SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 186

SMJEŠTANJE I PODEŠAVANJE VATROZIDA U MREŽI NA TEMELJU POLITIKA VISOKE RAZINE APSTRAKCIJE

Hrvoje Kristić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

DIPLOMSKI RAD br. 186

SMJEŠTANJE I PODEŠAVANJE VATROZIDA U MREŽI NA TEMELJU POLITIKA VISOKE RAZINE APSTRAKCIJE

Hrvoje Kristić

Zagreb, lipanj 2023.

SVEUČILIŠTE U ZAGREBU
FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

Zagreb, 10. ožujka 2023.

DIPLOMSKI ZADATAK br. 186

Pristupnik:	**Hrvoje Kristić (0036516452)**
Studij:	Računarstvo
Profil:	Računarska znanost
Mentor:	izv. prof. dr. sc. Stjepan Groš

Zadatak: **Smještanje i podešavanje vatrozida u mreži na temelju politika visoke razine apstrakcije**

Opis zadatka:

Vatrozid je uređaj koji provodi filtriranje mrežnog prometa u skladu s postavljenom politikom. Kod postojećih vatrozida se ova politika u pravilu implementira kroz skup pravila koja definiraju način postupanja s mrežnim prometom na temelju njegovih tehničkih karakteristika. Posljednjih godina se razvijaju pristupi koji omogućuju zadavanje politike vatrozida na visokoj razini apstrakcije, primjerice uporabom domenskih jezika ili opisa temeljenih na grafovima. Potom se te politiku visoke razine apstrakcije prevode u skup pravila, optimiziraju i raspoređuju na mrežu vatrozid uređaja. U sklopu diplomskoga rada potrebno je razviti programsko rješenje koje kao ulaz prima mrežnu politiku opisanu uporabom modela temeljenog na grafovima te opis topologije ciljane računalne mreže. Na temelju ulaza programsko rješenje traži optimalne pozicije za vatrozide u mreži, a potom generira konfiguracije svakog od vatrozida kako bi se ispoštovala politika visoke razine dana na ulazu. Programsko rješenje mora biti modularno i robustno. Radu priložiti izvorni kôd. Citirati korištenu literaturu i navesti dobivenu pomoć.

Rok za predaju rada: 23. lipnja 2023.

Sadržaj

Uvod

U današnje vrijeme svjedočimo sve većem broju kibernetičkih napada, nerijetko u vojne svrhe, čemu u prilog ide velik broj kibernetičkih napada u Rusko-Ukrajinskom ratu. Prema izvještaju, u Ukrajini je zabilježen desetostruki porast kibernetičkih napada [1]. Jedna od komponenti zaštite računalnih sustava je zaštita računalnih mreža. Računalne mreže sastoje se od niza različitih uređaja kao što su: usmjernici, preklopnici, poslužitelji, korisnička računala i drugi. Prvi korak zaštite takvih uređaja i cijelih mreža je korištenje vatrozida. Vatrozidi su mrežni uređaji koji kontroliraju mrežni promet te razdvajaju mreže različitih sigurnosnih zahtjeva [2]. Mreže s velikim brojem uređaja trebale bi sadržavati veći broj vatrozida [3] s obzirom na to da različiti segmenti takve mreže neće imati jednake sigurnosne zahtjeve.

Smještaj većeg broja vatrozida u mreži uglavnom se obavlja ručno, no taj posao može biti zahtjevan te često može rezultirati pogreškama s obzirom na to da pojedine organizacije mogu imati stotine vatrozida [3]. Također jednom kad su vatrozidi uspješno smješteni u mreži, potrebno je obaviti njihovo podešavanje, taj postupak isto tako može rezultirati pogreškama ako mu se ne posveti dovoljno pažnje. Iz navedenih razloga jasno je da postoji potreba za aplikacijom koja bi obavljala navedene poslove.

Cilj ovog diplomskog rada je riješiti problem ručnog smještanja i podešavanja vatrozida u mreži. U tu svrhu razvijena je aplikacija koja pokušava riješiti te probleme. Razvijena aplikacija sastoji se od dva dijela.

Prvi dio zadužen je za određivanje rasporeda vatrozida u mreži te se takav sustav temelji na sličnosti prometa između različitih uređaja, a inspiraciju uzima iz [3]. Izlaz ovog dijela prosljeđuje se u drugi dio aplikacije kako bi se moglo obaviti podešavanje svakog vatrozida.

