990	44	0	1401	0.9574	1	0.9783	0.9696
0.03	250	990	45	0	1400	0.9565	1	0.9778	0.9689
0.03	500	990	44	0	1401	0.9574	1	0.9783	0.9696
0.05	50	990	44	0	1401	0.9574	1	0.9783	0.9696
0.05	100	990	44	0	1401	0.9574	1	0.9783	0.9696
0.05	250	990	45	0	1400	0.9565	1	0.9778	0.9689
0.05	500	990	44	0	1401	0.9574	1	0.9783	0.9696
0.07	50	990	44	0	1401	0.9574	1	0.9783	0.9696
0.07	100	990	44	0	1401	0.9574	1	0.9783	0.9696
0.07	250	990	45	0	1400	0.9565	1	0.9778	0.9689
0.07	500	990	44	0	1401	0.9574	1	0.9783	0.9696
0.1	50	990	44	0	1401	0.9574	1	0.9783	0.9696
0.1	100	990	44	0	1401	0.9574	1	0.9783	0.9696
0.1	250	990	45	0	1400	0.9565	1	0.9778	0.9689
0.1	500	990	44	0	1401	0.9574	1	0.9783	0.9696

Pagal šio eksperimento metu gautus rezultatus galima teigti, kad geriausi rezultatai buvo gaunami naudojant 0,05 užterštumą ir 50 medžių kiekį, dėl greičiausio modelio apmokymo ir dėl gautų rezultatų – kurie **4.6 lentelė** beveik visose reikšmėse išlieka vienodi.

Siekiant įvertinti pasirinkto izoliuoto miško (angl. *Isolation forest*) modelio gebėjimą atskirti anomalų srautą nuo realių klientų veiklos sugeneruojama anomalijų balų pasiskirstymo histogramą, kurią galima matyti **29 pav.** grafike. Šiame grafike galima aiškiai matyti dvi izoliuotas duomenų grupes, kur mėlyna spalva indikuoja teisėtų klientų pasiskirstymą, o oranžinė identifikuoja kenksmingus klientus.

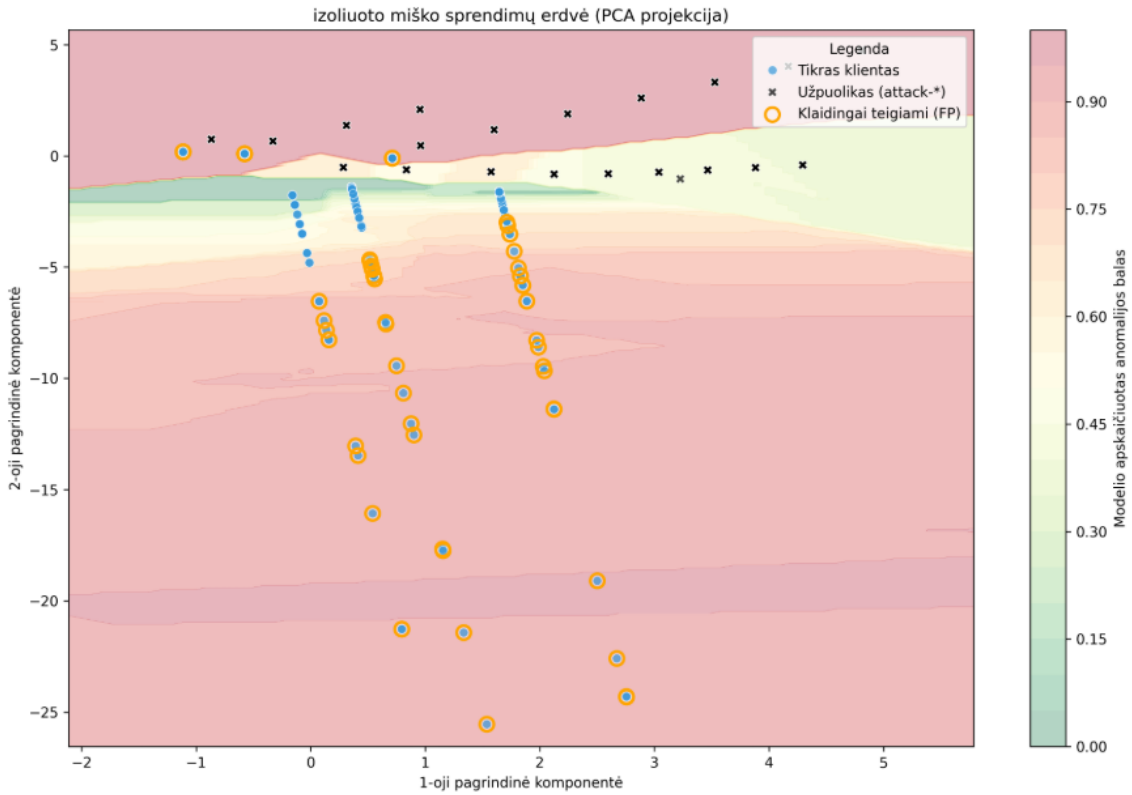
Pagal nutylėjimą Izoliuoto miško algoritmas anomalijas identifikuoja neigiamomis reikšmėmis, tačiau šiame darbe gautą rezultatą invertuojame ir normalizuojame į rėžį [0; 1], kur 1,0 reprezentuoja didžiausią anomalijos galimybę, kad gautus rezultatus būtų lengviau suprasti. Taip pat iš **29 pav.**

