Sustav za otkrivanje upada u mrežu zasnovan na algoritmu strojnog učenja

Varvodić, Benjamin

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU

FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni diplomski studij Automobilsko računarstvo i komunikacije

Sustav za otkrivanje upada u mrežu zasnovan na algoritmu strojnog učenja

Diplomski rad

Benjamin Varvodić

Osijek, 2024.

SADRŽAJ

1. UVOD

U današnjem svijetu, gdje su informacije bogatstvo, sigurnost mreža i informatičkih sustava postaje iznimno važna. Brzi razvoj tehnologije i rastuća povezanost putem interneta omogućili su organizacijama širom svijeta da postignu nevjerojatne uspjehe u poslovanju, upravo zbog trgovanja informacijama. Povećan broj i složenost kibernetičkih napada ugrožavaju sigurnost osjetljivih podataka i stabilnost mrežnih infrastruktura i u krajnjem slučaju, utječu na privatnost ljudi, najčešće krađom identiteta.

Sustavi za otkrivanje upada u mrežu (engl. *Intrusion Detection Systems*, IDS) razvijeni su kao odgovor na sve veću prijetnju kibernetičkih napada. Cilj IDS-a je identificirati i odgovoriti što brže na sumnjive aktivnosti koje mogu ukazivati na pokušaje upada u mrežu ili druge neželjene radnje. Tradicionalni IDS-ovi oslanjali su se na predefinirane uzorke i pravila kako bi detektirali napade, no s razvojem tehnologije, ove metode pokazale su se nedovoljno učinkovite protiv ljudske kreativnosti pri smišljanju napada [1].

Strojno učenje i duboko učenje nude rješenja za nadmašivanje ograničenja tradicionalnih IDS-a. Korištenjem algoritama strojnog i dubokog učenja, IDS-ovi temeljeni na strojnome učenju mogu analizirati velike količine podataka, prepoznati obrasce i anomalije te učiti iz novih napada kako bi ih efektivnije detektirali u budućnosti [1].

U ovom radu istražuje se implementacija algoritama strojnog i dubokog učenja kao IDS-a. Fokus ovog rada je na stvaranju baze podataka, te evaluaciji učinkovitosti različitih algoritama strojnog učenja i dubokog učenja u kontekstu mrežne sigurnosti na toj bazi.

Ovim radom nastojalo se istražiti kako različite značajke mrežnih paketa mogu biti korištene za prepoznavanje zlonamjernih aktivnosti te kako se pomoću strojnog učenja i dubokog učenja može razviti sustav koji je sposoban ne samo detektirati napade, već i kontinuirano učiti i prilagođavati se novim prijetnjama. Korištenjem stvarnih mrežnih podataka, simulacija napada i naprednih alata za analizu prometa, cilj ovog rada je stvoriti bazu podataka koja može učinkovito simulirati nekoliko stvarnih napada, u svrhu treniranja algoritama dubokog i strojnog učenja i evaluacije njihovih performansi.

2. SUSTAV ZA OTKRIVANJE UPADA U MREŽU I PREGLED POSTOJEĆIH RJEŠENJA

2.1. Sustav za otkrivanje upada

IDS je općeniti naziv za moćne alate koji prate mrežni promet računala ili cijelog sustava računala kako bi upozorio na neželjene ili sumnjive aktivnosti i pritom trenutačno izdao upozorenje administratoru [1]. Sustav funkcionira prateći mrežne podatke i uspoređujući ih s predefiniranim skupom pravila i uzoraka. Postoji 5 vrsta sustava za otkrivanje upada [1]:

- **Sustav za otkrivanje upada u mrežu (engl. *Network Intrusion Detection System,* NIDS)**

  NIDS je sustav koji se nalazi na predodređenom strateški odabranom položaju u mreži i proučava promet svih uređaja u mreži i uspoređuje s predefiniranim skupom pravila i uzoraka poznatih napada. Vrlo često se nalazi ispred vatrozida kako bi se osigurala i zaštita istog, zbog mogućih napada na vatrozid koji sprječava daljnje jednostavnije napade.

- **Sustav za otkrivanje upada u domaćina (engl. *Host Intrusion Detection System,* HIDS)**
  HIDS je sustav koji se nalazi na nezavisnom domaćinu (engl. *host*) ili uređaju u mreži i promatra promet isključivo na tom uređaju. Osim promatranja prometa, promatra i datoteke sustava kako ne bi došlo do promjena istih. Najčešće se koristi na kritičnim sustavima upravljanja gdje ne bi trebalo dolaziti do čestih ili redovnih promjena datoteka sustava.
- **Sustav za otkrivanje upada temeljen na sigurnosnom protokolu (engl. *Protocol-based Intrusion Detection System,* PIDS)**
  PIDS je sustav koji se nalazi na ulazu poslužitelja i promatra tok prometa sigurnosnog protokola prijenosa hiperteksta (engl. *Hypertext Transfer Protocol Secure,* HTTPS) i rukuje s prometom protokola prijenosa hiperteksta (engl. *Hypertext Transfer Protocol*, HTTP). HTTPS promet nije enkriptiran i stoga je ovakvo pozicioniranje ključno za siguran rad poslužitelja.
- **Sustav za otkrivanje upada temeljen na protokolu aplikacije (engl. *Application Protocol-Based Intrusion Detection System,* APIDS)**
  APIDS je sustav koji se nalazi u grupi poslužitelja i promatra tok prometa specifičnog aplikacijskog protokola. Najčešći primjer uporabe je za nadziranje i sigurnost poslužitelja strukturiranog jezika upita (engl. *Structured Query Language,* SQL) jer oni služe kao međusloj (engl. m*iddleware*) između baze podataka i poslužitelja.

- **Hibridni sustav za otkrivanje upada (engl. *Hybrid Intrusion Detection System*)**
  Hibridni sustav za otkrivanje upada je sustav temeljen na dvije ili više vrsta IDS-a. Najefektivniji je od svih navedenih vrsta jer kombinira više funkcionalnosti i pokriva više slabih točaka mreže.

2.2. Postojeća rješenja

U radu [2] obavljen je sustavni pregled nekoliko algoritama strojnog i dubokog učenja u svrhu IDS-a. Detaljno su analizirani algoritmi *Bayesova* mreža (engl. *Bayes net*), slučajna šuma (engl. *random forest,* RF), neuronska mreža, rekurentna neuronska mreža (engl. *recurrent neural network*, RNN) i dugo kratkoročno pamćenje (engl. *long short-term memory ,* LSTM).

Bayesova mreža je algoritam koji se sastoji od usmjerenih grafova, dakle koristi čvorove i usmjerene lukove za prikazivanje slučajnih varijabli i njihovih uvjetnih ovisnosti, čime omogućuje modeliranje kompleksnih vjerojatnosnih odnosa među podacima. Zbog međuovisnosti i povezanosti podataka, iz jednog dijela podataka se može izvući i drugi dio podataka, stoga se naziva i *Bayesovo* lančano pravilo [2].

RF je algoritam nadziranog učenja koji iz ulaznih podataka stvara stabla odluke na temelju kojih dobije predviđanje za svako stablo. Nakon svih dobivenih predviđanja sortira sve rezultate i na temelju toga bira stablo koje daje najbolje predviđanje [2].

Neuronska mreža je skup neurona povezanih u slojeve, koji služe za obradu podataka, oni se sastoje od ulaza, aktivacijske funkcije i izlaza. Sve veze između neurona imaju pridružene težine, kako bi se odredila važnost signala između dva neurona. Treniraju se metodom zvanom učenje, najčešće metodom propagacije pogreške unazad (engl. *backpropagation*) i metodom gradijentnog spusta [2].

RNN je vrsta neuronske mreže koja je osmišljena za rad s međusobno povezanim podacima. Za razliku od standardnih neuronskih mreža, RNN ima i povratne veze koje omogućavaju zadržavanje informacija o prethodnim ulazima. RNN ima jedan problem, a to je učenje povezanosti kroz duži vremenski period [2].

Taj problem rješava LSTM, koji je posebna vrsta neuronske mreže koja rješava problem nestajućeg gradijenta koji uzrokuje probleme s učenjem povezanosti podataka kroz duži vremenski period u rekurentnim neuronskim mrežama. Sastoji se od posebnih LSTM stanica (engl. *LSTM cell*) koje imaju 3 dijela: ulaz - određuje koliko novih informacija ulazi u stanicu, jedinica za zaborav - određuje koje se informacije odbacuju uz pomoć *sigmoid* funkcije (generira vrijednosti

između 0 i 1, ako je bliže 1, informacija ostaje) i izlaz - određuje koje se informacije iz stanice prenose dalje. Slojevi mogu biti ulazni, skriveni i izlazni [2].

Svi navedeni algoritmi trenirani su na bazi podataka *KDD Cup 99* i evaluirane s programom *Waikato* okruženje za analizu znanja (engl. *Waikato Environment for Knowledge Analysis, WEKA*) [2].

Evaluacija točnosti ovih algoritama daje sljedeće rezultate:

1. RF - **99,98%**
2. *Bayesova* mreža - **99,78%**
3. Dvoslojna neuronska mreža - **99,35%**
4. LSTM – **64,26%**
5. RNN - **53,18%**.

Ostale metrike korištene u ovom radu, osim točnosti su: broj lažnih uzbuna (engl. *false alarm rate*), odziv (engl. r*ecall)* i F1-mjera (engl. *F1-score*).

U radu [3] je također analizirana primjena algoritama strojnog i dubokog učenja u svrhu IDS-a. Obrađeni su algoritmi metoda potpornih vektora (engl. s*upport vector machine, SVM*), J48, RF i naivni Bayes (engl. *Naive Bayes*, NB). Ovdje su svi algoritmi trenirani na novijoj *NSL-KDD* bazi podataka.

SVM je klasifikacijski algoritam namijenjen za binarnu klasifikaciju. Odlikuje ga dobro generaliziranje, robusnost protiv lokalnog minimuma i korištenje malog broja parametara [3]. Robusnost protiv lokalnog minimuma predstavlja mogućnost algoritma da se ne "zaglavi" na lokalnom minimumu funkcije gubitka, već umjesto toga pronalazi globalni minimum funkcije gubitka. Funkcija gubitka (npr. srednja kvadratna pogreška) služi za kvalitativnu mjeru predviđanja modela, točnije ulaz u funkciju su stvarne oznake i predviđanja modela, a izlaz je numerička vrijednost koja označava razinu pogreške.

NB je jednostavan, ali skalabilan algoritam za klasificiranje koji je temeljen na *Bayesovom* teoremu. U ovom slučaju koristi se kao i SVM, za binarnu klasifikaciju, pripada li klasa napadu ili benignom prometu. Pretpostavlja da su sve značajke u vektoru podjednako nezavisne i važne [3].

J48 je implementacija C4.5 algoritma u programskom jeziku *Java*. C4.5 algoritam je vrsta algoritma za izgradnju stabla odluke, odlikuje se jednostavnošću korištenja. Jedan od problema ovog algoritma je što vrlo lako može doći do pretreniranja (engl. *overfitting*), to se rješava primjenom tehnike obrezivanja (engl. *pruning*) koja uklanja dijelove stabla koja slabo doprinose klasifikaciji [3].

NSL-KDD je zapravo prerađena *KDD Cup 99* baza podataka kako bi pružila realističniju bazu podataka [4]. Također je korišten program *WEKA* za evaluaciju, kao i u prethodnom radu. Ovdje je korišten drugačiji pristup evaluaciji rješenja te su zasebno evaluirane binarna klasifikacija (je ili nije napad) i višeklasna klasifikacija. Višeklasna klasifikacija se sastoji od 4 vrste napada, koji su odbijanje usluge (engl. *denial of service*, DOS), napad s udaljenog uređaja na lokalno (engl. *Remote to local*, R2L), sondiranje (engl. *probe*) i napad korisnika prema sustavu (engl. *User to root*, U2R).

Prva metoda evaluacije je bila križna validacija (engl. *cross-validation*), točnije baza podataka se podijeli na *n* dijelova i zatim se jedan od dijelova izdvoji za testiranje, a ostali se koriste za treniranje i tako iterativno dok se svih *n* dijelova ne iskoristi za testiranje. Takav pristup evaluaciji točnosti daje iduće rezultate, za binarnu i višeklasnu klasifikaciju:

1. RF **98,92%** / **99,99%**
2. J48 **98%** / **96,9%**
3. SVM **97,26% / 96,2 %**
4. NB **94,12% / 86,2%**

Drugi pristup evaluaciji točnosti bio je pomoću testnog skupa koji je predefiniran i izdvojen iz postojeće baze podataka, a daje rezultate, također za binarnu i višeklasnu klasifikaciju:

1. RF **98,77%** / **97,9%**
2. J48 **98,2%** / **97,4%**
3. SVM **97,79%** / **96,4%**
4. NB **94,97%** / **87,4%**

U radu [5] je opisano 6 modela IDS-a, specijaliziranih za IoT tehnologiju. Rješenja su trenirana na *UNSW-NB15* bazi podataka, gdje su podaci pretprocesirani metodom *min-max* skaliranja kako bi se od ukupno 41 značajke prisutne u bazi podataka izdvojilo 10 najvažnijih. Upravo tih 10 značajki sadrži najviše relevantnih podataka za otkrivanje napada. Predloženi modeli su: *XgBoost*,

CatBoost, K najbližih susjeda (engl. *K-nearest neighbors,* KNN), SVM, Kvadratna diskriminacijska analiza (engl. *quadratic discriminant analysis - QDA*) i NB.

XgBoost algoritam je algoritam pojačavanja (engl. *boosting*), on kombinira više jednostavnih modela tzv. slabe učenike u jedan složeni model, u ovom slučaju je to algoritam pojačavanja stabala odluke (engl. *tree boosting*) što znači da spaja više stabala odluke. S ovim algoritmom odrađen je i dodatni odabir značajki te je izračunata važnost pojedine značajke za treniranje.

CatBoost algoritam je moćan algoritam napravljen za rukovanje kategorijskim značajkama (podaci koji se mogu svrstati u grupe, npr. mrežni protokoli, TCP zastavice i sl.), ali svejedno može raditi i s kontinuiranim (podaci koji imaju vrijednost unutar nekog skupa ili omjera, npr. portovi) i brojčanim značajkama. Ovdje je korišten na način da na ulazu prima ulazne podatke i određen broj iteracija, zatim svaki idući model poboljša na temelju prošlog modela, točnije gradijenta funkcije greške. U svakoj iteraciji, *CatBoost* računa gradijent funkcije greške za svaki uzorak u skupu podataka. Gradijent predstavlja smjer i brzinu promjene funkcije greške u odnosu na promjene u predviđanjima modela. Funkcija greške kvantificira koliko se predviđanja modela razlikuju od stvarnih vrijednosti. *CatBoost* koristi gradijent ove funkcije kako bi odredio smjer u kojem bi se model trebao mijenjati kako bi smanjio grešku. Na kraju sve prethodne modele kombinira u konačni model [5].

KNN je jedan od najjednostavnijih algoritama nadziranog učenja. U ovom slučaju je za izračun udaljenosti između podataka (mjera sličnosti/razlike podataka u grupama, pomaže grupiranju podataka) korištena euklidska udaljenost. Prednost korištenja ovog algoritma je upravo jednostavnost korištenja.

QDA je također klasifikacijski algoritam kao i KNN, ali služi se kvadratnom funkcijom za grupiranje klasa umjesto euklidske udaljenosti [5].

Postignuti su rezultati točnosti:

1. *XGBoost* – **99,99%**
2. *CatBoost* – **99,99%**
3. KNN – **99,98%**
4. SVM – **99,98%**
5. QDA – **99,97%**
6. NB – **97,14%**

Autori u radu [6] naglašavaju važnost prepoznavanja napada u početnim fazama i prevenciji daljnjeg razvoja. Također, nastoji se pokazati koliko IDS-ovi mogu biti ovisni o bazama podataka na kojima su trenirani zbog ograničenosti podataka, prikazan je i utjecaj pojedinog algoritma na performanse sustava (koliko je računalno zahtjevan), ovisno o kojem algoritmu se radi.

Prijedlog rješenja temelji se na algoritmu koji se fokusira na jednoj značajci koju prati kroz sesiju i ona služi kao okidač za detaljniju analizu sesije. Odabrana značajka je broj poslanih paketa (engl. *forward packet count*) i nalazi se u svakoj klasifikaciji prometa, dakle u benignom prometu i svim klasama napada.

Pristup problemu u ovom radu je drugačiji jer zahtijeva poznavanje prosječnog broja paketa za svaku sekvencu paketa. Sve sekvence su početno benigne klasifikacije dok se ne dostigne određen prag broja paketa, nakon toga se ta sekvenca smatra sumnjivom. Kada se sekvenca označi kao sumnjiva, izdvaja se za daljnje promatranje. Detaljnom analizom prethodno izdvojene sekvence pokušava se odrediti radi li se o lažnoj uzbuni ili ako se radi o napadu, koji se točno napad dogodio.