Drugi dio aplikacije zadužen je za podešavanje svakog vatrozida. Ovaj dio aplikacije prvo određuje koja sigurnosna pravila će se postaviti na pojedini vatrozid. Nakon toga pravila na visokoj razini apstrakcije prevode se u iptables format.

Ovaj diplomski rad strukturiran je na sljedeći način. Prvo poglavlje, Smještanje vatrozida u mreži**,** detaljno opisuje algoritam za raspoređivanje vatrozida u mreži te neka njegova

ograničenja. Drugo poglavlje, Podešavanje vatrozida u mreži**,** opisuje postupak raspoređivanja iptables pravila na vatrozide i prevođenje pravila na visokoj razini apstrakcije u iptables pravila. Treće poglavlje, Alternativni algoritam, nedostatci i poboljšanja**,** daje uvid u alternativni algoritam raspoređivanja vatrozida koji ima manju vremensku složenost od algoritma opisanog u prvom poglavlju. Također opisuje kako se implementirana aplikacija može poboljšati. Četvrto poglavlje, Testiranje i demonstracija aplikacije**,** kao što mu i naziv govori, odnosi se na opis provedbe testiranja aplikacije te se po koracima prikazuje izvođenje algoritma. Zadnje poglavlje je Zaključak**,** u tom poglavlju sažeti su najbitniji dijelovi ostalih poglavlja, te se opisuje mogući daljnji rad na aplikaciji.

1. Smještanje vatrozida u mreži

Kao što je već rečeno u uvodnom poglavlju, potreba za automatiziranim smještanjem vatrozida u mreži proizlazi iz činjenica da ljudi rade greške te da je sam proces smještanja vatrozida zahtjevan proces. Naime ljudi su podložni različitim čimbenicima kao što su stres, motivacija, umor, okolina i drugi. Svi ti čimbenici mogu imati utjecaj na posao koji obavljaju pa na taj način čovjek može neispravno smjestiti neki od vatrozida u mrežu ili smjestiti manji broj vatrozida nego što je potrebno te time može narušiti sigurnost cijele mreže. Također u većim organizacijama obično postoji značajan broj računala, što rezultira i povećanim brojem vatrozida koji su potrebni za zaštitu tih računalnih sustava. Porastom broja vatrozida raste i količina posla koju inženjeri moraju obaviti.

Algoritam implementiran u sklopu ovog diplomskog rada pokušava riješiti navedene probleme pri čemu je inspiracija proizašla iz rada [3]. Rad [3] temelji se na logici i formalnoj verifikaciji. Ukratko, rad [3] koristi Dijkstra *guarded commands* [12] i *Product family algebra* [13] kako bi definirao skupove uređaja koji će se nalaziti pod nadzorom određenog vatrozida. Ti skupovi čine temelj cijelog rada te je većina rada posvećena definiranju i objašnjavanju tog koncepta. Isto tako u navedenom radu puno pažnje je posvećeno dokazivanju formalne ispravnosti definiranja takvih skupova kao i cijelog algoritma.

No važno je reči da ti dijelovi nisu obuhvaćeni u ovom diplomskom radu, već će se rad usmjeriti na samu implementaciju algoritma. Također važno je spomenuti kako implementirani algoritam ne osigurava formalnu ispravnost s obzirom na to da su učinjene neke promjene u odnosu na rad [3]. Te promjene bit će opisane kasnije u ovom diplomskom radu.

1.1. Razvojna okolina

Aplikacija za automatizirani smještaj vatrozida, pisana je koristeći programski jezik Python 3 [5], osim Pythona, aplikacija koristi i TreeLib biblioteku [4]. *TreeLib* je biblioteka otvorenog koda koja pruža mogućnost rada sa strukturom podataka stabla. Struktura stabla je potrebna kako bi se vatrozidi mogli hijerarhijski poredati. Drugim

riječima, potrebno je koristiti strukturu podataka gdje određeni objekt može imati druge objekte kao svoju djecu. Na taj način svaki vatrozid će biti zadužen za zaštitu svih uređaja koji su njegovi potomci.