diagramos galima matyti, kad didžioji kenksmingų klientų dalis susikaupusi dešinėje pusėje ties 1,0 reikšme, kas reiškia tvirtai atpažintas anomalijas. Keletas klasifikuotų kenksmingų klientų lieka už 0,7 slenksčio, tačiau, jeigu slenkstį nustatytume žemesnį, bet koks nuokrypis nuo realių klientų gali tapti klasifikuojamas kaip kenksmingas. Todėl tam ir naudojame hibridinį aptikimo metodą skirtingais laiko intervalais.



29 pav. Izoliuoto miško balų pasiskirstymas

Sekančioje diagramoje (žr. **30 pav.**) pateikiama sprendimų erdvės projekcija, gauta naudojant pagrindinių komponenčių analizę. Naudojant šį metodą galime šešių dimensijų požymių lauką projektuoti į dvimatę plokštumą, išlaikant didžiausią duomenų sklaidą. Diagramoje, norint atvaizduoti sprendimų paviršių, grafiko fonas reprezentuojamas spalviniu gradientu nuo žalios iki raudonos spalvos. Raudona spalva žymi aukštą anomalijų tikimybę, o žalia normalią veiklą. Juodi kryžiukai žymi žinomus užpuolikus, o mėlyni taškai realius klientus. Taip pat šioje diagramoje galime tiksliau matyti klaidingai teigiamus (FP) rezultatus, kur realūs klientai buvo klasifikuojami kaip kenksmingi žymint oranžiniu apskritimu, nors **29 pav.** grafike to nesimatė dėl didelio kiekio duomenų.

