УНИВЕРЗИТЕТ У БЕОГРАДУ

ЕЛЕКТРОТЕХНИЧКИ ФАКУЛТЕТ

Ђорђе Д. Јовановић

Рано откривање уређаја заражених ботнет малвером коришћењем метода детекције аномалија мрежних токова

докторска дисертација

Београд, 2024.

UNIVERSITY OF BELGRADE

SCHOOL OF ELECTRIC ENGINEERING

Đorđe D. Jovanović

Early discovery of the devices infected with botnet malware using network flow anomaly detection

Doctoral Dissertation

Belgrade, 2024

Садржај

Листа слика

Листа табела

Подаци о ментору и члановима Комисије

Ментор
проф. др Павле Вулетић, ванредни професор,
Универзитет у Београду, Електротехнички факултет

Чланови комисије

проф. др Жарко Станисављевић, ванредни професор
Универзитет у Београду, Електротехнички факултет

проф. др Горан Квашчев, редовни професор
Универзитет у Београду, Електротехнички факултет

др Татјана Давидовић, научни саветник
Математички институт САНУ

проф. др Дражен Драшковић, ванредни професор
Универзитет у Београду, Електротехнички факултет

проф. др Марија Пунт, ванредни професор
Универзитет у Београду, Електротехнички факултет

Датум одбране:_____

Рано откривање уређаја заражених ботнет малвером коришћењем метода детекције аномалија мрежних токова

Сажетак

Ботнети представљају скуп уређаја заражених малициозним софтвером, којима управља злонамерни администратор (ботмастер), чији је циљ извођење различитих напада на рачунарску инфраструктуру попут крађе података или дистрибуираних напада ускраћивања сервиса (енг. Distributed Denial of Service, скр. *DDoS*). У последње време, ботнети представљају све већу опасност, нарочито од тренутка када су мета постали уређаји типа интернет ствари (енг. *Internet of Things,* скр. *IoT*) чији је број значајно порастао, а који се често скромно одржавају. Већина досадашњих приступа у детекцији напада ботнета се фокусира на детекцију самих напада (најчешће *DDoS*), након што се напади десе. С обзиром на разорну моћ новијих ботнета, али и чињеницу да уређаји могу да буду део ботнета данима и месецима пре него што се употребе за напад, од великог је значаја да се ботнет открије што пре, како би се напад спречио, а ботнет неутралисао.

Овај рад нуди алтернативни приступ детекцији ботнета у односу на досадашње приступе: рану детекцију заражених уређаја посматрањем мрежних токова командне и контролне комуникације ботнета као временске серије и екстракцијом унутартоковских (енг. intraflow) статистичких одлика из њих. Циљ овог метода екстракције одлика јесте уштеда рачунарских ресурса потребних за детекцију, уз очувану високу прецизност детекције. Екстракција ових одлика и примена техника машинског учења су остварили циљ детекције ботнет комуникације пре него што се напад догоди. Као први корак, динамичком анализом понашања и статичком анализом кода малвера, истражене су карактеристике мрежног понашања примерака IoT ботнета који су прикупљани током четири године (2019-2023). Анализом прикупљених примерака, осмишљен је механизам за екстракцију карактеристичних временских низова и одлика из њих заснован на концепту софтверски дефинисаних мрежа. Испитивани су различити модели машинског учења, тестирани на једном јавном скупу података Чешког техничког универзитета (енг. Czech Technical University, скр. *CTU*), као и на прикупљеним ботнет примерцима. Остварени резултати детекције су били једнаки као у другим научним радовима из области, који су детекцију вршили из снимака комплетног саобраћаја на линку, уз режијске трошкове (потребну процесорску снагу, простор на дисковима итд.) мање до два реда величине.

У каснијој фази истраживања, прикупљени су додатни примерци ботнет апликација и проширен је скуп екстрахованих одлика. Потом је тестиран систем машинског учења базиран на екстремном градијентном ојачавању (енг. *Extreme Gradient Boosting*). Испитиване су

различите технике одабира одлика из временских низова, оптимизације хиперпараметара, модела, као и техника генерисања вештачких узорака. На крају, за радне процесе са најбољим резултатима, различитим груписањем скупа обучавања и тестног скупа, испитивана је могућност детекције узорака новијег датума старијим узорцима и дата је анализа резултата. Ова анализа је дала далеко боље резултате у детекцији ботнета уз вредност F1 мере од 0.9041 у случају радних процеса који не користе корак вештачког генерисања узорака, и вредност F1 мере од 0.9984 у случају радних процеса који користе корак вештачког генерисања узорака, чиме је значајно премашила прецизност детекције других радова и методологија у области чиме су потврђене полазне хипотезе овог рада да је коришћењем унутартоковских одлика могуће реализовати овакав систем детекције.

Кључне речи: информациона безбедност, детекција ботнета, машинско учење, софтверски дефинисане мреже

Научна област: Електротехника и рачунарство
Ужа научна област: Рачунарска техника и информатика

УДК број: 621.3

Early discovery of the devices infected with botnet malware using network flow anomaly detection

Abstract

Botnets represent a collection of devices infected with malicious software, controlled by a malicious administrator (botmaster), aiming to execute various attacks on computer infrastructure, such as data theft or DDoS (Distributed Denial of Service) attacks. Recently, botnets have become an increasing threat, especially since Internet of Things (IoT) devices, often lacking security measures of their own, and whose numbers have significantly grown in recent years, have become targets. Most existing approaches to botnet attack detection focus on identifying attacks (usually DDoS) post factum, i.e. after they occur. Given the destructive power of newer botnets and that devices can be part of a botnet for days or months before being used in an attack, it is crucial to detect the botnet as early as possible to prevent the attack and neutralize the botnet.

This thesis offers an alternative approach to botnet detection in comparison to previous methods: early detection of infected devices by observing botnet command and control network flows as time series and extracting intra-flow statistical features from them. The goal of this feature extraction method is to save computational resources required for detection, while at the same time maintaining high detection accuracy. The extraction of these features and the application of machine learning techniques achieved the goal of detecting botnet communication before an attack occurs.

As a first step, dynamic behavior analysis and static malware code analysis were conducted to investigate the network behavior characteristics of IoT botnet samples collected over four years (2019-2023). Based on the analysis of the collected samples, a mechanism was devised to extract characteristic time series and features from them based on the concept of software-defined networking. Various machine-learning models were tested on a public dataset from the Czech Technical University (CTU), and on collected botnet samples. The detection results were comparable to other relevant scientific works in the field that performed detection from complete traffic snapshots on the link, with overhead costs (required processing power, disk space, etc.) reduced by up to two orders of magnitude.

In the later stages of the research, additional botnet application samples were collected, and the set of extracted features was expanded. The machine learning system based on Extreme Gradient Boosting was then tested. Various techniques for feature selection from time series, hyper-parameter optimization, model selection, and artificial sample generation were examined. Finally, for pipelines with the best results, the possibility of detecting newer samples with older ones was investigated by grouping the training and test sets differently, and the results were analyzed. This analysis yielded far better botnet detection results, with an F1 score of 0.9041 for pipelines that did not use artificial

sample generation and an F1 score of 0.9984 for pipelines that did use artificial sample generation, significantly surpassing the detection accuracy of other studies and methodologies in the field. This confirmed the initial hypothesis of this paper that using intra-flow features can enable the realization of such a detection system.

Keywords: information security, botnet detection, machine learning, software-defined networks.

Scientific field: Electrical engineering and Computer Science.
Scientific subfield: Computer Science and Information Technology.

UDC number: 621.3

1 Увод

Ботнет представља мрежу рачунара (тзв. ботова) најчешће заражену злонамерним софтвером која је под контролом нападача - хакера, који се зове ботмастер. Термин бот, представља скраћеницу од речи робот, јер представља машину која извршава команде које задаје нападач. Ботмастер користи ове уређаје за нелегалне радње попут *DDoS* напада, ширења различитих врста вируса, крађу личних информација и кредендијала, рударење криптовалута, и многе друге [1].

Ботнети су више пута демонстрирали своју разорну моћ пре свега *DDoS* нападима. Са појавом Mirai ботнета, јавност је више обратила пажњу на овај тип малвера и инфраструктуре за спровођење напада. Разлог је тај што су ботнети базирани на овом малверу извршили нападе изузетно великог обима. Његова мрежа ботова је у једном тренутку бројала чак и 600.000 заражених уређаја, а по први пут су уређаји који су чинили ботнет били камере, мали кућни рутери, уређаји за снимање видеа и други слични уређаји који нису до тада били мета заразе. Генерисани су *DDoS* напади капацитета од више стотина гигабита у секунди који су онемогућавали нормално функционисање услуга више стотина хиљада корисника. Напади су циљали различите мете попут *Krebs on Security*[1] веб-сајта, хостинг провајдера *OVH*[2], *DNS* систем (енг. *Domain Name System,* скр. *DNS*) сервере *Din* и *Lonestar Cell*, а најпознатији је напад на *Deutsche Telekom*. Иако се Mirai малвер појавио 2016. године, ботнети базирани на њему су и даље актуелни. Познат је каснији случај компромитовања мрежне опреме, понајвише интернет протокола (енг. *Internet Protocol,* скр. *IP*) камера и штампача, где је ботнет који је коришћен био базиран на Mirai малверу [2]. У 2020. години извршен је *DDoS* напад обима 2.3 терабита у секунди на *AWS* сајт компаније Amazon [3]. У 2022. години, забележен је пораст политички мотивисаних DDoS напада на владине и образовне институције [4]. Извештај из 2023. године указује да је најзаступљенији тип малвера у DNS саобраћају управо ботнет тип малвера [5], што указује на константну актуелност теме детекције ботнета и њиховог сузбијања, због штете коју константно наносе. Иако већ више од деценије постоје многи истраживачки покушаји да се ботнети ефикасно

1 https://krebsonsecurity.com/
2 https://www.ovhcloud.com/en-gb/

детектују и сузбију, ефикасно и универзално решење још увек не постоји, а проналазак нових метода детекције је увек доводио до модификације начина рада малвера како би се избегла детекција.

Праћењем јавних база података детектованих малвера, може да се уочи да се практично свакодневно појави неколико нових малвера који формирају ботнете. Већина хакера користи кôд постојећих малвера, правећи минималне измене, попут измене статичких *IP* адреса или текстова у порукама, али и неке врло софистициране промене које ће бити приказане у тексту, све у сврху избегавања детекције од стране сигурносних система. На интернету су се појављивали и програми који омогућавају креирање сопственог ботнета, за чије коришћење није потребно велико познавање програмирања, што указује на чињеницу да значајна измена познатих вируса није високофреквентна појава. Један од скоријих ботнета, под називом *Dark Nexus*, је испрограмиран на основу кода постојећих *Mirai* и *Qbot* ботнета [6]. У већини случајева, за нове примерке ботнета, није лако одредити којој фамилији ботнета припадају, јер представљају неку комбинацију особина и кода различитих врста малвера.

Као што је речено, мрежна инфраструктура ботнета се састоји од заражених рачунара, који се зову ботови, и ботмастера, који њима управља. Управљање се врши путем командног и контролног канала (енг. *command and control,* скр. *C&C*). Једна од употреба контролног канала је ажурирање листе активних ботова. Како би се знало да ли је бот доступан, ботмастер и бот периодично комуницирају ради одржавања конекције. Ова комуникација се зове *C&C heartbeat*. Друга намена за контролни канал је да се шаљу различите команде ботовима, или пак да се иницира *DDoS* напад. Истражујући динамичке обрасце у комуникацији између ботмастера како би се осмислио систем који ефикасно открива такву комуникацију, могуће је благовремено зауставити деловање ботова пре него што дође до напада. Овај контролни канал је уједно и слаба тачка ботнета, будући да се прекидом комуникације путем њега ботови у мрежи изолују од ботмастер сервера који издаје команде за злонамерне активности. Како би заштитили овај канал, неки ботнети додатно користе напредне технике сакривања *IP* адреса ботмастера, попут *DNS fluxing* и алгоритма генерисања домена (енг. *Domain Generation Algorithms*, скр. *DGA*) [7]. *DNS fluxing* је техника којом се *IP* адреса скрива тако што се циклично смењују адресе иза неког имена домена. Алгоритми генерисања домена с друге стране генеришу више различитих интернет домена, који указују на исту *IP* адресу. Током прелиминарног истраживања [8], откривено је да већина савремених ботнет узорака чија су мета уређаји који се користе у оквиру интернета

ствари (енг. *Internet of Things,* скр. *IoT*), који су били анализирани у овој дисертацији, не користе напредније технике сакривања *IP* адреса. Такође је откривено понављање врло сличних образаца понашања контролног канала за различите фамилије ботнета, што сведочи о коришћењу истог или сличног кода за различите вирусне апликације. Периодична комуникација се разликовала у неким аспектима попут стрингова који су коришћени у пакетима, број размењених пакета, *IP* флегова, с друге стране, друге карактеристике су имале сличне и стабилне вредности, попут ниског и константног протока, симетричности тока пакета, периодичности, што се може користити за детекцију ботнета. Стога једна од хипотеза овог рада, а која је потврђена истраживањем је била да постоје обрасци понашања ботнет малвера који су заједнички за више различитих класа малвера, а који могу да се искористе за поуздану детекцију ботнета у дужем временском периоду.

Детекција ботнета се базира углавном на анализи статистичких параметара мрежног саобраћаја. У овом типу анализе постоји баланс између детаљности обраде података саобраћаја с једне стране, и могућности за складиштење и процесирања мрежног саобраћаја с друге. Методе детекције које се базирају на анализи мрежних токова, који за прикупљање података о мрежном току користе *IPFIX* или сличне механизме [9], памте само генералне информације о сваком току, попут дужине трајања тока, броја пакета и укупног броја пренетих бајтова. Стога, овај тип методе има неколико мана. Као прво, пун обим информација о мрежном току је забележен након што је ток завршен, имплицирајући да све информације о нападу су забележене *post factum*, тј. након што се напад одиграо. Мана овог приступа се огледа и у чињеници да комуникација између ботмастера и бота може да траје сатима, или данима, пре него што се иницира напад, што значи да ће, у зависности од начина бележења токова, овакав ток или бити забележен као неколико дисјунктних мањих токова, што ремети бележење и анализу статистичких одлика тока, или ће ток бити забележен у целости, а у том случају ће напад већ бити извршен. Као друго, општа статистика тока не описује комуникацију довољно детаљно, што може спречити детекцију малициозних активности. Алтернатива овом методу, с друге стране, лежи у бележењу садржаја сваког мрежног пакета, дајући потпун увид у стање у рачунарској мрежи. Међутим, овај приступ подразумева складиштење велике количине података, коју после треба адекватно обрадити. Примера ради, један дан забележеног саобраћаја на попуњеном линку од 100Gbps (енг. *Gigabits per Second*, скр. *Gbps*) генерише 1 петабајт података. Даље, будући да је садржај већине пакета послатих преко мреже енкриптован, обрада таквих пакета не би дала више информација од саме величине пакета. Стога је у оквиру рада постављена друга хипотеза да

је могуће оптимизовати капацитете за складиштење података о малверима и процесорско оптерећење, без губитка тачности детекције ботнет малвера. Ово је доказано кроз предложен оригинални *PI-BODE* (скраћено од *Programmable Intraflow-based IoT Botnet Detection*) систем, који се налази између два поменута приступа. Овај систем, базиран на софтверски дефинисаним мрежама, прикупља статистике мрежног тока, обогаћене статистичким параметрима садржаја тока, док је ток активан (тзв. статистика унутар мрежног тока – *intraflow* статистика). Фокус овог система је детекција IoT ботнета, који су у последњих неколико година, од појаве *Mirai* ботнета најразорнији ботнети. Детекција ботнета се базира на детекцији образаца у комуникацији путем контролног канала анализом статистичких особина садржаја тока. Овај приступ омогућава брзу детекцију ботнет контролног канала, у реалном времену, а не након завршетка мрежног тока или снимка пакета, што је још једна од полазних хипотеза ове тезе.

Истраживање се састојало из три фазе: прикупљање узорака реалног IoT ботнет малвера и анализа карактеристичног мрежног понашања у периоду од 2019. до 2020. године (прва фаза), дизајн система за детекцију ботнета на бази машинског учења (друга фаза), прикупљање нових узорака малвера и тестирање успешности различитих радних процеса машинског учења (трећа фаза). Узорци су прикупљани током четири године, у периоду од 2019. до 2023. године. Овакав временски период, омогућио је анализу сличности образаца мрежног понашања ботнет малвера. Стога је у оквиру рада постављена трећа хипотеза да је могуће детектовати нове варијанте ботнет малвера у систему обученом над старим примерцима истог типа ботнет малвера. Ове фазе истраживања ће бити детаљније описане у уводу поглавља 3. Будући да малициозни саобраћај чини мањину интернет саобраћаја, у свим скуповима података постоји небалансираност између малициозних и бенигних класа у скупу података. У раду је стога испитиван и утицај метода за вештачко генерисање узорака мањинске класе на детекцију малициозних примерака интернет саобраћаја. Уз примену међутоковских (енг. interflow) статистика и коришћење метода вештачког генерисања узорака мањинске класе, F1 мера је у случају најуспешнијег радног процеса имала вредност од чак 0.0084. Такође, извршена је и анализа могућности детекције у реалном времену. Анализа је показала да је детекција ботнет саобраћаја могућа оквирно око 5 минута након успостављања командног и контролног канала.

У другом поглављу су дате опште информације о ботнет класи малвера. Главна тема овог поглавља је кратак историјат ботнет малвера, њихов животни циклус, опис функционисања

контролног канала и до сада уочени начини избегавања детекције. У трећем поглављу је описана прва фаза истраживања, а то су: методологија прикупљања узорака IoT ботнет малвера и анализа карактеристичних образаца динамичког понашања прикупљених узорака. У четвртом поглављу је описана друга фаза истраживања, а то је пројектовање и реализација оригиналног PI-BODE система за детекцију IoT малвера базираног на софтверски дефинисаним мрежама и статистичкој анализи унутар мрежних токова [10]. Такође, рад овог система је верификован анализом детекције IoT ботнета, те је дат опис механизама машинског учења за детекцију ботнета. У петом поглављу је описана трећа фаза истраживања, која представља проширење друге фазе истраживања новим примерцима малвера из периода 2022-2023. Главни фокус овог поглавља је на проширеној методологији машинског учења, са посебним акцентом на третирање небалансираних скупова података. Такође, у петом поглављу су дати и анализа мрежног понашања новоприкупљених узорака ботнета, анализа најзначајнијих одлика које су утицале на класификацију, анализа могућности детекције у реалном времену и анализа стратегија детекције нових варијанти IoT ботнет малвера. У шестом поглављу биће дат закључак рада.

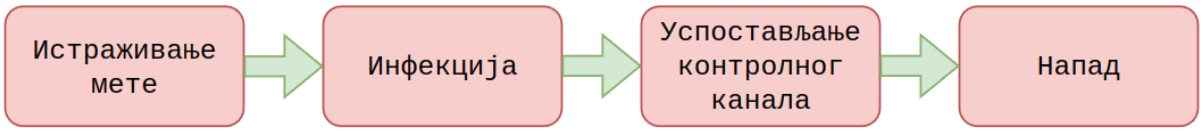
2 Преглед развоја ботнет малвера

Бот малвер представља програм који, примајући поруке од других програма путем рачунарске мреже, извршава команде које су му задате. Прву бот апликацију је осмислио Јарко Оикаринен са Универзитета у Оулуу, у Финској, 1988. године [11]. Наредне године, Грег Линдал, који је радио као оператор *Internet Relay Chat* (скр. *IRC*) сервера, је програмирао бота који се звао *GM*, који би путем *IRC* чета играо игру са учесницима. По узору на ове ботове настајао је низ *IRC* ботова, који нису коришћени у малициозне сврхе. Један од тих ботова, под именом *EggDrop*, је направљен 1993, и коришћен је да олакша управљање *IRC* каналом.

Априла 1998. је направљен први малициозни бот, *GTBot*, који је имао на стотине варијанти, које су имале у себи *IRC* клијентску апликацију. Разлог зашто је IRC коришћен у ове сврхе јесте тај што омогућава лако одржавање ботнета, тако што би се заражени рачунари прикључили *IRC* каналу за ћаскање, преко којег би им ботмастер слао команде. Један од највећих *IRC* ботнета у историји је био *TDL4*, који је бројао 4,500,000 заражених уређаја [12]. Временом се прешло на коришћење P2P (енг. *Peer-to-Peer,* скр. *P2P*) протокола, попут *HTTP* (енг. *Hypertext Transfer Protocol*, скр. *HTTP*), *ICMP* (енг. *Internet Control Message Protocol*, скр. *ICMP*), *TCP* (енг. *Transmission Control Protocol*, скр. *TCP*), *UDP* (енг. *User Datagram Protocol*, скр. *UDP*), за комуникацију између ботмастера и бота. Неки од познатијих ботнета овог типа су: *Storm*, *Conficker*, *Zeus*, *Mirai*, *Gafgyt,* који су служили за прикупљање личних података или DDoS напад. *Zeus* је у једном тренутку постао толико популаран, да су се појавили програми који су били у стању да направе сопствену верзију ботнета, без познавања програмирања [13]. Пораст у ботнетима који прислушкују банковне трансакције, попут *Bugot*, *KING*, *Gozi*, *Dyre* је забележен 2014. и 2015. године, док се 2016. године појавио ботнет који напада IoT уређаје, под називом *Mirai*, а који је генерисао врло разорне DDoS нападе. У табели 18, у Додатку А, дат је преглед података о најпознатијим ботнетима: година када су се појавили, име, број ботова, и протокол који користе [12]. О некима од њих чији су механизми рада посебно анализирани у овој дисертацији ће бити више речи у наставку овог поглавља.

2.1 Животни циклус ботнета

Најчешћи животни циклус ботнета се може поделити у четири фазе: истраживање мете, инфекција, успостављање контролног канала, и напад (Слика 1). У првој фази, нападач се одлучује за врсту уређаја која ће бити експлоатисана, и истражује потенцијалне начине на које може да се инфицира. Након тога, користи слабости мете, и инфицира је ботнет апликацијом. Потом се успоставља контролни канал са ботмастером, преко којег ботмастер и бот размењују поруке, и одржавају везу. Финалну фазу представља напад, током којег бот извршава намену за коју је створен. Током одржавања комуникације путем контролног канала, може доћи и до измене верзије ботнет апликације [11]. Такође, неке од ових фаза не морају нужно да се извршавају секвенцијално, већ могу у паралели да ботови истовремено и одржавају комуникацију са ботмастером и раде на проналажењу нових потенцијалних ботова. У наставку овог поглавља се свака од ових фаза детаљније описује.



Слика 1: Најчешћи животни циклус ботнета

2.1.1 Истраживање мете

У првој фази, хакер се упознаје са могућим нападима на мету коју одабере. Постоји више начина помоћу којих се мета може преварити да инсталира малициозну апликацију. Напади пецања (енг. *phishing attacks*) су напади током којих се жртви шаље мејл који садржи линк за преузимање вирусне апликације. Савршенија варијанта овог напада је тзв. *Spear phishing,* којим се жртви шаље циљани мејл у којем нападач користи неку информацију о жртви која би повећала вероватноћу преузимања малвера (нпр. зна се да жртва има рачун у некој банци, па се шаље мејл у име те банке и сл.). Процењује се да је ово најчешћи облик инфекције, достижући чак 90% укупног броја инфекција [14]. Битно је нагласити да је садржај тог мејла такав да или личи на поруку познатог сајта, или пак имитира поруку од људи познатих жртви напада. *Watering hole* напади [15] се изводе компромитовањем веб-сајтова које мета често

посећује, тако што се убаци вирусни код у њих. Овај код често експлоатише познате слабе тачке интернет претраживача жртве напада, преко чега се може добити приступ ресурсима на рачунару жртве, као и покренути спуштање вирусне апликације и њено инсталирање. Познат је случај из 2015, када је сајт медијске групе Форбс инфилтриран, и злонамерна апликација се ширила путем кода убаченог у *Flash Player* додатак. Могућ је и напад путем социјалног инжењеринга, где посредством комуникације са другим људима, мета дође у посед вируса. Још један метод за проналажење жртве јесте насумично скенирање, где се портови насумично одабране IP адресе скенирају, како би се установили отворени портови, а потом и инфицирао уређај. Овај приступ је у раним верзијама имао *Mirai* ботнет, док је касније уведена црна листа IP адреса владиних институција [16], како би се спречило брзо проналажење и гашење ботнета. Један од најопаснијих начина инфекције рачунара мете представља тзв. напад нултог дана (енг. *zero-day attack*), где хакер користи до тада непознате мане уређаја или софтвера који мета користи за инфекцију истих. Након фазе истраживања, следи фаза инфекције мете, током које се вирусна апликација инсталира на уређај.

2.1.2 Инфекција

У овој фази, ботмастер истражује слабе тачке мете и инсталира злонамерну апликацију на уређај мете. Начини на који се уређај инфицира не морају да буду напредни и да подразумевају познавање сложених механизама рада оперативних система или апликација, те да експлоатишу неке откривене слабости. Ово је видљиво на примеру једног од најразорнијих - *Mirai* ботнета. Наиме, ботови који су део овог ботнета прво скенирају насумичне IP адресе по портовима 23 и 2323, који су резервисани за Телнет протокол, заобилазећи адресе из листе која је дефинисана у коду апликације (адресе организација које се баве заштитом информационе безбедности или полиција, *Federal Bureau of Investigation* и слично). Ботови који врше ову специфичну улогу се такође називају и скенерским машинама. Уколико су ови портови отворени, *Mirai* ће покушати да се пријави на те уређаје, користећи предефинисани скуп креденцијала за IoT уређаје. Ови креденцијали представљају системска корисничка имена и лозинке за те уређаје, које корисници нису променили (тзв. подразумеване лозинке). Једном када се корисник успешно пријави на уређај, бот ће послати информације о уређају серверу, контактирати сервер, инсталирати вирусну апликацију и наставити даље скенирање. Овај метод инфекције уређаја је пре *Mirai* ботнета имао и *BASHLITE* ботнет, познатији и под именима *LizardStresser*, *Torlus* или *Gafgyt*, који је на идентичан начин покушавао да добије приступ Линукс уређајима, тиме што је имао листу 6 најчешћих коришћених корисничких имена и 14 лозинки. За разлику од тога, *Mirai* има листу

од 62 најчешћих парова корисничких имена и лозинки, циља IoT уређаје, и има метод за скенирање уређаја на интернету, како би се брже повећао број ботова. Након ове фазе, бот је спреман да прима команде од ботмастера, како би извршио своју малициозну намену.

2.1.3 Успостављање контролног канала

У трећој фази, заражени уређај успоставља комуникацију са ботмастером, путем неког од мрежних протокола. Први ботнети су користили IRC протокол. Наиме, у вирусну апликацију су убачене информације о IRC серверу и каналу, преко којег прима команде од ботмастера. Овај приступ има следеће мане:

- комуникација између ботова и ботмастера често није енкриптована, те ју је лакше детектовати;
- уписивање локације IRC канала у код малвера, што статичком анализом кода лако може да буде откривено и канал филтриран односно угашен;
- централна тачка, што значи да уколико сервер престане са радом, цео ботнет бива неутралисан.

Потом су се развили ботнети базирани на апликацијама за комуникацију, попут *AOL*, *MSN*, *ICQ* и *Twitter*[17]. Разлика између комуникације преко ових апликација и IRC канала је та што у случају ових апликација, потребно је да сваки бот направи налог за исту. Овај приступ има следеће мане:

- ботови морају да стално буду прикључени на интернет;
- ботмастер у сваком тренутку има ограничену количину налога, а тим самим, ограничену количину ботова на располагању;
- уписивање локације IRC канала у код малвера, што статичком анализом кода лако може да буде откривено и канал филтриран односно угашен;
- централна тачка, што значи да уколико сервер престане са радом, цео ботнет бива неутралисан.

Hypertext Transfer Protocol (скр. *HTTP*) протокол се такође може користити за управљање ботнетима. У овом случају, вирусна апликација усмерава заражени уређај да периодично посећује веб сервер, који контролише ботмастер. Једном када ботмастер одлучи да пошаље команду, или пак постави нову верзију злонамерне апликације, он промени садржај странице

на коју указује *URL* (енг. *Uniform Resource Locator*, скр. *URL*), тако да када ботови наредни пут буду посетили веб сервер, спустиће са њега команду, и отпочеће њено извршавање. Мана овог приступа је то да ботнет има један сервер који, уколико престане са радом, цео ботнет бива неутралисан.

Сви ови протоколи имају клијент-сервер архитектуру у којој ботови посећују неко заједничко место на интернету, са којег сви у исто време примају команде. Најновији ботнети, попут *Mirai* ботнета, користе и *P2P* архитектуру, у којој ботмастер комуницира са ботовима индивидуално, путем *TCP*, *ICMP*, или *UDP* протокола [11]. Овај приступ има следеће мане:

- недостатак енкрипције и аутентикације
- нескалабилан приступ, генерише веће количине саобраћаја у поређењу са клијент/сервер архитектуром

Контролни канал представља кључну и централну инфраструктуру сваког ботнета, без којег ни један ботнет не би могао да функционише. Стога је детекција и сузбијање контролног канала, односно комуникације ботмастера и ботова један од кључних метода детекције ботнета. Како је управо детекција овог канала комуникације предмет дисертације, у поглављу 2.2. ће архитектура контролног канала, механизми детекције и избегавања детекције од стране аутора малвера бити детаљније бити описани.

2.1.4 Напад

У финалној фази, када бот прими команду од ботмастера, он извршава команду, која укључује илегалне радње. Илегалне сврхе у које се могу користити ботови су [11]:

- **DDoS напад** – овде нападач/хакер има циљ да онемогући функционисање неке услуге на интернету. Ботови се користе за генерисање огромних интернет протока (који се данас мере и у терабитима у секунди) или огромног броја захтева да ресурси нападнуте мреже или сервера буду преоптерећени. Овај напад користе *Mirai* и *Gafgyt* ботнети [16].

- **Ширење малвера –** ботнети се могу користити у сврху слања нежељених мејлова (у даљем тексту: спам), као део стратегије за даље ширење рачунарских вируса. Једна од превара која се јавља овим путем је слање мејлова такве садржине да жртве напада посете вебсајт који је испрограмирао хакер, и тамо оставе битне информације, које се могу користити за даље нападе на мету. Пример ботнета који шири спам је *Necurs* [18],

чији ботови шаљу спам мејлове како би повећали број ботова. Такође, ботнети могу да се користе и за ширење других малвера, имајући улогу сервера са којег се спушта вирусна апликација. *Necurs* [18] и *Retadup* [19] су примери ботнета који, након заражавања рачунара, скидају додатне малвере.

- **Клик превара –** ботнет симулира кликтање корисника на неки линк на интернету, ради монетарне или друге користи. Пример овога су линкови за рекламе, које доносе зараду власнику сајта који има рекламе. Пример оваквог ботнета је *MethBot* [20].

- **Манипулација онлајн анкетама или играма** – бот имитира гласање/играње реалног корисника, чинећи да ботмастер добије предност тако што његов избор у анкети победи, или победи у онлајн игри. Ово је могуће, јер сваки бот има јединствену IP адресу, што већина анкета и онлајн игара узима у обзир као услов за проверу јединствености корисника.

- **Напад на *IRC* мреже за ћаскање** – ова врста напада је слична *DDoS* нападу, само што има за циљ гашење *IRC* канала за ћаскање уместо сервера.

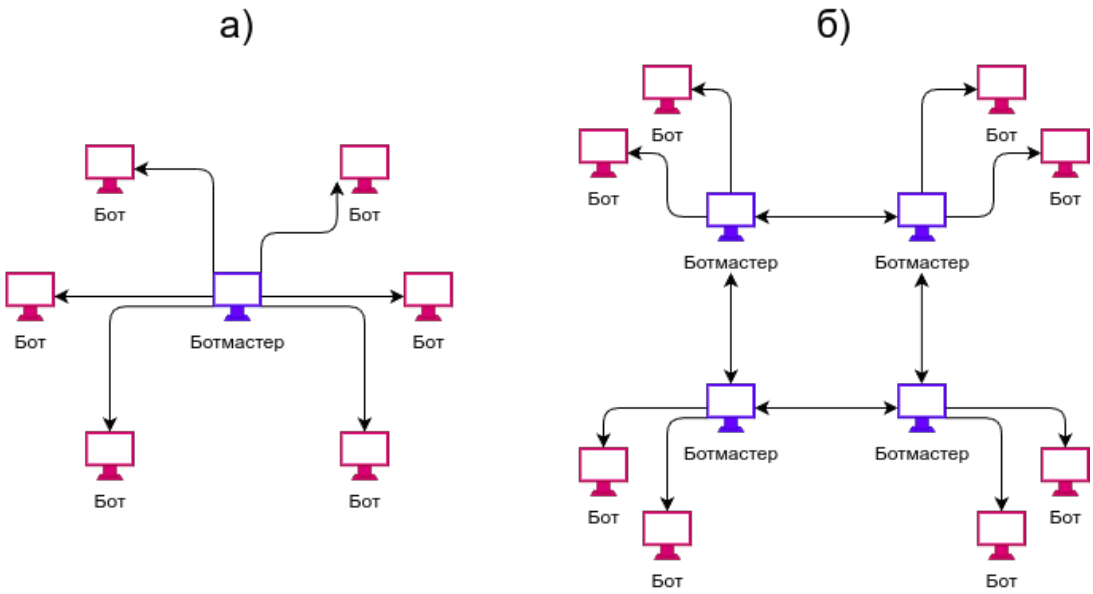
- **Долажење у посед личних података** – ботмастер има за циљ да дође у посед личних података жртве, попут бројева банковних рачуна, корисничких имена, лозинки. Будући да подаци кроз мрежу могу бити енкриптовани, овим путем ботмастер долази до првобитног садржаја пакета које мета шаље са свог рачунара. Пример оваквог ботнета је *Trickbot* [21].

- ***SQL (енг. Structured Query Language, скр. SQL) injection* напад** – овај напад укључује добијања приступа бази података на рачунару мете, користећи слабост софтвера који жртва користи. Тиме што се пажљиво конструише текстуални низ којим се врши претрага у бази података, добија се приступ њеном садржају, доводећи до крађе информација. *Asprox* ботнет користи овај напад [22].

- **Рударење криптовалута** – ботови могу да се, без знања корисника зараженог уређаја, прикључе мрежама које служе за рударење криптовалута. Овим путем ботмастер користи ресурсе туђих рачунара, како би повећао себи шансе за рударење криптовалута. *Coinhive* [23] и *Smominru* [24] служе за рударење *Monero* криптовалуте.

2.2 Командна и контролна архитектура мреже ботнета

Три главне компоненте сваког ботнета су: бот, ботмастер и командни и контролни канал. Архитектура ботнет мрежа може да буде централизована, у којој ботови имају једну заједничку тачку комуникације са једним ботмастером, дистрибуирана (енг. *Peer-to-Peer,* скр. *P2P*), у којој ботови имају индивидуалну комуникацију са једним од већег броја ботмастера који заједнички управљају мрежом, и хибридна, као комбинација претходне две (Слика 2). На основу редоследа слања порука у командном и контролном каналу, ботнети се деле на *push*, где ботмастер први шаље поруке за одржавање комуникације, и *pull*, где бот први шаље поруке за одржавање комуникације.



Слика 2: Архитектуре ботнет мрежа: а) централизована; б) дистрибуирана

Током низа година, будући да је кључан за функционисање ботнета, а самим тим и критична тачка ботнета, командни и контролни канал се развијао под утицајем развоја софтвера за детекцију малвера, како би се избегла детекција и омогућио несметан рад ботнета. Топологија самог ботнета је у директној корелацији са тиме колики проблем представља његова неутрализација. Основни принципи којима се воде хакери који конципирају ботнете су скалабилност, робусност, и анонимност.

Као што је наглашено на почетку овог поглавља, најранији дизајн архитектуре ботнета је био централизована архитектура путем IRC канала. IRC протокол је конципиран 1988. године, и користио се за потребе четовања преко интернета. Испрва се користио искључиво *TCP* протокол и порт 6667, да би се касније увела могућност сигурније комуникације путем *TLS*

(енг. *Transport Level Security*, скр. *TLS*) протокола, и број портова проширио на опсег 6660-6669 и порт 7000. Његова главна функција је могућност креирања канала, који служе као собе за ћаскање у којима се одвија групна конверзација. Канали се налазе на серверима, који су уједно део исте *IRC* мреже. Иако је већина канала јавна, постоје приватни канали за које је потребна ауторизација како би им се приступило. Корисници канала имају нивое приступа, који дефинишу које радње су им омогућене у каналу. Ова архитектура омогућава лако одржавање ботнета, уз минимално заузеће мрежних ресурса, будући да порука коју ботмастер пошаље у канал окида извршавање команде код свих заражених уређаја који су приступили *IRC* каналу. Упркос томе, ове ботнете је лако неутрализовати, што се чини искључивањем *IRC* канала, или пак стављањем одговарајуће *IP* адресе ботмастера на црну листу, чиме престаје свака комуникација између ботова и ботмастера. Поред тога, први ботнети су били примитивни по питању дизајна, јер су у коду вирусне апликације садржали *IP* адресу ботмастера, што се данас статичком анализом кода малвера тривијално открива. Оно што је ботмастерима отежавало ширење ботнета у случају коришћења *IRC* канала јесте нескалабилност мреже. Будући да је сваки канал подржавао око пар хиљада корисника, за одржавање иоле веће мреже, реда величине неколико стотина хиљада ботова, било је потребно неколико стотина сервера, који би, у случају да ботмастер жели да изврши заједнички напад од стране целог ботнета, морали да буду добро искоординисани.

Сходно томе, развили су се механизми за отежавање детекције ботмастера. Централизована архитектура је превазиђена, и замењена је дистрибуираном *P2P* архитектуром. *Storm worm* ботнет је користио протокол *Overnet.* Овај протокол је користио и истоимени сајт за дељење фајлова, да би потом његовом изменом настао протокол *Stormnet* [25]. Још један пример ботнета који је користио *Overnet* протокол је *Trojan.Peacomm* [26]. Недостатак централизације омогућава скалабилност ботнета, будући да чворови мреже одржавају минимално стање за сваку комуникацију, те заузеће ресурса расте спорије него број чворова у мрежи. Недостатак централне тачке за комуникацију обезбеђује робустност ботнета јер блокирањем неког од ботмастера неће бити у потпуности угашен цео ботнет, већ само његов део.