Osim samog razvoja aplikacije, važna komponenta ovog rada je i testiranje razvijene aplikacije te je potrebno opisati alate korištene pri testiranju te aplikacije. Za testnu okolinu u sklopu ovog diplomskog rada odabran je alat IMUNES [10]. *IMUNES* (Integrated Multiprotocol Network Emulator/Simulator) je alat otvorenog koda koji se koristi za emulaciju i simulaciju računalnih mreža razvijen na Fakultetu elektrotehnike i računarstva u Zagrebu. IMUNES je namijenjen za testiranje mrežnih protokola i konfiguracija te omogućuje korisnicima stvaranje i međusobno povezivanje virtualnih mrežnih čvorova i simuliranje različitih scenarija mrežnog ponašanja. Također podešavanje virtualne mreže moguće je obaviti korištenjem grafičkog sučelja. Sve navedene karakteristike čine ovaj alat idealnim za korištenje u svrhu testiranja razvijene aplikacije.

Nmap [11] ("Network Mapper") je alat otvorenog koda korišten za istraživanje i provjeru sigurnosti mreža. Iako je Nmap najčešće korišten za provjeru sigurnosti mreža, moguće ga je koristiti u raznim situacijama. Tako je Nmap u sklopu ovog diplomskog rada korišten za testiranje ispravnosti konfiguracije iptables pravila.

Osim korištenih alata, važno je spomenuti i podatke s kojima aplikacija radi, odnosno format ulaznih datoteka. Prije pokretanja aplikacije potrebno je urediti dvije datoteke:

- Apps - CSV datoteku koja se koristi kao baza znanja s podacima o često korištenim aplikacijama i protokolima unutar računalnih mreža.
- Vatrozid_pravila - Ova datoteka je JSON formata, ona sadrži podatke o pravilima za konfiguraciju vatrozida na temelju kojih se vatrozidi raspoređuju u mreži.
- Name_alias – CSV datoteka koja se sastoji od dva polja. Prvo polje odnosi se na naziv računala, a drugo na IP adresu tog računala.

Apps.csv datoteka djelomično je popunjena s nekim poznatijim protokolima te se šalje s aplikacijom. Korisnici aplikacije mogu nadopuniti tu datoteku sa svojim proizvoljnim protokolima koji se koriste u mreži njihove organizacije. Datoteka se može nadopuniti dodavanjem novih redaka pridržavajući se CSV sintakse, što podrazumijeva razdvajanje vrijednosti zarezom. Polja koja datoteka prihvaća su: naziv protokola ili aplikacije, *portovi* na kojima ta aplikacija radi, transportni protokol i iptables stanje. Na primjer, ako korisnik želi dodati Putty SSH klijent u bazu znanja, to može učiniti dodavanjem retka:

putty-SSH,22,tcp,"RELATED,ESTABLISHED".

Vatrozid_pravila je JSON datoteka koja sadrži pravila za vatrozid na visokoj razini apstrakcije, korisniku se isporučuje prazna. Ta datoteka ispunjava se s pravilima za vatrozid specifičnim za svaku organizaciju. Na kodu (*Kod 1.1*) prikazan je primjer jednog pravila za vatrozid.

```
{
    "firewall_rules": {
        "firewall-rule-1": {
            "from_objects": [
                "engineer_junior_workstation>putty-SSH>25",
                "engineer_senior_workstation>RDP>3389",
                "CEO_workstation>firefox>default"
            ],
            "idn": "firewall-rule-1",
            "to_objects": [
                "web_proxy"
            ]
        }
    }
}
```

Kod 1.1 – Primjer pravila visoke razine apstrakcije

Kao što se može vidjeti, sva pravila postavljaju se unutar `firewall_rules` bloka. Svako pravilo ima tri parametra, a to su: `from_objects, idn, to_objects`. From_objects predstavlja skup izvorišnih uređaja, aplikacija ili protokola i *portova*, koje su odvojene znakom „>“. Ako je za *port* upisana oznaka ANY, onda se za *port* koristi vrijednost upisana u bazi znanja. To_objects predstavlja odredišni uređaj, a idn predstavlja identifikator pravila. Cilj ovakve strukture je olakšati korisniku pisanje pravila za vatrozid, umjesto kompliciranijih iptables pravila, korisnik će napisati nekoliko ovakvih pravila. Također baza znanja omogućuje pisanje pravila osobama s manje tehničkog znanja ili osobama koje nisu upoznate s iptables sintaksom. Isto tako važno je spomenuti da je ova datoteka vrlo slična datoteci korištenoj na predmetu Diplomski projekt, koja se koristila za rješavanje problema raspoređivanja vatrozid pravila na vatrozide. Datoteka vatrozid_pravila može se dobiti iz prethodno navedene datoteke dodavanjem porta u

from_objects te uklanjanjem port vrijednosti iz to_objects, sve ostalo ostaje nepromijenjeno.