Ovaj rad se razlikuje od ostalih po tome što modeli nisu trenirani i testirani na samo jednoj, već na 3 baze podataka, točnije *ISCX2012, CIC-IDS2017* i *CSE-CIC-IDS2018*. Sve 3 baze su jedinstvene po načinu označavanja paketa, po obujmu podataka i broju značajki. Prije implementiranja poboljšanja izvršene su manje izmjene baza podataka, izbačene su sve klase napada s jednim zlonamjernim paketom po sesiji.

Evaluacijom točnosti pokazalo se da je metoda ranije klasifikacije poboljšala sve metrike postojećih algoritama: RF, *AdaBoost* stablo odluke, *XGBoost*, stroj ekstremnog učenja (engl. *Extreme Learning Machine*, ELM), duboka neuronska mreža (engl. *deep neural network*, DNN) i konvolucijska neuronska mreža (engl. *convolutional neural network*, CNN).

AdaBoost stablo odluke je algoritam realiziran kao kombinacija algoritma stablo odluke s algoritmom adaptivnog pojačavanja, adaptivan u smislu da prilagođava težine pogrešno klasificiranih podataka (veće težine pogrešno klasificiranim uzorcima). Radi na istom principu kao i *XGBoost* algoritam (zapravo kao bilo koji algoritam pojačavanja), a to je spajanje više slabih klasifikatora, u ovom slučaju stabala, u jedan jaki klasifikator [6].

ELM je algoritam za jednostavno treniranje mreže s prosljeđivanjem unaprijed (engl. *feedforward network*) razvijen kako bi se riješio jedan od glavnih nedostataka neuronskih mreža, dugo vrijeme treniranja i osjetljivost na lokalne minimume [6].

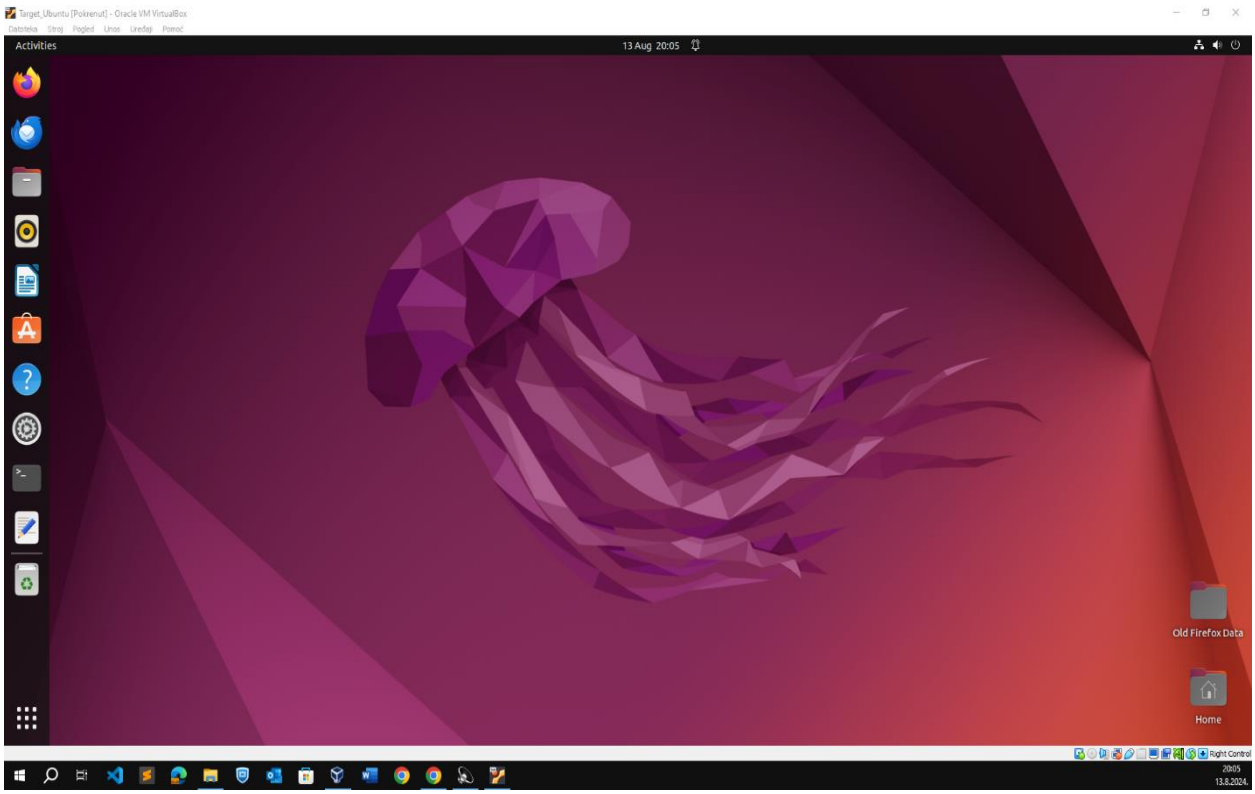
DNN je vrsta neuronske mreže koja se odlikuje velikim brojem skrivenih slojeva, što omogućava učenje složenih i apstraktnih podataka. Trenira se učenjem, kao i standardna neuronska mreža, metodama gradijentnog spusta i propagacijom pogreške unazad [6].

Rezultati ranije navedenog prijedloga su različiti s obzirom na kojoj su bazi podataka trenirani i testirani. Od 18 slučajeva (6 algoritama na 3 baze podataka), samo u jednom slučaju nema poboljšanja rezultata, a to je za DNN na *CSE-CIC-IDS2018* bazi. Čak i s implementiranim prijedlogom, uočljivo je da su F-1 mjere svih algoritama relativno niske, od 30-50% za algoritme dubokog učenja (ELM, DNN i CNN) dok ostali algoritmi daju F-1 mjeru veću od 75%. Zaključak rada je da bilo kakvo poboljšanje ne može "spasiti" sustav od loših performansi, ako baza podataka s kojom algoritam radi nema dovoljno dobrih trening podataka.

3. PREDLOŽENO RJEŠENJE

3.1. Izrada baze podataka

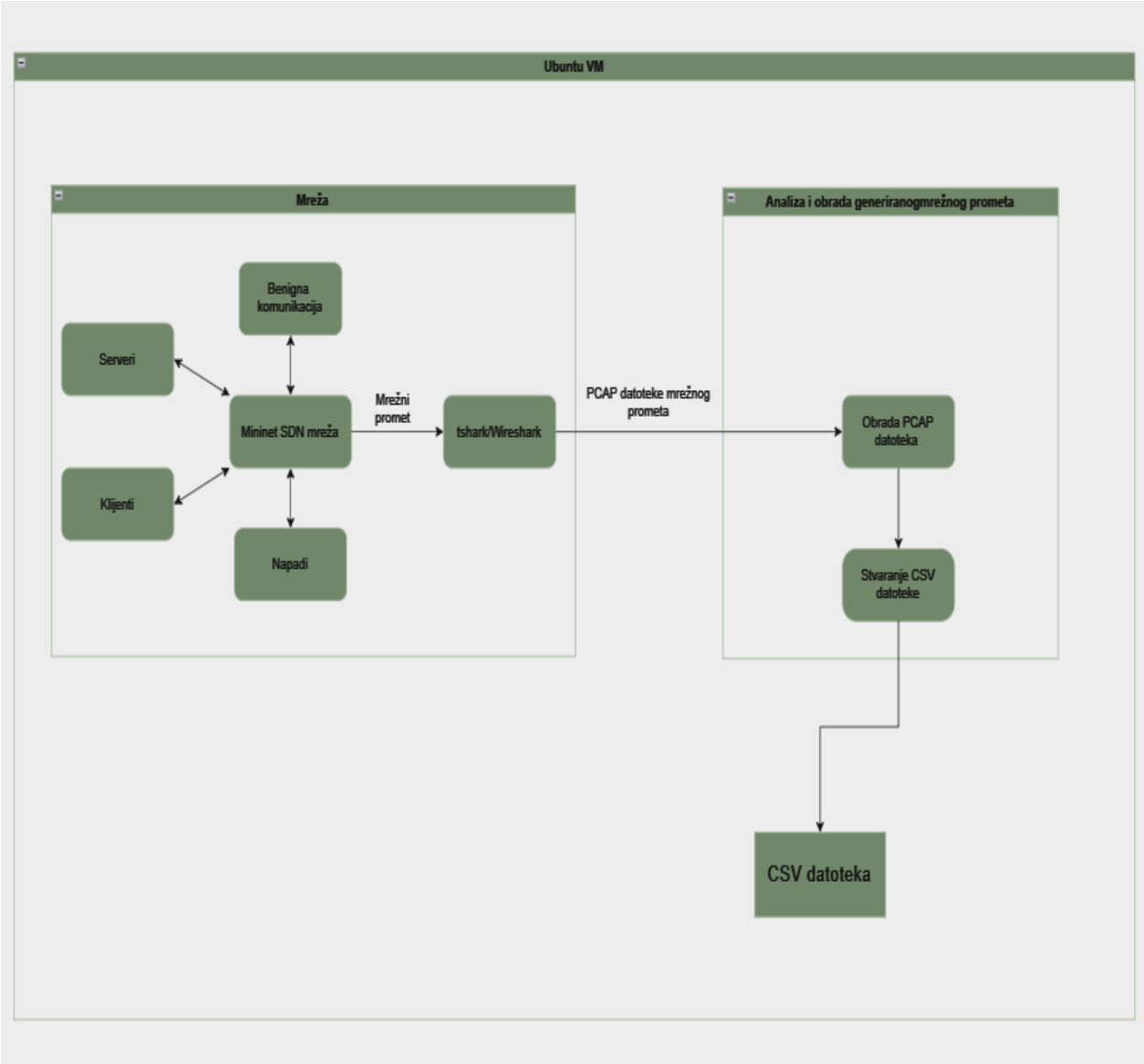
Realiziranje baze podataka započinje izradom virtualnog računala s operativnim sustavom (engl. *Operating System*, OS) *Ubuntu*, zbog nativnosti odabranog mrežnog simulatora *Mininet* tom OS-u. *Mininet* je jednostavan, ali koristan alat s kojim se mogu simulirati razne vrste mreža i mrežnih topologija, a jedini nedostatak je što nema grafičko sučelje, već se svo programiranje izvodi preko skripti i terminala. Na slici 3.1. prikazane su radna površina i sučelje *Linux* OS-a, kao i programska traka *Windows* 10 OS-a.



Slika 3.1. Sučelje Linux OS-a

Mininet omogućava stvaranje softverski definirane mreže (engl. *Software Defined Network,* SDN) na kojoj će se izgraditi baza podataka. S obzirom da se programira SDN, to znači da nema nikakvih dodatnih fizičkih dijelova, sav mrežni hardver je već programiran kao modul *Minineta*.

Postoji nekoliko skripti koje su potrebne za pravilan rad zamišljene mreže, kao i cjelovitu izradu baze podataka, a to su: glavna skripta, skripte poslužitelja, skripte poslužitelja, skripte napada, skripta za pozivanje napada, skripta za komunikaciju i skripta za analizu datoteka prikupljenih paketa (engl. *packet capture,* PCAP), odnosno PCAP datoteka.



Slika 3.2. Blok dijagram sustava za izradu baze podataka

Na slici 3.2. prikazan je blok dijagram cijelog sustava za izradu baze podataka, sastoji se od 3 glavne cjeline:

- ***Ubuntu VM*** - virtualno računalo korišteno za virtualizaciju *Ubuntu* OS-a, preko *OracleVM VirtualBox* alata.
- **Mreža** - softverski generirana mreža za generiranje podataka (mrežnog prometa), dodatno povezan *tshark* za prikupljanje mrežnog prometa u obliku PCAP datoteka.
- **Analiza i obrada PCAP datoteka** – obrada PCAP datoteka te konačno spajanje u CSV datoteku

Cjelinu "**mreža**" čine: glavna skripta, skripte poslužitelja (engl. s*erver*), skripta za benignu komunikaciju, skripte napada, skripta za odabir napada.

Glavna skripta sastoji se od dijelova za:

- **Oblikovanje mreže**

Definiranje upravljača, preklopnika i kontrolirane veze. Za upravljač je odabran modul *Controller* iz *Mininet* biblioteke *node.* Za preklopnik (engl. *switch*) odabran je *OVSSwitch* iz iste biblioteke, *OVSSwitch* predstavlja softversku realizaciju preklopnika. Za kontroliranu vezu odabran je *TCLink* iz biblioteke l*ink*, koja omogućava korištenje dijela *Linux* jezgre (engl. k*ernel*) za kontrolu prometa (engl. *traffic control,* TC) kako bi se povezali objekti mreže, dakle klijenti i poslužitelji s preklopnicima i preklopnici s usmjerivačem.

- **Dodavanje upravljača**

Upravljač obavlja osnovne funkcije mreže poput odobravanja usmjeravanja paketa, donošenje odluka o prioritetu paketa. Omogućuje pozivanje svih objekata u mreži (klijenti, poslužitelji, usmjerivač, preklopnici i slično) i njihovo oblikovanje i upravljanje. Također, na kontroleru je moguće definiranje posebnih pravila prosljeđivanja paketa na mreži.

- **Dodavanje i povezivanje usmjerivača (engl. *router)***

Usmjerivač je objekt *Mininet* mreže kojem se dodjeljuju posebne ovlasti, a to je kontrola nad paketima između dvije podmreže. Daju mu se ovlasti nad glavnim podmrežama 10.0.1.0/24 na kojoj su poslužitelji i 10.0.2.0/24 na kojoj su klijenti. Ovlasti se dodjeljuju korištenjem ranije navedenog *TCLink* koji stvara poveznicu između usmjerivača i preklopnika, dodjeljujući usmjerivaču adrese pristupnika (engl. *gateway address*) 10.0.1.254 i 10.0.2.254. Kada su usmjerivaču dodane adrese pristupnika, mogu se definirati pravila usmjeravanja između dvije podmreže. Dodatno se definiraju sekundarne IP adrese klijentske podmreže kao nastavak na primarnu IP adresu. Važno je natuknuti da primarne i sekundarne IP adrese koriste iste adrese pristupnika na usmjerivaču. Sekundarne IP adrese se koriste isključivo za provjeru integriteta baze podataka (jesu li svi paketi pravilno označeni) i ne koriste se za treniranje algoritama. Za treniranje algoritama koriste se isključivo primarne adrese (sekundarne adrese se vrate na primarne).

- **Dodavanje preklopnika (engl. *switch*)**

Preklopnik je objekt *mininet* mreže s posebnim ovlastima na dodijeljenoj podmreži. U skripti ih postoje dva, imenovani su „*s1*" i „*s2*", gdje *s1* upravlja poslužiteljskom podmrežom 10.0.1.0/24, a *s2* klijentskom podmrežom 10.0.2.0/24. Preklopnici omogućavaju prijenos paketa uređaja pripadne mreže prema usmjerivaču, a usmjerivač te iste pakete prenosi drugom preklopniku.

- **Izrada klijenata**

Klijenti su objekti *mininet* mreže koji izvršavaju određene naredbene linije prema odabranom uređaju. Kao takvi mogu se smatrati „stacionarnim" objektima jer ne obavljaju nikakvu pozadinsku aktivnost, dok ih se ne pozove da izvrše proslijeđenu naredbu. Nije ih potrebno dodatno programirati, osim dodjeljivanja imena, IP adrese i predviđene rute (na koje adrese pristupnika idu paketi). U ovoj mreži postoji 21 klijent.

- **Pokretanje i definiranje portova poslužitelja**

Poslužitelji su u suštini identični klijentima. Na isti način ih se programira. Ključna razlika između njihovog rada leži u tome da poslužitelji konstantno obavljaju pozadinsku aktivnost i šalju odgovore prema klijentima koji vrše komunikaciju s njima. Korištena su 3 HTTP poslužitelja, 2 poslužitelja protokola kontrole prijenosa (engl. *Transmission Control Protocol*, TCP), jedan poslužitelj protokola korisničkih *datagrama* (engl. *User Datagram Protocol*, UDP), jedan poslužitelj protokola prijenosa datoteka (engl. *File Server Protocol*, FTP), jedan SQL poslužitelj, jedan poslužitelj protokola sigurne ljuske (engl. *Secure Shell*, SSH) i 6 poslužitelja sustava domenskih imena (engl. *Domain Name System*, DNS). Broj servera je proizvoljno odabran i periodično uvećan, kroz realizacije napada.

