VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

Lukas Stankovičius

ATVIROJO KODO TINKLO SAUGUMO SPRENDIMŲ TYRIMAS
ANALYSIS OF OPEN SOURCE NETWORK SECURITY SOLUTIONS

Baigiamasis magistro darbas

Telekomunikacijų inžinerijos studijų programa, valstybinis kodas 61211EX052
Telekomunikacijų technologijų specializacija
Elektronikos ir elektros inžinerijos studijų kryptis

Vilnius, 2020

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKO FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

TVIRTINU
Katedros vedėjas
_____
(parašas)
Algirdas Baškys
(Vardas, pavardė)
2020-05-31
(Data)

Lukas Stankovičius

ATVIROJO KODO TINKLO SAUGUMO SPRENDIMŲ TYRIMAS
ANALYSIS OF OPEN SOURCE NETWORK SECURITY SOLUTIONS

Baigiamasis magistro darbas

Telekomunikacijų inžinerijos studijų programa, valstybinis kodas 61211EX052

Telekomunikacijų technologijų specializacija

Elektronikos ir elektros inžinerijos studijų kryptis

Vadovas doc. dr. Arūnas Šaltis 2020-05-31
(pedag. vardas, moskl. laipsnis, vardas, pavardė) (Parašas) (Data)

Lietuvių kalbos konsultantas dr. Anželika Gaidienė _____ _____
(pedag. vardas, moskl. laipsnis, vardas, pavardė) (Parašas) (Data)

MB darbas peržiūrėtas lietuvių kalbos konsultantės dr. Andželikos Gaidienės:
Kompiuterijos ir ryšių technologijų katedros administratorė Jolanta Saudargaitė

Vilnius, 2020

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS
KOMPIUTERIJOS IR RYŠIŲ TECHNOLOGIJŲ KATEDRA

Elektronikos inžinerijos studijų kryptis
Telekomunikacijų inžinerijos studijų programa
Telekomunikacijų technologijos specializacija

TVIRTINU
Katedros vedėjas

(*parašas*)
Prof. dr. Algirdas Baškys

BAIGIAMOJO MAGISTRO DARBO UŽDUOTIS

2018-11-07
Vilnius

Studentui Lukas Stankovičius
(*vardas, pavardė*)

Baigiamojo darbo tema:

Atvirojo kodo tinklo saugumo sprendimų tyrimas

Analysis of Open Source Network Security Solutions

(*lietuvių ir anglų kalbomis*)

Patvirtinta 20 18 m. Lapkričio mėn. 07 d. dekano įsakymu Nr. **250**

Baigiamojo darbo užbaigimo terminas 2020 m. birželio mėn. 03 d.

BAIGIAMOJO DARBO UŽDUOTIES PRADINIAI DUOMENYS IR REIKALAVIMAI

1. Darbo tikslas

Ištirti atvirojo kodo tinklo saugumo sprendimų atakų atpažinimą ir paketų apdorojimo našumą.

2. Duomenys ir reikalavimai

Baigiamajame darbe turi būti atlikta įsibrovimo sistemų struktūros ir programinės įrangos apžvalga. Apžvelgti veikimo principai ir konfigūracijos ypatumai. Atlikti atakų atpažinimo tyrimą naudojant skirtingus duomenų rinkinius. Atlikti sistemos parametrų įtakos, našumo tyrimus, keičiant konfigūracijos parametrus, pateikti rezultatus bei rezultatus atitinkančias išvadas.

3. Aprašo turinys

1. Įvadas
2. Užduoties analizė
3. Įsibrovimo aptikimo sistemų apžvalga
4. Tyrimo metodikos kūrimas
5. Matavimų rezultatai
6. Išvados

Literatūros ir informacinių šaltinių sąrašas
Priedai (CD, DVD, USB, kita)
Tyrimų metu naudotų skriptų ir jų išeities kodai

Projekto vadovas: (*parašas*) doc. dr. Arūnas Šaltis
(*mokslinis vardas ir laipsnis, vardas, pavardė*)

Užduotį gavau: 2018-10-17 (data) (*parašas*) Lukas Stankovičius
(*vardas, pavardė*)

Vilniaus Gedimino technikos universitetas
Elektronikos fakultetas
Kompiuterijos ir ryšių technologijų katedra

ISBN ISSN
Egz. sk. .....
Data .....-.....-.....

Antrosios pakopos studijų **Telekomunikacijų inžinerijos** programos magistro baigiamasis darbas

Pavadinimas	**Atvirojo kodo tinklo saugumo sprendimų tyrimas**
Autorius	**Lukas Stankovičius**
Vadovas	**Arūnas Šaltis**

Kalba: lietuvių

Anotacija

Atvirojo kodo tinklo saugumo sprendimų tyrimas Magistro baigiamasis darbas inžinerijos mokslų laipsniui. Vilniaus Gedimino technikos universitetas. Vilnius, 2020, 85 p., 31 iliustr., 16 lent., 42 bibl.

Magistro darbo tikslas – išanalizuoti „Snort“, „Suricata“ ir „Zeek“ įsibrovimo aptikimo sistemų atakų atpažinimą, sisteminių parametrų įtaką ir paketų apdorojimo našumą. Tyrimai atlikti naudojantis skirtingais duomenų rinkiniais, tinklo sąsajos moduliais, konfigūracijos parametrais. Nagrinėjami parametrai: atakų atpažinimas, pranešimų būsenos, paketų apdorojimo greitis, atmestų paketų ir CPĮ naudojimo kiekis procentais. Apibendrinti tyrimo rezultatai yra palyginami tarpusavyje. Darbo apimtis – 79 p. teksto be priedų, 31 iliustracijos, 16 lentelės, 42 bibliografiniai šaltiniai.

Prasminiai žodžiai: „Snort“, „Suricata“, „Zeek“, našumas, „Linux“, atmesti paketai

Vilnius Gediminas Technical University
Faculty of Electronics
Department of Computer Science and Communications Technologies

ISBN ISSN
Copies No. .....
Date .....-.....-.....

Master Degree Studies **Telecommunications Engineering** study programme Master Graduation Thesis

Title	**Analysis of Open Source Network Security Solutions**
Author	**Lukas Stankovičius**
Academic supervisor	**Arūnas Šaltis**

Thesis language: Lithuanian

Annotation

Analysis of Open Source Network Security Solutions Master's Thesis for the Degree of Engineering Sciences. Vilnius Gediminas Technical University. Vilnius, 2020, 85 p., 31 illustrations, 16 tables, 42 bibl.

The aim of the master's thesis is to analyze the attack detection of Snort, Suricata and Zeek intrusion detection systems, the influence of system parameters and packet processing efficiency. The research was performed using different data sets, network interface modules, configuration parameters. Parameters considered: attack detection, message status, packet processing speed, percentage of rejected packets and CPU usage. The summarized results of the study analized and compared. Thesis consists of: 79 p. text without appendixes, 31 illustrations, 16 tables, 42 bibliographical entries.

Keywords: Snort, Suricata, Zeek, performance, Linux, dropped packets

Vilniaus Gedimino technikos universiteto egzaminų sesijų ir baigiamųjų darbų rengimo bei gynimo organizavimo tvarkos aprašo
2 priedas

(Baigiamojo darbo sąžiningumo deklaracijos forma)

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS

Lukas Stankovičius, 20145274
(Studento vardas ir pavardė, studento pažymėjimo Nr.)
Elektronikos fakultetas
(Fakultetas)
Telekomunikacijų inžinerija, TETfm-18
(Studijų programa, akademinė grupė)

BAIGIAMOJO DARBO (PROJEKTO)
SĄŽININGUMO DEKLARACIJA

2020 m. birželio 02 d.

Patvirtinu, kad mano baigiamasis darbas tema „Atvirojo kodo tinklo saugumo sprendimų tyrimas“ patvirtintas 2018 m. lapkričio 07 d. dekano potvarkiu Nr. 250el, yra savarankiškai parašytas. Šiame darbe pateikta medžiaga nėra plagijuota. Tiesiogiai ar netiesiogiai panaudotos kitų šaltinių citatos pažymėtos literatūros nuorodose.

Mano darbo vadovas docentas daktaras Arūnas Šaltis.

Kitų asmenų indėlio į parengtą baigiamąjį darbą nėra. Jokių įstatymų nenumatytų piniginių sumų už šį darbą niekam nesu mokėjęs (-usi).

(Parašas)

Lukas Stankovičius
(Vardas ir pavardė)

TURINYS

SANTRUMPOS

AF_PACKET – „Linux“ branduolio tinklo sąsajos modulis

CPĮ – centrinis procesoriaus įrenginys (angl. *Central Processing Unit*)

DNS – domenų vardų sistema (angl. *Domain Name System*)

DoS – paslaugos uždraudimas (angl. *Denial of Service*)

DDoS – paskirstytas paslaugos uždraudimas (angl. *Distributed Denial of Service*)

ET – „Emerging Threats“ taisyklių rinkinys

FTP – failų perdavimo protokolas (angl. *Filer Transfer Protocol*)

HTTP – saityno protokolas (angl. *Hyper Text Transfer Protocol*)

HTTPS – padidinto saugumo žiniatinklio protokolas (angl. *Hyper Text Transfer Protocol*)

ĮAS – įsibrovimo atpažinimo sistema (angl. *Intrusion Detection System*)

IP – interneto protokolas (angl. *Internet Protocol*)

KN – klaidingai neigiamas pranešimas (angl. *False Negative*)

KT – klaidingai teigiamas pranešimas (angl. *False Positive*)

MAC – fizinio lygmens prieigos kontrolės adresas (angl. *Media Access Control*)

PCAP – srauto paketų išsaugojimo formatas (angl. *Packet Capture*)

PF_RING – „Linux“ branduolio tinklo sąsajos modulis

RAM – kompiuterio operatyvioji atmintis (angl. *Random Access Memory*)

RST – TCP protokole naudojama atsakyto vėliavėlė

SYN – TCP protokole naudojama susijungimo iniciavimo vėliavėlė

SSH – saugaus tunelio protokolas (angl. *Secure Shell*)

TCP – transporto kontrolės protokolas (*angl. Transmission Control Protocol*)

TT – tikrai teigiamas pranešimas (angl. *True Positive*)

UDP – vartotojo duomenų perdavimo protokolas (angl. *User Datagaram Protocol*)

URI – resursų identifikavimo žymuo (angl. *Uniform Resource Identifier*)

VRT – įmonės „Sourcefire“ pažeidžiamumo tyrimų komanda (angl. *Vulnerability Research Team*)

PAVEIKSLŲ SĄRAŠAS

LENTELIŲ SĄRAŠAS

ĮVADAS

Tyrimo objektas

Valdant tinklus, saugumo terminas suprantamas kaip duomenų apsaugojimas, naudojant įvairias autorizacijos ir apsaugos priemones. Tinklo saugumas susideda iš įvairių apsaugos sluoksnių, kurie įdiegti kraštiniuose maršruto parinktuvuose. Kiekvienas saugumo sluoksnis susideda iš įvairių taisyklių ir praktikų, kurių tikslas – stebėti ir laiku atpažinti neleistinus prisijungimus. Vartotojai turi prieigos vardą ir slaptažodį, kurie leidžia jiems autorizuotis tam tikroje sistemoje. Jeigu vartotojo vardas ir slaptažodis atitinka duomenų bazėje saugomus duomenis, vartotojui yra suteikiama prieiga, jeigu ne – prieiga nesuteikiama. Šis procesas yra išlikęs iki šiol, tačiau atsiranda poreikis, dėl padidėjusios autorizacijos galimybės, nuolat stebėti autorizaciją, duomenų apsikeitimą.

Tinklų saugumas apima visas darbo sritis, kuriose yra kompiuterinių tinklas. Saugumas reikalingas tokiuose procesuose kaip sutarčių, sandorių sudarymas, komunikacija tarp įvairių įmonių, valstybinių įmonių informacijos perdavimas ar individualus saugumas. Šiuolaikinis pasaulis vis labiau žengia informacijos skaitmenizavimo link, todėl kiekviena organizacija, kuri teikia paslaugas darbuotojams ir klientams, turi apsaugoti savąjį tinklą ir informaciją nuo atakų. Šiuolaikinės atakos pasižymi įvairove ir dideliu srauto kiekiu. Tampa vis sunkiau kontroliuoti atakas, naudojant ugniasienes, kurios praleidžia arba draudžia srautą pagal paketų charakteristikas. Reikalingos papildomos tinklo apsaugos priemonės, kurios reaguotų į paketų perduodamus duomenis. Vienos iš priemonių, saugančios įmonės vidinį tinklą nuo piktavališko srauto, yra įsibrovimo aptikimo sistemos. Šios priemonės dažnai būna pirmasis taikinys atakų sukėlėjams. IT industrija nuolat tobulėja ir keičiasi, todėl tinklo užpuolikai yra priversti adaptuotis ir ieškoti naujų įsibrovimo į aukos tinklą būdų. Įsibrovimo aptikimo sistemų kūrimas gali būti suprastas kaip nuolatinis procesas, kuriuo metu kovojama su užpuolikų bandymais įsilaužti.

Darbo tikslas ir uždaviniai

Saugumo sprendimai tampa aktualesni valdant kompiuterinių tinklus. Sistemų administratoriai pasirenka mokamus arba atvirojo kodo programinius paketus, siekdami apsaugoti įmonės tinklą. Šiuolaikinėms atakoms reikalinga programinė įranga, gebanti atpažinti atakas net esant intensyviam srautui.

Magistro baigiamojo darbo tikslas – atlikti atvirojo kodo tinklo saugumo sprendimų tyrimą, siekiant įvertinti programinių paketų našumą ir atakos atpažinimo tikslumą.
Darbo uždaviniai:

1. Atlikti įsibrovimo atpažinimo sistemų tikslumo tyrimą, esant mišriam srautui.
2. Atlikti įsibrovimo atpažinimo sistemų našumo tyrimą, keičiant konfigūracijos parametrus, išsiaiškinant, kuri iš atvirojo kodo sistemų yra tiksliausia ir sparčiausia.

Temos naujumas

Kiekvieną dieną atsiranda naujų saugumo pažeidimų ar spragų, kuriomis galima pasinaudoti siekiant piktavališkų kėslų. Atakas tarp vartotojų srautu sunku atpažinti, todėl jos dažnai lieka nepastebėtos. Išaugusi atakų perdavimo sparta taip pat kelia papildomų problemų siekiant visiško atakos atpažinimo.

Temos aktualumas

Šiuolaikinės įsibrovimo grėsmės patiriamos ne tik tinklo lygmeniu, bet ir programos lygmenyje. Saugumo sprendimai turi apsaugoti tinklą, paslaugas ir programas, turi suteikti saugų prisijungimą prie įmonės vidinio tinklo, saugų darbuotojų prisijungimą ir kontroliuoti interneto prieigą. Apie 70 % visų naujų atakų yra skirtos saityno programoms ir šių atakų skaičius vis auga. Įmonės turi rasti sprendimus galinčius apsaugoti saityno prieigą, saityno serverius ir programas. Saugumo programinės įrangos sprendimai turi būti lengvai įdiegiami ir išlikti aktualūs ilgą laiką.

1. UŽDUOTIES ANALIZĖ

Kibernetinio saugumo problema yra aktuali pasaulio šalims, įmonėms ir vartotojams, kuriems svarbu informacijos konfidencialumas. Šalys nuolat plėtoja saugumo sektorių dėl vis didėjančių atakų ir kylančių grėsmių pasisavinti informaciją. Remiantis 2019 m. įmonių saugumo patikrinimo testu – 75 % atliktų atakų yra saityno atakos, 25 % yra prieiga, gauta per kitas sistemas [29]. Atakos skirstomos į dvi kategorijas: pasyvios ir aktyvios. Pasyvi ataka, kada įsibrovėlis stebi perduodamą srautą neleistinu būdu (angl. *Eavesdropping*). Ši ataka nepakeičia siųstų duomenų, todėl yra sunkiau atpažįstama. Atakuojamas mazgas neturi jokios informacijos apie esamą ataką [33]. Pasyvios atakos gali būti klasifikuojamos į du tipus:

1. Pranešimo turinio, kai atakos sukėlėjas nori gauti perdavinėjamą informaciją.
2. Srauto analizavimas, kai atakos sukėlėjas negali gauti pranešimo turinio dėl šifravimo ar kitų priemonių, tačiau gali analizuoti persiunčiamą srautą tarp mazgų, siekiant rasti informacijos šabloną.

Aktyvios atakos metu įsibrovėlis bando modifikuoti persiunčiamus duomenis tarp mazgų, atakos skirstomos į tipus [33]:

1. Pertraukties ataka vyksta, kai neturintis prieigos atakos sukėlėjas bando apsimesti kitu asmeniu.
2. Modifikacijos ataka vyksta, kai yra modifikuojamas originalus pranešimas.
3. Klastotės ataka sukelia paslaugos uždraudimo (angl. *Denial of Service*) atakas, kurių tikslas atakos sukėlėjui gauti prieigą prie tinklo paslaugų ir jas uždrausti autorizuotam vartotojui.

Skirtumai tarp pasyvių ir aktyvių atakų [34][32]:

1. Aktyvi ataka modifikuoja gautą pranešimą, pasyvi atakos neatlieka jokio duomenų pakeitimo nutekintai informacijai.
2. Aktyvi ataka sukelia didžiulius nuostolius aukai, pavyzdžiui, serverio nulaužimas, kuriame veikia visos įmonės programos. Papildomas programų neveikimo laikas (angl. *Downtime*) turi finansines pasekmes, tačiau pasyvios atakos nesukelia sistemos griūties.
3. Pasyvi ataka grėsminga duomenų privatumui, aktyvi ataka grėsminga jos integralumui ir prieigai.
4. Auka pastebi, kada aktyvi ataka vyksta, pasyvi ataka nėra pastebima aukai.
5. Aktyvi ataka siekia gauti visą sistemos kontrolę, pasyvi ataka tik stebi perduodamą informaciją.

Šiuolaikinės atakos apima visus tinklo lygmenis ir įrangą, dėl to atakų prevencija yra sunkiai įgyvendinama, reikalingas atakų atpažinimas. Kovai su atakomis naudojami saugumo įrankiai: ugniasienės, įsibrovimo aptikimo ir įsibrovimo prevencijos sistemos. Ugniasienė filtruoja srautą, praleisdama arba atmesdama įeinančius ir išeinančius paketus. Įsibrovimo aptikimo sistemos atlieka tinklo srauto stebėjimą. Įsibrovimo aptikimo sistemos analizuoja esamo tinklo srauto perduodamus paketus pagal taisykles ir generuoja įspėjamąjį pranešimą, kai ataka yra atpažįstama. Įsibrovimo prevencijos sistemos analizuoja tinklo srautą pagal taisykles, tačiau, aptikus galimą ataką, atlieka prevencijos veiksmus: paketus blokuoja arba praleidžia. Baigiamajame magistro darbe bus tiriamos įsibrovimo aptikimo sistemos.

Analogiškuose tyrimuose dažnai tiriamas atvirojo kodo programinių paketų našumas. Atvirojo kodo programinių paketų „Snort“ ir „Suricata“ našumas rodiklis, paketai per sekundę, buvo tiriamas Klarksono universisteto mokslininkų [42]. Autoriai savo tyrime atliko paprastą palyginimą tarp įsibrovimo aptikimo sistemų. Autoriai tyrime palygina „Snort“ ir „Suricata“ paketų apdorojimo našumą, skaitant 10 duomenų rinkinių srautą. Duomenų rinkinį sudarė piktavališko ir normalaus srauto failai. Matavimai atlikti, keičiant taisyklių rinkinius: „ET-Open“, „ET-Free“, „ET-Pro“ ir „VRT“, keičiant programinių paketų atpažinimo algoritmus ir keičiant aparatinės įrangos parametrą – CPĮ branduolių skaičių iki 24. Straipsnyje pateikti rezultatai rodo, kad „Suricata“ ir „Snort“ paketų apdorojimo našumas tolygiai priklauso nuo išskiriamų branduolių skaičiaus, tik CPĮ branduolių naudojimo kreivės kitimas skiriasi. Matomas našumo skirtumas, naudojant skirtingus taisyklių rinkinius, autoriai pateikia rezultatus, kuriuose naudojant „ET-Pro“ rinkinį „Snort“ našumas pagreitėjo 7718, o „Suricata“ 49918 paketų per sekundę lyginant su bendruomenei skirtu „ET-Open“ taisyklių rinkiniu. Autoriai pabrėžia konfigūracijos pakeitimų svarbą, siekiant didesnės spartos. Matavimuose „Snort“ programinis paketas buvo sukonfigūruotas naudoti kelias gijas vienu metu, o „Suricata“ pakeistas paketų eilės algoritmas. Galutiniuose rezultatuose pasirodė, kad „Snort“ ir „Suricata“ programų našumas gali būti didinamas pagal procesoriaus branduolių kiekį, tačiau „Suricata“ parodė geresnius rezultatus beveik visuose testuose lyginant su „Snort“. Aptartas straipsnis yra įdomus, nes matuoja skirtingas paketų konfigūracijas ir tiria skirtingus taisyklių rinkinius per 10 duomenų rinkinių. Kita vertus, autoriai atliko našumo tyrimus paketų skaitymo režime, todėl pateikti rezultatai yra našumas idealiomis sąlygomis, kai sistemos nepatiria jokios perkrovos.

Kitas straipsnis, palyginantis „Snort“ ir „Suricata“ našumą, yra publikuotas Džamu mokslininkų 2020 m. [26]. Autoriai tyrimą motyvavo tuo, kad reikalinga šiuolaikinė, detali ir

techniškai aprašyta informacija apie populiariausius įsibrovimo aptikimo programinius paketus. Matavimai atlikti generuojant dideles spartas. Tirti „Snort“ 2.9.12 ir „Suricata“ 4.12 versijų programiniai paketai. Tyrimai atlikti perduodant srautą, esant skirtingiems paketų dydžiams: 1400, 1024 ir 512 baitų. Tiriant naudojamas tik TCP protokolas, generuotos perdavimo spartos: 10000, 20000, 30000, 40000 ir 50000 paketų per sekundę. Matavimo rezultatai parodė, kad „Suricata“ geba efektyviai naudoti kelis branduolius vienu metu, todėl matomi geresni paketų analizės rezultatai. Esant maksimaliai 50000 paketų per sekundę spartai ir 1400 paketų ilgiui, stebimas didelis atmestų paketų kiekio padidėjimas „Snort“ ir „Suricata“ paketuose. Autoriai šį rezultatą aiškina kaip abiejų įsibrovimo aptikimo sistemų nesugebėjimą analizuoti didelių paketų didelėmis spartomis. Iš gautų rezultatų matyti, kad „Suricata“ paketas buvo našesnis atliekant beveik visus matvimus nei „Snort“ ir yra laikomas idealiu sprendimu, jei nėra ribojami sistemos resursai. Tačiau „Snort“ yra našesnis sprendimas už „Suricata“, kai abi programos veikia vienos gijos architektūroje.

Atakų atpažinimo klausimas nagrinėjamas amerikiečių mokslininkų [24]. Straipsnyje analizuojamas „Snort“ ir „Suricata“ atakų atpažinimo tikslumas. Atlikti 54 testai, kurie išskirstyti į 9 kategorijas, reprezentuojančias skirtingas atakas, kurios vykdomos internete. „Snort“ ir „Suricata“ naudojo tą patį taisyklių rinkinį – „ET-Open“. Atakos generuotos naudojant „Pytbull“ programą. Iš gautų rezultatų matyti, kad „Snort“ neaptiko 16 atakų, „Suricata“ – 12 atakų. Autoriai šį skirtumą paaiškina, kad „Snort“ nevisiškai įjungė visas taisyklių rinkinio taisykles arba jos nėra iki galo optimizuotos. Autoriai pabrėžia, kad, kai abi programos neaptinka atakos, tai yra taisyklių rinkinio klaida, o ne skirtumai tarp aptikimo algoritmų. Siekiant tiksliau nustatyti, ar ataka yra atpažinta ir sumažinta klaidingai teigiamų pranešimų įtaka rezultatams, autoriai įvedė papildomą parametrą – įspėjamuosius pranešimus. Įspėjamieji pranešimai pagal kontekstą gali būti klaidingai teigiami arba tikrai teigiami pranešimai. Gautuose rezultatuose „Snort“ generavo 10, o „Suricata“ 8 klaidingai teigiamus pranešimus. Šis skirtumas autorių grindžiamas įvertinus įspėjamuosius pranešimus. Daugelis šių pranešimus generuoti testuojant įsibrovimo atakas, kai atliekamos tokios atakos kaip prievadų skenavimas ir informacijos rinkimas. Galutiniuose rezultatuose pateikiami tikslumo rezultatai, įvertinant klaidingai teigiamų, tikrai teigiamų ir įspėjamųjų pranešimų santykį. Gauta, kad „Snort“ tikslumas yra 81 %, „Suricata“ 91 %. Pagrindinis straipsnio trūkumas, kad matavimai atlikti tik su kenkėjišku srautu, todėl tokio tikslumo rezultatų negalima tikėtis, nagrinėjant miršrų srautą. Apžvelgę pateiktus analogiškus tyrimus, galime suformuluoti šiuos apibendrinimus:

1. Atvirojo kodo įsibrovimo aptikimo sistemų analizė yra aktuali šiuolaikinėje tinklo infrastruktūroje. Populiauriausi atvirojo kodo paketai – „Snort“ ir „Suricata“.
2. Atakų atpažinimas ir našumas yra dvi ypač nagrinėjamos temos. Apžvelgtuose tyrimuose atakų atpažinimas tikrinimas tik leidžiant atakos srautą, todėl esant realistiškam srautui galimi kitokie rezultatai.
3. Apžvelgiami programiniai paketai, kurie paremti šabloninėmis taisyklėmis, trūksta tyrimų, kurie apžvelgų kitus programinuis paketus.
4. Našumo tyrimuose nevisiškai išanalizuojamos įsibrovimo aptikimo sistemų konfigūracijos galimybės, atpažinimų algoritmų įtaka, tinklo modulių įtaka našumui.

2. ĮSIBROVIMO APTIKIMO SISTEMŲ APŽVALGA

Įsibrovimo aptikimo sistema – ĮAS yra procesas, kuris stebi įvykius mazge ar tinkle. Analizuoja įvykius, siekiant aptikti būsimus įsilaužimus ar tinklo taisyklių pažeidimus. Įvykiai gali būti sukelti daugelio priežasčių: kenkėjiškos programos veikla, neleistinai gauta prisijungimo informacija prie vidinių sistemos resursų, neleistini vartotojų veiksmai. Įsibrovimo aptikimo sistema automatizuoja įsibrovimo aptikimo procesą [39].

Įsibrovimo prevencijos sistemos procesas panašus į atpažinimo procesą, tik šiame procese bandoma net tik atpažinti atakas, bet ir jas sustabdyti. Įsibrovimo aptikimo ir prevencijos sistemos tikslas identifikuoti galimus incidentus, registruoti informaciją apie juos, bandyti sustabdyti incidentus ir pranešti apie įvykius sistemos administratoriams [19].

2.1. Įsibrovimo atpažinimo sistemų naudojimo būdai

Pagrindinė ĮAS funkcija yra atpažinti galimus saugumo incidentus stebimoje sistemoje. Pavyzdžiui, jei gaunama neleistina prieiga prie vidinių sistemos resursų, pasinaudojant programos pažeidžiamumu, tai yra incidentas, kurį ĮAS gali aptikti. Apie incidentą ĮAS praneša administratoriams, kurie imasi veiksmų problemai spręsti. Dažnai ĮAS naudojamas kaip įrankis analizei po incidento. ĮAS surinkto srauto ir žurnalinių įrašų duomenys suteikia daugiau informacijos apie incidento atsiradimo aplinkybes. Galimi ĮAS panaudojimo būdai [39]:

1. Saugumo politikų problemų nustatymas – ĮAS gali palaikyti tam tikrą saugumo kokybės kontrolę. Pavyzdžiui, nustatant, kada pakartotinai naudojamos ugniasienės taisyklės arba kada stebimas srautas, kuris turėjo būti užblokuotas ugniasienės.
2. Dokumentacija apie esančias atakas – ĮAS renka ir rašo informaciją į žurnalinius failus apie buvusias ir esamas atakas. Žurnaliniai įrašai organizacijos saugumo komandai padeda geriau suprasti atakų dažnumą ir charakteristikas, kurios būna nukreiptos prieš organizaciją. Ši informacija naudinga nustatant tinkamas apsaugos priemones. Informacija apie atakas gali būti naudojama, informuojant vadovybę su kokiomis grėsmėmis susiduria organizacija.
3. Saugumo politikų užtikrinimas – ĮAS atlieka saugumo politikų stebėseną, todėl organizacijos darbuotojai labiau laikosi saugumo politikos, žinant, kad jų veiksmai yra stebimi.

ĮAS tampa neatsiejama organizacijų saugumo dalis, užtikrinanti ne tik incidentų atpažinimą, bet ir visos organizacijos saugumo politikos išlaikymą.

2.2. Įsibrovimo atpažinimo sistemų metodologijos

ĮAS efektyvumas priklauso nuo to kaip sistema geba greitai ir tiksliai priimti sprendimus. Dauguma ĮAS sistemų naudojasi keliomis metodologijomis, integruojant jas kartu, siekiant suteikti visapusišką atakų atpažinimą [37].

2.2.1. Šablonu pagrįstas aptikimas

Šablonu grįsta metodologija stebi srautą ir ieško atitinkamo srauto šablono, pagal kurį priima sprendimus. Šablonu pagrįstas aptikimo procesas – palygina dabartinio srauto šabloną su jau žinomų įvykių šablonais, siekiant atpažinti galimus incidentus [39]. Pavyzdžiui, siekiama prisijungti prie įrenginio per SSH sąsają prie įrenginio kitu vardu nei numatyta anksčiau arba gaunamas kenkėjiškas laiškas, kuriama prisegtas failas su failo formatu „exe“.

Šablonu pagrįstas aptikimas yra paprasčiausias aptikimo mechanizmas, nes jis palygina esamus įvykius su jau buvusiais įvykiais. Įvykių aptikimas efektyvus tik tuomet, jeigu įsibrovimo įvykiai kartojasi. Pasikeitus atakos parametrams šis metodas tampa mažiau efektyvus, todėl reikalinga nuolatinė taisyklių priežiūra ir atnaujinimas. Šablonu paremta įsibrovimo atpažinimo sistema remiasi esamais šablonais, atpažįstant piktavališką srautą. Kada esamas šablonas sutampa su piktavališku srautu, perspėjimo pranešimas generuojamas tinklo administratoriui. Įspėjimo pranešimai gali atpažinti virusus, kirminus, kenkėjiškus laiškus, tinklo skenavimą ir generuojamas pažeidžiamumo atakas.

2.2.2. Anomalija pagrįstas aptikimas

Anomalija pagrįstas aptikimo procesas palygina įprastinio srauto parametrus su stebimo srauto parametrais, siekiant pastebėti nukrypimus. ĮAS, naudojančios šį metodą, turi atskirus profilius, kurie aprašo įprastinio srauto parametrus: vartotojų prisijungimai, programų valdymas, tinklų susijungimai. Profiliai yra sudaromi, stebint įprastinio srauto charakteristikas per tam tikrą laikotarpį [39].

Anomalija pagrįstoms ĮAS generuojamas srautas yra svarbesnis nei nešama informacija. Anomalija pagrįsti įrankiai remiasi tam tikrais rėmais, o ne šablonais. Pagrindinis bruožas – neįprastas vartotojo aktyvumas. Pavyzdžiui, vartotojas visuomet prisijungia prie tinklo iš specifinės vietos, šiai vietai pakitus į kitą, sistema generuoti perspėjimo pranešimus apie anomaliją.