Name_alias datoteka koristi se kako bi korisnik morao samo jednom upisati IP adresu za pojedino računalo te kasnije može koristiti naziv tog računala. Na taj način olakšava se podešavanje vatrozida te se smanjuje vjerojatnost pogreške jer je puno lakše koristiti intuitivno ime kao CEO_workstation nego neku IP adresu.

Nakon prethodnog pregleda ulaznih datoteka, rad će se usredotočiti na opis algoritma. Algoritam se sastoji od sljedeće 4 funkcije:

- Compute-GCD
- Build-network-graph
- Add-resources-to-net
- Optimize-network-graph

Svaka od te četiri funkcije bit će opisana u posebnom potpoglavlju.

1.2. Compute-GCD

Prije početka opisa algoritma važno je napomenuti da je algoritam Exp-Rns [3] korišten kao inspiracija za ovaj dio aplikacije.

Algoritam za smještanje vatrozida u mreži implementiran u sklopu ovog diplomskog rada temelji se na ideji grupiranja uređaja u mreži, na temelju izvorišnog računala i aplikacije ili protokola koji se koristi u komunikaciji između izvorišnog i odredišnog računala. Na temelju te ideje uvodi se pojam *Greatest Common Divisor (GCD)* [3]. *GCD* je u osnovi ništa drugo nego zajednički skup vatrozid pravila za određeni broj krajnjih računala. Na sljedećem kodu (*Kod 1.2*) prikazan je jedan primjer za određivanje GCD skupa.

```
{
    "firewall_rules": {
        "firewall-rule-1": {
            "from_objects": [
                "engineer_junior_workstation>putty-SSH>25",
                "engineer_senior_workstation>RDP>3389",
                "CEO_workstation>firefox>default"
            ],
            "idn": "firewall-rule-1",
```

```
                "to_objects": [
                    "web_proxy"
                ]
            },
            "firewall-rule-2": {
                "from_objects": [
                    "engineer_junior_workstation>putty-SSH>25",
                    "admin_workstation>RDP>3389"
                ],
                "idn": "firewall-rule-2",
                "to_objects": [
                    "email:proxy"
                ]
            }
        }
    }
```

Kod 1.2 - Primjer pravila za stvaranje GCD objekta

Za ovaj primjer GCD skup pravila sastojao bi se samo od pravila „`1.1.1.1>putty-SSH>25`“ zato što su u oba pravila izvorišni uređaj i *port* jednaki. Pravilo „`2.2.2.2>firefox>ANY`“ nije sadržano u „firewall-rule-2“ pa ne može biti uključeno u GCD, a pravilo „`1.1.1.2>RDP>3389`“ ima različiti izvorišni uređaj u odnosu na odgovarajuće pravilo u „firewall-rule-1“ pa ni ono ne može biti uključeno u GCD.

GCD je ključni pojam u ovom algoritmu, sve ostale funkcije se oslanjaju na taj skup te je izrazito važno shvatiti kako se konstruiraju GCD skupovi. U nastavku rada podrazumijeva se da je taj postupak jasan.

Do sada je opisano kako se konstruiraju GCD objekti, kako izgledaju i čemu služe ulazne datoteke, u nastavku će biti opisano kako se te komponente koriste u compute-GCD funkciji. Također prije samog opisa postupka, važno je objasniti što ova funkcija treba postići. Compute-GCD uzima dvije ulazne datoteke, iz njih određuje sva vatrozid pravila te skup svih uređaja i na kraju računa skup skupova međusobno sličnih uređaja na temelju vatrozid pravila. Dobiveni skupovi predstavljat će vatrozide, a ostale funkcije će na temelju tih skupova određivati koji vatrozid je zadužen za zaštitu pojedinog krajnjeg računala.

Postupak računanja skupova počinje s računanjem partitivnog skupa svih uređaja navedenih u vatrozid_pravila.json datoteci. Iz dobivenog partitivnog skupa potrebno je ukloniti prazan skup te one skupove koji sadrže samo jedan uređaj. Naime ti skupovi

nemaju druge uređaje s kojima mogu imati zajednička vatrozid pravila. Nakon ovog postupka ukupan broj skupova bit će jednak: $2^N - N - 1$, očito je da je složenost ovog algoritma jednaka eksponencijalnoj složenosti te je ta činjenica jedno od negativnih svojstava ovog algoritma.