У *P2P* мрежи, сваки чвор мреже може да постане ботмастер. Ово је повољно из два разлога: тиме се лакше сакрива ботмастер, као и олакшава одржавање велике количине чворова. У случају напада на ботнет, потребна је неутрализација велике количине чворова ботнета, како би са сигурношћу био неутрализовао цео ботнет. Ова архитектура је била инспирисана P2P

мрежама које су служиле за дељење фајлова путем интернета. У овој архитектури, сваки чвор комуницира за одређеним бројем чланова мреже, и само са њима и комуницира. Ова архитектура може да буде потпуно без структуре, што значи да сви чворови у мрежи имају исту улогу, или са структуром, где чворови имају различите улоге.

2.3 Методе прикривања командног и контролног канала

У овом одељку биће приказане методе прикривања комуникације између ботова и ботмастера. Командни и контролни канал најчешће постоји током целог животног века ботнета, те стога представља главну мету детекције и сузбијања. Начини рада ботнета су се развили кроз конкуренцију са методама детекције. Из тог разлога, превенција детекције прикривањем командног и контролног канала повећава робусност ботнета. Идеално, комуникација између бота и ботмастера би требало да има такве карактеристике, да алати за анализу саобраћаја не могу да је разликују од уобичајеног мрежног саобраћаја. Приступи за прикривање командног и контролног канала могу се поделити на: приступе који користе *DNS* протокол, *Tor* (енг. *The Onion Router*, скр. *Tor*) протокол, и приступе који користе имитацију протокола или стеганографију.

2.3.1 Приступи за прикривање командног и контролног канала који користе DNS протокол

Будући да је један од првих корака након инфекције малвер апликацијом успостављање комуникације са ботмастером, заражени уређај мора прво да сазна *IP* адресу ботмастера. Ту постоје два приступа: уписивање *IP* адресе у код малвер апликације, или коришћење *DNS* протокола за дохватање *IP* адресе ботмастера. Проблем са првим методом јесте што се статичком анализом кода може веома лако утврдити једна или више адреса ботмастера, а потом те адресе и ставити на црну листу, чиме се ботнет гаси. Стога ботови често користе DNS протокол да ступе у контакт са ботмастером.

2.3.1.1 DNS протокол

DNS протокол служи да повеже симболичка/текстуална имена ентитета на интернету, пријемчивија корисницима интернета, са њиховим *IP* адресама [27]. Користи порт 53 било путем *TCP* или *UDP* протокола, редом за трансфере зона или за слање захтева и одговора. Главна употреба *DNS* протокола је олакшица да корисник путем мрежне апликације приступи

неком уређају путем његовог имена, уместо да памти низ цифара који сачињавају његову *IP* адресу, што је посебно у случају *IPv6* тешко.

Због чињенице да *DNS* представља кључну услугу без које би интернет данас био практично неупотребљив, као и чињенице да се због тога *DNS* саобраћај никада не филтрира и пролази кроз све мрежне уређаје, неки хакери су се одлучили да рад ботнета сакрију путем истог. Главни приступи коришћења *DNS* протокола у ове сврхе су механизми: *DNS* ексфилтрација, *DNS* Fast Flux и *DGA* (енг. *Domain Generation Algorithm*, скр. *DGA*).

2.3.1.2 DNS ексфилтрација

DNS ексфилтрација представља метод скривања порука између бота и ботмастера путем DNS пакета, убацујући поруку у неко од поља DNS захтева [28][29]. Оно што код овог приступа отежава детекцију је чињеница да порука може бити енкриптована, као и чињеница да се DNS ексфилтрација користи од стране неких добро познатих онлајн алата, попут *Sophos* [29].

Један од ботнета који користе овај приступ је *Feederbot* [30]. Наиме, овај ботнет користи RDATA поље у DNS одговору за пренос команди. Ово поље може служити за пренос текстуалног садржаја. Ботмастер енкриптује садржај команде, потом га транспортује у RDATA поље, а остатак DNS поља има валидне вредности, што отежава детекцију. Цело поље је записано у base64 формату. Један део RDATA поља представља 32-бајтну CRC проверу (енг. *Cyclic Redundancy Check*), док је други део порука енкриптована RC4 (енг. *Rivest Cipher 4*) алгоритмом [30]. Други пример малвера који користи текстуалне поруке је W32.Morto worm [31], који користи овај механизам за ширење вирусних апликација. У овом случају, у DNS одговору се добију енкриптоване локације са којих је могуће скинути комплетну вирусну апликацију, након чега бот на рачунару инсталира ту апликацију.

DNS се може користити и на друге начине за размену порука. Сет Бромбергер, радећи за Министарство за енергетику Сједињених америчких држава, осмислио је начин за извлачење података од организација путем DNS упита. Наиме, ботмастер би морао да постави домен, и има контролу над ауторитативним сервером имена. Након што би бот успоставио комуникацију са одговарајућим DNS сервером, периодично би слао упите, у којима би сакривао податке које преноси. Ти подаци би испрва били енкриптовани, и додати на почетак домена ботмастера. Будући да ботмастер има контролу над ауторитативним сервером за тај домен, сваки DNS упит овог типа ће бити послат њему, и ботмастер ће моћи да реконструише послате податке, а у DNS одговору да боту пошаље поруке. Екстензија ове идеје је

предложена у [1], а она се састоји у томе да ботнет шаље DNS упите у исто време када и рачунар на којем се налази, како би избегао детекцију.

2.3.1.3 DNS Fast Flux

DNS Fast Flux представља метод за периодичну промену IP адресе која одговара DNS запису, како би било теже откривање праве IP адресе ботмастера. Ова пракса да се зависно од DNS упита, другачија IP адреса јавља у одговору, је широко распрострањена код CDN (енг. *Content Delivery Network*) мрежа. CDN мреже се користе у случају пружања глобалних веб услуга, као што је у случају сајтова Гугла, Фејсбука, Амазона, и других како би се обезбедило равномерно искоришћење сервера и оптимизовале перформансе приступа тако што би корисник који покушава да приступи сервису, сервис добио од њему локационо и тополошки најближег сервера. У случају CDN мрежа, DNS одговор генерално садржи више адреса, које су све валидне, а потом једна од тих бива одабрана као коначна дестинација за корисника.

Варијација механизма рада CDN мреже је *Fast-Flux Service Network* (FFSN), где се ботмастер крије иза велике количине мастер рачунара који су део ботнета. Суштина овог механизма јесте да се обезбеди да ботнет нема централну тачку, чијом неутрализацијом ботнет престаје са радом. Иако ће у код малвер апликације бити уписан јединствени домен ботмастера, тај домен се неће увек преводити у исту IP адресу, а DNS запис тог домена ће имати краће време трајања (енг. *Time to Live,* скр. *TTL*) од уобичајеног (реда величине од неколико минута до пар сати). То значи да различити ботови, иако шаљу DNS захтеве на исти домен, ступају у контакт са различитим IP адресама, што отежава детекцију ботнет мреже и свих ботмастера.

2.3.1.4 Domain Generating Algorithm - DGA

Други познати начин избегавања детекције IP адресе ботмастера путем DNS протокола јесте коришћење алгоритма за генерисање домена (енг. *Domain Generating Algorithm*, скр. DGA). Разлог коришћења овог приступа јесте тај да се избегне уписивање домена ботмастера директно у код малвер апликације, а самим тим и избегла детекција путем статичке анализе кода. Код овог приступа малвер насумично претражује домене које генерише DGA у коду малвера, који припадају ботмастеру, како би отпочела комуникација путем командног и контролног канала. Предуслов је да ботмастер мора да креира скуп домена које креира идентична инстанца DGA за ботнет мрежу, пре него што пусти у етар вирусну апликацију. Генератор случајних домена мора да буде конзистентан на сваком зараженом рачунару, те најчешће користи тренутни датум и време као улаз у тај алгоритам којим се добијају домени.

Генерисање домена најчешће користи један од два приступа: аритметички, где се на основу слова алфабета која се насумично бирају конструише име домена, и језички, где се на основу листе речи формира име домена. Код језичког приступа листа речи је или садржана у самој вирусној апликацији, или се пак извлачи из јавно доступног документа [32].

Главна предност овог алгоритма је тај што, генеришући велику количину домена, обезбеђује редундансу у структури командног и контролног канала. У случају да један домен нестане, други ће убрзо преузети његово место. Пример овога је *Conficker* бот који на свака три сата генерише 250 домена, који су базирани на тренутном времену [33]. Вирусна апликација потом покушава да контактира ботмастера генеришући домене користећи исти алгоритам. Главна мана овог приступа је да, уколико неко дође у посед начина генерисања ових домена, може да покуша да добије власништво над њима пре ботмастера, ефективно неутралишући ботнет.

2.3.2 Прикривање командног и контролног канала путем Tor протокола

Tor (енг. *The Onion Router*, скр. *Tor*) је услуга која омогућава анонимност приступа веб страницама путем интернета, коришћењем мреже Tor сервера [34]. Основни принцип рада Tor мреже јесте слање саобраћаја кроз чворове Tor мреже, на сваком кораку енкриптујући садржај пакета, како се би отежало прислушкивање истих и утврђивање почетне и крајње тачке пута пакета. На основу садржине пакета, веома је тешко утврдити ко су првобитни пошиљалац и прималац (изузев временском корелацијом пакета на различитим тачкама у мрежи на шта Tor мрежа није отпорна). Због ових особина, Tor мрежа је пожељна инфраструктура коју често користе хакери, а самим тим су постојали и случајеви када су малвери користили Tor мрежу као помоћ у одржавању командног и контролног канала. Како би се рачунар прикључио овој мрежи, довољно је само да инсталира Tor веб претраживач на свој рачунар. Тада рачунар може да шаље саобраћај путем Tor мреже. Једна од услуга Tor мреже се назива скривене услуге (енг. *hidden services*). Ова услуга омогућава да се чвор у Tor мрежи сакрије иза посредника (енг. *proxy*), скривајући идентитет чвора. То значи да један од чворова у мрежи служи као посредник, који енкриптује сав садржај који долази од чвора који користи ову услугу, и прослеђује га даље коме је намењен. Посреднику се приступа путем *.onion* линка. Пример ботнета који користи Tor мрежу је *ZHTrap* [35], који користи Tor мрежу преко посредника за комуникацију са командним и контролним каналом.

2.3.3 Прикривање командног и контролног канала путем опонашања протокола и стеганографије

Док је циљ коришћења Тор мреже прикривање комуникације између два уређаја на интернету, постоје методи чији је циљ прикривање чињенице да ико комуницира. Најчешћи метод овог типа је стеганографија. Стеганографија (од грчке речи која значи *тајновито писање)* јесте начин писања порука који нико други не може да чита, осим пошиљаоца и примаоца. Стеганографија се користи хиљадама година, и поново је заживела у дигиталном добу. Главна добит коришћења стеганографије је тај да учини комуникацију између два уређаја неразумљивом за било који други уређај. Постоје два главна начина за употребу стеганографије у сврхе скривања командног и контролног канала. Први је тај да се комуникација путем командног и контролног канала учини таквом да имитира други протокол, а други је утискивање поруке у други садржај, попут слике. Већина медија, попут текста, слика, аудио и видео фајлова, могу да се користе за пренос скривених порука. У најједноставнијем случају, могу се додати метаподаци који садрже ту информацију, но, овај приступ је лако открити. Алтернативни метод је измена садржаја самог фајла, тако да изгледа готово идентичан оригиналу, а ипак садржи скривену поруку. У случају слике, постоји метод скривања поруке у најмање значајном биту сваког пиксела, док је у случају аудио сигнала то увођење еха са паузама, где дужина паузе указује на то који је податак скривен. Стеганографија се, у комбинацији са HTTP протоколом, може користити да имитира уобичајено понашање корисника, попут посећивања интернет странице са које скида слике, видео фајлове, или звучне фајлове, како би се прикрила чињеница да долази до комуникације између бота и ботмастера. Пример ботнета који користи стеганографију за ширење злонамерне апликације је *Smominru*, који је апликацију скривао у слици познате певачице Taylor Swift [24].

Опонашање протокола је метод за избегавање детекције у којем се прикрива комуникација између ботова и ботмастера тако што се пренете поруке по облику чине као да припадају другом протоколу. У пракси, најчешћи начин је тај да пакети пренети преко Тор мреже имитирају саобраћај који генерише апликација Скајп (енг. *Skype*). Разлог зашто се користи баш ова апликација јесте тај што је брзина одзива велика, а заузеће на мрежи велико.

Skymorph [36] бот користи Тор саобраћај, маскиран као видео позив преко Скајпа. Он користи приватне чворове Тор мреже, и Скајп протокол, све у циљу избегавања детекције. Клијент шаље поруку брицу путем Скајпа на UDP порту великог броја, и почиње *Diffie-Hellman* протокол за размену кључева. У поруци се налази IP адреса, UDP порт, и јавни кључ. Потом

бриџ у одговору пошаље исте информације. Клијент и бриџ започну видео позив, али се бриџ не одазове. Током неуспешног успостављања позива, енкриптовани подаци се пошаљу путем установљених UDP портова, и успоставља се командни и контролни канал.

StegoTorus [37] пружа додатне могућности енкрипције пакета чворовима у Тор мрежи. Суштина овог метода је да се порука дели на мање пакете насумичне дужине, који се преносе са насумичним временом између пакета путем стеганографије. Како би емулирао реално понашање корисника, овај ботнет користи мрежни саобраћај генерисан преко Скајп или Вентрило позива из прошлости као модел својих порука. Пакети који се прослеђују користе валидна заглавља која шаље Скајп апликација. Неке верзије ове апликације користе исти принцип са HTTP пакетима, где се фалсификују захтеви и одговори, који личе на нормалан саобраћај, док се у ствари преносе поруке за одржавање командног и контролног канала. Ови HTTP пакети долазе из наснимљеног мрежног саобраћаја, док се у заглављима крију подаци који се преносе.

CensorSpoofer [37] користи нешто другачији приступ, у којем бот и ботмастер користе два сасвим различита протокола за комуникацију. Метод којим се ботмастер служи за скривање јесте тај што се скенирањем портова открије адреса на којој је порт за SIP (енг. *Session Initiation Protocol*) отворен. Ова адреса постаје кандидат за фиктивног хоста. Бот потом шаље HTTP захтеве ботмастеру путем мејла или сервиса за инстант поруке, док сервер враћа одговор у облику VoIP (енг. *Voice over Internet Protocol*) порука, имитирајући саобраћај фиктивног хоста на мрежи.

2.4 Закључак

Централну тачку сваког ботнета представља ботмастер. Уколико се детектује са којих IP адреса комуницира са ботовима, те адресе је могуће изоловати како би се ботнет неутралисао. Заједничку одлику свих ботнет мрежа представља комуникација између бота и ботмастера, коју чини командни и контролни канал и/или канал којим се одржава комуникација (енг. *heartbeat*). Детекцијом ове комуникације и чинилаца који је одвајају од уобичајене комуникације путем рачунарских мрежа, могуће је детектовати ботмастера и сузбити рад ботнета. Фактор који отежава детекцију су механизми скривања командног и контролног канала. Међутим, IoT малвери који су тема овог истраживања нису користили ниједан такав механизам, што је показано у наредном поглављу, а такође, фокус сакривања командног и контролног канала је више на сакривању идентитета крајњих тачака, а не толико

на обрасцима комуникације што ће бити показано у наставку да је од кључног значаја за детекцију ботнета.

3 Анализа командног и контролног канала савремених ботнета

Како би се анализирала мрежна комуникација командног и контролног канала савремених ботнета у циљу детекције истих, спроведено је истраживање у 3 фазе. Прва фаза је представљала прикупљање и анализирање понашања ботнета у периоду 2019-2021. године. Потом је уследила друга фаза, коју чини дизајнирање и реализација метода за детекцију ботнет саобраћаја на бази машинског учења и прикупљених узорака. У трећој и финалној фази је прикупљено још примерака ботнет малвера, у периоду 2021-2023. године, примена напреднијих техника машинског учења за детекцију истих, поређење различитих техника машинског учења, као и анализа успешности детекције новијих примерака ботнет малвера помоћу старијих.

Разлози за раздвајање и прикупљање примерака ботнет малвера у две фазе су вишеструки. Као прво, будући да може постојати више различитих варијанти једног ботнета, потребно је испитати колика је сличност међу њима. Пример за ботнет са великим бројем варијанти је *Mirai*, који је само у периоду 7.8.2016 - 30.9.2016. имао 24 варијанте [16]. Такође, праћење ботнет малвера током дужег временског периода, као и испитивање могућности новијих узорака помоћу старијих пружа директан увид у инваријантност неких одлика ботнета.

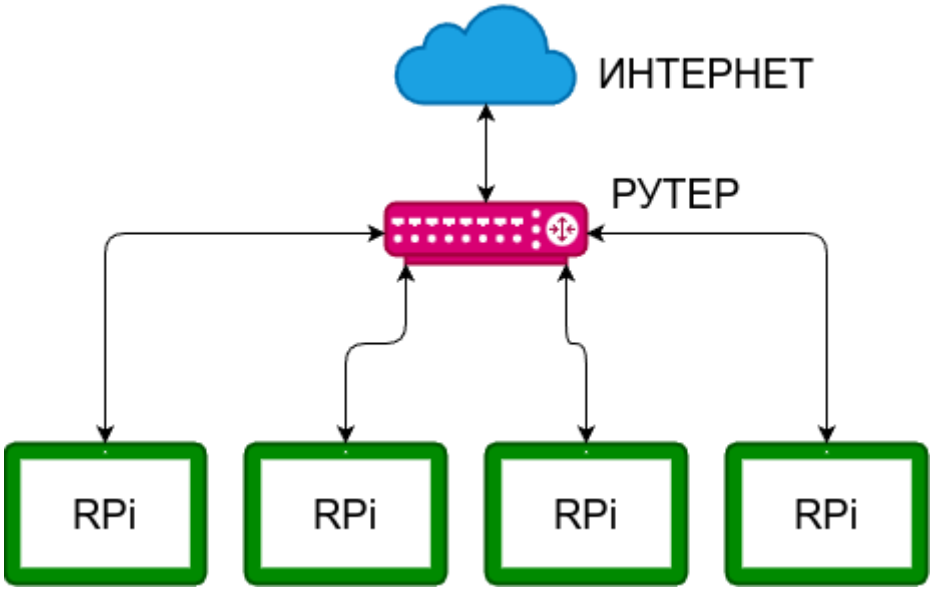
У овом поглављу биће описана методологија прикупљања узорака ботнет вируса, који су прикупљени у периоду прве фазе истраживања, као и закључци о карактеристичном мрежном понашању ботнет вируса.

3.1 Методологија прикупљања узорака ботнет малвера

Будући да се ботнети користе у малициозне сврхе, потребно је бити обазрив приликом прикупљања узорака, како не би дошло до нежељене инфекције лабораторијског окружења или како лабораторијско окружење не би послужило као платформа за напад. Оно што олакшава овај корак јесте чињеница да постоје базе података вирусних апликација, доступне на интернету (нпр. *URLhaus* [38]), које обележавају сваки примерак вирусне апликације

таговима, који указују на фамилију ботнета, и тип егзекутабилне апликације. У тренутку писања докторске дисертације, ова база података, основана од стране ентузијаста у области информационе безбедности из Швајцарске, а чији су подаци уграђени у већи број комерцијалних производа и производа отвореног кода, броји више од 2.500.000 локација на интернету на којима се налазе малвер вирусне апликације, док се сваког дана пријави између 200 и 400 нових локација [38]. *URLhaus* је почео са радом марта 2018. године, и заслужан је за неутрализовање више од 100.000 веб-сајтова који су делили вирусне апликације [39].

Прикупљени малвер је анализиран статичком анализом кода и динамичком анализом понашања малвера и то пре свега образаца њихове мрежне комуникације, као и анализом у оквиру *Cuckoo sandbox* окружења [40]. Како би се анализирало понашање прикупљених IoT ботнет апликација, коришћена су два *RaspberryPi 2B*, и два *RaspberryPi 3* уређаја са инсталираним *Raspbian* оперативним системом. *RaspberryPi* уређаји су потом повезани на рутер, који је повезан на интернет. Скица мрежне топологије је приказана на слици 3. На њих су, у првој фази истраживања, инсталиране преузете вирусне апликације у периоду између 15. јуна 2019. и 21. септембра 2020. Разлог зашто су одабрани ови уређаји јесте то што поседују ARM процесорску архитектуру, за коју је компајлирана већина анализираних IoT ботнета. Анализиране су две IoT ботнет фамилије вируса које су активно коришћене од стране хакера у том периоду: *Mirai* и *Gafgyt*. Анализирано је 19 примерака *Mirai* ботнета: *666_anon*, *ab.arm7*, *arm7.idopoc2*, *Astra*, *armv7l*, *kalon.arm*, *okane.arm*, *pandora.arm7*, *zehir.arm7*, *ntpdd.arm8*, *r4z0r.arm7*, *Hilix*, *hoho*, *Mercury.arm7*, *nuclear.arm7*, *RebornGang.arm*, *shibui.arm*, *smtpd*, *xs.arm7.* Том приликом забележена су три DDoS напада, који су прекидани одмах по уочавању почетка напада. Анализирано је и 18 примерака *Gafgyt* ботнета: *armv6l_1*, *armv6l*, *cc9arm6*, *soul.arm6*, *eagle.arm*, *eagle.arm7*, *Demon.arm4*, *TacoBellGodYo.arm4*, *frag.arm*, *yakuza.arm6*, *assailant*, *eagle.arm7*, *beastmode*, *boss*, *gang*, *packets.arm7*, *seraph.arm7*, *SNOOPY*, и забележено је пет DDoS напада на њима. Тестирање ботнет вируса је неопходно на реалним уређајима, будући да неки примерци ботнета могу да успешно детектују да су инсталирани на виртуелној машини, и након тога престану с радом [41]. Ова појава се другачије назива полиморфизам малвера – могућност детекције и различито понашање у различитим окружењима.



Слика 3: Мрежна топологија лабораторијског окружења

На свим уређајима су постављене јавне *IP* адресе, заједно са безбедносним правилима која штите остатак локалне мреже од комуникације са зараженим уређајима. Наиме, блокирани су пакети од и ка остатку локалне мрежне, изузев адресе DNS сервера. Сви вирусни примерци су прикупљени са сајта *URLHaus* [38].

Динамичко понашање малвера је било упоређено са резултатима добијеним *Cuckoo* софтвером који је познати анализатор малвера [40]. Принцип рада овог софтверског алата је следећи: малвер апликација се активира унутар изолованог окружења (познато под називом *sandbox*), да би се потом анализирало њено понашање. Изоловано окружење је у виду виртуелне машине која је мрежно изолована од остатка оперативног система. Анализа понашања прикупља све догађаје који се десе након покретања апликације, попут: приступи диску, креирање процеса, мрежног саобраћаја, и класификује догађаје од мање сумњивих ка више сумњивим. Резултат ове анализе је скор од 0 до 10, који квантификује целокупно сумњиво понашање апликације. Веће вредности скора значе сумњивије понашање. *Cuckoo* је у 3 случаја дао оцену нижу од 3, која указује да није реч о малициозном софтверу, док је у 21 случају дао оцену нижу од 5, што даје индикацију да је можда реч о малверу, али није дао више корисних понашања о динамичком понашању малвера. Ручна динамичка анализа описана у наставку овог поглавља је била далеко информативнија.

Поред тога, статичка анализа кода је коришћена, како би се у потпуности истражиле карактеристике малвера. Коришћени алат за статичку анализу кода је био *Ghidra*, који је развила америчка Национална служба за безбедност (енг. *National Security Agency*, NSA) и објавила 2019. године [42]. Главне функције овог програма су дисасемблирање, асемблирање,

декомпајлирање извршних фајлова, графовски приказ позива метода у оквиру апликације. Статичка анализа кода је често давала *IP* адресе ботмастера које су касније уочене и у обрасцима комуницирања, могућност анализе логике рада контролне комуникације (нпр. могле су да се уоче команде које је бот могао да изврши) као и могућност анализе који сегменти кода једног малвера су били преузимани у новијим варијантама. Међутим, даља анализа динамичког понашања малвера из анализе кода је тешка.

Динамичко мрежно понашање вирусних апликација је снимано помоћу *tcpdump* алата, и ти пакети су снимљени у *.pcap* фајловима који се и данас чувају и омогућавају понављање и проверу експеримената. Алат *tcpdump* је био покретан на зараженим уређајима у лабораторији пре покретања малвера, како би се ухватили сви пакети које малвер генерише од тренутка покретања. Све *IP* адресе које нису локалне су биле анализиране, како би се детектовали њихови власници и локације, користећи *IP* и *DNS* алате за претрагу.

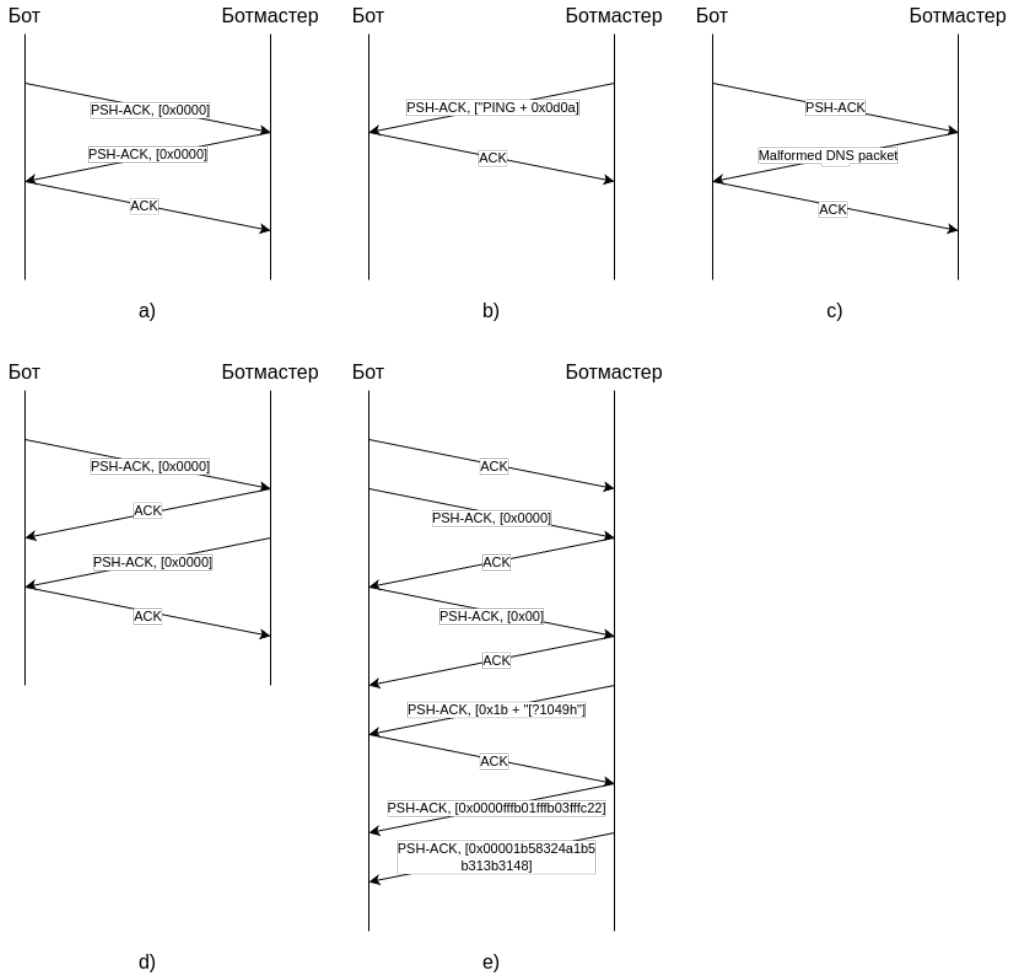
3.2 Анализа комуникације прикупљених узорака

У овом одељку, биће приказана анализа комуникација између бота и ботмастера, са детаљним приказом размене пакета. Прво ће бити представљене опште карактеристике ботнет саобраћаја, након чега ће уследити детаљнија анализа мрежног саобраћаја. У одељку 3.2.1 је приказана анализа саобраћаја *Mirai* примерака малвера, док је у одељку 3.2.2 приказана анализа саобраћаја *Gafgyt* примерака малвера. Анализа је подељена у три одељка: успостављање комуникације са командним и контролним каналом, одржавање конекције, и команда за напад.

3.2.1 Саобраћај *Mirai* примерака малвера

Код *Mirai* ботнета, конекцију иницира заражени уређај користећи *TCP* иницијализацију (енг. *handshake*), након које шаље информацију о уређају командном и контролном каналу. Анализирани примерци малвера су користили различите *TCP* портове за комуникацију са командним и контролним каналом најчешће ван опсега „добро познатих“ (енг. *well-known*) портова (нпр. 3301, 1337, 1791). У неким случајевима су коришћени добро познати портови, како би малвер прошао кроз мрежне баријере и антивирус софтвер (нпр. ntpdd.arm8 који је користио порт 53, резервисан за DNS саобраћај). Том приликом нису забележени стварни DNS упити преко овог порта. Конекција са ботмастером се одржава или од стране заражене машине (*pull* механизам), или пак од стране командног и контролног канала (*push*

механизам). У овој фази су се поруке размењивале периодично, са периодом који је најчешће у првој фази истраживања био 60 секунди. Тачан број размењених порука, као и садржај истих, зависи од појединачне вирусне апликације и приказан је на слици 4. Када је ботмастер одлучивао да изведе *DDoS* напад, слао је команду за напад зараженој машини, путем пакета са постављеним PSH и ACK TCP флеговима. Команда садржи све потребне параметре да се напад на неки уређај повезан на интернет изврши: *IP* адресу, протокол и порт по коме је потребно извршити напад. Заражена машина одговара ACK пакетом, и почиње напад. За све тестиране вирусне апликације, фаза успостављања конекције је била идентична. Дијаграм тока пакета за фазу одржавања комуникације је дат на слици 4. Садржина појединачних пакета је дата у заградама, стрингови су означене знацима навода, док је хексадецимални код означен са почетним *0x*. Знак плус означава конкатенацију стрингова.



Слика 4: Mirai комуникација путем командног и контролног канала у случају примерака вирусних апликација: a) ab.arm7, pandora.arm7 и kalon.arm; b) armv7l; c) ntpdd8.arm8; d) okane.arm и r4z0r.arm7; e) zehir.arm7

Током фазе напада, командни и контролни канал шаље стринг који садржи команду за напад, након чега напад започиње. Команда може да буде у два облика, текстуални и хексадецимални. На пример, у случају *armv7l*, командни и контролни канал шаље PSH-ACK пакет следеће садржине:

```
".UDP 40.81.59.133 30142 20 32 0 10 "
```

На основу команде и забележеног мрежног саобраћаја, извлачи се закључак да се користи *UDP* протокол. Користећи статичку анализу кода показано је да остали параметри команде, слева надесно, представљају: *IP* адресу мете, дестинациони порт, трајање напада, опција да ли се користе SOCK_DGRAM или SOCK_RAW за креирање сокета, величина бафера, и број напада. Након што прими команду, заражена машина шаље ACK пакет, и напад започиње. Ово је пример када команда има текстуалну форму.

У случају *r4z0r*, командни и контролни канал шаље PSH-ACK пакет следеће садржине:

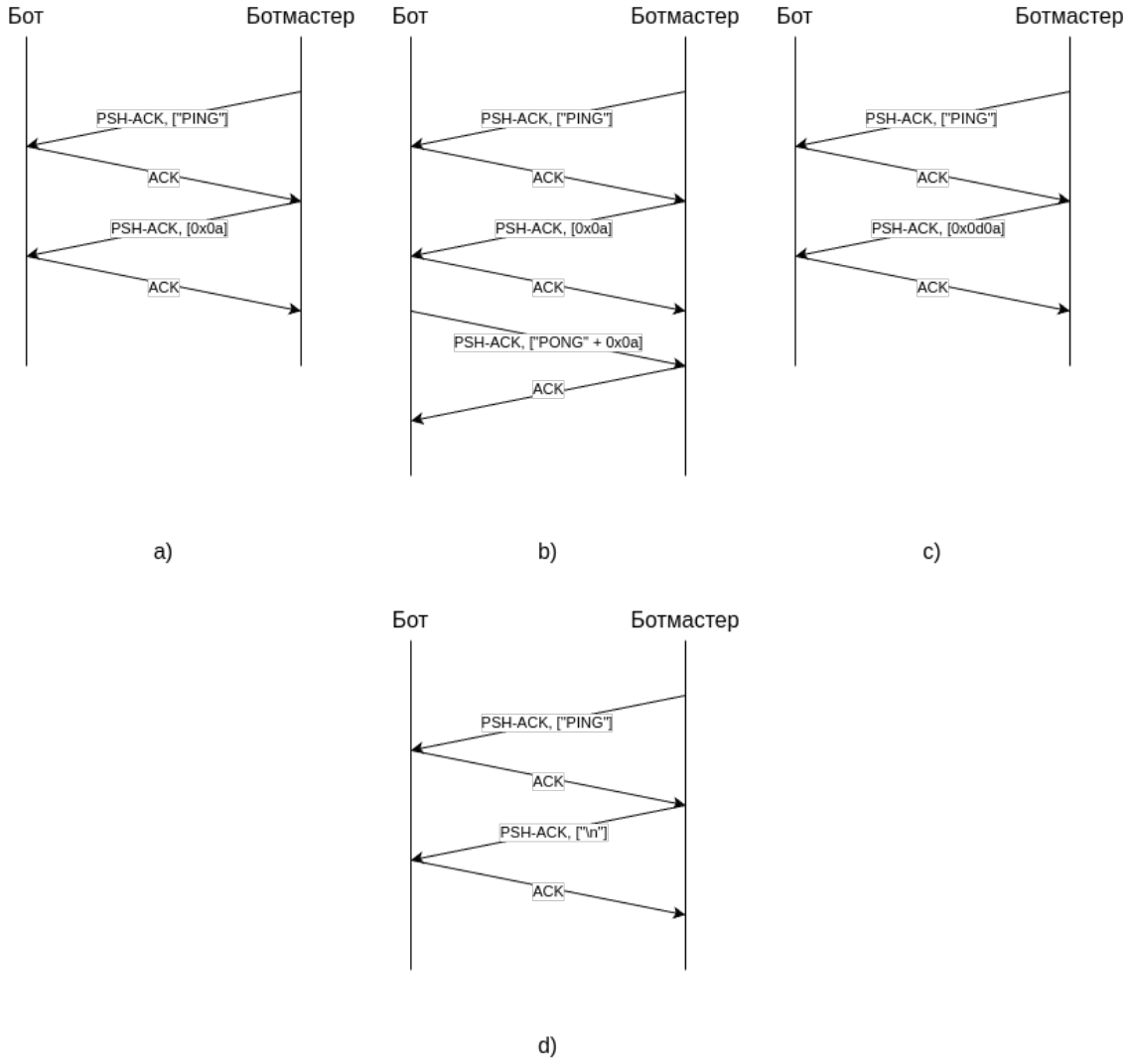
```
"00 12 00 00 00 1e 09 01 67 5f dd a7 20 01 07 02 38 30"
```

Стринг команде је написан у хексадецималном облику. Када се хексадецимални код претвори прво у децимални, па у ASCII, дестинациони порт и *IP* адреса мете се могу дешифровати (0x 67 5f dd a7, што даје 103.95.221.167 као дестинациону адресу мете; 0x3830, што конвертовањем у ASCII даје вредност 80 за дестинациони порт). Након примања команде, заражена машина шаље командном и контролном каналу ACK пакет, и бот почиње са нападом, тачно на ове *IP* адресе наведене у команди, што је уочено у снимљеним .pcap фајловима.

3.2.2 Саобраћај *Gafgyt* примерака малвера

У случају *Gafgyt* ботнета, слично као код Mirai ботнета, заражена машина иницира *TCP* сесију по различитим портовима. Примерци су користили различите *TCP* портове за командни и контролни канал (нпр. 87, 17769, 58215). Ботмастер се контактира директно путем *IP* протокола без коришћења разрешења на *DNS* серверу. Након што се успостави комуникација, заражена машина шаље ботмастеру путем командног и контролног канала стринг који описује хардверско и софтверско окружење у којем је покренут малвер. Ова конекција се одржава периодичном разменом пакета, са периодом од 60 секунди. Тачан број пакета зависи од вирусне апликације као и садржаја пакета. Напослетку, ботмастер шаље

зараженој машини команду за напад, путем пакета са постављеним PSH-ACK флеговима, са свим потребним параметрима за напад. Бот одговара ACK пакетом, након чега почиње *DDoS* напад. У случају свих вирусних апликација, фаза успостављања конекције је идентична. Дијаграм тока пакета за фазу одржавања комуникације је дат на слици 5. Садржина појединачног пакета је дата у заградама, стрингови су означени знацима навода, док је хексадецимални код означен са почетним *0x*. Знак плус означава конкатенацију стрингова.



Слика 5: Gafgyt комуникација ӣуѣем командноӣ и конѣролноӣ канала у случају ӣримерака вирусних аӣликација: а) cc9arm6, Demon.arm4, eagle.arm и soul.arm6; b) eagle.arm7 и TacoBellGodYo.arm4; c) frag.arm; d) yakuza.arm6

У фази напада, командни и контролни канал шаље пакет са командом за напад, након чега напад почиње. Команда се добија у текстуалном формату, при чему порука може бити енкриптована.

На пример, у случају *TacoBellGodYo* малвера, командни и контролни канал шаље пакет са постављеним PSH-ACK флеговима следеће садржине:

```
"!* STD 174.214.34.8 4370 300"
```

Док у случају *eagle* малвера, команда има следећи формат:

```
"!* UDP 13.83.247.215 30217 100 32 0 10"
```

Посматрајући ову команду, и поредећи је са снимљеним мрежним саобраћајем, први аргумент представља протокол који се користи. У оба случаја, у питању је *UDP* протокол. Користећи алате за статичку анализу кода, откривено је да се користи синтакса слична оној код *Mirai* малвера. Аргументи ове команде представљају: *IP* адресу мете, дестинациони порт, дужину трајања напада, опцију да ли се користе SOCK_DGRAM или SOCK_RAW за креирање сокета, величину бафера, и број напада. Након примања команде, заражена машина шаље командном и контролном каналу ACK пакет, и бот почиње са нападом.

Декомпајлирање вирусних апликација и статичка анализа кода су још открили да постоје и други могући напади попут коришћења *TCP* протокола преко предефинисаног порта, као и друге команде које бот може да прими, попут престанка бота са радом и паузирање рада бота.