Anomalija pagrįstas aptikimas yra labiau efektyvus, aptinkant naujas atakas, nei šabloninis aptikimas. Pavyzdžiui, kompiuterinis mazgas yra apkrečiamas tam tikra programine įranga, kuri limituoja sistemos resursus ir naudoja juos piktavališkoms prasmėms. Anomalija pagrįstas aptikimas aptinka nuokrypius nuo įprastinio kompiuterio darbo ir gali pranešti apie vykstantį incidentą. Anomalija pagrįstas metodas turi didesnę tikimybę sugeneruoti klaidingai teigiamus pranešimus, nes bet koks nukrypimas nuo profilio charakteristikų yra laikomas kaip galimas įsibrovimo įvykis [27]. Sudarant srauto profilį, reikia atsižvelgti į srauto dinamiškumą, siekiant sumažinti klaidingai teigiamų pranešimų tikimybę. Metodo efektyvumą galima pagerinti, pasinaudojant neuronų tinklų (angl. *Artificial Neural Networks*) technikomis [31]:

1. Neapibrėžta logika (angl. *Fuzzy Logic*) – paremta tikimybėmis, naudojasi reikšmėmis nuo 0 iki 1, pagal tai yra sprendžiama anomalijos tikimybė sraute.
2. Dirbtiniai neuroniniai tinklai – gali būti panaudojami nestruktūrizuoto srauto suskirstymui į grupes ir atskyrimui nuo įprasto srauto ir paveikto anomalijų.
3. Palaikymo vektorių mašinos (angl. *Support Vector Machines*) – gali būti naudojama atpažinti įsibrovimo įvykius, kai duomenų kiekis nėra didelis ir kai duomenų aprėptis nekeičia sprendimo tikslumo.

2.2.3. Protokolo būsenos analizė

Protokolo būsenos analizės procesas yra panašus į anomalijų atpažinimą. Šioje metodikoje ĮAS palygina standartinių protokolų profilių charakteristikas su stebimo srauto naudojamų protokolų charakteristikomis, siekiant atpažinti nuokrypius. Pagrindinis skirtumas, kad protokolo būsenos analizėje naudojami gamintojo sukurti universalūs profiliai, kurie nusako atitinkamo protokolo veikimą [38].

Pagrindiniai privalumai, kad ĮAS šiuo metodu supranta ir stebi būsenas tinklo, transporto ir programos lygmens protokolus. Pavyzdžiui, jeigu vartotojas prisijungia prie FTP serverio neautentifikuodamas, anonimiškai, ir bando vykdyti užklausas, kurios galimos tik autorizuotam vartotojui, tai gali būti žymima kaip įtartinas elgesys. Šiuo metodu ĮAS gali sekti netinkamą komandų ar užklausų eiliškumą. Pavyzdžiui, ta pati užklausa vykdoma kelis kartus iš eilės – SYN užklausa TCP protokole arba komandų vykdymas ne iš eilės, pagal protokolo struktūrą [27].

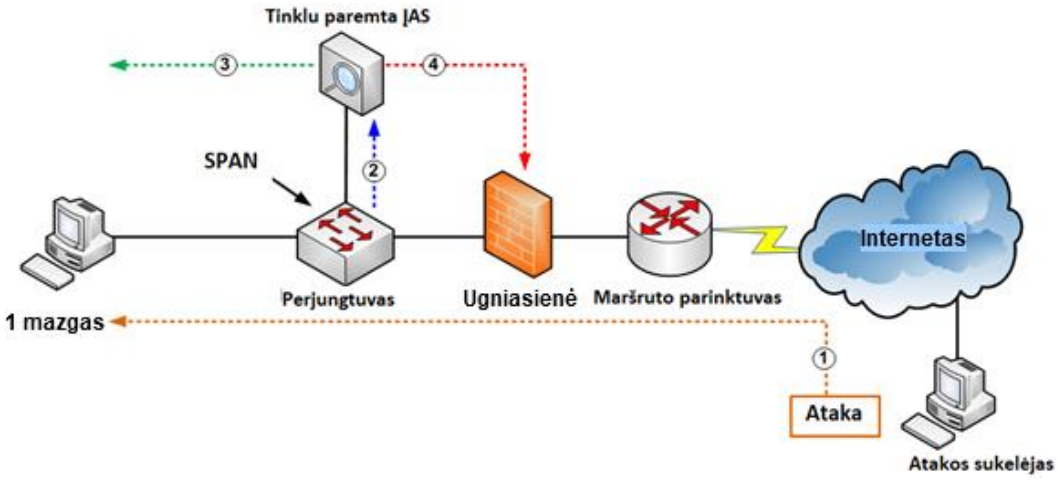
Pagrindinis metodologijos trūkumas yra tai, kad sistema reikalauja didelio resursų kiekio dėl išsamaus protokolų analizavimo ir tarnybinių duomenų kiekio, stebint kiekvieną sesiją. Taip pat metodas negali atpažinti įsibrovimų, kurie nepažeidžia standartinių protokolų charakteristikų.

2.3. Įsibrovimo atpažinimo sistemų tipai

Yra keli įsibrovimo atpažinimo sistemų architektūrų tipai. Pasirinktas architektūros tipas priklauso nuo organizacijos ir esamos tinklo infrastruktūros. ĮAS architektūros tipus galima išskaidyti į atskiras grupes, kurie pasižymi specifiniais bruožais.

2.3.1. Tinklu paremta įsibrovimo atpažinimo sistema

Tinklu paremtos įsibrovimo atpažinimo sistemos stebi ne atskirus įrenginius, o patį tinklą arba tinklo segmentą [37]. Pagrindinis tikslas – analizuoti tinklo ir programų tarpusavio naudojamus protokolus, siunčiamus paketus ir veiksmus, siekiant atpažinti piktavališką srautą. Pagrindinis bruožas, kad ĮAS yra už tinklo ribų. 2.2 pav. pateikiamas pavyzdys kaip vienas iš ĮAS įdiegimo variantų organizacijoje tinkle. Tinklu paremtos sistemos stebi tam tikrus tinklo segmentus arba įrenginius, analizuoja tinklo ir programų protokolo veiksmus, siekiant atpažinti įtartiną srautą [38]. Dažniausiai tinklu paremta ĮAS įdiegiama už ugniasienės ribų, VPN (angl. *Virtual Private Network*) serverių, nuotolinės prieigos serverių ir belaidžio tinklo.



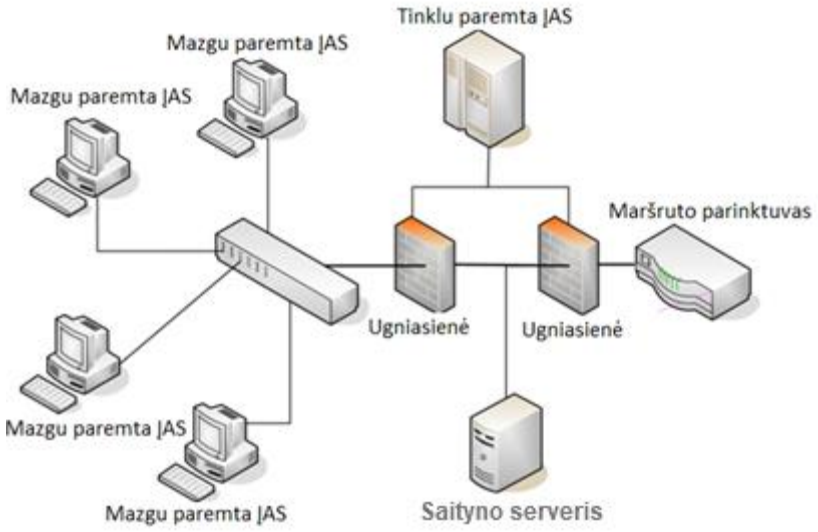
2.1 pav. Tinklo struktūra, kurioje įdiegta įsibrovimo atpažinimo sistema [38]

Struktūra, kaip atakos sukėlėjas bandytų pasiekti vidinio tinklo mazgą, pateikta 2.1 pav. Joje matyti ir ĮAS įdiegimo vieta, taip pat matyti, kad sistema apribota nuo tiesioginės prieigos prie mazgo. Tokioje struktūroje ĮAS neriboja perdavimo spartos į mazgą. Įeinančių paketų stebėjimą atlieka pasinaudojant perjungtuvų funkcija – paketų kopijavimu iš vieno prievado į kitą (angl. *Switched Port Analyzer)*. Tokiu būdu ĮAS žino, kas yra tinkle, ir nedaro jam tiesioginės įtakos.

2.3.2. Tinklo mazgu paremta įsibrovimo atpažinimo sistema

Tinklo mazgu paremta ĮAS pasižymi, kad tinklo srautas stebimas tik ant vieno mazgo. Dažniausiai tokio tipo sistemos ne tik stebi mazgo įeinantį ir išeinantį srautą, bet kartu stebi ir vidinius failų sistemos pokyčius. Pateikiama tinklo mazgo paremtos ĮAS įdiegimas 2.3 pav. Vidiniai mazgo procesai ir generuojamas srautas analizuojamas tinklo mazgu paremtomis ĮAS [38].

Bendrai galima pateikti pagrindines sistemų funkcijas: tinklo srauto stebėjimas, sistemos veiklos registravimas, procesų analizė, programų kontrolė ir stebėjimas, failų modifikacijos įvykių įrašymas į žurnalus, stebimi konfigūracijos pakeitimai [38]. Tokio tipo sistemos dažniausiai yra įdiegiamos mazguose, kur laikoma svarbi informacija kaip duomenų bazėse, paslaugų serveriuose.



2.2 pav. Tinklo mazgu paremta įsibrovimo atpažinimo sistema [25]

2.3.3. Hibridinis įsibrovimo atpažinimo sistemų tinklas

Tinklu paremta ĮAS stebi tinklą ar jo segmentą, mazgu paremta ĮAS stebi mazgo failų sistemą, tačiau sistemos yra izoliuotos viena nuo kitos. Izoliuotos ĮAS neturi visos informacijos apie tinkle esančius procesus ir gali būti labiau pažeidžiamos naujų atakų [25]. Skirtingų tipų ĮAS sujungimas į vieną tinklą leidžia efektyviau dalintis informaciją iš kitų ĮAS. Didesnis informacijos kiekis leidžia tiksliau atpažinti įsilaužimus lokaliai arba visame tinkle. Tinklas, kuris sujungia

tarpusavyje skirtingas ĮAS bendram informacijos apsikeitimui, vadinamas bendradarbiavimo įsilaužimo aptikimo tinklu (angl. *Cooperative Intrusion Detection Network*) [25].

2.4. Atvirojo kodo įsibrovimo atpažinimo įrankiai

ĮAS leidžia identifikuoti problemas, susijusias su saugumo pažeidimo taisyklėmis, esamų grėsmių ir atakų registravimu ir laikytis prevencijos. Šiuo metu ĮAS sistemos tapo neatsiejamu įrankiu, kuriant saugumo infrastruktūrą bet kuriame tinklo mazge ar organizacijoje.

Šiuolaikinis pasaulis vis labiau koncentruojasi į saugumo kryptį, todėl kiekvienais metais išleidžiami nauji arba patobulinti atviro kodo saugumo sprendimai. Pateikiama keletas populiariausių atviro kodo saugumo įrankių [37]:

1. „Snort“ – viena seniausių atvirojo kodo programinių įrankių, sukurtas stebėti ir registruoti saugumo incidentus tinklo paremta ĮAS.
2. „Suricata“ – atvirojo kodo įsibrovimo aptikimo ir prevencijos sistema, palaikanti daugiagyslę architektūrą.
3. „OpenDLP“ – atvirojo kodo duomenų apsaugos įrankis, gebantis identifikuoti jautrią sistemos informaciją iš „Windows“, „Linux“ operacinių sistemų ar „MySQL“ duomenų bazės.
4. „OSSIM“ – šio programinio paketo tikslas pateikti saugumo valdymo įrankius, kuriuos naudojant kartu galima suteikti tinklo administratoriams išsamų vaizdą apie esančio tinklo būklę.
5. „Honeyd“ – tai yra mažo dydžio servisas, kuris sukuria virtualiuosius mazgus tinkle. Šių mazgų paskirtis imituoti esantį tinklą ar jo mazgus, taip apsaugant tikrąjį tinklą nuo įsibrovimų.
6. „Samhain“ – tai tinklo mazgo paremta ĮAS, kuri suteikia failų integralumo patikrinimą ir registravimo failų analizę ir stebėseną, prievadų stebėseną ir slaptų procesų paleidimą.

Tinklo saugumas yra vienas iš prioritetų bet kokio tipo organizacijai. Naudojant ĮAS įmonės gali suprasti, analizuoti, atpažinti ir suvaldyti piktavališkus veiksmus tiek įmonės viduje, tiek stebint srautą iš išorinio tinklo. Atvirojo kodo programinės įrangos didžiausias privalumas – programinis kodas, leidžiantis vartotojams keisti programinius parametrus pagal esamą situaciją ir reikalavimus. Įsibrovimo įranga, kuri gali būti pritaikyta prie esamos tinklo struktūros yra geriausias saugumo užtikrinimas. Kiekviena atvirojo kodo ĮAS turi skirtingas konfigūracijos galimybes ir skirtingai reaguoja į atakas ir perdavimo spartą. Reikia išnagrinėti pasirinktų atvirojo kodo ĮAS pagal konfigūracijos galimybes ir atakų atpažinimą.

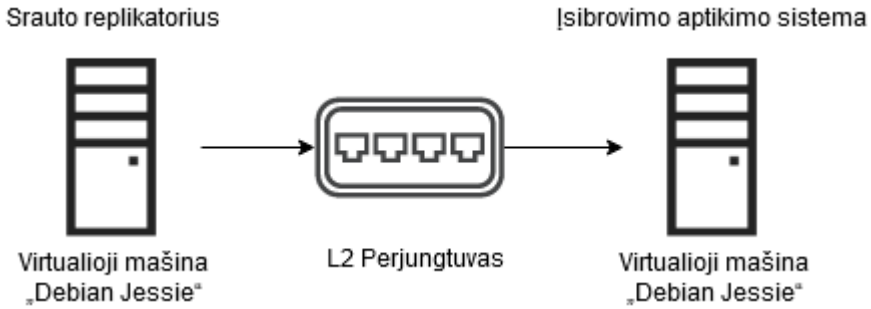
3. TYRIMO METODIKOS KŪRIMAS

Matavimo aplinka susideda iš tyrimų apibrėžimų, vykdomų žingsnių, naudojamų įrankių ir testuojamos programinės įrangos analizės. Šiame skyriuje apibrėžiama tyrimų metodika, kuria vadovaujamasi baigiamajame darbe. Pateikiami sąvokų apibrėžimai, tyrimų schemos, eksperimentų principai ir žingsniai, atliekant įsibrovimo aptikimo programinių paketų analizę.

3.1. Tyrimų matavimo schemos

Tyrimuose analizuosime ĮAS gebėjimą aptikti atakas, paketų apdorojimo greitį, paketų atmetimą, aparatinės įrangos resursų sunaudojimą. Testavimo aplinka bus paremta „VMWare“ ir „Docker“ virtualizavimo įrankiais. Siekiant, kad ĮAS sėkmingai apdorotų duomenų rinkinio paketus – naudojamasi ĮAS PCAP failų skaitymo režimu. Esant įjungtam failo skaitymo režimui, ĮAS maksimaliu leidžiamu pajėgumu skaito paketus, tokiu būdu panaikinama paketų atmetimo galimybė, kuri susidarytų, siunčiant paketus iš kito mazgo. Tyrime analizuojami ĮAS generuojami įspėjimų pranešimai, atpažįstant ataką. Jeigu įspėjimų pranešimai atitinka generuojamą ataką ir laiką – žymima, kad ĮAS atpažino ataką, jeigu ne – žymima, kad ĮAS neatpažino atakos.

Pirmame, sisteminių parametrų tyrime, yra skaitomas duomenų failas ir yra stebimas paketų apdorojimo greičio kitimas. Aparatinė įranga, kurioje veikia ĮAS turi „Intel(R) Core(TM) i5-9500“ procesorių su 6 branduoliais, 64 GB darbinės atminties, operacinė sistema yra „Ubuntu 18.04“. Našumo tyrimo loginė schema pateikiama 3.1 pav.



3.1 pav. Našumo tyrimo loginė schema

Našumo tyrime, aplinką sudaro dvi virtualiosios mašinos, veikiančios tame pačiame „VMware“ fiziniame serveryje. Teorinis tinklo kanalo pralaidumas tarp virtualiųjų mašinų yra 10 Gbps. Virtualioji mašina, kuri generuoja perdavimo srautą, turi „Intel(R) Xeon(R) CPU E5645“ procesorių su 1 virtualiuoju branduoliu, 4 GB darbinės atminties, operacinė sistema „Debian Jessie“, tinklo plokštė – „VMXNET3“. Virtualioji mašina, kurioje veikia ĮAS, turi „Intel(R)

Xeon(R) CPU E5645“ procesorių su 4 virtualiaisiais branduoliais, 16 GB darbinės atminties, operacinė sistema „Debian Jessie“, tinklo plokštė – „VMXNET3“. Virtualiųjų mašinų palyginimas pateikiamas 3.1 lentelėje.

3.1 lentelė. Virtualiųjų mašinų specifikacijos palyginimas

Parametras	Srauto replikatorius	Įsibrovimo aptikimo sistema
Modelis	„HP Proliant DL380 G3“	
CPĮ	„Intel(R) Xeon(R) CPU E5645“	
RAM	4 GB	16 GB
Tinklo sąsajos korta	„VMXNET3“	

3.2. Duomenų rinkinių apžvalga

Gilinantis į ĮAS perspėjimų pranešimų svarbą, reikalingos sąlygos, kurios būtų standartizuotos, susistemintos, galinčios būti pakartotos daugelį kartų. ĮAS analizės tyrimuose rekomenduotina naudoti susistemintus duomenų rinkinius.

Vienas pirmųjų sistemintų rinkinių buvo „DARPA“ duomenų rinkinys, kuris buvo sukurtas MIT „Lincoln“ laboratorijoje. Šie duomenų rinkiniai dar žinomi kaip „DARPA’98“, „DARPA’99“ [22][23]. „DARPA’98“ yra 7 GB dydžio failas, kuriame yra 7 savaičių bandomasis srautas, vykstantis simuliuotame tinkle, kuriame vyksta atsitiktinės atakos. „DARPA’99“ duomenų rinkinyje pirma ir trečia savaitės neturi atakų, jos skirtos apmokyti ĮAS, grįstas anomalija. Antros savaitės generuotame sraute buvo vykdomos atakos. Atakų srautas paimtas iš „DARPA’98“ duomenų rinkinio, kartu pridėtos ir naujos atakos. Likusios dvi savaitės duomenų rinkinyje yra testavimo savaitės.

Kitas populiarus duomenų rinkinys yra „KDDCUP’99“. Jis buvo sukurtas konferencijos „The Fifth International Conference on Knowledge Discovery and Data Mining“ konkursui, kuris reikalavo sukurti modelį, gebanti atskirti „gerus“ ir „blogus“ srautus [7]. Duomenų rinkinyje yra 4,9 milijonų srautų, kurie pažymimi kaip „normalus“ arba „atakuojantis“ srautas. Atakos buvo imituojamos kariniame tinkle.

„DARPA“ ir „KDDCUP“ duomenų rinkiniai ilgą laiką buvo laikomi standartu, testuojant įsibrovimo aptikimo sistemas, tačiau keičiantis IT industrijai – kartu keičiasi ir užpuolikų taktikos. Nėra tikslinga tirti šiuolaikines įsibrovimo aptikimo sistemas su senais duomenimis. Yra praėjęs ilgas laiko tarpas nuo „DARPA“ ir „KDDCUP“ duomenų rinkinių sukūrimo. „DARPA“ ir „KDDCUP“ negali tiksliai reprezentuoti šiuolaikinio srauto arba turi būti naudojami su papildomais duomenų rinkiniais, ką pastebime analogiškuose tyrimuose [30][21]. Todėl tyrimui

bus naudojami naujesni duomenų rinkiniai, gebantys tiksliau apibrėžti šiuolaikinį tinklo srautą ir generuojamas atakas. Apžvelgiame naujesnius nekomercinius duomenų rinkinius.

„CDX“ sukurtas Jungtinių Valstijų karinės akademijos (angl. *United States Military Academy*) 2009 m. duomenų rinkinys buvo sukurtas tinklo varžybose. Duomenų rinkinį sudaro srautas iš saityno programų, pašto serverių, DNS užklausų, failų parsisiuntimo užklausų. Atakos generuotos naudojant tokius įrankius kaip „Nikto“, „Nessus“ ir „WebScarab“. Duomenų rinkinys gali būti naudojamas testuoti ĮAS taisyklių rinkinius, tačiau rinkinyje per mažai duomenų ir skirtingų atakų [30].

„Kyoto“ duomenų rinkinys sukurtas Kyoto universitete 2009 m. Duomenų rinkinio srautas surinktas iš avilių (angl. *Honeypots*). Šie serveriai skirti pritraukti atakas, siekiant geriau suprasti generuojamų atakų charakteristikas ir pobūdį. Pagrindinis duomenų rinkinio trūkumas – nėra įvairaus normalaus srauto. Atakos metu normalus srautas generuojamas tik per DNS užklausas ir prisijungimus prie pašto serverių. Sumažėja klaidingai teigiamų pranešimų tikimybė, todėl duomenų rinkinys nėra tinkamas testavimui, esant mišriam srautui [30].

„UMASS“ duomenų rinkinys sukurtas Masačusetso universitete 2011 m. Duomenų rinkinį sudaro tinklo srautai – failai surinkti iš belaidžio tinklo įrenginių. Duomenys generuoti naudojantis tik TCP protokolo paremtu failo atsisiuntimo scenarijumi. Duomenų rinkinyje nėra pakankamo normalaus ir atakos srauto [30].

Daugelis prieinamų tinklo srautų yra organizacijų vidinių tinklų pavyzdžiai. Atsiranda dvi problemos: privatumo problema – srauto duomenys yra anonimiški ir neatspindi dabartinių tendencijų, duomenų rinkiniams trūksta statistinių charakteristikų. Šie trūkumai yra priežastys, kodėl dar nėra tobulo duomenų rinkinio [40]. Tyrėjai turi naudotis duomenų rinkiniais, kuriuos jie gali naudoti ir analizuoti laisvai, tačiau dažnai duomenų rinkiniai būna neoptimalūs. Ieškant tinkamų duomenų rinkinių pasirinkti du rinkiniai, naudojami tyrime. Pateikiami duomenų rinkiniai pasižymintys statistinėmis charakteristikomis, leidžiantys tiksliai analizuoti atakų laiką ir veiksmus.

„CICIDS“ sukurtas Naujojo Brusviko universitete 2017 m. Duomenų rinkinys skirtas ĮAS analizei. Sukurdami šį duomenų rinkinį autoriai motyvavo tuo, kad IT industrija nuolat keičiasi, todėl užpuolikai yra priversti taikyti vis naujus įsibrovimo būdus į aukos tinklą [2]. Duomenų rinkinys simuliuoja mažą verslo biurą. Ši aplinka sudaryta iš kelių vartotojų ir vidinių servisų, tokių kaip pašto ar saityno serveriai. „Python“ programinės kalbos kodas simuliuoja vartotojų srautą, naršymą internete, siunčiamus ar gaunamus elektroninius laiškus. Siekiant kuo realistiškesnio

srauto, simuliuojami vartotojai turi kintamas darbo valandas, pertraukų laiką, taip pat skirtingas atliekamas užduotis [40].

„ISCX“ sukurtas Naujoje Brusviko universitete 2012 m. duomenų rinkinys remiasi realių vartotojų srautu, siekiant sukurti sisteminį informacijos rinkinį [6]. Šis rinkinys susideda iš HTTP, SMTP, SSH, IMAP, POP3 ir FTP protokolo srautų. Kiekviename iš rinkinio failų sukurti realaus srauto profiliai, kurie atspindi šių dienų aktualijas ir žmonių elgseną. Kiekvienam iš šių scenarijų yra atitinkamai sugeneruotos atakos, kurios gali pasitaikyti šiuolaikiniame pasaulyje. Siekiant tikslingai simuliuoti vartotojų elgseną, universiteto darbuotojų srautas buvo suskirstytas į atskirus profilius, kurie pasižymi tam tikromis savybėmis [6]. Atakų scenarijai buvo sukurti ir įvykdyti, remiantis realaus pasaulio atakų pavyzdžiais.

„ISCX“ duomenų rinkinį sudaro sužymėti tinklo srautai (angl. *Flows)*, failai su visais paketų duomenimis PCAP formatu. Duomenų rinkinį sudaro 7 dienos tinklo srauto, kuriame yra normalus srautas ir kenkėjiškas. Naudojantis skirtingais duomenų srautais bus galima sistemingai palyginti kaip priklauso rezultatai tarp skirtingų įsibrovimo aptikimo sistemų ir duomenų.

Taip pat pasirinktas duomenų rinkinys „CIC DoS“ sukurtas 2017 m. Duomenų rinkinys turi normalų ir piktavališką srautą. Rinkinyje užfiksuotos 4 rūšių atakos, kurios vykdytos naudojantis skirtingais srauto generavimo įrankiais. Iš viso išsaugoti 8 skirtingi DoS srautai, vykdyti prieš 10 WEB serverių [1].

„CICIDS“ duomenų rinkinys susideda iš PCAP failų, paimtų per visą darbo savaitę (nuo pirmadienio iki penktadienio) surinkto tinklo srauto. Srautas buvo gautas darbo valandomis nuo 9 val. iki 17 val. PCAP failų specifikacija pateikiama 3.2 lentelėje.

3.2 lentelė. „CICIDS“ duomenų rinkinio failų duomenys

Savaitės diena	Srauto tipas	Dydis, GB
Pirmadienis	Normalus srautas	11
Antradienis	Atakos ir normalus srautas	11
Trečiadienis	Atakos ir normalus srautas	13
Ketvirtadienis	Atakos ir normalus srautas	7,8
Penktadienis	Atakos ir normalus srautas	8,3

„ISCX“ duomenų rinkinys susideda iš PCAP failų, paimtų per 7 dienas – nuo penktadienio iki ketvirtadienio. Srautas buvo rinktas 8 val. per dieną. PCAP failų specifikacija pateikiama 3.3 lentelėje.

3.3 lentelė. „ISCX“ duomenų rinkinio failų duomenys

Savaitės diena	Srauto tipas	Dydis, GB
Penktadienis	Normalus srautas	16,1

Šeštadienis	Normalus srautas	4,22
Sekmadienis	Infiltravimas į vidinius sistemos servisus ir normalus srautas	3,95
Pirmadienis	HTTP DoS ir normalus srautas	6,85
Antradienis	DoS ir normalus srautas	23,4
Trečiadienis	Normalus srautas	17,4
Ketvirtadienis	Jėgos metodo ataka SSH protokolu ir normalus srautas	12,3

3.3. Įsibrovimo aptikimo sistemų tikslumo tyrimas

3.3.1. Tyrimo parametrai

Pagrindinis tyrimo tikslas – palyginti skirtingų ĮAS gebėjimą atpažinti atakas, esant mišriam srautui. Tyrime tiriama ar ĮAS sėkmingai atpažino ataką. Darbe remiamasi keturiomis ĮAS pranešimų būsenomis, kurios apibūdina sistemos tikslumą [27][21].

Tikrai teigiamas pranešimas vyksta tuomet, kai ĮAS identifikuoja tam tikrą veiksmą ar įvykį kaip ataką, kuri iš tikrųjų ir yra ataka prieš saugomą sistemą. Kitaip tariant, tikrai teigiamas pranešimas yra būsena, kuri nurodo sėkmingai identifikuotą ataką.

Tikrai neigiamas pranešimas nurodo priešingą procesą – tai būsena, kai ĮAS identifikuoja procesą kaip tinkamą ir leidžiamą ir tokiu būdu jį ignoruoja. Šie pranešimai kelia didžiausią grėsmę, nes ataka nėra atpažįstama ir tai gali sukelti sistemoje saugumo incidentą.

Problemos kyla, kai ĮAS generuoja klaidingai teigiamus perspėjimo pranešimus. Tai būsena, kada ĮAS normalų srautą palaiko pavojingu sistemai, kitaip tariant, klaidingai teigiamas pranešimas yra netikro pavojaus (angl. *False Alarm*) pranešimas. Klaidingai neigiama būsena sukelia didžiausią pavojų saugomai sistemai. Ši būsena nutinka tada, kai ĮAS piktavališką srautą laiko tinkamu ir jį ignoruoja, taip kompromituodama sistemą. Nors sistemos administratorius gali nesunkiai atpažinti, kur tikras, o kur netikras pavojus, tačiau didelis kiekis klaidingų pranešimų, sukeltų iš vienos taisyklės, gali trukdyti administratoriui atpažinti tikrą ataką [21].

Klaidingi teigiami pranešimai yra rimta problema ĮAS, nes jie mažina ĮAS našumą. Dėl šios priežasties vienas iš pagrindinių iššūkių, kuriant ĮAS, yra klaidingų perspėjimų sumažinimas, o ne gebėjimas aptikti atakas. Informaciniuose šaltiniuose pateikiami būdai ir technikos, siekiant sumažinti klaidingai teigiamų pranešimų skaičių [21]. Klaidingi perspėjimo pranešimai arba klaidingai teigiami perspėjimo pranešimai yra svarbiausias matavimų kriterijus, analizuojant ĮAS tikslumą [21]. Šių pranešimų kiekis nusakys, kaip klaidingai arba teisingai ĮAS atpažįsta atakas. Tyrimo matavimo kriterijai:

1. Atakos atpažinimas

2. KT (klaidingai teigiamas) perspėjimo pranešimas
3. TT (tikrai teigiamas) perspėjimo pranešimas

3.3.2. Tyrimo konfigūracija

Tyrime analizuojamas programinių paketų: „Snort“, „Suricata“, „Zeek“ gebėjimas nustatyti ataką. Tyrime naudojami laisvai prieinami taisyklių rinkiniai „Snort“ ir „Suricata“, skirti programiniams paketams. „Talos“ ir „ET Open“ taisyklės yra sukurtos tinklo saugumo valdymo komandų, todėl taisyklės būna ištestuotos ir patvirtintos, kad bendruomenė galėtų naudotis. „Zeek“ nenaudoja taisyklių rinkinių, todėl „Talos“ ir „ET Open“ taisyklės negali būti pritaikytos šiai ĮAS. 3.4 lentelėje pateikiami taisyklių duomenys ir ĮAS versijos.

3.4 lentelė. ĮAS programinės įrangos versijos ir taisyklių rinkinių duomenys

ĮAS	Versija	Taisyklių rinkinys	Taisyklių skaičius	Taisyklių sukūrimo data
„Snort“	2.9.15	„Talos“	18172	2020-02-14
„Suricata“	5.0.2	„ET Open“	15141	2020-02-16
„Zeek“	3.1.2	-		

Tyrime koncentruojamasi į mišrųjį srautą, todėl PCAP failai, kurie turėjo tik normalų srautą, be atakų, nebuvo tirti. Tyrimai buvo atliekamai, paleidus programines įrangas, nurodant joms skaityti srauto duomenis iš PCAP failų.

- „Snort“ paleidimo komanda: `snort -c /etc/snort/snort.conf -r (failo pavadinimas)`
- „Suricata“ paleidimo komanda: `suricata -c /etc/suricata/suricata.yaml -r (failo pavadinimas)`
- „Zeek“ paleidimo komanda: `zeek -r (failo pavadinimas)`

3.4. Sistemos parametrų įtakos įsibrovimo aptikimo sistemų našumui tyrimas

Žinoma, kad aparatinės įrangos parametrai ir konfigūracija daro įtaką programinės įrangos veikimui. Priimta manyti, kad kuo daugiau resursų skiriama sistemos procesui, tuo galima tikėtis didesnio efektyvumo. Tyrimo tikslas – išanalizuoti, kokia yra sistemos parametrų įtaka ĮAS skaitymo režimui.