Nakon određivanja partitivnog skupa i uklanjanja nepotrebnih skupova, potrebno je svakom od dobivenih skupova pridijeliti zajednički skup vatrozid pravila, odnosno potrebno je odrediti GCD skup kao u gornje opisanom postupku. U ovom trenutku potrebno je provjeriti postoje li skupovi uređaja koji imaju jednake skupove pravila. Sve takve skupove potrebno je ukloniti s obzirom na to da će svi skupovi s jednakim vatrozid pravilima predstavljati isti vatrozid. Postupak uklanjanja duplikata provodi se tako da se zadrže oni skupovi koji sadrže najveći broj uređaja, a ako postoji više takvih skupova, nasumično se odabire jedan. Cilj je odabrati najveći skup uređaja s obzirom na to da želimo minimizirati potreban broj vatrozida. Zadnji korak koji je potrebno napraviti u ovoj funkciji je uklanjanje GCD skupova koji sadrže vatrozid pravila koja su podskup nekog drugog GCD skupa. Drugim riječima, ako postoji GCD skup koji sadrži pravila A,B,C i postoji drugi GCD skup s pravilima A,B,C,D, onda je potrebno ukloniti prvi GCD skup.

Na kraju compute-GCD funkcije potrebno je stvoriti GCD objekte od dobivenih skupova. GCD objekt sadrži dva atributa, to su skup krajnjih računala i skup vatrozid pravila. Skup takvih GCD objekata se zatim šalje ostalim funkcijama na obradu.

1.3. Build-network-graph

Druga po redu funkcija u Exp-Rns algoritmu kao argument prima skup GCD objekata te je zadužena za konstruiranje podatkovne strukture stabla od dobivenih GCD objekata. Stablo je ostvareno koristeći već spomenutu biblioteku *TreeLib*. Izlaz prethodne funkcije je običan skup GCD objekata, među kojima ne postoji nikakav raspored. Rezultat funkcije build-network-graph bit će stablo gdje će korijenski čvor predstavljati vatrozid postavljen na rubu mreže, a svi ostali vatrozidi imat će čvor roditelj koji je također vatrozid.

Na početku stvaranja stabla, potrebno je odrediti korijenski čvor. GCD objekt koji sadrži sva krajnja računala u svome skupu postaje korijenski čvor. Također potrebno je voditi evidenciju čvorova koji su dodati u stablo, odnosno jednom kad je GCD objekt dodan u stablo, potrebno ga je ukloniti iz skupa svih GCD objekata. Nakon što je stvoren korijenski čvor, potrebno je odrediti skup onih GCD objekata koji imaju najveći broj vatrozid pravila.

Takav skup predaje se pomoćnoj funkciji – add-nodeset-to-g, te se cijeli postupak ponavlja dok se svi GCD objekti ne dodaju u stablo.

Funkcija add-nodeset-to-g zadužena je za dodavanje GCD objekata s jednakim brojem vatrozid pravila u stablo. Naravno, redoslijed kojim se GCD objekti dodaju nije proizvoljan, objekti se postupno dodaju, počevši s objektom koji ima najviše računala, a završavajući s objektom koji ima najmanje računala. Na taj način postiže se da vatrozidi koji su zaduženi za zaštitu većeg broja računala budu postavljeni bliže korijenu stabla. Takav zahtjev je razuman, vatrozidi koji se nalaze na manjoj dubini stabla filtrirat će općenitiji internetski promet, a oni vatrozidi koji se nalaze na većoj dubini bit će zaduženi za specifičniji promet.

Zadnja odluka koju funkcija mora donijeti prije nego li se GCD objekt doda u stablo jest, koji objekt će biti roditelj tom objektu, dvije su mogućnosti, korijenski čvor ili neki drugi čvor. Korijenski čvor bit će odabran kao roditelj, ako u stablu već ne postoji čvor koji sadrži vatrozid pravila koja su nadskup skupu pravila čvora koji trenutno želimo dodati. U takvom slučaju trenutni čvor postat će dijete takvog čvora. U suprotnom slučaju trenutni čvor postat će dijete korijenskog čvora.