- **Pokretanje *tshark-a***

Kako bi se uvjerili da se komunikacija i napadi izvršavaju pravilno, potrebno je pratit mrežni promet uz pomoć analizatora paketa, u ovom slučaju korišten je *Wireshark*. Uz pomoć *Wireshark*-a prati se promet na svakom poslužitelju, kao i na obje podmreže, kako bi se dobili potpuni podaci o prometu. *Wireshark* ima mogućnost spremanja prometa u datoteke, koje se nazivaju PCAP datoteke. S obzirom da *Mininet* terminal nema grafičko sučelje, koristi se *tshark*, konzolna verzija *Wiresharka*. *Tshark* se pokrene u pozadini svakog poslužitelja, točnije na sučelju svakog poslužitelja i na sučeljima obje podmreže, sve zabilježene podatke također sprema u PCAP datoteke, po jednu za svaki poslužitelj i sučelje. Prije korištenja *tshark-a*, potrebno ga je instalirati.

- **Petlja za komunikaciju**

Mrežna komunikacija ne odvija se kontinuirano, već u iteracijama, u kojima se događa oko 90 sekundi komunikacija između poslužitelja i klijenata. Sama komunikacija programirana je u zasebnu skriptu koja se pozove, i dok se skripta izvodi, nakon određenog vremena, otprilike 60 sekundi, pozove se još jedna skripta za napade. Ovisno o napadu i njegovoj jačini, za vrijeme

odvijanja istog, benigna komunikacija ili prestane, ili se odvija sporije. Nakon završetka napada, događa se još 30 sekundi benigne komunikacije.
Cjelokupan kod za detaljniji pregled glavne skripte dostupan je u prilogu P.3.1.

Skripte poslužitelja:

- **HTTP poslužitelj**
  Jednostavan HTTP poslužitelj realiziran koristeći *pythonove http.server* i *socketserver* biblioteke. Implementira jednostavne funkcionalnosti, poput odgovaranja na *curl* upite, što omogućava *SimpleHTTPRequestHandler* klasa iz *http.server* biblioteke.

- **TCP poslužitelj**
  Jednostavan TCP poslužitelj realiziran koristeći *pythonovu socket* biblioteku. Omogućena je višedretvenost uz pomoć *threading* biblioteke. Poslužitelj šalje povezanom klijentu njegove podatke kako bi dodatno potvrdio povezanost. Poslužiteljima su dodijeljeni redom portovi 124 i 125.

- **UDP poslužitelj**
  Realiziran istim principom kao i TCP poslužitelj. Jedina je razlika rukovanje UDP paketima. Poslužitelju je dodijeljen port 126.

- **FTP poslužitelj**
  Jednostavan FTP poslužitelj realiziran koristeći *pythonovu pyftpdlib* biblioteku. Ovim poslužiteljem omogućen je jednostavan prijenos datoteka. Dodan je korisnik s apsolutnim pravima za rukovanje datotekama unutar direktorija mreže, to je omogućeno *DummyAuthorizer* klasom. Dodijeljen mu je standardni FTP port 21.

- **SQL poslužitelj**
  Ovaj poslužitelj nema zasebnu skriptu već je realiziran naredbom „*service mysql start*" koja pokreće ugrađeni servis na *Linux* OS-u koji rukuje bazama podataka. Dodijeljen je port 128.

- **DNS poslužitelj**
  Jednostavan DNS poslužitelj implementiran u koristeći *pythonovu dnslib* biblioteku za rukovanje DNS upitima i s podrškom za TCP i UDP protokole na portu 53.

Cjelokupan kod za detaljniji pregled skripti poslužitelja dostupan je u prilogu P.3.1.

Skripta za pozivanje napada i pojedine skripte napada:

Odabrano je i uspješno simulirano 5 napada, svaki napad napisan je u zasebnoj skripti, koja se poziva u skripti za nasumičan odabir napada. Ta skripta pozove skriptu odabranog napada, dodijeli mu izvršitelja, točnije nasumično se odabere klijent koji će za tu iteraciju izvršavati napad, zatim se odredi opseg napada, u rasponu od 75-250 malicioznih paketa. Realizirani su sljedeći napadi:

- ***Syn flood***

Napad namijenjen pretrpavanju i prekidu TCP komunikacije. Zlouporabljuje standardno TCP rukovanje, gdje u benignom prometu, klijent šalje početni signal „*syn*" kojim označava želju za uspostavljanjem komunikacije s drugim uređajem, npr. poslužiteljem. Drugi uređaj povratno šalje signal „*syn-ack*", što označava da ima kapaciteta za komunikaciju i spreman je za povezivanje. Nakon toga klijent šalje „*ack*" signal kojim se uspostavlja i započinje komunikacija. *Syn flood* šalje veliki broj „*syn"* paketa prema žrtvi, najčešće TCP poslužitelju. Velik broj „*syn"* paketa s različitih maskiranih adresa (engl. s*poofed adress*) dovodi do stanja sličnom beskonačnoj petlji, ako se ne zaustavi, jer napadnuti poslužitelj nastavlja odgovarati sa „*syn-ack"* paketima prema napadaču, očekujući potvrdu o zaključivanju početka komunikacije. Ovaj napad pripada DoS vrsti napada.

- **UDP *flood***

Vrsta DoS napada koja se temelji na zlouporabi načina rada UDP protokola. UDP protokol nema mehanizam rukovanja kao TCP, već je uvijek otvoren za komunikaciju, što ga čini manje pouzdanim jer nema nikakvog redoslijeda slanja podataka niti potvrde o slanju i primitku. Dakle, napadač šalje veliki broj UDP paketa prema žrtvi, najčešće UDP poslužitelju. Postoje dva ishoda napada i nijedan nije dobar. Prvi je da se pogodi port korišten za UDP poslužitelj i UDP poslužitelj je preplavljen UDP paketima s različitih adresa, na koje ne može odgovoriti jer ili ne postoje ili je prava adresa zamaskirana s drugom. Ukoliko se ne pogodi port UDP poslužitelja, tad UDP poslužitelj odgovara paketima Internet protokola za kontrolne poruke (engl. *Internet Control Message Protocol*, ICMP) paketima s porukom da port nije dostupan, zbog velikog broja odgovora, UDP poslužitelj često otkaže, odnosno prekine s radom zbog prekomjerne uporabe resursa.

- ***Smurf attack***

Vrsta DoS napada koja napada adresu pristupnika usmjerivača. Napad je zamišljen na način da napadač odabere jednog od legitimnih klijenata i ukrade njegovu IP adresu. S ukradenom IP adresom pokreće napad u obliku velikog broja ICMP zahtjeva prema adresi pristupnika odabrane žrtve. Željeni učinak napada je onemogućavanje standardnog rada usmjerivača.

- ***Port scanning***

Vrsta napada sondiranja, razlikuje se od ostalih definiranih napada po tome što ne onemogućava normalan rad sustava već ga „pročešlja", kako bi otkrio otvorene portove napadnutog poslužitelja. S tom informacijom mogu se olakšati budući napadi s napadačke strane, npr. UDP *flood*. Funkcionira na način da se s različitih IP adresa šalju „*syn*" paketi prema određenom broju portova, očekujući pozitivan „*syn-ack*" odgovor od otvorenog porta.

- **DNS *amplification***

Vrsta napada distribuiranog odbijanja usluge (engl. *Distributed Denial of Service*, DDoS) gdje je cilj zloupotrijebiti postojeći DNS poslužitelj kako bi se onemogućio rad jednog ili više DNS poslužitelja. U ovoj realizaciji napadač zamaskira svoju adresu s jednom od adresa DNS poslužitelja i započinje napad prema ostalim DNS poslužiteljima, koristeći tip upita „*ANY*", što potiče poslužitelj da odgovori sa svim podacima o tom poslužitelju. Shodno tome, velik broj takvih upita može prouzrokovati prekid rada poslužitelja.

Cjelokupan kod za detaljniji pregled skripti napada i skripte za pozivanje napada dostupan je u prilogu P.3.1.

Skripta za komunikaciju:

Skripta simulira složeno mrežno okruženje u kojem klijenti kontinuirano izvode različite mrežne aktivnosti prema poslužiteljima unutar mreže. Aktivnosti uključuju:

- Web zahtjevi (HTTP): Klijenti šalju HTTP zahtjeve prema unaprijed definiranim URL-ovima poslužitelja.
- FTP aktivnosti: Klijenti nasumično odabiru datoteke za prijenos (engl. *upload*) ili preuzimanje (engl. *download*) putem FTP-a.
- TCP i UDP komunikacija: Klijenti šalju podatke prema TCP i UDP poslužiteljima koristeći *netcat* alat.
- *MySQL* upiti: Klijenti šalju upite prema *MySQL* poslužitelju.
- SSH sesije: Klijenti se povezuju na poslužitelje putem SSH-a i izvršavaju jednostavne naredbe.
- Simulacija prijenosa videozapisa: Klijenti pokušavaju pristupiti video prijenosima na poslužitelju
- DNS upiti: DNS poslužitelji međusobno prosljeđuju upite kako bi se odredilo koja domena je tražena.

Omogućena je i višedretvenost (engl. *multithreading*) kako bi omogućila istovremeno izvršavanje mrežnih aktivnosti od strane više klijenata. Svaka mrežna operacija koja se izvršava od strane klijenta zaštićena je *cmd_lock* mehanizmom, koji osigurava da se naredbe šalju klijentima na siguran način bez međusobnog preklapanja. Zbog ograničenih resursa, osim korištenja *cmd_lock* mehanizma, koristi se i intervalno slanje u rasponu 10 sekundi, kako se mreža ne bi preplavila benignim prometom.
Cjelokupan kod za detaljniji pregled skripte za komunikaciju dostupan je u prilogu P.3.1.

Cjelinu „**Analiza i obrada PCAP datoteka**" čini skripta za analizu i obradu PCAP datoteka te sortiranje prikupljenih podataka u CSV datoteku.

Skripta za analizu i obradu PCAP datoteka te stvaranje CSV datoteke:

Ova skripta analizira mrežni promet iz PCAP datoteka koristeći *Scapy*, identificirajući i klasificirajući različite protokole i potencijalne mrežne napade, te zapisuje rezultate u CSV datoteku. Dakle, skripta ima 2 glavna dijela:

Analiza protokola i izvlačenje podataka:

U ovom dijelu opisani su protokoli koji su korišteni, kao i ključni podaci (značajke) koje sadržavaju. Također je navedena i svrha određenih značajki

Ethernet

Izvučene informacije:

eth_src: MAC adresa izvora.

eth_dst: MAC adresa odredišta.

Svrha: MAC adrese služe za identifikaciju uređaja unutar lokalne mreže i mogu pomoći u otkrivanju anomalija kao što je promjena uobičajenih MAC adresa (moguće zlonamjerne aktivnosti). Napadači vrlo često ne mogu promijeniti svoje MAC adrese bez korištenja dodatnog napada ARP *spoofing* koji nije realiziran u ovoj bazi podataka.

Internetski protokol (engl. *Internet Protocol*, IP)

Izvučene informacije:

src_ip: IP adresa izvora.

dst_ip: IP adresa odredišta.

ip_len: Duljina IP paketa.

ip_id: Identifikacijski broj paketa.

ip_flags: Zastavice koje pokazuju različite atribute

ip_frag: Pomak fragmenta, ako je paket fragmentiran.

ip_ttl: Životni vijek *(engl. Time-To-Live)*, broj skokova kroz mrežu prije nego što paket bude odbačen.

ip_proto: Dodatni protokol uz IP paket (npr. TCP, UDP, ICMP).

ip_chksum: Kontrolni zbroj (engl. *checksum*) IP zaglavlja.

ip_options: Opcije postavljene u IP zaglavlju.

Svrha: Ove informacije su ključne za praćenje IP prometa između različitih uređaja i za detekciju neobičnih obrazaca u mrežnom prometu. Najvažnije značajke od ovih su *src_ip* i *dst_ip* jer se iz njih prepoznaje odakle i gdje idu IP paketi.

TCP

Izvučene informacije:

src_port: Izvorni port.

dst_port: Odredišni port.

seq_num: Redni broj TCP segmenta.

ack_num: Broj potvrde (*ack*).

tcp_flags: Zastavice u TCP zaglavlju (*syn, ack, fin, rst,* itd.).

window_size: Veličina prozora.

tcp_len: Duljina TCP segmenta.

tcp_chksum: TCP kontrolni zbroj.

tcp_options: Opcije TCP zaglavlja.

tcp_urgptr: Hitni pokazivač (engl. *urgent pointer*), označava pakete s prednošću prijenosa.

is_ssh: Dodatna oznaka ako je promet na portu 22 (standardni SSH port).

Svrha: TCP zastavice i portovi su ključni za prepoznavanje vrsta napada kao što su *SYN flood*, *port scanning*, i druge anomalije (npr. prekid veza, ponovno slanje paketa i slično) u TCP komunikaciji.

UDP

Izvučene informacije:

src_port: Izvorni port.

dst_port: Odredišni port.

udp_len: Duljina UDP segmenta.

udp_chksum: UDP kontrolni zbroj.

Svrha: Segmenti UDP paketa se koriste za prepoznavanje napada kao što su UDP *flood* i DNS *amplification*.

ICMP

Izvučene informacije:

icmp_type: Tip ICMP poruke (npr. 8 za *Echo Request*).

icmp_code: ICMP kod (ovisno o tipu poruke).

icmp_echo_request: Zastavica koja označava *Echo Request*.

Svrha: Značajka *icmp_echo_request* se koristi za prepoznavanje napada poput *Smurf* napada, koji uključuje slanje velikog broja *Echo Request* paketa.

DNS

Izvučene informacije:

dns_query: DNS upit (domena koja se traži).

dns_resp: DNS odgovor (IP adresa ili druga informacija koja se vraća).

Svrha: DNS promet je ključan za prepoznavanje napada poput DNS *amplification*, gdje se koristi legitimni DNS poslužitelj za pojačavanje napada.

Protokol za rješavanje adrese (engl. *Address Resolution Protocol,* ARP)

Izvučene informacije:

arp_op: Operacija ARP-a (1 za zahtjev, 2 za odgovor).

arp_psrc: IP adresa izvora.

arp_pdst: IP adresa odredišta.

Svrha: ARP se koristi za prepoznavanje anomalija u mreži, poput ARP *spoofinga*, gdje napadač lažno predstavlja svoju MAC adresu.

Prepoznavanje napada:

U ovom dijelu opisan je način na koji su označeni i izdvojeni napadi u mreži. Prije opisa označavanja napada, važno je napomenuti da se adrese iz sekundarnih domena (10.0.20, 10.0.21, 10.0.23, 10.0.24 i 10.0.26) ne koriste za treniranje algoritama, već samo njihove adrese iz primarne domene (10.0.2.), tako što se domene sekundarnih adresa uz pomoć jednostavnog koda zamijene s primarnom domenom.

SYN *Flood*

Skripta prati broj TCP paketa koji dolaze iz specifičnih IP adresa. Ako *src_ip* počinje s „10.0.20.", TCP promet se označava kao napad (SYN *flood*)

UDP *Flood*

Skripta prati broj UDP paketa koji dolaze iz specifičnih IP adresa. Ako *src_ip* počinje s „10.0.21.", UDP promet se označava kao napad (UDP *flood*).

***Smurf* napad**

Skripta prepoznaje ICMP *echo request* (tip 8) pakete i označava ih kao dio *Smurf* napada ako dolaze iz IP adrese koja počinje s „10.0.23." i ako klijent koji je komunicirao s IP adresom „10.0.23" cilja adrese pristupnika (adrese koje završavaju s '.255').