3.4.1. Tyrimo parametrai

Taisyklių rinkinį sudaro keliolika tūkstančių taisyklių, kurias ĮAS reikia apdoroti, priimant kiekvieną paketą. Galima manyti, kad kuo mažiau taisyklių ir kuo taisyklės efektyviau parašytos, tuo didesnis ĮAS našumas. Našumas mažėja dėl papildomo darbinės atminties kiekio ir procesoriaus resursų. Reikia atsižvelgti į tai, kad skirtingų taisyklių naudojimas taip pat gali daryti įtaką našumui. Pagrindinis ĮAS paketų apdorojimo sisteminis resursas – CPĮ, todėl laikome, kad kuo daugiau skirsime CPĮ resursų ĮAS, tuo labiau didės ĮAS efektyvumas.

Pagrindinė tyrimo užduotis – apžvelgti CPĮ, taisyklių rinkinių ir darbinės atminties įtaką ĮAS našumui skaitymo režime. Pateikiami sistemos parametrų įtakos tyrimo punktai.

Tyrimo tikslas – apžvelgti aparatinės įrangos, konfigūracijos, taisyklių rinkinių įtaką ĮAS našumui. Tiriamas scenarijus yra vienas iš ĮAS naudojamų būdų, kada atliekama srauto analizė ne realiame laike. Šis būdas dažniausiai naudojamas tinklo analizei po incidento. Pagrindiniai metodo limitai yra aparatinės įrangos komponentai: CPĮ, RAM, kietojo disko našumas [24]. Daroma prielaida, kad kuo parametrai geresni, tuo didesnis našumas.

3.4.2. Tyrimo konfigūracija

Tiriama CPĮ branduolių, darbinės atminties kiekio ir taisyklių konfigūracijos įtaka ĮAS, sistema dirbo skaitymo režime. Testuojami programiniai paketai: „Snort“, „Suricata“ ir „Zeek“. „Snort“ ir „Suricata“ naudoja „ET Open“ ir „Talos“ taisyklių rinkinius, „Zeek“ naudoja standartinius scenarijus. Testas atliekamas tris kartus, siekiant išvengti programinės įrangos anomalijų ir užtikrinti rezultatų kartotinumą. Matavimo kriterijus – paketai per sekundę, tai kiekis paketų, kuriuos gauna, dekoduoja ir išanalizuoja pagal taisykles ĮAS.

3.5. Įsibrovimo aptikimo sistemų našumo tyrimas

Našumas yra svarbi ĮAS programinės įrangos dalis. ĮAS atlieka sunkias paketų apdorojimo, grupavimo, atmetimo operacijas, kurios reikalauja resursų. Būna atvejų, kada sistema gali aptikti ataką, tačiau yra limituota, kiek paketų gali apdoroti vienu metu. Atmetant analizuojamus paketus, ĮAS efektyvumas krenta, nes nėra taikomas visas paketo apdorojimo procesas. Informaciniuose šaltiniuose ĮAS dažnai tiriama, esant skirtingoms platformoms, operacinėms sistemoms, konfigūracijos parametrams, siekiant atrasti didžiausią našumą [18][36]. Informaciniuose šaltiniuose atlikti tyrimai rodo, kad perduodamas srautas daro įtaką ĮAS našumui, todėl tyrimo metu perduodamas srautas turi nekisti [20].

Tyrimo tikslas – palyginti skirtingas ĮAS ir išanalizuoti našumą. Atrasti ribą, kada ĮAS atmeta paketus, įvertinti konfigūracijos įtaką paketų praradimui. Apžvelgti skirtingas srauto apdorojimo konfigūracijos galimybes, kurias palaiko programinė įranga. Palyginti tarpusavyje skirtingas našumo konfigūracijas ir pateikti apibendrintus rezultatus.

3.5.1. Įsibrovimo aptikimo sistemų našumo parametrai

Tiriant programinės įrangos našumą, reikalingas našumo parametrų pokyčių stebėjimas. Pagal parametrų pokyčius galime įvertinti konfigūracijos teigiamą arba neigiamą įtaką ĮAS našumui. Konfigūracijos pakeitimai keičiami, atsižvelgiant į pagrindinį ĮAS funkcionalumą – paketų analizę. Analogiškuose tyrimuose stebimi parametrai, kurie tiesiogiai priklauso nuo ĮAS našumo [36]. Kiekviename tyrimo punkte fiksuojami parametrai:

1. Perdavimo spartos greitis – virtualiosios mašinos perdavimo sparta į virtualiąją mašiną, kurioje veikia ĮAS.
2. Atmestų paketų kiekis – ĮAS atmestų paketų kiekis po paketų priėmimo iš tinklo sąsajos.
3. Priimtų paketų kiekis – ĮAS dekoduotų ir išanalizuotų paketų kiekis.
4. CPĮ apkrova – procesoriaus suminė apkrova per esamus branduolius.

Kiekviename tyrimo punkte parametrų pokyčių rezultatai grupuojami, išvedamas rezultatų vidurkis. Iš rezultatų bus galima daryti išvadas apie kiekvienos ĮAS našumą ir pokytį nuo priimamų duomenų spartos ir konfigūracijos.

3.5.2. Tyrimo konfigūracija

3.5 lentelė. Programinių paketų pavadinimai ir jų versijos

Programinės įrangos pavadinimas	Versija
„Debian“ operacinė sistema	10.3
„Docker“ programinis paketas	19.0.3
„TCPReplay“ programinis paketas	4.3.1
PF_RING tinklo modulis	7.6.0
„HyperScan“ taisyklių palyginimo algoritmas	5.2.1
„Snort“ programinis paketas	2.9.15
„Suricata“ programinis paketas	5.0.2
„Zeek“ programinis paketas	3.1.2

„Snort“ ir „Suricata“ naudoja „ET Open“ taisyklių rinkinį. „Zeek“ naudoja visus standartiškai suteikiamus scenarijus. Rezultatai gali ypač skirtis, priklausomai nuo generuojamo srauto.

Tiriant srautas generuotas 60 sekundžių, didinant perduodamų duomenų spartą iki 3 Gbps. Generuojamas srautas yra realus, nes atkuriami paketai iš „CIC DoS“ duomenų rinkinio PCAP failo. Kadangi siunčiamas srautas yra kintantis, o ne vien pastovūs TCP ar UDP lygmens paketai, todėl atkuriamo srauto sparta tampa limituota. Pirminiai tyrimai parodė, kad, atkuriant „CIC DoS“ duomenų failą, pasiekiamas 3 Gbps srauto limitas dėl srauto charakteristikų.

Siekiant, kad atkuriamas duomenų rinkinio srautas pasiektų ĮAS veikiančiame mazge, paketų IP ir MAC adresai turi būti perrašyti. Paketų perrašymas vykdomas „tcprewrite“ komanda, kuri įrašoma kartu su „TCPReplay“ programiniu paketu. Perrašyti duomenų rinkinio gavėjų ir šaltinių IP, MAC adresai į virtualiųjų mašinų IP ir MAC adresus. Paketų perrašymo komanda:

```
tcprewrite --infile=AppDDos.pcap --outfile=AppDDos.pcap --
srcipmap=0.0.0.0/0:10.128.67.63 --dstipmap=0.0.0.0/0:10.128.67.64 --enet-
smac=08:00:27:67:10:68 --enet-dmac=08:00:27:67:10:68 --fixcsum
```

Perrašytas PCAP failas yra atkuriamas, naudojant „TCPReplay“ programinį paketą. Naudojama srauto atkūrimo komanda, kai nurodome perdavimo srauto spartą 50 Mbps ir laiką 60 s.

```
tcpreplay -i ens192 --mbps 50 --netmap --duration 60 -K --loop 100 --unique-ip
AppDDos.pcap
```

3.5.3. Veiksniai, darantys įtaką įsibrovimo aptikimo sistemų našumui

Našumo tyrimo tikslas – ištirti atmestų paketų kiekį ir CPĮ naudojimą, didėjant perdavimo srautui. Pagal atliktus ankstesnius tyrimus ir apžvalgą galima sudaryti prielaidas, kurie veiksniai daro didžiausią įtaką ĮAS našumui. Pagrindiniai faktoriai:

1. Programinės įrangos architektūra ir įdiegimas:
   1.1. Atpažinimo algoritmas
   1.2. Programinės įrangos versija
   1.3. Programinės įrangos konfigūracijos parametrai
   1.4. Įjungtų modulių kiekis ir našumas
2. Operacinės sistemos tinklo jungties paketų surinkimo modulio efektyvumas:
   2.1. PCAP
   2.2. AF_PACKET
   2.3. PF_RING
3. Aparatinės įrangos našumas:

3.1. CPĮ

3.2. Tinklo sąsajos sąsaja

4. Atpažinimo taisyklės:

4.1. Taisyklių kiekis

4.2. Taisyklių našumas

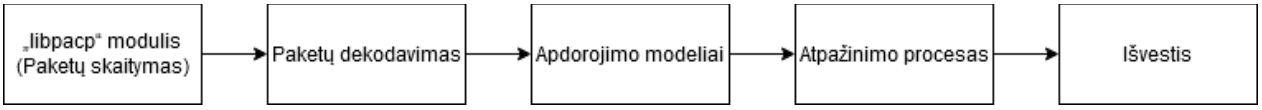
3.6. Tyrimo metu naudojama programinė įranga

3.6.1. „Snort“ programinė įranga

„Snort“ atvirojo kodo programinis paketas sukurtas 1998 metais įkūrėjo ir buvusio „Sourcefire“ įmonės CTO – Martino Roescho,. „Snort“ dabar yra kuriama ir tobulinama „Cisco“ kompanijos, kuri nusipirko „Sourcefire“ kompaniją 2013 metais. Programinis paketas „Snort“ yra suderinamas su „Ubuntu“, „Debian“, „Fedora“, „CentOS“, „FreeBSD“ distribucijomis ir „Windows“ platformomis [9].

„Snort“ paketo apdorojimo schema susideda iš dviejų dalių: atpažinimo mechanizmo ir pritaikomo taisyklių rinkinio. Priešprocesiniai komponentai, taisyklės ir pranešimų išvestis yra „Snort“ sudedamosios dalys, kurios gali būti individualiai valdomos.

Atvaizduojama „Snort“ programinės įrangos schema, kurioje pateikiama paketo apdorojimo eiga programinėje įrangoje 3.2 pav.



3.2 pav. „Snort“ programinės įrangos paketo apdorojimo seka [8]

Kiekvienas „Snort“ mechanizmo paketų apdorojimo žingsnis atlieka savąją funkciją. Galime apibendrinti eigos funkcijas:

1. Paketai, priimti tinklo plokštės, yra nuskaitomi „Linux“ paketų skaitymo modulio PCAP.
2. Paketų dekoderis atlieka OSI (angl. *Open Systems Interconnect*) antro lygmens kadrų analizę, dekoduoja kadrų antraštes, skaičiuoja kontrolines sumas. Aptikus anomaliją gautame kadre, paketas yra atmetamas.
3. Apdorojimo moduliai atlieka pirminius paketų patikrinimo žingsnius – gali sukelti perspėjimo pranešimą arba visiškai atmesti paketą. Pagrindiniai apdorojimo moduliai [9]:
    a. „Frag3“ apdorojimo modulis atlieka pirminę patikrą nuo fragmentuotų paketų. Paketų fragmentavimas yra viena iš technikų, kaip įsibrovėliai gali paslėpti informaciją, įsilaužimo kodą, fragmentuotuose paketuose, apeinant ĮAS sistemų apsaugą.

b. „Stream5“ atlieka TCP ir UDP protokolų sesijų būsenų stebėseną. Aptinka tokias anomalijas kaip SYN paketo gavimas už TCP paketų siuntimo lango ribos ar kontrolinės sumos klaidas.
c. „sfPortscan“ modulis sukurtas aptikti pirminius atakos požymius. Pirminėje fazėje atakos sukėlėjo atliekama analizė apie aukos sistemą. Gaunama informacija apie tai, kokius tinklo protokolus ir prievadus mazgas palaiko. „Nmap“ programinė įranga atlieka mazgų skenavimą ir informacijos rinkimą. „sfPortscan“ apsaugo nuo „nmap“ programos informacijos rinkimo ir generuoja perspėjimo pranešimus.
d. HTTP „Inspect“ modulis atlieka duomenų buferio dekodavimą. Iš dekoduotų duomenų suranda HTTP antraštes, normalizuoja laukus.

4. Atpažinimo dalyje atliekamas svarbiausias žingsnis – šabloninių taisyklių pritaikymas. Taisyklės turi šablonus, pagal kuriuos galima atpažinti atakos srautą. Galimo incidento pranešimai generuojami į vartotojo sąsają.

„Snort“ palaiko tik vieną giją, todėl gali naudoti tik vieną CPĮ branduolį veikimo metu. N Yra nauja „Snort3“ Beta versija, kuri palaiko keletą gijų vienu metu, tačiau ji nėra pakankamai stabili tyrimams atlikti, todėl jos analizės buvo atsisakyta. Siekiat išspręsti vienos gijos problemą, „Snort“ programos procesas gali būti paleistas kelis kartus, taip paketų analizė padalinama per veikiančius procesus. Keleto procesų paleidimui papildomai reikalinga tinklo sąsajos biblioteka – PF_RING.

Tyrimo metu orientuojamasi į „Linux“ operacinės sistemos aplinką, todėl instaliavimo žingsniai pateikiami „Linux“ aplinkoje „Ubuntu“ distribucijoje [9]. Šiai distribucijai nėra sukurtos lengvai prieinamos talpyklos, todėl programinę įrangą reikia sukompiliuoti. Tyrimo supaprastinimui naudojami „Docker“ programos vaizdai, kurie sukuria „Snort“ programos vaizdą, iš kurio paleidžiamas programos konteineris. „Docker“ vaizdo programinio kodo ištraukos pateikiamos priede.

„Snort“ svarbi dalis yra DAQ (angl. *Data Acquisition Library*) biblioteka, kuri pakeičia tiesiogines kreipties funkcijas į PCAP paketų priėmimo modulį [10]. Pagrindinė paskirtis – palengvinti modulių įjungimą ir išjungimą, neatliekant papildomos „Snort“ konfigūracijos. Pagrindiniai DAQ bibliotekos naudojami moduliai:

1. PCAP modulis – standartinis DAQ bibliotekos pasirinkimas. Atlieka tiesioginius funkcijos kvietimus į PCAP modulį.
2. AF_PACKET modulis – modulis gebantis išskirti papildomą atminties kiekį paketų analizei.

3. „NFQ“ – IPTABLES paketų apdorojimo modulis.
4. „IPQ“ – senesnis IPTABLES paketų apdorojimo modulis.

„Snort“ programinis paketas gali būti sukonfigūruotas trimis režimais [9]:

- Priėmimo režimas (angl. *Sniffer Mode*) – skaitomi paketai iš tinklo ir nuolat atnaujinami konsolėje.
- Paketų registratoriaus režimas (angl. *Packet Logger Mode*) – registruoja paketus diske.
- Tinklo įsibrovimo aptikimo sistema (angl. *Network Intrusion Detection System*) – atlieka srauto atpažinimą ir analizę. Labiausiai kompleksiškas ir daugiausiai konfigūruojamas režimas.

„Snort“ standartinė paleidimo komanda susideda iš parametrų: tinklo adreso, žurnalo saugojimo vietos, *snort.conf* konfigūracijos failo. Komandos pavyzdys:

```
snort -d -h 192.168.1.0/24 -l ./log -c snort.conf
```

Komanda paleis „Snort“ programinį paketą paprasčiausiu režimu, į žurnalą bus išsaugomi paketai, kurie atitiks šablonus specifikuotus *snort.conf* faile [9].

„Snort“ programinė įranga naudoja aprašytas taisykles ir taisyklių rinkinius. „Snort“ dokumentacija pateikia kelis būdus kaip atnaujinti programinės įrangos taisyklių duomenų bazę. „Snort“ pateikia tris taisyklių rinkinius.

Pirmasis bendruomenės taisyklių rinkinys yra sertifikuotas „Talos“ saugumo komandos rinkinys. „Snort“ bendruomenės nariai patys kuria taisykles ir atnaujina taisyklių rinkinį kasdien. Kasdien vartotojų atnaujinamas rinkinys yra naudingas, kai reikia aptikti naujai pasirodžiusias atakas. Tačiau dėl prastesnės taisyklių kokybės didėja klaidingai teigiamų pranešimų tikimybė.

Antrasis taisyklių rinkinys skirtas užregistruotiems vartotojams. Taisyklių rinkinys yra kurtas, ištestuotas ir patvirtintas „Talos Security Intelligence and Research Team“ organizacijos, tačiau atsilieka 30 dienų nuo komercinio varianto, todėl pasirodžius nulinės dienos atakoms (angl. *Zero-day*) gali jų neaptikti. Registruotų vartotojų taisyklių rinkinį kuria „Talos“ saugumo komanda, todėl klaidingai teigiamų pranešimų tikimybė mažėja. Bendruomenės ir registruotų vartotojų taisyklių rinkinys suteikiamas nemokamai.

Trečiasis yra prenumeratorių taisyklių rinkinys (angl. *Subscriber*), tai yra visas taisyklių rinkinys, atnaujinamas kas savaitę arba kai atsiranda nauja grėsmė. Taisyklių rinkinys yra komercinis. Taisyklių rinkiniai valdomi per „PulledPork“ įrankį ar rankiniu būdu, įrašant jas į

„Snort“ direktoriją. „Snort“ programinės įrangos perspėjimo pranešimo formatas dažniausiai atrodo taip [2]:

```
[1:469:1] ICMP PING NMAP [Classification:
Attempted Information Leak] [Priority: 2]: <eth1> {ICMP} 10.120.66.1 ->
10.22.33.106
```

- Pirmasis laužtiniuose skliaustuose esantis skaičius nurodo generatoriaus identifikacijos numerį, kuris nusako „Snort“ komponentą, kuris sugeneravo perspėjimą.
- Antrasis skaičius nurodo „Snort“ identifikacijos numerį, tai tam tikra taisyklių indikacija ir atpažinimas sistemoje.
- Trečiasis skaičius nurodo revizijos identifikacijos numerį, tai tam tikras taisyklės pokyčių žymėjimas.

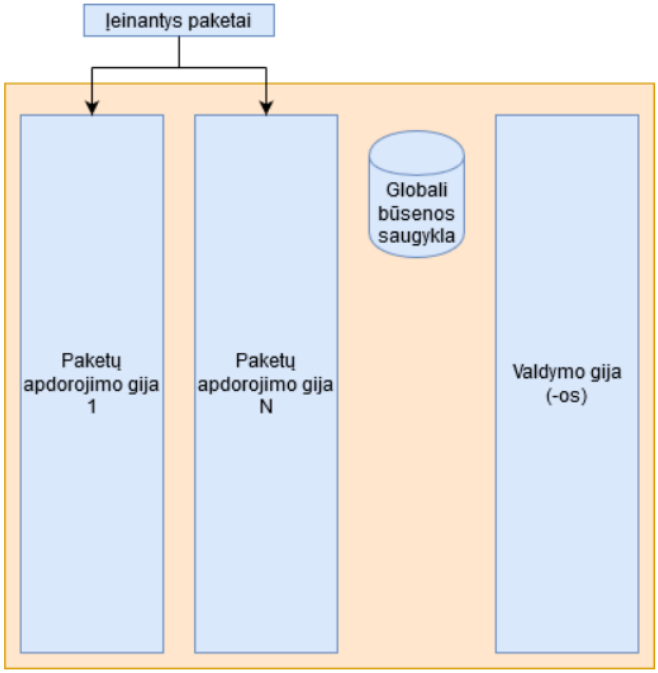
Tyrimo atveju šios indikacijos nekeičia ar kitaip nedaro tiesioginės įtakos tyrimo rezultatams, todėl per daug nėra gilinamasi, kokie identifikacijos numeriai kokiems komponentams yra priskirti.

3.6.2. „Suricata“ programinė įranga

„Suricata“ yra atvirojo kodo produktas, kurį kuria „Open Information Security“ – nepelno siekianti organizacija [11]. Pirminis programos pristatymas įvyko 2010 m. sausio 1 d. „Suricata“ stiprus konkurentas „Snort“ įrankiui, gebantis dirbti įsibrovimo atpažinimo ir prevencijos režimais, turintis aktyvią bendruomenę. „Suricata“ taip pat palaiko populiariausias platformas tokias kaip „Linux“, „MAC“, „FreeBSD“, „Unix“ ir „Windows“.

Tyrimo metu orientuojamasi į „Linux“ operacinės sistemos aplinką, todėl instaliavimo žingsniai pateikiami „Linux“ aplinkoje „Ubuntu“ distribucijoje [11]. Šiai distribucijai nėra sukurtos lengvai prieinamos talpyklos, todėl programinę įrangą reikia sukompiliuoti. Tyrimo supaprastinimui naudojami „Docker“ programos vaizdai, kurie sukuria „Snort“ programos vaizdinį, iš kurio paleidžiamas programos konteineris. „Docker“ vaizdo programinis kodas pateikiamas priede.

Paketų apdorojimas ir eiga yra panaši į „Snort“ , paketai priimami PCAP moduliu, dekoduojami ir į nusiunčiami į atpažinimo variklį, kuriame „Snort“ ir „Suricata“ veikimas išsiskiria. „Suricata“ efektyviausia režimo veikimo schema pateikiama 3.3 pav.



3.3 pav. „Suricata“ programinės įrangos veikimo režimo architektūra [13]

„Suricata“ architektūra paremta keliomis gijomis ir eilėmis. Kelios gijos gali būti aktyvios vienu metu ir atlikti skirtingus paketų analizavimo veiksmus. Paketų persiuntimas į kitą giją atliekamas naudojant eiles. Paketai yra apdorojami vienoje gijoje vienu metu, tačiau keletas paketų gali būti apdorojami „Suricata“ paketų analizės mechanizmo tuo pačiu metu. Viena gija gali turėti keletą modulių, tačiau tik tas vienas modulis gali būti aktyvus vienu metu. Efektyvus paketų paskirstymas tarp gijų leidžia „Suricata“ ĮAS efektyviai naudoti mazgo CPĮ branduolių resursus.

„Suricata“ ,kaip ir „Snort“, perspėjimų pranešimų generavimui naudoja taisyklių šablonus. „Suricata“ suteikia komandinės eilutės įrankį *suricata-update*, kuris parsiunčia naujausias taisykles ir atnaujina esamas. Standartiškai „Suricata“ naudoja „ET Open“ (angl. *Emerging Threats*) [4] taisyklių rinkinį, kuris yra atnaujinamas kas mėnesį.

3.6.3. „Zeek“ programinė įranga

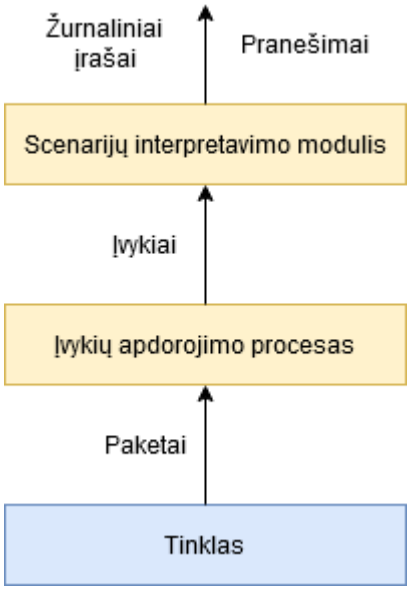
Vern Paxson sukūrė pirminę „Zeek“ versiją dar prieš du dešimtmečius. Pirminiai darbai buvo pradėti 1995 m. „Lawrence Berkeley National Laboratory“. „Berkeley“ laboratorija pradėjo programos paleidimus 1996 m.

„Zeek“ yra pasyvus, atvirojo kodo tinklo srauto analizatorius [17]. Pagrindinė „Zeek“ programinė įranga yra skirta saugumo stebėsenai, kuri tikrina įeinantį srautą dėl įtartinų veiksmų. „Zeek“ pateikia išsamius žurnalus apie tinklo aktyvumą, pradedant nuo kadro antraščių analizės,

baigiant HTTP sesijų URI užklausų analize, MIME tipais, SSL sertifikatais ir DNS užklausomis. Dažnai „Zeek“ programinė įranga naudojama ne tik saugumo tikslams, bet ir našumo tyrimams ir tinklo problemų sprendimams įvykdyti [17].

„Zeek“ suteikia įdiegtą be konfigūracijos (angl. *Out of the Box*) funkcionalumą: failų ištraukimas iš HTTP sesijų, kenkėjiškų programų aptikimą, pranešimą apie programų pažeidžiamumus, populiarių saityno programų identifikaciją, SSH jėgos metodo (angl. *Brute-force*) atakos atpažinimą.

„Zeek“ sistema suteikia standartinius scenarijus, parašytus „Zeek“ programavimo kalba, kurie skirti srauto analizei. „Zeek“ kūrėjai pabrėžia programinės įrangos lankstumą. ĮAS nesiremia tradiciškai taisyklių šablonais, bet remiasi įvykiais ir jų analize [17]. Pabrėžiama, kad „Zeek“ nėra klasikinė šablonu paremta ĮAS, nors ji palaiko standartinį aptikimo pagal šabloną funkcionalumą. „Zeek“ programinė kalba suteikia galimybę atrasti kenkėjišką veiklą, remiantis anomalija ir elgesio analize. „Zeek“ programinė įranga suteikia vartotojui galimybę rašyti savus scenarijus pagal savo saugomą infrastruktūrą, nurodant svarbiausias aplinkos saugomas vietas. 3.4 pav. pateikiamas grafikas, nusakantis „Zeek“ programinės įrangos architektūrą.



3.4 pav. „Zeek“ programinės įrangos architektūra [16]

„Zeek“ sudaro du atskiri komponentai: įvykių variklis (angl. *Event Engine*) ir įvykių interpretatorius (angl. *Event Interpretator*). Įvykių mechanizmas suskirsto ateinančius paketus į atskirus įvykius. Šie įvykiai parodo, kas buvo nustatyta, bet nenurodo įvykio priežasties ir ar įvykis yra kritiškas. Pavyzdžiui, įeinanti HTTP užklausa sukelia „http_request“ įvykį, kuriame nurodomas

IP adresas, prievadai, HTTP versija ir URI užklausa. Tolimesnis komponentas atlieka įvykių interpretavimo funkciją [17].

Scenarijų interpretatorius atlieka įvykių analizę pagal parašytus scenarijus. „Zeek“ programavimo kalba leidžia detaliai išanalizuoti įvykius ir parametrus, pagal kuriuos galima pritaikyti saugumo taisykles – kokių veiksmų imtis.

Orientuojamasi į „Linux“ operacinės sistemos aplinką, todėl instaliavimo žingsniai pateikiami „Linux“ aplinkoje „Ubuntu“ distribucijoje [17]. „Zeek“ programiniai įrangai taip pat naudojamas „Docker“ vaizdas, palengvinantis testavimo pakartotinumą.

3.6.4. Įsibrovimo aptikimo sistemų programinių paketų palyginimas

Pasirinkti tyrimuose naudojami „Snort“, „Suricata“ ir „Zeek“ programiniai paketai. Kiekviena ĮAS aptarta atskirai, šiame poskyryje pateikiama bendra lentelė, kurioje apžvelgiami pagrindiniai parametrai, reikalingi tyrimui. Bendras programinių paketų funkcionalumo palyginimas pateikiamas 3.6 lentelėje.

3.6 lentelė. „Snort“, „Suricata“ ir „Zeek“ programinių paketų funkcionalumo palyginimas

Parametras	„Snort“	„Suricata“	„Zeek“
Atpažinimo metodas	Paremtas šablonu	Paremtas šablonu	Paremtas anomalija
Taisyklių rinkinys	„Talos“, „ET Open“	„Talos“, „ET Open“	Oficialūs scenarijai
Palaikomos CPĮ gijos	Viena	Kelios	Viena
Palaikomi tinklo moduliai	PCAP, AF_PACKET, PF_RING	PCAP, AF_PACKET, PF_RING	PCAP, PF_RING
Atpažinimo algoritmai	„ac“, „ac-bnfa“, „lowmem“, „ac-split“, „intel-cpm“	„ac“, „ac-ks“, „hyperscan“	-
Paketų skaitymo režimas	Taip, keli failai vienu metu	Taip, vienas failas vienu metu	Taip, vienas failas vienu metu

3.6.5. „Docker“ programinis paketas

„Docker“ programinis paketas yra PaaS (angl. *Platform as a Service*), kuris atlieka operacinės sistemos lygmens virtualizaciją. Konteineriai yra izoliuoti vienas nuo kito, kontroliuojantys tik savame konteineryje paleistą programą. Konteineriai turi atskiras bibliotekas, konfigūracijos failus, bendravimas tarp konteinerių vyksta per dedikuotus kanalus, darbo režimus [3].

Naudojantis „Docker“ virtualizacijos galimybėmis tyrimo valdymas supaprastėja, programos būna atskirtos viena nuo kitos ir nedaro įtakos tarpusavyje. Pateikiama „Ubuntu“ distribucijos programinio paketo įdiegimo žingsniai [3]:

1. Atnaujinamos programinių paketų saugyklos

```
sudo apt-get update
```

2. Įdiegiami paketai siekiant naudotis HTTPS protokolu

```
sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    gnupg-agent \
    software-properties-common
```

3. Įdiegiamas oficialus „Docker“ GPG raktas:

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

4. Įdiegiama „Docker“ saugykla pagal distribuciją

```
sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

5. Atnaujinami paketų duomenys

```
sudo apt-get update
```

6. Įdiegiamas „Docker“ programinis paketas

```
sudo apt-get install docker-ce docker-ce-cli containerd.io
```

3.6.6. „TCPReplay“ programinis paketas

„TCPReplay“ yra atvirojo kodo programinė įranga, skirta redaguoti ir atkurti prieš tai užfiksuotą tinklo srautą. Įrankio pradinė paskirtis buvo pakartoti kenksmingų srautų modelius įsibrovimo aptikimo ir prevencijos sistemoms. Dabar funkcionalumas išaugo iki galimybės pakartoti sesijas žiniatinklio serveriams [15]. Programinis įrankis leidžia klasifikuoti srautą į kliento arba serverio srautą, perrašyti 2, 3 ir 4 tinklo sluoksnio paketus. Atsirado galimybė iš naujo perleisti srautą per įvairius tinklo įrenginius. Tai įrankis, periodiškai naudojamas organizacijose, laboratorijose, tinklo tyrimuose [15]. Taip pat naudojamas ugniasienių ir įsibrovimo aptikimo sistemų tyrimuose. Pateikiama „Ubuntu“ distribucijos programinio paketo įdiegimo žingsniai:

1. „TCPReplay“ nėra įprastinis „Linux“ sistemų distribucijos įrankis, todėl jį reikia sukompiliuoti [15].