3.3 Закључци о командном и контролном каналу савремених IoT ботнета

Ова анализа је служила како би се детектовао образац мрежног понашања који је карактеристичан за најновије верзије *Mirai* и *Gafgyt* малвера за IoT уређаје. Може се извући неколико битних закључака. Као прво, како би се избегла анализа садржине пакета која користи анализу базирану на потписима, различите верзије малвера користе пакете различите садржине у командама командног и контролног канала и команди напада. Друго, све вирусне апликације су ступиле у контакт са ботмастером директно, без употребе DNS упита. То значи да малвер за IoT уређаје још увек не користи напредне технике за прикривање адресе

ботмастера, попут DNS флукса и алгоритма генерисања домена, али такође је битно да се ова карактеристика може користити као улаз у анализаторе сумњивог понашања. Треће, сви примерци оба типа малвера су показали карактеристични циклус командног и контролног канала, као и периодично одржавање комуникације, што се не може детектовати анализом токова или заглавља пакета. Разлог томе је што анализа тока извлачи агрегатне статистике између два уређаја на мрежи, не посматрајући време приспећа сваког пакета, док анализа заглавља пакета посматра саобраћај на нивоу индивидуалног пакета. Ниједан од ова два приступа не посматра мрежни саобраћај као временску серију која је сачињена од мрежних пакета. Четврто, статичка анализа кода је детектовала стратегије које хакери користе како би избегли детекцију: константна измена портова, стрингова који се користе у пакетима контролног канала, и IP адреса преко којих ботови комуницирају. Ипак, целокупна архитектура малвера остаје иста током дужег временског периода, без измене карактеристичног обрасца комуникације. Анализа коришћењем *sandbox* алата није била нарочито корисна, будући да је скуп информација везаних за динамичку анализу малвера скроман за детаљнију анализу (откривена је најчешће само IP адреса). Ови резултати су усмерили даље истраживање у правцу креирања програмског процесирања пакета заснованог на софтверски дефинисаним мрежама које даје богатије статистичке карактеристике мрежних токова потребне за детекцију командног и контролног канала, а што је приказано у наредном поглављу.

4 Детекција ботнета коришћењем статистичких параметара унутар мрежних токова

У овом поглављу биће речи о постојећим системима детекције ботнет саобраћаја, као и скуповима података који се користе за такве системе. Потом ће бити описан оригиналан метод прикупљања података из временских серија ботнет саобраћаја, као и систем за детекцију ботнета базиран на софтверски дефинисаним мрежама - *PI-BODE*. Затим следи опис система за детекцију ботнет саобраћаја базираног на машинском учењу, опис његових делова, као и експеримената који су спроведени. На крају поглавља, биће дата дискусија резултата.

4.1 Преглед научне литературе у области детекције ботнета и напада изведених ботнетима

Као што је претходно речено, током претходне деценије, разорни напади од стране ботнета на велике рачунарске инфраструктуре су привукли велику пажњу научне заједнице широм света. Две најистраженије теме у области везаној за ботнете су: 1. детекција напада и образаца напада (најчешће истраживани напади су *DDoS* напади), који обухвата највећи део истраживања ботнета, и 2. детекција ботнет инфраструктуре и понашања ботова. Пошто је главна особина *DDoS* напада велики број конекција које долазе од различитих IP адреса, или пак велики обим протока, циљ истраживања у правцу детекције напада почива на откривању аномалија у броју конекција и укупног обима протока из снимака мрежног саобраћаја [9]. Ипак, тај приступ даје информацију о нападу након што се исти десио. Други приступ – детекција комуникационе инфраструктуре ботнета даје могућност детекције ботова и ботнета и пре него што се покрену *DDoS* напади, што потенцијално има већи практични значај јер заиста може да заштити неку мрежни инфраструктуру. Детаљни преглед радова у овој области, који најчешће користе приступ базиран на машинском учењу, је дат у [43], а најзначајнији радови за истраживање у оквиру ове дисертације су надени у наставку овог поглавља.

Једна од првих студија које се тичу комуникације путем командног и контролног канала [44] се фокусирала на *IRC* комуникацију, екстрахујући 16 одлика из заглавља пакета. Ове одлике представљају статистике тока, попут укупног броја пакета, укупног броја бајтова, хистограма величина пакета, варијансе бајтова и времена међудолазака пакета (енг. *inter-arrival times*) за сваки ток. С друге стране, скорија студија [45] је дала богатији модел података о мрежном саобраћају, као и скуп података од 29 забележених и 14 генерисаних одлика мрежног саобраћаја, који се тичу специфичности група токова, као и 3 категоричка поља. У међувремену, неки аутори су истраживали одлике мрежног саобраћаја ботнет комуникације за случај комуникације путем командног и контролног канала код ботнета базираних на *HTTP* протоколу [46][47]. Ванг и др. [48] су осмислили систем под називом *BotMark*, који комбинује статистичке и графовске одлике мрежног саобраћаја и хибридну анализу користећи скорове сличности, стабилности и аномалије. Ти скорови се у овом методу користе као улаз за ансамбл метод, са циљем класификације мрежног саобраћаја. Једна од нарочито занимљивих карактеристика овог система је метод детекције, који се ослања на посматрану сличност токова (ботови имају сличне комуникационе обрасце са ботмастером) као основу за детекцију ботнет комуникације.

Кусак и др. [49] су у свом раду направили систем за детекцију *ransomware*[3] типа компјутерских вируса заснован на машинском учењу и екстракцију одлика на бази софтверски дефинисаних мрежа. Овај тип компјутерских вируса енкриптује податке са инфицираног рачунара, и потом тражи новац за њихово декриптовање. Скуп мрежних одлика у овом раду укључује одлике попут времена међудолазака пакета, однос између броја долазних и одлазних пакета и дужинâ налета пакета, које имају специфичне вредности у случају комуникације компјутерских вируса. Још један рад који предлаже употребу софтверски дефинисаних мрежа за детекцију опасности на *IoMT* (*IoT* у медицинске сврхе) је рад Лиакат и др. [50]. Аутори овог рада су користили софтверски дефинисане мреже за прикупљање података о пакетима и доношење одлука о прослеђивању пакета, што се може искористити за заустављање ширења ботнета. Механизам детекције користи конволуционе неуралне мреже (енг. *Convolutional Neural Networks,* скр. *CNN*) и *Cuda Deep* неуралну мрежу са краткорочном и дугорочном меморијом (енг. *Cuda Deep Neural Network Long Short Term Memory,* скр. *cuDNNLSTM*). CNN је коришћен као метод за одабир одлика, док је за класификацију коришћена *cuDNNLSTM* мрежа. Билц и др. [51] у свом раду се фокусирају на детекцију ботнет комуникације путем командног и контролног канала користећи *NetFlow*

3 Тип малвера који врши енкрипцију корисничких датотека, и тражи откуп за шифру која би их декриптовала.

статистичке параметре за алгоритам машинског учења. *NetFlow* је протокол који је развила фирма Cisco, а чија главна намена је прикупљање статистике о мрежним токовима инспекцијом заглавља пакета [52]. Аутори су приметили неке карактеристичне обрасце ботнет комуникације у примерцима малвера које су анализирали, попут периодичног генерисања нових токова, величине токова, образаца у приступу клијената и међутоковско понашање у времену. Међутоковске статистичке величине су потом искористили како би добили богатији скуп одлика, које су касније искоришћене за детекцију командног и контролног канала. Блез и др. [53] су предложили технику детекције аномалија која указује на промену на мрежном порту за детекцију ботнета. Овај метод је оријентисан на иницијалну фазу животног циклуса ботнета – *TCP* скенирање других уређаја, и адаптиран је да детектује чак и најмање промене. Корионитис и др. [45] су радили на детекцији различитих напада на *IoT* инфраструктуру укључујући скенерске нападе (скенирање и посматрање понашања), *DoS* и крађу информација. Рад Де ла Торе Пара и др. [54] даје преглед IoT ботнета и метода за детекцију истих. Аутори су користили *LSTM* (енг. *Long-Short Term Memory*, скр. *ЛСТМ*) неуралне мреже за детекцију напада. Скуп напада у том раду укључује разне типове преплављивања мреже, али такође и скенирање као припремну фазу за напад и слање спама. Курниабуди и др. [55] су анализирали одабир одлика користећи информациону добит као критеријум, и креирали неколико класификационих модела машинског учења. Показује се да предикција модела зависи од одабира броја одлика, и да оптималан број одлика зависи од модела до модела. Значајни правац истраживања је био усмерен ка методама прикривања ботнета путем *DNS* протокола, попут алгоритма за генерисање домена и *DNS* флукса [7][56] [32][57][58]. Будући да анализирани *IoT* примерци малвера нису користили такве механизме, ове технике нису узете у разматрање.

Наведени радови из литературе у области детекције ботнета се могу поделити на основу начина екстракције података из мрежног саобраћаја. У претходно наведеним радовима [9][43] [44][45][48][50][55] подаци о мрежном садржају се екстрахују из целих пакета. Овај приступ даје велики број могућности за екстракцију одлика, али с друге стране изискује велику количину рачунарских ресурса. Као контраст томе, у радовима [46][47][49][51][53] је екстракција података вршена из самих токова. Како токови пролазе кроз рачунарску мрежу, тако се прикупљају њихове статистике. Овај приступ има предност у виду уштеде рачунарских ресурса, док је с друге стране скуп прикупљених одлика мали.

За разлику од претходно поменутих приступа, који су базирани на анализи садржаја пакета или анализи мрежних токова, *PI-BODE* систем користи потпуно оригиналан приступ [10]. Прикупљају се информације о мрежним токовима у облику временских низова који се формирају периодичним одабирањем одговарајућих бројача на мрежним уређајима. Из ових временских низова се статистичком анализом извлаче одлике и креира нови скуп података за детекцију командног и контролног канала. Овај скуп података је значајно мањи, омогућавајући уштеду приликом складиштења и времену процесирања, при чему обезбеђује у најмању руку исту прецизност детекције, што је показано у наставку овог поглавља. Статистичке величине токова се екстрахују користећи особине програмабилних елемената рачунарске мреже док су токови у оптицају, што омогућава детекцију ботнета док контролна комуникација још увек траје. Слично као и остала истраживања, *PI-BODE* користи технике машинског учења за детекцију ботнет фазе одржавања везе путем командног и контролног канала.

4.2 Анализирани малвер и скупови података у скорашњим истраживањима у области ботнета

Једна од кључних одлука у истраживању детекције малвера и упада је избор узорака малвера и скупова података који се користе за обуку и евалуацију алгоритама. Недавно истраживање [59] нагласило је неке проблеме са скуповима података који се често користе за истраживање. Неки од тих проблема укључују старост скупа података или услове под којима је саобраћај записан, што постаје ирелевантно због промена у обрасцима напада током година. Пример за то је *NSL-KDD* скуп података, који је записан пре више од двадесет година, а ипак се још увек користи у неким скорашњим истраживачким студијама [60][61][62]. Коришћење овог скупа података може се оправдати при истраживању волуметријских *DDoS* напада (код којих се жртва бомбардује огромним бројем пакета), који се нису много мењали током времена. Међутим, друге карактеристике ботнета, као и друге претње информационе безбедности, имају тенденцију да стално мењају свој образац понашања како би избегле детекцију, те је коришћење оваквих скупова података у таквим ситуацијама данас неоправдано.

Неке карактеристике, као што су скенирање, механизми ширења злонамерних програма или комуникација са серверима путем командног и контролног канала, доживљавају промене током времена [41]. За такве променљиве карактеристике, употреба застарелих скупова података даје резултате ограничене употребљивости. Осим старости скупова података, у раду

Ал-Хадрамија и др. [59] су истакнути проблеми са означавањем ових скупова података, нпр. информације о условима под којима су записани. Међутим, требало би истаћи још један проблем са курентним скуповима података који се често користе у научним радовима за ову врсту истраживања. Скупови података се записују током ограниченог времена и представљају тренутни приказ скупа претњи у том врло кратком временском интервалу, што не може дати реалан приказ о понашању одређене врсте ботнета, будући да неки ботнети мењају свој код и понашање од једне верзије до друге, како би избегли детекцију [41]. У Табели 1 су резимирани неки од скупова података коришћени у научним радовима за детекцију ботнета.

Табела 1: Скупови података у којима су присутни ботнети

Скуп података	Радови	Година снимања	Трајање снимања
NIMS	[63]	2014	1 дан
AWID	[62]	2015	9 дана
BotMark	[49]	2016	16 дана
ISOT	[44]	2017	7 дана
CIC-IDS	[55][60][62]	2017	5 дана
Bot-IoT	[44][46]	2018	6 дана, снимано у оквиру једног месеца

Како би се избегли модели који су претерано подешени на један специфичан скуп података и напад, потребно је да се скуп података креира током дужег периода и стално обнавља. Такође, истраживање би требало да се фокусира на налажење одлика које се ретко мењају, док би верификација модела машинског учења требало да се врши на различитим и најновијим скуповима података, сниманим током дужег временског периода. Због тога је у овом истраживању креиран нов скуп података (ETF-IoTB, Elektrotehnički fakultet-Internet of Things Botnet), базиран на записаним узорцима злонамерних програма снимљеним током прве фазе овог истраживања, а који је и јавно објављен и доступан за даљи научни рад и анализу. Такође, верификација је вршена добро познатим скупом података *IoT23* [64] Чешког техничког универзитета из Прага. Детаљи о скупу података су наведени у остатку овог одељка.

4.2.1 ETF-IoTB скуп података

Скуп података ETF-IoTB добијен је записивањем саобраћаја уређаја заражених злонамерним узорцима према методологији описаној у поглављу 3.1 [65]. Креирани скуп података испуњава неке препоруке о томе шта чини исправан скуп података о ботнету [66]: укључује

стварну комуникацију ботнета, а не симулацију, има непознат/регуларан саобраћај, има истинитосне лабеле за обучавање и оцењивање метода и укључује различите типове ботнета. Бенигни саобраћај је формиран као снимљени саобраћај једног рачунара са редовног радног места током дана.

Заражени уређаји били су стално надгледани како би се детектовале ситуације када долази до наглог повећања обима саобраћаја као индикација да заражени уређај учествује у DDoS нападу. Уређаји су обично радили у пре-нападној фази животног циклуса ботнета, када су регистроване иницијализација комуникације, као и комуникација путем командног и контролног канала, док су у неколико случајева били прекинути чим су били примећени DoS напади са заражених уређаја, како је описано у датом скупу података. Више детаља о скупу података и примећеним обрасцима контролне комуникације, као и иницирању команди за напад, наведено је у поглављу 3.2.

4.2.2 IoT23 скуп података

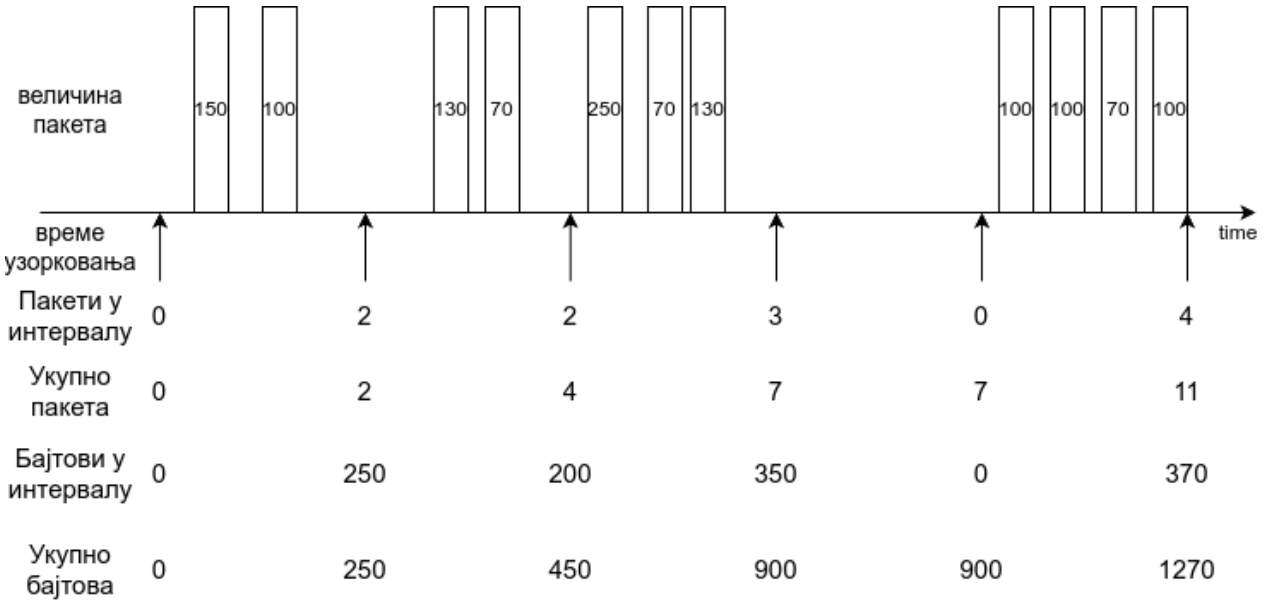
IoT23 скуп података је најновији скуп података који је представио Чешки технички универзитет у Прагу. Снимљен је у универзитетској лабораторији *Stratosphere lab*, као део пројекта *Aposemat*, спонзорисаног од стране антималвер компаније *Avast Software* [67]. Скуп података садржи означени бенигни и саобраћај ботнета сакупљен на IoT уређајима. IoT23 скуп података садржи 23 сценарија, од којих 20 садрже саобраћај злонамерних програма, док остала 3 садрже бенигни саобраћај. Узорци саобраћаја у овом скупу података су снимљени у периоду између 2018. и 2020. године. Злонамерни програми су такође снимљени на RaspberryPi уређајима, док су сценарији са бенигним саобраћајем снимљени на различитим IoT уређајима: Philips HUE паметној LED лампи, *Amazon Echo* паметном личном асистенту и *Somfy* паметној брави. IoT23 скуп података се састоји од 7 *Mirai*, 1 *Muhstick*, 1 *Kenjiro*, 2 *Torii*, 1 *IRCBot* и 1 *Gafgyt* узорка. Скупови података су означени и садрже 8 ознака повезаних са комуникацијом путем командног и контролног канала.

4.3 Анализа унутартоковских статистичких одлика

Унутартоковски статистички параметри могу се добити из фајлова насталих снимањем мрежних пакета на одређеном мрежном интерфејсу (*pcap* и *pcapng* фајлови) користећи софтверски алат попут *Joy* алата фирме *Cisco* [68], који је способан да издвоји неке унутартоковске одлике (нпр. вероватносна расподела величине пакета по мрежном току,

ентропија или Волш-Хадамар трансформација). За разлику од овог приступа који захтева велику количину простора на диску јер се статистичке особине извлаче из комплетног снимка свих пакета на неком линку, *PI-BODE* користи програмабилне мрежне уређаје и *OpenFlow* [69] протокол из области софтверски дефинисаних мрежа (енг. *Software Defined Networks,* скр. *SDN*) за прикупљање статистике у реалном времену. Уместо да захвати сваки пакет, PI-BODE систем путем *OpenFlow* протокола, узоркује са мрежног уређаја статистику параметара мрежних токова периодично и формира временске низове ових метрика из којих се касније израчунавају статистичке особине токова.

Слањем *OpenFlow* поруке OFPFlowStatsRequest периодично, *SDN* контролер захтева од свича укупан број пакета и укупну количину података за сваки мрежни ток. Периодичним узорковањем ових бројача и рачунањем разлике између резултата последњег и претпоследњег узорка, стварају се временске серије по мрежном току за број пакета и бајтова. Ове временске серије користе се као основа за прорачуне различитих статистичких одлика, које су описане у одељку 4.3.2. Слика 6 показује како се подаци о мрежном току прикупљају са програмабилног *SDN* свича.



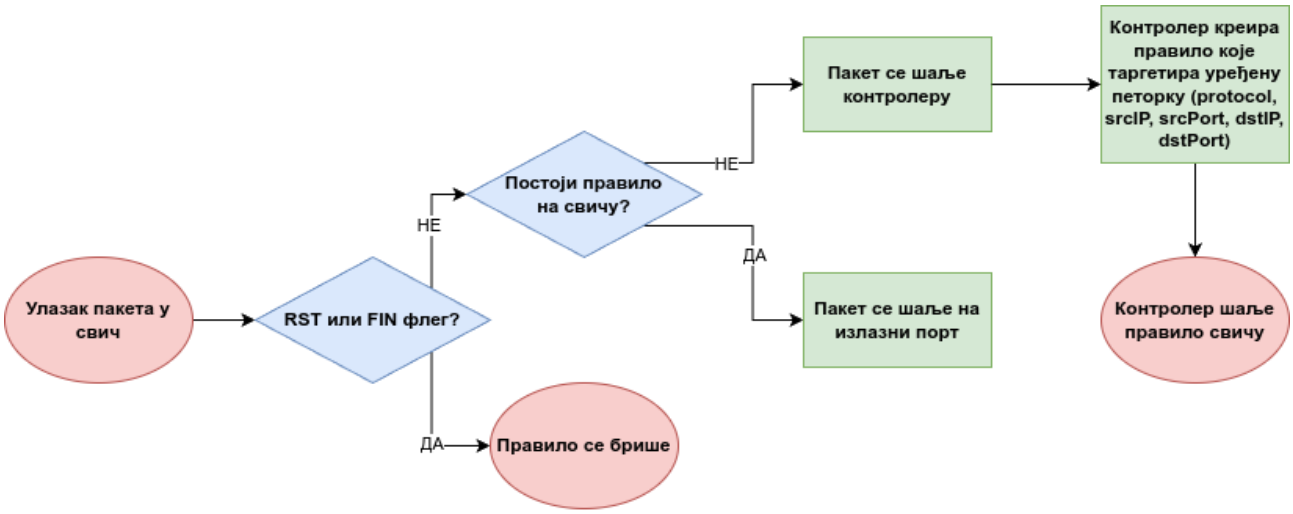
Слика 6: Прикупљање података о мрежном току

Два конфигурациона параметра која су коришћена су: период узорковања t_s и вредност времена неактивности мрежног тока t_{ia}. За први параметар t_s, коришћен је период од 1 секунде као општи референцни параметар, који је показао обећавајуће резултате за детекцију и малу мрежну оптерећеност. Разлог је тај да су периодични обрасци ботнет мрежних токова били такви да су најкраће забележене периоде биле реда 15 секунди. Финија грануларност

узорковања је могућа, али није се показала као сврсисходна, односно, тачност детекције се није побољшавала јер се нису бележиле никакве нове особине токова, а количина података који морају да се чувају са повећањем фреквенције неминовно расте. Избор другог параметра, $t_{ia,}$ је важан за мрежне токове који имају или дугачке периоде неактивности или не поседују експлицитну сигнализацију краја мрежног тока (нпр. *UDP* мрежни токови). Дуже задато време неактивности представља веће оптерећење ресурса јер истекли мрежни токови остају дуже у табели свича (до истека t_{ia}), али и једноставнију обраду на станици за анализу. Обрада је једноставнија зато што није потребно спајати делове мрежних токова са дугим периодима неактивности ако су подељени у више различитих мрежних токова. У свим експериментима, после анализе одабрана је вредност t_{ia} од 2 минута за време неактивности, јер су најдужи примећени периоди у комуникацији путем командног и контролног канала били 60 секунди, чиме је осигурано непрекидно снимање комуникације путем командног и контролног канала. Ови параметри такође дефинишу кључна ограничења система за детекцију *PI-BODE*. Лошији резултати детекције се могу очекивати ако се обрасци комуникације путем командног и контролног канала промене на начин који би их учинио тешко детектабилнима са изабраним скупом конфигурационих параметара (период узорковања и време неактивности). Периодично одржавање везе између бота и ботмастера за случај ботнет малвера са периодом краћим од периода узорковања или дужим од времена неактивности би били тешки за детекцију. Међутим, прво, у случају таквих промена, систем се лако може реконфигурисати да се прилагоди променама, а друго, као што ће бити показано у овој дисертацији, периодичност контролне ботнет комуникације је током целог периода праћења малвера била између 15 и 60 секунди и није се значајно мењала.

Једноставан алгоритам за прикупљање статистика о *TCP* мрежним токовима дат је на слици 7. За сваки *TCP* мрежни ток, кључни догађаји (први пакет и завршетак фазе иницијализације *TCP* сесије) узрокују додавање или брисање правила мрежног тока на свичу. *SDN* свич ради као транспарентни бриџ на слоју везе *TCP/IP* архитектуре, где се статистика пакета бележи на бази мрежног тока. Сваки нови мрежни ток, дефинисан као петорка (protocol - тип протокола , *srcIP* - *IP* адреса изворишта, *srcPort* - порт изворишта, *dstIP* - *IP* адреса одредишта, *dstPort* - порт одредишта), ствара нови унос у бази података свича, на основу којег се одређује прослеђивање пакета. Сваки мрежни ток има постављену `OFPFF_SEND_FLOW_REM` заставицу како би се прекидач натерао да пошаље сажете информације контролеру при уклањању мрежног тока. *UDP* или *TCP* мрежни токови са

периодима неактивности дужим од времена неактивности бивају уклоњени из базе података на свичу, након истека времена неактивности.



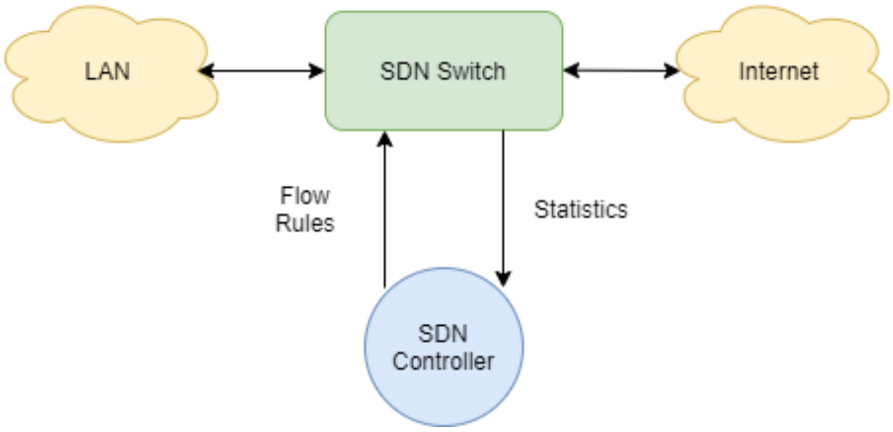
Слика 7: Алгоритам за прослеђивање пакета на SDN свичу

Унутартоковске серије података обезбеђују компактнији скуп података и значајно мање оптерећење у мрежи. Тачан добитак у количини пренетог саобраћаја са мрежног елемента потребан за детекцију напада је тешко оценити, јер зависи од различитих параметара у датом узорку мрежног саобраћаја као што су: дистрибуција дужине мрежних токова, број пакета по мрежном току, методологија завршетка мрежног тока и други. Илустроваћемо разлику користећи један пример који је извађен из *ETF-IoTB* скупа података [65]. Један од фајлова снимака пакета са незлонамерним саобраћајем садржао је укупно 1.474.536 пакета, што је створило 5.171 мрежних токова у укупној величини од 2.496 гигабајта. Трајање снимања пакета износило је 12.641 секунду, са просечно 116 пакета или 196 килобајта у секунди који су прослеђивани с мрежног елемента ка станици за анализу. Исти тај саобраћај, када су коришћени параметри узорковања мрежних токова од 1 секунде, створио је просечни саобраћај са мрежног елемента ка контролеру у опсегу између 685 и 6.979 бајтова у секунди за различите вредности времена неактивности мрежног тока између 5 и 600 секунди, што представља смањење у складиштењу и мрежном оптерећењу између 28 и 280 пута.

4.3.1 *PI-BODE* Систем

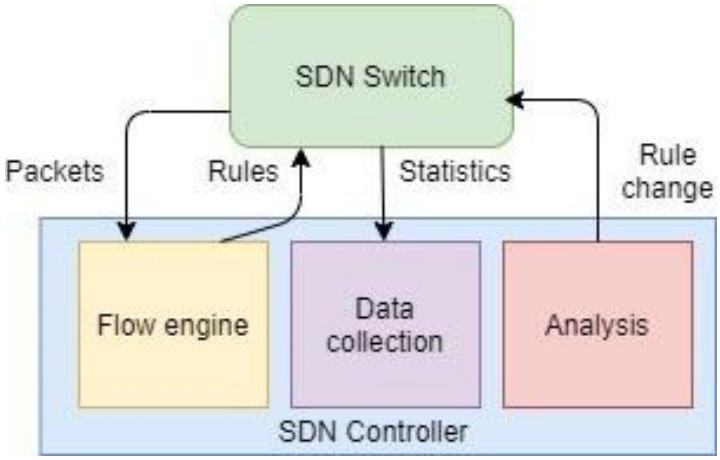
Пошто већина *IoT* уређаја нема довољно хардверских капацитета, софтвер на њима је често једноставан, а додавање додатних сигурносних мера на *IoT* уређајима није увек могуће због ограничених капацитета. Ово значи да сигурносне мере морају бити имплементиране на нивоу мреже, односно на мрежним уређајима, у овом случају на програмабилним *SDN*

свичевима. Оптимално решење је поставити уређај за детекцију напада на путањи пакета који теку између заштићених уређаја и интернета, самим тим дејствујући као систем за детекцију и спречавање упада у рачунарске системе (енг. *Intrusion Detection System,* скр. *IDS* и *Intrusion Prevention System,* скр. *IPS*) у исто време (Слика 8).



Слика 8: Схема система за детекцију ботнета

Софтвер контролера *PI-BODE* састоји се од три компоненте (приказане на Слици 9): нит за усмеравање мрежног саобраћаја (на слици Flow engine блок), нит за сакупљање података (на слици Data collection блок) и нит за анализу података (на слици Analysis блок). Нит за прослеђивање мрежног саобраћаја на основу пакета добијених од стране свича креира правила која служе за прослеђивање пакета кроз мрежу. Сваки мрежни ток (који обухвата податке из заглавља мрежног и транспортног слоја) има свој улаз у табели мрежних токова свича, али што се тиче прослеђивања пакета, свич врши основно прослеђивање на слоју везе. Нит за сакупљање података користи се за прикупљање броја пакета и бајтова за све мрежне токове сваке секунде, а сакупљени подаци се затим стављају у одговарајуће временске серије како је описано у Одељку 4.3.2.



Слика 9: SDN контролер

За све мрежне токове такође су израчунати укупно трајање мрежног тока, укупан број пакета и број уноса у емпиријској дистрибуцији. Ови параметри се не сматрају одликама мрежног тока које се користе у процесу детекције, већ се користе за рачунање других одлика. Додатно, трајање мрежног тока је и критеријум који одређује да ли ће мрежни ток бити употребљен у процесу детекције. Мрежни токови краћи од 60 секунди се не користе за детекцију ботнет комуникације путем командног и контролног канала. Неки ботови у склопу *ETF-IoTB* скупа података (на пример, xs.arm7 и nuclear.arm7) користе периодичну размену путем командног и контролног канала код које сваки пар пакета представља други мрежни ток (користи различите портове за сваки контакт). У овим случајевима, дефинисање мрежног тока као пар (*IP* адреса изворишта, *IP* адреса одредишта) може да захвати комуникацију путем командног и контролног канала, а самим тим и да направи табеле мрежних токова компактнијима. Међутим, у остатку истраживања рађено је са петоркама како је дефинисано на почетку Одељка 4.3. Разлог томе је случај када ботмастер успостави више различитих комуникација са ботом, где свака иде путем различитих парова портова. Уколико би се мрежни ток дефинисао за овај случај као пар (*IP* адреса изворишта, *IP* адреса одредишта), не би била приказана стварна слика комуникације између бота и ботмастера.

4.3.2 Унутартоковске одлике мрежног саобраћаја

Како је описано на почетку Одељка 4.3, за сваки мрежни ток, узорци броја пакета и бајтова стављени су у два временска низа, где се групишу сви бројеви бајтова и пакета у оквиру периоде од једне секунде. Из ових временских низова извучене су 22 елементарне статистичке одлике, које одговарају одликама које су коришћене у другој научној литератури приказаној у поглављу 4.1. Табела 2 садржи 14 одлика које ће бити анализиране у остатку овог рада, кратке описе и вредности површине испод криве (енг. *Area under the curve,* скр. *AUC*), чиме се показује проценат исправности детекције класа, уколико би се за детекцију користила само дата одлика.

Одлике 1-11 су изведене из две временске серије – низа добијеног из броја бајтова и низа добијеног из броја пакета. Одлике 12-14 су извучене из емпиријске расподеле времена t_{ia}. Време t_{ia} је број секунди где за одређени ток није било саобраћаја. Емпиријска расподела времена t_{ia} састоји се од броја појављивања различитих времена t_{ia} у временској серији броја бајтова. Разлог за стварање овакве дистрибуције је да се опажено понашање ботнет командног и контролног канала састоји од периодичних размена између командног и контролног канала и бота већи део времена. Стога, захватањем интервала неактивности

мрежног тока и анализом њихове дистрибуције, могуће је открити ово периодично понашање. Објашњење процеса прикупљања мрежног тока дато је у Одељку 3.1.

Табела 2: Опис прикупљених одлика

Број	Назив одлике	Опис одлике	AUC вредност
1	Просечан проток мрежног тока (bytes/s)	Број бајтова у секунди за временску серију, који се поново рачуна након што се сваки нови узорак дода временској серији.	0.96
2	Просечан број пакета у мрежном току (пакети/s)	Број пакета у секунди за временску серију, који се поново рачуна након што се сваки нови узорак дода временској серији.	0.93
3	Медијана протока временске серије (median)	Медијана за временске серије	0.553
4	Разлика просечних вредности протока	Разлика између просечних вредности протока у оба смера, за случај двосмерних *TCP/UDP* комуникација	0.738
5	Однос просечних вредности протока	Однос просечних вредности протока у оба смера, за случај двосмерних *TCP/UDP* комуникација	0.805
6	Стандардна девијација протока	Стандардна девијација протока за временске серије	0.863
7	Корелација (Corr)	Пирсонов коефицијент корелације између смерова у двосмерној *TCP/UDP* комуникацији	0.551
8	АДФ статистика	Статистика проширеног Дики-Фулер теста којим се испитује стационарност и постојање трендова [70]	0.511
9	П-вредност за АДФ тест	П-вредност проширеног Дики-Фулер теста [70]	0.607
10	Аутокорелација (Autocorr)	Аутокорелација временске серије која садржи број бајтова, која је израчуната за вредност помераја који одговара највећој вредности за вероватноћу у емпиријској расподели времена тишине мрежног тока. Ова одлика има за циљ да детектује периодично понашање мрежног тока	0.985
11	Ентропија	Ентропија временске серије протока, користећи формулу: $H(X)=-P(x_i)\ln(P(x_i))$ где је $P(x_i)$ вероватноћа да ће пакети бити одређене величине	0.767
12	Однос најбољи/остали (Best/Other ratio)	Однос вероватноћа између најчешће и свих осталих вредности у вероватносној расподели времена тишине мрежног тока	0.893
13	Однос најбољи/ други по реду (Best/Second ratio)	Однос вероватноћа између најчешће и друге по реду вредности у вероватносној расподели времена тишине мрежног тока	0.85
14	Најбоље две (Top Two)	Збир најчешће две вредности у вероватносној расподели времена тишине мрежног тока	0.849

4.4 Методологија ботнет детекције командног и контролног канала

Да би се открила комуникација путем командног и контролног канала која циља *IoT* уређаје, примењене су стандардне методе машинског учења, које су даље оптимизоване коришћењем радног процеса за науку о подацима (енг. *data science*) који укључује следеће кораке: екстракцију и обраду података, селекцију одлика, подешавање хиперпараметара и класификацију. Један корак који се често користи као део радног процеса за науку о подацима јесте трансформација података, тачније, нормализација и стандардизација [71]. Ови поступци

трансформишу опсег или дистрибуцију података за сваку одлику, што омогућава одређеним класификаторима да покажу боље перформансе. Међутим, постоји разлог зашто се не користи трансформација података у моделима машинског учења који се примењују у сигурносним системима. Нормализација и стандардизација се заснивају на трансформацији вредности одлика у одређеном скупу података, користећи статистичке особине сваке одлике. Циљ система за откривање је открити малициозни софтвер који није познат током фазе обуке. Са трансформисаним подацима, нови малициозни софтвер може имати одлике чије вредности излазе из ограничења нормализованог или стандардизованог скупа података који је коришћен за тренинг.

Одабир одлика (енг. Feature Selection) представља корак у радном процесу машинског учења, током којег се бира подскуп иницијално екстрахованих одлика, у циљу што бољег резултата радног процеса и смањења грешака прекомерног обучавања. Хиперпараметри представљају параметре коришћеног модела машинског учења. Оптимизација хиперпараметара је корак у радном процесу машинског учења који подразумева одабир вредности хиперпараметара који даје што боље перформансе. Класификација представља финални производ радног процеса машинског учења, где се оптимизовани модел тестира на тестном скупу, и бележи се скор као мера успешности модела.

4.5 Опис класификатора

Класификатори коришћени у овој фази истраживања су они који су коришћени у радовима који су се појавили до те фазе како би се макар приближно упоредиле тачности детекције са остатком литературе: *k*-најближих суседа (енг. *K-nearest neighbors*, скр. *KNN*), логистичка регресија и случајне шуме (енг. *random forest*). Тачно поређење са резултатима детекције других научних радова није сасвим могуће јер су врло ретко аутори остављали скупове података, подешавања хиперпараметара својих модела или код класификатора како би објективно поређење могло да се изврши. За имплементацију ових алгоритама коришћена је Python библиотека *sklearn* [72].

4.5.1 Алгоритам *k*-најближих суседа

Алгоритам *k*-најближих суседа је непараметарски метод који се користи за препознавање образаца [73]. Ово је уједно и најједноставнији алгоритам машинског учења. Свака тачка података (енг. *data point*) је представљена n-димензионалним вектором у простору одлика.

Фаза обуке алгоритма састоји се од чувања тачака података са ознакама. Фаза класификације почиње разматрањем позиције сваке нове тачке података у простору одлика. Затим, користећи метрику растојања, одређује се *k*-најближих суседа, и тачка података се класификује тако што се израчуна најзаступљенија класа међу суседима. Будући да се фаза обуке алгоритма састоји само од чувања тачака података са ознакама, за овај алгоритам се не може рећи да заиста тренира модел [74].