Port Scanning

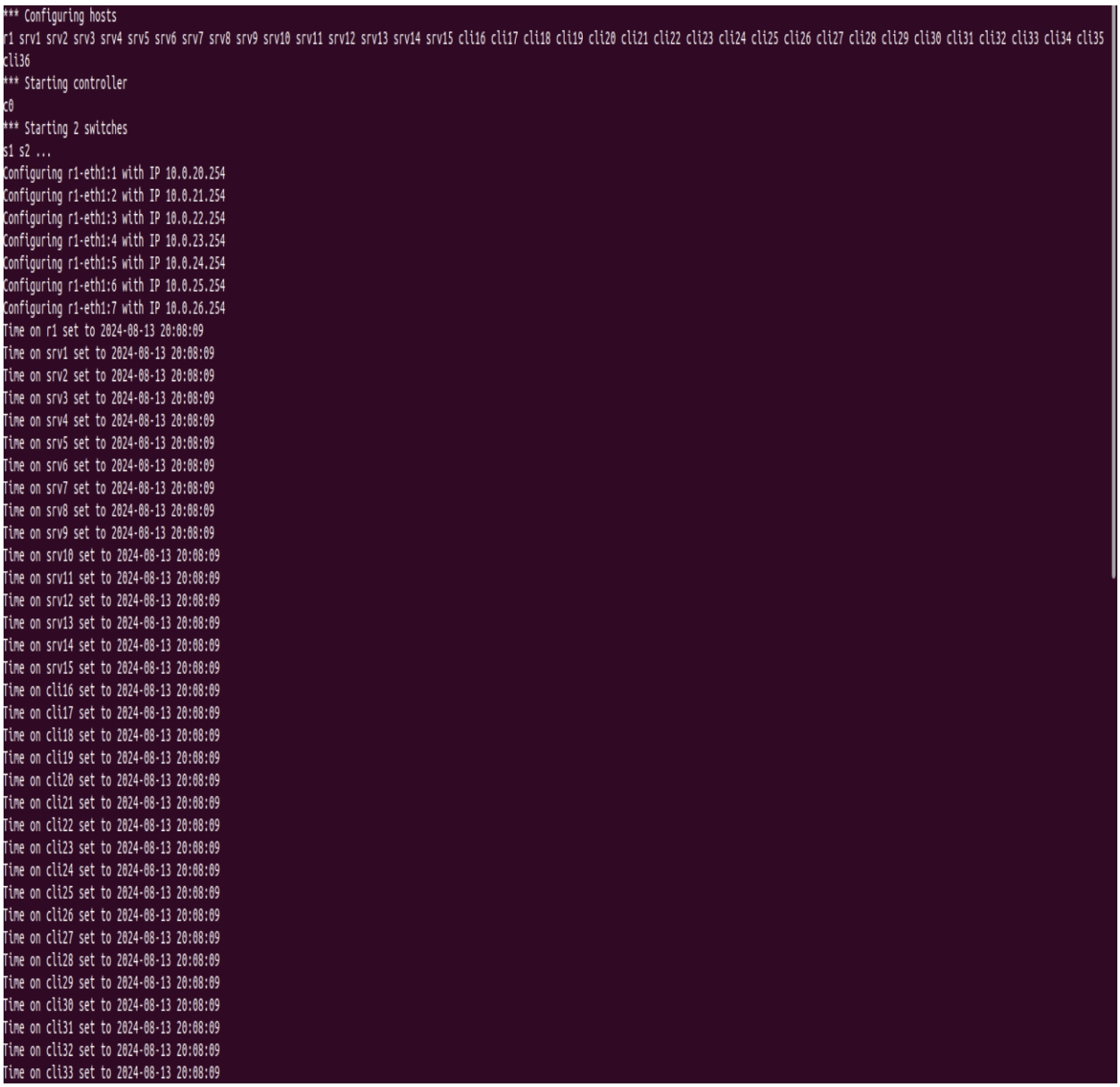
Skripta prati broj TCP paketa koji dolaze iz specifičnih IP adresa. Ako *src_ip* počinje s „10.0.24.", TCP promet se označava kao napad (*Port scanning*)

DNS *Amplification*

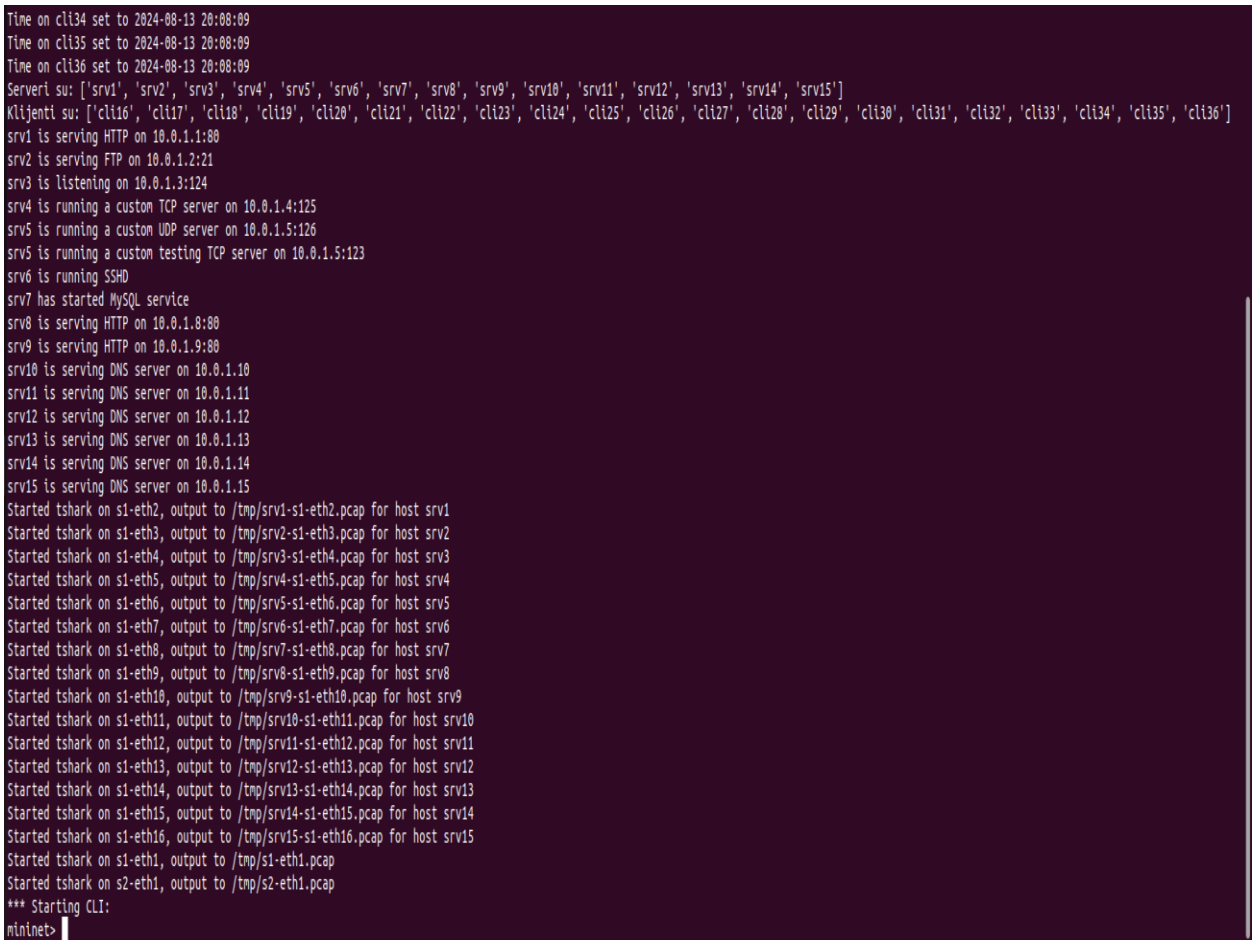
Skripta prepoznaje DNS upite i odgovore, te označava promet kao napad (DNS *amplification*) ako dolazi iz IP adrese koja počinje s „10.0.26.“ i usmjerena je prema jednom od DNS poslužitelja i ako promet između specifičnih DNS poslužitelja nakon detektirane IP adrese značajno poraste. Po uzoru na ostale baze podataka, svi paketi napada su označeni kao napad.

Cjelokupan kod za detaljniji pregled skripte za analizu i obradu PCAP datoteka te stvaranje CSV datoteke dostupan je u prilogu P.3.1.

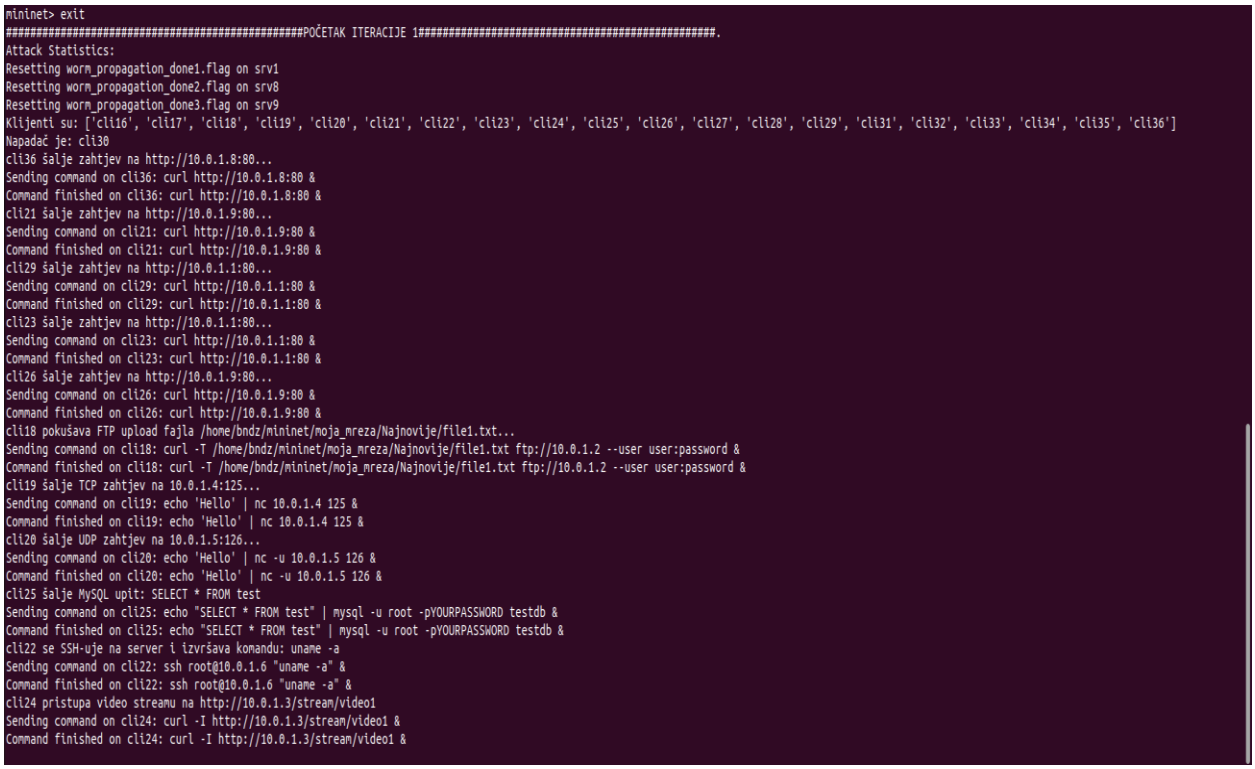
Na slikama 3.3. i 3.4. prikazana je mreža prije pokretanja, a na slici 3.5. prikazan je izgled terminala dok je skripta pokrenuta.



Slika 3.3. Prvi dio terminala u kojem je pokrenuta mreža



Slika 3.4. Drugi dio terminala u kojem je pokrenuta mreža



Slika 3.5. Prikaz rada skripte za komunikaciju

Na slici 3.6. prikazan je izgled dijela baze podataka, otvorene u *Sublime* uređivaču teksta.

Slika 3.6. Vrh baze podataka s vidljivim imenima stupaca

Postupak se ponavlja za izradu testnog i validacijskog skupa, samo smanjenog volumena (za trening skup korišteno je 500 iteracija komunikacije, za test skup 250 i za validacijski 50).

Nakon stvaranja sva 3 seta, konačni rezultat je baza podataka sveukupne veličine 2,48GB, razlog veličine je velik broj redova, oko 6,5 milijuna za trening set, 3,25 milijuna za test set i 650 tisuća za validacijski set.

4. EVALUACIJA PREDLOŽENOG RJEŠENJA I ZAKLJUČAK

Pregledom postojećih radova, odabrani su sljedeći algoritmi strojnog i dubokog učenja:

- **Gaussov Naivni Bayes (engl. *Gaussian Naive Bayes,* GNB)**
- **RF**
- **Klasifikator pojačavanja gradijenta histograma (engl. *Histogram gradient boosting classifier,* HGBC)**
- **CNN**
- **LSTM**

Trening skup je stvoren sustavom za stvaranje baze podataka, opisan u poglavlju iznad. Izrađen je tako što je sustav radio 500 iteracija, stvarajući otprilike 6500000 redaka. Test skup je izrađen od 250 iteracija, stvarajući otprilike 3250000 redaka. Isto tako je stvoren validacijski skup od 50 iteracija, stvarajući otprilike 650000 redaka.

Za odabir najboljih parametara algoritama RF, GNB i HGBC korištena je metoda *RandomizedSearchCV*. Prvo se postavi parametarska "mreža", odnosno definiraju se koji parametri se žele mijenjati i koje vrijednosti se žele koristiti. Shodno tome u daljnjem opisu algoritama, biti će opisani samo parametri čije su predefinirane vrijednosti (vrijednosti definirane u pripadnim bibliotekama algoritama) promijenjene. Kada se postavi parametarska "mreža", algoritam *RandomizedSearchCV* bira nasumične kombinacije odabranih vrijednosti parametara i s tim parametrima odradi treniranje i evaluaciju algoritama. CV u imenu *RandomizedSearchCV* znači da to odrađuje metodom križne validacije, više puta (u ovom slučaju 5) s istim parametrima trenira i evaluira algoritam s nasumično odabranim podskupovima trening skupa. Zbog vremena izvedbe odabrano je 10 iteracija, odnosno 10 kombinacija parametara, za koje je svaki algoritam treniran i evaluiran 5 puta. Dakle, kako bi se odabrali parametri, svaki algoritam je istreniran i evaluiran 50 puta. Nakon svih 50 treniranja i evaluacija, kombinacija parametara koja daje najveću F1-mjeru koristi se za treniranje (na cijelom trening skupu) i evaluaciju na testnom skupu.

Zbog mogućnosti korištenja balansiranja klasa pomoću težina, odnosno *class_weight='balanced'*, težine se dodjeljuju obrnuto proporcionalno broju uzoraka klasa, više zastupljena klasa (veći broj uzoraka) – manja težina i obrnuto. Ovime se rješava problem nebalansiranosti skupova podataka što može dovesti do pretreniranja. Ova metoda se koristi u svim algoritmima osim u GNB, gdje se koriste metode *oversampling* i *undersampling*, prvo se koristi oversampling na najmanje zastupljenoj klasi napada kako bi se povećao broj uzoraka tri puta. Nakon toga se ostale klase

napada metodom *undersampling* izjednače s tom klasom. Klasa benignog prometa se metodom *undersampling* smanji na 2 puta veću količinu uzoraka klasa napada.

Vrši se višeklasna klasifikacija, s ukupno 6 klasa koje su : „*benign,* DNS *amplification, Smurf attack, Syn flood,* UDP *flood* i *Port scanning*“, dakle benigni promet i 5 napada. Trening, test i validacijski skup ranije su ranije podijeljeni, dok se stvarala baza podataka, što je opisano u prethodnom poglavlju.

Prije treniranja, vrši se predobrada podataka, koja se sastoji od enkodiranja kategorijskih podataka, koristeći *Label encoder*, primjenjuje se na sva 3 skupa. Uz to se dodaje 7.5% šuma u podatke, što znači da se 7.5% vrijednosti zamijeni s nasumično odabranom postojećom vrijednošću, primjenjuje se samo na trening skup.

HGBC je moćan algoritam zasnovan na stablima odluke i tehnici pojačavanja, koja kako je već ranije opisano, gradi jedan snažni model kombinirajući više stabala.

Podesivi parametri su:

- *learning_rate* - brzina učenja, konstanta s kojom se množe listovi stabala, “0.01”
- *max_iter* - maksimalan broj iteracija odnosno stabala, “150”
- *max_depth* – maksimalna dubina stabla, “6”
- *min_samples_leaf* - minimalan broj uzoraka po listu, “100”
- *l2_regularization* - "kažnjavanje” listova koji koriste velike vrijednosti težina ulaznih parametara kako bi se spriječilo pretreniranje, “0.00001”

Vrlo je prilagodljiv za rad s različitim vrstama podataka, čak i s neodređenim brojčanim podacima (engl. *Not a Number*, NaN) [7].

GNB je jednostavan, ali skalabilan algoritam za klasificiranje koji je temeljen na *Bayesovom* teoremu i pretpostavci da su svi podaci normalno distribuirani (*Gaussova* distribucija) [8]. Podesivi parametar i odabrana vrijednost je

- *var_smoothing* - konstanta koja se dodaje varijanci modela da se izbjegne pretreniranje, “0.000000001”.

RF je algoritam nadziranog učenja koji iz ulaznih podataka stvara stabla odluke na temelju kojih dobije predviđanje za svako stablo. Stabla rade s izdvojenim podskupovima baze podataka (1 stablo = 1 podskup), kako bi se izbjeglo pretreniranje i postigla što veća generalizacijska moć.