```
wget https://github.com/appneta/tcpreplay/tarball/master
sudo apt-get install build-essential libpcap-dev
```

2. Išarchyvuojame parsiųstą failą, atidarome aplankalą, paleidžiame kompiliavimo komandas:

```
./configure
make
sudo make install
```

Norint pasiekti didžiausią perdavimo spartą pagal atkuriamo failo galimybes reikia įdiegti „netmap“ papildinį. „Netmap“ modulis skirtas didelio srauto įvesties-išvesties paketų apdorojimui [14]. Dažniausiai naudojama vartotojų programose, kurios apdoroja grynus (angl. *Raw*) paketus. Tokios programos yra įvairūs srauto generatoriai, srauto stebėsenos programos, duomenų kaupikliai, programinės įrangos maršruto parinktuvai [14]. Srauto generavimo modulis yra pakeičiamas į „netmap“ modulį.

„Netmap“ naudojamos technikos, siekiant didesnės spartos:

- Įvesties išvesties API operacijų pakeitimas
- Įrenginių tvarkyklių pakeitimas
- Išsiuntimo priėmimo buferių dydžiai paskirstymas
- Įrenginio atmintyje dedikuota atminties vieta buferiams

Norint, kad „TCPReplay“ palaikytų „netmap“ funkcionalumą reikia „netmap“ modulį įdiegti į aparatinę įrangą. Pateikiama instaliacijos eiga [14]:

1. Parsiunčiama „netmap“ programinio kodo saugykla.

```
git clone https://github.com/luigirizzo/netmap.git
cd netmap
```

2. Konfigūruojame „netmap“ modulį. „Netmap“ modulis bus sukompiliuotas su „Intel“ procesoriaus tvarkyklėmis

```
./configure
```

3. Sukuriamas branduolio modulis.

```
make
```

4. Instaliuojami nauji sukurti moduliai į vartotojo direktoriją.

```
sudo make install
```

Analogiškai įdiegiamas „netmap“ modulio palaikymas „TCPReplay“ programinėje įrangoje. Konfigūracijos komandoje reikia nustatyti „netmap“ modulio repozitorijos vietą ir sukompiliuoti iš naujo „TCPReplay“ programą.

```
./configure --with-netmap=/home/user/netmap
make
sudo make install
```

Naudojantis „netmap“ moduliu „TCPReplay“ programa apeis tinklo tvarkykles ir į tinklo buferius bus rašomi paketai tiesiogiai, tai leis pasiekti maksimalius linijos srautus. Vienintelis limitas – procesoriaus resursai.

Tyrime naudojami papildomi tinklo moduliai. Vienas iš jų PF_RING modulis, kurio standartiškai nėra „Linux“ distribucijose, jį reikia sukompiliuoti kaip atskirą branduolio modulį, norint didesnės ĮAS apdorojimo spartos. Sukompiliuotas modulis įkraunamas į atmintį ĮAS naudojimui. Pateikiama kompiliavimo seka [35]:

```
git clone https://github.com/ntop/PF_RING.git
cd PF_RING/kernel
make
sudo insmod ./PF_RING.ko
cd ../userland
make
```

Našumo tyrime testuojamas „Suricata“ atpažinimo algoritmas „hyperscan“. Šis algoritmas nėra standartiškai įdiegiamas kartu su „Suricata“, todėl jį reikia sukompiliuoti [11]. Pateikiama „hyperscan“ algoritmo kompiliavimo seka.