Nakon što build-network-graph i add-nodeset-to-g funkcije završe s izvođenjem, stablo bi trebalo sadržavati sve GCD objekte hijerarhijski poredane. Na nultoj razini stabla nalazit će se vatrozid koji je zadužen za najopćenitija vatrozid pravila, a na najvišim razinama nalazit će se vatrozidi zaduženi za specifična pravila za pojedina krajnja računala.

1.4. Add-resources-to-net

Prethodno definirana funkcija bila je zadužena za dodavanje GCD objekata odnosno vatrozida u stablo, no nije bila zadužena za dodavanje krajnjih računala. Za taj posao zadužena je add-resources-to-net funkcija.

Jednako kao u add-nodeset-to-g funkciji, redoslijed umetanja čvorova u stablo nije proizvoljan, cilj je da krajnja računala završe na razinama veće dubine u stablu, a poslužitelji bi trebali završiti na manjim dubinama. To se postiže tako da se skup svih GCD objekata poreda silazno po broju vatrozid pravila koje taj GCD objekt sadrži.

Nakon toga stvara se skup svih računala R (krajnja računala i poslužitelji) te još jedan skup R2 koji je kopija tog skupa. Nakon toga slijedi iteriranje po prethodno poredanom skupu

objekata. Za svaki GCD objekt i za svako računalo u tom objektu, provjerava se je li to računalo već dodano u stablo tako da se provjeri nalazi li se to računalo u R2. Ako se ne nalazi, to znači da je računalo već dodano u stablo, inače dodaje se kao dijete onog GCD objekta koji se nalazi na najvećoj mogućoj razini stabla, a istovremeno se to računalo nalazi u njegovom skupu računala. Drugim riječima cilj je probati prvo dodati računalo na najveću moguću dubinu stabla, a ako ta pozicija nije odgovarajuća za to računalo, onda se odabire drugi GCD objekt na manjoj dubini. Takav postupak je potreban kako bi se osiguralo da računala korisnika završe na većoj dubini stabla od poslužiteljskih računala.

U ovom trenutku važno je primijetiti strukturu dobivenog stabla, naime čvorovi listovi su uvijek računala, a svi ostali čvorovi predstavljaju vatrozide.

1.5. Optimize-network-graph

Nakon što su sva računala i svi vatrozidi uspješno dodani u stablo na prvi pogled moglo bi se zaključiti kako je Exp-Rns algoritam uspješno završen, no postoji mogućnost da stablo nakon add-resources-to-net funkcije sadrži redundantne vatrozide. Pažljiviji čitatelj možda je uočio da prilikom dodavanja računala u stablo, nije opisano što će se dogoditi s vatrozidima koji nemaju čvorove djecu. S obzirom na to da više GCD objekata može u svom skupu računala sadržavati isto računalo, situacija gdje neki vatrozidi ostanu bez čvorova djece može se pojaviti vrlo često.

Iz navedenog razloga definira se optimize-network-graph funkcija. Navedena funkcija rješava tri različite situacije redundantnih vatrozida:

- Čvor list je vatrozid, čvorovi braća su računala
- Čvor list je vatrozid, čvorovi braća ne postoje
- Čvor list je vatrozid, čvorovi braća su vatrozidi

Prvu situaciju je lako riješiti, potrebno je samo provjeriti sve čvorove listove te ako su ti čvorovi vatrozidi, potrebno ih je ukloniti.

Druga i treća situacija nešto su kompliciranije, u drugoj situaciji također je potrebno ukloniti redundantni vatrozid, no s obzirom na to da je taj vatrozid jedino dijete svojeg roditeljskog čvora, potrebno je provjeriti je li i roditeljski čvor redundantan vatrozid. Taj postupak se ponavlja sve dok postoje vatrozidi koje je potrebno ukloniti.

Treća situacija vrlo je slična drugoj, jedina razlika je u tome da je potrebno ukloniti sve vatrozide na toj razini, pritom naravno pazeći da se ne ukloni čvor računalo ili čvor vatrozid koji nije redundantan.

Završetkom ove funkcije završava i cjelokupni Exp-Rns algoritam te se izlaz ovog algoritma predaje algoritmu za podešavanje vatrozida u mreži.

Za kraj važno je spomenuti da je automatizirano smještanje vatrozida u mrežu neistražena tema i ne postoji puno radova koji pokušavaju riješiti taj problem. Većina sličnih radova usmjerena je na definiranje zahtjeva za ostvarivanje sigurne računalne mreže te testiranje iste, no malo koji rad pokušava riješiti problem određivanja topologije takve mreže. Jedan od ciljeva ovog rada je pokušati potaknuti druge na istraživanje ove teme.