Nakon svih dobivenih predviđanja, na temelju postignutih rezultata (metrika) algoritam bira najbolje stablo odluke [9]. Podesivi parametri i odabrane vrijednosti:

- n_*estimators* - broj stabala, "200"
- *criterion* - funkcija za mjerenje kvalitete podjele (proces dijeljenja čvora na dva ili više podčvorova), "*entropy"*
- *max_depth* - maksimalna dubina stabla, "6"
- *min_samples_split* – minimalan broj uzoraka za podjelu čvora (dio gdje se donosi odluka) stabla, "150"
- *min_samples_leaf* – minimalan broj uzoraka po listu (dio gdje se donosi konačna odluka o predviđanju) stabla, "250"

CNN je vrsta neuronske mreže sastavljena od konvolucijskih slojeva, koji primjenjuju konvolucijske filtere na ulazne podatke. Najčešće se koriste za obradu slike [10]. Korišteni model sastoji se od slojeva koji uključuju konvolucijske slojeve, slojeve za reduciranje dimenzija, slojeve sažimanja, *Flatten* slojeve i potpuno povezane slojeve. Struktura korištenog modela započinje preoblikovanjem ulaza na jednodimenzionalni vektor. Srž mreže sastoji se od 3 konvolucijska sloja, između svakog se nalazi sloj sažimanja (engl. p*ooling layer*) i sloj normalizacije po grupi (engl. *batch normalization layer*) kako bi se spriječilo pretreniranje koliko god je moguće (ukupno 3 sloja sažimanja i 3 sloja normalizacije). Nakon toga izlaz iz zadnjeg konvolucijskog sloja (nakon što prođe sloj sažimanja i sloj normalizacije), šalje se na *flatten* sloj, koji osigurava kompatibilnost s potpuno povezanim slojevima (engl. *dense layer*). Povezani slojevi daju na izlaz vjerojatnosti za svaku klasu, što omogućuje modelu donošenje konačne klasifikacije

Podesivi parametri slojeva (i slojevi) su:

- c*onv1D* – konvolucijski sloj, broj filtera, veličina filtera, koji određuju koliko informacija model prikuplja iz podataka te aktivacijska funkcija, koja određuje hoće li se ili neće koristiti neuron. "64/128/256", "3", "*ReLu*"
- *maxpooling1D* – sloj sažimanja, veličina sažimanja dimenzija značajki. "2"
- sloj normalizacije po grupi – normalizira ulazne vrijednosti svakog sloja u neuronskoj mreži tako da imaju standardnu distribuciju, sa srednjom vrijednošću nula i varijancom jedan. Normalizacija se provodi nad svakom mini-grupom podataka (engl. *batch*) "*default"*
- *dropout* - postotak neurona koji se ignoriraju tijekom treniranja, kako bi se spriječilo pretreniranje. "0.5"

- *Flatten* sloj, po potrebi se smanjuje dimenzija vektora koji se šalje u potpuno povezani(e) sloj(eve) i osigurava kompatibilnost s istim. "*default*"
- potpuno povezani sloj - broj neurona u potpuno povezanim slojevima koji daju konačna predviđanja, broj filtera, aktivacijska funkcija i L2 regularizacija. "256/6", "*ReLu/Softmax*", "0.00001" - samo prvi sloj

Ostali parametri se definiraju izvan slojeva i oni su:

- *learning_rate* - brzina učenja, kojom se model prilagođava tijekom svakog prolaska kroz podatke, „0.0001"
- *batch_size* - broj uzoraka po epohi „128"
- *epochs* - broj epoha (prolaza) kroz cijeli skup podataka „50"
- optimizator – minimizira funkciju gubitka, „*Adam*"
- *loss* – funkcija gubitka, mjeri mjeri koliko je predviđanje modela različito od stvarnih vrijednosti, „*sparse_categorical_crossentropy*"

LSTM je poseban tip rekurentnih neuronskih mreža RNN-a dizajniran za rad sa sekvencijalnim podacima. Sastoji se od posebnih LSTM stanica (engl. *LSTM cell*) koje imaju 3 dijela: **ulaz** - određuje koliko novih informacija ulazi u stanicu, **jedinica za zaborav** - određuje koje se informacije odbacuju i **izlaz** - određuje koje se informacije iz stanice prenose dalje. LSTM stanice omogućuju mreži da zapamti relevantne informacije iz prošlih vremenskih koraka i zanemari irelevantne. To omogućuje rješavanje problema dugoročnih ovisnosti u podacima, s čime standardni RNN modeli imaju poteškoća. Slojevi mogu biti ulazni, skriveni i izlazni [10]. Podesivi parametri su:

- broj LSTM jedinica - određuje broj memorijskih ćelija unutar LSTM sloja koje bilježe informacije kroz vremenske korake, „*default*"
- *dropout* isto kao u CNN, broj slojeva - omogućava slojevito postavljanje više LSTM stanica kako bi se poboljšala sposobnost modela da uči složene uzorke u podacima, „0.3"
- *batch size* - koliko uzoraka se koristi u svakoj iteraciji treniranja „128"
- vremenski korak - broj vremenskih koraka koje LSTM koristi za predviđanje u svakom trenutku, „*default*".

Testiranjem se pokazalo da LSTM nije dovoljno dobar za primjenu u svrhu IDS-a (značajno lošiji rezultati od CNN), jer se fokusira samo na vremenske sekvence, stoga je predložena ideja CNN-LSTM hibrida, kako bi se istražila performansa jednog hibridnog algoritma, u odnosu na ostale

"originalne". CNN-LSTM zamišljen je da koristi već postojeći CNN model za prepoznavanje ključnih značajki u vremenskim sekvencama. Korišteni su sljedeći parametri (navedeni samo oni čije su predodređene vrijednosti mijenjane): *learning_rate* - „0.0001", *batch_size* - „128", *epochs* - „50", optimizator – „*Adam*", *dropout* - „0.5". Izlaz iz CNN-a je kraći niz vremenski ovisnih ključnih značajki i koristi se kao ulaz za LSTM slojeve, sastoji se od 2 LSTM sloja, oba sloja imaju *dropout*=0.3 što znači da nasumično isključuje 30% neurona). Prvi ima aktiviran parametar *return_sequence=true* (vraća cjelokupnu sekvencu skrivenih stanja) jer idući sloj očekuje sekvencu kao ulaz. Drugi sloj LSTM-a radi sa sekvencama skrivenih stanja iz prvog sloja i na izlaz šalje samo posljednje stanje svake sekvence. Izlaz iz drugog LSTM sloja šalje se u *Flatten* sloj, iako je već vektor dimenzija 128x1, *flatten* sloj osigurava kompatibilnost s prvim potpuno povezanim slojem. Prvi potpuno povezani sloj pretvara ulaz u 256 neurona s *ReLu* aktivacijskom funkcijom. *ReLu* je odabran zbog uvođenja nelinearnosti u model (radi modeliranja složenih odnosa između značajki baze podataka) te za rješavanje problema nestajućeg gradijenta. Uz pomoć *dropout*=0.5, koristi se samo pola neurona, ideja iza toga je izbjeći pretreniranje. Na kraju uz pomoć posljednjeg potpuno povezanog sloja sa 6 neurona jer ima 6 klasa, sa *softmax* aktivacijskom funkcijom generiraju se vjerojatnosti za svaku klasu, što omogućuje modelu donošenje konačne klasifikacije.

Struktura predloženog hibrida CNN-a i LSTM-a vidljiva je na slici 4.1. Na istoj slici može se vidjeti i struktura ranije spomenutog CNN-a, dok je struktura ranije spomenutog LSTM-a identična CNN-u, jedina je razlika što su se umjesto konvolucijskih slojeva koristili LSTM slojevi. Radi zadržavanja što boljih rezultata, CNN, LSTM i CNN LSTM hibrid su koristili mogućnost ranog zaustavljanja (engl. *early stopping*) kako bi se algoritmi zaustavili dok su metrike i performanse najbolje (relativno), takav pristup je moguć jer navedeni algoritmi se epohalno (iterativno) treniraju. Algoritmi RF, HGBC i GNB nemaju te mogućnosti već prolaze jednom kroz cijeli trening skup, stoga je nemoguće pratiti njihove metrike kroz treniranje.

Korištene metrike su:

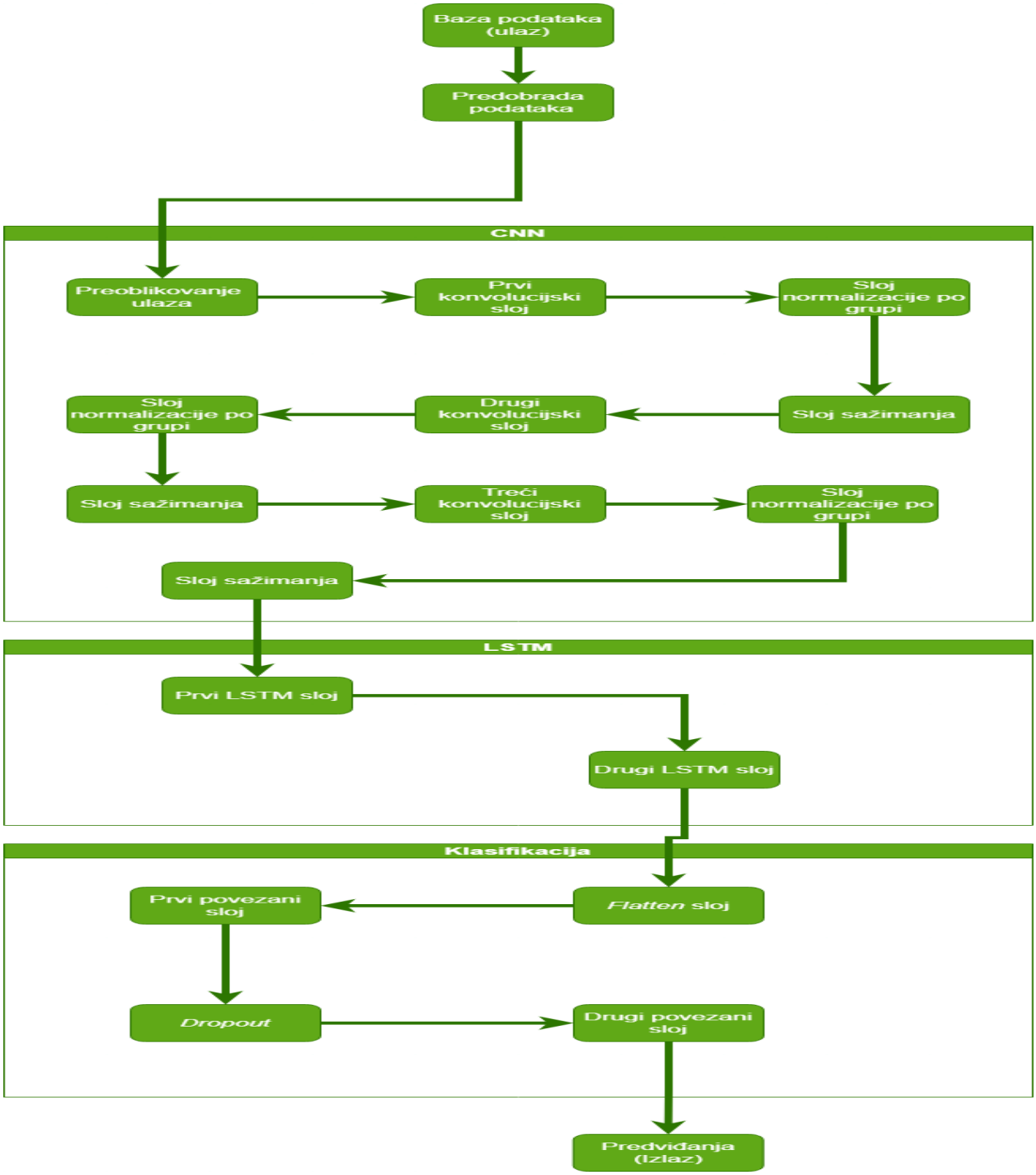
Točnost (engl. *Accuracy*) = $\frac{TP}{TP+TN+FP+FN}$, gdje su stvarno pozitivno predviđanje (engl. *true positive*, TP), stvarno negativno predviđanje (engl. *true negative*, TN), lažno pozitivno predviđanje (engl. *false positive*, FP) i lažno negativno predviđanje (engl. *false negative*, FN) [1].

Preciznost (engl. *precision*) = $\frac{TP}{TP + FP}$, mjera za izračun koliko dobro model predviđa.

Odziv = $\frac{TP}{TP+FN}$, predstavlja koliko dobro model prepoznaje stvarno pozitivno predviđanje.

F1-mjera $= 2 \cdot \frac{preciznost \cdot osjetljivost}{preciznost + osjetljivost}$, harmonijska sredina preciznosti i odziva, što znači da uzima u obzir obje mjere i daje balansiranu ocjenu modela.

Broj lažno pozitivnih rezultata (engl. *false positive rate*) $= \frac{FP}{TP}$ koliko često model netočno klasificira, npr. za benigni promet kaže da je jedan od napada.



Slika 4-1.Struktura predloženog CNN-LSTM modela

Dodatne metrike su:

- Krivulja radne karakteristike primatelja (engl. *receiver operating characteristic*, ROC) grafički prikaz performansi klasifikacijskog modela, na X osi se nalazi broj stvarno pozitivnih rezultata, a na Y osi broj lažno pozitivnih rezultata [12].
- Površina ispod krivulje (engl. *area under the curve* (AUC) je grafički prikaz koliko dobro model razlikuje pozitivne rezultate od negativnih, odnosi se na ROC krivulju [12]. Rezultat može biti između 0 i 1, što označava snagu klasifikatora (koliko će puta točno predvidjeti).
- Matrica zabune (engl. *confusion matrix*) - daje detaljan prikaz kako je model klasificirao stvarne i predviđene uzorke u matričnom prikazu, može se točno vidjeti gdje je ostvareno točno predviđanje, isto tako ako je netočno predviđanje, možemo vidjeti s kojom je klasom zamijenjeno [13].

Metrike preciznost, odziv i F-mjera su rađene u 3 verzije:

- Makro (engl. *macro*) – metrika se računa na razini svake klase, a zatim se računa prosjek metrika svih klasa kao konačna vrijednost [14]. Npr. izračuna se točnost za svaku klasu i koristeći izračunate točnosti izračuna se prosječna vrijednost te metrike.
- Mikro (engl. *micro*) – metrika se računa kao konačna vrijednost na razini svih uzoraka, sve pozitivne i negativne klasifikacije se uzimaju u obzir, bez obzira kojoj klasi pripadaju. Dakle klase se zanemaruju i gleda se ukupno cijeli skup podataka kao jedna cjelina [14].
- Ponderirana (engl. w*eighted*) – metrika se računa na razini svake klase, isto kao i makro, no prije konačnog izračuna, izračunate metrike se množe s težinom koju algoritam samostalno dodijeli klasama [14].

U tablici 4.1. prikazani su rezultati evaluacije algoritama, a na slikama od 4.2., 4.3., 4.4., 4.6., 4.7., 4.9., 4.10., 4.11., 4.12., 4.14., 4.15., 4.16. predočene su metrike ROC/AUC i matrica zabune za sve algoritme. Također na slikama 4.5., 4.8. i 4.13. prikazani su grafovi točnosti i gubitka (engl. a*ccuracy and loss graph*) za CNN, LSTM i CNN LSTM hibrid.

Metrike označene crvenom bojom u tablici 4.1. su ključne metrike koje su korištene u daljnjoj evaluaciji algoritama. Označene su crvenom bojom radi lakšeg snalaženja u tablici i smanjivanja pogreške korištenja krive metrike, npr. makro preciznosti umjesto ponderirane preciznosti.

Ime algoritma / Metrike	**GNB**	**RF**	**HGBC**	**CNN**	**LSTM**	**CNN-LSTM hibrid**
Točnost	48.73%	88.34%	93.98%	90.01%	78.03%	91.43%
Ponderirana preciznost	92.09%	98.92%	99.27%	98.96%	94.47%	99.12%
Makro preciznost	23.87%	66.10%	76.59%	67.35%	37.52%	68.83%
Mikro preciznost	48.73%	88.34%	93.98%	90.01%	78.03%	91.44%
Ponderirani F1	60.84%	92.92%	96.31%	93.95%	84.86%	94.81%
Makro F1	21.30%	68.02%	79.98%	68.69%	38.24%	69.96%
Mikro F1	48.73%	88.34%	93.98%	90.01%	78.03%	91.44%
Ponderirani odziv	48.73%	88.34%	93.98%	90.01%	78.03%	91.44%
Makro odziv	41.02%	97.19%	98.22%	90.91%	62.93%	91.93%
Mikro odziv	48.73%	88.34%	93.98%	90.01%	78.03%	91.44%
Broj lažno pozitivnih predviđanja	10.25%	2.33%	1.20%	2.00%	4.39%	1.71%

Tablica 4.1.Rezultati multiklasne klasifikacije

Pregledom rezultata, odrađen je ponderirani prosjek metrika, uzimajući u obzir samo ponderirane metrike, točnost i broj lažno pozitivnih predviđanja. Shodno tome F1 ima težinu 40%, odziv 20%, broj lažno pozitivnih predviđanja 20%, preciznost 15% i točnost 5%.

Radi razumijevanja dolje opisanih rezultata, opisat će se postupak izračuna ponderiranog prosjeka GNB algoritma. Dakle, **crveno označene metrike iz tablice množe se s njihovim težinama i zbrajaju međusobno**, jedina iznimka je metrika broja lažno pozitivnih rezultata, čiji rezultat se invertira (u ovom slučaju, 1-0.0125=0.875). Broj lažno pozitivnih rezultata invertira se iz razloga što je to obrnuto proporcionalna varijabla, što je niža, označava bolji rezultat.