```
apt-get install cmake ragel
apt-get install libboost-dev
sudo apt-get python-dev libbz2-dev
wget https://dl.bintray.com/boostorg/release/1.66.0/source/boost_1_66_0.tar.gz
tar xvzf boost_1_66_0.tar.gz
cd boost_1_66_0
./bootstrap.sh --prefix=~/tmp/boost-1.66
./b2 install
git clone https://github.com/intel/hyperscan
cd hyperscan
mkdir build
cd build
cmake -DBUILD_STATIC_AND_SHARED=1 -DBOOST_ROOT=~/tmp/boost-1.66 ../
make
sudo make install
```

4. MATAVIMO REZULTATAI

4.1. Įsibrovimo aptikimo sistemų tikslumo tyrimo rezultatai

Tyrimo tikslas – išanalizuoti ĮAS tikslumą, esant mišriam srautui. Šiame skyriuje pateikiami tikslumo tyrimo rezultatai, aparatinės įrangos specifikacija, tiriami duomenų rinkiniai, taisyklių rinkiniai. Pagal matavimo struktūrą ir tyrimo konfigūraciją atliekami tyrimai, rezultatai apibendrinami ir pateikiami lentelėse ir grafikuose.

Apibendrinti tyrimų rezultatai pateikiami atskirose lentelėse pagal duomenų failą. Testuoti duomenų rinkinio failai, kuriuose buvo atliktos įsibrovimo atakos. „Zeek“ programinis paketas generuoja kitokius perspėjimo pranešimus nei „Snort“ ar „Suricata“. „Zeek“ generuoja įspėjamąjį failą, vadinamą *notice.log*, kuriame pateikiamas įspėjamasis anomalinis srautas. Remiantis šiuo failu, palyginami pranešimo generavimo laikas ir atliekamos atakos laikas, jeigu jie sutampa – laikoma, kad „Zeek“ sėkmingai atpažino ataką. Laikoma, kad „Snort“ ir „Suricata“ sėkmingai atpažino ataką, kai generuojamas pranešimas atspindi ataką ar jos pobūdį.

Testuoti duomenų failai iš „CIDIDS“ duomenų rinkinio. „Tuesday-WorkingHours“ pirmajame faile atliktos dvi atakos FTP „Patator“ ir SSH „Patator“. Duomenų failo tyrimo rezultatai pateikiami 4.1 lentelėje. Šios dvi atakos remiasi tuo pačiu principu – jėgos metodu (angl. *Brute Force*). Į atakuojamą sistemą bandoma įsilaužti atspėjant prisijungimo duomenis. Pagrindinis šių atakų bruožas – staiga išaugęs, kylantis bandymų prisijungti skaičius. Iš rezultatų matyti, kad FTP „Patator“ ataką aptiko visos ĮAS, o SSH „Patator“ aptiko tik „Snort“ ir „Suricata“ sistemos.

4.1 lentelė. „Tuesday-WorkingHours“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	FTP „Patator“	Taip	FTP komandos parametrai turi neleistiną formatą (angl. *FTP command parameters contained potential string format)*
„Suricata“		Taip	Galimas FTP jėgos metodo panaudojimas (angl. *Potential FTP BruteForce attempt response*)
„Zeek“		Taip	-
„Snort“	SSH „Patator“	Taip	SSH protokolo versijų nesutapimas (angl. *Protocol mismatch*)
„Suricata“		Taip	Galima SSH protokolo skenavimo ataka (angl. *Potential SSH scan outbound*)
„Zeek“		Ne	-

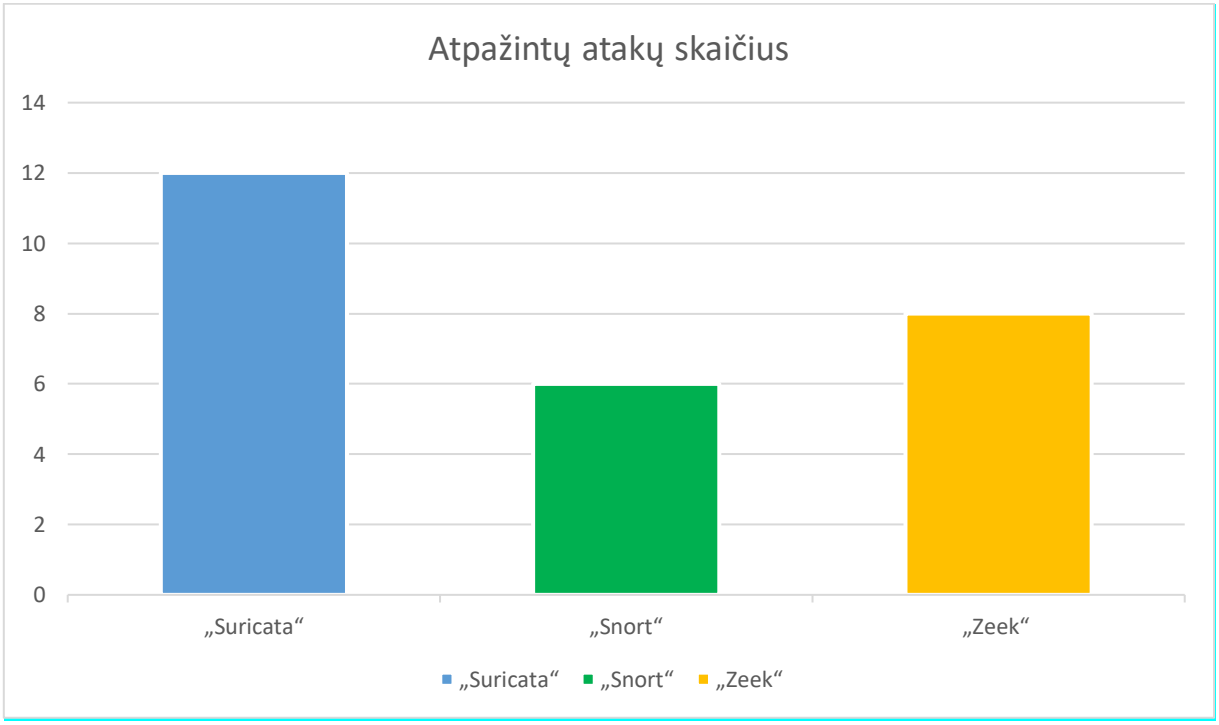
„Wednesday-WorkingHours“ antrame faile atliktos įvairių rūsių DoS tipo atakos. Dėl to, kad rezultatų yra labai daug, antrojo failo tyrimo rezultatų lentelė pateikiama A priedo 4.2 lentelėje. Pagrindinis atakų tikslas – apriboti arba išvis panaikinti atakuojamo mazgo pasiekiamumą. DoS „Slowloris“ ataka atliekama siunčiant nevisas HTTP užklausas. Šią ataką aptiko „Snort“ ir „Zeek“ programinė įranga.

DoS „Slowhttptest“ paremta HTTP užklausomis ataka, kuri įgyvendina „Slowloris“, „Slow HTTP Post“, „Slow Read“ atakas kartu. Šią ataką aptiko „Snort“ ir „Zeek“, „Suricata“ nesugebėjo atpažinti „Slowloris“ atakos, todėl „SlowHttpTest“ ataka taip pat nebuvo aptikta. DoS „Hulk” ataka generuoja nestandartinius HTTP tipo paketus dideliais kiekiais. Ataką aptiko tik „Zeek“ programinė įranga. DoS „GoldenEye“ ataka pasižymi tuo, kad iš vieno mazgo galima generuoti lygiagrečiai daug HTTP prisijungimo sesijų vienu metu. Šią ataką aptiko „Suricata“ ir „Zeek“ programinė įranga.„Heartbleed“ ataka yra „OpenSSL“ pažeidžiamumo išnaudojimo ataka. Pažeidžiamumas išnaudojamas, kada mazgas, bendraujantis su kitu mazgu per TLS protokolus, siunčia „Heartbeat“ paketą, siekiant patikrinti ar kitas mazgas yra vis dar pasiekiamas. „Heartbeat“ paketą sudaro užšifruoti duomenys ir informacija apie paketo ilgį. Būtent informacija apie paketo ilgį tampa pažeidžiamumu. Šį pažeidžiamumą aptiko „Suricata“ ir „Zeek“ programinė įranga.

„Thursday-WorkingHours“ trečiame faile generuojamos atakos buvo iš „Windows“ operacinės sistemos. Trečiojo failo tyrimų rezultatų lentelė taip pat pateikta A priedo 4.3 lentelėje. Pirmoji ataka yra tinklalapio jėgos metodo ataka, šios atakos neatpažino nei viena ĮAS. Antroji ataka „XSS“ (angl. *Cross Site Scripting*) atlieka piktavališko kodo įdiegimą į tinklalapius ar kitus mazgus. Šios atakos, kaip ir pirmosios, neaptiko nei viena ĮAS. SQL „Injection“ ataka atpažįstama iš neįprastų SQL kalbos siunčiamų komandų, kuriomis stengiamasi gauti prieigą prie duomenų bazės vartotojų prisijungimų. Šią ataką aptiko „Snort“ ir „Suricata“ programinė įranga. „Suricata“ ĮAS sugebėjo aptikti tik vieno pobūdžio infiltravimosi ataką – „Dropbox“ kenkėjiško failo parsisiuntimą.

Ketvirtojo failo tyrimų rezultatų lentelė pateikiama A priedo 4.4 lentelėje. Ketvirtajame faile generuojama „Botnet ARES“ ataka. Atakos suteikia įsibrovėliui prisijungimą prie vidinių aukos resursų. Šią ataką atpažino „Snort“ ir „Suricata“ ĮAS. Antroji ataka priskiriama prie informacijos rinkimo apie mazgą kategorijos. Ši ataka atliekama siekiant sužinoti kuo daugiau informacijos apie mazgo naudojamus prievadus su tikslu jais pasinaudoti. Šią ataką atpažino „Suricata“ ir „Zeek“.

Iš duomenų rinkinio „ISCX“ analizuoti keturi duomenų failai. Duomenų faile „testbed13-jun“ vykdomas infiltravimas į vidinius resursus iš vidinio serviso. Duomenų failo tyrimo rezultatai pateikiama A priedo 4.5 lentelėje. Infiltravimas suprantamas kaip pirminis informacijos rinkimas apie esančius vidinius resursus. Prievadų skenavimas, bandymas atspėti prisijungimus, piktavališkos programinės įrangos prisijungimus. Šias atakas ar bent dalį iš jų aptiko „Suricata“ ir „Zeek“ programinė įranga. Antrame „testbed14-jun“ duomenų faile generuota HTTP DoS ataka, kurią aptiko „Suricata“ ir „Zeek“ programinė įranga. Duomenų failo tyrimo rezultatai pateikiami A priede 4.6 lentelėje. Trečiame „testbed15-jun“ duomenų faile atlikta DoS ataka. Duomenų failo tyrimo rezultatai pateikiami A priede 4.7 lentelėje. Ataką aptiko tik „Zeek“ programinė įranga. Paskutiniame „testbed17-jun“ duomenų faile atliktos SSH protokolo jėgos metodo atakos, kada steigiamasi įsilaužti į mazgą per SSH protokolą, bandant skirtingus prisijungimus. Duomenų failo tyrimo rezultatai pateikiami A priedo 4.8 lentelėje. Šią ataką sėkmingai atpažino „Snort“ ir „Suricata“. Bendras atpažintų atakų skaičius pateikiamas 4.1 pav.



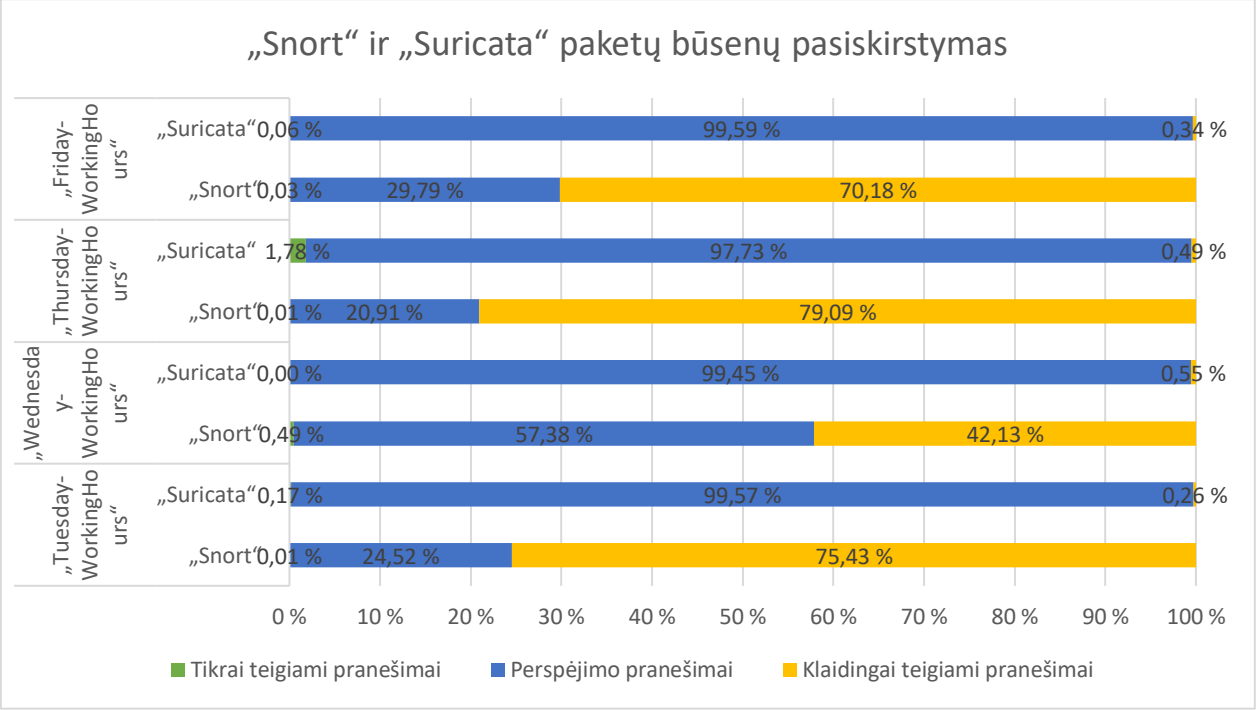
4.1 pav. „Snort“, „Suricata“ ir „Zeek“ suminis atpažintų atakų skaičius per abejus „CIDIDS“ ir „ISCX“ tirtus duomenų rinkinius

Iš viso atlikta 20 atakų per tirtus duomenų rinkinius. „CIDIDS“ duomenų rinkinyje atlikta 16 atakų. „Suricata“ atpažino 9 atakas, „Snort“ – 4, o „Zeek“ – 6. „ISCX“ duomenų rinkinyje atliktos 4 atakos. „Suricata“ atpažino 3, „Snort“ – 1 ir „Zeek“ – 3 atakas. Pagal rezultatus geriausiai

pasirodė „Suricata“ programinė įranga atpažindama 12 atakų iš 20. ĮAS geba atpažinti atakas, esant mišriam srautui. Kitas žingsnis – plačiau išanalizuoti gautus pranešimus apie atakas.

Žinome, kad ĮAS geba atpažinti ataką, tačiau iš aptikimo pranešimo nebus jokios naudos, jeigu ĮAS generuos daug klaidingai teigiamų pranešimų. Klaidingai teigiami pranešimai užgožia svarbiausią perspėjimo pranešimą – tai viena didžiausių ĮAS efektyvumo problemų. Svarbu atkreipti dėmesį, kad šablonu pagrįstos ĮAS dažnai gali neaptikti atakos, nes tiesiog nėra sukurtos panašios taisyklės į atliekamą ataką. Nors ataka nėra tiesiogiai atpažįstama, svarbu įvertinti, kiek pašalinių pranešimų ĮAS generuoja, esant atakai. Šių įspėjimų pagrindinis požymis – kiekis. Apžvelgiame, ar ĮAS sugeba aptikti atakas, esant mišriam srautui, koncentruojantis į tai, ar ĮAS atpažino paketą, susijusį su ataka. Analizuojame, kiek pranešimų ir kokių tipų ĮAS generavo analizės metu. Žinoma, kad klaidingai teigiami pranešimai užima didžiąją dalį visos analizės pranešimų kiekio [28]. Poskyryje apžvelgsime „Snort“ ir „Suricata“ pranešimų tipų pasiskirstymą. „Zeek“ nesiremia šabloninėmis taisyklėmis, ši programinė įranga remiasi anomalija, kuri pateikiama žurnaliniuose failuose, todėl palyginti rezultatus su „Snort“ ir „Suricata“ tiksliai negalime.

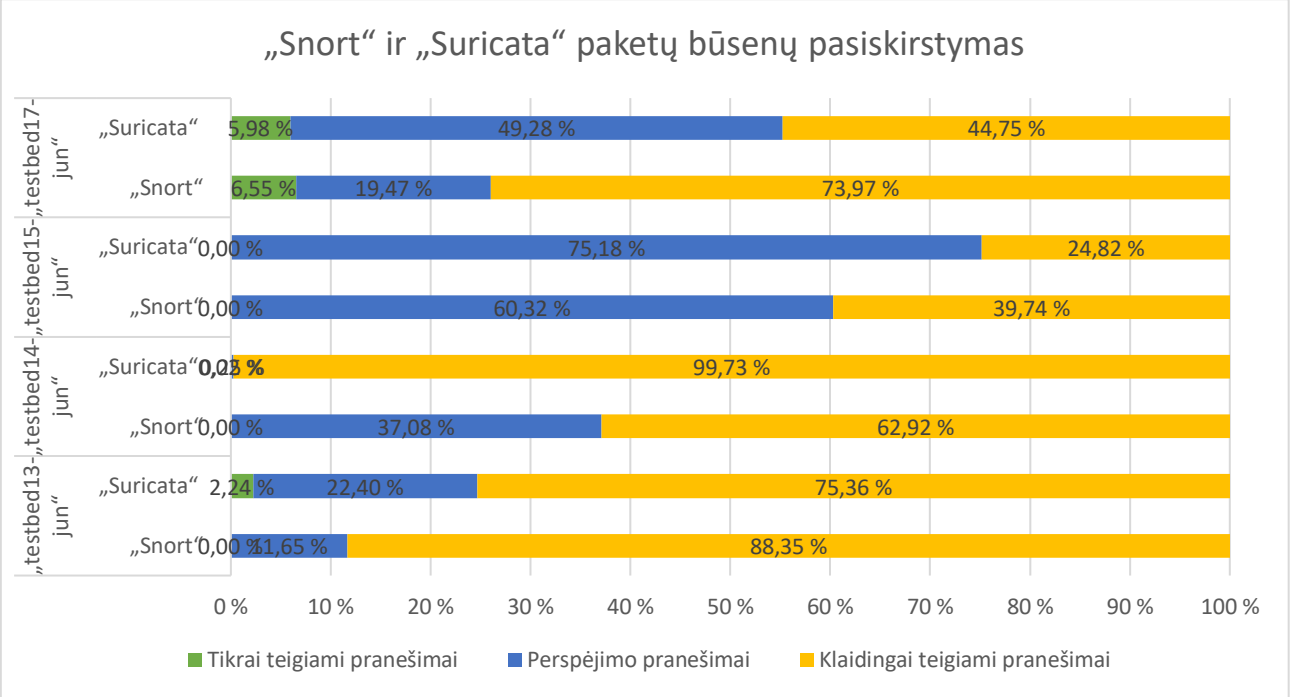
Klaidingai teigiamų pranešimų analizė tampa sudėtingesnė, nes atsiranda neaiškumo atvejis [53]. Pavyzdžiui, atliekamas prievadų skenavimas, kurio metu į kiekvieną prievadą yra siunčiamas SYN paketas. ĮAS gali tiesiogiai neaptikti ir nepažymėti, kad dabar atliekama prievadų skenavimo ataka prieš saugomą mazgą. Tačiau pati ataka generuoja papildomus perspėjimo pranešimus, kurie gali aptikti ĮAS ir atkreipti dėmesį, kad kažkokia veikla yra vykdoma. Siekiant įvertinti tokius atvejus, tokio tipo pranešimai išskiriami į perspėjimo pranešimų kategoriją. Tokia analizės metodika taikoma ir analogiškuose darbuose [53]. Žinoma, jeigu jėgos metodo atakos metu ĮAS pažymi, kad tai yra viruso įsibrovimas į vidinius sistemos parametrus, tokie pranešimai žymimi kaip KT. Pateikiamas duomenų rinkinio „CIDIDS“ pranešimų tipų pasiskirstymo grafikas 4.2 pav. Abiejų duomenų rinkinių pranešimų tipų analizės lentelės pateikiamos B priede 4.9 ir 4.10 lentelėse.



4.2 pav. „CIDIDS“ duomenų rinkinio sugeneruotų pranešimų pasiskirstymas „Snort“ ir „Suricata“ programiniuose paketuose

Matome, kad TT pranešimų skaičius yra minimalus pagal suminį pranešimų skaičių. Iš esmės tokio santykio reikia tikėtis, vien dėl ĮAS prigimties. Tačiau būtina pabrėžti ir perspėjimo pranešimų kiekį, kuris tampa kaip pagalba atpažįstant tikras atakas. „Tuesday-WorkingHours“ duomenų faile buvo FTP ir SSH jėgos atakos, kurios iš savęs generuoja didelius kiekius SYN paketų. „Snort“ ir „Suricata“ sistemos sėkmingai atpažino atakas. Perspėjimo pranešimai, tiek ir „Suricata“ tiek ir „Snort“, susideda iš informacijos apie SYN paketų kiekį, viršijantį TCP lango dydį ir kontrolės sumos klaidas. Pagal šiuos bruožus jau galima daryti prielaidą, kad yra vykdoma ataka prie sistemą. „Wednesday-WorkingHours“ duomenų faile atliktos įvairių tipų DoS atakos, kurių pašalinis bruožas yra dideli kiekiai SYN paketų, keistos HTTP paketų antraštės, netikslūs paketų ilgiai. Visi šie bruožai patenka į perspėjimo pranešimus.

„Thursday-WorkingHours“ duomenų faile atliktos infiltravimo atakos, kurios susideda iš prievadų skaitymo, informacijos rinkimo, prisijungimų bandymų ir kitų būdų, siekiant patekti į vidinius sistemos servisus. „Friday-WorkingHours“ duomenų faile atliktos „Botnet“, prievadų skenavimo ir DDoS atakos, šios atakos stipriai siejasi su SYN paketais, jų kiekių ir TCP antraščių korektiškumu. Dažniausiai tokio tipo atakos turi keistus, neatitinkančius standartų paketus, kurie gali sukelti pirminį įtarimą apie ataką. Pateikiamas duomenų rinkinio „ISCX“ pranešimų tipų pasiskirstymas 4.3 pav.



4.3 pav. „ISCX“ duomenų rinkinio sugeneruotų pranešimų pasiskirstymas „Snort“ ir „Suricata“ programiniuose paketuose

„Testbed13-jun“ duomenų faile atliktos infiltravimo atakos, kurios iš savo prigimties yra sunku atpažinti pagal šablonines taisykles. Tačiau išlieka infiltravimo atakų bruožai, kuriuos stebime per perspėjimo pranešimus. „Testbed14-jun“ įdomus duomenų failas, kadangi pateikia kitokius rezultatus nei ankstesni tyrimo punktai. Šiame duomenų faile atlikta HTTP DoS ataka, jos aptikti nepavyko „Snort“, tačiau „Snort“ generavo žymiai daugiau perspėjimo pranešimų apie įtartiną veiklą nei „Suricata“. „Testbed15-jun“ DDoS tipo atakos negalėjo atpažinti „Snort“ ir „Suricata“ sistemos, nes nei viena nesudarė pranešimo apie tokio tipo ataką. Tačiau tiek „Snort“, tiek ir „Suricata“ generavo sąlyginai daug perspėjimo pranešimų, kurie būdingi DDoS atakoms. „Testbed17-jun“ duomenų faile daugiausiai gauta TT pranešimų, kadangi vykdyta paprasta jėgos metodo ataka į SSH protokolą. Abi sistemos „Suricata“ ir „Snort“ susidorojo su šia ataka tiek teisingai ją atpažindamos, tiek suteikdamos perspėjimo pranešimų.

„Snort“ perspėjimo pranešimai susideda iš įspėjimų apie RST paketo siuntimą ne TCP perdavimo lange, TCP sesijos užmezgimą be trijų rankų paspaudimo procedūros, HTTP antraščių neatitikimą. „Suricata“ perspėjimo pranešimai susideda iš TCP kontrolinės sumos klaidos pranešimų, HTTP antraščių klaidų. Svarbu paminėti, kad „Suricata“ žymiai daugiau generavo pranešimų, lyginant su „Snort“. „Suricata“ paremta daugiagysliu paketų apdorojimu, todėl tai gali daryti įtaką pranešimų kiekio generavimui.

Tik skaičiuojant TT pranešimų kiekį ĮAS pasirodo prasčiau dėl tiesiog per mažo jų kiekio, kuris atkreiptų administratoriaus dėmesį. Įvertinant generuojamų perspėjimo pranešimų kiekį, kuris atakos metu nenurodo tikslios atakos, tačiau geba identifikuoti apie veiksmus, nukreiptus prieš saugomą mazgą, galima susidaryti realistiškesnį vaizdą.

Palyginant „Snort“ ir „Suricata“ ĮAS, „Suricata“ generuoja mažiau klaidingai teigiamų pranešimų, daugiau perspėjimo pranešimų nei „Snort“. „CIDIDS“ duomenų rinkinio analizėje „Suricata“ perspėjimo pranešimų kiekis nekrito 99 %, o „Snort“ keitėsi 20–57 % intervale. „ISCX“ duomenų rinkinio analizėje stebimas mažesnis perspėjimo pranešimų kiekis, kintantis 22–75 % intervale. „Snort“ perspėjimo pranešimų kiekis kito 12–60 % intervale. Iš gautų rezultatų galima teigti, kad „Suricata“ dažniau atpažįsta atakas nei „Snort“ ir „Zeek“ ir taip pat generuoja mažiau klaidingai teigiamų pranešimų lyginant su „Snort“ ĮAS.

4.2. Sistemos parametrų įtakos tyrimo rezultatai

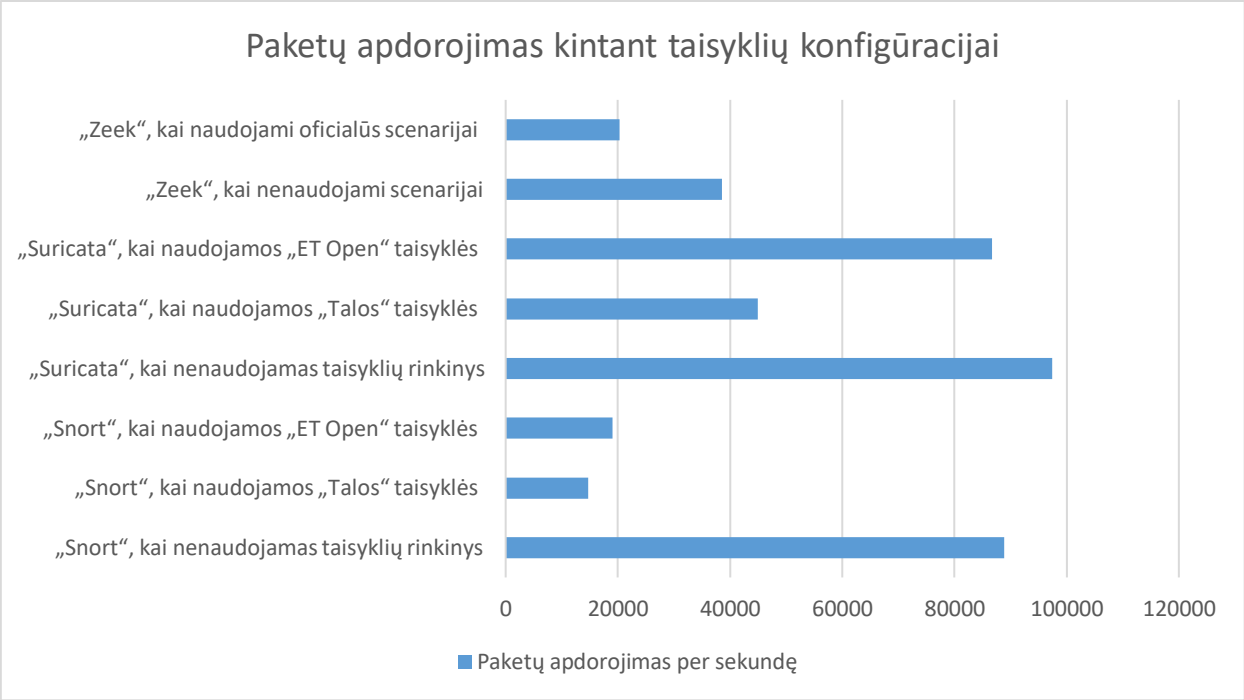
Atliktas sistemos parametrų įtakos tyrimas. Gauti tyrimo rezultatai apibendrinami ir pateikiamas rezultatų vidurkis. Rezultatai pateikiami atskiruose grafikuose. Pirmojo tyrimo rezultatų grafinis atvaizdavimas pateikiamas 4.4 pav.

Pirmajame punkte tirta taisyklių konfigūracijos įtaka ĮAS našumui. Aparatinei įrangai išskirti 4 CPĮ branduoliai ir 8 GB darbinės atminties, siekiant išvengti resursų limitavimo. Pirmieji rezultatai parodė, kad „Snort“ ir „Suricata“ be taisyklių pasiekia aukštą paketų skaitymo greitį – sekundę atitinkamai 88874 ir 97465 paketų. „Zeek“ programinė įranga pasiekė mažesnį rezultatą – 38600, tačiau rezultatas realistiškas, nes „Zeek“ ĮAS atlieka gilesnę ir platesnę paketų analizę nei pastarosios ĮAS. Įdomus pirminis rezultatas, nes „Snort“ sistema, nors ir gali naudoti tik vieną branduolį, nedaug atsiliko nuo daugiagyslės ĮAS – „Suricata“. Paketų apdorojimo greitis tarp ĮAS skiriasi apie 10000 paketų.

Antrajame punkte „Snort“ ir „Suricata“ ĮAS buvo sukonfigūruotos naudoti „ET Open“ taisyklių rinkinį. Atitinkamai taisyklės parsiųstos pagal sistemos versiją tiek „Snort“, tiek ir „Suricata“ programinei įrangai. Gauti rezultatai parodė stiprų paketų perdavimo greičio sumažėjimą „Snort“ programinėje įrangoje – 19086. „Suricata“ naudoja daugiagyslį veikimą ir pasiekia – 86745 paketų apdorojimo greitį. „Zeek“ programinė įranga buvo paleista su visais scenarijais, tada „Zeek“ sistema – pasiekė 20304 paketų perdavimo per sekundę greitį.

Trečiajame punkte „Snort“ ir „Suricata“ ĮAS sukonfigūruotos naudoti „Talos“ taisyklių rinkinį. Šis rinkinys yra oficialiai prieinamas tik „Snort“ programinei įrangai, tačiau gali būti

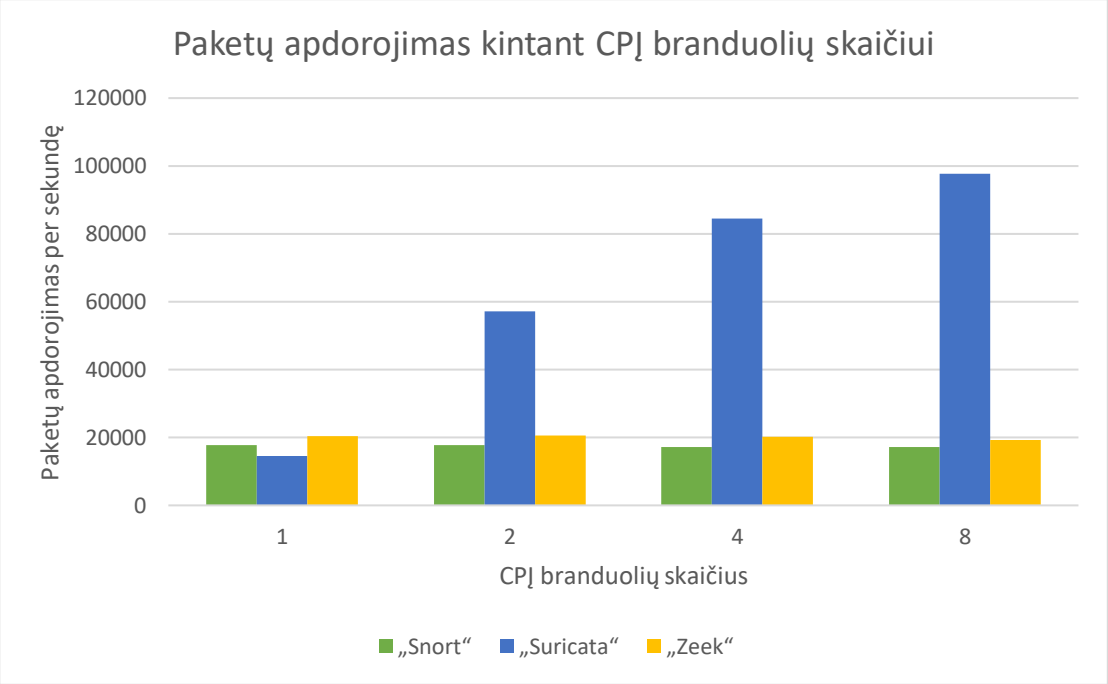
pritaikytas ir „Suricata“ dėl taisyklių sintaksės panašumo. „Snort“ pasiekė 14726, o „Suricata“ 44998 paketų per sekundę skaitymo greitį. Matyti, kad programinė įranga sunkiai apdoroja „Talos“ taisyklių rinkinį, ypač krito „Suricata“ rezultatai. Taisyklės sintaksės pokyčiai turėjo pakankamai didelę įtaką „Suricata“ ĮAS našumui.



4.4 pav. „Snort“, „Suricata“ ir „Zeek“ paketų apdorojimo spartos kitimas esant skirtingiems taisyklių rinkiniams, kai CPĮ branduolių kiekis yra 4, darbinė atmintis – 8 GB

Tyrimo punktai parodė, kad ĮAS paketų skaitymo ir dekodavimo varikliai yra greiti, o našumas tarp „Snort“ ir „Suricata“ skiriasi apie 10 %. Rezultatai keičiasi, pritaikant taisyklių konfigūraciją. Didžiausią našumą prarado „Snort“ ĮAS naudojant „Talos“ taisyklių rinkinį – našumas krito apie 85 %, „Suricata“ apie 5 %. Lyginant „Zeek“ programinės įrangos skaitymo greitį, nesant papildomoms taisyklėms, našumas apie 50 %. „Suricata“ ĮAS, naudojant „ET Open“ taisyklių rinkinį, našumas sumažėjo apie 10 %.

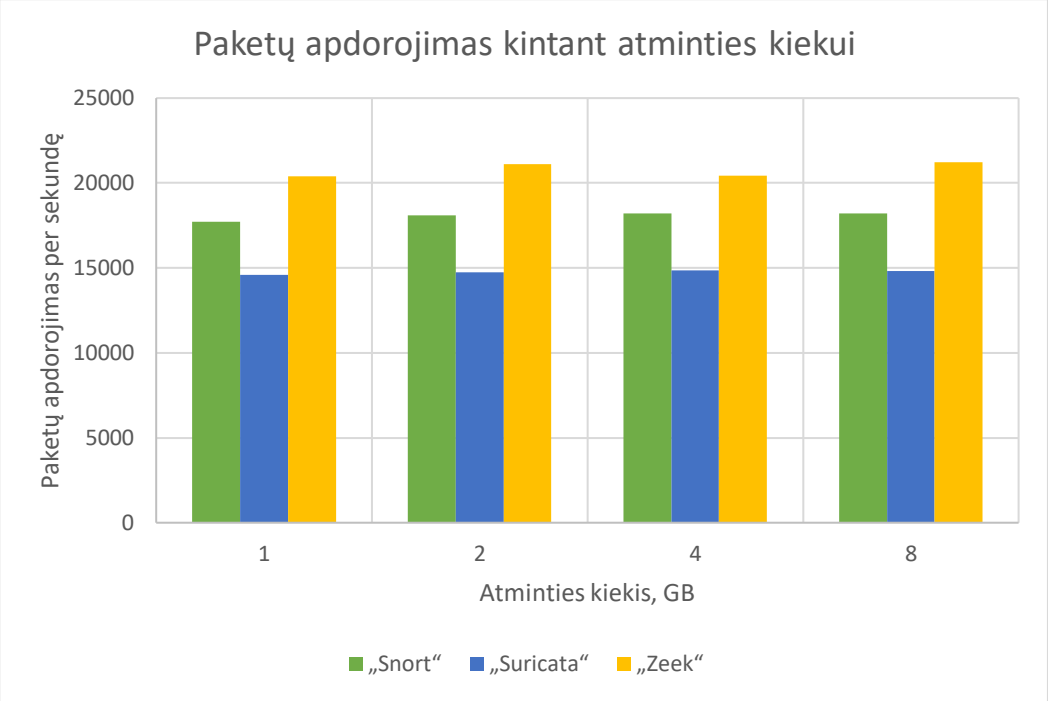
Toliau siekiama išanalizuoti CPĮ branduolių kiekio įtaką paketų skaitymo režimui ĮAS. Virtualiojoje aplinkoje branduolių skaičius buvo keičiamas į 1,2,4 ir 8. Darbinės atminties kiekis konfigūruotas – 1 GB. Tyrime naudojamas „ET Open“ taisyklių rinkinys „Snort“ ir „Suricata“ sistemoms, o „Zeek“ naudoja oficialius scenarijus. Galutiniai rezultatai pateikiami 4.5 pav.