Дистанца се може рачунати на више начина, на пример:

$$d_{manhattan} = \sum_{i=1}^{n} |x_i - y_i| \quad \text{(Менхетн)} \tag{1}$$

$$d_{euclidean} = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \quad \text{(Еуклидова)} \tag{2}$$

$$d_{minkowski} = \left(\sum_{i=1}^{n} \left(|x_i - y_i|\right)^p\right)^{1/p} \quad \text{(Минковски)} \tag{3}$$

Где су *x*, *y* тачке података за које се рачуна дистанца, док је *n* број димензија тачке. Након што се израчуна који су то K-најближи суседи, приступа се класификацији, тако што сваки од суседа има неки утицај на класификацију. Суседи могу имати сви подједнак утицај на класификацију, те тачка бива смештена у класу у којој је већина суседа. Могуће је и овај утицај калибрисати тако да је дистанца суседа од тачке обрнуто пропорционална утицају, како би суседи који су ближи имали већи утицај на класификацију. Ова два метода су најчешћа за мерење утицаја *k*-најближих суседа на тачку коју треба класификовати. *sklearn* библиотека такође пружа кориснику могућност дефинисања сопствене методе за мерење утицаја суседа.

Кључна тачка овог алгоритма је метод како се проналазе суседи одређене тачке. Наравно, најочигледније решење је израчунавање дистанце тачке од свих тачака у скупу за тренирање, и проналажење *k* најмањих дистанци. Ова метода има квадратну сложеност, чинећи њено извршавање неефикасном. Стога су осмишљени алгоритми који користе бинарна стабала за претрагу суседа, наиме *KD Tree* [75] и *Ball Tree* [76].

KD Tree је бинарно стабло претраге по више димензија, имајући у виду ефикасно складиштење података [75]. Како само име ове структуре података каже, претрага се врши у *к* димензија. Сваки ниво стабла врши претрагу по другој димензији, почевши од корена стабла,

које врши претрагу по димензији 0, наредни ниво по димензији 1, и тако до *k-1* димензија. Потом, наредни ниво стабла врши поново претрагу по димензији нула. Вршење претраге овде значи дати одговор на питање да ли је вредност тачке по *к*-тој димензији већа или мања од вредности датог чвора. Након што се дође до листа, претрага се завршава. У сваком листу се налази уланчана листа тачака које испуњавају ове услове претраге, те се тражени чвор налази проласком кроз листу. Сложеност овог алгоритма је логаритамска, чинећи га ефикаснијим од претходно поменуте тривијалне методе. Овај алгоритам се показао ефикасним за проблеме малих димензија (мањих од 20), док за већи број димензија се показује неефикасним [72].

Ball Tree је структура података која је створена како би се исправио проблем *KD Tree* структуре за случај већег броја димензија [76]. Идеја ове структуре података је да третира *n* димензија тачке као димензије у Декартовом простору. Сваки чвор представља *n*-то димензиону сферу, док су њени листови сфере које она садржи. У сваком чвору је садржан низ од *к* бројева, као и величина пречника сфере, док листови представљају скуп тачака које садржи та сфера. Претрага се врши тако што се бира сфера чији центар садржи тачку која се тражи. Унос у ову структуру података је захтевнији него код *KD Tree*, но претрага је ефикаснија у већини случајева.

У *sklearn* библиотеци, алгоритам *k*-најближих суседа има следеће параметре:

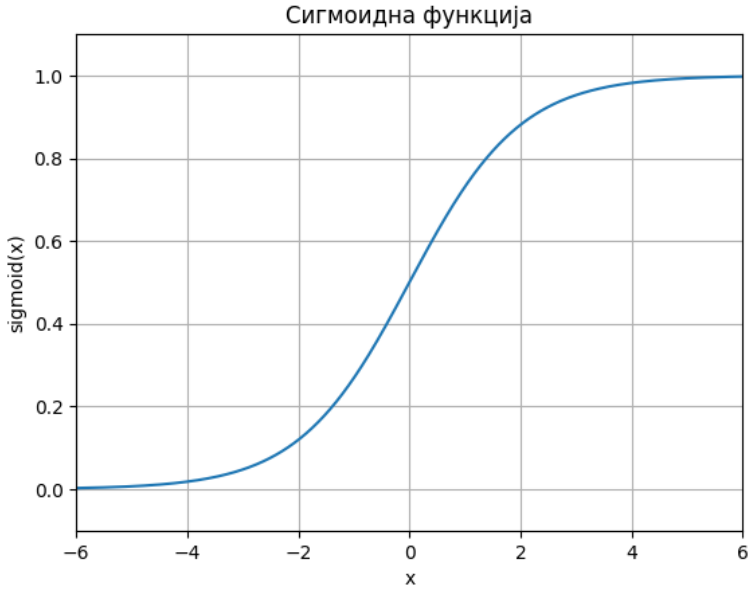
- *n_neighbors* – број суседа који имају утицај на класификацију тачке
- *weights* – принцип пондерисања утицаја суседа
- *algorithm* – алгоритам који се користи за претрагу тренинг скупа тачака
- *leaf_size* – овај параметар се користи за случајеве *KD Tree* и *Ball Tree* алгоритама, и представља максималан број тачака који лист ових стабала може да садржи
- *p* – за случај када се користи метрика Минковског, овај параметар дефинише степен у формули за дистанцу
- *metric* – метрика која се користи за дистанцу. Може бити и посебно дефинисана метрика
- *metric_params* – за случај када је метрика посебно дефинисана, овај параметар садржи додатне аргументе, уколико су потребни, за такву метрику

- *n_jobs* – овај параметар представља број језгара процесора који се користи за паралелизацију алгоритма. Вредност од -1 значи да се користе сва језгра процесора.

Хиперпараметри који су разматрани за *KNN* класификатор су следећи: број суседа (*n_neighbors*), тежинска функција (*weights*), алгоритам за претрагу суседа (*algorithm*), и метрика растојања (*metric*), док су хиперпараметри који дају најбоље резултате класификације дати у табели 4.

4.5.2 Логистичка регресија

Овај модел машинског учења је дискретна верзија линеарне регресије. Логистичка регресија користи сигмоидну функцију како би класификовала тачке у *n*-то димензионом простору [77]. Погодна особина ове функције јесте што мапира цео реални скуп бројева на интервал од 0 до 1. Сигмоидна функција је приказана на слици 10.



Слика 10: Сигмоидна функција

Главне одлике ове функције су да тежи 0 како *x* тежи -∞, док тежи 1 како *x* тежи ∞, док њена вредност у нули износи 0.5. Формула сигмоидне функције је:

$$sigmoid(x)=\frac{1}{1+e^{-x}} \tag{4}$$

Користећи репрезентацију тачке података као *n*-димензионалних вектора означених са x_i, полиномијалних реалних коефицијената означених са β_i и вредности грешке означене са ε, унос у сигмоидну функцију има следећи облик:

$$f(x)=\beta_1 x_1+\beta_2 x_2+\ldots+\beta_n x_n+\beta_0+\varepsilon \tag{5}$$

Вредност вероватноће се налази између 0 (тачка података сигурно припада класи *0*) и 1 (тачка података сигурно припада класи *1*). Класе *0* и *1* се дефинишу пре класификације. Током фазе тренинга, свака тачка података се уноси у сигмоидну функцију, прилагођавајући коефицијенте и вредност грешке тако да се постигне најбоље прилагођавање. Суштина подешавања коефицијената на основу скупа за тренинг јесте да се они подесе тако да вредност сигмоидне функције за тачке класе 0, односно класе 1, буду што ближе вредности 0, односно 1. Стога се коефицијенти подешавају тако да се минимизује грешка, која се дефинише као средња квадратна грешка. Ова грешка мери просечну вредност квадрата разлике између тачне и предвиђене вредности. Формула средње квадратне грешке гласи:

$$J(w)=\frac{1}{m}\sum_{i=1}^{m}\frac{1}{2}(\hat{y}(x_i)-y_i)^2 \tag{6}$$

Где скуп за тренинг садржи m парова (x_i, y_i), где x_i представља n-то димензиону тачку података, а y_i је класа којој та тачка припада. $\hat{y}(x_i)$ представља вредност која је на излазу класификатора за тачку x_i. Проблем ове методе је што теже конвергира глобалном оптимуму, те се у пракси користи модификована функција грешке.

$$J(w)=\frac{1}{m}\sum_{i=1}^{m}-[y_i\log(\hat{y}(x_i))+(1-y_i)\log(1-\hat{y}(x_i))] \tag{7}$$

Ова функција има вредност нула када је предикција тачна, а све већу вредност како се предикција одмиче од тачне вредности.

У зависности од алгоритма који се користи за минимизацију функције грешке, може се додати сабирак, са циљем да се смањи варијанса код комплекснијих модела [74]. Овај поступак се назива регуларизација модела. Постоје три главне врсте регуларизације: L1 (која се назива и ласо регуларизација), L2 (која се назива и гребен регуларизација) и *ElasticNet*. *ElasticNet* представља комбинацију прва два облика регуларизације. Формуле за ове типове регуларизације су:

$$a\sum_{j=1}^{n}|\beta_j|, 0<a\leqslant 1 \tag{8}$$

$$a\sum_{j=1}^{n}\beta_j^{\,2}, 0<a\leqslant 1 \tag{9}$$

$$a\sum_{j=1}^{n}|\beta_j|+b\sum_{j=1}^{n}\beta_j^{\,2}, 0<a\leqslant 1, 0<b\leqslant 1, a+b=1 \tag{10}$$

Начин на који се минимизује функција грешке зависи од алгоритма који се користи. Постоји пет главних алгоритама за логистичку регресију: *LIBLINEAR*[78], *LBFGS*[79], *Newton-CG*[80], *Newton-Cholesky*[81], *SAG*[82], *SAGA*[83]. Сваки од ових алгоритама може користити одређене врсте регуларизације, што је описано Табелом 3.

Табела 3: Алгоритми за логистичку регресију и њихове регуларизације

Алгоритам	Регуларизације
LIBLINEAR	L1, L2
LBFGS	L2
Newton-CG	L2
Newton-Cholesky	L2
SAG	L2
SAGA	L1, L2, ElasticNet

У *sklearn* библиотеци, алгоритам логистичке регресије има следеће параметре:

- *penalty* – овај параметар представља облик регуларизације који може бити урачунат
- *dual* – параметар који означава да ли је формулација проблема ограничена или регуларизована
- *tol* – толеранција, вредност која означава минимално побољшање функције грешке које се узима у обзир
- *C* – јачина регуларизације, коефицијент који стоји испред сабирка који представља регуларизацију
- *fit_intercept* – овај параметар означава да ли ће се користити сабирак β_0 као додатна променљива за сигмоидну функцију
- *intercept_scaling* – овај параметар фиксира вредност β_0

- *class_weight* – овај параметар дефинише посебне тежинске коефицијенте за различите класе
- *random_state* – вредност *seed* који користе неки решавачи
- *solver* – параметар који дефинише који се решавач користи за подешавање коефицијената
- *max_iter* – максимални број итерација решавача
- *multi_class* – како се третира проблем класификације када је број класа већи од 2
- *verbose* – дефинише испис на конзолу
- *warm_start* – дефинише да ли се решење из претходног позива решавача користи као почетно решење новог позива функције фитовања модела
- *n_jobs* – овај параметар представља број језгара процесора који се користи за паралелизацију алгоритма. Вредност од -1 значи да се користе сва језгра процесора.
- *l1_ratio* – овај параметар се користи у случају *ElasticNet* регуларизације, и дефинише који удео тога чини L1 регуларизација

Хиперпараметри који су разматрани за логистичку регресију су следећи: јачина регуларизације (*C*), принцип пондерисања класа (*class_weight*), опција додавања константе у функцију одлучивања (*fit_intercept*), максималан број итерација (*max_iter*), решавач (*solver*), и толеранција за критеријум заустављања (*tol*), док су хиперпараметри који дају најбоље резултате класификације дати у табели 4.

4.5.3 Алгоритам случајне шуме

Случајне шуме припадају класи модела који се зову ансамбл модели. У суштини, ансамбл метод комбинује многе једноставне или слабе класификаторе, а класификација се врши коришћењем већинског гласања. Основни класификатор у овом методу је стабло одлучивања, које се састоји од чворова и листова. Сваки чвор садржи граничну вредност за одлику, док листови садрже информације о томе којој класи одређена тачка податка припада. Избором подскупа одлика, као и рангирањем функције, одлике се рангирају и стабло одлучивања се конструише. Ово се постиже стављањем одлика као граничних вредности унутар стабла одлучивања, крећући се од одлике са највишим рангом до најнижег. Случајне шуме се састоје од одређеног броја случајних стабала, а класа за тачку податка се одређује већинским

гласањем. Ова класа алгоритама се показала најбољом за табеларне податке, те се очекивало да алгоритам случајне шуме да најбоље резултате [84][85].

Стабло одлучивања се конструише по следећем принципу: потребно је одредити које одлике највише утичу на класификацију. На основу критеријума одабира (ентропија, Џини коефицијент, или логаритамски губитак), израчунава се утицај сваке од одлика на класификацију, и одређују границе које деле скуп података на мање подскупове. Формуле за ентропију, Џини коефицијент и логаритамски губитак су:

$$H(X)=\sum_{i=1}^{K} p(i)\log(p(i)) \tag{11}$$

$$G(X)=1-\sum_{i=1}^{K} p_i^2 \tag{12}$$

$$L_{\log}(Y,P)=-\log(Y|P)=-\frac{1}{N}\sum_{i=0}^{N-1}\sum_{k=0}^{K-1} y_{i,k}\log(p_{i,k}) \tag{13}$$

Где је K број класа у скупу података, N је број примерака, $y_{i,k}$ је израчуната припадност узорка i класи k, а $p_{i,k}$ је вероватноћа да узорак i припада класи k.

Све одлике се сортирају по вредностима критеријума одабира, од бољих вредности ка горим. За сваку од одлика се потом одређује граница којом се скуп података раздваја на мањи подскуп. Потом, почевши од одлике са највећим скором, у стабло одлучивања се додају чворови који су дефинисани овим одликама, формирајући све веће стабло. Изградња стабла одлучивања престаје онда када се у листовима стабла налазе примерци само једне класе. Претрага у стаблу одлучивања почиње од корена, и, на основу вредности одлика, долази се до листа, на основу чега се одређује припадност одговарајућој класи. Постоји више алгоритама за изградњу стабла одлучивања [77]:

- Итеративни дихотомизатор 3 (енг. *Iterative Dichotomiser 3,* скр. *ID3*): Овај алгоритам користи похлепни (енг. *greedy*) алгоритам претраживања с врха надоле, тако што у свакој итерацији бира најбољу одлику за дељење скупа података.
- *C4.5*: Унапређена верзија ID3 која уводи враћање. У овом случају алгоритам пролази кроз конструисано стабло и замењује гране са листовима ако се тиме побољшава чистоћа.

- Класификационо и регресионо стабло (енг. *Classification and Regression Tree, CART*): Конструише стабло користећи бинарно дељење, поштујући ограничења у погледу минималног броја узорака за нови чвор и максималне дубине стабла.
- Хи-квадрат аутоматски детектор интеракције (*CHAID*): Овај алгоритам често се користи у директном маркетингу. Укључује компликоване статистичке концепте, али у основи одређује оптималан начин спајања предиктивних променљивих како би се најбоље објаснио исход.

Предност стабла одлучивања је што могу да веома добро опишу цео скуп података над којим су изграђена. Али, ово са собом повлачиману прекомерног обучавања модела на том скупу података, чинећи га неупотребљивим за нове податке.

Због проблема прекомерног обучавања стабла одлучивања, почело се размишљати о комбиновању више стабала одлучивања, која заједничким гласањем врше предикцију над скупом података. Приликом изградње сваког стабла, оно добија насумични подскуп скупа за тренинг, као и насумични скуп одлика на основу којих треба да изврши поделу. Сврха овог поступка је конструкција стабала одлучивања која ће међу собом имати ниску корелацију у погледу доношења одлука у класификацији, како би се што боље објаснила варијанса у скупу података и избегло прекомерно обучавање.

У *sklearn* библиотеци, алгоритам случајне шуме има следеће параметре:

- *n_estimators* – број стабала одлучивања у случајној шуми
- *criterion* – критеријум за изграђивање стабла одлучивања
- *max_depth* – максимална дубина стабла одлучивања
- *min_samples_split* – минимални број узорака који може поделити унутрашњи чвор стабла одлучивања
- *min_samples_leaf* – минимални број узорака који може бити садржан у листу стабла одлучивања
- *min_weight_fraction_leaf* – минимална вредност суме тежинских коефицијената узорака у листу стабла одлучивања
- *max_features* – максимални број одлика који се користи
- *max_leaf_nodes* – максимални број листова у стаблу одлучивања

- *min_impurity_decrease* – минимална вредност за критеријум одлучивања која је потребна да би се поделио чвор у стаблу одлучивања
- *bootstrap* – овај параметар говори о томе да ли се цео скуп података користи за изградњу стабла одлучивања или не
- *oob_score* – уколико се не користи цео скуп података, овај параметар говори да ли се узорци који нису коришћени за изградњу одређеног стабла одлучивања користе за процену његовог скора
- *n_jobs* – овај параметар представља број језгара процесора који се користи за паралелизацију алгоритма. Вредност од -1 значи да се користе сва језгра процесора.
- *random_state* – овај параметар представља *seed* који се користи у случају када се изградња стабла одлучивања врши на основу дела скупа података
- *verbose* – дефинише испис на конзолу
- *warm_start* – дефинише да ли се решење претходног позива решавача користи као почетно новог позива функције калибрисања модела
- *class_weight* – принцип пондерисања класа приликом изградња стабла одлучивања
- *ccp_alpha* – како би се смањила вероватноћа да се претерано калибрише стабло одлучивања, примењује се алгоритам сечења стабла да би се смањила његова сложеност. Овај параметар дефинише коефицијент комплексности за стабло одлучивања
- *max_samples* – максимални број узорака за тренирање стабла одлучивања

Хиперпараметри који су разматрани за класификатор случајне шуме су следећи: тежине класа (*class_weight*), критеријум за изградњу стабла одлучивања (*criterion*), максималан број одлика (*max_features*), и укупан број естиматора (*n_estimators*), док су хиперпараметри који дају најбоље резултате класификације дати у табели 4.

4.5.4 Одабир одлика

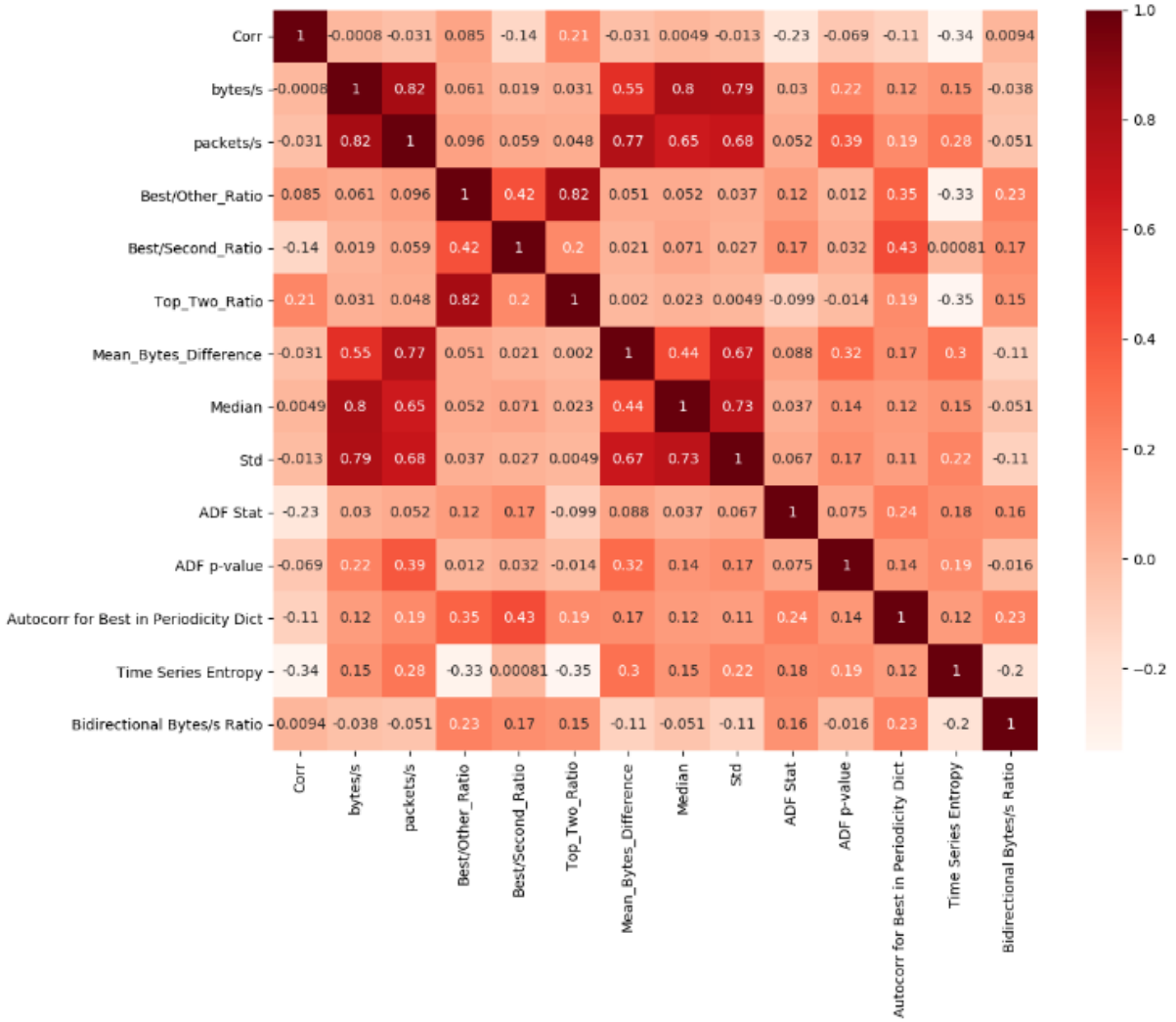
Смањење димензионалности је важан корак како би се створио класификатор прилагодљив за имплементацију у мрежним елементима у смислу меморије и ангажовања процесорске снаге, посебно када се очекује рад у реалном времену. За сврху одређивање утицаја појединачних одлика на класификацију, користи се површина испод криве (енг. Area Under the Curve, AUC),

која представља површину испод графика пријемне операционе одлике (енг. *Receiving Operating Characteristic*, ROC).

ROC је графикон који, за сваку вредност одлике, показује однос правих позитивних (енг. *true positive ratio, TPR*) и однос правих негативних (енг. *true negative ratio, TNR*). Број стварно позитивних и број стварно негативних означени су као TP (енг. *true positive, TP*) и TN (енг. *true negative, TN*), док су број лажно позитивних и број лажно негативних означени као FP (енг. *false positive, FP*) и FN (енг. *false negative, FN*). TPR и TNR се рачунају на следећи начин:

$$TPR = TP / (TP+FN),\ TNR = 1 - Specificity = 1 - TN / (TN + FP). \quad (14)$$

ROC је крива, а вредност испод површине криве (тј. AUC) показује колико добро одређена одлика правилно раздваја податке у класе. Вредности AUC се налазе у интервалу [0,1], где вредност ближа јединици указује на бољу оцену одлике. Одлике са вредностима AUC већим или једнаким 0.85 су аутокорелација, пропусност (бајтови/с), пакети/с, однос најбољи/остали, стандардна девијација, однос најбољи/други најбољи. Међутим, одлике које се бирају за процес класификације не би требале бити међусобно корелисане, јер то значи да описују исти или сличан скуп података. Зато је корелациона матрица коришћена за испитивање корелације између свих парова одлика. Корелациона матрица је приказана на слици 11. Велика корелација (већа од 0,8) забележена је између одлика просечни проток (бајтови/с) и просечни број пакета у секунди. Због тога је пет одлика одабрани за улазне одлике за процес класификације: аутокорелација, просечни проток (бајтови/с), однос најбољи/остали, стандардна девијација протока и однос најбољи/други најбољи.



Слика 11: Матрица корелације за прикупљене одлике

4.5.5 Хиперпараметри класификатора

Сваки класификатор има скуп хиперпараметара који дефинишу његово понашање. У овој фази анализирамо комбинацију хиперпараметара класификатора која максимизује оцене коришћене у класификацији помоћу методе *fit* и *score* који је имплементиран методом *GridSearchCV* [72] у *sklearn* библиотеци програмског језика Python. Метод је благо модификован како је описано у наставку да би се подударао са анализираним небалансираним скуповима података. Скуп оцена коришћених у овој дисертацији су прецизност (енг. *precision, PREC*), одзив (енг. *recall, REC*) и F1-мера (*F1*) [73]. Ове оцене су дате у следећим једначинама:

$$PREC = TP / (TP + FP), \quad (15)$$

$$REC = TP / (TP + FN),$$

$$F1 = 2 \cdot PREC \cdot REC / (PREC + REC).$$

Прецизност је мера која показује колико лажних позитивних постоји током класификације, док је одзив мера која показује колико лажних негативних постоји током класификације. F1-мера представља меру која има за циљ да анализира баланс између исправности и покривености при класификацији позитивних инстанци [86], пошто укључује и прецизност и повратак.

Иако је тачност (енг. a*ccuracy*) интуитивнија метрика, ове три мере су изабране због разлике у величинама нормалних и ботнет токова која је већа од 20 процената, што се сматра неизбалансираним скупом података. У случају небалансираних скупова података, прецизност као мера не би дала тачан резултат, јер не би исправно приказала каква је предикција за случај примерака мањинске класе. Чак и високе вредности тачности могу донети слабе стопе класификације за мањинску класу и класификатори могу имати низак потенцијал предвиђања [87]. Разлог томе је што сваки глобални скор не узима у обзир скор за сваку појединачну класу, те самим тим таква мера не говори о успешности модела машинског учења. Мора се напоменути да код избалансираних скупова података овај проблем не постоји. Скупови података о ботнет комуникацији путем командног и контролног канала су небалансирани зато што је ботнет комуникација путем командног и контролног канала релативно ретка појава у мрежном саобраћају у поређењу са свим осталим саобраћајем и обично производи само један или мали број токова одједном. Проблем небалансираности класа је веома карактеристичан у области детекције мрежних упада јер упади, а посебно контролна комуникација чине веома мали део укупног саобраћаја. Овај проблем и један предлог решења је обрађен у раду [88].

Балансирано узорковање је коришћено да би се подаци поделили на подскупове за обучавање и тренирање у односу 90/10. Подаци су подељени на такав начин како би се задржао однос класа из скупа података, што се зове стратификовано узорковање. Сличан приступ је представљен у [89], како би се обухватила небалансираност класа у скупу података. Након поделе скупа података, модел се обучава и тестира, а оцене се издвајају. Овај поступак се понавља 100 пута, а средње вредности се израчунавају за све оцене (Monte Carlo унакрсна валидација [90]). Овај приступ су аутори применили како би имали велики број варијација тренинг и тест сценарија, али и како би се минимизовао број тестираних сценарија. Поново, јер је скуп података неизбалансиран, овај метод је пригоднији од опште унакрсне валидације са k скупова (енг. *k-fold cross-validation*), која дели цео скуп података на k једнаких делова. Такав приступ чини да неки делови скупа имају ниску присутност ботнет података. Након израчунавања свих оцена, претражује се листа комбинација параметара за оне који имају

максималне вредности за све оцене. У Табели 4 су дате вредности за хиперпараметре за скупове података ETF-IoTB и IoT23.

Табела 4: Вредности хиперпараметара за коришћене алгоритме и скупове података

Алгоритам	Хиперпараметар	ETF-IoTB	IoT23
k-најближих суседа	n_neighbors	5	5
	class_weight	uniform	uniform
	algorithm	ball_tree	ball_tree
	distance	manhattan	manhattan
Логистичка регресија	C	0.25	0.25
	class_weight	None	None
	fit_intercept	false	false
	max_iter	100	100
	solver	newton-cg	lbfgs
	tol	0.001	10^{-5}
Случајна шума	class_weight	balanced_subsample	balanced_subsample
	criterion	entropy	entropy
	max_features	None	None
	n_estimators	200	200

4.5.6 Евалуација класификације PI-BODE методе

Овај одељак приказује и разматра резултате класификације скупа података формираног у другој фази истраживања за три класификатора: класификатор *k*-најближих суседа, логистичку регресију и алгоритам случајне шуме, користећи вредности хиперпараметара одређене у претходном одељку. Прецизност, повратак и F1-мера за PI-BODE метод детекције дате су у Табели 5 и 6 за ETF-IoTB и IoT23 скуп података, респективно. Сви резултати се односе на тестни скуп.

Табела 5: Резултати за ETF-IoTB скуп података

Модел	Ботнет саобраћај прецизност	Ботнет саобраћај одзив	Ботнет саобраћај F1-мера	Регуларни саобраћај прецизност	Регуларни саобраћај одзив	Регуларни саобраћај F1-мера
KNN	**0.9363**	0.9141	**0.9218**	0.9967	**0.9973**	**0.9970**
Логистичка регресија	0.9131	**0.9247**	0.9150	**0.9971**	0.9962	0.9967
Случајна шума	0.5954	0.6240	0.5994	0.9859	0.9948	0.9902

Табела 6: Резултати за IoT23 скуп података

Модел	Ботнет саобраћај прецизност	Ботнет саобраћај одзив	Ботнет саобраћај F1-мера	Регуларни саобраћај прецизност	Регуларни саобраћај одзив	Регуларни саобраћај F1-мера
KNN	**0.8683**	0.8759	**0.8690**	**0.8558**	0.8382	**0.8423**
Логистичка регресија	0.8347	**0.8778**	0.8510	0.8522	0.7858	0.8093
Случајна шума	0.5564	0.6071	0.5780	0.7566	**0.8733**	0.7724

Из ових резултата може се извући неколико закључака. Прво, високе вредности резултата на оба скупа података доказују да се овај метод може користити за откривање ботнет комуникације путем командног и контролног канала са високом прецизношћу. Друго, иако је селекција одлика вршена на скупу података *ETF-IoTB*, резултати методе детекције на *IoT23* скупу података који садржи различите узорке малвера (*Muhstik*, *Kenjiro*, *Torii*, поред оних у *ETF-IoTB* скупу) показују сличну прецизност у поређењу са другим истраживачким радовима. Може се закључити да су кључне одлике овакве врсте комуникације у различитим узорцима малвера сличне у оба скупа података. Такође, обучавање модела који користи одлике добијене из статистике унутар тока података може открити комуникацију путем командног и контролног канала узорака малвера који нису у скупу података коришћеном за обуку модела, па је детекција малвера нултог дана могућа. Треће, резултати показују да су за оба скупа података, *KNN* и Логистички регресиони класификатори дали најбоље резултате, уз мале разлике између оцена прецизности и повратка. То значи да постоји мала разлика између броја лажно негативних и лажно позитивних случајева, што показује стабилност детекције. Конзистентност у перформансама класификатора показује да предложени метод добро функционише без обзира на скуп података који се користи. С једне стране, *KNN* класификатор показује нешто бољу оцену од Логистичког регресионог класификатора, док са друге стране, *KNN* има много већу употребу меморије и није практичан за употребу у мрежним елементима. Будући да би ове моделе требало имплементирати у сигурносни систем, препоручује се коришћење Логистичког регресионог класификатора, јер даје готово најбоље оцене међу свим класификаторима, уз одржавање употребе меморије на нижем нивоу.

Радови референцирани нумерацијама [47][48][49], слично приступу описаном у овој дисертацији, тражили су да открију ботнете и ренсомвере преко комуникације путем

командног и контролног канала. У раду [47], истраживање је показало тачност од 80% у класификацији периодичности ботнета, што је ниже од оцена које су добијене у овом истраживању за најбоље класификаторе. Табеле 7 и 8 приказују резултате истраживања представљених у радовима [48] и [49], респективно. У раду [48], систем детекције *BotMark* показао је високу тачност (0.9994) и ниску оцену лажно позитивних. Међутим, пошто су F1-мере ниске за све претходне истраживачке методе (0.115 за *BotMark*, упоређено са 0.915 и 0.85 за PI-BODE Логистички регресиони класификатор) док је оцена лажно позитивних такође ниска, ово значи да је оцена лажно негативних висока, што додатно подразумева да је стабилност класификатора *BotMark* ниска и свакако много нижа него код PI-BODE.

Табела 7: Резултати детекције ботнета за рад [48]

Модел	F1-мера	Прецизност	TPR	FPR
Similarity	0.087247	0.9904	0.9836	0.009645
Stability	0.060732	0.9868	0.9115	0.013181
C-flow	0.152788	0.9949	0.9836	0.005108
Graph	0.034811	0.9166	0.9836	0.083478
BotMark	0.115207	0.9994	0.9836	0.000641

Аутори у [49] користе тачност, одзив и F1-меру као оцене за свој класификатор. У оба случаја представљена у [49], са 8 и 28 одлика, вредности оцена се крећу у опсегу од 0.83 до 0.89. Ово представља лошије резултате класификације у односу на *PI-BODE* чије се оцене крећу у опсегу од 0.914 до 0.936 за класификатор *k*-најближих суседа и класификатор на бази логистичке регресије када се систем обучава, верификује и тестира на скупу података *ETF-IoTB*. Чак и када се класификатор обучи на скупу података *ETF-IoTB* и тестира на скупу података IOT23 (случај откривања нових претњи), оцене се крећу у опсегу од 0.83 до 0.88, што је упоредиво са оценама добијеним у [49]. Такође, важно је напоменути да оба ова рада приказују само оцене у вези са целим скупом података. У случају балансираних скупова података, оцене могу бити приказане на овај начин. Међутим, пошто злонамерни софтвер представља мали део саобраћаја на мрежи, такви скупови података природно су дисбалансирани. Оцене представљене на овај начин треба третирати са опрезом, пошто нема информација о перформансама машинског учења и на нормалном и злонамерном саобраћају. На крају, детекција комуникације злонамерног софтвера за отежавање описана у [49] базира

се на анализи пуног снимка пакета, што значи да има значајно веће мрежне и складишне захтеве.

Табела 8: Резултати детекције ботнета у односу на број одлика, приказани у раду [49]

Број одлика	F1-мера	Прецизност	TPR
28	0.83	0.89	0.86
8	0.86	0.87	0.87

4.6 Закључак

У овој фази истраживања је предложен, реализован и евалуиран оригиналан метод PI-BODE за откривање комуникације ботнета путем командног и контролног канала заснован на SDN и унутартоковским статистикама. Он омогућава креирање скупа података са више статистичких одлика мрежних токова него методе засноване само на *NetFlow*, *IPFIX* и сличним протоколима, али не користи анализу свих пакета на вези. Овакав метод представља компромисно решење које штеди меморију и ресурсе процесора за до два реда величине, али ипак пружа довољно детаља о оствареним комуникацијама на мрежи који омогућавају разликовање злонамерног и нормалног саобраћаја. Експериментални резултати су показали да PI-BODE постиже исту или већу тачност откривања ботнета као и слични системи који користе анализу целокупног саобраћаја уз значајно мање режијске трошкове.

Још неки закључци се могу извести из приказаних истраживања. Прво, резултати доказују да се комуникација ботнета са ботмастером може открити користећи статистике унутар тока и предложене одлике саобраћаја. Друго, с обзиром на то да је обука и процена PI-BODE вршена на скуповима података који садрже различите обрасце злонамерног софтвера, показује се да различите верзије злонамерног софтвера имају сличне карактеристике комуникације путем командног и контролног канала и да нови обрасци злонамерног софтвера могу бити откривени системом обученим на старијим обрасцима злонамерног софтвера. Треће, коришћење SDN контролера као алата за прикупљање статистика унутар тока омогућава стварање једноставног система за спречавање улаза за уређаје који немају заштиту, као што су IoT уређаји. Користећи PI-BODE приступ као основу, трећа фаза истраживања ће истражити додатне статистичке параметре временских низова добијених унутартоковском анализом, анализирати друга динамичка својства злонамерног софтвера која се могу

користити за откривање и разматрати стратегије обучавања модела у светлу промене начина рада IoT ботнет малвера. Истраживање приказано у овој дисертацији може бити проширено коришћењем других напредних алгоритама машинског учења, као што су модели подршке (енг. *support models*) и модели дубоког учења (енг. *deep learning models*).

5 Детекција новијих варијанти ботнет CnC комуникације (2022-2023)

У претходним фазама истраживања показано је да коришћење унутартоковских статистичких параметара за детекцију ботнета даје резултате једнаке тачности као методе детекције које користе статистичке параметре изведене из потпуних снимака свих пакета на линку коришћењем истих алгоритама машинског учења, али уз значајно мање режијске трошкове. У наредној фази истраживање је усмерено према:

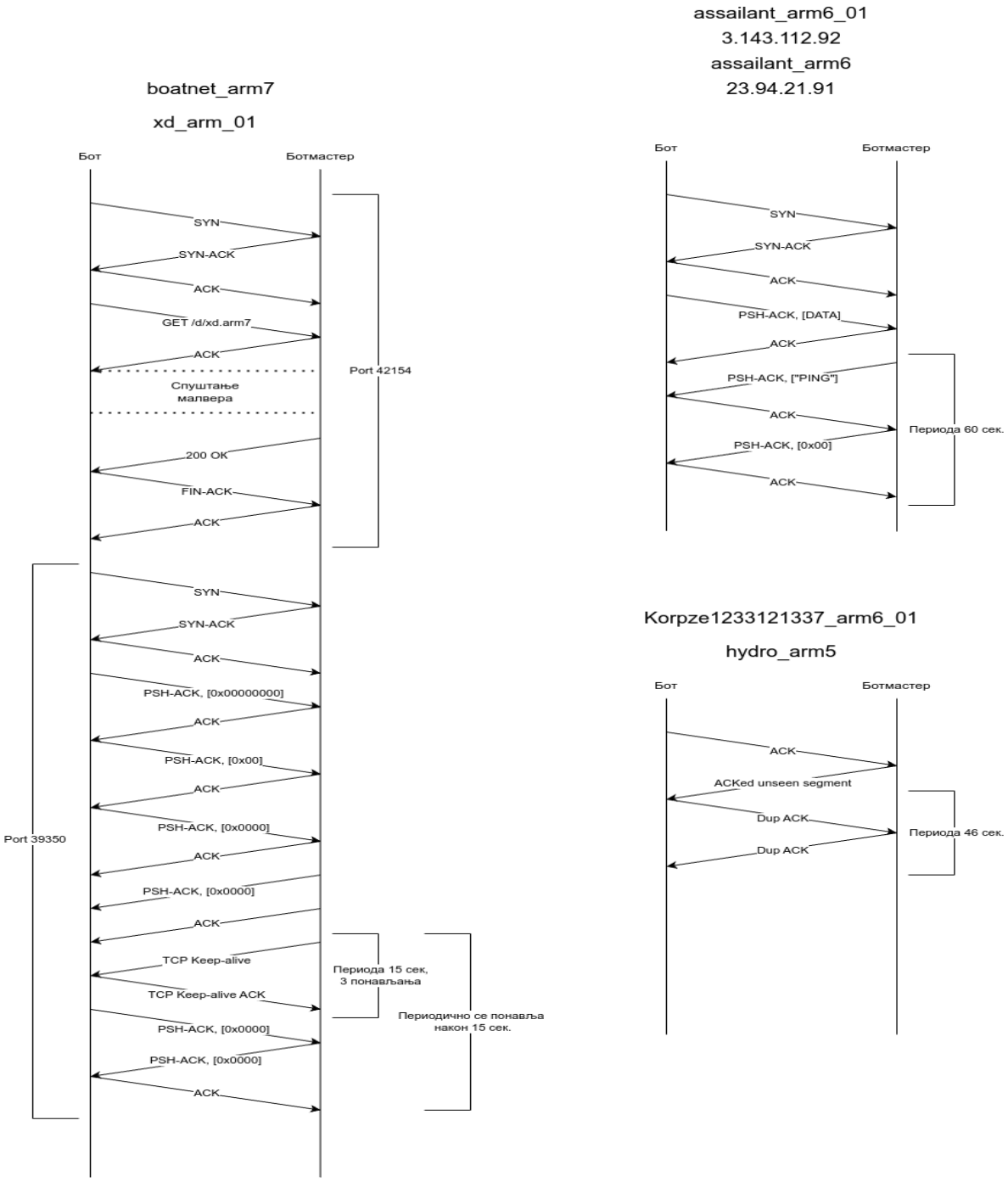
- анализи богатијег скупа одлика за опис временских низова и тиме квалитетнијег описа унутартоковских статистичких особина
- анализи новијих алгоритама машинског учења под називом градијентно ојачавање (енг. *gradient boosting*) [91][92][93] који су се показали као најбољи за табеларне податке и у области информационе безбедности [84][85],
- решавању проблема врло небалансираних скупова података какви су увек они који садрже податке о нападима на ИКТ инфраструктуру,
- анализи инваријантности мрежног понашања ботнет контролне комуникације током дужег временског периода, као и могућности детекције непознатих ботнета коришћењем детектора тренираних на старим примерцима малвера.