Zbog preglednosti u formuli, za broj lažno pozitivnih predviđanja, koristit će se kratica „blpp“.

$$(\mathbf{F1} * \mathbf{0.4} + \mathbf{odziv} * \mathbf{0.2} + (\mathbf{1} - \mathbf{blpp}) * \mathbf{0.2} + \mathbf{preciznost} * \mathbf{0.15} + \mathbf{točnost} * \mathbf{0.05}) * \mathbf{100}$$

$$= (0.6084 * 0.4 + 0.4873 * 0.2 + 0.8975 * 0.2 + 0.9209 * 0.15 + 0.4873 * 0.05) * 100$$

$$= (0.2434 + 0{,}0975 + 0{,}1795 + 0{,}1381 + 0{,}0244) * 100$$

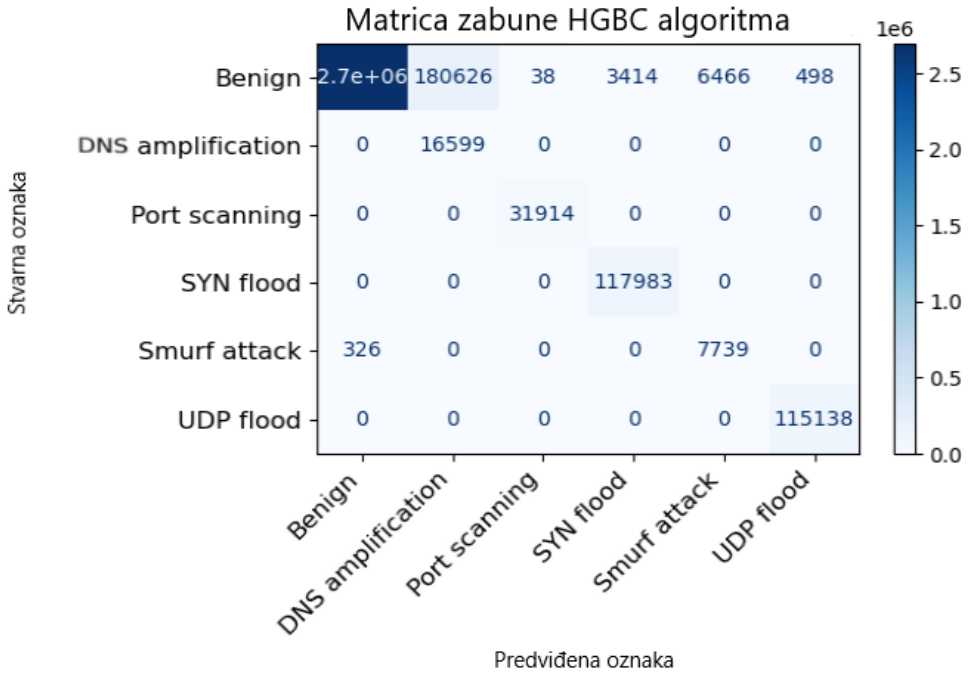
$$= (0.6829) * 100$$

$$= \mathbf{68.29}\%$$

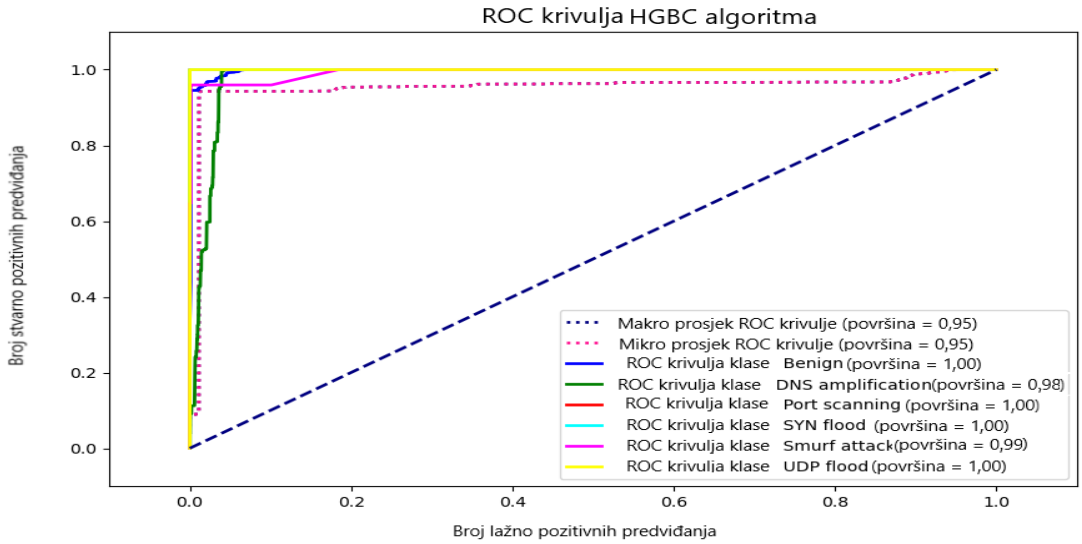
Prateći gornji postupak, dobiven je sljedeći poredak:

- HGBC **96.67%**
- CNN-LSTM hibrid **95.31%**
- CNN **94.53%**
- RF **93.63%**
- LSTM **86.74 %**
- GNB **68.29%**

Prije konačne evaluacije treba uzeti u obzir i vizualne metrike, priložene fotografije 4.2. do 4.16., postavljene su, radi lakšeg pregleda, prema gornjem poretku.



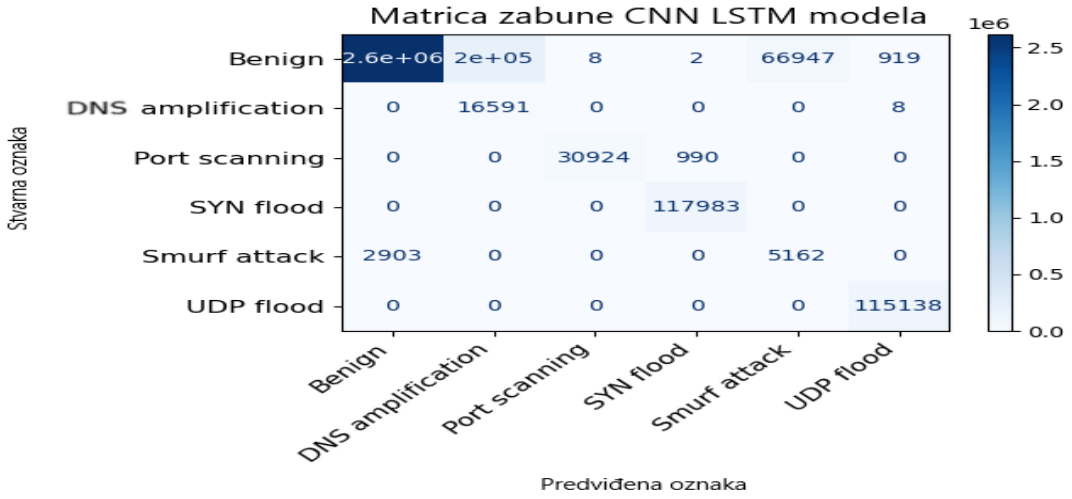
Slika 4-2. Matrica zabune HGBC algoritma



Slika 4.3. ROC/AUC metrike HGBC algoritma

Na slici 4.2. vidljiva je matrica zabune HGBC algoritma. Primjećuje se da je model jako dobro naučio kako razlikovati između napada i benignog prometa, uz iznimku malog dijela krivog klasificiranja *smurf* napada kao benigni promet. Model je dosta „oprezan“, ali precizan, što objašnjava mali broj benignih paketa označenih kao neki od napada, kao i samo 326 paketa *smurf* napada koji nisu pravilno klasificirani.

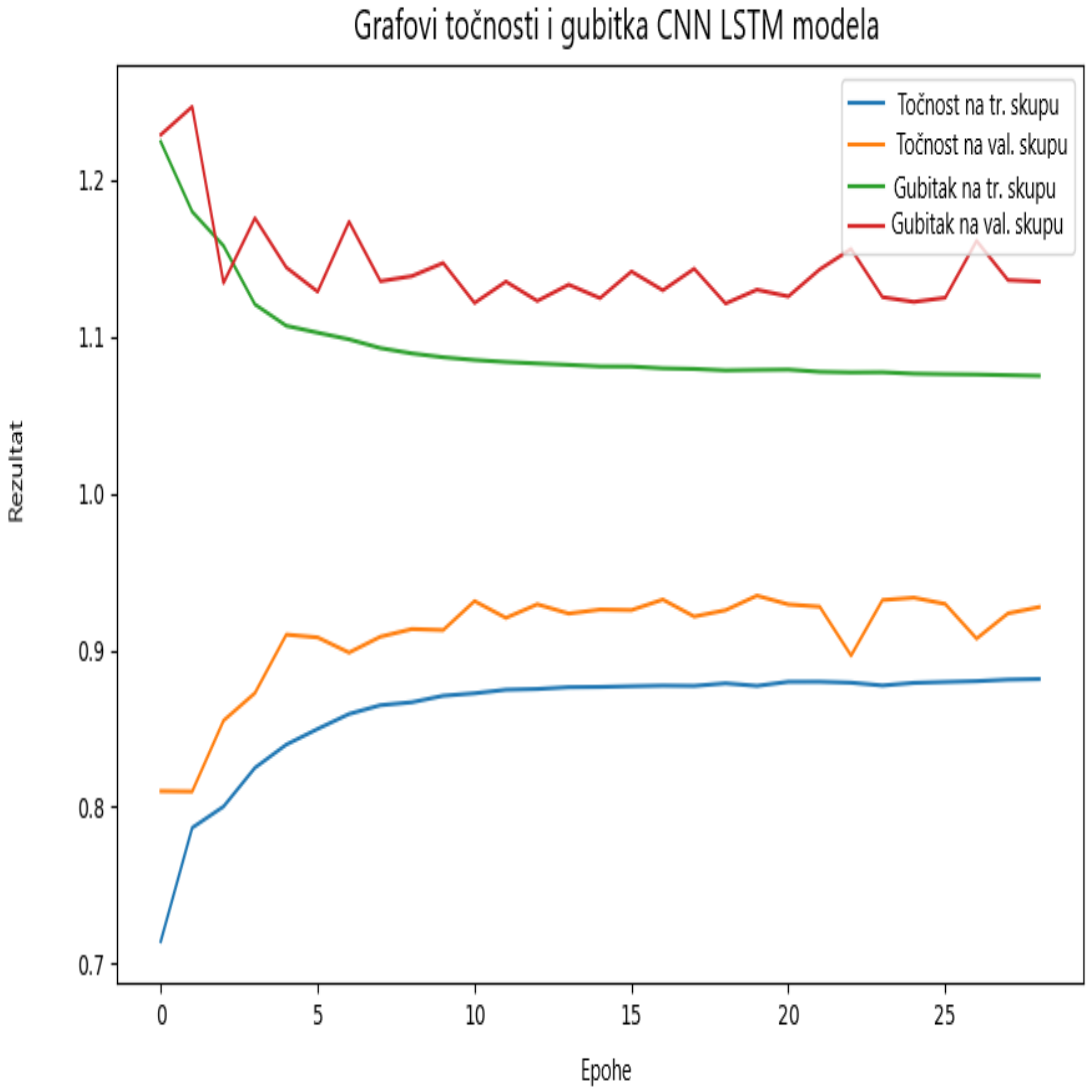
Na slici 4.3. vidljive su ROC krivulje klasa napada, gdje se uočava velik broj savršenih klasifikatora, što znači da je površina ispod 5 krivulja je jednaka 1, 5 klasifikatora točno predviđa svoju klasu u 100% slučajeva. To može biti znak potencijalnog pretreniranja, odnosno pretjeranog prilagođavanja trening skupu.



Slika 4.4. Matrica zabune CNN LSTM modela

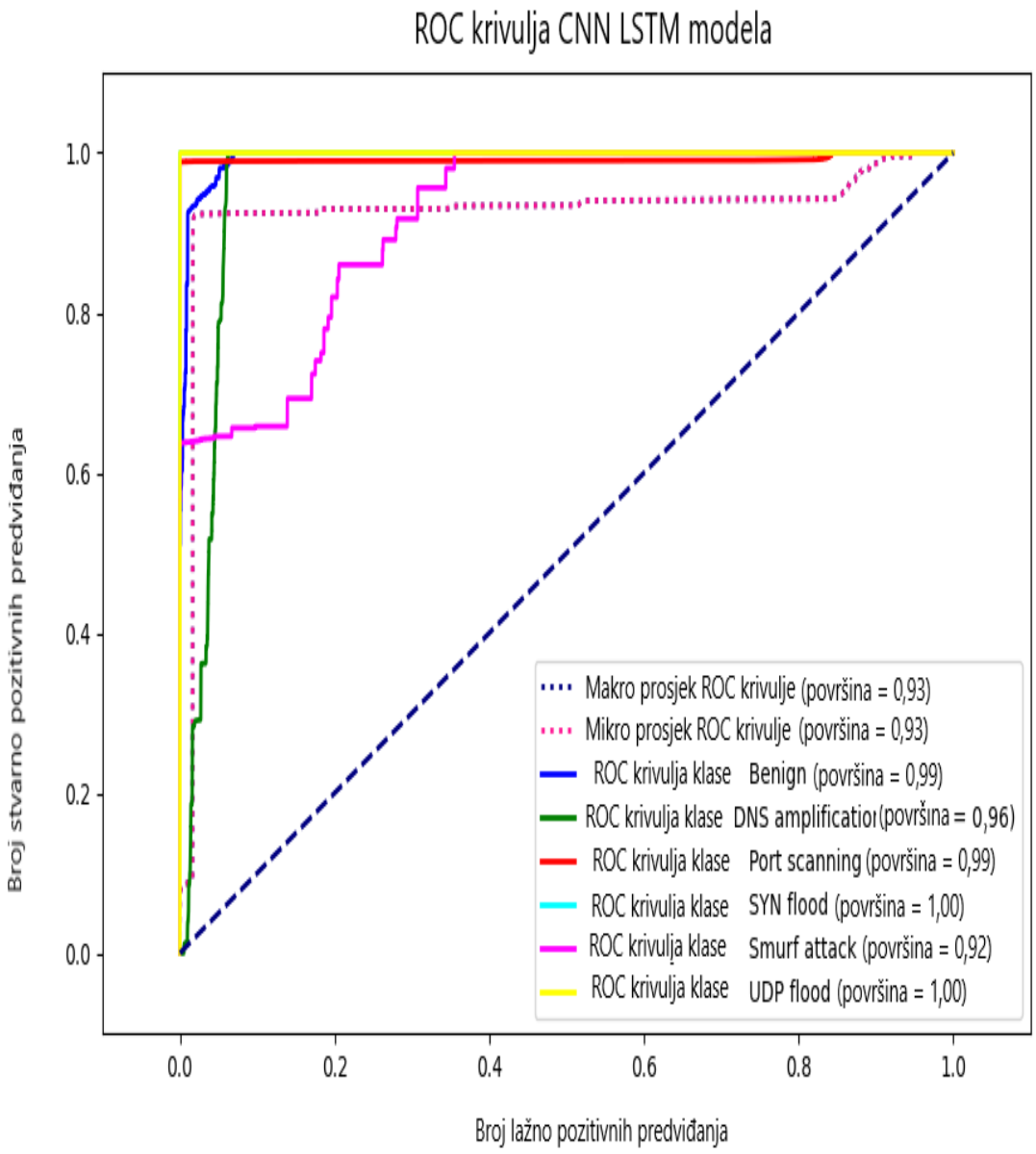
Na slici 4.4. vidljivi su jako dobri rezultati rada CNN LSTM modela. Ovaj model je relativno dobro naučio razlikovati napade od benignog promete, ali se pati s razlikovanjem vrsta napada, najviše *port scanninga* i SYN *flood*. Donekle lošije performanse od HGBC algoritma vidljive su po tome što je više benignih paketa označeno kao napadi, dakle model je dosta oprezan, kao i HGBC algoritam, ali malo manje precizan s obzirom da je krivo klasificirao 2903 paketa *smurf* napada kao benigni promet, skoro 9 puta više nego HGBC algoritam.

Na slici 4.5. vidljivi su grafovi točnosti i gubitka na trening i validacijskim skupovima. Nema naznaka pretreniranja, s obzirom na relativno male oscilacije gubitka i točnosti na validacijskom skupu, što upućuje na to da ovaj algoritam može relativno dobro generalizirati na neviđenim podacima.