1.6. Izmjene u odnosu na izvorni rad

S obzirom na to da je velika inspiracija za algoritam smještanja vatrozida u mrežu preuzeta iz rada [3], potrebno je navesti izmjene koje su napravljene nad Exp-Rns algoritmom.

Prva od izmjena je pojednostavljivanje parametara koji se uspoređuju prilikom određivanja sličnosti između vatrozid pravila. Izvorni Exp-Rns algoritam određuje GCD objekte usporedbom: izvornog računala, *porta*, transportnog protokola i iptables stanja. Modificirani Exp-Rns algoritam razvijen u sklopu ovog diplomskog rada uspoređuje samo izvorna računala i *portove*. Ta izmjena je napravljena iz dva razloga:

1) Cilj cjelokupne aplikacije jest razviti algoritam koji koristi politike na visokoj razini apstrakcije, uključivanjem svih parametar gubi se smisao tog zahtjeva.
2) Većina protokola korištenih u bazi znanja automatski određuje koji će se transportni protokol koristiti te kakvo će biti iptables stanje.

Druga napravljena izmjena je da promet kojem je izvor ili odredište Internet, mora prolaziti kroz posrednički *web* ili neki drugi poslužitelj. Takav zahtjev je dodan zbog različite vrste vatrozid pravila u izvornom i ovom radu, ali više o tome će biti ispričano u 3. poglavlju Alternativni algoritam, nedostatci i poboljšanja.

Zadnja izmjena je nedostatak add-1F-resources funkcije. Ta funkcija je zadužena za dodavanje GCD objekata za one uređaje koji nemaju niti jedno vatrozid pravilo. S obzirom na zahtjeve za ulazne datoteke, ta funkcija nije potrebna ovom algoritmu i zato nije implementirana.

Opisom gore navedenih funkcija završeno je ovo poglavlje te je detaljno objašnjen algoritam kojim se vatrozidi smještaju u mrežu. Iduće poglavlje koristit će rezultate algoritma definiranog u ovom poglavlju kako bi se opisalo podešavanje vatrozida.

2. Podešavanje vatrozida u mreži

Prva komponenta aplikacije za smještanje i podešavanje vatrozida u mreži opisana je u prvom poglavlju te se ta komponenta temelji na Exp-Rns algoritmu. Rezultat implementiranog algoritma je stablo s čvorovima listovima koji predstavljaju računala, a svi ostali čvorovi su vatrozidi. To stablo predstavlja strukturu računalne mreže neke organizacije, ali vatrozidi koji se nalaze u tom stablu nisu podešeni kako bi mogli obavljati svoju osnovnu zadaću to jest filtrirati mrežni promet.

Ovo poglavlje nastoji opisati metode i postupke kojima se postiže automatizirano prevođenje pravila visoke razine apstrakcije u iptables pravila te podešavanje odgovarajućih vatrozida tim pravilima. Za ostvarivanje navedenih funkcionalnosti potrebno je riješiti dva problema:

1) Određivanje vatrozida koji mora biti podešen s pojedinim pravilom.
2) Prevođenje pravila na visokoj razini apstrakcije u iptables pravila.

Aplikacija za automatizirano podešavanje vatrozida u mreži, također je pisana koristeći programski jezik Python 3 i isto tako koristi *TreeLib* biblioteku. Isto tako koriste se apps.csv i vatrozid_pravila.json ulazne datoteke.

2.1. Pravila za vatrozid

Vatrozid pravila na visokoj razini apstrakcije opisana su u prošlom poglavlju, ovdje će ukratko biti opisan iptables konfiguracijski alat s obzirom na to da će rezultat ovog dijela aplikacije biti velik broj iptables pravila.

Iptables je alat za podešavanje vatrozida na Linux sustavima [6]. *Iptables* alat sastoji se od nekoliko komponenti: *tables*, *chains*, *rules* i *targets* [7].

- Tablica (*Tables*) – Komponente koje povezuju slične radnje, svaka tablica sastoji se od nekoliko *chainova*. Tablice mogu biti: filter, nat, mangle i raw.
- Lanac (*Chains*) – Lanac je niz iptables pravila. Lanci mogu biti: *input*, *output*, *forward*, *prerouting* i *postrouting*.