У наредним поглављу биће представљени сваки од корака понаособ, уз посвећивање више простора коришћеним методима и алгоритмима.

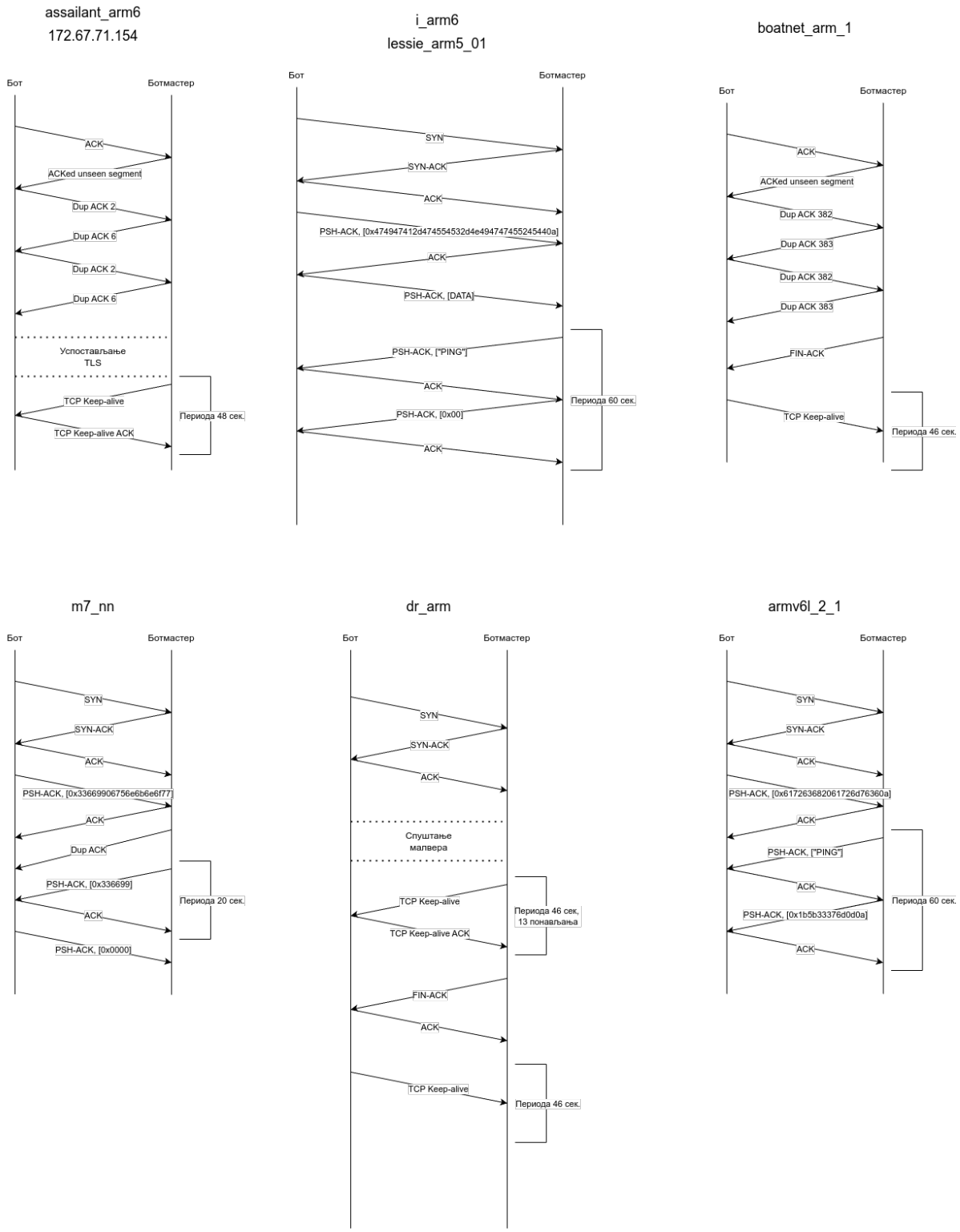
5.1 Ботнет CnC саобраћај нових узорака

У периоду од 20.12.2022. до 27.11.2023, према истој методологији описаној у поглављу 3.1 прикупљани су нови примерци IoT малвера чије је динамичко понашање анализирано. Било је анализирано укупно 14 нових примерака ботнет малвера из класе *Mirai*. На сликама 12 и 13 су приказани дијаграми комуникације између бота и ботмастера за новоприкупљене узорке.

Уочени су неки нови обрасци у комуникацији, попут коришћења различитих портова за спуштање вирусне апликације на заражене уређаје и контролну комуникацију (*boatnet_arm7, xd_arm_01*), додатно прикривање комуникације успостављањем TLS сесије (*assailant_arm6*), једносмерно периодично одржавање комуникације (*boatnet_arm1*, *dr_arm*). Међутим, иако постоје ови нови обрасци, периодична комуникација између бота и ботмастера као начин за одржавање стања бота и даље чини саставни део комуникације. Периода у случају *boatnet_arm7, xd_arm_01* малвера износи 15 секунди, у случају m7_nn малвера износи 20 секунди, у случају *Korpze1233121337_arm6_01, hydro_arm5, assailant_arm6, boatnet_arm_1, dr_arm* малвера износи 46 секунди, док у случају assailant_arm6_01, assailant_arm6, i_arm6, *lessie_arm5_01*, *armv6l_2_1* износи 60 секунди. Ово постојање дуготрајућих одлика у понашању ботнета указује на то да би требало да буде могуће детектовати нове варијанте ботнет малвера у систему обученом над старим малвером, што је је једна од полазних хипотеза ове дисертације која ће у наставку овог поглавља бити и доказана.



Слика 12: Ботнет саобраћај малвер узорака прикупљених у периоду 2022-2023 - 1



Слика 13: Ботнет саобраћај малвер узорака прикупљених у периоду 2022-2023 - 2

5.2 Екстракција одлика временских низова коришћењем библиотеке *tsfresh*

Екстракција одлика представља корак машинског учења током којег се бележе и складиште одлике које могу бити релевантне за решавање проблема. За ове експерименте, коришћен је обогаћени скуп одлика у односу на истраживање у претходној фази, добијен библиотеком *tsfresh* [94], која је прилагођена за екстракцију статистичких одлика из временских серија.

Корак који је претходио екстракцији одлика јесте, као што је описано раније у овој дисертацији, формирање временских низова бајтова и бројева пакета мрежних токова, према методологији описаној у поглављу 3.1. Списак одлика са њиховим називима у библиотеци *tsfresh* дат је у табели 9. Наиме, поред одлика које нуди ова библиотека, мануелно је интегрисана аутоматизација за екстракцију статистика аутокорелације. Ова одлика приказана је у датој табели, и носи назив `autocorrelation_stats`.

Табела 9: Одлике временских низова које екстрахује tsfresh библиотека

Назив одлике	Назив одлике
abs_energy	longest_strike_below_mean
absolute_maximum	maximum
absolute_sum_of_changes	mean
augmented_dickey_fuller	mean_abs_change
autocorrelation_stats	mean_change
benford_correlation	mean_n_absolute_max
cid_ce	mean_second_derivative_central
count_above_mean	median
count_below_mean	minimum
fft_agregated	percentage_of_reocurring_datapoints_to_all_datapoints
first_location_of_maximum	percentage_of_reocurring_values_to_all_values
first_location_of_minumum	quantile
fourier_entropy	ratio_beyond_r_sigma
has_duplicate	ratio_value_number_to_time_series_length
has_duplicate_max	root_mean_square
has_duplicate_min	sample_entropy
index_mass_quantile	skewness
kurtosis	standard_deviation
large_standard_deviation	sun_of_reoccurring_data_points
last_location_of_maximum	sum_of_reoccurring_values
last_location_of_minumum	sum_values
lempel_ziv_complexity	symmetry_looking
length	variance
linear_trend	variance_larger_than_standard_deviation
longest_strike_above_mean	variation_coefficient

Због касније дате анализе одлика које највише утичу на класификацију, а тиме и анализе особина понашања ботнета, а како се не би нарушила динамика текста, детаљан приказ свих коришћених *tsfresh* одлика дат је у Додатку Б.

5.3 Синтетичко генерисање примерака мањинске класе

Карактеристичан проблем у анализи података и детекцији аномалија у области информационе безбедности је проблем небалансираности скупова података над којима се ради анализа. Ово је последица тога што најчешће фазе напада на информациону безбедност рачунарских инфраструктура (изузимајући волуметријске DDoS нападе) представљају мањи део у односу на укупну количину бенигног саобраћаја или других података које се прикупљају на уређајима [88]. Тако и у примерима ботнет контролних мрежних токова у којима заражени уређај емитује тек неколико малих пакета у минуту, ова класа чини значајно мањи део укупне количине свих мрежних токова у једној мрежи чији се обим мери данас најчешће гигабитима пренетих података у секунди. У коришћеном скупу података који због природе форсираног заражавања уређаја има неуобичајену велику количину ботнет саобраћаја, тек нешто мање од 3% података чине мрежни токови ботнета. Стога малвер токови представљају мањинску класу, а овакав скуп података се може сматрати небалансираним.

Неке од често коришћених метода за побољшање обучавања система и детекцију у условима небалансираних скупова података су методе креирања синтетичких узорака. Оне имају значајну примену у радним процесима машинског учења у области мрежне безбедности, као што је приказано у радовима [95][96][97][98][99][100]. Једна од карактеристичних техника су различите варијанте *SMOTE* (енг. *Synthetic Minority Oversampling Technique*, у преводу техника синтетичког узорковања мањинске класе) алгоритма које су приказане у наредним одељцима дисертације. Ова техника је примењива у случају надгледаног учења (енг. *supervised learning*), будући да су потребне лабеле на основу којих ће се детектовати која је мањинска класа коју треба узорковати. У радовима [101][102], представљене су модификације SMOTE алгоритма, и примењене на проблем детекције малвера. Креирањем синтетичких узорака очекује се побољшање перформанси модела и опште могућности генерализације, што је приказано у радовима [103][104], који се односе на креирање синтетичких узорака у области детекције малвера. У експериментима у оквиру овог истраживања биће испитан утицај трију метода на резултате класификације: *Borderline*

SMOTE [105], *SMOTE ENN* [106][107], и *ADASYN* [108], који су имплементирани у библиотеци *imbalanced-learn* [109]. Такође, биће истражен и случај када се не врши вештачко узорковање примерака мањинске класе. Овај корак у радном процесу се врши одмах након екстракције података.

5.3.1 SMOTE метода

SMOTE [106], је популаран алгоритам који се користи за решавање проблема небалансираности класа у скуповима података, ради повећања прецизности модела машинског учења. Главни принцип рада је додавање примерака мањинске класе генерисањем синтетичких примерака, наместо понављања истих примерака, као што функционишу технике базиране на узорковању. Креирањем синтетичких примерака који су стратегијски распоређени уз границу одлучивања, *SMOTE* помаже у побољшању способности модела да тачно класификује примерке мањинске класе.

Основна концепција *SMOTE* методе укључује креирање синтетичких инстанци мањинске класе тако што се бира узорак из мањинске класе и налазе његових *k*-најближих суседа. Нове инстанце се затим креирају бирањем случајних тачака између изабраног узорка и његових суседа. *SMOTE* се широко користи у сценаријима где небалансираност између класа доводи до лоше перформансе модела, као што су детекција малвера, медицинска дијагностика или детекција аномалија. Међутим, важно је напоменути да *SMOTE* може унети и одређени ниво шума у скуп података. Овај шум настаје из тога што мањинска класа из скупа података може садржати аномалије, те ће *SMOTE* креирањем инстанци близу аномалних тачака података повећати број тачака података за које није сигурно да ли би у реалним условима могле да се класификују као припадници мањинске класе, уводећи шум у скуп података. Из овог разлога, овај метод треба користити с опрезом, посебно у ситуацијама где је расподела података комплексна.

Први корак у имплементацији *SMOTE* алгоритма је идентификација мањинске класе у скупу података. Алгоритам на основу броја узорака, пребројавањем узорака сваке класе, детектује која класа је мањинска. Када је мањинска класа идентификована, следећи корак је израчунавање *k*-најближих суседа за сваки од узорака те класе. Потом, се генеришу синтетички примерци мањинске класе. Бирањем насумичног примерка мањинске класе, као и насумичног суседа истог примерка, синтетички примерак се генерише на правој линији у вишедимензионом простору између ових тачака, тако што се тачне вредности сваке од одлика

бирају насумично. Овај процес се понавља онолико пута колико је потребно генерисати вештачких примерака.

Иако овај метод генерише нове примерке мањинске класе на насумичан начин тако да буду у делу вишедимензионог простора који оивичава мањинска класа, постоји и неколико мана *SMOTE* алгоритма. Једно од значајних ограничења *SMOTE* методе је њена осетљивост на шум у подацима. Пошто *SMOTE* генерише синтетичке примерке на основу постојећих узорака мањинске класе и њихових најближих суседа, узорци са статистичким грешкама у мањинској класи могу значајно утицати на генерисање синтетичких примерака. Ово потенцијално може довести до стварања синтетичких узорака који представљају тражену дистрибуцију мањинске класе.

Такође, у сценаријима где се класе преклапају у простору одлика, *SMOTE* може насумично увести синтетичке узорке који доводе до прекомерног обучавања (енг. *overfitting*). Генерисање синтетичких инстанци дуж границе одлуке, са намером да се побољша генерализација модела може повећати ризик од прекомерног обучавања, што може довести до мање репрезентативног модела.

Такође, ефикасност *SMOTE*-а у раду са комплексним расподелама података је ограничена. У случајевима где мањинска класа има комплексну расподелу или обухвата више подгрупа са различитим одликама, једноставна природа генерисања синтетичких узорака *SMOTE*-а можда неће потпуно обухватити комплексност дистрибуције података. Ово може резултовати недовољном репрезентацијом мањинске класе и можда неће довољно побољшати перформансе модела у таквим сценаријима.

Одређивање броја суседа који се разматрају (параметар k) у *SMOTE* методи је критичан параметар који утиче на процес генерисања синтетичких узорака. Избор неодговарајуће вредности за k може довести до субоптималне синтезе инстанци мањинске класе, што потенцијално може утицати на укупну ефикасност *SMOTE*-а. Осим тога, избор оптималне вредности за k често захтева доменско знање и детаљно разумевање скупа података, што може бити изазовно у пракси.

Још једно ограничење SMOTE метода је да перформансе алгоритма опадају уколико се сама мањинска класа састоји од инстанци чије се одлике статистички значајно разликују. Иако његов циљ јесте да избалансира расподелу инстанци мањинске класе, он не узима у обзир неизбалансираност унутар појединачних одлика. Ово може резултовати синтетичким

узорцима који не одражавају расподелу одлика прецизно, потенцијално доводећи до недовољног или прекомерног обучавања.

5.3.2 Алгоритам *Borderline SMOTE*

У одговору на ограничења традиционалног *SMOTE* метода, алгоритам *Borderline SMOTE* [105] нуди проширење, тиме што се фокусира на граничне инстанце мањинске класе. Ово су инстанце које су близу границе одлуке, што их чини кључним за процес учења.

Први корак у *Borderline SMOTE* алгоритму укључује идентификацију инстанци мањинске класе које су смештене близу границе одлуке. Ово су узорци који имају статистички малу дистанцу у односу на границу која одваја мањинску класу од већинске класе у простору одлика. Ове граничне инстанце обично су изазовније за тачно класификовање и зато се посебно обраћа пажња на њих у алгоритму.

Начин на који се утврђују граничне инстанце је следећи: за сваки примерак мањинске класе се бележи колико је примерака мањинске и већинске класе у k-најближих суседа. Уколико је број примерака већинске класе већи од половине броја суседа, таква инстанца се обележи као гранична инстанца мањинске класе.

Када се донесе одлука о граничним инстанцама на основу њихове близине до границе одлуке, алгоритам рачуна тзв. *поене чланства* (енг. *membership points*) у класи за ове инстанце. Ови поени показују вероватноћу да свака гранична инстанца припада мањинској класи, узимајући у обзир њен положај у односу на границе одлуке. Након рачунања поена за чланство у класи, *Borderline SMOTE* алгоритам инстанце са најнижим поенима сврстава у граничне. Ове изабране инстанце постају жижне тачке за генерисање синтетичких примерака. Синтетички примерак се генерише као насумична тачка у вишедимензионом простору која се налази између граничне инстанце и њених суседа из мањинске класе, чиме се јача граница између класа.

Слично као и код традиционалног *SMOTE* метода, *Borderline SMOTE* алгоритам наставља са генерисањем синтетичких инстанци на основу изабраних граничних инстанци. Циљ овог приступа је смањивање потенцијалног утицаја података са шумом и обезбеђивање да синтетички узорци прецизно захвате основне одлике мањинске класе.

Једна од кључних предности *Borderline SMOTE*-а је његова способност да се прилагоди комплексним дистрибуцијама података унутар мањинске класе. Фокусирајући се специфично на граничне инстанце, алгоритам има за циљ да прецизније захвати сложености дистрибуције

података, чиме се решавају ограничења традиционалног *SMOTE*-а у раду са комплексним структурама података.

5.3.3 Алгоритам уређених најближих суседа

Још један приступ који коригује ограничења *SMOTE* метода је алгоритам уређених најближих суседа (енг. *Edited Nearest Neighbors,* скр. *ENN*) [107]. *ENN* је техника подузорковања већинске класе (енг. *subsampling*) која ради тако што уклања одређене инстанце из већинске класе које могу бити погрешно класификоване. Алгоритам прво идентификује погрешно класификоване инстанце користећи приступ *k*-најближих суседа. *SMOTE* метод који користи алгоритам уређених суседа се назива скраћено *SMOTEENN*.

Када се идентификују погрешно класификоване инстанце, *ENN* селективно уклања инстанце из већинске класе које су погрешно класификоване, са циљем смањења утицаја података са шумом на процес учења. Тиме *ENN* помаже у побољшању баланса између мањинске и већинске класе, ефикасно решавајући проблем осетљивости на податке са шумом, што за *SMOTE* алгоритам представља ограничење.

Поред смањења утицаја података са шумом, алгоритам *ENN* такође решава проблем прекомерног обучавања у случају преклапања примерака класа, тиме што уклања погрешно класификоване инстанце већинске класе које доприносе преклапању региона.

Додатно, способност *ENN*-а да се прилагоди комплексности расподеле података, посебно унутар мањинске класе, чини га вредном алтернативом *SMOTE* методима. Селективним уклањањем погрешно класификованих инстанци већинске класе, ENN је способан да детектује детаље расподеле података и побољша репрезентацију мањинске класе у процесу машинског учења.

5.3.4 Алгоритам *ADASYN*

Адаптивно синтетичко узорковање (енг. *Adaptive Synthetic Sampling,* ADASYN) [108] је проширење алгоритма *SMOTE*, дизајнирано да коригује нека његова ограничења. Кључни принцип који стоји иза *ADASYN*-а је адаптивно генерисање синтетичких узорака за мањинску класу на основу расподеле постојећих узорака, са фокусом на оне области у којима су густо распоређени примерци мањинске класе.

Алгоритам *ADASYN* почиње идентификацијом инстанци мањинске класе које се налазе у близини границе одлуке, слично као и *Borderline SMOTE* приступ. Ове инстанце се сматрају

информативнијим за побољшање перформанси класификације (у даљем тексту ће се користити термин информативне инстанце), пошто представљају области простора особина које су изазовније за учење.

Када су информативне инстанце мањинске класе идентификоване, *ADASYN* одређује степен неизбалансираности за сваку од ових инстанци, рачунајући однос мањинских узорака према већинским узорцима у њиховим околинама. Прво се за сваки чвор израчуна колики проценат његових суседа представља већинска класа. Потом се ове вредности сумирају, и проценат суседа сваког чвора се дели са овом сумом, како би се израчунао удео чвора у неизбалансираности класе. Овај корак омогућава алгоритму да форсира генерисање синтетичких узорака у регионима где је небалансираност класа више изражена.

Следећи стадијум укључује додељивање већих преференција за генерисање синтетичких узорака у областима са већим степеном неизбалансираности, чиме се динамички прилагођава процес синтезе на основу локалне дистрибуције мањинских инстанци. На тај начин, циљ *ADASYN* је суочавање са изазовом варијабилности степена неизбалансираности у различитим регионима простора особина, што доприноси ефикаснијем раду с неизбалансираним скуповима података у поређењу са традиционалним *SMOTE*-ом.

Након одређивања броја синтетичких узорака који треба генерисати у свакој околини, *ADASYN* прелази на стварање синтетичких инстанци, додељујући већу важност регионима са израженијом небалансираношћу класа. Ова адаптивна процедура има за циљ да се суочи са различитим нивоима неизбалансираности у различитим областима простора особина, што доприноси стварању робусније репрезентације мањинске класе.

Будући да је проблем детекције ботнета у природи такав да се ботнет саобраћај може сматрати мањинском класом, у експериментима ће бити тестирани радни процеси који укључују различите технике вештачког генерисања узорака. Разлог томе јесте испитивање колико вештачко генерисање узорака утиче на предикцију ботнет класе. Реалистична примена овог принципа приликом сакупљања података би била да се сакупи већи број примерака ботнет саобраћаја, повећавајући његову заступљеност у скупу података.

5.4 Одабир одлика

Одабир одлика је кључан корак у радном процесу машинског учења који укључује избор најрелевантнијих одлика за изградњу предиктивног модела. Процес одабира одлика помаже у

побољшању перформанси модела тако што смањује прекомерно, поједностављује модел и убрзава процес тренирања. Избором најважнијих одлика може се побољшати интерпретација модела и постићи боље генерализације на новим подацима.

Сви алгоритми који ће бити представљени у овом поглављу се базирају на одабиру подскупа иницијалног скупа одлика, базирано на резултатима класификације алгоритма машинског учења. Потом се за сваку одлику мери значајност, како би се одлучило које одлике колико доприносе повећању скора. За меру значајности је одабрана SHAP вредност (енг. *Shapley Additive Explanations*) [110][111].

SHAP вредности се заснивају на кооперативној теорији игара, и обезбеђују конзистентан начин доделе значајности свакој одлици, за одређени проблем предвиђање. Ове вредности представљају допринос сваке одлике разлици између стварног предвиђања и просечног предвиђања. Другим речима, SHAP вредности објашњавају како присуство или одсуство одлике утиче на излаз модела. Једна од кључних предности SHAP значајности је што омогућава локално прецизно, а глобално конзистентно објашњење предвиђања модела. Ово значи да за одређени пример, SHAP вредности могу објаснити допринос сваке одлике предвиђању, а када се агрегирају преко више примера, они обезбеђују конзистентну меру значајности одлика.

Редукција димензија представља корак приликом којег се бира подскуп иницијалних одлика, у циљу издвајања најзначајнијих одлика за дати проблем. У спроведеним експериментима, биће испитане следеће методе: рекурзивна елиминација одлика (енг. *Recursive Feature Elimination*, RFE) [112], Рекурзивно додавање одлика (енг. *Recursive Feature Addition*, RFA) [113], и *Boruta* [114][115].

5.4.1 Рекурзивна елиминација одлика

Рекурзивна елиминација одлика, скраћено RFE, је техника одабира одлика чији је циљ избор подскупа најважнијих одлике за дати модел машинског учења. Ово је итеративни приступ, који ради тако што рекурзивно уклања најмање релевантну (у даљем тексту најслабију) одлику (или одлике), док се не достигне дефинисани број одлика.

Кораци метода рекурзивне елиминације одлика су следећи: одабир модела, одабир скора, циклична елиминација најслабијих одлика. Као први корак, потребно је одабрати модел на основу чијег успеха ће се мерити колико су одлике скупа података значајне, као и одабрати у односу на који скор ће се рачунати. Затим следи корак цикличне елиминације најслабијих

одлика. Почиње се од целокупног скупа одлика, и израчунава се скор. Потом, избацивањем једне по једне одлике, поново се израчунава скор. Одлике се рангирају по томе колико је њихово избацивање побољшало, односно погоршало скор модела машинског учења, и бира се одређени број најслабијих одлика које треба избацити. Циклично избацивање најслабијих одлика се понавља док се не достигне критеријум заустављања, који може бити одређени број одлика, или праг погоршања скора модела.

Једна од кључних предности рекурзивне елиминације одлика је то што помаже у смањивању прекомерног обучавања, тиме што се одабирају одлике које су најрелевантније за модел. Итеративним уклањањем најслабијих одлика, RFE може побољшати генерализацију и перформансе модела на непознатим подацима. Додатно, RFE такође може довести до смањења времена тренирања и закључивања, јер се фокусира на подскуп најважнијих одлика. Ово га чини посебно корисним када се ради са високодимензионалним скуповима података, где је избор најинформативнијих одлика кључан за интерпретацију и перформансе модела.

Важно је напоменути да, иако је RFE моћна техника избора одлика, она има нека ограничења. На пример, можда неће добро радити када постоје високо корелисане одлике или када је однос сигнала и шума низак. У таквим случајевима могу бити потребне алтернативне технике избора одлика или технике обраде и модификације података.

5.4.2 Рекурзивно додавање одлика

Рекурзивно додавање одлика, скраћено RFA, је комплементарна техника рекурзивној елиминацији одлика, која има за циљ да изабере најважније одлике за модел машинског учења. За разлику од RFE који има за принцип рада итеративно уклањање одлика, RFA ради тако што додаје одлике у модел на основу њиховог значаја рекурзивним путем.

Као што је то случај са алгоритмом рекурзивне елиминације одлика, и код овог алгоритма постоји корак одабира модела и скора. Потом се, почевши од празног скупа одлика, у свакој итерацији додају једна или више одлика које највише повећавају скор за дати модел. Циклично убацивање најјачих одлика се понавља док се не достигне критеријум заустављања, који може бити одређени број одлика, или праг погоршања скора модела.

Предност рекурзивног додавања одлика је што може помоћи у идентификацији и укључивању додатних информативних одлика које могу бити занемарене током почетног процеса избора одлика. Рекурзивним додавањем одлика на основу њиховог значаја, RFA

може побољшати могућност модела да захвати комплексне везе и шаблоне у подацима, што води ка појачаној перформанси на непознатим подацима.

Као и RFE, рекурзивно додавање одлика такође има своја ограничења. Итеративна природа RFA може довести до преобучавања ако се не контролише пажљиво, посебно када се дода велики број одлика. Додатно, RFA може бити рачунски захтеван, посебно за високодимензионе скупове података са великим бројем потенцијалних одлика које треба размотрити за додавање.

5.4.3 Борута алгоритам избора одлика

Борута алгоритам је метод избора одлика који је специфично дизајниран за идентификацију свих релевантних одлика у скупу података. Посебно је користан при раду са високодимензионалним скуповима података са великим бројем потенцијалних одлика. Борута је базиран на алгоритму случајне шуме и способан је да обрађује различите типове података, укључујући континуиране, категоричке и мешане променљиве.

Суштина Борута алгоритма је у томе да ради тако што креира *сенке одлика* (енг. *shadow features*) које представљају случајне пермутације оригиналних колона у скупу података. Упоређујући значај оригиналних одлика са значајем њихових одговарајућих „сенки“, Борута одређује значајност сваке одлике у предвиђању циљне варијабле. Ако одлика има већу значајност него „сенке“ на статистички значајан начин, сматра се релевантном и задржава се. У супротном, сматра се нерелевантном и уклања се. Овај процес се итеративно понавља док се не идентификују све релевантне одлике.

Борута има за циљ да обухвати све релевантне одлике, укључујући комплексне интеракције и нелинеарне односе, што побољшава перформансе целокупног модела. Такође, може се користити са било којим моделом, и може да се прилагоди мешавини континуираних и категоричких променљивих, елиминишући потребу за претпроцесирањем података да би се конвертовали типови података.

Алгоритам Борута има неколико корака:

Иницијализација: Борута почиње креирањем *сенки одлика*, које су премештене копије оригиналних одлика. Ове *сенке одлика* се користе као мерило за одређивање значајности стварних одлика. Битно је напоменути да је се за сваку одлику креира неколико њених сенки. У једној од имплементација број сенки је барем 5 [115]. Све сенке се потом измешају, како би се избегла корелација са одзивом.

Обучавање случајне шуме: Алгоритам случајне шуме се неколико пута тренира, сваки пут над проширеним скупом одлика. Будући да се алгоритам случајне шуме састоји из скупа стабала одлучивања, нека од стабала ће садржати оригиналне одлике, а друга сенке. Самим тим се може одредити за сваку од одлика колико сама одлика, а колико њене *сенке* доприносе губитку прецизности. Ова величина се назива и значајност одлике. Потом се, за сваку од одлика, броји колико пута је њена значајност била већа од максимума значајности *сенки одлика*. Очекује се да ће средња вредност и варијанса ове величине за *n* пуштених експеримената са алгоритмом случајне шуме износити:

$$EX(X)=0.5\,N\,,S=\sqrt{0.25\,X}$$

где је *X* случајна променљива, *EX* математичко очекивање, а *S* стандардна девијација. Ове вредности за математичко очекивање и стандардну девијацију представљају вредности за бинарну расподелу када су оба исхода једнако вероватна. За сваку од одлика се потом спроведе Z-тест, и рачуна се Z-скор. Овај статистички тест показује са којом извесношћу дата популација има нормалну расподелу, са задатом средњом вредношћу. Уколико је вредност скора 0, средња вредност је идентична задатој. Уколико је пак 1, оне се разликују за вредност једне стандардне девијације. У суштини, овај тест служи да одговори на питање да ли одлика има већу значајност од своје *сенке*. Ако значајност одлике значајно превазилази значајност њене *сенке*, означава се као *потврђена релевантна*. Ако је њена значајност нижа, означава се као *одбачена нерелевантна*. Ако значајност одлике није значајно различита од њене *сенке*, означава се као *потврђена привремено*.

Итеративни процес: *Потврђене привремене* одлике се разматрају за поновну евалуацију у наредним итерацијама док се не идентификују стабилни скуп релевантних одлика. У свакој итерацији, модел случајне шуме се поново обучава да би се поново оценила значајност одлика, на основу потврђених и привремених одлика.

Финални скуп одлика: Када се итерације споје и идентификује се стабилан скуп релевантних одлика, оне се користе за крајње обучавање и закључке модела.

Међутим, овај приступ има своје мане. Борута алгоритам зависи од методе случајних шума, што захтева времена за обучавање уколико је скуп одлика велик. Такође, могуће је да дође до прекомерног обучавања, будући да се селекција одлика врши на бази модела машинског учења.

5.5 Подешавање хиперпараметара

Селекција хиперпараметара представља оптимизацију вредности хиперпараметара модела, у циљу побољшања рада модела машинског учења. Овај корак се извршава након корака одабира одлика. За имплементацију корака одабира одлика и подешавања хиперпараметара у радном процесу машинског учења коришћен је пакет *shap-hypertune* [116]. Током подешавања хиперпараметара се након дефинисања простора хиперпараметара у виду скупа вредности за сваки хиперпараметар, као и одабира алгоритма, тражи скуп вредности хиперпараметара који максимизује дефинисани резултат класификације. За ове експерименте коришћен је алгоритам стабала Парзен проценитеља [117][118], који врше процену густине расподеле, имплементиран у библиотеци *hyperopt* [119],

5.5.1 Алгоритам стабала Парзен проценитеља

Алгоритам стабала Парзен проценитеља (енг. *Tree Parzen Estimators*, скр. *TPE*) [117][118] је метода која користи Бајесову оптимизацију да би ефикасно претраживала најбоље хиперпараметре за модел. Итеративно ажурирање функције вероватноће омогућава *TPE* да ефикасно истражује простор хиперпараметара, што га чини посебно корисним за просторе високе димензије и нелинеарне просторе претраге.

Приликом примене *TPE* за подешавање хиперпараметара, битно је имати детаљно разумевање како алгоритам моделира функцију циља (енг. *goal function*) и балансира истраживање и експлоатацију. Ово укључује пажљиви избор хиперпараметара за штимовање и дефинисање простора претраге на ком ће се извршити оптимизација. Кључни кораци коришћења *TPE* за извршавање подешавања хиперпараметара укључују следеће:

1. **дефинисање простора претраге**: Први корак је дефинисање хиперпараметара за штимовање и простора претраге на ком ће се извршити оптимизација. Ово укључује спецификацију опсега или расподеле за сваки хиперпараметар.
2. **иницијализација TPE алгоритма**: Када је простор претраге дефинисан, потребно је иницијализовати TPE алгоритам. Ово укључује постављање почетних тачака и дефинисање функција вероватноће за истраживање и експлоатацију хиперпараметара.
3. **итеративно ажурирање функција вероватноће**: TPE алгоритам итеративно ажурира функције вероватноће на основу процењене функције циља. Ово омогућава алгоритму

да ефикасно истражује простор хиперпараметара и фокусира се на обећавајуће регионе.

4. **балансирање истраживања и експлоатације**: TPE балансира истраживање и експлоатацију тако што фаворизује конфигурације хиперпараметара које су показале обећавајуће резултате, али истражује и нове конфигурације како би обезбедио опсег претраге.
5. **процењивање и избор хиперпараметара**: TPE алгоритам предлаже конфигурације хиперпараметара на основу ажурираних функција вероватноће. Ове конфигурације се затим процењују користећи функцију циља, а најбоља се бира на основу критеријума оптимизације.

Пратећи ове кључне кораке, TPE може ефикасно оптимизовати хиперпараметре и побољшати општу перформансу модела машинског учења.

5.6 Класификација

За предикцију, која се за случај предикције дискретних вредности назива и класификацијом, биће коришћен модел машинског учења под називом *градијентно ојачавање.* Наиме, биће испитане две имплементације екстремног градијентног ојачавања, *LightGBM* [120] и *XGBoost* [121][122][123]. За имплементацију два модела машинског учења коришћен је пакет *shap-hypertune* [116].

5.6.1 Градијентно ојачавање

Градијентно ојачавање [91][92][93] је техника машинског учења која је стекла популарност због своје способности да изврши класификацију са високом тачношћу. Она ради тако што гради низ стабала одлучивања, при чему свако стабло учи на грешкама претходног. Ова итеративна оптимизација омогућава градијентном ојачавању да креира снажан предиктивни модел спајајући снаге више слабијих модела.

Постоји неколико предности у коришћењу градијентног ојачавања као технике машинског учења. Неке од кључних предности укључују:

- Побољшана тачност: Градијентно ојачавање често резултује веома тачним предвиђањима због способности да минимизира грешке прве и друге врсте, што доприноси бољој генерализацији;

- Обрађује различите типове података: Градијентно ојачавање може радити са различитим типовима података, укључујући категоричке и нумеричке, без потребе за обимним претпроцесирањем. Такође, у стању је да ради са одликама којима недостаје вредност за већину података, без икаквог додатног корака у радном процесу машинског учења;
- Значајности одлика: Пружа увиде у значајности одлика, омогућавајући корисницима да разумеју које одлике највише утичу на предвиђања;
- Робусност према одступањима: Градијентно ојачавање је робустно према одступањима и може их ефикасно обрађивати, смањујући утицај података са шумом на перформансе модела;
- Скалабилност: Овај алгоритам је у стању да ефикасно ради са великим скуповима података, што га чини погодним за широк спектар примена, укључујући анализу великих скупова података.

Ове предности чине овај метод популарним избором за различите задатке машинског учења, посебно када је висока предиктивна тачност од кључног значаја.

5.6.1.1 Принцип рада градијентног ојачавања

Градијентно ојачавање ради тако што оптимизује одређену вредност функције циља додавањем слабих стабала одлучивања (слаб у овом контексту значи да је способност класификације испод оптимума), и то чини итеративним путем. Итеративни приступ укључује обучавање нових модела на грешкама које производе претходни модели, постепено минимизирајући укупну грешку и побољшавајући прецизност предвиђања. Вредност функције циља у контексту математичке оптимизације представља величину коју треба минимизовати или максимизовати процесом оптимизације. У случају градијентног ојачавања, у контексту машинског учења, то представља задату статистичку величину коју треба максимизовати на скупу за обучавање. То може бити било која величина која пореди предвиђене вредности са реалним, у односу на задати скуп података, попут прецизности, осетљивости, F1-мере итд.

Главни циљ градијентног ојачавања је да се минимизира претходно дефинисана функција губитка, која квантификује разлику између предвиђања модела и стварних вредности функције циља. Када се дода ново слабо стабло одлучивања, фокусира се на минимизирање функције губитка захваљујући грешкама од претходних модела.

Име градијентно ојачавање потиче од коришћења оптимизације градијентног спуста за минимизирање функције губитка. Градијенти функције губитка у односу на предвиђања модела се израчунавају, и ново слабо стабло одлучивања се обучава да апроксимира негативни градијент, водећи модел у правцу који смањује губитак.