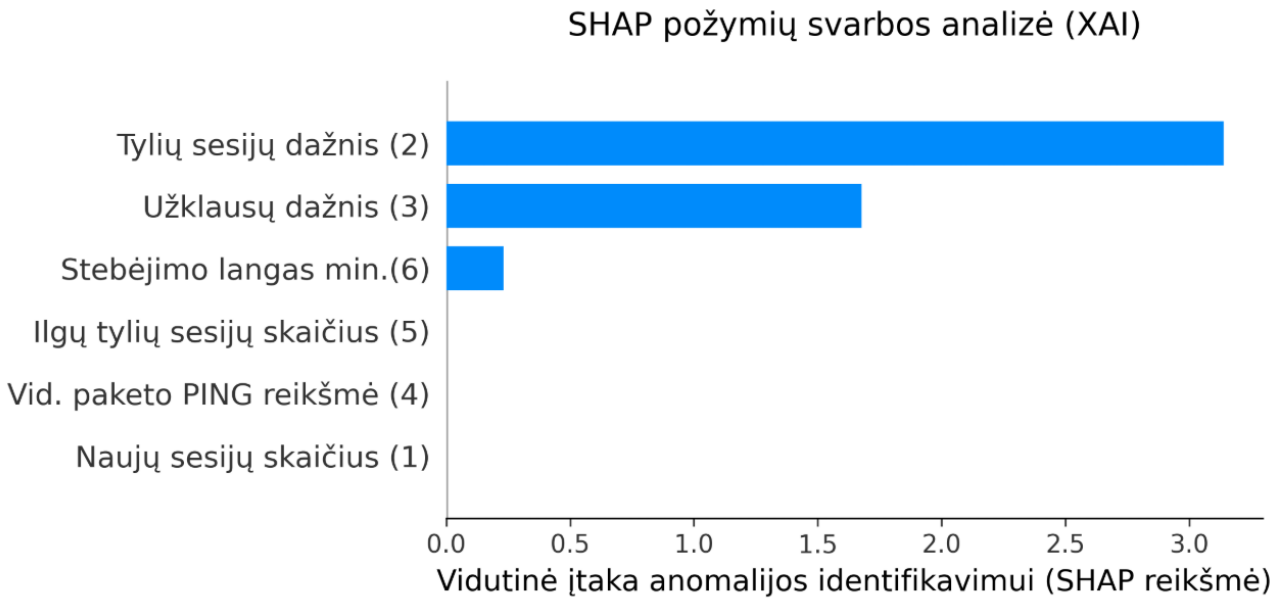


30 pav. Izoliuoto miško sprendimų erdvė

4.4. Izoliuoto miško modelio parametrų įtakos analizė naudojant SHAP metodą

Paskutinio eksperimento tikslas yra nustatyti, kurie iš naudojamų duomenų atributų turėjo didžiausią įtaką izoliuoto miško modelio veiklai ir taip siekiant suprasti anomalijų klasifikavimo logiką. Kadangi izoliuoto miško algoritmas tiesiogiai nepateikia kintamųjų svarbos įverčių, šiam tikslui pasitelkiamas SHAP (angl. *SHapley Additive exPlanations*) metodika. Tai yra žaidimų teorija pagrįstas aiškinamojo dirbtinio intelekto įrankis, kuris kiekvienam duomenų požymiui priskiria matematinę vertę, indikuojančią jo tiesioginį indėlį į galutinę anomalijos prognozę.

Iš **31 pav.** pateiktos SHAP diagramos galime matyti, kad didžiausią vaidmenį lėtųjų atakų aptikimo procese atlieka du pagrindiniai parametrai: tylių sesijų dažnis bei užklausų dažnis, o stebėjimo lango trukmė modelio sprendimams turi tik minimalią įtaką. Kiti trys stebimi požymiai: ilgų tylių sesijų skaičius, vidutinė paketo *PING* reikšmė, naujų sesijų skaičius turi nulinę arba artimą nuliui įtaką modelio rezultatams. Toks rezultatų pasiskirstymas įrodo, kad modelis geba efektyviai identifikuoti esminius lėtųjų atakų vektorius.