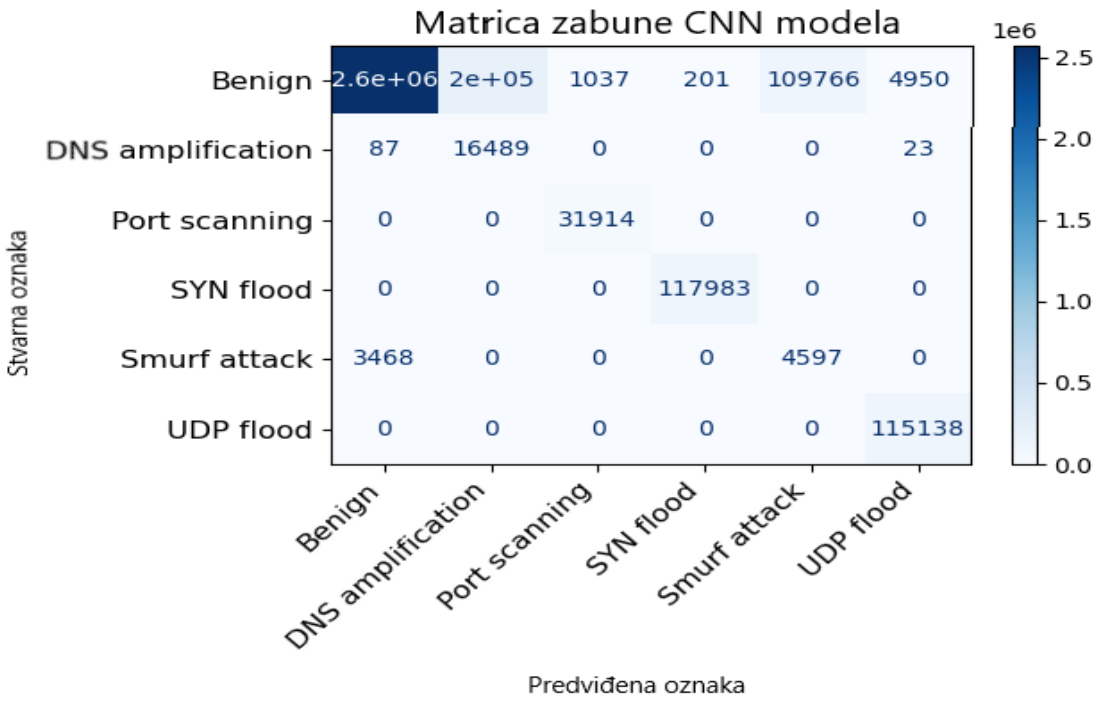
Slika 4.5. Grafovi točnosti i gubitka CNN LSTM modela

Na slici 4.6. prikazane su ROC krivulje CNN LSTM modela, gdje je vidljiv dosta manji broj savršenih klasifikatora, što govori da model jako dobro generalizira, a da pritom nije pretreniran. Savršeni klasifikatori UDP i SYN *flooda* mogu se objasniti jednostavnošću tih napada, poslužitelj se preplavljuje UDP paketima ili TCP paketima sa „*syn*“ zastavicom.



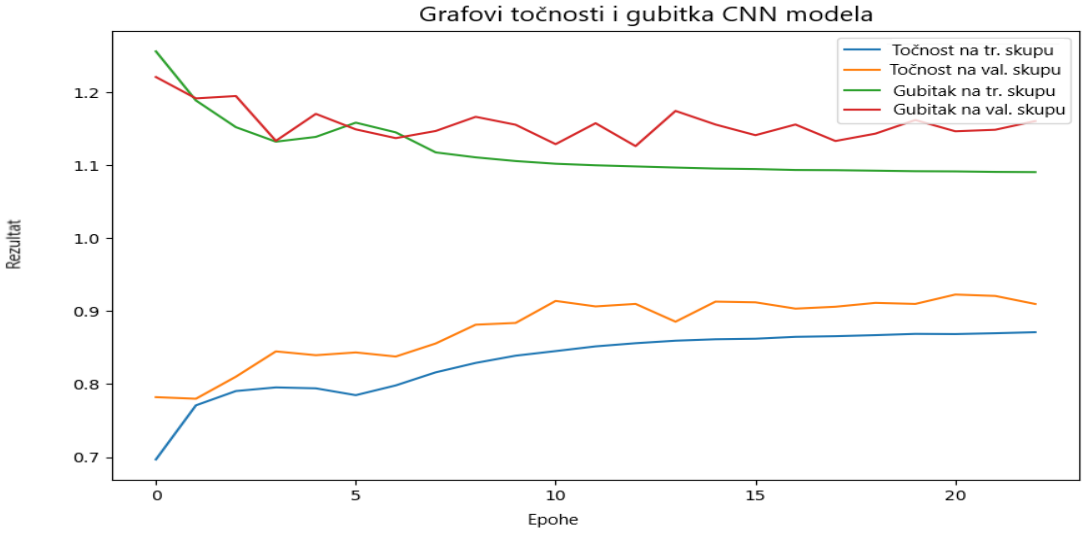
Slika 4.6. ROC/AUC metrike CNN LSTM modela

Na slici 4.7. vidljiva je matrica zabune CNN modela. Generalizacijska moć nešto je lošija od predloženog CNN LSTM modela i HGBC algoritma, što se može potvrditi većom opreznošću nego prethodna dva modela, uz nižu preciznost. Isto tako, ovaj algoritam je osim 3468 paketa *smurf* napada, propustio i 87 paketa DNS *amplificationa*, jedna vrsta napada više koja je promakla nego u prethodna dva modela.



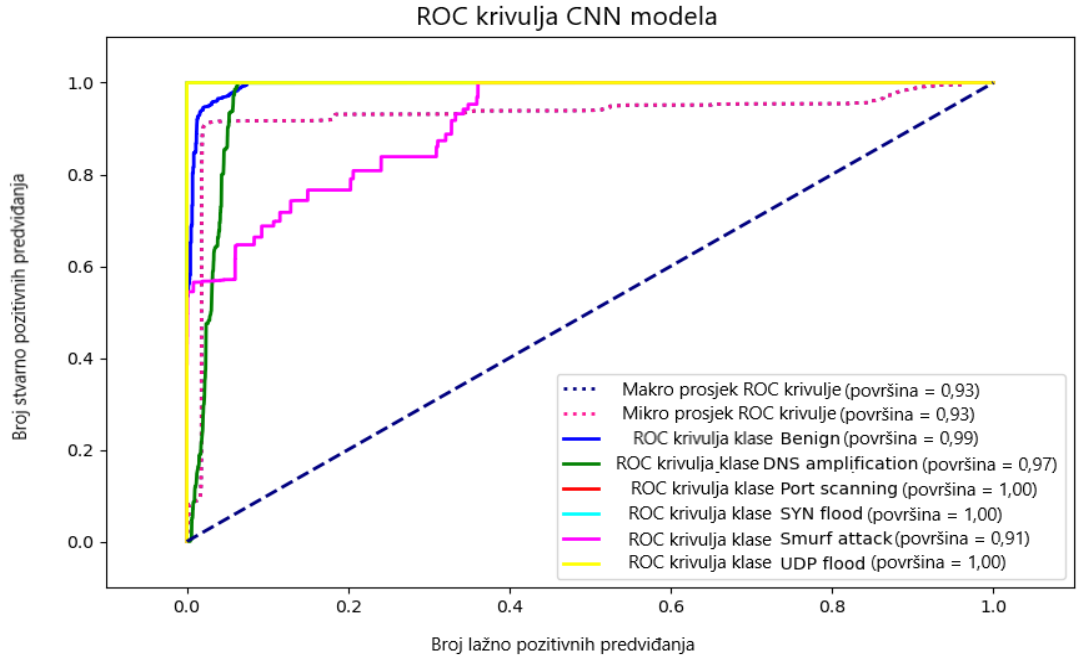
Slika 4.7. Matrica zabune CNN modela

Na slici 4.8. primijećene su nešto veće oscilacije u grafovima gubitka na trening i validacijskom skupu, što ukazuje na to da je model nešto slabije naučio poveznice između napada i benignog prometa u usporedbi s predloženim CNN LSTM modelom.



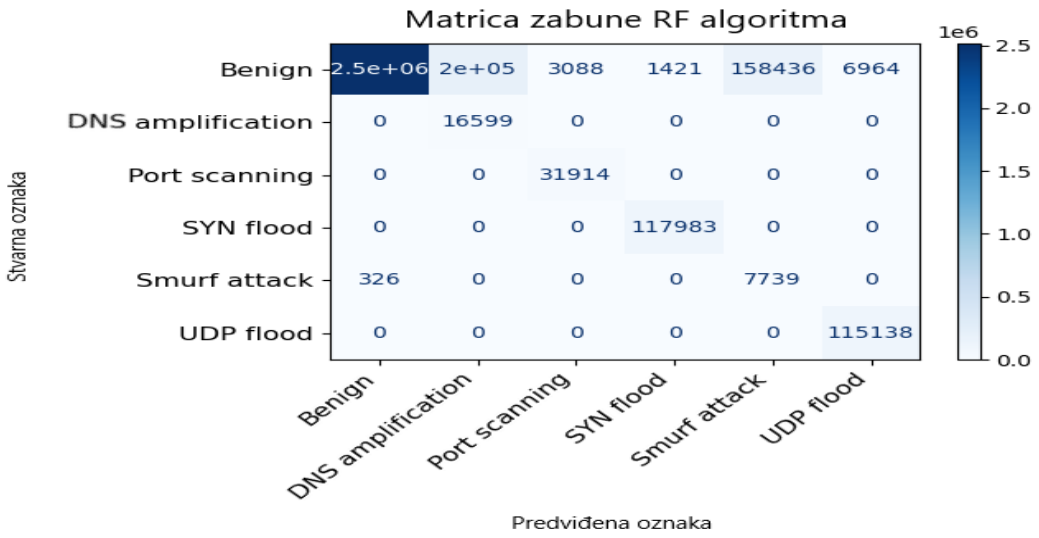
Slika 4.8. Grafovi gubitaka i točnosti CNN modela

Na slici 4.9. vidljiva su samo 3 savršena klasifikatora, u usporedbi s CNN LSTM modelom je lošije, ali i dalje dovoljno dobro za pouzdano generaliziranje većine novih napada.



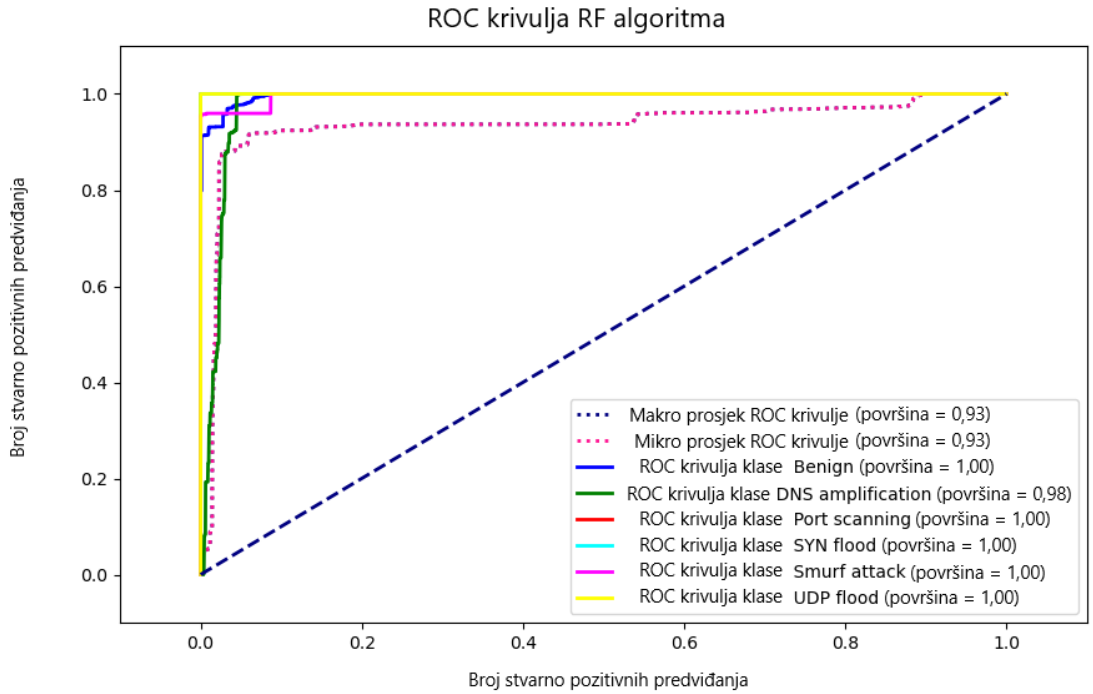
Slika 4.9. ROC/AUC metrika CNN modela

Na slici 4.10. vidljiva je matrica zabune RF algoritma, na kojoj se vidi da je model dobro naučio međusobno razlikovati napade, dok je i dalje dosta oprezan s nižom preciznošću od prethodnih algoritama. Ovaj algoritam je propustio samo 326 paketa *smurf* napada, što je sumnjivo, s obzirom na lošije metrike od prethodnih modela.



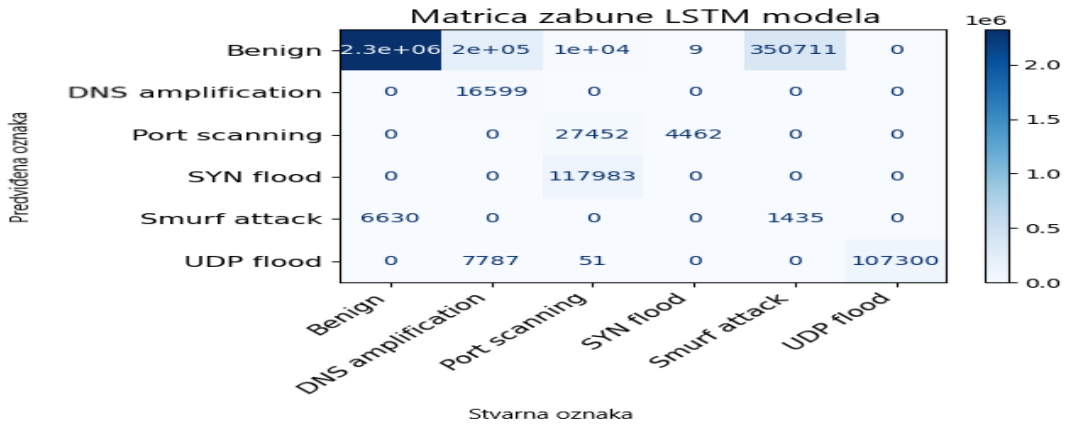
Slika 4.10. Matrica zabune RF algoritma

Na slici 4.11. vidljiva je ROC krivulja RF algoritma, dosad najveći broj savršenih klasifikatora potvrđuje sumnje navedene u opisu prethodne slike. Dakle slična situacija kao i s HGBC algoritmom, model se pretjerano prilagodio trening skupu.



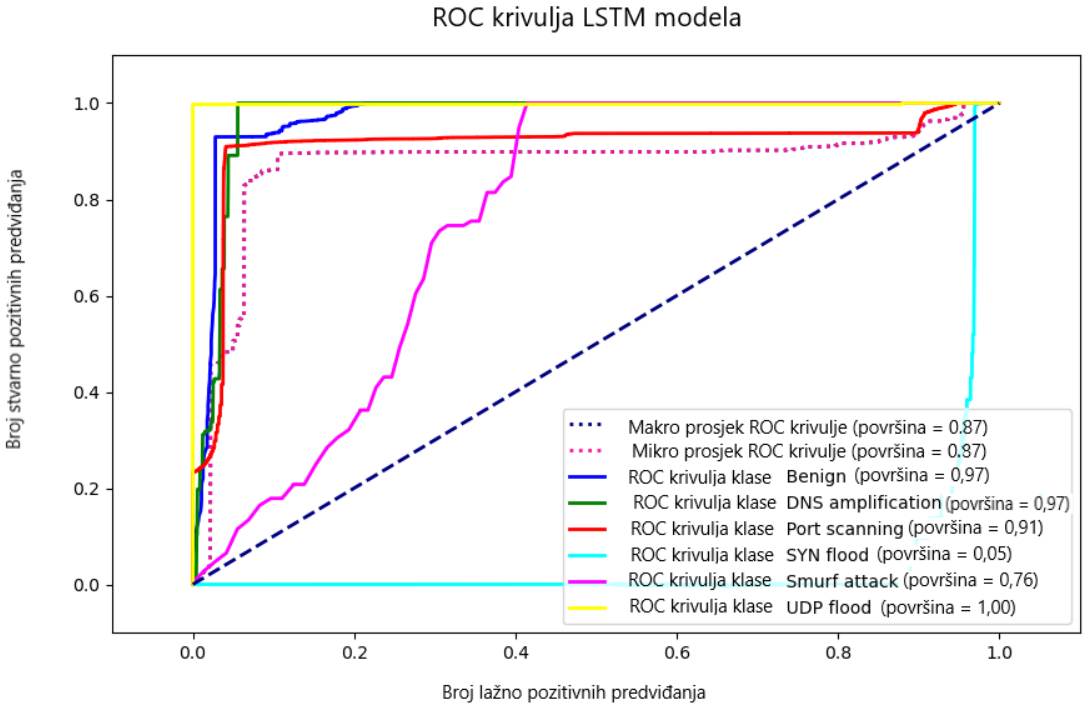
Slika 4.11. ROC/AUC metrika RF algoritma

Na slici 4.12. vidljiva je matrica zabune LSTM modela. Model je dosta oprezniji od prethodnih modela, ali i s punom manjom preciznošću, što potvrđuju i lošije performanse modela. Iako je model propustio samo jednu vrstu napada, rezultati potvrđuju ranije navode da je LSTM model naučio samo vremenske uzorke, dok je poveznice između napada i benignog prometa slabije shvatio.