4.5 pav. Paketų skaitymo sparta per sekundę kintant branduolių skaičiui

Rezultatai patvirtino galimas prielaidas, didėjant branduolių kiekiui, „Suricata“ paketų apdorojimo kiekis per sekundę didėjo tolygiai. Kadangi „Snort“ ir „Zeek“ programinė įranga paremta vienos gijos technologija, jų rezultatai iš esmės nesiskyrė nuo vieno branduolio tyrimo rezultato. Būtina paminėti, kad ir nesikeičiant rezultatams, „Zeek“ programinė įranga turėjo geresnį našumą nei „Snort“ ir „Suricata“ paketai. „Zeek“ paketų skaitymo sparta per sekundę – 20603, „Snort“ – 17733, „Suricata“ – 14587. Esant limituotiems resursams „Suricata“ veikia prasčiausiai.

Kitu tyrimu buvo siekiama išanalizuoti darbinės atminties įtaką ĮAS skaitymo režimo spartai. Virtualiojoje aplinkoje darbinės atminties kiekis buvo didinamas iki 1,2,4 ir 8 GB, naudojamas vienas CPĮ branduolys. Tyrime naudojamas „ET Open“ taisyklių paketas „Snort“ ir „Suricata“ ĮAS, o „Zeek“ naudoti standartiniai oficialūs scenarijai. Galutiniai rezultatai pateikiami 4.6 pav.



4.6 pav. Paketų skaitymo sparta paketais per sekundę kintant darbinės atminties kiekiui

Rezultatai patvirtino prielaidas, kad darbinės atminties kiekio didinimas nedaro įtakos ĮAS paketų skaitymo režimui. Per atliktus tyrimus nei vienos programinės įrangos rezultatai nepasikeitė, jie išliko pastovūs. Tyrimo rezultatai suteikė žinių tolimesniam našumo tyrimui. Paketų skaitymui didžiausią įtaką darė taisyklių konfigūracija ir CPĮ branduolių kiekis, kurį labiausiai naudojo „Suricata“ ĮAS. Darbinė atmintis sudarė minimalią įtaką skaitant paketus.

4.3. Įsibrovimo aptikimo sistemų našumo tyrimo rezultatai

Įsibrovimo atpažinimas yra dalis tinklo saugumo, kuris apdoroja didelius kiekius srauto. Ankstesniuose tyrimuose apžvelgėme, kaip ĮAS atlieka paketų analizę, esant stabiliam paketų apdorojimo greičiui, paketų skaitymo režime. Paketų skaitymo režime ĮAS apriboja paketų skaitymo greitį, siekiat sėkmingai apdoroti kiekvieną paketą ir priimti sprendimą dėl pranešimo generavimo.

Tinklo saugumo monitoringas negali būti sudarytas vien tik iš praeities įvykių analizės – būtina realaus laiko srauto stebėsena [18]. Problema kyla didelių srautų tinkluose, kuriuose įdiegta ĮAS. Dažnai sistemos negali apdoroti didelio srauto dėl resursų trūkumo ar pačios programinės įrangos apdorojimo mechanizmo. Kuo didesnis kiekis dekoduotų paketų yra atmestų ĮAS, tuo didesnė tikimybė piktavališkam srautui įsibrauti į sistemą nepastebėtam.

Našumo tyrimui svarbus ir perdavimo srautas. Apžvelgiant ĮAS našumo analizės straipsnius matyti, kad autoriai atlieka našumo tyrimus, naudojant paprastus TCP ar UDP protokolų paketus, o tai neatspindi realaus perdavimo srauto [18]. Analizuojant korektiškus, standartizuotus paketus be atakų galime išvystyti didesnį ĮAS našumą, nei yra iš tikrųjų. Būtina naudoti įvairų srautą, kuris atspindėtų realaus srauto tendencijas.

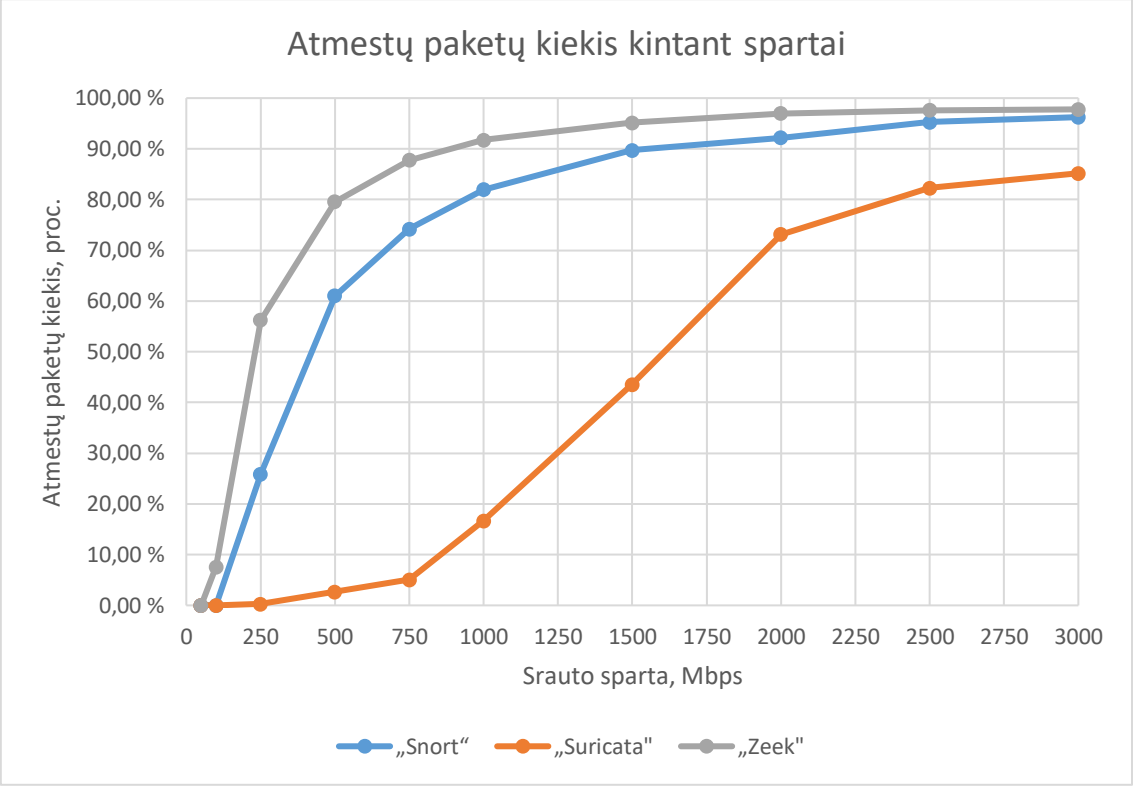
Remiantis apibrėžtais punktais, tyrimo tikslas – ištirti koks ĮAS našumas, esant realaus laiko stebėsenos režimui. Ištirti našumo priklausomybę nuo perdavimo srauto. Atlikti konfigūracijos parametrų analizę ir apžvelgti konfigūracijos parametrus, gebančius padidinti ĮAS našumą.

4.3.1. Našumo tyrimo rezultatai esant standartinei konfigūracijai

Gauti tyrimo rezultatai apibendrinti ir išvestas rezultatų vidurkis. Galutiniai rezultatai pateikiami atskiruose grafikuose, nurodant tyrimo punktą. Iš viso atlikta 600 matavimų, esant skirtingoms ĮAS konfigūracijoms.

Atliktas tyrimo punktas, kada ĮAS konfigūracijos parametrai yra standartiniai. Srauto metu matuoti parametrai: CPĮ naudojimas procentais ir atmestų paketų kiekis procentais. Rezultatuose pateikiamas CPĮ naudojimas atvaizduoja keturių branduolių naudojimą. Pavyzdžiui, 25 % reiškia, kad ĮAS naudoja vieną branduolį iš keturių galimų. Bendras atmestų paketų kiekis kintant spartai pateikiamas 4.7 pav.

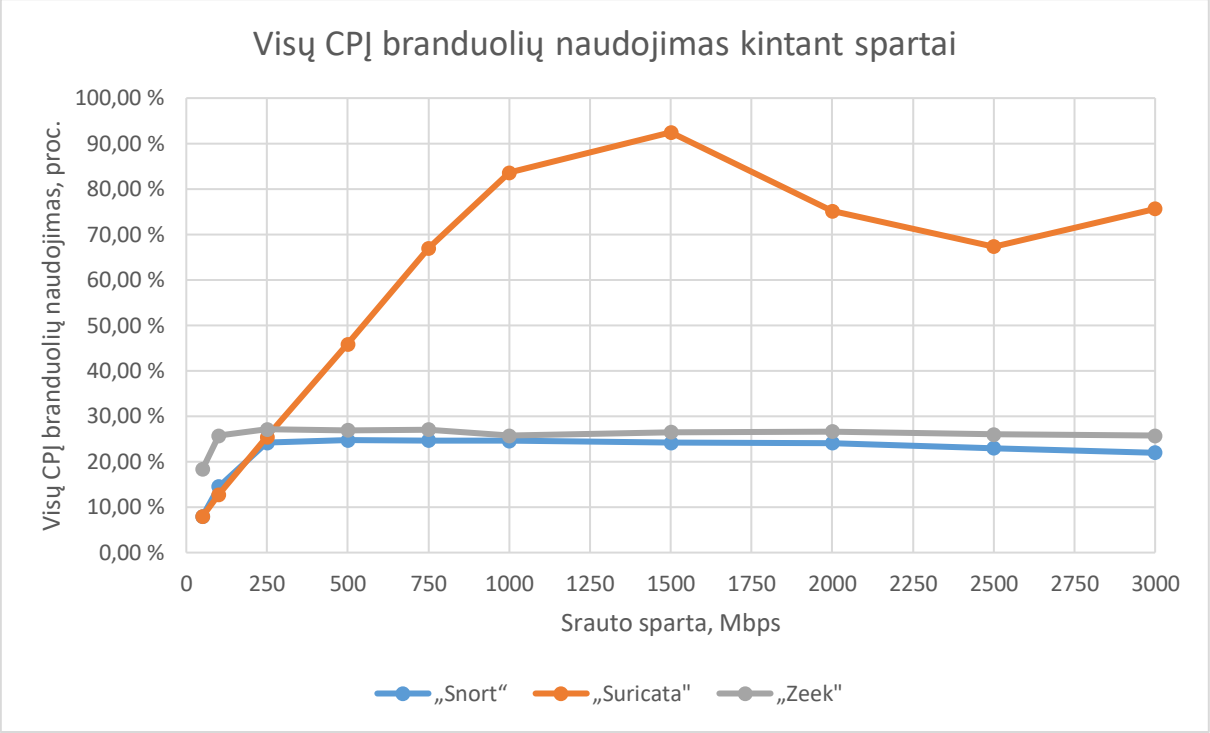
Tyrimo punkto rezultatai parodė, kad „Snort“ ir „Zeek“ lūžio taškas, kada pastebimas atmestų paketų kiekis, yra apie 100 Mbps. „Snort“, esant 100 Mbps perdavimo spartai, atmetė 0,01 % visų gautų paketų. „Zeek“, esant tai pačiai spartai, atmetė 7,60 % paketų. Staigus atmestų paketų kiekio šuolis pasireiškia tarp 100 Mbps ir 250 Mbps. Šiame tarpe „Snort“ atmestų paketų kiekis išaugo nuo 0,01 % iki 25,89 %, „Zeek“ nuo 7,60 % iki 56,20 %. „Suricata“ pastebimas paketų atmetimas įvyksta esant 500 Mbps spartai, kai atmestų paketų kiekis – 2,68 %. Esant 1 Gbps spartai „Snort“ ir „Zeek“ ĮAS atmestų paketų kiekis atitinkamai 81,98 % ir 91,75 %, o „Suricata“ išlaiko sąlyginai mažą paketų atmetimą – 16,70 %. Nuo 1 Gbps spartos pradeda staigiai didėti ir „Suricata“ ĮAS atmestų paketų kiekis, esant 1,5 Gbps atmesti paketai sudaro beveik 50 % viso gauto srauto.



4.7 pav. Atmestų paketų kiekis kintant srauto spartai procentais, esant standartinei ĮAS konfigūracijai

Tyrimo punkto rezultatai parodė, kad „Snort“ ir „Zeek“ ĮAS pradeda atmesti paketus žymiai greičiau nei „Suricata“ ĮAS. „Snort“ ir „Zeek“ ĮAS yra paremtos vienos gijos architektūra, todėl išskiriamas tik vienas procesas gijai apdoroti paketus. Matome, kad vieno procesoriaus naudojimas geba apdoroti tik iki 150 Mbps srauto, kai dar nėra didelio paketų atmetimo. Bendras grafikas atvaizduojantis visų CPĮ branduolių naudojimą 4.8 pav.

„Snort“ esant 250 Mbps spartai pasiekia 25 % CPĮ naudojimo ribą, kas reiškia, kad „Snort“ procesas visiškai naudoja virtualiosios mašinos vieną branduolį. Galima sakyti, kad 250 Mbps jau yra virš „Snort“ programos ribos, nes atmestų paketų kiekis yra didelis – 25,89 %. „Zeek“ CPĮ naudojimas jau esant 50 Mbps yra aukštas – 18,3 %. Taip nutinka todėl, kad „Zeek“ atlieka gilesnę ir detalesnę paketų analizę nei „Snort“ ar „Suricata“ ĮAS. „Suricata“ CPĮ naudojimas kinta tiesiškai iki 1–1,5 Gbps spartos ribos. Matomas įdomus rezultatas – pasiekus 1,5 Gbps spartą ir spartai didėjant iki 2,5 Gbps, „Suricata“ CPĮ naudojimas krenta. Stebimas PCAP paketų priėmimo modulio ir „Suricata“ riba. Paketų atmetimas didėja dėl „Suricata“ variklio ribotumo ir kartu krenta CPĮ naudojimo kiekis, nes apdorojama vis mažiau paketų.



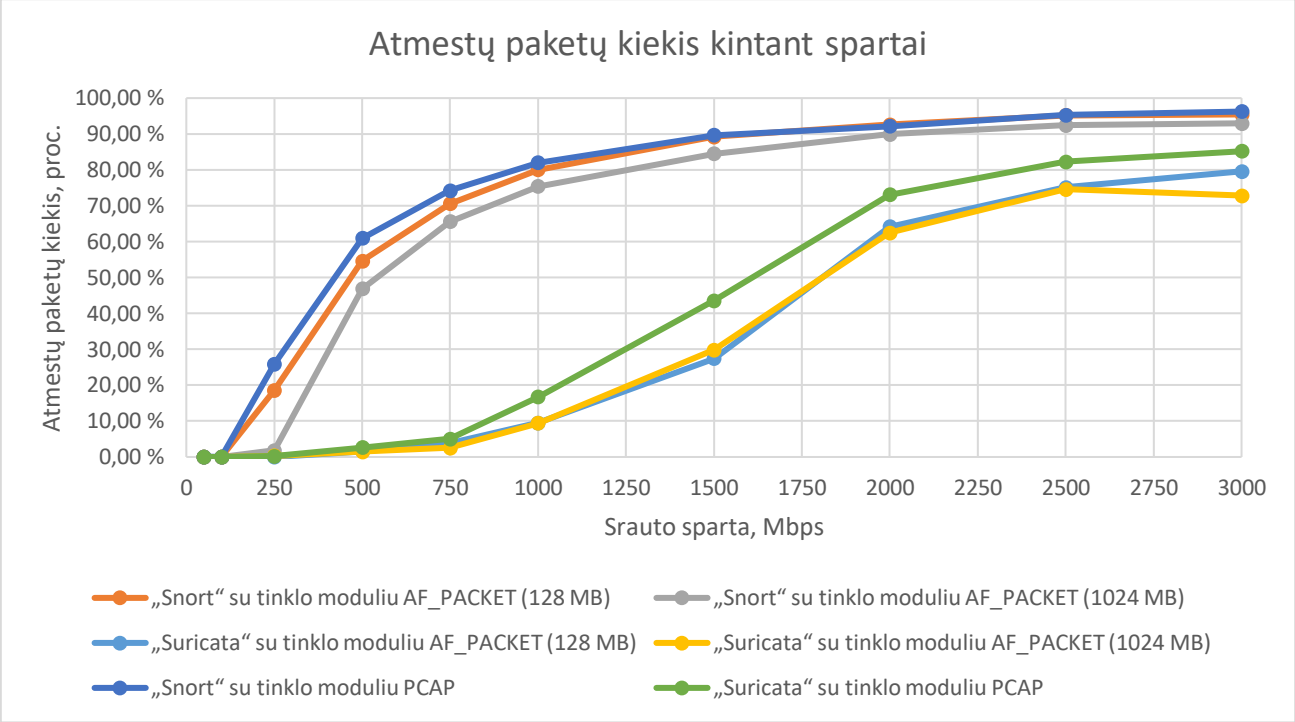
4.8 pav. Visų CPĮ branduolių naudojimas kintant srauto spartai procentais, esant standartinei ĮAS konfigūracijai

4.3.2. Našumo tyrimo rezultatai taikant AF_PACKET modulį

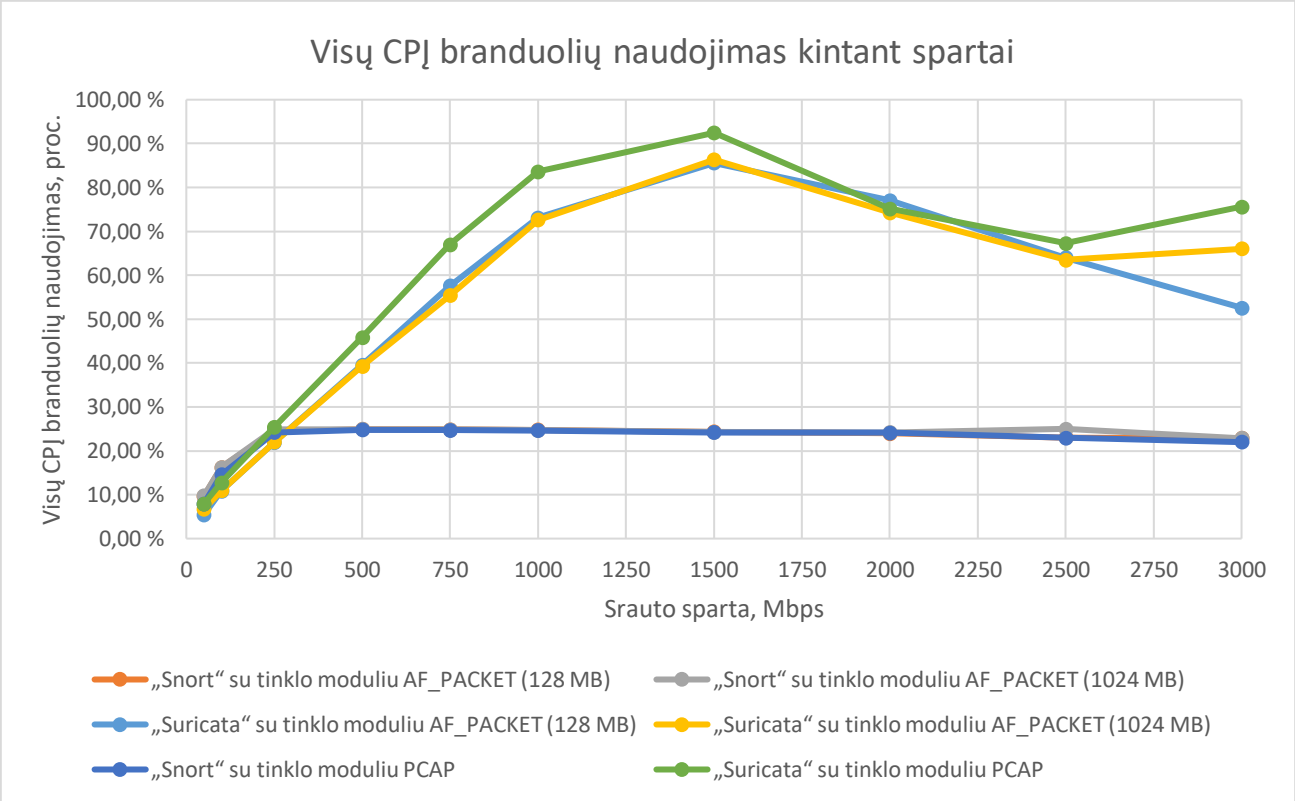
Pirmasis tyrimo punktas parodė, kad standartinės konfigūracijos „Snort“ sėkmingai apdoroja paketus iki 250 Mbps, „Zeek“ iki 100 Mbps, o „Suricata“ iki 750 Mbps. Siekiame rasti ĮAS konfigūracijos būdus, kurie sumažintų atmestų paketų kiekį.

Vienas iš limituojančių faktorių yra PCAP modulis, kuris yra tarpininkas tarp tinklo sąsajos ir ĮAS. „Snort“ ir „Suricata“ ĮAS suteikia galimybė pakeisti paketų priėmimo modulį į AF_PACKET, kuris leidžia tiesiogiai priimti paketus iš tinklo sąsajos be papildomų bibliotekų. AF_PACKET modulis geba išskirti darbinės atminties kiekį priimamiems paketams.

Tyrimai atlikti, kai AF_PACKET moduliui išskirta standartinė darbinė atmintis – 128 MB ir, kai atmintis padidinta iki 1024 MB. „Zeek“ ĮAS neturi AF_PACKET modulio, todėl „Zeek“ ĮAS neįtraukta į tyrimo punktą. Srauto metu matuoti parametrai: CPĮ naudojimas procentais ir atmestų paketų kiekis procentais. Bendras atmestų paketų kiekis ir viso CPĮ naudojimo grafikai kintant spartai pateikiamas 4.9 pav. ir 4.10 pav.



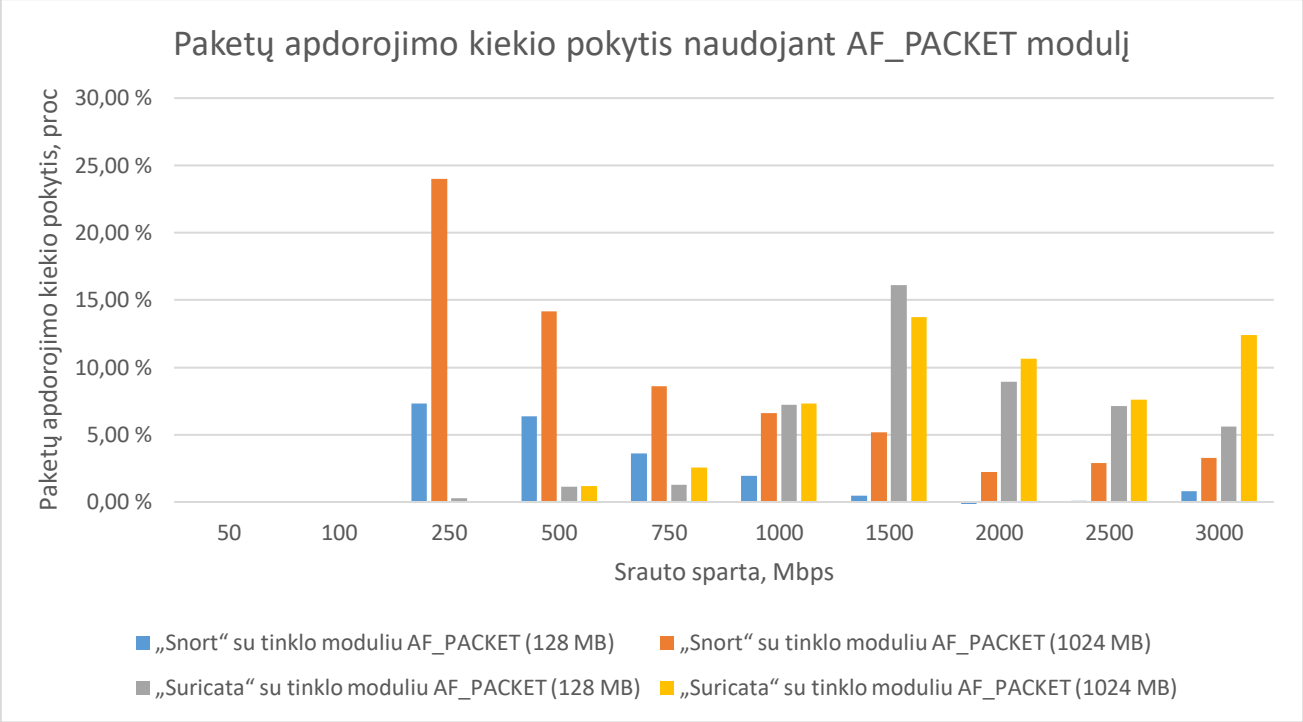
4.9 pav. „Snort“ ir „Suricata“ atmestų paketų kiekis kintant spartai, naudojant AF_PACKET modulį, kai darbinės atminties kiekis yra 128 MB ir 1024 MB



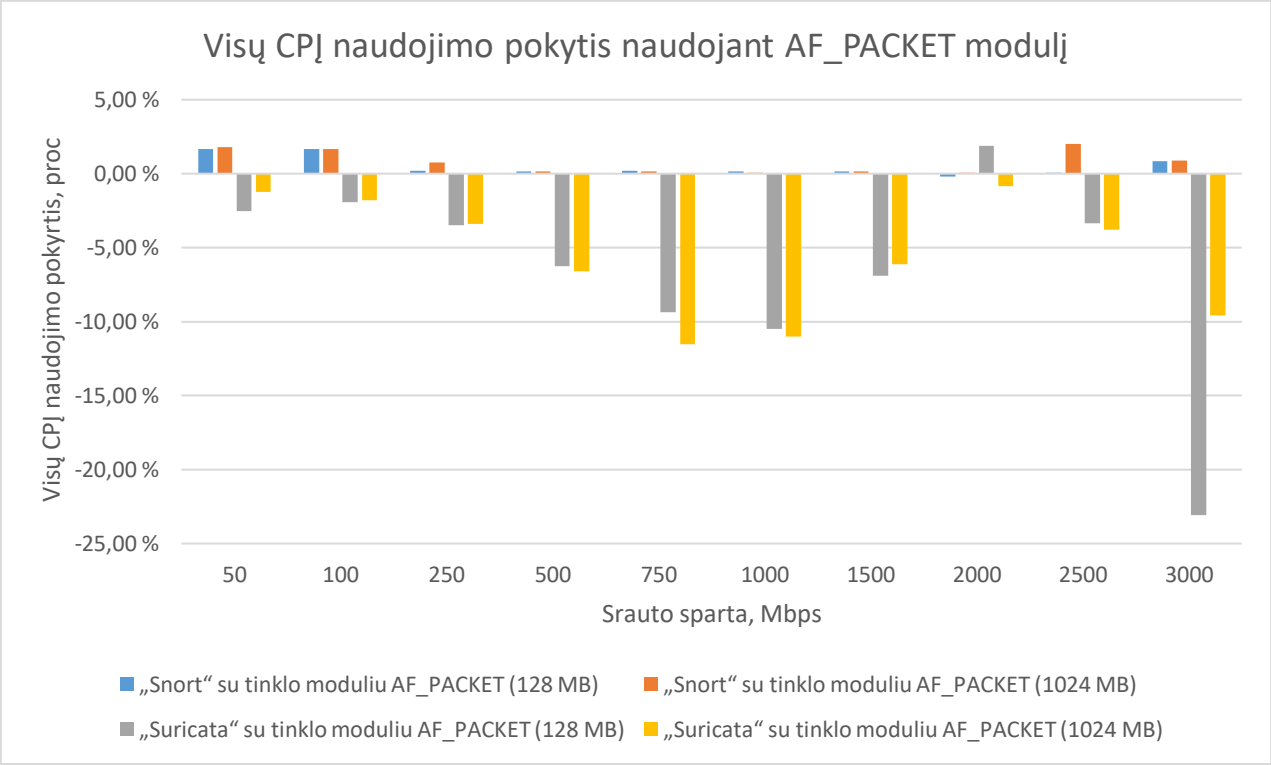
4.10 pav. „Snort“ ir „Suricata“ visų CPĮ branduolių naudojimas kintant spartai, naudojant AF_PACKET modulį, kai darbinės atminties kiekis yra 128 MB ir 1024 MB

Iš grafiko matyti, kad „Snort“ atmestų paketų kiekio kitimo tendencija nesikeičia. Didesnis atminties kiekis, skirtas paketų priėmimo buferiams, leidžia apdoroti didesnį paketų srautą. Geriausias rezultatas matyti, kai naudojamas AF_PACKET modulis su 1024 MB išskirta atmintimi. Ši konfigūracija leidžia sumažinti atmestų paketų kiekį iki 1,88 % prie 250 Mbps. Matomas staigus kreivės pokytis artėjant prie 500 Mbps ribos, kai atmestų paketų kiekis tampa 46,84%. Kreivė pasiekia piką, kai daugiau nei 80 % viso srauto yra atmesti paketai. Nors atmestų paketų kiekis yra mažesnis – iki 10 %. lyginant 128 MB ir 1024 MB rezultatus, tačiau didėjant srauto spartai „Snort“ kitas limituojantis faktorius yra CPĮ naudojimas, ne buferių atmintis. Vienas „Snort“ ĮAS procesas negali efektyviai naudoti sistemos resursų ir apdoroti įeinančio srauto. Todėl visų CPĮ naudojimo „Snort“ kreivės nesikeičia, esant skirtingoms „Snort“ konfigūracijoms. Pagal ‚Snort“ CPĮ naudojimo grafiką galime daryti prielaidą, kad sumažinus CPĮ naudojimą apie 10 %, tokį pat rezultatą pastebėsime su atmestais paketais. „Suricata“ geriau naudoja CPĮ resursus dėl žiedo buferio privalumų, kurių neturi „Snort“ ĮAS. „Suricata“ artėja prie resursų limitų – kreivės artėja viena prie kitos srautui didėjant nuo 1,5 Gbps. Didėjant srautui išskirstas atminties kiekis nedaro įtakos „Suricata“ našumui. Iš grafiko kreivių matosi pokytis naudojant AF_PACKET modulį vietoje standartinio PCAP modulio. „Suricata“ ĮAS efektyviai naudoja ne tik papildomą buferio atmintį, bet ir žiedo buferio funkcionalumą. Pritaikius AF_PACKET modulį „Suricata“ ĮAS geba sumažinti iki 15 % atmestų paketų kiekį. Bendras paketų apdorojimo kiekio pokytis pateikiamas 4.11 pav. ir CPĮ naudojimo pokytis 4.12 pav. naudojant AF_PACKET tinklo modulį.

Naudojant AF_PACKET tinklo modulį „Snort“ rezultatai ypač pagerėjo ties 250 Mbps matavimo tašku. Lyginant su standartine konfigūracija paketų apdorojimo kiekis padidėjo 7,32 % naudojant 128 MB ir 24,01 % naudojant 1024 MB. 250 Mbps taške buvo didžiausias „Snort“ paketų apdorojimo išaugimas. Toliau didėjant srautui paketų apdorojimo pokytis mažėja, o nuo 1 Gbps spartos pokytis yra mažesnis nei 5 %. Naudojant AF_PACKET modulį „Suricata“ didžiausias paketų apdorojimo pokytis yra 1500 Mbps matavimo taške. Šiame matavimo taške paketų apdorojimo kiekis padidėjo 16,11 %, kai darbinė atmintis yra 128 MB. Toliau didėjant spartai paketų apdorojimo pokytis mažėja. Suricata“ AF_PACKET modulis veikia kartu su žiedo buferiu (angl. *Ring Buffer*), kuris lygiagrečiai paskirsto paketus tarp procesoriaus gijų. Tokiu būdu nereikalinga papildoma operacija, perkeliant paketus iš priėmimo buferio į skaitymo buferį. Todėl matomi geresni rezultatai prie didesnių spartų.



4.11 pav. „Snort“ ir „Suricata“ paketų apdorojimo kiekio pokytis procentais naudojant AF_PACKET tinklo modulį, kai darbinė atmintis yra 128 MB ir 1024 MB



4.12 pav. „Snort“ ir „Suricata“ visų CPĮ branduolių naudojimo kiekio pokytis procentais naudojant AF_PACKET tinklo modulį, kai darbinė atmintis yra 128 MB ir 1024 MB

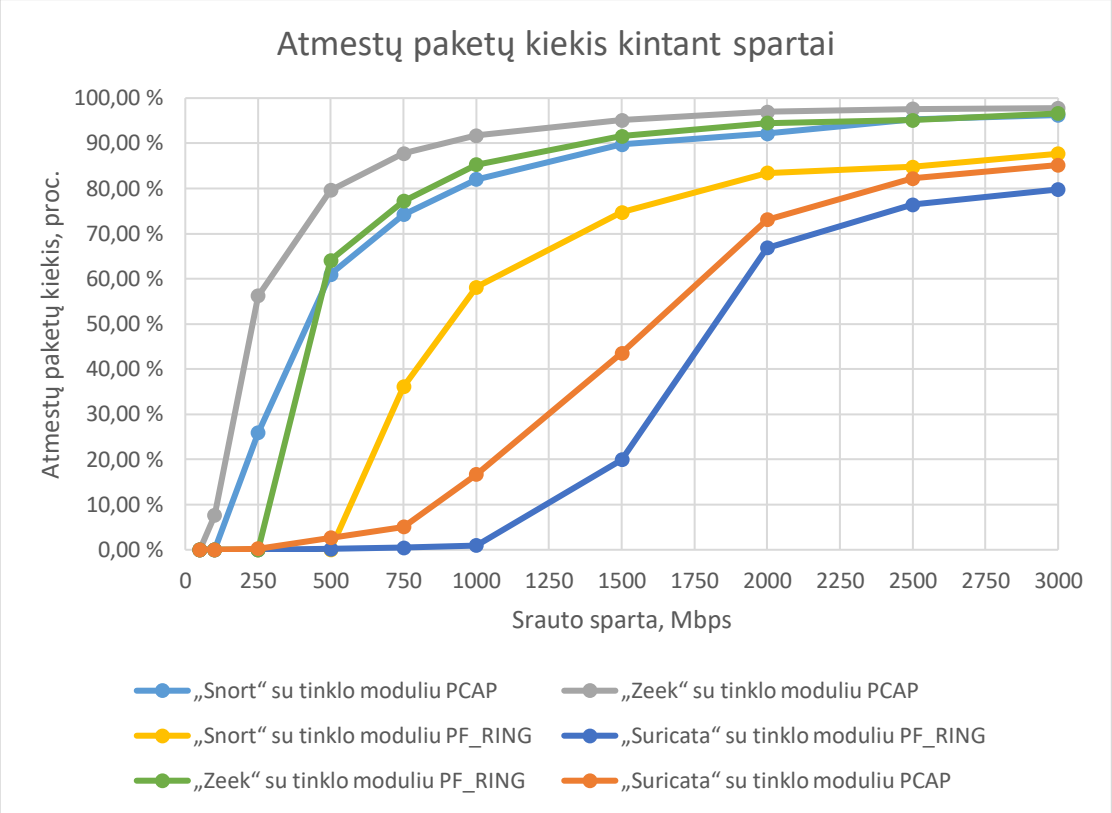
Naudojant AF_PACKET matomi geresni „Snort“ ir „Suricata“ rezultatai, lyginant su įprastine konfigūracija, kuri naudoja PCAP modulį. AF_PACKET gali būti naudojamas kaip konfigūracijos sprendimas, siekiant didesnės spartos. „Snort“ minimalus atmestų paketų kiekis buvo naudojant 1024 MB darbinę atmintį, esant 250 Mbps spartai, tada atmestų paketų kiekis – 1,88%. Sprendimas yra nepakankamas, jeigu tinklas apdoroja didesnį nei 250 Mbps perdavimo srautą. „Suricata“ minimalus atmestų paketų kiekis buvo naudojant 1024 MB darbinę atmintį, esant 750 Mbps – 2,51 %.

4.3.3. Našumo tyrimo rezultatai taikant PF_RING modulį

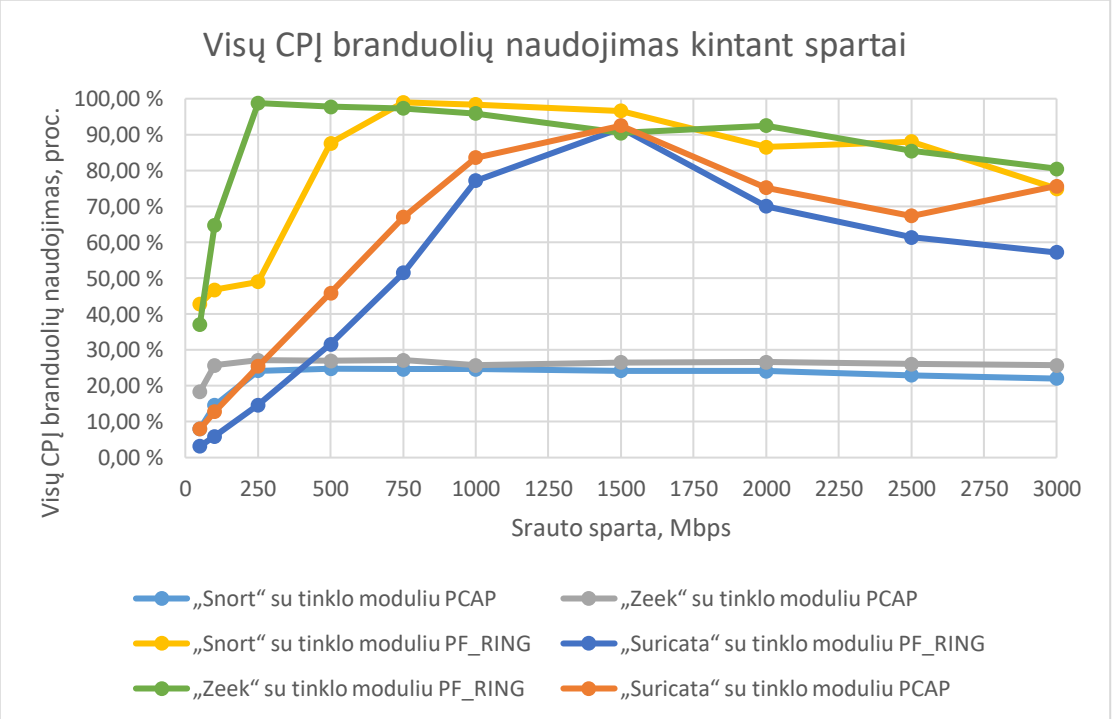
Pakeičiant tinklo jungties modulį į AF_PACKET, tiek „Snort“, tiek ir „Suricata“ ĮAS parodė geresnius rezultatus, atmesdamos mažiau paketų. „Snort“ CPĮ naudojimas liko tas pats, nes naudojamas tik vienas branduolys, o „Suricata“ pagerino CPĮ naudojimą apie 10 %.

Toliau tiriamas ĮAS našumas, naudojant PF_RING modulį. Modulio pagrindinis privalumas, kad paketų priėmimas vyksta žiedu. „Linux“ operacinė sistema naudojant NAPI (angl. *New API*) kopijuoja paketus iš tinklo plokštės į PF_RING buferį, iš kurio vartotojo programos skaito paketus. Kiekviename CPĮ cikle atliekamas paketų kopijavimas ir skaitymas vienu metu. PF_RING gali paskirstyti įeinančius paketus į kelis apdorojimo žiedus vienu metu [7]. Naudojantis šiuo funkcionalumu galima sukurti ne vieną, bet kelis procesus vienu metu. Šį funkcionalumą gali naudoti „Snort“, „Suricata“ ir „Zeek“ ĮAS. Pagal PF_RING dokumentaciją procesai priskiriami į tą patį telkinį, kuriame paskirstomi įeinantys paketai [7]. Srauto metu matuoti parametrai: CPĮ naudojimas procentais ir atmestų paketų kiekis procentais. Bendras atmestų paketų kiekis kintant spartai pateikiamas 4.13 pav, CPĮ naudojimo 4.14 pav. Paketų apdorojimo kiekio pokyčio grafikas pateikiami 4.15 pav. CPĮ naudojimo pokyčio grafikas pateikiamas 4.16 pav.

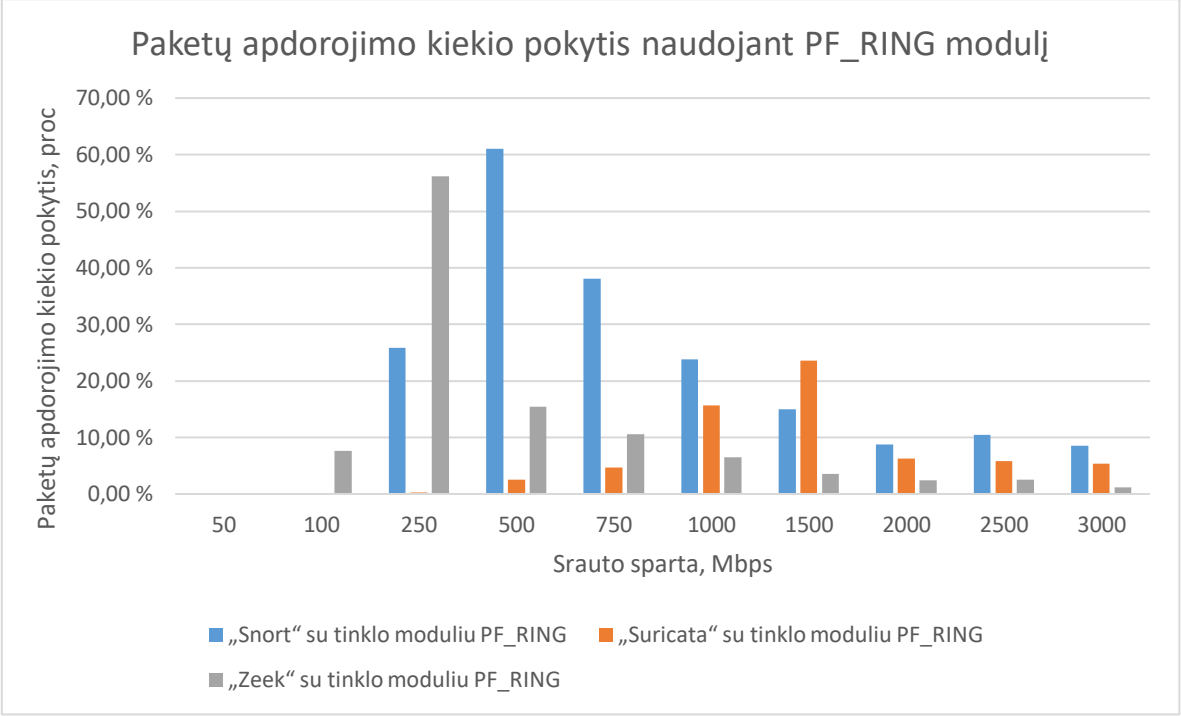
Sukurti keturi atskiri „Snort“ ir „Zeek“ procesai kiekvienam iš branduolių, įeinantys paketais žiedo principu buvo paskirstomi į atskirus procesus apdorojimui. Naudojant PF_RING pasiekiami geresni „Snort“ rezultatai. Iki 500 Mbps perdavimo srauto nebuvo atmestų paketų. Lyginant su pradine „Snort“ konfigūracija, paketų apdorojimo kiekis padidėjo 60,99 %. Pasiekiant 750 Mbps ribą, paketų apdorojio kiekis padidėjo 36,1 %, kas yra 2 kartus daugiau nei naudojant įprastinę konfigūraciją. Grafike matyti, kad atmestų paketų kiekio kreivės kitimas išlieka toks pats – staigus šuolis ir užlinkimas artėjant prie didesnių spartų. Naudojant PF_RING, staigus atmestų paketų šuolis vyksta vėliau ir įvyksta ties 750 Mbps riba. Skirtumas tarp kreivių sumažėja, didėjant spartai virš 1 Gbps.



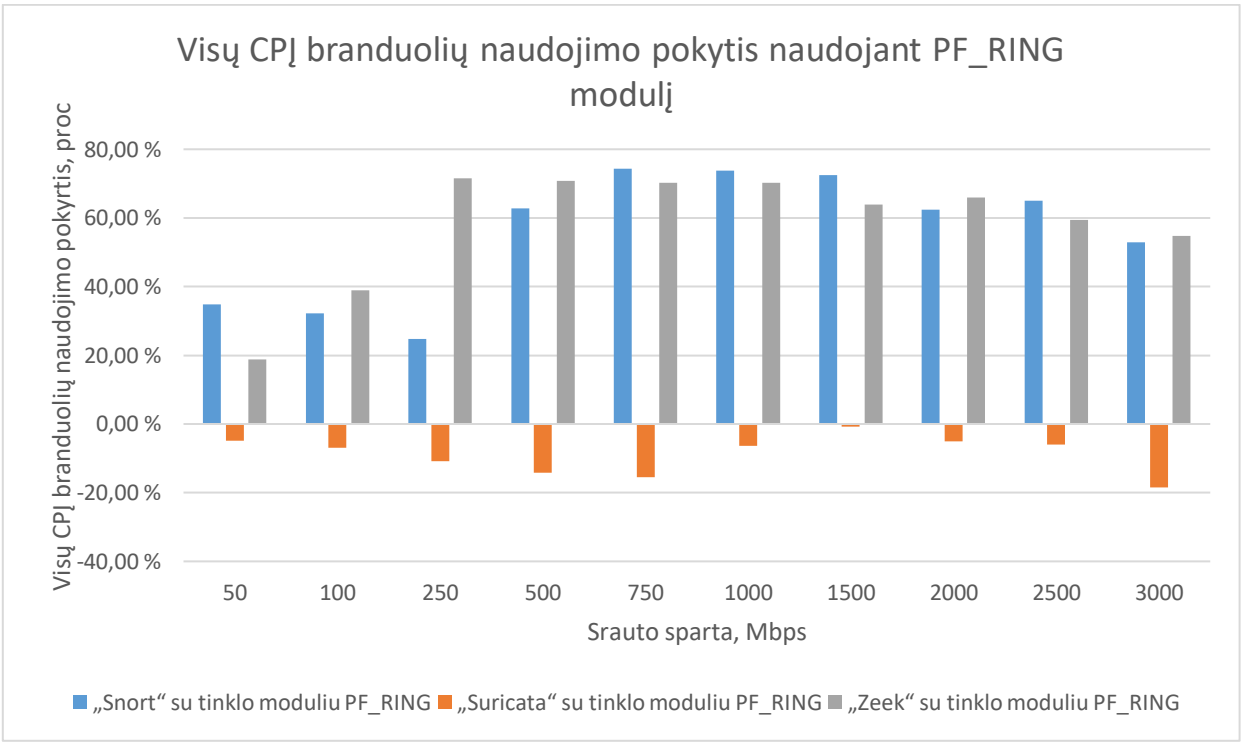
4.13 pav. „Snort“, „Suricata“ ir „Zeek“ atmestų paketų kiekis procentais kintant spartai, naudojant PF_RING modulį



4.14 pav. „Snort“, „Suricata“ ir „Zeek“ visų CPĮ branduolių naudojimas procentais kintant spartai, naudojant PF_RING modulį



4.15 pav. „Snort“, „Suricata“ ir „Zeek“ paketų apdorojimo kiekio pokytis procentais naudojant PF_RING tinklo modulį



4.16 pav. „Snort“, „Suricata“ ir „Zeek“ visų CPĮ branduolių naudojimo kiekio pokytis procentais naudojant PF_RING tinklo modulį

Naudojant PF_RING modulį, „Suricata“ rezultatų skirtumas nėra toks didelis kaip „Snort“, tačiau paketų apdorojimo kiekio padidėjimas yra svarbus. Lyginant su standartine konfigūracija, „Suricata“ paketų apdorojimo kiekis padidėjo 15,70 % 1000 Mbps spartai. Didžiausias skirtumas tarp „Suricata“ kreivių yra 1000–1500 Mbps ruože, nuo 2000 Mbps kreivių skirtumas tampa apie 5 %. PF_RING modulis nuima dalį darbo nuo procesoriaus priimant paketus, sumažindamas API užklausų, tokiu būdu įgaunamas efektyvesnis paketų priėmimas. „Snort“ tyrime PF_RING leidžia sukurti atskirus procesus taip sukuriant daugiagyslę funkciją ĮAS, o „Suricata“ naudoja PF_RING efektyviau paskirstydama CPĮ resursus.

Naudojant PF_RING „Zeek“ rezultatai ypač pagerėjo iki 500 Mbps ribos. Skirtingai nei „Snort“ ar „Suricata“ ĮAS „Zeek“ veikia telkinio principu [17]. Klasteryje išskiriami procesai vadinami darbiniais (angl. *Workers*), kurie priima srautą. Tai yra atskiri mazgai, išskirstyti po visą tinklą arba viename mazge priimantys įeinantį srautą. Klasteryje veikia įgaliotinis (angl. *Proxy*) procesas, valdantysis (angl. *Manager*) procesas, skirtas telkinio valdymui ir procesas, skirtas informacijos rašymui į žurnalinius įrašus (angl. *Logger)*. Iš viso gaunami 7 aktyvūs „Zeek“ procesai veikiant PF_RING režimu. Dokumentacijoje rekomenduotina valdymo, įgaliotinio ir žurnalinių failų procesus paleisti skirtingose mašinose, o darbinius procesus laikyti vienoje mašinoje dėl paketų paskirstymo. „Zeek“ naudojant PF_RING tinklo modulį matoma, kad esant 250 Mbps spartai, paketų apdorojimo kiekis padidėjo 56,2 %. Pastebima riba tarp 250-500 Mbps, atsiranda CPĮ resursų limitas esant 500 Mbps ribai – matomas staigus atmestų paketų šuolis nuo 0 % iki 64,1 %. Didinant perduodamų duomenų spartą, atmestų paketų kiekis pasiekia 90 % ribą. Staigus šuolis atmestų paketų kiekyje yra dėl kelių priežasčių: „Zeek“ atlieka detalesnę paketų analizę nei „Snort“ ir „Suricata“, ir papildomai „Zeek“ procesai reikalauja daugiau CPĮ resursų. Nuo 500 Mbps taško matyti, kad tyrimo schemoje CPĮ nebesusidoroja su veikiančias procesais, todėl didėja atmestų paketų kiekis.

Matyti, kad „Snort“ ir „Suricata“ sąlyginai išlaiko tolygų CPĮ resursų naudojimą, didėjant perdavimo spartai. „Zeek“ kreivė staigiai kyla aukštyn ties 250 Mbps, tuo metu CPĮ dirba visu pajėgumu, siekiant apdoroti srautą ir suvaldyti „Zeek“ procesus. „Zeek“ CPĮ piko taškas buvo esant 250 Mbps, „Snort“ esant 750 Mbps, o „Suricata“ esant 1500 Mbps.