31 pav. SHAP požymių svarbos analizė

Atlikto tyrimo rezultatai, gauti taikant α = 0,8 bei slenkstį 0,7, rodo aukštą lėtų atakų aptikimo efektyvumą. Žiūrint iš **4.4 lentelės**, matoma, kad realizuoto metodo gautas bendras klasifikavimo tikslumas siekia 98,8 %, šį rezultatą lyginus su M. Al-Fayoumio ir Q. Abu Al-Haijos darbe [39] pristatytu geriausiu sprendimų medžiu (DTC) modeliu, pasiekusiu 99,5 % tikslumą. Lyginant šiame darbe realizuotą metodą su M. Al-Fayoumio ir Q. Abu Al-Haijos darbu svarbu paminėti, kad naudoti duomenų rinkiniai yra skirtingi, todėl gauti rezultatai gali skirtis panaudojus vienodus duomenų rinkinius. Tačiau lyginant su skirtingais duomenų rinkiniais šiame darbe realizuotas metodas pasiekia 100 % Atpažįstamumą (angl. *Recall*), kuris viršija M. Al-Fayoumio ir Q. Abu Al-Haijos darbe pasiektą 99,4 % rezultatą. Taip pat pasiektas 97,1 % preciziškumas bei 98,5 % F1 įvertis lenkia kitus minėtų autorių tirtus modelius, pavyzdžiui, daugiasluoksnio perceptrono (MLP) 91,5 % tikslumą ar dirbtinio neuroninio tinklo (ANN) 83,8 % tikslumą. Tokie rezultatai rodo, kad naudojant hibridinį lėtų atakų aptikimo metodą užtikrina aukšta patikimumą aptikti kenksmingus klientus.

4.5. Eksperimentinės dalies išvados

Atlikus realizuoto metodo eksperimentus galime daryti šias išvadas:

1. Eksperimentinio tyrimo metu patvirtinta, kad naudojant aptikimo metodus atskirai taisyklėmis paremtu gaunamas 98,8 % tikslumas, o mašininio mokymosi gaunamas 98,2 % tikslumas, naudojant hibridinį aptikimo metodą metodas išlaiko 98,8 % bendrą klasifikavimo tikslumą (angl. *Accuracy*).
2. Nustatyta, kad alfa koeficientas lygus 0,8 užtikrina optimaliausią pusiausvyrą tarp hibridinio metodo, laikant, kad taisyklėm paremtas aptikimas sudaro didžiausią svorį.
3. Naudojimas skirtingų laikų intervalai padidina realizuoto metodo tikslumą, nes jeigu viename laiko intervale kenksmingas klientas lieka nepastebėtas vis tiek išlieka didelė tikimybė būti aptiktam kitame laiko intervale.
4. Hibridinis metodas užtikrina stabilų 98,5 % F1 įvertį, kas rodo metodo robastiškumą ir tinkamumą realaus laiko MQTT 5.0 srauto analizei.