Док итеративни процес траје, сваки ново слабо стабло одлучивања доприноси својим предвиђањима ансамблу (скупу стабала одлучивања), а модел се ажурира како би се смањиле грешке и побољшала перформанса. Коначни модел је тежинска комбинација ових слабих проценитеља, где се коефицијенти отежињења одређују стопом учења и бројем итерација.

Спајањем више слабих стабала одлучивања на пажљиво утврђен начин, градијентно бустовање производи предиктивни модел високе тачности, који је способан за генерализацију. Ансамбл приступ омогућава крајњем моделу да фиксира комплексне односе у подацима док избегава прекомерно обучавање, што га чини посебно ефикасним за широк спектар задатака машинског учења.

5.6.1.2 Хиперпараметри алгоритама градијентног ојачавања

Хиперпараметри су параметри који се не уче од стране модела током тренирања већ се постављају пре почетка тренинга. Они значајно утичу на понашање и перформансе алгоритма градијентног бустовања. Кључни хиперпараметри алгоритама градијентног ојачавања:

- број стабала (*n_estimators*): Овај хиперпараметар одређује број стабала одлучивања које ће бити изграђена. Већи број стабала може довести до прекомерног обучавања, док мањи може резултовати недовољно добром генерализацијом;
- стопа учења (или Смањење) (*learning_rate*): Стопа учења контролише допринос сваког стабла финалном предвиђању. Она је кључна за проналажење правилног балансирања између пресликавања и недовољног пресликавања. Нижа стопа учења захтева више стабала за достизање истог нивоа перформанси модела, обезбеђујући робустан модел;
- параметри специфични за стабла: укључују максималну дубину стабала, минималне узорке потребне за раздвајање чвора и минималне узорке потребне на сваком листу чвора. Ови параметри контролишу структуру појединачних стабала и значајно утичу на перформансе модела;

- однос узорака (*subsample*): Овај хиперпараметар одређује фракцију узорака која ће бити коришћена за обуку сваког стабла. Може помоћи у контроли пресликавања увођењем случајности у процес тренирања;
- функција губитка: Избор функције губитка, као што су средња квадратна грешка за регресионе проблеме или бинарни крос-ентропијум за класификационе проблеме, одређује циљ који модел тежи да минимизира током тренинга;
- регуларизациони параметри: алгоритми градијентног ојачавања често укључују L1 и L2 регуларизацију, како би се спречило прекомерно обучавање путем пенализације великих коефицијената.

5.6.1.3 Кључни кораци алгоритма градијентног ојачавања

Алгоритам градијентног ојачавања укључује неколико кључних корака који доприносе његовој способности да произведе предвиђања високе тачности. Први корак је иницијализација модела са константним вредностима, обично средњим вредностима циљних вредности за регресионе проблеме или логаритамски однос за бинарне класификационе проблеме. Затим се од предвиђања почетног модела одузимају стварне циљне вредности како би се добили остаци. Ови остаци представљају грешке које модел треба да исправи у наредним итерацијама.

Користећи остатке, тј. грешке претходног процењивача, конструише се ново стабло одлучивања за предвиђање тих остатака. Циљ је минимизирање остатака обучавањем стабла на грешкама претходног модела. Овај корак је такође познат као креирање *слабог процењивача* јер су појединачна стабла обично плитка и имају ограничену предиктивну моћ сама по себи.

Предвиђања новог стабла додају се предвиђањима претходног модела, ефективно смањујући остатке. Хиперпараметар стопе учења контролише допринос сваког стабла финалном предвиђању, спречавајући пресликавање скалирањем утицаја сваког стабла.

Процес изградње стабла за предвиђање остатака и ажурирања модела новим стаблом понавља се за одређени број итерација, који се одређује хиперпараметром *n_estimators*. У свакој итерацији ново стабло фокусира се на грешке претходних итерација, постепено побољшавајући предиктивну тачност модела.

На крају, предвиђања свих стабала се спајају како би се креирао крајњи модел. Предвиђања појединачног стабла се отежине на основу стопе учења и броја стабала, а потом се

отежињене вредности предвиђања сабирају, и добија се резултат предикције целокупног модела. Овај принцип даје снажан предиктивни модел, који се може добро генерализовати на нове податке.

Ови кључни кораци показују како алгоритам градијентног ојачавања итеративно побољшава предиктивне способности модела исправљајући грешке претходних модела. Комбинација више слабих модела резултира снажним и тачним предиктивним моделом.

5.6.2 XGBoost

XGBoost, што је скраћеница од *eXtreme Gradient Boosting* [121][122][123], је популарна имплементација алгоритма градијентног ојачавања која је стекла широку популарност у машинском учењу и научној анализи података. Позната је по својој рачунарској ефикасности, скалабилности и способности да испоручи високу предиктивну тачност. Ова библиотека имплементира низ оптимизација у погледу следећих ставки: ефикасније складиштење података и време тренирања, смањивање могућности прекомерног тренирања, руковање различитим скуповима података и типовима проблема.

Технике које служе за оптимизацију складиштења података и времена тренирања су:

- Конструкција стабала: за разлику од егзактног алгоритма гранања стабала, *XGBoost* користи приступ где за свако стабло, или пак за сваки чвор, дефинише скуп перцентила на који се дели део скупа података који улази у чвор, редукујући број калкулација;
- Ефикасно складиштење података: подаци се складиште у CSC (енг. *Compressed Column*) формату, како би заузели што мање простора у меморији рачунара. Уз то се за сваку колону скупа података паралелно рачунају статистике, како би се лакше нашле одговарајуће вредности за гранање чворова приликом конструкције стабала;
- Рано заустављање: *XGBoost* укључује функционалност раног заустављања, која омогућава моделу да аутоматски прекине са тренирањем, за случајеве када више не побољшава на валидационом скупу података. Ово помаже у спречавању прекомерног тренирања и смањењу времена обуке;
- Паралелно и дистрибуирано програмирање: Библиотека је дизајнирана да искористи паралелне и дистрибуиране рачунарске системе, што је чини погодном за задатке машинског учења велике скале и дистрибуиране. Ово омогућава ефикасну обуку и предикцију на великим скуповима података.

Технике које служе за смањење прекомерног обучавања су:

- Алгоритам орезивања стабала: *XGBoost* укључује механизам за орезивање стабала током процеса конструкције. Ова функција помаже у спречавању прекомерног тренирања уклањањем раздвајања изнад одређене дубине, што доводи до конзервативнијих и генерализованијих стабала;
- Технике случајног избора: технике случајног избора су такође имплементиране у *XGBoost* како би се смањила могућност прекомерног обучавања и повећала брзина обуке. Технике случајног избора укључене у *XGBoost* су случајни подскупови за обуку појединачних стабала и случајно избацивање колона на нивоу стабала и чворова стабала. Овим се постиже да стабла боље генерализују дати скуп података;
- Регуларизациони сабирци: *XGBoost* користи регуларизовани циљ учења који се састоји од функције губитка која се минимизира и регуларизационог сабирка (L1 и L2 регресија) који контролише сложеност модела. Ово осигурава баланс између предиктивне тачности и сложености модела, што доводи до робустних и генерализованих модела;
- Унакрсна валидација: *XGBoost* пружа уграђену подршку за унакрсну валидацију, што омогућава поуздану евалуацију модела и оптимизацију хиперпараметара. Ово је важно за осигуравање генерализације модела и избегавање прекомерног тренирања.

Технике за руковање различитим скуповима података и типовима проблема:

- Прилагодљива функција губитка: *XGBoost* омогућава корисницима да дефинишу прилагођене функције губитка на основу специфичних одлика проблема. Ова флексибилност омогућава моделу да оптимизује на основу различитих врста циљева, као што су регресија, класификација и рангирање;
- Руковање вредностима које недостају: *XGBoost* има уграђене могућности за руковање недостајућим вредностима током обуке и предикције, смањујући потребу за обрадом података и техникама импутације.

5.6.3 LightGBM

LightGBM, што је скраћеница од *Light Gradient Boosting Machine* [120], је још једна моћна имплементација алгоритма градијентног ојачавања која нуди неколико предности у погледу брзине и ефикасности. Позната је по својој способности да обрађује велике скупове података

и испоручује моделе машинског учења високих перформанси. Неке од кључних карактеристика *LightGBM* укључују:

- Градијентно базирано одабирање са једне стране: *LightGBM* примењује GOSS (енг. *Gradient One-Sided Selection*), технику која користи вредности градијента за ефикасно одабирање података током процеса обуке. Пошто ће инстанце са већим градијентном више допринети информационом добитку, алгоритам ће преферирати такве инстанце, и одбацити остале. Ово помаже у прорећивању и избору инстанци са већим градијентима, смањујући рачунарске ресурсе потребне за обуку;
- Ексклузивно спајање одлика: *LightGBM* укључује EFB (енг. *Exclusive Feature Binding*), који спаја међусобно искључиве одлике у једну, смањујући број одлика и побољшавајући рачунарску ефикасност. На пример, уколико су категоричке варијабле кодиране, овај принцип ће спојити две или више категоричких варијабли у једну помоћу мапирања;
- Раст стабала по листовима: За разлику од традиционалног раста стабала по дубини, *LightGBM* шири стабла на начин који је приоритетно усмерен на чворове који воде значајним побољшањима током тренирања. Овај приступ доприноси бржој конвергенцији и смањује сложеност;
- Руковање категоричким одликама: *LightGBM* нуди ефикасно руковање категоричким одликама, конвертујући их у оптималне тачке раздвајања, уклањајући потребу за енкодирањем и побољшавајући брзину тренирања;
- Алгоритам базиран на хистограму: *LightGBM* користи алгоритам базиран на хистограму за брзо рачунање тачака раздвајања током конструкције стабала, што доводи до побољшане ефикасности, посебно са великим скуповима података;
- Подршка за паралелно и GPU (енг. *Graphical Processing Unit*) учење: *LightGBM* пружа подршку за паралелно и GPU учење, омогућавајући ефикасно коришћење хардверских ресурса за убрзање тренирања и предикције.

5.7 Имплементација експеримената

У овом одељку биће представљена поставка експеримената по претходно наведеним и објашњеним алгоритмима и корацима процеса машинског учења, као и параметри који су

коришћени за исте. Сви експерименти су вршени на скупу података ETF-IoTB [65], допуњеним новим узорцима истих класа вируса из 2022. и 2023. године.

5.7.1 Поставка експеримената

У првој групи експеримената су тестиране комбинације различитих алгоритама за сваки корак радног процеса машинског учења. За сваки радни процес, експерименти су поновљени 100 пута, и узета је средња вредност свих скорова. Однос скупа за обучавање и тестирање је 90/10. Прво је спроведена група експеримената која у свом радном процесу не садржи корак вештачког генерисања узорака, и упоређени су резултати рада модела *XGBoost* и *LightGBM*. Примећено је да је модел који је дао најбоље резултате у овој групи експеримената *LightGBM*, те је он коришћен за другу групу експеримената, како би се поредио утицај различитих техника за вештачко генерисање узорака.

У табели 10 испод је дат приказ тестираних радних процеса за овај скуп података.

Табела 10: Коришћени радни процеси за детекцију ботнет саобраћаја

Редни број	Техника за вештачко генерисање узорака	Алгоритам одабира одлика	Модел
1		RFA	XGBoost
2		RFA	LightGBM
3		RFE	XGBoost
4		RFE	LightGBM
5		Boruta	XGBoost
6		Boruta	LightGBM
7	ADASYN	RFA	LightGBM
8	ADASYN	RFE	LightGBM
9	ADASYN	Boruta	LightGBM
10	Borderline SMOTE	RFA	LightGBM
11	Borderline SMOTE	RFE	LightGBM
12	Borderline SMOTE	Boruta	LightGBM
13	SMOTEENN	RFA	LightGBM
14	SMOTEENN	RFE	LightGBM
15	SMOTEENN	Boruta	LightGBM

Поред ових експеримената, спроведена је и друга група експеримената која тестира хипотезу да ли се током времена мењају мрежне одлике ботнета до те мере да их више није могуће детектовати постојећим системом на бази машинског учења. То је учињено одређивањем који узорци ботнета иду у скуп за обучавање, а који у тестни скуп. Овај део истраживања приказан је у одељку 5.7.6.

5.7.2 Параметри екстракције одлика из временских низова

За неке од одлика, библиотека *tsfresh* дефинише параметре за екстракцију. За сваку вредност или комбинацију вредности параметара, ствара се нова одлика у скупу података. У табели 11 су наведени параметри и коришћене вредности параметара само за оне одлике које имају параметре.

Табела 11: Коришћене вредности параметара одлика

Одлика	Параметар	Вредности параметра
augmented_dickey_fuller	*attr*	*teststat, pvalue, usedlag*
	autolag	*AIC, BIC, t-stat, None*
cid	*normalize*	*True, False*
fft_aggregated	*aggtype*	*centroid, variance, skew, kurtosis*
fourier_entropy	*bins*	од 2 до 10
index_mass_quantile	*q*	од 0.05 до 0.95, са кораком од 0.05
large_standard_deviation	*r*	од 0.05 до 0.95, са кораком од 0.05
lempel_ziv_complexity	*bins*	од 2 до 10
linear_trend	*attr*	*pvalue*, *rvalue*, *intercept*, *slope*
mean_n_absolute_max	*number_of_maxima*	од 2 до 5
quantile	*q*	од 0.05 до 0.95, са кораком од 0.05
ratio_beyond_r_sigma	*r*	од 0.25 до 2, са кораком од 0.25
symmetry_looking	*r*	од 0.05 до 1, са кораком од 0.05

5.7.3 Одабир одлика и оптимизација хиперпараметара

У овом одељку биће приказани коришћени параметри за корак одабира и разматране вредности хиперпараметара модела.

Први корак сва три одабрана алгоритма за одабир одлика је одабир модела и одабир мере успешности. Модел који је коришћен за експерименте је LightGBM, док је коришћена мера прецизност. Дефиниције свих параметара су подробније описане у библиотеци *shap-hypertune*. У табелама 12 и 13, приказане су коришћене вредности параметара за алгоритме одабира одлика, док су у табелама 14 и 15 представљени разматрани простори хиперпараметара за *XGBoost* и *LightGBM* моделе.

Табела 12: Параметри RFA и RFE алгоритама за одабир одлика

Параметар	Вредности
n_iter	100
sampling_seed	0
importance_type	*shap_importances*
step	0.02

Табела 13: Параметри Boruta алгоритма за одабир одлика

Параметар	Вредности
max_iter	100
sampling_seed	0
importance_type	*shap_importances*
early_stopping_boruta_rounds	1000
perc	10

Табела 14: Разматрани простор хиперпараметара за XGBoost модел

Параметар	Вредности
booster	*gbdt, dart, rf*
n_estimators	случајан број из униформне расподеле у интервалу од 100 до 200
learning_rate	случајан број из логаритамске униформне расподеле у интервалу од 0.01 до 0.2

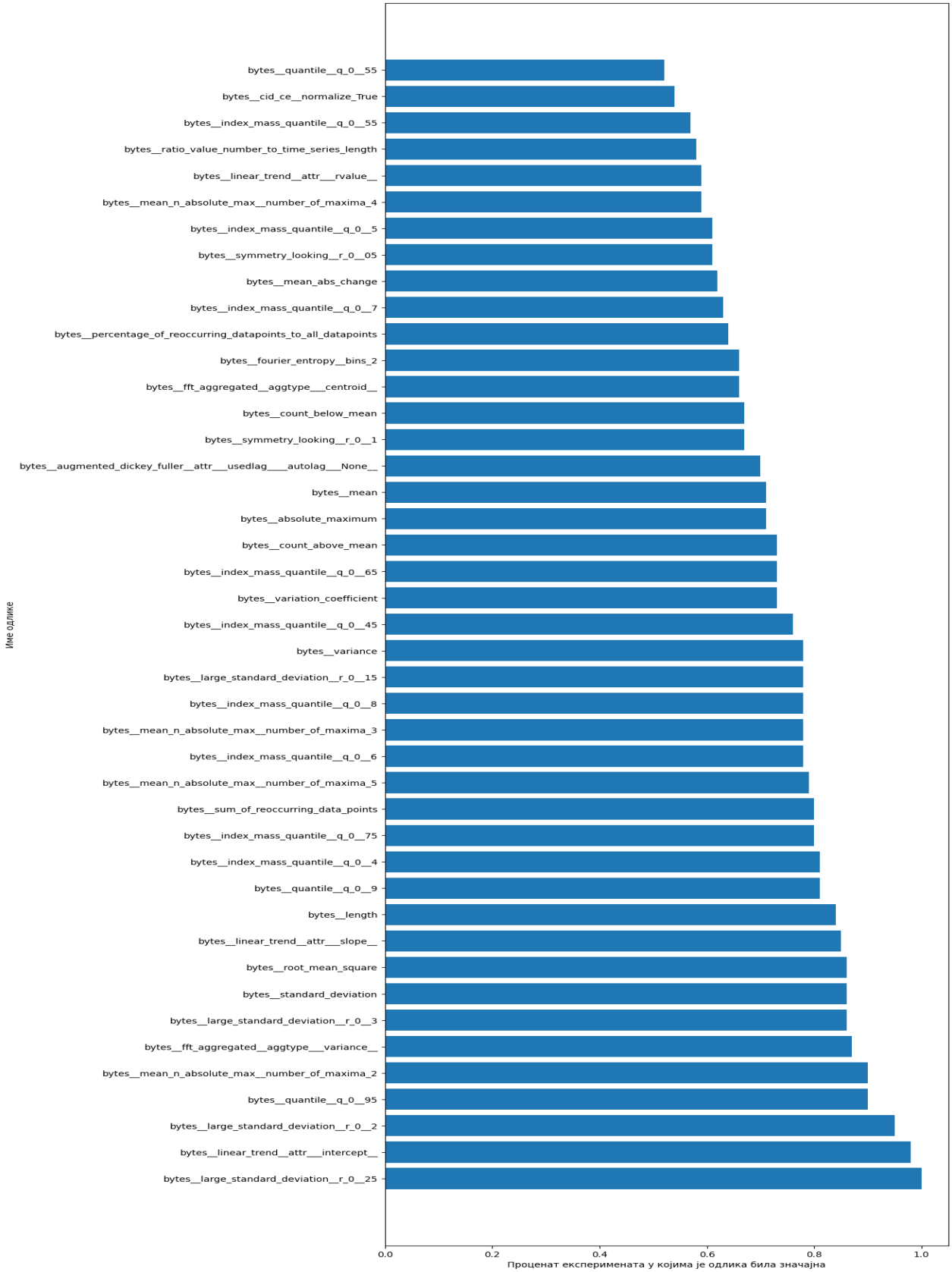
Табела 15: Разматрани простор хиперпараметара за LightGBM модел

Параметар	Вредност
boosting_type	*gbdt, dart, rf*
n_estimators	случајан број из униформне расподеле у интервалу од 1 до 200
learning_rate	случајан број из логаритамске униформне расподеле у интервалу од 0.01 до 0.2
bagging_fraction	случајан број из униформне расподеле у интервалу од 0.5 до 0.95
num_iterations	случајан број из униформне расподеле у интервалу од 100 до 200
bagging_freq	случајан број из униформне расподеле у интервалу од 2 до 10

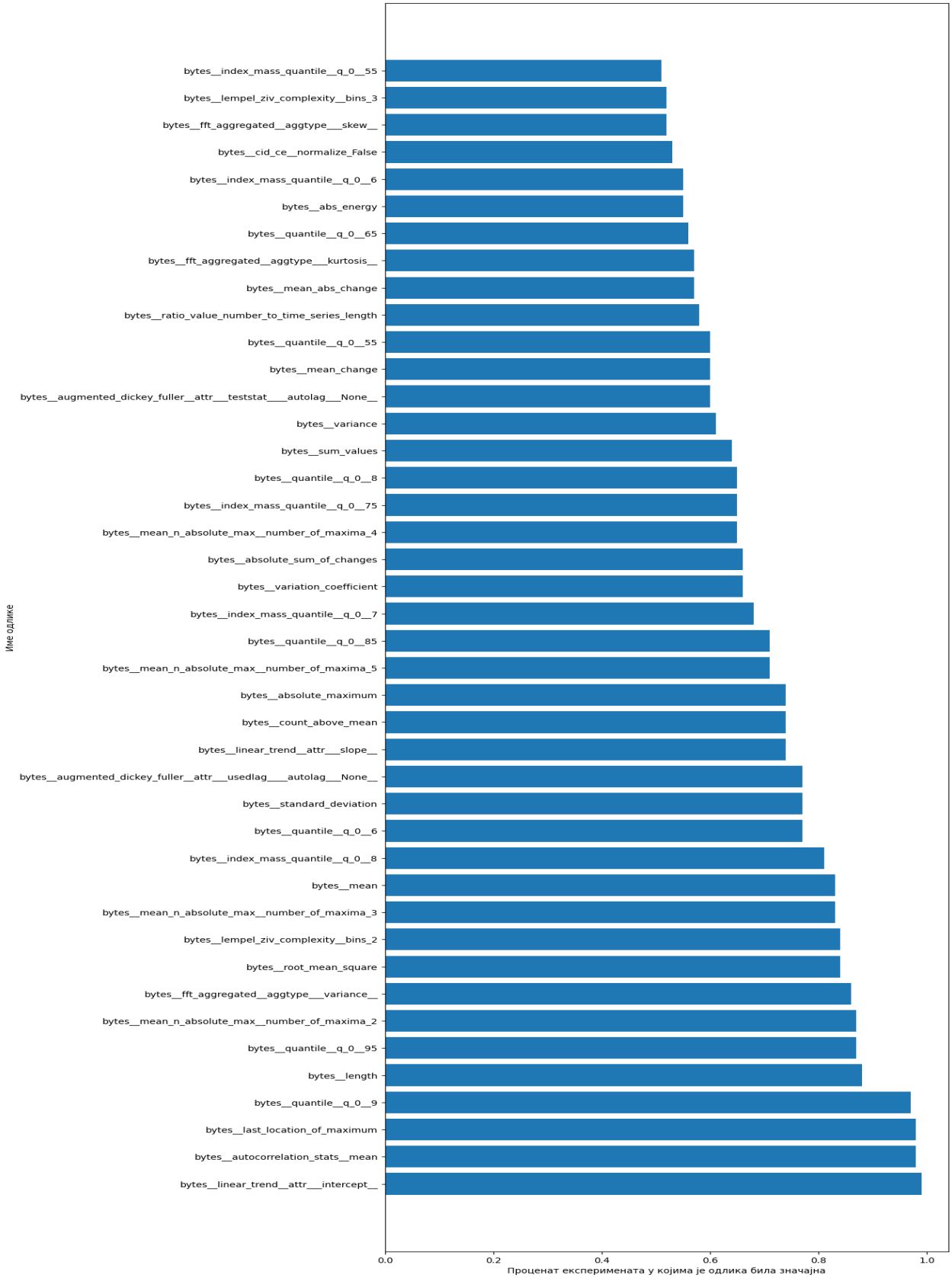
5.7.4 Резултати процеса одабира одлика

Будући да је сваки од радних процеса понављан по 100 пута, и да су при сваком понављању скуп за обучавање и тестни скуп различити, оно што је уочено је то да скуп одлика које су највише утицале на класификацију није био у потпуности идентичан, што је приказано у наредној анализи. Почетни број одлика који је екстрахован је 170. Затим, за обе варијанте радног процеса, са и без корака вештачког генерисања узорака, је екстрахован податак које су одлике утицале на резултат класификације, и са коликом значајношћу. Такође, за сваку одлику се екстрахује у колико експеримената је одлика утицала на резултат класификације. Уколико одлика има параметре, биће размотрен утицај одлике са свим задатим вредностима параметара понаособ (на пример - quantile има параметар q, који одређује вредност квантила), док ће за дискусију утицаја одлике рангирања одлике за различите вредности параметара бити агрегирана.

Број одлика који је утицао у макар једном експерименту у случају радних процеса са кораком вештачког генерисања узорака је 153, док је у случају радних процеса без корака вештачког генерисања узорака тај број 141. Све одлике су поређане по значају, односно по томе у колико експеримената су утицале на класификације. Ради читљивости, на сликама 14 и 15 су приказане све оне одлике које су биле значајне у барем 50% експеримената (у случају радних процеса са и без корака вештачког генерисања узорака, респективно). На Y-оси је приказан назив одлике, док је на X-оси приказан проценат експеримената у којима та одлика игра улогу у класификацији.



Слика 14: Најзначајније одлике у случају радних процеса са кораком вештачког генерисања узорака



Слика 15: Најзначајније одлике у случају радних процеса без корака вештачког генерисања узорака

Ови резултати показују да су следеће одлике највише утицале на класификацију:

- *linear_trend__attr___intercept* (2. на слици 14, 1. на слици 15) – указује на то да се временски низ може добро апроксимирати линеарном регресијом. Чињеница да је ова одлика једна од најзначајнијих указује да се малициозни и бенигни токови најчешће разликују у овој особини. Како ова особина указује на постојање трендова у временском низу, може се закључити да једна класа токова има, а друга нема ову особину. Када се зна да су контролни и *heartbeat* токови ботнета константног протока, а да класичне TCP сесије, које чине највећи дао бенигног саобраћаја, имају тренд повећања протока због постојања механизама спорог старта и повећања TCP прозора, онда постаје јасан разлог због ког је ова одлика значајна.

- *large_standard_deviation* (1, 3, 7, 20. на слици 14) – указује на то да ће ботнет саобраћај имати карактеристичну ширину стандардне девијације у односу на разлику минималне и максималне вредности у временској серији. У случају нормалног саобраћаја, проценат токова чија ће стандардна девијација бити већа од одређеног процента разлике минималне и максималне вредности износи изнад 90 посто, док тај проценат за ботнет саобраћај износи максимално 54 посто. Ово је очекивано, обзиром да већину временске серије ботнет саобраћаја чине вредности које се понављају, формирајући низ мале стандардне девијације.

- *autocorrelation_stats__mean* (2 на слици 15) – указује на то да ће средња вредност аутокорелације за све вредности закашњења имати различите вредности за ботнет и бенигни саобраћај. Ово је очекивано, обзиром да ботнет саобраћај чине репетитивне и периодичне временске серије са тачно одређеном вредношћу за померај једнак периоди слања пакета.

- *Quantile* (4, 11, 43 на слици 14, а 4, 6, 14, 22, 27, 32, 36 на слици 15) – указују на варијабилност и опсег праћене променљиве, у овом случају броја бајтова у временским слотовима. На основу параметара добијених експериментима, уочено је да су у случају нормалног саобраћаја, просечне вредности квинтила у оквиру временских слотова између 20.000 и 38.000 бајтова, док та вредност за ботнет саобраћај износи између 100 и 750 бајтова. Разлика у овој вредности код бенигног и ботнет саобраћаја је последица мање-више константних величина пакета у ботнет контролној и *heartbeat* комуникацији и практично било којих вредности величина

пакета (унутар технологијом одређених маргина – од 64 до 1500 бајтова) које могу да се појаве у осталим мрежним токовима.

- *mean_n_absolute_max* (5, 16, 18, 38. на слици 14, а 7, 11, 20, 25. на слици 15) – указује на то да се средња вредност првих неколико максималних вредности у временској серији значајно разликује код пакета унутар ботнет и бенигних токова. Просечна вредност ових одлика у случају нормалног саобраћаја износи између 25000 и 36000 бајтова, док вредност код ботнет саобраћаја износи између 500 и 1000. Слично као код претходног параметра, овај резултат је очекиван, будући да временске серије ботнет саобраћаја већином сачињавају пакети идентичне (и релативно мале) дужине, те ће и средња вредност неколико максималних вредности такве временске серије бити у одређеном опсегу за ботнет саобраћај.
- *fft_aggregated__aggtype___variance* (6. на слици 14, 9. на слици 15) – указује на то да да се на основу варијансе вредности коефицијената дискретне Фуријеове трансформације може детектовати ботнет саобраћај. Ово је такође очекивано, с обзиром на то да ботнет саобраћај сачињавају периодични временски низови, те ће њихова Фуријеова трансформација имати карактеристичан облик са вршним вредностима на периоди слања пакета.

Будући да је ботнет комуникација путем командног и контролног канала периодична и понавља се, за расподелу временске серије такве комуникације се очекује да неће имати велику варијансу, временска серија неће имати линеарне трендове, и имаће већину вредности које се понављају на тачно одређеним размацима, што има као последицу и карактеристични фреквентни спектар. С обзиром на чињеницу да горенаведене одлике оцењују ове особине, и чине већину одлика које су најчешће утицале на детекцију ботнет саобраћаја, то потврђује тезу да се периодичност и понављање у временским низовима ботнет саобраћаја, као и недостатак трендова могу користити као метод детекције ботнет комуникације путем командног и контролног канала.

5.7.5 Резултати експеримената детекције ботнета са различитим моделима машинског учења

У табели 16 су представљени резултати експеримената. Оно што се примећује је чињеница да радни процеси који користе корак вештачког генерисања узорака показују веће скорове. Ово је потврђено и у радовима који су користили сличну методологију [103][104]. Разлика је у

томе што у овој дисертацији није коришћена трансформација одлика. Овај резултат показује да, уколико би се прикупило више узорака ботнет малвера, прецизност система за детекцију базираног на машинском учењу би била већа. Свакако, за даљу дискусију и експерименте, радни процеси који користе корак вештачког генерисања узорака, и они који то не користе биће разматрани одвојено, јер се структурално разликују једни од других.

Табела 16: Резултати коришћених радних процеса

Редни број	Ботнет саобраћај прецизност	Ботнет саобраћај повратак	Ботнет саобраћај F1-мера	Регуларни саобраћај прецизност	Регуларни саобраћај повратак	Регуларни саобраћај F1-мера
1	0.9068	0.7536	0.817	0.9934	0.9977	0.9956
2	**0.9098**	**0.775**	**0.8274**	**0.994**	**0.9977**	**0.9959**
3	0.9059	0.72	0.7933	0.9925	0.9978	0.9952
4	0.8909	0.7233	0.7902	0.9926	0.9974	0.995
5	0.8606	0.695	0.7607	0.9919	0.9967	0.9943
6	0.8777	0.67	0.7476	0.9912	0.9972	0.9942
7	0.9969	0.9971	0.997	0.9971	0.9969	0.997
8	0.9926	0.9953	0.994	0.9953	0.9926	0.9939
9	0.9936	0.996	0.9948	0.996	0.9936	0.9948
10	0.997	0.9959	0.9964	0.9959	0.997	0.9964
11	0.9935	0.9944	0.994	0.9944	0.9935	0.994
12	0.9945	0.9951	0.9948	0.9952	0.9945	0.9948
13	**0.9991**	**0.9992**	**0.9992**	**0.9992**	**0.9991**	**0.9991**
14	0.9978	0.9965	0.9971	0.9964	0.9977	0.9971
15	0.9986	0.9981	0.9984	0.9981	0.9986	0.9984

Резултати показују да је радни процес под редним бројем 2 (RFE и *XGBoost*) имао највеће скорове у категорији радних процеса који немају корак вештачког генерисања узорака, док је радни процес под редним бројем 13 (*SMOTEENN*, *Boruta* и *LightGBM*) имао највеће скорове у категорији радних процеса који имају корак вештачког генерисања узорака. Ова два радна процеса биће разматрана за даље експерименте.

5.7.6 Резултати експеримената варирања тестног скупа

Будући да су радни процеси 2 и 13 показали најбоље резултате на ботнет скупу у погледу F1-мере, ова два радна процеса су одабрана за другу групу експеримената у којима су

анализиране стратегије обучавања старим и новим примерцима ботнета, као и могућност детекције нових варијанти ботнета обучавањем над старим примерцима. Прва група експеримента је извршена тако што су узорци из оригиналног скупа података ETF-IoTB (2019-2021) [65] стављени у скуп за обучавање, док су новији узорци (2022-2023) чинили тестни скуп. Ови експерименти су означени у табели 17 са П1. Друга група експеримената је извршена тако што су узорци из оригиналног скупа података ETF-IoTB [65], заједно са половином новијих узорака стављени у скуп за обучавање, док је остатак новијих узорака чинио тестни скуп. Ови експерименти су означени у табели 17 са П2. Коначно, у трећој групи експеримената, означеној са П3, и скуп за обучавање и тест скуп су формирани од свих прикупљених узорака малвер токова – и новијих и старијих. Ови експерименти су тестирани на радним процесима који су се најбоље показали у експериментима описаним у поглављу 5.7.5: један радни процес из групе експеримената која садржи корак вештачког генерисања узорака, као и један радни процес из групе експеримената који не садржи исти. У табели 17 су приказани резултати ових експеримената.

Табела 17: Резултати експеримената који укључују тестирање хипотезе о инваријабилности одлика ботнета

Скуп података	Редни број радног процеса	Ботнет саобраћај прецизност	Ботнет саобраћај повратак	Ботнет саобраћај F1-мера	Регуларни саобраћај прецизност	Регуларни саобраћај повратак	Регуларни саобраћај F1-мера
П1	**2**	**0.9706**	**0.8462**	**0.9041**	**0.9869**	**0.9978**	**0.9923**
П1	13	0.9871	0.8669	0.9231	0.8807	0.9886	0.9315
П2	2	0.8929	0.8971	0.8913	0.9948	0.9945	0.9946
П2	13	1.0	0.9302	0.9638	0.934	1.0	0.9659
П3	2	0.9098	0.775	0.8274	0.994	0.9977	0.9959
П3	**13**	**0.9991**	**0.9992**	**0.9992**	**0.9992**	**0.9991**	**0.9991**

Ови резултати очекивано показују да, што је већи број новијих примерака малвера у скупу за обучавање, мањи је проценат лажно негативних резултата, што значи да је генерална тачност система за детекцију већа. Међутим, интересантније је уочити да тестови извршени тако што је систем био обучаван искључиво старим, а детектовао је нове ботнет малвере (случај П1), код кога су добијени резултати 0.9706 и 0.9871 показују да је могуће са великом прецизношћу детектовати нове ботнет малвере на којима систем није обучаван (тзв. *zero-day* нападе). То такође приказује чињеницу да, иако је постојао значајан временски размак између нових и

старих узорака од четири године, посматране одлике мрежног понашања су остале довољно сличне да се ботнет мрежна комуникација може детектовати са значајном успешношћу.

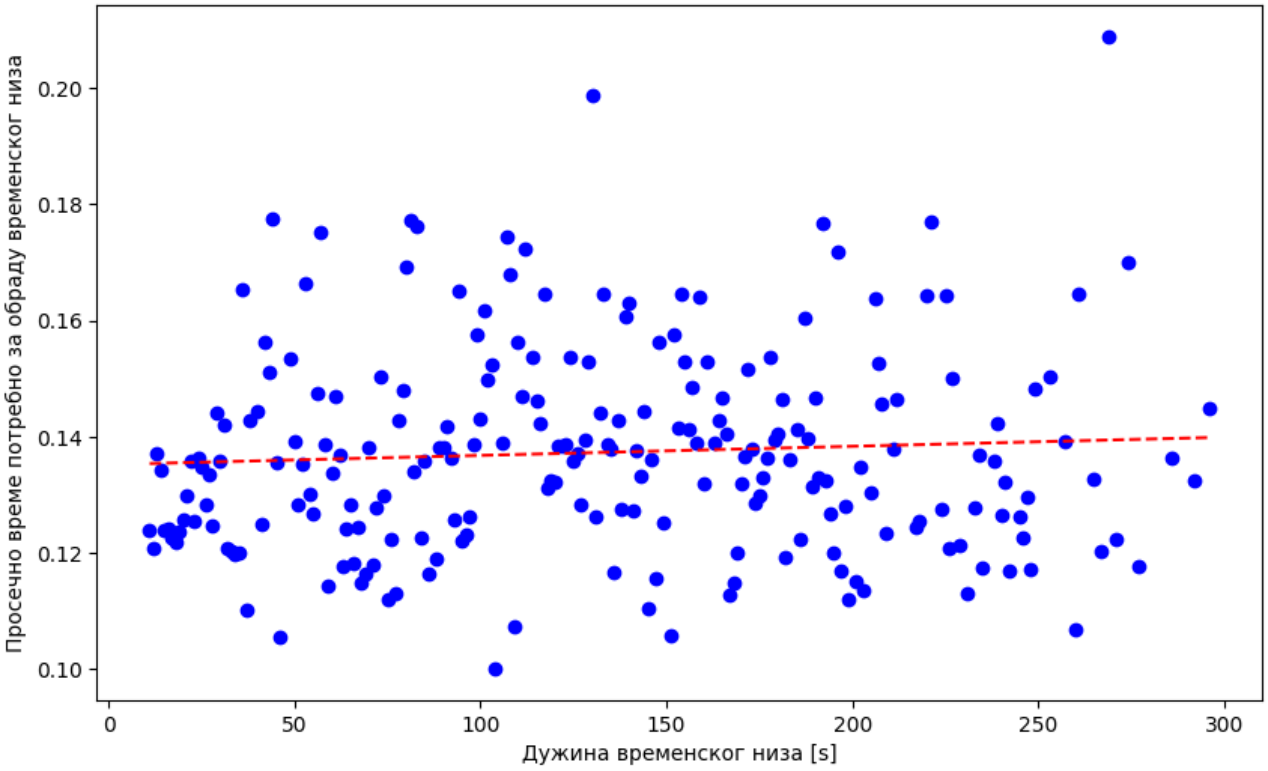
5.7.7 Анализа времена обраде временских токова и времена стабилизације најзначајнијих одлика

У овом одељку биће дата анализа процене трајања екстракције свих коришћених одлика временских низова, као и времена стабилизације најзначајнијих одлика за случај ботнет саобраћаја. Сврха ове анализе јесте дискусија о могућности конструкције система за детекцију ботнет саобраћаја у реалном времену.

Користећи скуп података из овог одељка, за сваки од мрежних токова је засебно покренут процес екстракције статистичких података из временских серија. Забележене су дужина сваког мрежног тока, као и време потребно за екстракцију. Потом је за сваку индивидуалну дужину тока узета средња вредност времена потребног за екстракцију, и примењена је линеарна интерполација.

Експерименти везани за екстракцију параметара су вршени на рачунару са процесором *AMD Ryzen 7 5825U*, који има 16GB RAM меморије, на *Python* програмском језику, верзије интерпретатора 3.10.12 , у *Jupyter* окружењу верзије *jypyter_client* 8.0.3.

За потребе графичког приказа резултата приказан је сегмент скупа података дужина до 300 секунди, будући да већи део скупа података припада овом подскупу. График зависности времена обраде од дужине временског тока приказан је на слици 16.