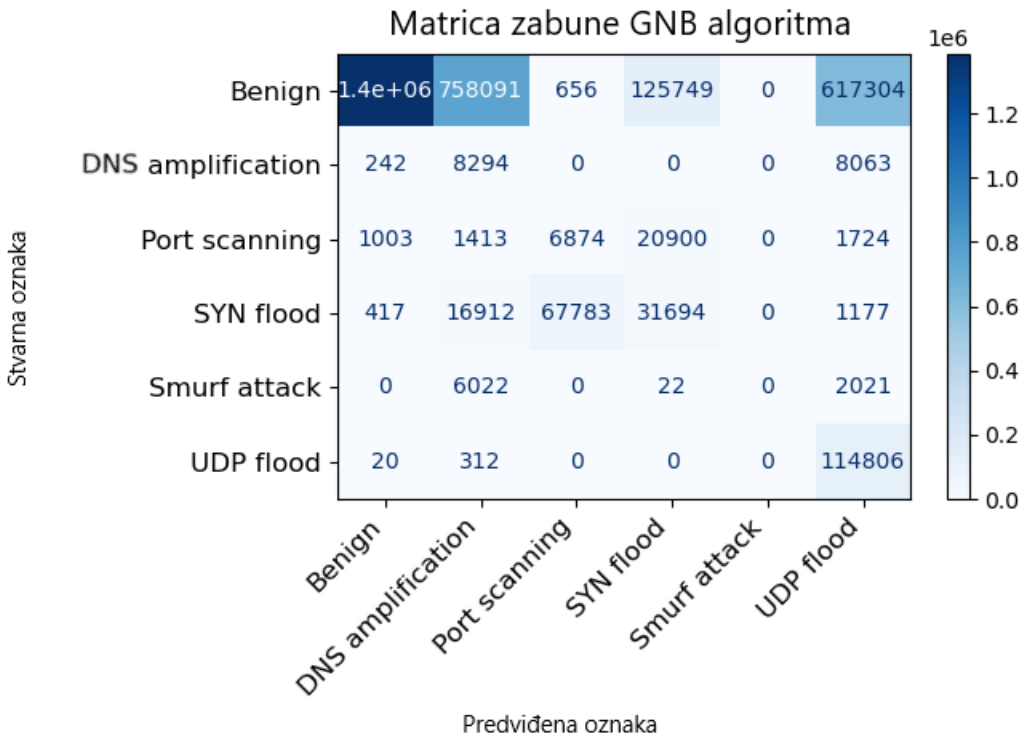
Slika 4.12. Matrica zabune LSTM modela

Slika 4.13. Grafovi gubitaka i točnosti LSTM modela

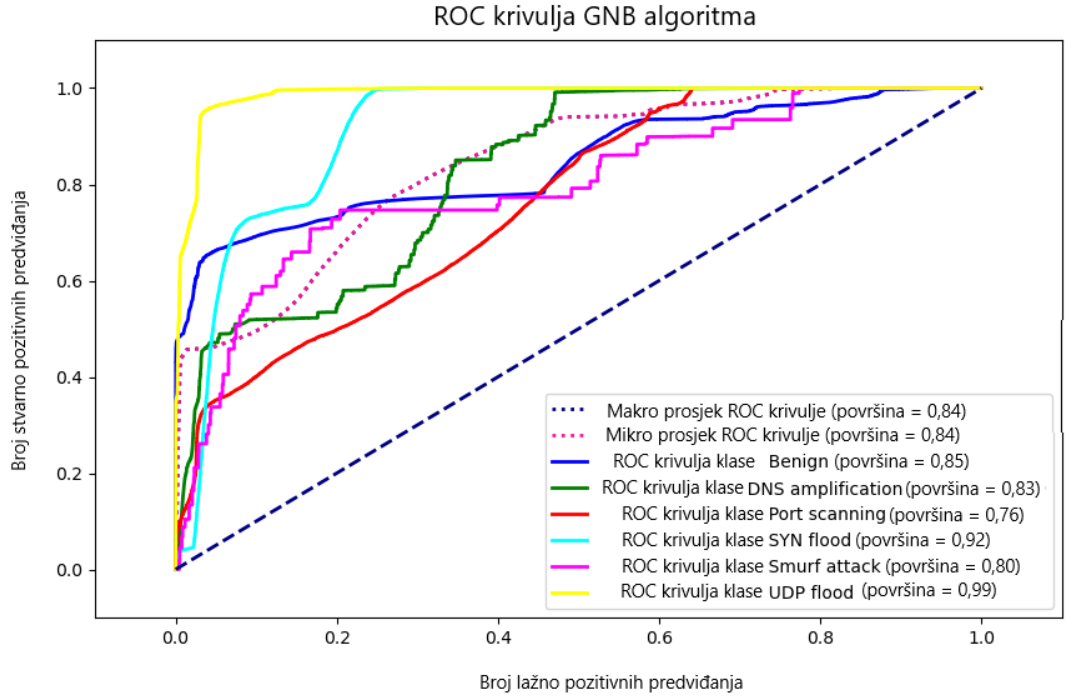


Slika 4.14. ROC/AUC metrika LSTM modela

Slike 4.13. i 4.14. potvrđuju navode iz opisa slike 4.12. Samo jedan savršen klasifikator dobar je znak da se model nije pretjerano prilagodio trening skupu. Dosta lošija generalizacijska moć vidljiva je i s ROC krivuljom klase SYN *flood* i *smurf* napad, gdje je model, za razliku od svih prethodnih modela slabije naučio klasificirati navedene klase.



Slika 4.15. Matrica zabune GNB algoritma



Slika 4.16. ROC/AUC metrika GNB-a

Na slikama 4.15. i 4.16. smatra se vizualnim opravdanjem najlošijih rezultata GNB algoritma od svih evaluiranih algoritama. Jedina vrsta napada koju model nije propustio je *smurf* napad. Isto

tako slabo međusobno razlikuje napade što je vidljivo najvećim brojem lažno pozitivnih predviđanja od svih algoritama.

Grafičke metrike potvrđuju poredak stvoren ponderiranim prosjekom, iako velik broj savršenih klasifikatora za HGBC sugerira moguće pretreniranje, stoga se može reći da predloženi CNN-LSTM hibrid ima najbolju generalizacijsku moć (samo 2 savršena klasifikatora, mala vjerojatnost pretreniranja). Pretpostavlja se da bi dodatnim finim podešavanjem parametara, CNN-LSTM hibrid mogao postići još bolje rezultate od HGBC-a.

ZAKLJUČAK

U ovom radu, baza podataka korištena za trening i evaluaciju algoritama strojnog učenja, stvorena je prikupljanjem sintetičkog mrežnog prometa. Mrežni promet, benigni i maliciozni, realiziran je u mrežnom simulatoru *mininet.* Mreža je stvorena kao skup servera i klijenata koji međusobno komuniciraju uz pomoć upravljača, usmjerivača i 2 preklopnika. Nakon određenog vremena dogodi se jedan od realiziranih napada. Realizirani su napadi: *smurf* napad, DNS *amplification, syn flood*, UDP *flood* i *port scanning*.. Uz pomoć *tsharka,* konzolne verzije *Wiresharka,* promet je spremljen u PCAP datoteke iz kojih su izvučeni podaci i sortirani u CSV datoteke. Za treniranje i evaluaciju, korišteni su algoritmi strojnog i dubokog učenja, koji su GNB, RF, HGBC, CNN i LSTM. Isto tako je predložen i hibrid CNN-LSTM, kako bi se stvorio uvid u mogućnosti i generalizacijsku moć hibridnih algoritama. Korištene metrike su točnost, preciznost, odziv, F1 rezultat i broj lažno pozitivnih predviđanja, osim numeričkih metrika, korištene su i matrica zabune, ROC krivulja i AUC za sve algoritme, kao i grafovi točnosti i gubitka za modele CNN, LSTM i CNN LSTM hibrid. Evaluacija se izvršavala na testnom skupu, uzimajući u obzir navedene numeričke i grafičke evaluacijske metrike.

Pregledom i analizom rezultata jasno je da različiti algoritmi strojnog učenja imaju svoje prednosti i mane kada ih se primjeni za detekciju upada u mrežu. Svi algoritmi, osim LSTM i GNB pokazali su se vrlo učinkovitima u višeklasnoj klasifikaciji odnosno razlikovanju napada od benignog prometa, što je vidljivo poretkom prema ponderiranom prosjeku:

- HGBC **96.67%**
- CNN-LSTM hibrid **95.31%**
- CNN **94.53%**
- RF **93.63%**
- LSTM **86.74 %**
- GNB **68.29%**

Zbog velikog broja savršenih klasifikatora, za HGBC koji je rangiran prvi po ponderiranom prosjeku, može se pretpostaviti blago pretreniranje. Razlog tome je prevelika prilagodba trening skupu na većini klasa (ROC=1 za određene klase) i može se pretpostaviti da na novijim neviđenim podacima možda i ne bi toliko generalizirao, slična situacija je i s RF-om. Nadalje, drugo rangirani, predloženi CNN-LSTM hibrid ima najmanji broj pretreniranih klasa (samo 2 od 6), nakon čega slijedi CNN (3 od 6) i može se pretpostaviti da bi se daljnjim finim podešavanjem parametara postigli još bolji rezultati ponderiranih metrika (i potencijalno u potpunosti iskorijenilo

pretreniranje). LSTM, je predzadnji u poretku, loše rezultate objašnjava njegov način rada, odnosno sve klasifikacije je temeljio samo na vremenskim uzorcima. GNB algoritam pokazao se najmanje učinkovitim zbog pretpostavke o neovisnosti značajki, što u slučaju kompleksnih mrežnih podataka, koji su vrlo često povezani preko više značajki, nije realistična pretpostavka.

Za kraj, može se reći da su algoritmi dubokog učenja, prikladniji za upotrebu kao IDS, iako imaju marginalno niže rezultate od nekih algoritama strojnog učenja, upravo zbog veće mogućnosti prilagodbi strukture modela i podesivih parametara.

LITERATURA

[1] „*Intrusion detection systems*“, GeeksForGeeks, 18.6.2024., dostupno na „https://www.geeksforgeeks.org/intrusion-detection-system-ids/”, datum pristupa 23.7.2024.

[3] Y.S.Almutairi, B. Alhazmi , A. Munshi, „*Network intrusion detection using machine learning techniques*“, ResearchGate, srpanj 2022., dostupno na „https://www.researchgate.net/publication/361672089_Network_Intrusion_Detection_Using_Machine_Learning_Techniques“ , datum pristupa 25.7.2024.

[4] „*ISCX NSL-KDD dataset 2009*“, University of New Brunswick, 2009., dostupno na: „https://www.unb.ca/cic/datasets/nsl.html“, datum pristupa 26.7.2024.

[5] Y.K.Saheed, A. I. Abiodun, S. Misra, M.K. Halone, R. Colomo-Palacios, „*A machine learning-based intrusion detection for detecting internet of things network attacks*“, ScienceDirect, 28. ožujak 2022., dostupno na „https://www.sciencedirect.com/science/article/pii/S1110016822001570“, datum pristupa 26.7.2024.

[6] T.Kim, W.Pak, „*Robust network intrusion detection system based on machine-learning with early classification*“, IEEE Xplore, 20.siječanj 2022., dostupno na „https://ieeexplore.ieee.org/abstract/document/9687553“, datum pristupa 29.7.2024.

[7] “*HistGradientBoostingClassifier*”, scikit-learn, 2024., dostupno na “https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.HistGradientBoostingClassifier.html “, datum pristupa 12.9.2024.

[8] “*GaussianNB*”, scikit-learn, 2024., dostupno na “https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html“, datum pristupa 12.9.2024.

[9] “*RandomForestClassifier*”, scikit-learn, 2024., dostupno na “https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html“, datum pristupa 12.9.2024.

[10] “*Introduction to Convolution Neural Network*”, GeeksForGeeks, 9.9.2024., dostupno na “https://www.geeksforgeeks.org/introduction-convolution-neural-network/ “, datum pristupa 13.9.2024.

[11] "*What is LSTM – Long Short Term Memory?*", GeeksForGeeks, 10.6.2024., dostupno na "https://www.geeksforgeeks.org/deep-learning-introduction-to-long-short-term-memory/ ", datum pristupa 14.9.2024.

[12] "*Classification: ROC and AUC*", Google Developers Machine learning, 3.9.2024., dostupno na "https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc ", datum pristupa 14.9.2024.

[13] "*Understanding the Confusion Matrix in Machine Learning*", GeeksForGeeks, 18.7.2024., dostupno na "https://www.geeksforgeeks.org/confusion-matrix-machine-learning/ ", datum pristupa 14.9.2024.

[14] "*Averaging Methods*", Testing with Kolena, 28.8.2024., dostupno na "https://docs.kolena.com/metrics/averaging-methods/ ", datum pristupa 14.9.2024.

SAŽETAK

U ovom radu istražena je primjena algoritama strojnog i dubokog učenja za detekciju upada u mrežu (engl. *Intrusion Detection System*, IDS). U radu je izrađena baza podataka simuliranih napada, pregled, odabir i implementacija različitih algoritama strojnog i dubokog učenja, te evaluacija njihove učinkovitosti za višeklasnu klasifikaciju. U bazi podataka realizirano je 5 vrsta napada: *syn flood*, UDP *flood*, DNS *amplification*, *smurf* napad i *port scanning*. Predložena je struktura hibrida CNN-a (engl. *Convolutional Neural Network*) i LSTM-a (engl. *Long short term memory*) radi istraživanja uporabe dobrih strana oba algoritma. Algoritmi poput CNN-a i predloženog CNN LSTM hibrida, pokazali su visoku učinkovitost, dok je *Histogram Gradient Boosting Classifier* (HGBC) nadmašio ostale, ali pod cijenu potencijalnog pretreniranja. Zaključak rada je da je hibridni pristup CNN LSTM vrlo korisna kombinacija dvaju moćnih algoritama dubokog učenja, jedina prepreka je pronalazak optimalnih parametara.

Ključne riječi: strojno učenje, duboko učenje, sustav za detekciju upada u mrežu, mrežna sigurnost

NETWORK INTRUSION DETECTION SYSTEM BASED ON MACHINE LEARNING ALGORITHM

ABSTRACT

In this paper, the application of machine and deep learning algorithms for an intrusion detection system is explored. The paper includes the creation of a simulated attack database, review, selection and implementation of various machine and deep learning algorithms as well as the evaluation of their multiclass classification effectiveness. 5 network attacks were succesfully recreated for use in the database: SYN flood, UDP flood, DNS amplification, smurf attack and port scanning. A hybrid convolutional neural network (CNN) and long short term memory (LSTM) structure is suggested, to use the good sides of both algorithms. Algorithms such as CNN and the suggested CNN LSTM hybrid have shown very high efficiency, while the Histogram Gradient Boosting Classifier (HGBC) surpassed all of the evaluated algorithms, albeit at the cost of potential overfitting issues. The conclusion of the paper is that the hybrid CNN LSTM approach is a very efficient combination of two exceptionally powerful deep learning algorithms, while the only obstruction being the tuning of parameters.

Keywords: machine learning, deep learning, intrusion detection system, network security

PRILOZI

Prilog 3.1. GitHub repozitorij kodova sustava za izradu baze podataka :

https://github.com/benjaminvarvodic/Diplomski_Varvodi-_Benjamin_2024.git

ŽIVOTOPIS

Benjamin Varvodić rođen je u Vinkovcima 29.04.2000. gdje je pohađao i završio Osnovnu školu Antuna Gustava Matoša i matematički smjer Gimnazije Matije Antuna Reljkovića. 2019. godine upisuje preddiplomski studij elektrotehnike na Fakultetu elektrotehnike, računarstva i informacijskih tehnologija (FERiT) Osijek. 2022. godine u roku završava preddiplomski studij i iste godine upisuje diplomski studij automobilskog računarstva i komunikacija. Na zadnjoj godini diplomskog studija postaje stipendist tvrtke *TTTech Auto*.

_____

Potpis autora