5. Išvados

1. Atlikta pažeidžiamumų analizė parodė, kad daiktų interneto tinkluose greta tradicinėms sistemoms būdingų pažeidžiamumų egzistuoja specifinei, DI įrenginiams būdingi veiksniai, tokie kaip energijos išnaudojimas ar prastas fizinis saugumas. Šių pažeidžiamumų prevencijos svarbą pagrindžia tai, kad sėkmingai realizuotos atakos gali sukelti ne tik milžiniškus finansinius nuostolius, bet ir tiesioginę grėsmę žmonių sveikatai ar gyvybei.
2. Atlikta penkių skirtingų daiktų internete naudojamų protokolų specifikacijos analizė, parodė, kad protokolo pasirinkimą lemia sistemos našumas ir funkcinis sudėtingumas. Mažų sąnaudų protokolų tokių kaip MQTT ar CoAP svarbą lemia jų pritaikomumas ribotų išteklių įrenginiams, nepaisant bazinių saugumo mechanizmų trūkumo. Tuo tarpu sudėtingesni protokolai kaip AMQP ar XMPP užtikrina didesnį duomenų perdavimo patikimumą ir saugumą, tačiau reikalauja gerokai didesnių skaičiavimo bei energetinių resursų.
3. Atlikta kibernetinių atakų priežasčių analizė MQTT protokolą naudojančiuose DI įrenginiuose, nustatyta, kad pagrindiniai veiksniai, dėl kurių protokolas tampa užpuolikų taikiniu, yra paties protokolo specifikacijos trūkumai bei nepakankamai saugi jo realizacija. Dėl šių priežasčių piktavaliai gali išnaudoti prigimtines protokolo savybes tokioms atakoms kaip lėtosios DoS atakos.
4. Remiantis nustatytais MQTT protokolo pažeidžiamumais bei jų priežastimis, sukurtas hibridinis lėtų atakų aptikimo metodas, kuris analizuodamas MQTT tinklo paketus, sujungia taisyklėmis paremtą mechanizmą ir mašininio mokymosi izoliuoto miško algoritmą. Tokiu būdu užtikrinant ir jau žinomą atakų specifikų ir anomalijų aptikimą.
5. Pagal sukurtą aptikimo metodą realizuotas įrankis, gebantis identifikuoti lėtas DoS atakas, analizuojant tinklo paketus iš duomenų bazės. Norint verifikuoti ar aptikimo metodas tinkamai veikia buvo realizuoti papildomi moduliai: MQTT tinklo srauto skanavimo ir apdorojimo modulis, realių klientų simuliavimo įrankis, bei įrankis galintis simuliuoti lėtas DoS kibernetines atakas skirtingais būdais.
6. Atliktas realizuoto prototipo eksperimentinis tyrimus, parodė, kad geriausius lėtų atakų aptikimo rezultatus užtikrina pasirinkto $\alpha = 0{,}8$ koeficiento bei 0,7 anomalijų slenksčio reikšmės. Taikant šias reikšmes pasiektas 98,8 % bendras klasifikavimo tikslumas (angl. *Accuracy*) ir 98,5 % F1 įvertis.

Literatūros sąrašas

1. ESTELLA, T. - GHAYATRIE, N.A.I. - LEVINA, A. - PRASETYO, A. IoT: What is, Concern, and How to Secure? A Systematic Literature Review. *2022 6th International Conference on Informatics and Computational Sciences (ICICoS)* [interaktyvus]. 2022. p. 24–29. [žiūrėta 2024-12-29]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/9930559?casa_token=QEHLRdI-irAAAAAA:59ko1PX4aTGdw62r4tYYbhRRkDdlkA6X4KoYCZ6r0pU2HfQnamZIm-xx8mX3MqyBPsSNTYW_L94>.

2. ANAND, P. - SINGH, Y. - SELWAL, A. - ALAZAB, M. - TANWAR, S. - KUMAR, N. IoT Vulnerability Assessment for Sustainable Computing: Threats, Current Solutions, and Open Challenges. *IEEE Access* . 2020. Vol. 8, p. 168825–168853. .

3. AZROUR, M. - MABROUKI, J. - GUEZZAZ, A. - KANWAL, A. Internet of Things Security: Challenges and Key Issues. *Security and Communication Networks* . 2021. Vol. 2021, no. 1, p. 5533843. .

4. MAO, W. - ZHAO, Z. - CHANG, Z. - MIN, G. - GAO, W. Energy-Efficient Industrial Internet of Things: Overview and Open Issues. *IEEE Transactions on Industrial Informatics* . 2021. Vol. 17, no. 11, p. 7225–7237. .

5. AKTER, S. - KHALIL, K. - BAYOUMI, M. Hardware Security in the Internet of Things: A Survey. *2023 IEEE 36th International System-on-Chip Conference (SOCC)* [interaktyvus]. 2023. p. 1–6. [žiūrėta 2024-12-30]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/10256974?casa_token=B26CAetX0nEAAAAA:AigdZw-AcXtXPlrGVTmvBVJeU9XghqKsjzbzz8vX8NW1vJHiutr-AjamUbyBuh0TKaTSU0hClfI>.