Tyrimo rezultatai parodė, kad naudojantis PF_RING tinklo moduliu gaunami žymiai geresni rezultatai, lyginant su standartine ĮAS konfigūracija, kai naudojamas PCAP tinklo modulis. Didžiausias atmestų paketų kiekio sumažėjimas buvo „Snort“, esant 500 Mbps spartai atmestų paketų kiekis nukrito nuo 61 % iki 0 %. Panašų pokytį stebime ir su „Zeek“ ĮAS, esant 250 Mbps

spartai atmestų paketų kiekis sumažėjo nuo 56 % iki 0 %. Naudojantis PF_RING moduliu, „Suricata“ esant 1000 Mbps perdavimo spartai sumažino atmestų paketų kiekį iki 1 %.

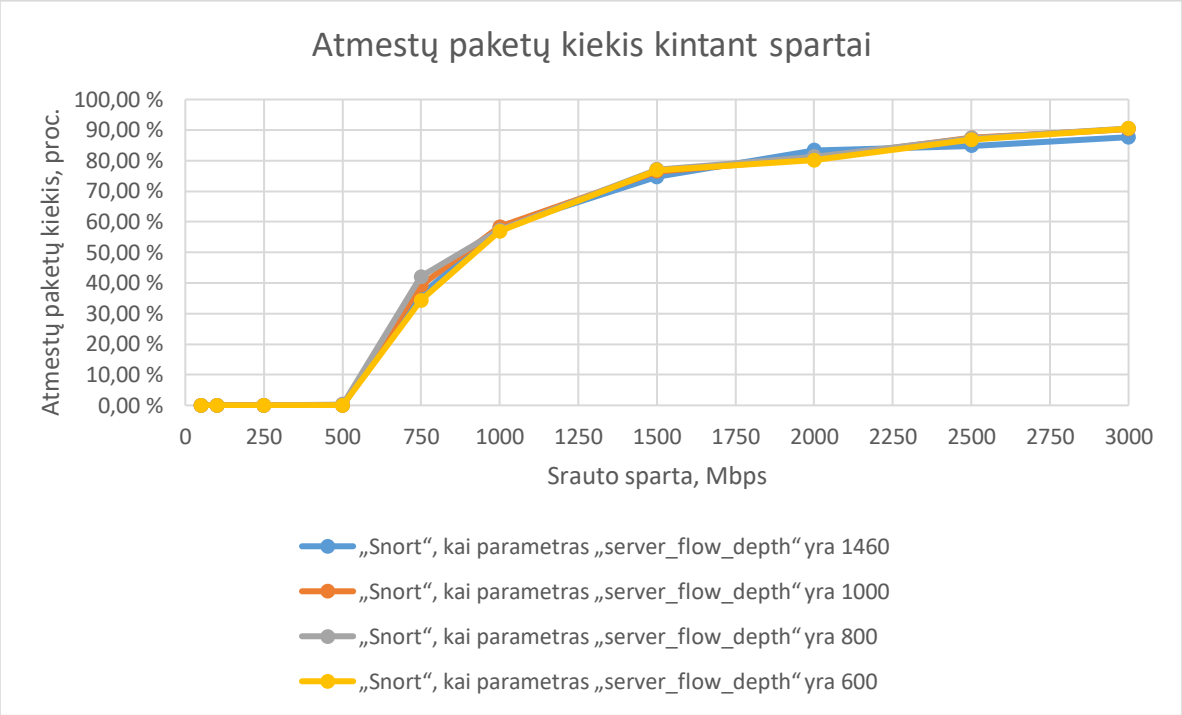
4.3.4. Našumo tyrimo rezultatai taikant PF_RING modulį ir konfigūracijos pakeitimus

Našumo tyrimo rezultatai parodė, kad esant standartiniai konfigūracijai ĮAS negali apdoroti didelio srauto kiekio. Naudojant papildomus tinklo modulius, galima gauti geresnius našumo rezultatus. Kita ĮAS optimizavimo dalis yra ĮAS konfigūracijos parametrai. Svarbu paminėti, kad ĮAS optimizavimas priklauso nuo stebimo srauto charakteristikų. Skirtingi konfigūracijos parametrai gali turėti skirtingą įtaką srauto apdorojimui. Poskyryje apžvelgiami bendri konfigūracijos parametrai, kurie nepritaikyti specifiniam srautui. Keičiami konifgūracijos parametai: atminties kiekis skirtas paketų apdorojimui, paketų apdorojimo eilės dydis ir atpažinimo algoritmai.

Tiriamos „Snort“ ir „Suricata“ ĮAS ir sistemų konfigūracijos pakeitimai, „Zeek“ ĮAS nėra testuojama, nes neturi prieinamų vartotojui konfigūracinių failų. Vartotojui nesuteikiama galimybė keisti nustatytuosius parametrus kaip ĮAS apdoroja srautą. „Snort“ ir „Suricata“ ĮAS konfigūracijos pakeitimai testuojami kartu su PF_RING tinklo moduliu, nes naudojantis šiuo moduliu pasiekiami geriausi našumo rezultatai. Siekiame patikrinti kaip konfigūracijos pakeitimai darys įtaką našumui ir kokį maksimalų našumą, esant tyrimo sąlygomis, galima gauti. „Snort“ parametrai keičiami *snort.conf* konfigūraciniame faile, „Suricata“ parametrai keičiami *suricata.yml* konfigūracijos faile.

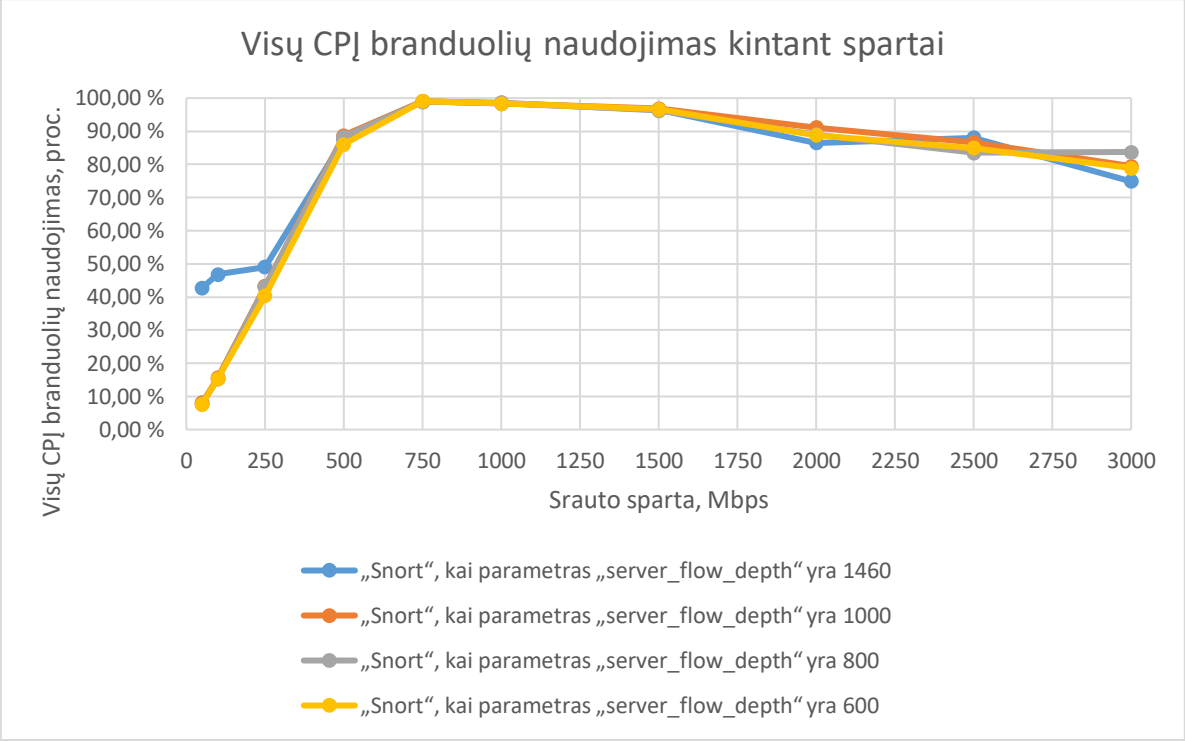
„Snort“ dokumentacijoje nurodomi konfigūracijos parametrai, kurie gali pagerinti ĮAS našumą [39]. Vienas iš jų yra HTTP paketų apdorojimo ilgis. Keičiamas „Snort“ priešprocesininio modulio „http_inspect“ konfigūracinis parametras „server_flow_depth“. Priešprocesinis modulis atsakingas už HTTP duomenų dekodavimą iš paketo, HTTP atranščių dekodavimą [39]. Konfigūracinis parametras „server_flow_depth“ nurodo, kiek baitų reikia apdoroti iš serverio siunčiamų paketų. Standartiškai parametras nustatytas apdoroti 1460, kas yra visas kadro ilgis. Pagal „Google“ tyrimą, dažniausiai HTTP pagrindinė informacija susideda iš 700–800 baitų. Mažinant „server_flow_depth“ parametrą, bus reikalingi mažesni resursai, apdorojant serverio siunčiamus atgalinius paketus klientui, tačiau mažinant parametro dydį kartu didėja atakos tikimybė. Tikimybė didėja, nes mažesnis paketo kiekis duomenų bus išanalizuota ĮAS. Parametras „client_flow_depth“ nurodo kliento siunčiamų paketų analizės dydį, šis parametras neliečiamas,

siekiant aptikti potencialias atakas. Bendras atmestų paketų kiekio kitimo grafikas, kai keičiama „server_flow_depth“ parametro vertė, pateikiamas 4.17 paveiksle.



4.17 pav. „Snort“ atmestų paketų kiekis kintant srauto spartai naudojant PF_RING tinklo modulį, kai parametro „server_flow_depth“ reikšmė yra 600, 800, 1000 ir 1460

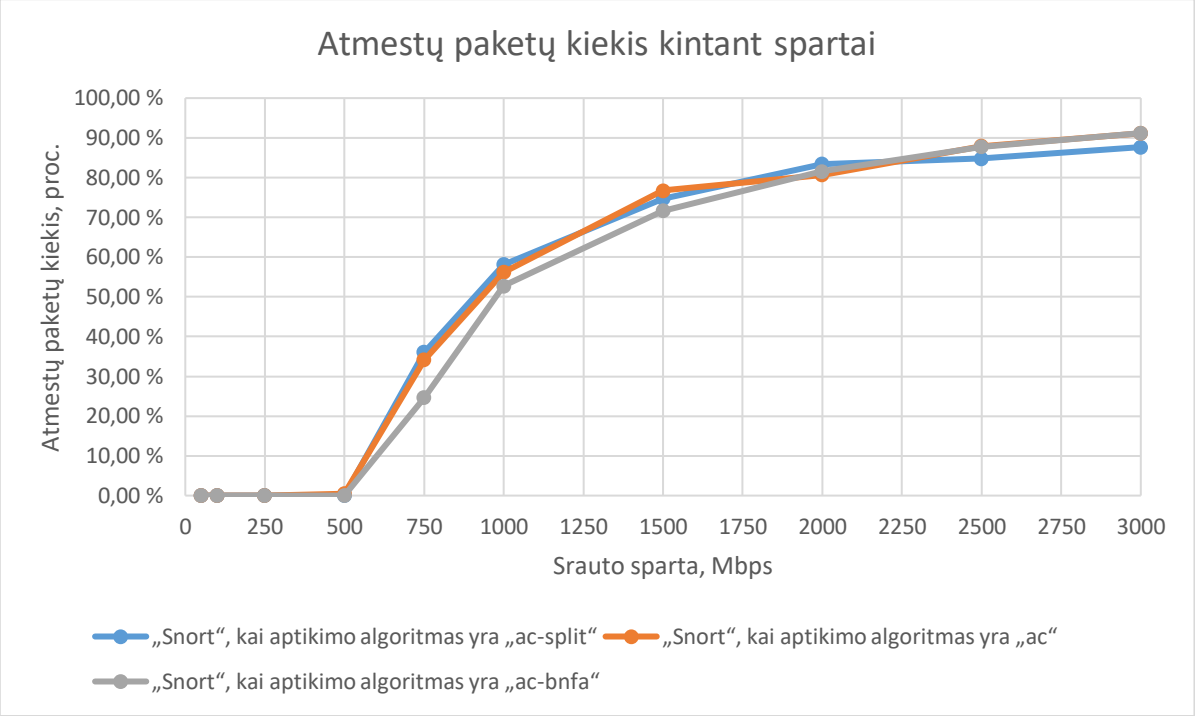
Gauti rezultatai rodo kreivių pokytį 500–1000 Mbps ruože, bet pokyčiai yra minimalūs. Atmestų paketų kiekis išliko beveik toks pats visose kreivės taškuose, didžiausias skirtumas buvo ties 750 Mbps perdavimo sparta. Kai „server_flow_depth“ yra 1000, atmestų paketų kiekis – 39,25 %, kai 800 – 42,04 % ir kai 600 – 34,36 %. Sumažinus parametro vertę nuo 1000 iki 600 apdorojimo baitų, atmestų paketų kiekis sumažėjo 5 %. Iš pirmo žvilgsnio tai nėra didelis pokytis, tačiau ĮAS dirbant beveik maksimaliu pajėgumu, tai yra reikšmingas pokytis. Galima daryti išvadą, jog dar labiau sumažinus parametro vertę, pamatytume dar mažesnį atmestų paketų skaičių, tačiau didintume atakos praleidimo tikimybę. Parametras „server_flow_depth“ gali turėti ir didesnę įtaką, jeigu srautą sudarytų didesnis kiekis paketų, kur serveris atsako į klientų užklausas. Bendras CPĮ naudojimo grafikas keičiant „server_flow_depth“ vertę pateikiamas 4.18 paveiksle.



4.18 pav. „Snort“ CPĮ naudojimas kintant srauto spartai, naudojant PF_RING tinklo modulį, kai parametro „server_flow_depth“ reikšmė yra 600, 800, 1000, 1460

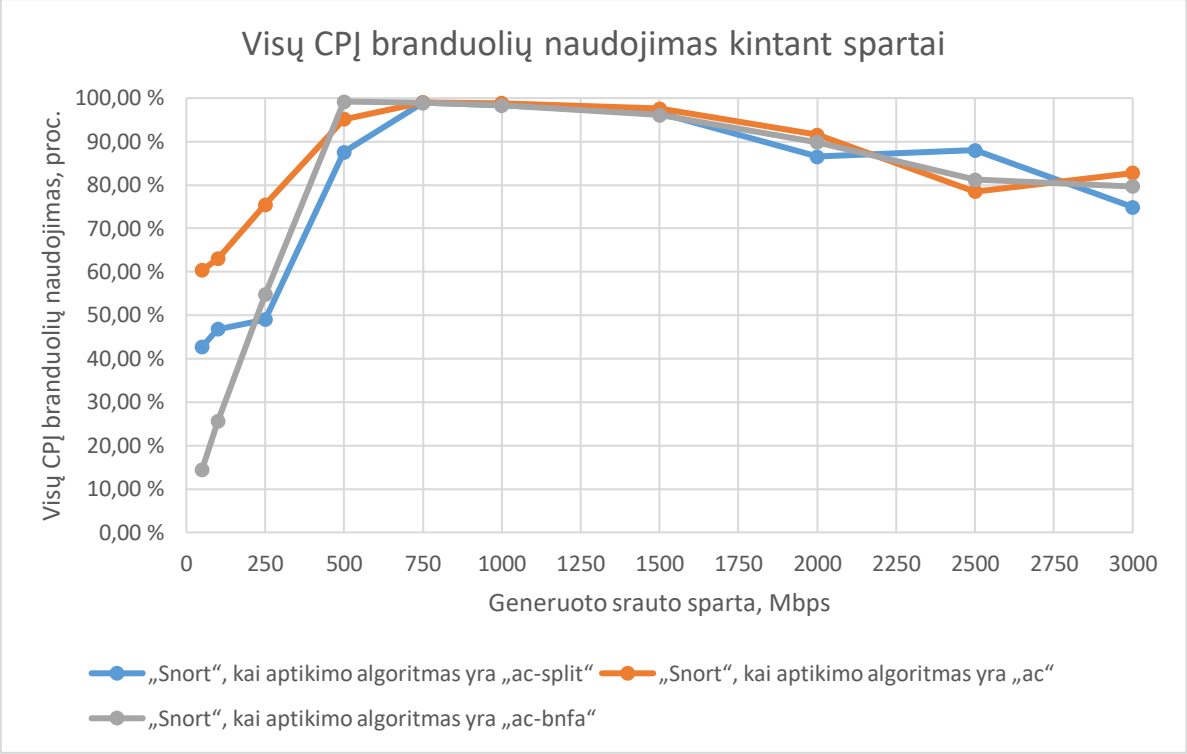
Iš gautų rezultatų matyti, kad „Snort“ visiškai naudoja 4 branduolius paketų apdorojimo procese. Atlaisvinti resursai, sumažinus parametro vertę, yra priskiriami kitam apdorojimo procesui, todėl grafike nematyti CPĮ resursų sumažėjimo. CPĮ naudojimas mažėja tik artėjant prie spartų, kurios viršija 1500 Mbps, nuo čia prasideda virtualiosios mašinos resursų limito įtaka ĮAS.

Kitame tyrimo punkte keičiamas „Snort“ šablono aptikimo algoritmo parametras „search-method“. Iš tinklo priimamas paketas yra dekoduojamas. Dekodavimo procese atrenkama informacija apie naudojamus protokolus pakete. Ši informacija išsaugoma ir persiunčiama į prieš procesinius modulius ir aptikimo mechanizmą. Aptikimo mechanizmas naudoja šablono aptikimo algoritmą, kuris standartiškai yra „ac-split“ (angl. *Aho Corasic*). Aptikimo mechanizmas taiko šablono algoritmą, siekiant susieti esamas taisykles ir paketo duomenis. Jeigu paketo duomenys atitinka taisyklę ar taisykles – generuojamas pranešimas. „Snort“ dokumentacija pateikia galimus algoritmus, siekiant didesnio našumo [41]. Tyrimui pasirenkami trys algoritmai pagal dokumentaciją, turintys didžiausią našumą. Bendras atmestų paketų kiekis, keičiant šablono aptikimo algoritmą, pateikiamas 4.19 paveiksle.



4.19 pav. „Snort“ atmestų paketų kiekis kintant srauto spartai, naudojant PF_RING tinklo modulį, kai aptikimo algoritmas yra „ac“, „ac-split“ ir „ac-bnfa“

Iš rezultatų matyti, kad prasčiausi rezultatai yra naudojant „ac-split“ algoritmą. Taip yra dėl to, kad algoritmas nebando grupuoti taisyklių ir kiekviena taisyklė taikoma kiekvienam paketui. Kuo perduodamas srautas yra vienodesnis, tuo našumas būtų didesnis, naudojantis šiuo algoritmu. Pakeitus į „ac“ algoritmą, rezultatai neturėjo reikšmingo pokyčio, todėl „ac“ ir „ac-split“ kreivės yra beveik identiškos. Didžiausias skirtumas matomas, naudojant kitą „ac“ algoritmo variantą – „ac-bnfa“. Pagal dokumentaciją, tai yra nedaug atminties reikalaujantis ir aukšto našumo algoritmas. Geresni rezultatai matyti nuo 750 Mbps iki 2000 Mbps. Esant 750 Mbps spartai, atmestų paketų kiekis sumažėjo 10 %, lyginant su „ac“ ir „ac-split“ algoritmais. Esant „ac“ algoritmui, atmestų paketų kiekis 750 Mbps spartai yra 36,1 %, o „ac-bnfa“ yra 24,7 % Kituose taškuose rezultatai skiriasi apie 5 %. Bendras CPĮ naudojimo grafikas, kintant spartai, pateikiamas 4.20 paveiksle.

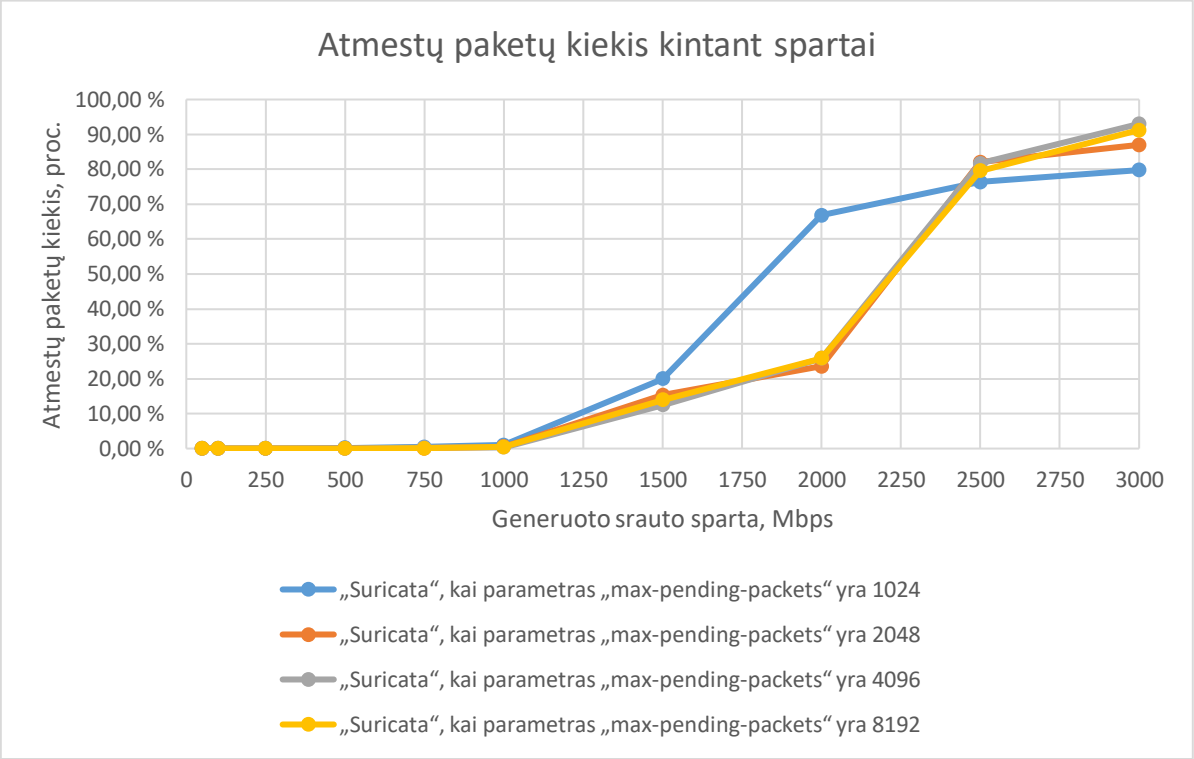


4.20 pav. „Snort“ CPĮ naudojimas kintant srauto spartai, naudojant PF_RING tinklo modulį, kai aptikimo algoritmas yra „ac“, „ac-split“ ir „ac-bnfa“

Daugiausiai resursų sunaudoja „ac“ algoritmas. Esant 50 Mbps spartai, CPĮ naudojimas siekia 60,38 %. Dokumentacijoje pateikta, kad „ac“ algoritmas reikalauja didesnių sistemos resursų, taip suteikiant didesnį našumą. Rezultatuose matome, kad „ac“ algoritmas ankščiau naudoja didelius CPĮ resursus, bet reikšmingo papildomo našumo nesuteikia. Efektyviausias CPĮ resursų naudojimas yra „ac-bnfa“, lyginant su kitais algoritmais. Esant 50 Mbps spartai CPĮ naudojimas naudojantis algoritmu „ac“– 60,4 %, „ac-split“ – 42,8 %, o „ac-bnfa“ – 14,5 %. Iš tirtų algoritmų efektyviausias „ac-bnfa“ algoritmas, nes mažiau apkrauna CPĮ resursus, todėl „Snort“ programinė įranga gali daugiau apdoroti paketų.

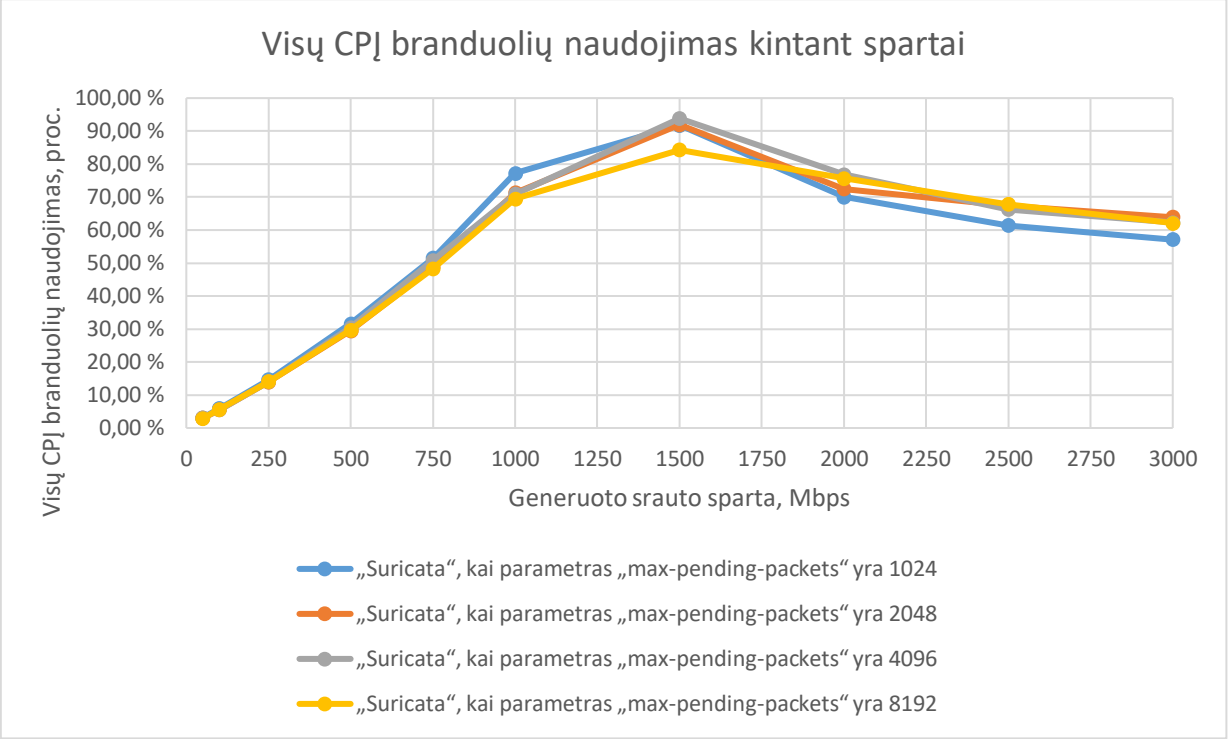
„Suricata“ ĮAS dokumentacijoje pateikiami konfigūracijos parametrų pakeitimai galintys pagerinti ĮAS našumą [12]. „Suricata“ tyrimo punkte keičiamas „max-pending-packets“ parametras. Šis parametras nustato, kiek maksimaliai atpažinimo mechanizmas gali apdoroti paketų vienu metu. Jeigu paketų yra daugiau negu priimamų paketų eilės dydis – paketai yra atmetami, tai mažina ĮAS našumą. Didinant parametrą, „Suricata“ procesoriaus gijos bus labiau apkraunamos iki tam tikros ribos. Peržengus tam tikrą ribą gijos nebespės apdoroti paketų, nes paketų bus per daug. Standartinė parametro vertė yra 1024, tai pagal „Suricata“ kūrėjus yra

konservatyvi reikšmė. Atliekamas tyrimas didinant „max-pending-packets“ parametro reikšmę kas du kartus. Bendras atmestų paketų kiekis pateikiamas 4.21 paveiksle.



4.21 pav. „Suricata“ atmestų paketų kiekis kintant srauto spartai, naudojant PF_RING tinklo modulį, kai parametro „max-pending-packets“ reikšmė yra 1024, 2048, 4096 ir 8192

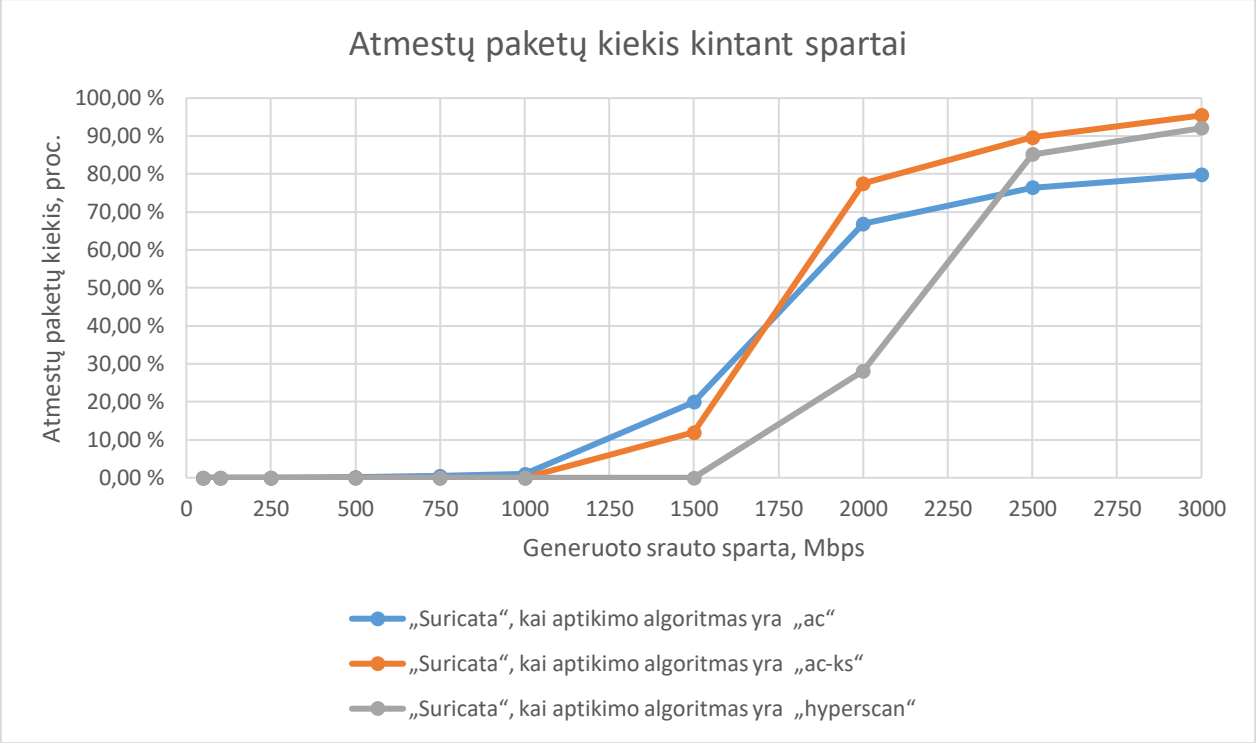
Padidinus parametro vertę virš 1024 matomi geresni rezultatai, esant 1500 ir 2000 Mbps perdavimo spartai. Esant 1500 Mbps spartai, didinant „max-pending-packets“ parametro vertę, atmestų paketų kiekis sumažėjo 6–7 %. Didžiausias skirtumas stebimas ties 2000 Mbps srauto sparta, parametro pokyčiai sumažina atmestų paketų kiekį nuo 67 % iki 25 %. Toliau didinant spartą, atmestų paketų kiekis buvo didesnis grafikuose, kur parametras buvo padidintas. Tyrime naudojant šią schemą greitai pasiekiame ribą, kai didesnis „max-pending-packets“ didinimas neduoda didesnės naudos. Bendras CPĮ naudojimo grafikas pateikiamas 7.16 paveiksle. CPĮ naudojimo skirtumas matomas 750-2000 Mbps spartos intervale. Intervale CPĮ naudojimas sumažėjo 3 %, tik padidinus parametro vertę iki 8192 CPĮ naudojimas sumažėjo esant 2000 Mbps sparta nuo 91,8 % iki 84,3 %, mažesnis CPĮ naudojimas nesumažino atmestų paketų skaičiaus.



4.22 pav. „Suricata“ CPĮ naudojimas kintant srauto spartai, naudojant PF_RING tinklo modulį, kai parametro „max-pending-packets“ reikšmė yra 1024, 2048, 4096 ir 8192

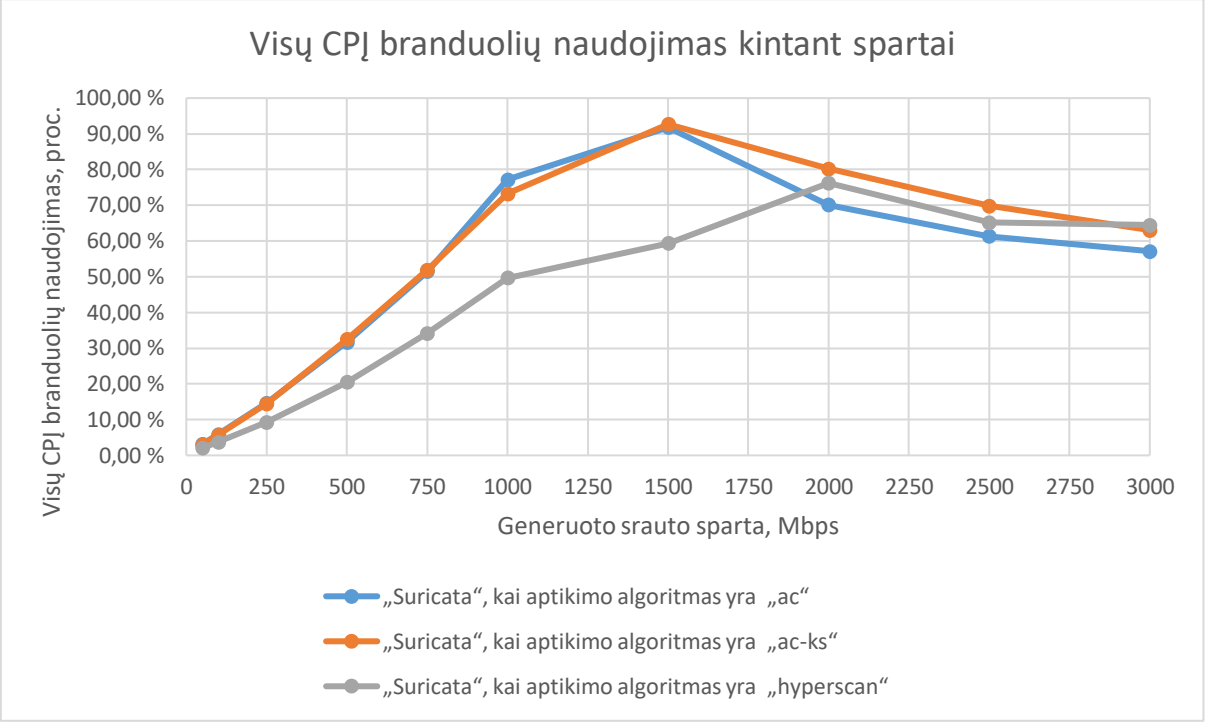
„Suricata“ ir „Snort“ palaiko skirtingus variantų atpažinimo algoritmus, tokius kaip „ac“ ir „ac-ks“. „Suricata“ palaiko ir „Intel“ kompanijos sukurtą „Hypercan“ algoritmą, kuris pasižymi dideliu našumu. „Hyperscan“ algoritmas nebuvo ištestuotas „Snort“ ĮAS, nes testuojama versija nepalaiko „Hyperscan“ įskiepio. „Hyperscan“ naudoja hibridinius automatinius metodus, leidžiančius vienu metu palyginti didelį taisyklių kiekį, naudojantis reguliariomis išraiškomis [5]. Sukompiliavus „Suricata“ su „hyperscan“ algoritmu atlikti testai su skirtingais algoritmais. Bendras atmestų paketų kiekio grafikas pateikiamas 4.23 paveiksle.

Iš grafiko kreivių matyti, kad prasčiausiai rezultatai yra naudojant „ac“ algoritmą, kuris „Suricata“ ĮAS yra standartinis pasirinkimas. Pakeitus algoritmą į „ac-ks“ esant 1500 Mbps sparta atmestų paketų kiekis, sumažėjo nuo 20 % iki 11,9 %, toliau didinant spartą „ac-ks“ algoritmas prasčiau tvarkosi su įeinančia sparta ir atmestų paketų kiekis padidėja iki 77,5 %. Geriausi rezultatai yra „yperscan“ algoritmo. Naudojant šį algoritmą „Suricata“ sugebėjo išlaikyti 0 % atmestų paketų kiekį, esant 1500 Mbps, tai yra 20 % skirtumas, naudojant vietoj standartinio „ac“ algoritmo. Padidinus spartą iki 2000 Mbps, „hyperscan“ algoritmas išlaiko efektyvesnį paketų apdorojimą, nes atmestų paketų kiekis siekia 28,1 %. Atmestų paketų kiekis sumažėjo 40 % esant šiuo tašku. Toliau didinant perdavimo spartą skirtumai tarp algoritmų nėra dideli, nes esant šioms spartoms atsiranda CPĮ resursų ribojimas.



4.23 pav. „Suricata“ atmestų paketų kiekis kintant srauto spartai, naudojant PF_RING tinklo modulį, kai aptikimo algoritmas yra „ac“, „ac-ks“ ir „hyperscan“

Bendras CPĮ naudojimo grafikas pateikiamas 4.24 pav. Atmestų paketų kiekio grafiko rezultatai atitinka CPĮ naudojimo grafiko kreivių kitimą. CPĮ naudojimo skirtumai tarp „ac“ ir „ac-ks“ algoritmų yra minimalūs. Labiausiai skiriasi ties 1000 Mbps sparta, o toliau didinat spartą, CPĮ naudojimas, naudojant „ac-ks“ algoritmą išauga, ką matome ir su atmestų paketų kiekio grafiku. Iš matavimų rezultatų matyti, kad „hyperscan“ yra efektyviausias algoritmas, gebantis sumažinti CPĮ naudojimą nuo 10 iki 30 %. Algoritmas „hyperscan“ efektyviai taiko taisyklių pritaikymą, todėl CPĮ resursai sumažėja. Didėjant spartai algoritmo efektyvumas CPĮ atžvilgiu didėja, nes prie didesnių spartų CPĮ naudojimas yra mažesnis, lyginant su standartiniais algoritmais. Didžiausias CPĮ naudojimas buvo esant 2000 Mbps ribai, kai CPĮ pasiekė 76,2 %. Toliau didėjant spartai, atsiranda CPĮ resursų ribos, kurių negali kompensuoti ir „hyperscan“ algoritmas.

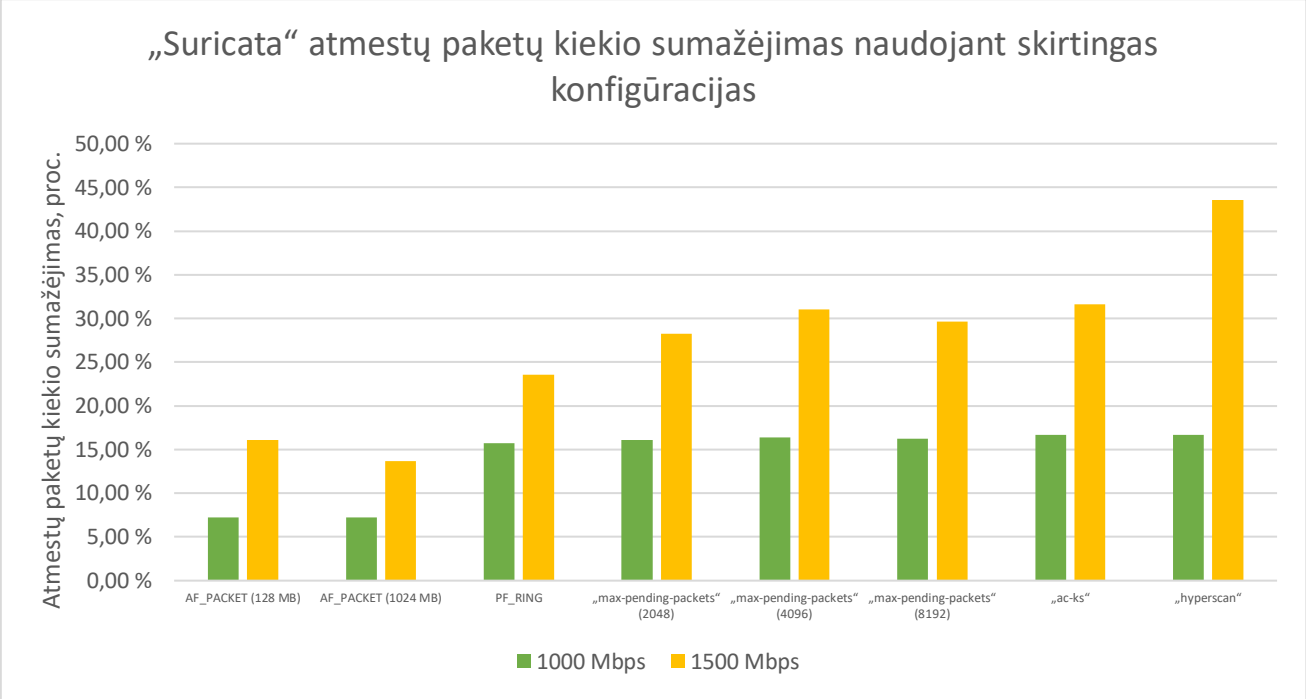


4.24 pav. „Suricata“ CPĮ naudojimas kintant srauto spartai, naudojant PF_RING tinklo modulį, kai aptikimo algoritmas yra „ac“, „ac-ks“ ir „hyperscan“

Iš atlikto tyrimo galima apibendrinti gautus rezultatus. Keičiant priešprocesoriaus parametro „http_inspect“ vertes galima gauti efektyvesnį CPĮ naudojimą „Snort“ ĮAS, tačiau parametro įtaka labiau priklauso nuo generuojamo srauto charakteristikų. Tyrime keičiant „server_flow_depth“ parametrą, matėme 750–1500 Mbps intervale 1–3 % atmestų paketų kiekio sumažėjimą. Keičiant „Snort“ atpažinimo algoritmus geriausi rezultatai buvo naudojant „ac-bnfa“ algoritmą. Naudojant „ac-bnfa“ algoritmą, atmestų paketų kiekis sumažėjo 12 % esant 750 Mbps ribai.

Keičiant „Suricata“ parametro „max-pending-packets“ vertę, didžiausias pokytis buvo esant 2000 Mbps ribai. Padidinus vertę, atmestų paketų kiekis sumažėjo apie 30 %. Likusiuose matavimo taškuose skirtumai yra minimalūs. Iš „Suricata“ ĮAS atpažinimo algoritmų geriausi rezultatai yra „hyperscan“ algoritmo. Naudojant šį algoritmą buvo pasiekta 0 % atmestų paketų riba, esant 1500 Mbps srauto taškų. Algoritmas daugiausiai sumažino CPĮ naudojimo kiekį, esant 2000 Mbps riba.

Tyrimų rezultatai parodė, kad „Suricata“ sistema yra našiausia, apdorojant aukštą srauto intensyvumą. „Suricata“ sistemos atmestų paketų kiekio sumažėjimas, lyginant su standartine konfigūracija pateikiamas 4.25 pav.



4.25 pav. „Suricata“ atmestų paketų kiekio sumažėjimas naudojant skirtingas konfigūracijas, lyginant su standartine konfigūracija

Pateikiami 1000 Mbps ir 1500 Mbps matavimo taškų rezultatai. Šiuos taškuose matomas didžiausias atmestų paketų sumažėjimas, lyginant su standartine konfigūracija. Iš rezultatų galima teigti, kad PF_RING tinklo modulis yra spartesnis nei AF_PACKET modulis. Naudojant PF_RING tinklo modulį „Suricata“ atmestų paketų kiekis sumažėjo 16 %, o AF_PACKET 7 % esant 1000 Mbps. Matyti, kad „Suricata“ visiškai apdoroja replikuojamą srautą, nes atmestų paketų kiekis yra lygūs 0 % naudojant PF_RING. Didinant spartą iki 1500 Mbps reikalinga papildoma konfigūracija, siekiant kuo mažesnio atmestų paketų kiekio. Naudojant „ac-ks“ algoritmą, atmestų paketų kiekis sumažėjo 31 %, o naudojant „hyperscan“ algoritmą, atmestų paketų kiekis sumažėjo 43 %, esant 2000 Mbps. Galima teigti, kad „Suricata“ yra sparčiausia iš tirtų sistemų ir geriausi rezultatai pasiekiami kartu naudojant PF_RING modulį ir „hyperscan“ algoritmą.

IŠVADOS

Baigiamojo magistro darbo tikslas – išanalizuoti pasirinktų programinių paketų skirtumus, atpažįstant atakas ir apdorojant įeinantį srautą. Buvo nagrinėjami „Snort“, „Suricata“ ir „Zeek“ programiniai paketai. Tirti tokie parametrai: atakų atpažinimas, pranešimų būsenos, paketų apdorojimo greitis, atmestų paketų ir visų CPĮ naudojimo kiekis procentais. Matavimai buvo atliekamai naudojant „CICIDS“ ir „ISCX“ duomenų rinkinius atpažįstant atakas ir duomenų rinkinį „CIC DoS“ tiriant našumą. Ištirtas programinių paketų našumas, keičiant konfigūracijos parametrus ir tinklo sąsajos modulius. Atlikus tyrimus galima suformuluoti tokias išvadas:

1. Programinė įranga neaptiko daugelio duomenų rinkiniuose vykdomų atakų. Kuo sudėtingesnė ataka, tuo sunkiau įsibrovimo aptikimo sistemai atpažinti ataką. Iš 20 atliktų atakų „Suricata“ atpažino 60 %, „Snort“ – 25 % ir „Zeek“ – 45 % visų atliktų atakų.
2. Pranešimų kiekis, nurodantis tikslią ataką, užima nedidelę dalį visų generuojamų pranešimų kiekio. Tiriant „CICIDS“ duomenis „Suricata“ ir „Snort“ tikslių pranešimų kiekis buvo iki 1 %. Tiriant „ISCX“ duomenis „Snort“ ir „Suricata“ generavo iki 6,55 % tikslių pranešimų. Dažniausiai klydo „Snort“ sistema, generuodama klaidingai teigiamus pranešimus nuo 42 % iki 88 %. „Snort“ generavo perspėjimo pranešimų nuo 20 % iki 57 %. Daugiausiai perspėjimo pranešimų generavo „Suricata“ intervale nuo 0 % iki 75 %.