Слика 16: График зависности времена потребног за екстракцију и дужине временске серије

На основу линеарне интерполације, просечно време потребно за екстракцију свих одлика има линеарну зависност:

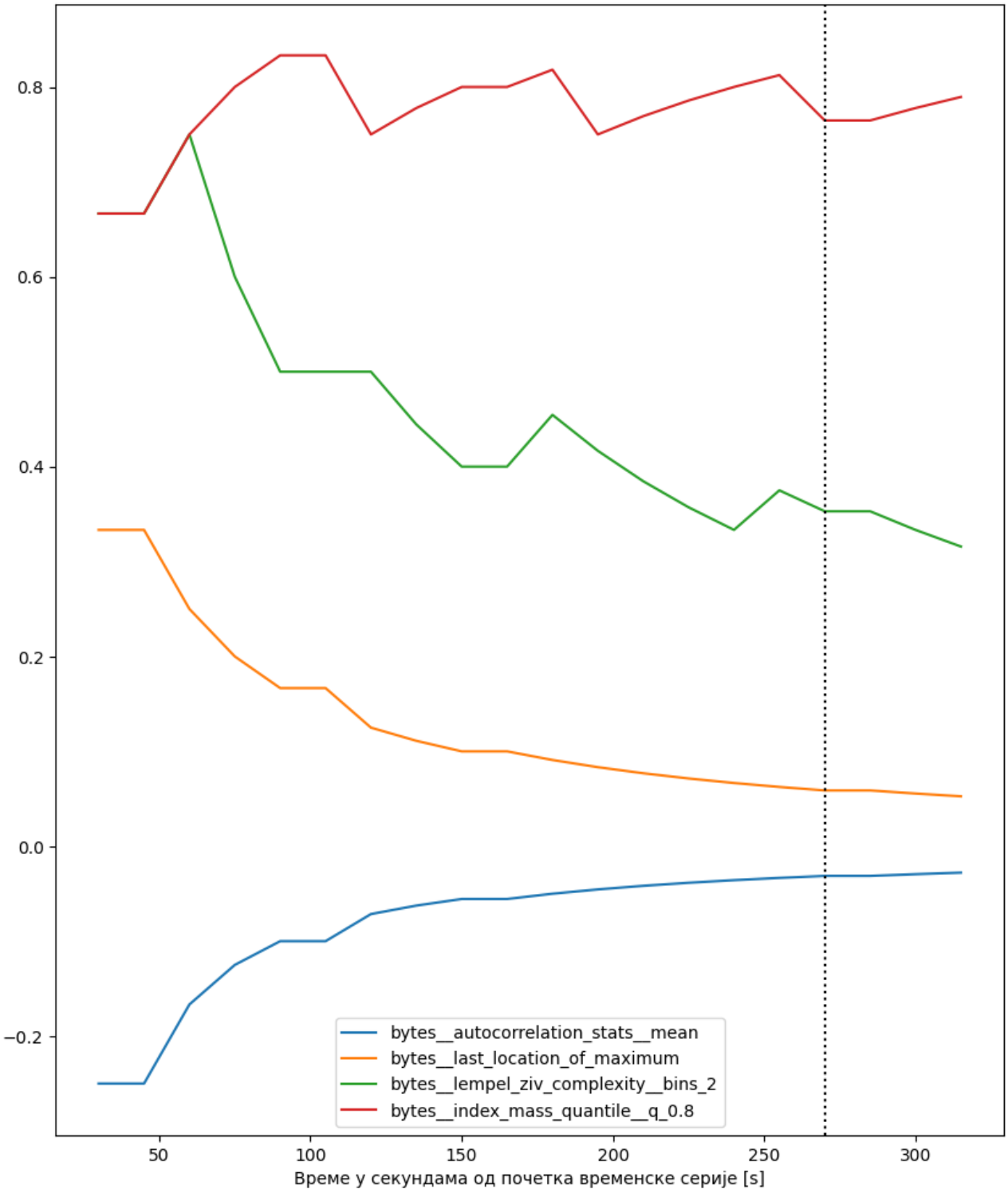
$$t = 2.01 \cdot 10^{-5} \cdot l + 0.146 [s]$$

Где је t време потребно за екстракцију, а l дужина временског низа. Екстракција свих 4585 токова је трајала 35s, што даје приближно 7.6 ms/ток. Време за екстракцију свих токова је краће него када би се из сваког од токова појединачно вршила екстракција. Разлог томе јесте чињеница да библиотека *tsfresh* користи паралелизацију приликом обраде, чије предности није могуће искористити у случају екстракције статистичких одлика једног засебног тока.

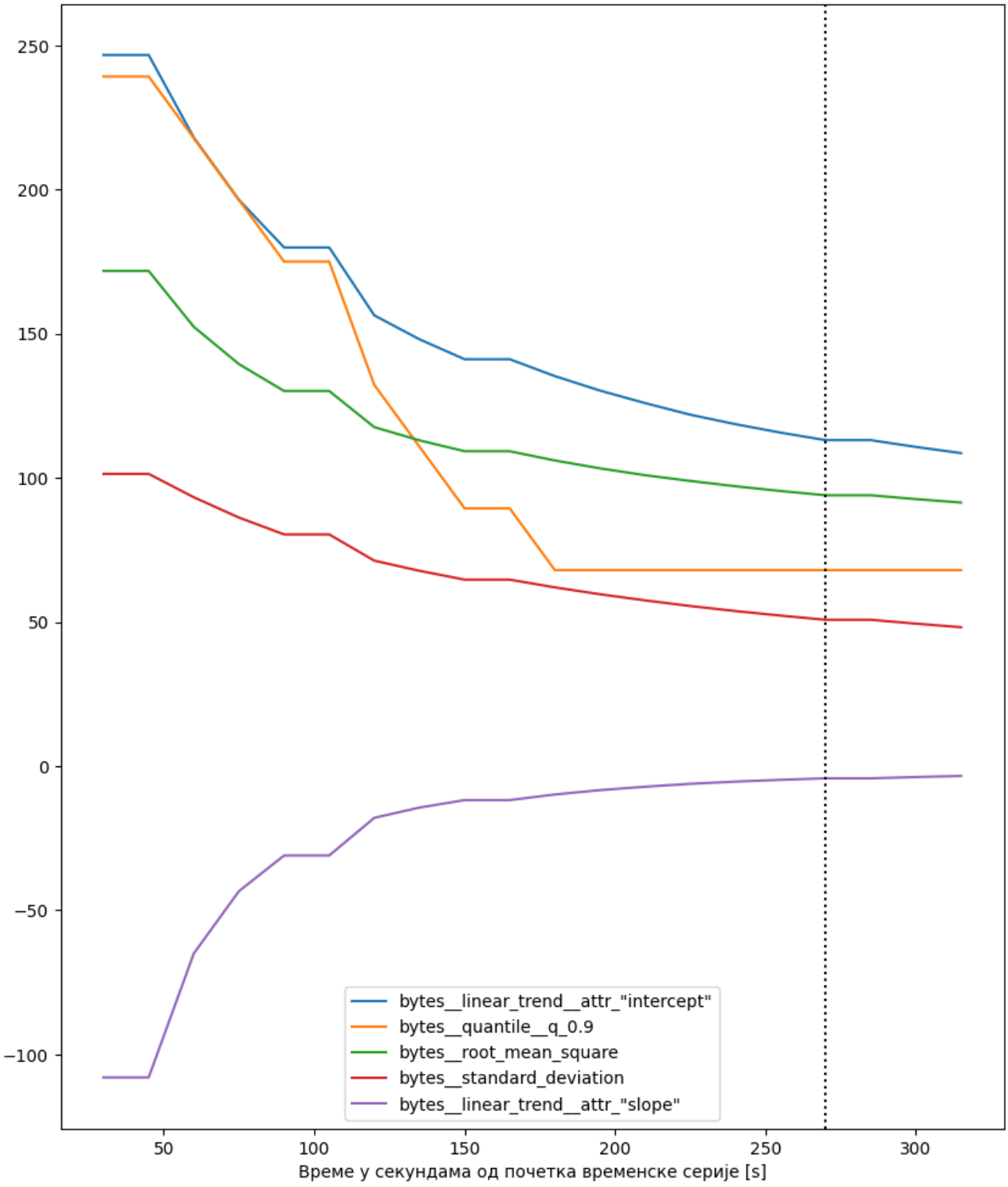
Како би се проценио потенцијал стварања система за детекцију ботнета у реалном времену, потребно је установити да ли један такав систем може у реалном времену обрадити количину токова присутну у реалистичнијим условима интернет саобраћаја. Рад [124] је истраживао дужине и величине токова у реалном интернет саобраћају. Пример реалног интернет саобраћаја који је употребљен јесте саобраћај са кампуса *AGH* универзитета у Кракову, у Пољској. Скуп података је прикупљан 30 дана, и садржи преко 4 милијарди токова и 275 TB интернет саобраћаја, што обрачунавањем износи 1555 токова у секунди. На основу прикупљеног скупа података већина токова је краћа од 16 пакета, чак 90,5756 посто. Будући да се ботнет саобраћај већински састоји из дугачких токова, кратки токови не би били

обрађени од стране система за детекцију ботнет саобраћаја на бази машинског учења. То значи да преостали саобраћај који треба обрадити чини 9,4244 посто укупног саобраћаја, што износи 146 токова у секунди. Уколико се узме да је просечна брзина обраде једног тока 7,6ms, време потребно да се обраде сви релевантни токови мрежног саобраћај износи 1.1 секунд. Време обраде мрежних токова показује да постоји потенцијал за стварањем система за детекцију ботнета у реалном времену, који би био у стању да ради у реалним условима коришћењем предложених метода.

Како би се адекватно проценило колика је дужина низа ботнет саобраћаја потребна за детекцију, посматране су вредности одлика једног примерка ботнет саобраћаја кроз време, и бележена је вредност сваке од одлика. Одабране одлике представљају девет најзначајнијих одлика, имајући у виду обе врсте радних процеса, наиме са и без корака вештачког генерисања узорака. Разлог посматрања ових одлика кроз време јесте како би се установило у ком тренутку неће бити велике промене у вредностима ових одлика. На основу тог тренутка може се дати процена за које минимално време је могуће детектовати ботнет саобраћај. На сликама 17 и 18 је дат график вредности ових одлика у времену. Посматране одлике су мерене са кораком од 15 секунди. Мерене одлике су приказане на два различита графика због прегледности слике и различитих скала посматраних параметара.



Слика 17: График зависности најбитнијих одлика у времену за ботнет саобраћај - 1



Слика 18: График зависности најбитнијих одлика у времену за ботнет саобраћај - 2

Неформалном анализом, са графика се може видети да након 270 секунди вредност већине ових одлика, које према претходној анализи имају највећи утицај на класификацију, доживљава мале промене. Ово имплицира да је, у безбедносном систему у реалном времену, потребно око 270 секунди мерења мрежних токова како би се детектовао ботнет саобраћај. Такође, имплицира се да није потребно разматрати мрежне токове краће од 270 секунди, чиме систем добија на ефикасности у виду времена потребног за обраду мрежних токова. На основу формуле за израчунавање времена обраде из овог одељка, време потребно за екстракцију података из временског низа те дужине износи 0,151403 секунди. Због кратког времена за екстракцију података, закључује се да је могуће дизајнирати систем за детекцију ботнет саобраћаја на основу статистичких података мрежних токова, који би радио у реалном времену, и који би имао за циљ детекцију саобраћаја између бота и ботмастера пре него што дође до напада, смањујући штету коју би ботнет могао да нанесе.

6 Закључак

Ова дисертација је резултат четворогодишњег истраживања (2019-2023) понашања IoT ботнета које је спроведено у више фаза. а чији је циљ откривање природе динамичког мрежног понашања ботнета и могућност изградње ефикасног система који би успешно детектовао комуникацију између бота и ботмастера пре него што дође до напада.

У првој фази истраживања (2019-2021), али и из касније анализе спроведене током 2022-2023 су детектовани обрасци мрежног понашања који су карактеристични за најновије верзије *Mirai* и *Gafgyt* малвера за IoT уређаје. У овој фази (али и током целокупног истраживања) је уочено да малвер за IoT уређаје још увек не користи напредне технике за прикривање адресе ботмастера, попут DNS флукса или алгоритама генерисања домена. Друго, сви примерци малвера су показали карактеристични рад командног и контролног канала ботнета који се заснива на периодичном одржавању комуникације, са различитим периодама и бројем пакета. Ове карактеристике су указале да мрежну конверзацију између бота и ботмастера треба посматрати као временску серију, за разлику од постојећих приступа у стручној и научној литератури који посматрају мрежну комуникацију или на нивоу пакета или целог тока.

У другој фази истраживања која је обухватила примерке малвера из прве две године истраживања је предложен, демонстриран и верификован нови оригинални метод за откривање комуникације ботнета путем командног и контролног канала заснован на SDN и унутартоковским статистикама - PI-BODE. Он пружа више статистичких одлика описа саобраћаја него методе засноване на чистим мрежним токовима (на пример *NetFlow*), али не користи анализу свих пакета на вези. Овакав метод представља компромисно решење које штеди меморију и ангажовање процесора за до два реда величине, али ипак пружа довољно детаља о мрежним конверзацијама како би се уочила разлика између злонамерног и нормалног саобраћаја. Експериментални резултати су показали да PI-BODE постиже исту или већу тачност откривања ботнета као и слични системи који користе анализу свог саобраћаја. Додатни закључци се могу извести из приказаних истраживања. Прво, резултати доказују да се комуникација ботнета са ботмастером може открити користећи статистике унутар тока и предложене одлике саобраћаја. Друго, с обзиром на то да је обука и процена PI-BODE вршена на скуповима података који садрже различите примерке злонамерног

софтвера, показује се да различите верзије злонамерног софтвера имају сличне карактеристике комуникације путем командног и контролног канала, што даје назнаке да и нови примерци злонамерног софтвера могу потенцијално бити откривени системом обученим на старијим обрасцима злонамерног софтвера. Ови закључци уједно значе да су у потпуности потврђене и прве две полазне хипотезе ове дисертације: 1. да постоје обрасци понашања ботнет малвера који су заједнички за више различитих класа малвера, а који могу да се искористе за поуздану детекцију ботнета у дужем временском периоду и 2. да је могуће оптимизовати капацитете за складиштење података о малверима као и процесорско оптерећење, без губитка прецизности детекције ботнет малвера. Уједно ови резултати су представљали основу за даље прикупљање примерака ботнет апликација и експанзију постојећег система.

Трећа фаза истраживања је представљала проширење друге фазе. Прикупљени су нови узорци ботнет саобраћаја током 2022. и 2023. године, проширен је скуп прикупљених статистичких одлика, коришћене су најмодерније технике за сваки корак радног процеса машинског учења, уведено је коришћење техника вештачког генерисања узорка за решавање проблема небалансираности скупа података. Новији узорци ботнет саобраћаја су потврдили резултате прве фазе да ботнет саобраћај није значајно променио описане шаблоне понашања. Из спроведених експеримената се може извући више закључака. Прво, коришћење вештачког генерисања узорака повећава прецизност постојећег система. Будући да је ботнет саобраћај заступљен у значајно мањој мери од нормалног мрежног саобраћаја, овај закључак сугерише да се са прикупљањем већег броја узорака ботнета повећава и прецизност система за детекцију базираног на машинском учењу, али и да постоје механизми којима могу да се постигну одлични резултати класификације и када је скуп прикупљених малициозних примерака релативно мали. Друго, резултати експеримената са различитим поделама скупова за обучавање показују да се одлике једне групе узорака ботнет саобраћаја могу користити за детекцију новоприкупљених узорака и да је могућа детекција нових малвера коришћењем система обучаваног подацима прикупљеним из старијих, што је уједно и потврда треће полазне хипотезе овог рада. То значи да, током 4 године прикупљања ботнет узорака, комуникација између бота и ботмастера путем командног и контролног канала није суштински променила своју форму. Целокупан закључак истраживања јесте да је могуће детектовати ботнет мрежно понашање између бота и ботмастера на основу карактеристика конверзације. Самим тим се може и осмислити систем за превенцију упада који би ботнете детектовао пре него што од ботмастера добију команду да изврше напад. Додатне анализе,

које су обухватале процене времена екстракције одлика у зависности од дужине временског низа, времена потребног за обраду токова у реалном окружењу, као и анализа времена стабилизације најбитнијих одлика у зависности од дужине временског низа су показале да је могуће реализовати детекцију ботнета у реалном времену, а не након завршетка мрежног тока или снимка пакета, чиме је потврђена и четврта полазна хипотеза овог рада.

Наредна истраживања на тему детекције ботнет саобраћаја би имала више праваца. Наиме, потребна су додатна истраживања, чији би циљ био додатна експанзија постојећег скупа параметара који се екстрахују из мрежних токова, као и анализа најзначајнијих одлика. Ово истраживање би обухватало и проналажење нових примерака ботнет вируса, из што више различитих класа, не искључиво IoT, како би се анализирало њихово мрежно понашање, и, последично, установио скуп одлика са најбољим перформансама за различите класе ботнета.

Проблем детекције малициозног понашања у области информационих технологија је веома сложен. Једном пронађени и проучени принципи за детекцију малициозног понашања ће сутра нападачима послужити као мотив за проналажење начина за сакривање рада њиховог софтвера и заражених уређаја. Може се очекивати да, уколико је као у овој дисертацији показано да су одлике којима се разликује понашање командне и контролне комуникације малициозног и бенигног саобраћаја на пример варијанса, стандардна девијација, мере периодичности или разлика у постојању линеарних трендова, да нападачи покушају да сакрију командне и контролне канале модификацијом понашања малвера како би опонашали бенигни саобраћај и још боље утопили своју комуникацију у њега. Ово представља нову класу, тзв. непријатељских (енг. *adversarial*) напада у којима нападачи анализирају могућности и начин рада детектора и модификују понашање малвера у складу са тиме. Наредне фазе истраживања ће анализирати нове трендове у понашању IoT малвера, као и могућности за модификацију рада ботнета кроз варирање параметара који су служили за детекцију, како би се пронашли нови, робуснији механизми детекције. Такође, пошто се рачунарски вируси стално развијају, обука на основу скупа података који садржи курентне ботнете можда неће бити применљива за откривање ботнета у будућности. Због тога би будућа истраживања требало да буду усмерена на развој методологије машинског учења са могућношћу континуираног учења.

Будући да је коначни циљ дизајнирања система за детекцију у реалном времену његова имплементација у реалистичном окружењу, потребна су додатна истраживања, чија би сврха била оптимизација радних процеса машинског учења, како би се минимизовало време детекције. Ово истраживање би обухватало анализу расположивог програмабилног мрежног

хардвера и нових парадигми управљања и екстракције одлика (нпр. P4 програмски језик), анализу перформанси различитих метода за екстраховање најбитнијих одлика, оптимизацију хиперпараметара, као и што већег скупа модела машинског учења. Као наставак ових истраживања би била спроведена идентична анализа за систем базиран на дубоком учењу.

A Додатак А: историјски приказ ботнет малвера по годинама

У табели 18, приказани су најпознатији примерци ботнета по годинама од године настајања ове класе рачунарских вируса до тренутка писања рада.

Табела 18: Најпознатији ботнети

Година	Име	Број ботова	Протокол
1993	Eggdrop	--	IRC
1998	GTBot	--	mIRC
1998	NetBus	--	HTTP
2002	Sdbot/Rbbot	---	IRC
2002	Agobot	--	IRC
2003	Spybot	--	P2P, IRC
2003	Sinit	--	P2P
2004	Bobax	100.000	--
2004	Bagle	230.000	SMTP
2006	Rustock	150.000	IRC
2007	Akbot	1.300.000	IRC
2007	Cutwail	1.500.000	SMTP
2007	Srizbi	450.000	IRC
2007	Storm	160.000	P2P
2007	Gozi	100+ банковних рачуна	HTTP[125]
2008	Conficker	10.500.000	HTTP/P2P
2008	Mariposa	12.000.000	IRC/HTTP
2008	Sality	1.000.000	P2P
2008	Asprox	15.000	HTTP
2008	Gumblar	--	HTTP
2008	Waledac	80.000	SMTP/P2P
2008	Onewordsub	40.000	SMTP
2008	Xarvester	10.000	SMTP
2008	Mega-D	509.000	HTTP
2008	Torpig	180.000	HTTP/IRC
2008	Bobax	185.000	HTTP

Година	Име	Број ботова	Протокол
2008	Lethic	260.000	IRC
2008	Kraken	495.000	IRC
2009	Maazben	50.000	SMTP
2009	Grum	560.000	SMTP
2009	Festi	--	SMTP/DoS
2009	BredoLab	30.000.000	HTTP/SMTP
2009	Donbot	125.000	HTTP
2009	Wopla	20.000	HTTP
2009	Zeus	3.600.000	--
2010	Kelihos	300.000	P2P
2010	TDL4	4.500.000	IRC
2010	LowSec	11.000	HTTP
2010	Gheg	30.000	DoS
2011	Flashback	600.000	P2P
2012	Necurs	9.000.000	HTTP, UDP[18][126]
2012	Chameleon	120.000	HTTP
2013	Boatnet	500+ сервера	--
2014	Dyre	200.000+	HTTP[127]
2014	Emotet	-	HTTP[128]
2014	Bashlite	1.000.000+ IoT уређаја	IRC, TCP[129][130]
2015	Retadup	850.000[131]	HTTP[19]
2016	Mirai	1.000.000+ IoT уређаја	TCP, UDP [16]
2016	TrickBot	200.000	HTTP са SSL[21]
2016	MethBot	570.000+ IP адреса[20]	-
2016	Hajime	120.000+	DHT, uTP[132]

Година	Име	Број ботова	Протокол
2017	Smominru	500.000+[133][24]	-
2017	WireX	70.000+	HTTPS[134]
2018	Coinhive	200.000+	HTTP[23]
2018	Satori	2000+	HTTP[135][136]
2019	GoBrut	2000+	HTTP[137]
2019	Kerberods	-	HTTP[138]
2020	FluBot	-	SMS, HTTPS[139]
2021	ZHTrap	-	HTTPS, Tor[35]
2022	Zerobot	-	TCP, HTTP[140]
2023	InfectedSlurs	10000+[141]	-

Б Додатак Б: опис коришћених одлика временских низова

У овом додатку су описане коришћене статистичке одлике временских низова за радне процесе машинског учења из поглавља 5.

`abs_energy` представља апсолутну енергију временске серије, дефинисану као:

$$E = \sum_{i=1,\dots,n} x_i^2$$

`absolute_maximum` представља највећу апсолутну вредност у временској серији.

`absolute_sum_of_changes` представља суму апсолутних вредности узастопних промена у временском низу, дефинисану као:

$$E = \sum_{i=1,\dots,n-1} \left| x_{i+1} - x_i \right|$$

`augmented_dickey_fuller` представља статистике које су излаз ADF теста (енг. *Augmented Dickey-Fuller test*). Augmented Dickey-Fuller тест је статистички тест који се може користити да се утврди да ли је у временском серијском скупу података присутан јединични корен (енг. *unit root*) [70][142]. Јединични корен је карактеристика временске серије која указује да она није стационарна, што значи да њене статистичке особине, као што су средња вредност и варијанса, нису константне током времена. Другим речима, јединични корен указује на то да временске серије имају стохастички тренд уместо детерминистичког тренда.

Нулта хипотеза ADF теста је да временска серија има јединични корен, што указује да је она нестационарна. Насупрот томе, алтернативна хипотеза је да је временска серија стационарна, што значи да нема јединичног корена. ADF тест се често користи у економетрији и финансијама да би се оценио присуство јединичног корена у економским временским серијама [143]. ADF тест се извршава по следећим корацима:

- Формулација хипотезе: Први корак је јасно дефинисање нулте и алтернативне хипотезе за ADF тест. Као што је већ поменуто, нулта хипотеза претпоставља присуство јединичног корена, што указује на то да временска серија није стационарна;
- Одабир реда кашњења: Следећи корак укључује избор одговарајућег реда помераја за тест. Ред помераја се односи на број јединичних кашњења укључених у регресиони модел. Избор реда кашњења је битан да би се адекватно ухватила аутокорелациона структура у временским серијским подацима;

- Тражење одговарајућег регресионог модела: Након одређивања реда кашњења, ADF тест захтева израчунавање регресионог модела, обично ауторегресивног модела, како би се анализирало присуство јединичног корена. Овај регресиони модел представља основу за оцењивање стационарности временске серије;
- Рачунање статистичке значајности тест: ADF тест рачуна статистичку значајност теста, која пореди процењени коефицијент из регресионог модела са критичним вредностима да би се утврдила стационарност временске серије. Величина статистичке значајности теста у односу на критичне вредности користи се за прихватање или одбацивање нулте хипотезе.

`autocorrelation_stats` представља скуп корелационих вредности које су добијене рачунањем корелације временске серија са истом том серијом, за различите вредностима закашњења (помераја).

Аутокорелација је важан концепт у анализи временских серија [142]. Она мери корелацију између временске серије и закашњене верзије себе саме. Висока аутокорелација на одређеној вредности јединичног закашњења указује да су вредности у временском низу за то закашњење повезане једна са другом. Ово може помоћи у идентификацији понављајућих образаца или сезонских трендова у подацима. Она се дефинише математички као:

$$\frac{1}{(n-l)\sigma^2}\sum_{t=1}^{n-l}(x_t-\mu)(x_{t+l}-\mu)$$

где је n дужина временске серије, l вредност закашњења, σ варијанса, μ средња вредност, тј. математичко очекивање.

`benford_correlation` представља вредност Бенфордове корелације за временски низ. Бенфордова корелација је статистичка техника која се користи за анализу усаглашености нумеричких података са Бенфордовим законом [144][145]. Бенфордов закон, такође познат као закон првог знака, тврди да је у многим природним скуповима података вероватно да ће почетна цифра било ког броја бити мала. Конкретно, предвиђа се да ће цифра 1 бити почетна око 30% времена, а вероватноћа за сваку следећу цифру експоненцијално опада. У случају овог скупа података, бројеви чије су цифре посматране су бројеви бајтова по секунди у временској серији.

У контексту Бенфордове корелације, циљ је да се процени колико је добро дати скуп нумеричких вредности усклађен са очекиваном расподелом коју је предложио Бенфордов

закон. Ова анализа укључује израчунавање фреквенције појављивања почетних цифара у скупу података и упоређивање са очекиваним фреквенцијама на основу Бенфордовог закона.

Бенфордова корелација има примену у различитим областима као што су форензичко рачуноводство, откривање малверзација и верификација целовитости података. Посебно је корисна за идентификацију аномалија и нерегуларности у великим скуповима података, јер одступања од Бенфордовог закона могу указивати на потенцијалну манипулацију или грешке у подацима.

У пракси, Бенфордова корелација укључује израчунавање стварних и очекиваних фреквенција водећих знакова у скупу података, а затим примењује статистичке тестове за оцену корелације. Технике као што су хи-квадрат тестови или Пирсонови коефицијенти корелације често се користе за квантификацију степена усаглашености са Бенфордовим законом.

`cid_ce` представља меру растојања инваријантне сложености (енг. *Complexity-invariant Distance*, скраћено CID).

Мера растојања инваријантне сложености је статистичка метода која се користи за квантификацију сложености временских серијских података [146]. За разлику од традиционалних мера растојања које су осетљиве на скалу и величину података, мера CID фокусира се на захватање сложености и структуре временских серија, што је посебно корисно за поређење и анализу временских серија са различитим нивоима сложености.

Мера CID заснива се на концепту пермутационе ентропије, која квантификује неправилност и непредвидљивост временске серије рачунањем ентропије њених ординалних шаблона. Ординални шаблони су секвенце вредности које се добијају из релативног поретка тачака података у одређеној величини прозора. Рачунањем пермутационе ентропије ових ординалних шаблона, мера CID пружа робусну и инваријантну меру у односу на скалу оцене сложености присутне у временској серији.

Додатно, мера CID је предност у сценаријима где традиционалне мере растојања могу дати погрешне резултате због разлика у скали или јединицама временских серијских података. Њена способност да детектује дубинску комплексност временских низова, уз то да има велику толеранцију на различите скале или амплитуде скупа података.

`count_above_mean` представља број вредности временске серије које су веће од средње вредности.

`count_below_mean` представља број вредности временске серије које су мање од средње вредности.

`fft_agregated` представља статистичке параметре FFT трансформације (енг. *Fast Fourier Transform*, *FFT*) над временском серијом. Статистички параметри које се екстрахују су средња вредност, варијанса, коефицијент симетрије и куртозис (коефицијент тежине репа дистрибуције).

`first_location_of_maximum` представља временски тренутак када се први пут јавља максимална вредност у временској серији.

`first_location_of_minumum` представља временски тренутак када се први пут јавља минимална вредност у временској серији.

`fourier_entropy` представља ентропију густине спектралне снаге (скраћено ГСС) временске серије.

Фуријеова ентропија је мера која се користи за квантификацију количине информација или предвидљивости присутне у представи у фреквенцијском домену сигнала која се добија брзом Фуријеовом трансформацијом (енг. F*ast Fourier Transform, FFT*) [142]. Она пружа увид у расподелу енергије преко различитих фреквенција и ниво неуређености или насумичности унутар фреквенцијских компоненти сигнала. Израчунавање Фуријеове ентропије укључује процену расподеле Фуријеових коефицијената или густина спектралне снаге и одређивање степена неизвесности или предвидљивости у фреквенцијском домену датог сигнала.

tsfresh библиотека користи Велчов метод [94] за израчунавање Фуријеове ентропије који укључује деобу временских серијских података на преклапајуће сегменте, примену тзв. *прозорске функције* (енг. *window function*) на сваки сегмент како би се смањило тзв. *спектрално расипање* (енг. *spectral leak*), а затим израчунавање густина спектралне снаге (даље у тексту ГСС) за сваки сегмент коришћењем брзе Фуријеове трансформације [147]. ГСС представља расподелу енергије преко различитих фреквенција у скупу података. Након што се добије ГСС за сваки сегмент, Велчов метод рачуна просеке спектралне процене из преклапајућих сегмената како би се добило *глаткији* и поузданији приказ спектра снаге. Овај процес упросечавања помаже умањењу варијансе спектралних процена и обезбеђује тачнији приказ карактеристика у фреквенцијском домену сигнала. Када се добије просечна ГСС, Фуријеова ентропија може се израчунати процењивањем расподеле Фуријеових коефицијената или густина спектралне снаге. Мера ентропије квантификује ниво

неизвесности, случајности или предвидљивости у фреквенцијским компонентама сигнала, пружајући вредне увиде у динамику и шаблоне присутне у фреквенцијском домену датог сигнала.

`has_duplicate` представља одлику која бележи да ли има дупликата у временској серији.

`has_duplicate_max` представља одлику која бележи да ли има дупликата максималне вредности у временској серији.

`has_duplicate_min` представља одлику која бележи да ли има дупликата минималне вредности у временској серији.

`index_mass_quantile` представља релативни индекс од којег се одређени квинтил вредности временске серије налази лево.

`kurtosis` представља статистичку меру која квантификује тежину репа расподеле вероватноће. Он пружа увиде у облик открива информације о присуству екстремних вредности и концентрацији података око средње вредности. Већа вредност куртозиса указује на *теже* репове, односно да екстремне вредности имају већу вероватноћу појављивања, док нижа вредност куртозиса указује на *лакше* репове и расподелу концентрисанију око средње вредности. Математичка формула куртозиса је:

$$Kurt[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^4\right]$$

где је X случајна променљива, σ варијанса, μ средња вредност, тј. математичко очекивање.

`large_standard_deviation` представља одлику која бележи да ли је стандардна девијација временске серије већа од процентуалне разлике минимума и максимума. Дефинише се као:

$$std(X) > r \cdot (max(X) - min(X))$$

где је X случајна променљива, r представља број од 0 до 1 који дефинише проценат разлике, *min* и *max* минималну и максималну вредност, а *std* стандардну девијацију.

`last_location_of_maximum` представља индекс на којем се јавља последња максимална вредност.

`last_location_of_minumum` представља индекс на којем се јавља последња минимална вредност.

`lempel_ziv_complexity` представља вредност Лемпел-Зив комплексности за временску серију. Лемпел-Зив комплексност, такође позната као LZ комплексност, је мера која се користи за квантификацију комплексности и одсуство правилности низа или временске серије [148]. Заснована је на Лемпел-Зив алгоритму, који је првобитно дизајниран за компресију података, али је нашао примену у области рачунске сложености и анализе временских серија.

Лемпел-Зив алгоритам ради тако што идентификује понављајуће обрасце или поднизове унутар улазног низа и кодира их помоћу референтних показивача на њихове прве појаве. Сложеност низа се затим одређује укупним бројем различитих образаца или поднизова идентификованих током процеса кодирања. У контексту анализе временских серија, Лемпел-Зив комплексност се може применити да би се оценила нерегуларност и комплексност временских образаца присутних у подацима. Виша вредност комплексности Лемпел-Зива указује на више нерегуларних и комплексних временских серија, док нижа вредност сугерише на предвидивост вредности у временској серији.

`length` представља дужину временске серије.

`linear_trend` представља скуп статистичких мера који се могу екстраховати примењујући линеарну регресију на временску серију [142]. Линеарна регресија методом најмање разлике квадрата је основни статистички метод који се користи за моделирање односа између зависне променљиве и једне или више независних променљивих. Циљ је пронаћи најбоље прилагођену линеарну једначину која описује однос између променљивих, минимизирајући збир квадрата разлика између посматраних и предвиђених вредности. Овај приступ омогућава процену коефицијената линеарне једначине, што омогућава увиде у јачину и смер односа између променљивих.

Примена линеарне регресије методом најмање разлике квадрата проширује се на анализу временских серија олакшавајући моделирање и предвиђање будућих вредности на основу историјских података. Фитовањем линеарног регресионог модела на податке временских серија, може се проценити значај независних променљивих и њихов утицај на зависну променљиву током времена. Додатно, линеарна регресија омогућава идентификацију трендова, узорака и асоцијација унутар временских серија, што помаже у развоју методологија прогнозирања и предиктивних модела.

`longest_strike_above_mean` представља дужину највећег низа чији чланови имају вредности више од средње вредности.

`longest_strike_below_mean` представља дужину највећег низа чији чланови имају вредности ниже од средње вредности.

`maximum` представља вредност максимума временске серије.

`mean` представља средњу вредност временске серије.

`mean_abs_change` представља средњу вредност апсолутних разлика суседних чланова временске серије. Дефинише се као:

$$\frac{1}{n-1}\sum_{i=1,\dots,n-1}\left|x_{i+1}-x_i\right|$$

где n представља дужину низа, а x_i i-ти сабирак низа.

Ова мера пружа увид у просечно растојање података од средње вредности, пружајући информације о општој варијабилности и флуктуацијама у временским серијама. Разматрајући апсолутне разлике, у обзир се узимају и позитивна и негативна одступања од средње вредности, пружајући робусну индикацију дисперзије података. Средња апсолутна разлика је посебно корисна за разумевање типичне величине флуктуација у временским серијама, што може бити важно за оцењивање опште стабилности и конзистентности посматране променљиве.

`mean_change` представља средњу вредност разлика суседних чланова временске серије. Дефинише се као:

$$\frac{1}{n-1}\sum_{i=1,\dots,n-1}x_{i+1}-x_i$$

где n представља дужину низа, а x_i i-ти сабирак низа.

Ова мера пружа увид у просечни правац и величину промене између узастопних тачака података, пружајући информације о општем тренду и правцу временских серија. Рачунањем средње вредности разлика, може се оценити постојана промена променљиве у праћеном периоду, пружајући важне увиде у динамику и шаблоне присутне у временским серијама.

`mean_n_absolute_max` представља аритметичку средину фиксног броја максималних вредности временске серије.

`mean_second_derivative_central` представља средњу вредност централне апроксимације другог извода временске серије. Дефинише се као:

$$\frac{1}{2\cdot(n-2)}\sum_{i=1,\ldots,n-1}\frac{1}{2}\left(x_{i+2}-2\,x_{i+1}+x_i\right)$$

где n представља дужину низа, а x_i i-ти сабирак низа.

`median` представља медијану временске серије. У анализи временских серија, медијана служи као још једна битна мера централне тенденције која може пружити важне увиде у расподелу праћене променљиве током времена. За разлику од средње вредности, која је осетљива на екстремне вредности или одступања, медијана представља средњу вредност скупа података када су подаци постављени у растућем редоследу.

`minimum` представља најмању вредност у временској серији.

`percentage_of_reocurring_datapoints_to_all_datapoints` представља однос броја тачака података које имају вредности које се понављају унутар временске серије и укупне дужине временске серије.

`percentage_of_reocurring_values_to_all_values` представља однос броја вредности које се понављају унутар временске серије и укупног броја различитих вредности временске серије.

`quantile` представља вредност одабраног квантила расподеле временске серије. Квантил се дефинише као вредност низа која је већа од дефинисаног постотка вредности у низу. Квантили пружају увиде у распрострањеност података и посебно су корисни за описивање варијабилности и опсега праћене променљиве у различитим временским интервалима. Анализа различитих квантила може понудити комплетније разумевање дистрибуције и централног понашања података временских серија, омогућавајући утврђивање конкретних тачака од интереса и процену различитих нивоа екстремности или централности унутар скупа података.

`ratio_beyond_r_sigma` представља проценат вредности временског низа које су веће од дефинисаног умношка стандардне девијације.

`ratio_value_number_to_time_series_length` представља количник броја јединствених вредности у временској серији и дужине временске серије.

`root_mean_square` представља средњу квадратну вредност (енг. *Root Mean Square,* скраћено *RMS*) временске серије. Дефинише се као:

$$RMS = \sqrt{\frac{1}{n}\sum_i x_i^2}$$

где n представља дужину низа, а x_i i-ти сабирак низа.

RMS је посебно користан за разумевање општег варијабилитета и амплитуде праћене промењиве током одређеног периода.

`sample_entropy` представља ентропију узорка временске серије. Она представља меру комплексности која количински оцењује регуларност и непредвидљивост флуктуација током временских серија. Дефинише се као негативни природни логаритам условне вероватноће да две сличне секвенце за одређени број тачака остану сличне у наредној тачки, искључујући самопоклапања. Нижа вредност ентропије узорка показује већу сличност са самим собом или регуларност у подацима, док виша вредност представља већу комплексност или нерегуларност . У временској серији, ентропија узорка је посебно корисна за следеће циљеве:

- оцена комплексности: ентропија узорка може се користити за оцену нивоа комплексности и нерегуларности у подацима временских серија. Више вредности узорачке ентропије указују на већу непредвидљивост и нерегуларност, док ниже вредности наговештавају регуларне шаблоне у подацима.
- препознавање шаблона: Рачунањем ентропије узорка за различите сегменте временске серије, могу се идентификовати шаблони или трендови који приказују различите степене комплексности. Ово може помоћи у идентификацији и карактеризацији нестандардног понашања или динамике у подацима.
- откривање аномалија: Узорачка ентропија може се користити за откривање аномалија или необичних шаблона у подацима временских серија. Нагле промене или неочекиване флуктуације у вредностима ентропије узорка могу указивати на потенцијалне аномалије које захтевају додатна истраживања.