6. YANG, X. - SHU, L. - LIU, Y. - HANCKE, G.P. - FERRAG, M.A. - HUANG, K. Physical Security and Safety of IoT Equipment: A Survey of Recent Advances and Opportunities. *IEEE Transactions on Industrial Informatics* . 2022. Vol. 18, no. 7, p. 4319–4330. .

7. JAVADPOUR, A. - JA'FARI, F. - TALEB, T. - ZHAO, Y. - YANG, B. - BENZAÏD, C. Encryption as a Service for IoT: Opportunities, Challenges, and Solutions. *IEEE Internet of Things Journal* . 2024. Vol. 11, no. 5, p. 7525–7558. .

8. MERALA, N. *Defending IoT against escalating cyber threats like botnet attacks, data privacy issues and inadequate patch management capabilities* [interaktyvus]. [s.l.]: Dublin, National College of Ireland, 2022. [žiūrėta 2025-01-04]. Prieiga per internetą: <https://norma.ncirl.ie/6024/>.

9. REKHA, S. - THIRUPATHI, L. - RENIKUNTA, S. - GANGULA, R. Study of security issues and solutions in Internet of Things (IoT). *Materials Today: Proceedings* . 2023. Vol. 80, p. 3554–3559. .

10. GLAROUDIS, D. - IOSSIFIDES, A. - CHATZIMISIOS, P. Survey, comparison and research challenges of IoT application protocols for smart farming. *Computer Networks* . 2020. Vol. 168, p. 107037. .

11. AL-MASRI, E. - KALYANAM, K.R. - BATTS, J. - KIM, J. - SINGH, S. - VO, T. - YAN, C. Investigating Messaging Protocols for the Internet of Things (IoT). *IEEE Access* . 2020. Vol. 8, p. 94880–94911. .

12. BANG, A.O. - RAO, U.P. - VISCONTI, A. - BRIGHENTE, A. - CONTI, M. An IoT Inventory Before Deployment: A Survey on IoT Protocols, Communication Technologies, Vulnerabilities, Attacks, and Future Research Directions. *Computers & Security* . 2022. Vol. 123, p. 102914. .

13. MAURYA, R. Application of Restful APIs in IOT: A Review. *International Journal for Research in Applied Science and Engineering Technology* . 2021. Vol. 9, no. 2, p. 145–151. .

14. ABDULGHANI, R.M. - ALREHILI, M.M. - ALMUHANNA, A.A. - ALHAZMI, O.H. Vulnerabilities and Security Issues in IoT Protocols. *2020 First International Conference of Smart Systems and Emerging Technologies (SMARTTECH)* [interaktyvus]. 2020. p. 7–12. [žiūrėta 2024-11-29]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/9283795>.

15. LAKSHMINARAYANA, S. - PRASEED, A. - THILAGAM, P.S. Securing the IoT Application Layer From an MQTT Protocol Perspective: Challenges and Research Prospects. *IEEE Communications Surveys & Tutorials* . 2024. Vol. 26, no. 4, p. 2510–2546. .

16. BHATTACHARJYA, A. - ZHONG, X. - WANG, J. - LI, X. CoAP—Application Layer Connection-Less Lightweight Protocol for the Internet of Things (IoT) and CoAP-IPSEC Security with DTLS Supporting CoAP. FARSI, M. - DANESHKHAH, A. - HOSSEINIAN-FAR, A. - JAHANKHANI, H.Sud. *Digital Twin Technologies and Smart Cities* [interaktyvus]. Cham: Springer International Publishing, 2020. p. 151–175. [žiūrėta 2025-01-15]. ISBN 978-3-030-18732-3Prieiga per internetą: <https://doi.org/10.1007/978-3-030-18732-3_9>.

17. HARTH GHASSAN HAMID Survey on IoT application layer protocols. *ResearchGate* [interaktyvus]. [žiūrėta 2025-01-05]. Prieiga per internetą: <https://www.researchgate.net/publication/349716921_Survey_on_IoT_application_layer_protocols>.