3. Nustačius CPĮ branduolių skaičių iki 1, „Snort“ ir „Zeek“ sistemos, paremtos vienos gijos architektūra, yra našesnės esant failų skaitymo režimui nei „Suricata“, kuri paremta kelių gijų architektūra. „Snort“ paketų apdorojimo greitis buvo 18 %, o „Zeek“ 30 % didesnis nei „Suricata“.
4. Atlikę pirminius našumo tyrimo matavimus, matome, kad naudojant standartinę konfigūraciją, „Snort“ ir „Zeek“ atmestų paketų kiekis yra minimalus, kai duomenų srauto intensyvumas yra 100–250 Mbps, o „Suricata“ – iki 750 Mbps. Didinant srauto intensyvumą sistemose matomas staigus atmestų paketų šuolis ir CPĮ išaugimas.
5. Nestandartiniai tinklo sąsajos moduliai padidina įsibrovimo aptikimo sistemų našumą. Naudojant AF_PACKET modulį, „Snort“ paketų apdorojimo kiekis, esant 250 Mbps spartai ir naudojant 128 MB atmintį, padidėjo 7 %, o naudojant 1024 MB atmintį, padidėjo 24 %. Toliau didinant srautą, paketų apdorojimo pokytis mažėja, o nuo 1 Gbps spartos pokytis yra mažesnis nei 5 %. „Suricata“ paketų apdorojimo keikis daugiausiai padidėjo 16 %, esant 1500 Mbps spartai. Kituose matavimo taškuose paketų apdorojimo kiekis padidėjo apie 6 %. Našumo

rezultatai gerėja, nes suteikiami papildomi resursai paketų buferyje, sumažinama procesoriaus apkrova, matomas efektyvesnis „Snort“ ir „Suricata“ paketų apdorojimas.

6. Naudojant PF_RING tinklo modulį, paketų apdorojimo našumas padidėja kelis kartus. „Snort“ paketų apdorojimo kiekis daugiausiai padidėjo 61 %, esant 500 Mbps. „Zeek“ paketų apdorojimo kiekis padidėjo 56 %, esant 250 Mbps spartai. „Suricata“ atmestų paketų kiekis buvo iki 1 %, esant 1000 Mbps spartai.
7. „Snort“ pirminio procesoriaus „http_inspect“ parametro „server_flow_depth“ reikšmė darė minimalią įtaką našumui. Sumažinus parametro vertę nuo 1000 iki 600 apdorojimo baitų, atmestų paketų kiekis sumažėjo 5 %. Parametras „server_flow_depth“ turėtų didesnę įtaką, jeigu perdavimo srautą sudarytų didesnis kiekis paketų, kuriuose serveris atsako į klientų užklausas.
8. Keičiant paketų eilės ilgio parametrą, „Suricata“ atmestų paketų kiekis sumažėjo 42 %, esant 2000 Mbps spartai. Paketų eilės ilgio parametras, didesnis nei 1024, efektyviai sumažina atmestų paketų kiekį, kai eilės dydis yra limituojantis faktorius.
9. Algoritmų matavimo rezultatai parodė, kad aptikimo mechanizmo algoritmas daro didelę įtaką įsibrovimo aptikimo sistemos našumui. Efektyviausias „Snort“ aptikimo mechanizmas yra „ac-bnfa“, o „Suricata“ – „hyperscan“. Esant 750 Mbps spartai „Snort“ atmestų paketų kiekis sumažėjo 12 %, kituose taškuose apie 5 %. „Suricata“ naudodama „hyperscan“ sumažino 20 % atmestų paketų kiekį, esant 1500 Mbps spartai, ir 39 %, esant 2000 Mbps spartai.
10. Įrodyta, kad galima pasiekti dideles paketų apdorojimo spartas, išlaikant minimalų atmestų paketų kiekį. Norint efektyviai pritaikyti įsibrovimo aptikimo sistemas esamame tinkle, būtina papildoma sistemos parametrų konfigūracija ir tinkamas aptikimo algoritmų pasirinkimas.
11. „Snort“ minimalus 0 % atmestų paketų kiekis buvo esant 500 Mbps srautui, naudojant PF_RING ir „ac-bnfa“ algoritmą. „Zeek“ minimalus 0 % atmestų paketų kiekis buvo esant 250 Mbps, naudojant PF_RING.
12. Tyrimai parodė, kad sparčiausia ir tiksliausia yra „Suricata“ sistema, aptikusi daugiausiai atakų esant mišriam srautui, o esant 1500 Mbps srauto spartai išlaikė minimalų, t. y. 0 % atmestų paketų kiekį. Geriausi „Suricata“ rezultatai pasiekiami naudojant PF_RING ir „hyperscan“ konfigūraciją.

INFORMACIJOS ŠALTINIAI

1. *„CIC DoS“ duomenų rinkinio dokumentacija*. (n.d). Prieiga per internetą: https://www.unb.ca/cic/datasets/dos-dataset.html
2. *„CICIDS” duomenų rinkinio dokumentacija*. (n.d). Prieiga per internetą: https://www.unb.ca/cic/datasets/ids-2017.html
3. *„Docker“ programinės įrangos dokumentacija*. (2020). Prieiga per internetą: https://docs.docker.com/engine/install/ubuntu/
4. *„Emerging Threats“ taisyklių rinkinio dokumentacija*. (2020). Prieiga per internetą: https://rules.emergingthreats.net/
5. *„Hyperscan“ paprastosios išraiškos atpažinimo modulio išeities kodo „GitHub“ saugykla*. (2020). Prieiga per internetą: https://github.com/intel/hyperscan
6. *„ISCX“ duomenų rinkinio aprašymas*. (n.d). Prieiga per internetą: https://www.unb.ca/cic/datasets/ids.html
7. *„KDD Cup“ duomenų rinkinio aprašymas*. (1999). Prieiga per internetą: https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
8. *„Snort“ programinės įrangos architektūros schema*. (2014). Prieiga per internetą: https://truica-victor.com/wp-content/uploads/2014/03/snort-arch-.png
9. *„Snort“ programinės įrangos dokumentacija*. (2020). Prieiga per internetą: https://www.snort.org/documents/snort-users-manual
10. *„Snort“ programinės įrangos įskiepio DAQ aprašymas*. (2020). Prieiga per internetą: https://www.snort.org/faq/readme-daq
11. *„Suricata“ programinės įrangos dokumentacija*. (2020). Prieiga per internetą: https://suricata.readthedocs.io/en/suricata-5.0.1/quickstart.html#installation
12. *„Suricata“ programinės įrangos našumo gerinimo dokumentacija*. (2020). Prieiga per internetą: https://suricata.readthedocs.io/en/suricata-5.0.2/performance/tuning-considerations.html
13. *„Suricata“ programinio paketo darbo režimai*. (2020). Prieiga per internetą: https://suricata.readthedocs.io/en/suricata-5.0.0/performance/runmodes.html
14. *„TCPReplay“ papildomo tinklo modulio „netmap“ programinio kodo saugykla*. (2020). Prieiga per internetą: https://github.com/luigirizzo/netmap

15. *„TCPReplay“ programinio paketo dokumentacija*. (2020). Prieiga per internetą: https://tcpreplay.appneta.com/
16. *„Zeek“ programinės įrangos architektūra*. (2020). Prieiga per internetą: https://docs.zeek.org/en/current/_images/architecture.png
17. *„Zeek“ programinės įrangos dokumentacija*. (2020). Prieiga per internetą: https://docs.zeek.org/en/current/intro/index.html
18. Alhomouda, A., Munira, R., Dissoa, J., P., Awan, I., Al-Dhelaan, A. (2011). *Performance Evaluation Study of Intrusion Detection Systems.* Riyadh. Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S1877050911003498
19. Azhagiri, M., Rajesh, A., Karthik, S. (2015). *Intrusion Detection and Prevention System: Technologies and Challenges.* Tamil Nadu. Prieiga per internetą: https://www.researchgate.net/publication/287208734_Intrusion_Detection_and_Prevention_System_Tchnologies_and_Challenges
20. Bul‘ajoul, W., James, A., Pannu, M. (2015). *Improving network intrusion detection system performance through quality of service configuration and parallel technology.* Prieiga per internetą: https://www.sciencedirect.com/science/article/pii/S0022000014001767
21. Clarke, N., Furnell, S., Tjhai, G., Papadaki, M. (2008). *Investigating the problem of IDS false alarms: An experimental study using Snort.* Plymouth. Prieiga per internetą: https://www.researchgate.net/publication/45813553_Investigating_the_problem_of_IDS_false_alarms_An_experimental_study_using_snort
22. *DARPA'98 duomenų rinkinio dokumentacija*. (1998). Prieiga per internetą: https://www.ll.mit.edu/r-d/datasets/1998-darpa-intrusion-detection-evaluation-dataset
23. *DARPA'99 duomenų rinkinio dokumentacija*. (1999). Prieiga per internetą: https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset
24. Eugene, A. (2011). *A comparative analysis of the Snort and Suricata intrusion-detection systems.* Monterey, California. Prieiga per internetą: https://pdfs.semanticscholar.org/981b/ad87d793f685e1736d38dfd056b73385d9e8.pdf
25. Fung, C. (2011). *Collaborative Intrusion Detection Networks and Insider Attacks.* Waterloo. Prieiga per internetą: https://pdfs.semanticscholar.org/08b3/ad17a1b9f9f5968c868bfe30794fb15e7ae9.pdf
26. Gupta, A., Sharma, L. (2020). *Performance Evaluation of Snort and Suricata Intrusion Detection Systems on Ubuntu Server.* Prieiga per internetą:

https://www.researchgate.net/publication/337446926_Performance_Evaluation_of_Snort_and_Suricata_Intrusion_Detection_Systems_on_Ubuntu_Server

27. Ho, C., Lin, Y., R., Lai, Y., Chen, I., Wang, F., Tai, W. (2012). *False Positives and False Negatives from Real Traffic with Intrusion Detection/Prevention Systems.*. Taiwan. Prieiga per internetą: https://www.researchgate.net/publication/260986371_False_Positives_and_False_Negatives_from_Real_Traffic_with_Intrusion_DetectionPrevention_Systems
28. Hubballi, N., Suryanarayanan, V. (2014). *False Alarm Minimization Techniques in Signature-Based IntrusionDetection Systems: A Survey*. Birmingham. Prieiga per internetą: https://www.sciencedirect.com/science/article/abs/pii/S0140366414001480
29. *Įmonių įsilaužimo bandymų statistika ir išvados*. (2019). Prieigą per internetą: https://www.ptsecurity.com/ww-en/analytics/corp-vulnerabilities-2019
30. Maciá-Fernández, Gabriel et. al. (2017). *UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. Computers & Security*. Granada. Prieiga per internetą: https://www.researchgate.net/publication/321187534_UGR'16_A_new_dataset_for_the_evaluation_of_cyclostationarity-based_network_IDSs
31. Mehmood, Y., Habiba, U., Shibli, M., A., Masood. R. (2013). *Intrusion Detection System in Cloud Computing: Challenges and Opportunities.* Islamabad. Prieiga per internetą: https://www.researchgate.net/publication/271555430_Intrusion_Detection_System_in_Cloud_Computing_Challenges_and_opportunities
32. Pandey, S. (2011). *Modern Network Security: Issues and challenges.* Lucknow. Prieiga per internetą: https://www.researchgate.net/publication/267691532_MODERN_NETWORK_SECURITY_ISSUES_AND_CHALLENGES
33. Pareek, S., Dey, R., Gautam, A. (2017). *Different Type Network Security Threats and Solutions, A Review. 2017 m.* Jaipur, Prieiga per internetą: https://www.researchgate.net/publication/316782131_Different_Type_Network_Security_Threats_and_Solutions_A_Review
34. Peng, T., Leckie, C., Ramomohanaro, K. (2007). *Survey of Network-Based Defense Mechanisms Countering the DoS and DDoS Problems.* Melbourne. Prieiga per internetą: https://www.cs.ucf.edu/~dcm/Teaching/COT4810Spring2011/Literature/DenialOfServiceAttacks.pdf

35. *PF_RING tinklo modulio dokumentacija*. (2020). Prieiga per internetą: https://www.ntop.org/guides/pf_ring/
36. Pihelgas, M. (2012). *A Comperative Analysis of OpenSource Intrusion Detection Systems.* Tallinn. Prieiga per internetą: http://mauno.pihelgas.eu/files/Mauno_Pihelgas-A_Comparative_Analysis_of_OpenSource_Intrusion_Detection_Systems.pdf
37. Pir, R., M. (2014). *Intrusion Detection Techniques and Open Source Intrusion Detection (IDS) Tools.* Sylhet. Prieiga per internetą: https://www.academia.edu/8673348/Intrusion_Detection_Techniques_and_Open_Source_Intrusion_Detection_IDS_Tools
38. Samrin, R., Vasumathi, D. (2017). *Review on Anomaly based Network Intrusion Detection System.* Hyderabad Priega per internetą: http://utopia.duth.gr/~kdemertz/papers/citation/08284655.compress.pdf
39. Scarfone, K., Mell, P. (2007). *Guide to Intrusion Detection and Prevention Systems (IDPS), NIST Special Publication on Computer security.* Gaithersburg. Prieiga per internetą: https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-94.pdf
40. Sharafaldin, I., Lashkari, A, H., Ghorbani, A. A. (2017). *Toward Generating a New Intrusion Detection Dataset and Intrusion Traffic Characterization*. New Brunswick. Prieiga per internetą: https://www.scitepress.org/Papers/2018/66398/66398.pdf
41. Sturges, S. (2009). *Using Perfmon and Performance Profiling to Tune Snort Preprocessors and Rules.* Columbia. Prieiga per internetą: https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/030/original/WhitePaper_Snort_PerformanceTuning_2009.pdf
42. White, J., Fitsimmons, T., Matthews, J. (2013). *Quantitative Analysis of Intrusion Detection Systems: Snort and Suricata*. Prieiga per internetą: https://www.researchgate.net/publication/236146583_Quantitative_Analysis_of_Intrusion_Detection_Systems_Snort_and_Suricata

PRIEDAI

A priedas. Atakų atpažinimo tiriamų failų rezultatų lentelės

4.2 lentelė. „Wednesday-WorkingHours“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	DoS „Slowloris“	Taip	Neteisingas RST paketas, galima DoS ataka (angl. *Gateway invalid RST denial of service attempt*)
„Suricata“		Ne	-
„Zeek“		Taip	-
„Snort“	DoS „Slowhttptest“	Taip	Naudojamas „slowhttptest“ atakos įrankis (angl. S*lowhttptest DoS tool*)
„Suricata“		Ne	-
„Zeek“		Taip	-
„Snort“	DoS „Hulk“	Ne	-
„Suricata“		Ne	-
„Zeek“		Taip	-
„Snort“	DoS „GoldenEye“	Ne	-
„Suricata“		Taip	Galima „GoldenEye“ DoS ataka į vidinę sistemą (angl. *Inbound GoldenEye DoS attack*)
„Zeek“		Taip	-
„Snort“	„Heartbleed“	Ne	-
„Suricata“		Taip	Galimas „HeartBleed“ pažeidžiamumo išnaudojimas (angl. *HeartBleed Large HeartBeat Response*)
„Zeek“		Taip	-

4.3 lentelė. „Thursday-WorkingHours“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	Saityno ataka jėgos metodas	Ne	-
„Suricata“		Ne	-
„Zeek“		Ne	-
„Snort“	Saityno ataka „XSS“	Ne	-
„Suricata“		Ne	-
„Zeek“		Ne	-
„Snort“	Saityno ataka „SQL“ injekcija	Taip	Galima SQL užklausos injekcija (angl. *Possible SQL injection attempt*)
„Suricata“		Taip	Galima SQL užklausos injekcija (angl. *Possible SQL injection attempt*)
„Zeek“		Ne	-
„Snort“	Įsibrovimo ataka „Dropbox” failo parsisiuntimas	Ne	-
„Suricata“		Taip	Galimai kenkėjiškas „Windows“ sistemos failas (angl. *Windows Microsoft Windows DOS*)
„Zeek“		Ne	-
„Snort“	Įsibrovimo ataka „Cool disk“	Ne	-
„Suricata“		Ne	-
„Zeek“		Ne	-
„Snort“		Ne	-

„Suricata“	Įsibrovimo ataka „Dropbox“ failo parsisiuntimas	Taip	Galimai kenkėjiškas „Windows“ sistemos failas (angl. *Windows Microsoft Windows DOS*)
„Zeek“		Ne	-

4.4 lentelė. „Friday-WorkingHours“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	„Botnet ARES“	Taip	Kenkėjiškos programos išeinantys paketai (angl. *WinTrojanCannibal RAT outbound*)
„Suricata“		Taip	„Python“ programos užklausos į WEB serverį (angl. *UserAgent pythonrequests Inbound to Webserver*)
„Zeek“		Ne	-
„Snort“	Prievadų skenavimas	Ne	-
„Suricata“		Taip	Galimas SSH protokolo skenavimas (angl. *Potential SSH scan outbound)*
„Zeek“		Taip	-
„Snort“	DoS „LOIT“	Ne	-
„Suricata“		Taip	Galimai naudojamas DoS atakos įrankis (angl. *Likely AnonMafiaIC DoS tool*)
„Zeek“		Ne	-

4.5 lentelė. „testbed13-jun“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	Infiltravimas į sistemą iš vidinių adresų	Ne	-
„Suricata“		Taip	Įtartini MSSQL duomenų bazės prisijungimai (angl. *Suspicious inbound to MSSQL*)
„Zeek“		Taip	-

4.6 lentelė. „testbed14-jun“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	HTTP DoS	Ne	-
„Suricata“		Taip	Galimai naudojamo jėgos metodo užklausos atsakas (angl. *Potential BruteForce attempt response*)
„Zeek“		Taip	-

4.7 lentelė. „testbed15-jun“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	DoS	Ne	-
„Suricata“		Ne	-
„Zeek“		Taip	-

4.8 lentelė. „testbed17-jun“ duomenų failo tyrimo rezultatai

ĮAS	Atakos pavadinimas	Ar atpažinimas buvo sėkmingas?	Įspėjamasis pranešimas
„Snort“	SSH protokolo jėgos metodo ataka	Taip	SSH protokolo versijos nesutapimas (angl. *sppssh protocol mismatch*)
„Suricata“		Taip	Galimas SSH protokolo skenavimas (angl. *Potential SSH Scan*)
„Zeek“		Ne	-

B priedas. Pranešimų tipų rezultatų analizės lentelės.

4.9 lentelė. „CIDIDS“ duomenų rinkinio pranešimų tipų pasiskirstymas

Duomenų failas	ĮAS	Pranešimų skaičius	TT pranešimų skaičius	Perspėjimo pranešimai	KT pranešimų skaičius
„Tuesday-WorkingHours“	„Snort“	159502	12	39117	120313
	„Suricata“	1830010	3071	1822146	4793
„Wednesday-WorkingHours“	„Snort“	355888	1754	204191	149943
	„Suricata“	2386391	6	2373220	13165
„Thursday-WorkingHours“	„Snort“	138541	11	28963	109576
	„Suricata“	1336989	23744	1306659	6586
„Friday-WorkingHours“	„Snort“	164418	56	48978	115384
	„Suricata“	1404174	851	1398479	4844

4.10 lentelė. „ISCX“ duomenų rinkinio pranešimų tipų pasiskirstymas

Duomenų failas	ĮAS	Pranešimų skaičius	TT pranešimų skaičius	Perspėjimo pranešimai	KT pranešimų skaičius
„testbed13-jun“	„Snort“	33748	0	3933	29815
	„Suricata“	6643	149	1488	5006
„testbed14-jun“	„Snort“	50968	0	18898	32070
	„Suricata“	12057	3	30	12024
„testbed15-jun“	„Snort“	153391	0	92528	60963
	„Suricata“	82348	0	61911	20437
„testbed17-jun“	„Snort“	94914	6221	18484	70209
	„Suricata“	18089	1081	8914	8094

C priedas. Dalyvavimo jaunųjų mokslininkų konferencijoje „Mokslas – Lietuvos ateitis. Elektronika ir elektrotechnika" sertifikatas

VILNIAUS GEDIMINO TECHNIKOS UNIVERSITETAS
ELEKTRONIKOS FAKULTETAS

XXIII-toji Lietuvos jaunųjų mokslininkų konferencija
„MOKSLAS – LIETUVOS ATEITIS"
ELEKTRONIKA IR ELEKTROTECHNIKA

PAŽYMA

Lukas Stankovičius

Elektronikos (T170), Mikroelektronikos (T171) ir Telekomunikacijų inžinerijos (T180) jungtinėje sekcijoje perskaitė pranešimą

ANALYSIS OF OPEN SOURCE NETWORK SECURITY SOLUTIONS

Mokslo komiteto pirmininkas prof. habil. dr. Romanas Martavičius

Sekcijos pirmininkas doc. dr. Nerijus Paulauskas

2020 m. balandžio 30 d.
Vilnius