У низу дужине N се бира узорак дужине m. Потом се, за узорке дужине $m + 1$ и m рачуна број парова чија је дистанца (за задату функцију дистанце) мања од задате толеранције r. Негативни логаритам количника ова два броја представља ентропију узорка. Математичка формула ентропије узорка гласи:

$$SampEn = -\ln\left(\frac{A}{B}\right),$$

$$A - \textit{број парова низа дужине } m + 1, \textit{ чија је дистанца мања од } r,$$
$$B - \textit{број парова низа дужине } m, \textit{ чија је дистанца мања од } r$$

`skewness` представља меру симетрије дистрибуције временске серије. Пружа увид у облик дистрибуције, указујући да ли су подаци концентрисани више на једној страни у односу на другу. Позитиван коефицијент симетрије показује да је реп на дистрибуцији дужи на десној страни, што указује на могуће екстремне вредности. С друге стране, негативан коефицијент симетрије показује да је реп на дистрибуцији дужи на левој страни, што може указивати на могуће аутлајере или екстремно ниске вредности. Математичка формула мере симетрије гласи:

$$Skew[X] = E\left[\left(\frac{X-\mu}{\sigma}\right)^3\right]$$

где је X случајна променљива, σ варијанса, μ средња вредност, тј. математичко очекивање.

`standard_deviation` представља стандардну девијацију временске серије.

`sum_of_reoccurring_data_points` представља суму тачака података које се понављају унутар временске серије.

`sum_of_reoccurring_values` представља суму вредности које се понављају унутар временске серије.

`sum_values` представља суму свих вредности које се појављују у временској серији.

`symmetry_looking` представља одлику која бележи да ли је апсолутна разлика средње вредности и медијане мања од умношка разлике максимума и минимума. Представља меру симетричности расподеле временске серије, дефинисану у односу на умножак.

`variance` представља варијансу временске серије.

`variance_larger_than_standard_deviation` представља одлику која бележи да ли је варијанса временске серије већа од стандардне девијације.

`variation_coefficient` представља коефицијент варијације временске серије. Коефицијент варијације, је бездимензионална мера која исказује стандардну девијацију као проценат средње вредности. Он пружа релативну меру расејавања која омогућава упоређивање различитих скупова података, без обзира на њихове редове величина. Коефицијент варијације је посебно користан за следеће циљеве у анализи временских серија:

- Релативна варијабилност: Рачунањем коефицијента варијације за временску серију, може се проценити релативна варијабилност података, узимајући у обзир средњу вредност. Ово омогућава упоређивање стопе промене између различитих временских интервала или променљивих, пружајући увид у њену конзистентност у односу на средњу вредност;
- Стандардизација расејања: Изразивши стандардну девијацију као проценат средње вредности користећи коефицијент варијације стандардизује меру расејања, чинећи је лакшом за тумачење и упоређивање различитих скупова података са различитим редовима величина;
- Процена стабилности: Нижи коефицијент варијације указује на стабилнију и конзистентнију расподелу података око средње вредности, док виши коефицијент варијације указује на већу релативну варијабилност и расејање. Будући да ботнети имају предвидив образац размене порука истих величина, расподела таквих временских низова има мању варијабилност.

Литература

1: Vormayr, G., Zseby, T., Fabini, J.: Botnet Communication Patterns, IEEE Commun. Surv. Tutorials, vol. 19, no. 4, pp. 2768–2796, 10.1109/COMST.2017.2749442 (2017)

2: Chen, R., Niu, W., Zhang, X., Zhuo, Z., Lv, F.: An Effective Conversation-Based Botnet Detection Method, Math. Probl. Eng., vol. 2017, pp. 1–9, 10.1155/2017/4934082 (2017)

3: Cloudflare,Famous DDoS Attacks, https://www.cloudflare.com/learning/ddos/famous-ddos-attacks/ (приступљено 4.6.2024.)

4: Akamai, FS-ISAC, The Evolution of DDoS: Return of the Hacktivists, https://www.akamai.com/site/en/documents/research-paper/the-evolution-of-ddos-return-of-the-hacktivists.pdf (приступљено 4.6.2024.)

5: Akamai, Ransomware on the Move - Evolving Exploitation Techniques and the Active Pursuit of Zero-Days, https://www.akamai.com/blog/security/ransomware-on-the-move-evolving-exploitation-techniques (приступљено 4.6.2024.)

6: B. Krebs, Zyxel Flaw Powers New Mirai IoT Botnet Strain, https://krebsonsecurity.com/2020/03/zxyel-flaw-powers-new-mirai-iot-botnet-strain/ (приступљено 4.6.2024.)

7: Štampar, M., Fertalj, K.: Applied Machine Learning in Recognition of DGA Domain Names, Computer Science and Information Systems, vol. 19, No. 1, 205-227., 10.2298/CSIS210104046S (2022)

8: Jovanović Đ., Vuletić P.: Analysis and Characterization of IoT Malware Command and Control Communication, Telfor Journal Vol.12 No.2, p. 80-85, 10.5937/telfor2002074B (2020)

9: Ibrahim, J., Gajin, S.: Entropy-based Network Traffic Anomaly Classification Method Resilient to Deception. Computer Science and Information Systems, Vol. 19, No. 1, p. 87-116., 10.2298/CSIS201229045I (2022)

10: Jovanović Đ., Vuletić P.: PI-BODE: Programmable Intraflow-based IoT Botnet Detection system, Computer Science and Information Systems, Vol. 21, No. 1, 37-56, 10.2298/CSIS211116064J (2024)

11: Tyagi, A. K.,Aghila, G.: A Wide Scale Survey on Botnet, International Journal of Computer Applications, Vol. 34, No. 9, 9-22, 10.5120/4126-5948 (2011)

12: Osagie, Maxwell Scale Uwadia, Enagbonma, Osatohanmwen, Inyang, Amanda Iriagbonse: The Historical Perspective of Botnet Tools, Current Journal of Applied Science and Technology, Vol. 32, No. 6, p. 1-8, 10.9734/cjast/2019/v32i630040 (2019)

13: Binsalleeh, H., Ormerod, T., Boukhtouta, A., Sinha, P., Youssef, A., Debbabi, M., Wang, L.: On the analysis of the Zeus botnet crimeware toolkit, 2010 Eighth International Conference on Privacy, Security and Trust, pp. 31-38, 10.1109/PST.2010.5593240 (2010)

14: Cloudflare, Introducing Cloudflare's 2023 phishing threats report, https://blog.cloudflare.com/2023-phishing-report/ (приступљено 4.6.2024.)

15: Fortinet, Watering Hole Attack, https://www.fortinet.com/resources/cyberglossary/watering-hole-attack (приступљено 4.6.2024.)

16: Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J.A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z.: Understanding the Mirai botnet, SEC'17, p. 1093–1110., 10.5555/3241189.3241275 (2017)

17: Zi Chu, Steven Gianvecchio, Haining Wang, Sushil Jajodia, Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?, IEEE Transactions on Dependable and Secure Computing, Vol. 9, No. 6, p. 811-824 10.1109/TDSC.2012.75 (2012)

18: The Hacker News, Microsoft Hijacks Necurs Botnet that Infected 9 Million PCs Worldwide, https://thehackernews.com/2020/03/necurs-botnet-takedown.html (приступљено 04.06.2024.)

19: trendmicro.com, Mining Worm Goes Polymorphic, Gets AutoHotKey Variant, https://www.trendmicro.com/en_us/research/18/d/monero-mining-retadup-worm-goes-polymorphic-gets-an-autohotkey-variant.html(приступљено 04.06.2024.)

20: bleepingcomputer.com, Russian Methbot Operation Makes up to $5 Million per Day from Click-Fraud, https://www.bleepingcomputer.com/news/security/russian-methbot-operation-makes-up-to-5-million-per-day-from-click-fraud/ (приступљено 4.6.2024.)

21: Lumen Blog, A Look Inside The TrickBot Botnet, https://blog.lumen.com/a-look-inside-the-trickbot-botnet/ (приступљено 4.6.2024.)

22: zdnet - Dancho Danchev, Fast-Fluxing SQL injection attacks executed from the Asprox botnet, https://www.zdnet.com/article/fast-fluxing-sql-injection-attacks-executed-from-the-asprox-botnet/ (приступљено 4.6.2024.)

23: zdnet – Charlie Osborne, MikroTik routers enslaved in massive Coinhive cryptojacking campaign, https://www.zdnet.com/article/mikrotik-routers-enslaved-in-massive-coinhive-cryptojacking-campaign/ (приступљено 4.6.2024.)

24: cyberscoop, The anatomy of the MyKings botnet, and why it matters for security, https://cyberscoop.com/mykings-botnet-sophos-smominru/ (приступљено 4.6.2024.)

25: Holz T. et al.:Measurements and Mitigation of Peer-to-Peer-based Botnets: A Case Study on Storm Worm, Proc. 1st Usenix Workshop on Large-scale Exploits and Emergent, pp. 1-9, (2008)

26: Grizzard J. B, et al.: Peer-to-peer botnets: Overview and case study, HotBots'07: Proceedings of the first conference on First Workshop on Hot Topics in Understanding Botnets, pp. 1-8, (2007)

27: Tanenbaum, A, Wetherall, D.: Computer Networks (5th Edition), ESBN-13: 978-0-13-212695-3 (2010)

28: Žiža, K., Tadić, P., Vuletić, P.: DNS exfiltration detection in the presence of adversarial attacks and modified exfiltrator behaviour, International Journal of Information Security, Vol. 22, No. 6, p. 1-16, 10.1007/s10207-023-00723-w (2023)

29: Akamai, What is DNS data exfiltration?, https://www.akamai.com/glossary/what-is-dns-data-exfiltration (приступљено 04.06.2024.)

30: Dietrich, C.J.,Feederbot Botnet Using DNS as Carrier for Command and Control (C2), https://chrisdietri.ch/post/feederbot-botnet-using-dns-command-and-control/ (приступљено 04.06.2024.)

31: SCMagazine, Morto using DNS for command-and-control, https://www.scmagazine.com/news/morto-using-dns-for-command-and-control (приступљено

06.04.2024.)

32: Plohmann, D., Yakdan, K., Klatt, M., Bader, J., Gerhards-Padilla, E.: A Comprehensive Measurement Study of Domain Generating Malware, Open access to the Proceedings of the 25th USENIX Security Symposium is sponsored by USENIX, pp. 1996–2014, 10.5555/3241094.3241115 (2016)

33: Dave Piscitello, Conficker Summary and Review, https://www.icann.org/en/system/files/files/conficker-summary-review-07may10-en.pdf (приступљено 06.04.2024.)

34: The Tor Project, Tor Official Website, https://www.torproject.org/ (приступљено 06.04.2024.)

35: netlab360, ZHTrap, https://blog.netlab.360.com/new_threat_zhtrap_botnet_en/ (приступљено 06.04.2024.)

36: Ars Technica, Tor traffic disguised as Skype video calls to fool repressive governments, https://arstechnica.com/tech-policy/2012/04/tor-traffic-disguised-as-skype-video-call-to-fool-repressive-governments/ (приступљено 06.04.2024.)

37: Houmansadr A. et al.: The Parrot is Dead: Observing Unobservable Network Communications, 2013 IEEE Symposium on Security and Privacy, pp. 65-79, 10.1109/SP.2013.14 (2013)

38: abuse.ch, URLHaus, a database of malware URLs, https://urlhaus.abuse.ch/ (приступљено 06.04.2024.)

39: Security Affairs, Urlhaus Identified and Shut Down 100,000 Malware Sites in 10 Months,

40: Cuckoo Foundation,Cuckoo Documentation, https://securityaffairs.com/80200/malware/urlhaus-malware-sites.html (приступљено 04.06.2024.) https://cuckoo.readthedocs.io/en/latest/ (приступљено 04.06.2024.)

41: Gardiner, J., Cova, M., Nagaraja, S.: Command & Control: Understanding, Denying and Detecting, vol. cs.CR, no. February, 10.48550/arXiv.1408.1136 (2014)

42: NSA, Ghidra Homepage, https://www.ghidra-sre.org/ (приступљено 11.06.2024.)

43: Asadi, M., Jabraeil Jamali, M. A., Parsa, S., Majidnezhad, V.: Detecting botnet by using particle swarm optimization algorithm based on voting system. Future Generation Computer Systems, vol. 107, 95–111., 10.1016/j.future.2020.01.055 (2020)

44: Livadas C., Walsh, R., Lapsley, D.E., Strayer, W.T.: Using machine learning techniques to identify botnet traffic, LCN, pp. 967–974., 10.1109/LCN.2006.322210 (2006)

45: Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset, Future Generation Computer Systems, vol. 100, p. 779-796, 10.1016/j.future.2019.05.041 (2019)

46: Lee, J.-S., Jeong, H., Park, J.-H., Kim, M., Noh, B.-N.: The activity analysis of malicious http-based botnets using degree of periodic repeatability, 2008 International Conference on Security Technology, pp. 83–86., 10.1109/SecTech.2008.52 (2008)

47: Eslahi, M., Rohmad, M. S., Nilsaz, H., Naseri, M. V., Tahir, N. M., Hashim, H.: Periodicity classification of HTTP traffic to detect HTTP Botnets, 2015 IEEE Symposium on Computer

Applications & Industrial Electronics (ISCAIE), p. 119–123. 10.1109/ISCAIE.2015.7298339 (2015)

48: Wang, W., Shang, Y., He, Y., Li, Y., Liu, J.: BotMark: Automated botnet detection with hybrid analysis of flow-based and graph-based traffic behaviors, Information Sciences, Vol. 511, p. 284–296. 10.1016/j.ins.2019.09.024 (2020)

49: Cusack, G., Michel, O., Keller, E.: Machine Learning-Based Detection of Ransomware Using SDN, In Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks & Network Function Virtualization - SDN-NFV Sec'18, pp. 1–6., 10.1145/3180465.3180467 (2018)

50: Shahzana Liaqat, S., et al.: SDN orchestration to combat evolving cyber threats in Internet of Medical Things (IoMT), Computer Communications, Volume 160, p. 697-705, 10.1016/j.comcom.2020.07.006 (2020)

51: Bilge, L. et al.: DISCLOSURE: Detecting Botnet Command and Control Servers Through Large-Scale NetFlow Analysis, ACSAC '12, 10.1145/2420950.2420969 (2012)

52: Hofstede, R. et al.:Flow Monitoring Explained: From Packet Capture to Data Analysis with NetFlow and IPFIX, IEEE Communications Surveys & Tutorials,16,,2037–2064,(2014)

53: Blaise, A., Bouet, M., Conan, V., Secci, S.: Detection of zero-day attacks: An unsupervised port-based approach, Computer Networks, vol.180, 10.1016/j.comnet.2020.107391 (2020)

54: De La Torre Parra, G., Rad, P., Choo, K.-K. R., Beebe, N.: Detecting Internet of Things attacks using distributed deep learning, Journal of Network and Computer Applications, Vol. 163, 10.1016/j.jnca.2020.102662 (2020)

55: Kurniabudi, et al.: CICIDS-2017 Dataset Feature Analysis With Information Gain for Anomaly Detection, IEEE Access, Volume 8, p. 132911-132921, 10.1109/ACCESS.2020.3009843 (2019)

56: Sharifnya, R., Abadi, M.: DFBotKiller: Domain-flux botnet detection based on the history of group activities and failures in DNS traffic, Digit. Investig., vol. 12, pp. 15–26, 10.1016/j.diin.2014.11.001 (2015)

57: Tong, V., Nguyen, G.: A method for detecting DGA botnet based on semantic and cluster analysis, ACM Int. Conf. Proceeding Ser., vol. 08, pp. 272–277, 10.1145/3011077.3011112 (2016)

58: Wang, T. S., Lin, H. T., Cheng, W. T., Chen, C. Y.: DBod: Clustering and detecting DGA-based botnets using DNS traffic analysis, Computer Security, vol. 64, pp. 1–15, 10.1016/j.cose.2016.10.001 (2017)

59: Al-Hadhrami, Y., Hussain, F. K.: Real time dataset generation framework for intrusion detection systems in IoT, Future Generation Computer Systems, Vol. 108, p. 414–423., 10.1016/j.future.2020.02.051 (2020)

60: de Souza, C. A., et al.: Hybrid approach to intrusion detection in fog-based IoT environments, Computer Networks, 180, 107417., 10.1016/j.comnet.2020.107417 (2020)

61: Hosseini, S., Zade, B. M. H.: New hybrid method for attack detection using combination of evolutionary algorithms, SVM, and ANN, Computer Networks, Vol. 173, 10.1016/j.comnet.2020.107168 (2020)

62: Zhou, Y., Cheng, G., Jiang, S., Dai, M.: Building an efficient intrusion detection system based on feature selection and ensemble classifier, Computer Networks, 174, 10.1016/j.comnet.2020.107247 (2020)

63: Shafiq, M., et al.: Selection of effective machine learning algorithms and Bot-IoT attacks traffic identification for internet of things in smart city, Future Generation Computer Systems, Vol. 107, p. 433–442. 10.1016/j.future.2020.02.017 (2020)

64: Parmisano, A., Garcia, S., Erquiaga, M. J.: A labeled dataset with malicious and benign IoT network traffic. Stratosphere Laboratory, 10.5281/zenodo.4743746 (2020)

65: Jovanovic, G., Vuletić, P.: ETF IoT Botnet Dataset, Mendeley Data, V1, 10.17632/nbs66kvx6n.1 (2021)

66: García et al.: An Empirical Comparison of Botnet Detection Methods, Computers & Security, Vol. 45, p. 100-123, 10.1016/j.cose.2014.05.011 (2014)

67: Stratosphere labs,Aposemat IoT Project, https://www.stratosphereips.org/aposemat (приступљено 11.06.2024.)

68: Cisco, Joy: A package for capturing and analyzing network flow data and intraflow data, for network research, forensics, and security monitoring, https://developer.cisco.com/codeexchange/github/repo/cisco/joy/ (приступљено 4.6.2024.)

69: Open Networking Foundation,OpenFlow Switch Specification, Version 1.5.1, ONF TS-025, https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.5.1.pdf (приступљено 6.4.2024.)

70: Dickey, D. A., Fuller, W. A.:Distribution of the Estimators for Autoregressive Time Series with a Unit Root, Journal of the American Statistical Association, Vo. 74, No. 366, p. 427-431, 10.2307/2286348 (1979)

71: Raschka, S. , About Feature Scaling, https://sebastianraschka.com/Articles/2014_about_feature_scaling.html (приступљено 4.6.2024.)

72: sklearn, sklearn documentation, https://scikit-learn.org/stable (приступљено 4.6.2024.)

73: Skiena, S. S.: The Data Science Design Manual, ISBN-13: 978-3-319-55444-0 (2017)

74: Chris Albon: Python Machine Learning Cookbook: Practical Solutions from Preprocessing to Deep Learning, ISBN-13: 978-1098135720 (2023)

75: Bentley, J.L.: Multidimensional Binary Search Trees Used for Associative Searching,

Communications of the ACM, Vol. 18, No. 9, p. 509 – 517, 10.1145/361002.361007 (1975)

76: Omohundro, S.M.: Five Balltree Construction Algorithms, International Computer Science Institute, Berkeley, TR-89-063 (1989)

77: Liu, Y.: Python Machine Learning By Example, ISBN-13: 978-1800209718 (2020)

78: Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, Chih-Jen Lin: LIBLINEAR: A Library for Large Linear Classification, Journal of Machine Learning Research,Vol. 9, p. 1871-1874, 10.1145/1390681.1442794 (2008)

79: Byrd, R. H. , Lu, P., Nocedal, J.: A Limited Memory Algorithm for Bound Constrained Optimization, SIAM Journal on Scientific Computing, Vol. 16, No. 5, 10.1137/0916069 (1995)

80: Xin-Yuan, Z., Defeng, S., Kim-Chuan, T.: A Newton-CG Augmented Lagrangian Method for Semidefinite Programming, SIAM Journal on Optimization,Vol. 20, No. 4, 10.1137/080718206 (2010)

81: Forsgren, A., Gill, P.E., Murray, W.: Computing modified Newton directions using a partial Cholesky factorization, SIAM Journal on Scientific Computing, Vol. 16, No. 1, 10.1137/0916009 1995 (1995)

82: Schmidt, M., Le Roux, N., Bach, F.: Minimizing Finite Sums with the Stochastic Average Gradient, Mathematical Programming, Vol. 162, p. 83–112, 10.48550/arXiv.1309.2388 (2017)

83: Defazio, A., Bach, F., Lacoste-Julien, S.:SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives, NIPS'14: Proceedings of the 27th International Conference on Neural Information Processing Systems, Vol. 1, pp. 1646 – 1654, 10.48550/arXiv.1407.0202 (2014)

84: Rincy, T.N., Gupta, R.: Ensemble learning techniques and its efficiency in machine learning: A survey, 2nd International Conference on Data, Engineering and Applications (IDEA), pp. 1-6, 10.1109/IDEA49133.2020.9170675 (2020)

85: Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems?, Journal of Machine Learning Research, Vol. 15, p. 3133-3181, 2014

86: Fernández, A., García, S., Galar, M., Prati, R.C., Krawczyk, B., Herrera, F.: Learning from Imbalanced Data Sets, Springer, 10.1007/978-3-319-98074-4 (2018)

87: Gibert, D., Mateu, C., Planes, J.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges, Journal of Network and Computer Applications, Vol. 153, 10.1016/j.jnca.2019.102526 (2020)

88: Milošević, M., Ćirić, V.: Extreme minority class detection in imbalanced data for network intrusion, Computers & Security, Vol. 123, 10.1016/j.cose.2022.102940 (2022)

89: Apruzzese, G., Colajanni, M., Marchetti, M.: Evaluating the effectiveness of Adversarial Attacks against Botnet Detectors, 2019 IEEE 18th International Symposium on Network Computing and Applications (NCA), 10.1109/NCA.2019.8935039 (2019)

90: Dubitzky, W., Granzow, M., Berrar, D.: Fundamentals of data mining in genomics and proteomics, Springer, ISBN-13: 978-0-387-47509-7 (2007)

91: Natekin, A., Knoll, A.: Gradient boosting machines, a tutorial, Frontiers in Neurorobotics, Vol. 7, No. 21, 10.3389/fnbot.2013.00021 (2013)

92: Sheridan et al.: Extreme Gradient Boosting as a Method for Quantitative Structure-Activity Relationships, Journal of Chemical Information and Modeling, Vol. 56, No. 12, 10.1021/acs.jcim.6b00591 (2016)

93: Jerome H. Friedman: Stochastic Gradient Boosting, Computational Statistics & Data Analysis, Vol. 38, No. 4, p. 367-378, 10.1016/S0167-9473(01)00065-2 (2002)

94: TSFresh, TSFresh: a Python package that provides automatization of feature extraction for time series, https://tsfresh.readthedocs.io/en/latest/ (приступљено 4.6.2024.)

95: Samed, A., Murat, D.:STL-HDL: A new hybrid network intrusion detection system for imbalanced dataset on big data environment, Computers and Security, Vol. 110, No. 2, 10.1016/j.cose.2021.102435 (2021)

96: Masoudi-Sobhanzadeh, Y., Emami-Moghaddam, S.:A real-time IoT-based botnet detection method using a novel two-step feature selection technique and the support vector machine classifier, Computer Networks, Vol. 217, 10.1016/j.comnet.2022.109365 (2022)

97: Aborujilah, A. et al.: SMOTE-Based Framework for IoT Botnet Attack Detection, Advances in Cyber Security, pp. 287–296, 10.1007/978-981-33-6835-4_19 (2021)

98: Mohammadi, S., Babagoli, M.:A novel hybrid hunger games algorithm for intrusion detection, International Journal of Information Security, Vol. 22, No. 5, p. 1177–1195, 10.1007/s10207-023-00684-0 (2023)

99: Bojarajulu, B., Tanwar, S.:Customized convolutional neural network model for IoT botnet attack detection, Signal, Image and Video Processing,Vol. 18, No. 6-7, p. 5477–5489, 10.1007/s11760-024-03248-4 (2024)

100: Rust-Nguyen, N., Sharma, S., Stamp, M.: Darknet traffic classification and adversarial attacks using machine learning, Computers & Security, Vol. 127, No. 1, 10.1016/j.cose.2023.103098 (2023)

101: Rustam, F., Delia Jurcut, A.: Malicious traffic detection in multi-environment networks using novel S-DATE and PSO-D-SEM approaches, Computers & Security, Vol. 136, No. 5, 10.1016/j.cose.2023.103564 (2024)

102: Qing, Y., Liu, X., Du Y.: Mitigating data imbalance to improve the generalizability in IoT DDoS detection tasks, The Journal of Supercomputing, Vol. 80, No. 7, p. 9935–9960, 10.1007/s11227-023-05829-5 (2024)

103: Alfrhan, A. et al., SMOTE: Class Imbalance Problem In Intrusion Detection System, 2020 International Conference on Computing and Information Technology (ICCIT-1441), 10.1109/ICCIT-144147971.2020.9213728 (2020)

104: Gonzalez-Cuautle, D. et al.: Synthetic Minority Oversampling Technique for Optimizing Classification Tasks in Botnet and Intrusion-Detection-System Datasets, Applied Sciences, Vol. 10, No. 3, 10.3390/app10030794 (2020)

105: Han, H., Wang, W.-Y., Mao, B.-H.: Borderline-SMOTE: A New Over-Sampling Method in Imbalanced Data Sets Learning, Lecture Notes in Computer Science, Vol. 3644, No. 5, pp. 878-887, 10.1007/11538059_91 (2005)

106: Chawla, N. V. et al.: SMOTE: Synthetic Minority Over-sampling Technique, Journal of Artificial Intelligence Research, Vol. 16, No. 1, p. 321–357, 10.1613/jair.953 (2002)

107: Wilson, D. L.: Asymptotic Properties of Nearest Neighbor Rules Using Edited Data, IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, Vol. 2, No. 3, p. 408-421, 10.1109/TSMC.1972.4309137 (1972)

108: He, H. et al.: ADASYN: Adaptive Synthetic Sampling Approach for Imbalanced Learning, 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), 10.1109/IJCNN.2008.4633969 (2008)

109: imbalanced-learn, Imbalanced-learn: a Python library which implements algorithms that work with imbalanced data, https://imbalanced-learn.org/stable/index.html (приступљено 6.4.2024.)

110: Molnar, C., Interpretable Machine Learning: A Guide for Making Black Box Models Explainable, ISBN-13: 979-8411463330 (2023)

111: Lundberg, Scott M., Lee, Su-In: A unified approach to interpreting model predictions, NIPS 2017, 10.48550/arXiv.1705.07874 (2017)

112: Chen, Xue-wen, Jeong, Jong Cheol: Enhanced Recursive Feature Elimination, Sixth International Conference on Machine Learning and Applications, 10.1109/ICMLA.2007.35 (2007)

113: Tarfa Hamed, Recursive Feature Addition: a Novel Feature Selection Technique, Including a Proof of Concept in Network Security, PhD Thesis (2017)

114: Kursa, Miron B., Jankowski, A., Rudnicki, Witold R.: Boruta – A System for Feature Selection, Fundamenta Informaticae, Vol. 101, No. 4, p. 271-285, 10.3233/FI-2010-288 (2010)

115: Kursa, Miron B., Rudnicki, Witold R., Feature Selection with the Boruta Package, Journal of Statistical Software, Vol. 36, No. 11, p. 1-13, 10.18637/jss.v036.i11 (2010)

116: cerlymarco, Shap-hypertune: A python package for simultaneous Hyperparameters Tuning and Features Selection for Gradient Boosting Models, https://github.com/cerlymarco/shap-hypetune (приступљено 6.4.2024.)

117: Watanabe, S.: Tree-Structured Parzen Estimator: Understanding Its Algorithm Components and Their Roles for Better Empirical Performance, https://www.semanticscholar.org/reader/28bcb8383fde12bc4c243fcb179c3425bedc889f (приступљено 6.4.2024.)

118: Nguyen, D. A. et al.: Improved automated CASH optimization with tree parzen estimators for class imbalance problems, 2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA), 10.1109/DSAA53316.2021.9564147 (2021)

119: Bergstra, J., Yamins, D., Cox, D. D.: Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures, Proceedings of the 30th International Conference on Machine Learning, pp. 115-123, (2013)

120: Ke et al.: LightGBM: A Highly Efficient Gradient Boosting Decision Tree, 31st Conference on Neural Information Processing Systems (NIPS 2017), pp. 3149-3157, (2017)

121: Chen, T., Guestrin, C.: XGBoost: A Scalable Tree Boosting System, KDD '16: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 785 – 794, 10.1145/2939672.2939785 (2016)

122: Bentéjac et al.: A Comparative Analysis of XGBoost, 10.1007/s10462-020-09896-5 (2019)

123: Sagi, O., Rokach, L.: Approximating XGBoost with an interpretable decision tree, Information Sciences, Vol. 572, p. 522-542, 10.1016/j.ins.2021.05.055 (2021)

124: Jurkiewicz, P. et al.: Flow length and size distributions in campus Internet traffic, Computer Communications, Vol. 167, No. 1, p. 15-30, 10.1016/j.comcom.2020.12.016 (2020)

125: cleafy.com, Digital banking fraud: how the Gozi malware works, https://www.cleafy.com/cleafy-labs/digital-banking-fraud-how-the-gozi-malware-work

(приступљено 6.4.2024.)

126: blackhatethicalhacking.com, Necurs: Uncovering the Sophisticated Botnet, https://www.blackhatethicalhacking.com/articles/necurs-uncovering-the-sophisticated-botnet/ (приступљено 6.4.2024.)

127: Symantec Security Response,Dyre: Emerging threat on financial fraud landscape, https://docs.broadcom.com/doc/dyre-emerging-threat (приступљено 6.4.2024.)

128: Hornet Security, What Is Emotet? How Can I Protect Myself?, https://www.hornetsecurity.com/en/knowledge-base/emotet/ (приступљено 6.4.2024.)

129: securityweek.com, BASHLITE Botnets Ensnare 1 Million IoT Devices, https://www.securityweek.com/bashlite-botnets-ensnare-1-million-iot-devices/ (приступљено 6.4.2024.)

130: NHS, BASHLITE Botnet, https://digital.nhs.uk/cyber-alerts/2018/cc-2557 (приступљено 6.4.2024.)

131: thehackernews.com, French Police Remotely Removed RETADUP Malware from 850,000 Infected PCs, https://thehackernews.com/2019/08/retadup-botnet-malware.html (приступљено 6.4.2024.)

132: Sam Edwards, Ioannis Profetis,Hajime: Analysis of a decentralized internet worm for IoT devices, https://web.archive.org/web/20161230182045/https://security.rapiditynetworks.com/publications/2016-10-16/hajime.pdf (приступљено 6.4.2024.)

133: goanywhere.com, What is the Smominru Botnet, https://www.goanywhere.com/blog/what-is-the-smominru-botnet (приступљено 6.4.2024.)

134: cloudflare.com, The WireX Botnet: How Industry Collaboration Disrupted a DDoS Attack, https://blog.cloudflare.com/the-wirex-botnet (приступљено 6.4.2024.)

135: radware.com, Satori IoT Botnet Variant, https://www.radware.com/security/ddos-threats-attacks/threat-advisories-attack-reports/satori-iot-botnet/ (приступљено 6.4.2024.)

136: Unit 42, Satori: Mirai Botnet Variant Targeting Vantage Velocity Field Unit RCE Vulnerability, https://unit42.paloaltonetworks.com/satori-mirai-botnet-variant-targeting-vantage-velocity-field-unit-rce-vulnerability/ (приступљено 6.4.2024.)

137: alertlogic, GoBrut Botnet ELF Variant and New C2 Discovered, https://www.alertlogic.com/blog/gobrut-botnet-elf-variant-and-new-c2-discovered/ (приступљено 6.4.2024.)

138: trendmicro.com - Warren Adam Sto. Tomas, Trojan.Linux.KERBERDS.A, https://www.trendmicro.com/vinfo/us/threat-encyclopedia/malware/trojan.linux.kerberds.a (приступљено 6.4.2024.)

139: bitdefender, New FluBot and TeaBot Global Malware Campaigns Discovered, https://www.bitdefender.com/blog/labs/new-flubot-and-teabot-global-malware-campaigns-discovered/ (приступљено 6.4.2024.)

140: Microsoft Threat Intelligence, Microsoft research uncovers new Zerobot capabilities, https://www.microsoft.com/en-us/security/blog/2022/12/21/microsoft-research-uncovers-new-zerobot-capabilities/ (приступљено 6.4.2024.)

141: Akamai, InfectedSlurs Botnet Spreads Mirai via Zero-Days, https://www.akamai.com/blog/security-research/new-rce-botnet-spreads-mirai-via-zero-days (приступљено 6.4.2024.)

142: Hamilton, Time Series Analysis, Princeton University Press, ISBN-13: 978-0691042893 (1994)

143: Green, Econometric Analysis, Pearson, ISBN-13: 978-0134461366 (2017)

144: Benford, F.: The Law of Anomalous Numbers, Proceedings of the American Philosophical Society, Vol. 78, No. 4, pp. 551-572 (1938)

145: Hill, Theodore P.: A Statistical Derivation of the Significant-Digit Law, Statistical Science, Vol. 10, No. 4, pp. 354-363 (1995)

146: Batista, E.A.P.A., et al.: CID: an efficient complexity-invariant distance for time series, Data Mining and Knowledge Discovery, Vol. 28, No. 3, 10.1007/s10618-013-0312-3 (2013)

147: Welch, P.: The use of the fast Fourier transform for the estimation of power spectra: A method based on time averaging over short, modified periodograms, IEEE Transactions on Audio and Electroacoustics, Vol. 15, No. 2, 10.1109/TAU.1967.1161901 (1967)

148: Lempel, A., Ziv, J.: On the Complexity of Finite Sequences, IEEE Trans. on Information Theory, Vol. 22, No. 1, p. 75–81, 10.1109/TIT.1976.1055501 (1976)

Биографски подаци

Ђорђе Јовановић, мастер инжењер електротехнике и рачунарства

Ђорђе Јовановић је рођен 7. августа 1994. године у Београду, Република Србија, где је са одличним успехом завршио Основну школу „Стеван Дукић“ и Прву београдску гимназију.
У септембру 2017. године дипломирао је на Електротехничком факултету у Београду на смеру за Рачунарску технику и информатику са просечном оценом 9,17 и оценом 10 на дипломском испиту са темом „Виртуелне мреже реализоване у СДН технологији“.
У септембру 2018. године завршио је мастер студије на Електротехничком факултету у Београду на смеру за Рачунарску технику и информатику са просечном оценом 10,00 и оценом 10 на дипломском испиту са темом „Имплементација лабораторијског окружења за софтверски дефинисане мреже помоћу класе Zodiac FX свичева“.
Докторске студије на модулу за рачунарску технику и информатику је уписао 2018. године. Током докторских студија остварио је просечну оцену 10,00.
Од априла 2019. године ангажован је на Математичком институту САНУ као истраживач-приправник, а од априла 2022. године је ангажован као истраживач сарадник. У оквиру научноистраживачког рада, бавио се применом метахеуристика, блокчејн технологијама, машинским учењем и софтверски дефинисаним мрежама. Објавио је 30 радова у часописима и конференцијама, од којих 9 као првоименовани аутор.
Течно говори енглески језик, има напредни ниво знања из француског, средњи ниво знања из руског и кинеског, и служи се грчким и италијанским језиком.

У Београду, 23.08.2024. године.

образац изјаве о коришћењу

Изјава о коришћењу

Овлашћујем Универзитетску библиотеку „Светозар Марковић“ да у Дигитални репозиторијум Универзитета у Београду унесе моју докторску дисертацију под насловом:

Рано откривање урођених болести коришћењем метода дубоког учења

која је моје ауторско дело.

Дисертацију са свим прилозима предао/ла сам у електронском формату погодном за трајно архивирање.

Моју докторску дисертацију похрањену у Дигиталном репозиторијуму Универзитета у Београду и доступну у отвореном приступу могу да користе сви који поштују одредбе садржане у одабраном типу лиценце Креативне заједнице (Creative Commons) за коју сам се одлучио/ла.

1. Ауторство (CC BY)
2. Ауторство – некомерцијално (CC BY-NC)
3. Ауторство – некомерцијално – без прерада (CC BY-NC-ND)
4. Ауторство – некомерцијално – делити под истим условима (CC BY-NC-SA)
5. Ауторство – без прерада (CC BY-ND)
6. Ауторство – делити под истим условима (CC BY-SA)

(Молимо да заокружите само једну од шест понуђених лиценци.
Кратак опис лиценци је саставни део ове изјаве).

Потпис аутора

У Београду, 23. 8. 2024.

образац изјаве о истоветности штампане и електронске верзије докторског рада

Изјава о истоветности штампане и електронске верзије докторског рада

Име и презиме аутора Ђорђе Јовановић

Број индекса 2017/5001

Студијски програм Рачунарска техника и информатика

Наслов рада Рано откривање уређаја заражених ботнет малвером коришћењем метода детекције аномалија мрежних токова

Ментор проф. др Павле Вулетић

Изјављујем да је штампана верзија мог докторског рада истоветна електронској верзији коју сам предао/ла ради похрањивања у **Дигиталном репозиторијуму Универзитета у Београду.**

Дозвољавам да се објаве моји лични подаци везани за добијање академског назива доктора наука, као што су име и презиме, година и место рођења и датум одбране рада.

Ови лични подаци могу се објавити на мрежним страницама дигиталне библиотеке, у електронском каталогу и у публикацијама Универзитета у Београду.

Потпис аутора

У Београду, 23.8.2024.

образац изјаве о ауторству

Изјава о ауторству

Име и презиме аутора Ђорђе Јовановић

Број индекса 2017/5001

Изјављујем

да је докторска дисертација под насловом

Рано откривање уређаја заражених ботнет малвером коришћењем метода детекције аномалија мрежних токова

- резултат сопственог истраживачког рада;
- да дисертација у целини ни у деловима није била предложена за стицање друге дипломе према студијским програмима других високошколских установа;
- да су резултати коректно наведени и
- да нисам кршио/ла ауторска права и користио/ла интелектуалну својину других лица.

Потпис аутора

У Београду, 23.8.2024.

Ђ. Јовановић