18. TIGHTIZ, L. - YANG, H. A Comprehensive Review on IoT Protocols' Features in Smart Grid Communication. *Energies* . 2020. Vol. 13, no. 11, p. 2762. .

19. SEOANE, V. - GARCIA-RUBIO, C. - ALMENARES, F. - CAMPO, C. Performance evaluation of CoAP and MQTT with security support for IoT environments. *Computer Networks* . 2021. Vol. 197, p. 108338. .

20. SILVA, D. - CARVALHO, L.I. - SOARES, J. - SOFIA, R.C. A Performance Analysis of Internet of Things Networking Protocols: Evaluating MQTT, CoAP, OPC UA. *Applied Sciences* . 2021. Vol. 11, no. 11, p. 4879. .

21. GUPTA, P. - M, I.O.Prabha. A Survey of Application Layer Protocols for Internet of Things. *2021 International Conference on Communication information and Computing Technology (ICCICT)* [interaktyvus]. 2021. p. 1–6. [žiūrėta 2025-01-16]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/9510140?casa_token=f9DPq8RgwdQAAAAA:WJQCuhVf4nVRr4BlsGb17frlmwjQQS-jcet7hneiCjmW4woAl1bs7VcPCf0Z5SvUofcOW1Za3A>.

22. ALKWIEFI, M. - KHALIFEH, A. M2M Protocols: An Overview on LwM2M and XMPP Machine-to-Machine Protocols in IoT Context. *2024 IEEE/ACIS 24th International Conference on Computer and Information Science (ICIS)* [interaktyvus]. 2024. p. 209–215. [žiūrėta 2025-01-16]. Prieiga per internetą: <https://ieeexplore.ieee.org/document/10778343>.

23. IQBAL, F. - GOHAR, M. - KARAMTI, H. - KARAMTI, W. - KOH, S.-J. - CHOI, J.-G. Use of QUIC for AMQP in IoT networks. *Computer Networks* . 2023. Vol. 225, p. 109640. .

24. KUMAR, D. - KUMAR, D. Real-Time Application Layer Protocols to support lightweight mechanism in Internet of Things for e-Healthcare systems. *2024 5th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)* [interaktyvus]. 2024. p. 676–694. [žiūrėta 2025-01-16]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/10578420?casa_token=b4qpWnDYhykAAAAA:krBLEhZ4V3Jz71F9Qqocvt9SKUcO04bQPxWRf0QvNaPO5Dft_4mbh-g8c_aPlWSGHfA_VI69dQ#full-text-header>.

25. HINTAW, A.J. - MANICKAM, S. - ABOALMAALY, M.F. - KARUPPAYAH, S. MQTT Vulnerabilities, Attack Vectors and Solutions in the Internet of Things (IoT). *IETE Journal of Research* . 2023. Vol. 69, no. 6, p. 3368–3397. .

26. LAGHARI, S.U.A. - LI, W. - MANICKAM, S. - NANDA, P. - AL-ANI, A.K. - KARUPPAYAH, S. Securing MQTT Ecosystem: Exploring Vulnerabilities, Mitigations, and Future Trajectories. *IEEE Access* . 2024. Vol. 12, p. 139273–139289. .

27. ATILGAN, E. - OZCELIK, I. - YOLACAN, E.N. MQTT Security at a Glance. *2021 International Conference on Information Security and Cryptology (ISCTURKEY)* [interaktyvus]. 2021. p. 138–142. [žiūrėta 2024-11-29]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/9654337>.

28. HUSNAIN, M. - HAYAT, K. - CAMBIASO, E. - FAYYAZ, U.U. - MONGELLI, M. - AKRAM, H. - GHAZANFAR ABBAS, S. - SHAH, G.A. Preventing MQTT Vulnerabilities Using IoT-Enabled Intrusion Detection System. *Sensors* . 2022. Vol. 22, no. 2, p. 567. .

29. VACCARI, I. - AIELLO, M. - CAMBIASO, E. SlowITe, a Novel Denial of Service Attack Affecting MQTT. *Sensors* . 2020. Vol. 20, no. 10, p. 2932. .

30. BUCCAFURRI, F. - DE ANGELIS, V. - NARDONE, R. Securing MQTT by Blockchain-Based OTP Authentication. *Sensors* . 2020. Vol. 20, no. 7, p. 2002. .

31. HSIEH, C.-F. - CHANG, C.-K. The Security Method in MQTT Protocol for Internet of Things. *2021 International Conference on Technologies and Applications of Artificial Intelligence (TAAI)* [interaktyvus]. 2021. p. 280–284. [žiūrėta 2025-01-18]. Prieiga per internetą: <https://ieeexplore.ieee.org/document/9778046>.

32. KAUR, J. - SINGH, J. - KAUR, K. A Review on Machine Learning Based Techniques for MQTT Based Attacks for the IoT Systems. *2024 International Conference on Cybernation and Computation (CYBERCOM)* [interaktyvus]. 2024. p. 461–465. [žiūrėta 2025-02-17]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/10803264?casa_token=A1ii_qFuegAAAAAA:uBcLve80fE0wCX9zeiiUOeQ05aRpcgPPOFGebPVv2AR0djzGMlYzNzUd-0UdvQ0Ogbr6KhT_uA>.

33. EMPL, P. - BÖHM, F. - PERNUL, G. Process-Aware Intrusion Detection in MQTT Networks. *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy* [interaktyvus]. New York, NY, USA: Association for Computing Machinery, 2024. p. 91–102. [žiūrėta 2024-11-25]. Prieiga per internetą: <https://dl.acm.org/doi/10.1145/3626232.3653271>.

34. BHAI, R.N. - PATIL, M.S. Machine Learning Based Detection of DoS Attacks in Integrated MQTT-CoAP Heterogeneous IoT Network. *2024 5th International Conference for Emerging Technology (INCET)* [interaktyvus]. 2024. p. 1–5. [žiūrėta 2024-11-25]. Prieiga per internetą: <https://ieeexplore.ieee.org/document/10593222/?arnumber=10593222>.

35. ROLDÁN-GÓMEZ, J. - BOUBETA-PUIG, J. - CASTELO GÓMEZ, J.M. - CARRILLO-MONDÉJAR, J. - MARTÍNEZ MARTÍNEZ, J.L. Attack Pattern Recognition in the Internet of

Things using Complex Event Processing and Machine Learning. *2021 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* [interaktyvus]. 2021. p. 1919–1926. [žiūrėta 2025-02-20]. Prieiga per internetą: <https://ieeexplore.ieee.org/abstract/document/9658711?casa_token=zZs8cTUHlJcAAAAA:G6EEzrqq_PXVsQXKi9-D6SdaSFtqz8pCPJ-TRsvd5h8fqrMYgaYzIyusR5rgP0SO18dMEzbIOw>.

36. CHABCHOUB, Y. - TOGBE, M.U. - BOLY, A. - CHIKY, R. An In-Depth Study and Improvement of Isolation Forest. *IEEE Access* . 2022. Vol. 10, p. 10219–10237. .

37. CamillaCP/MQTT-traffic-generator. [interaktyvus]. [žiūrėta 2026-01-19]. Prieiga per internetą: <https://github.com/CamillaCP/MQTT-traffic-generator?tab=MIT-1-ov-file>.

38. MQTT Version 5.0 | OASIS Standard. [interaktyvus]. [žiūrėta 2026-05-06]. Prieiga per internetą: <https://docs.oasis-open.org/mqtt/mqtt/v5.0/mqtt-v5.0.html>.

39. AL-FAYOUMI, M. - ABU AL-HAIJA, Q. Capturing low-rate DDoS attack based on MQTT protocol in software Defined-IoT environment. *Array* . 2023. Vol. 19, p. 100316. .