- Pravila (*Rules*) – Iptables pravila su konfiguracijske upute koje se koriste za filtriranje, preradu i upravljanje mrežnim paketima
- *Targets* - Naredbe koje definiraju vatrozidu što treba napraviti s pojedinim paketom. Akcije mogu biti: *accept*, *drop*, *reject* i *return*.

Prije definiranja postupaka koji su potrebni za ostvarivanje ovog dijela algoritma, važno je napomenuti podržane vrste iptables pravila koja aplikacija generira.

Aplikacija na svom izlazu generira iptables pravila koja se nalaze u *filter* tablici. Filter tablica je jedina tablica koja se koristi zato što je ona jedina zadužena za filtriranje prometa, a to je glavna svrha vatrozida pa tako i ovog rada. Ostale tablice nisu podržane u ovoj aplikaciji.

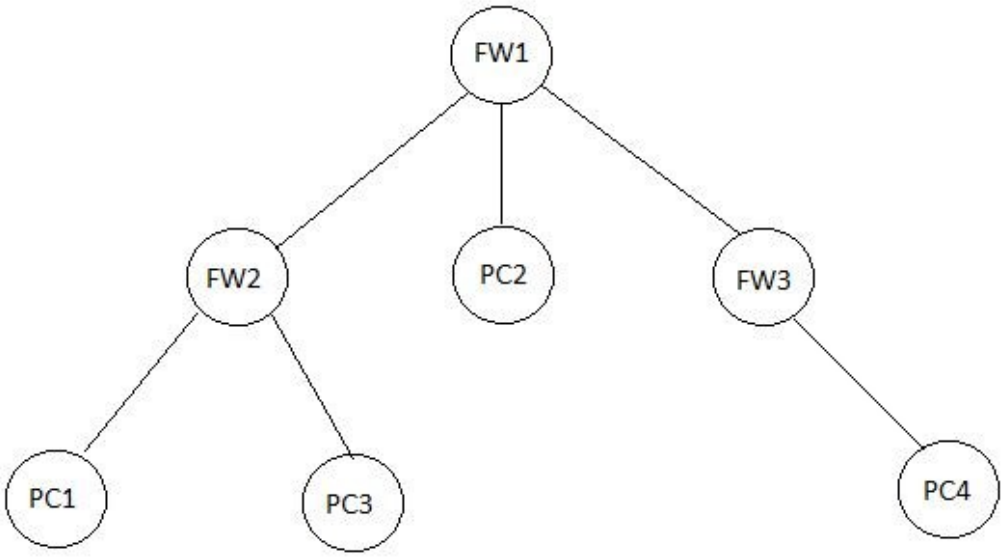
Isto tako aplikacije je ograničena na generiranje pravila koja su sadržana u *input*, *output* i *forward* lancima, slijedi kratak opis svakog od tih lanaca.

- *Input* – Koristi se za obradu mrežnih paketa koji su namijenjeni računalu na kojem je definirano pravilo s *input* lancem.
- *Output* – Koristi se za obradu mrežnih paketa koji se šalju s računala na kojem je definirano pravilo s *output* lancem.
- *Forward* – Koristi se za prosljeđivanje mrežnih paketa drugim uređajima. Najkorišteniji lanac prilikom podešavanja vatrozida.

Također sva iptables pravila koja ova aplikacija generira, imat će *accept* akcijske (target) vrijednosti. Takvo ponašanje je odabrano zato što se u izlaznoj konfiguracijskoj datoteci za vatrozide sav mrežni promet na početku odbacuje, a tek onda se omogućuje onaj mrežni promet koji je dozvoljen u napisanim pravilima. Druga mogućnost je na početku omogućiti sav mrežni promet te pravila za vatrozid koristiti za odbacivanje mrežnog prometa definiranog u tim pravilima. Druga mogućnost zahtijevala bi pisanje mnogo većeg broja pravila te iz tog razloga aplikacija generira samo *allow* pravila. Treća mogućnost bila bi omogućiti promet s pojedinih podmreža te onda onemogućiti promet samo sa specifičnih uređaja unutar tih mreža, no takvo rješenje bi uvodilo nepotrebne komplikacije te nije razmatrano za implementaciju.

2.2. Algoritam za podešavanje vatrozida

Cilj ovog potpoglavlja je detaljno objasniti kako algoritam za podešavanje vatrozida radi. Kao što je već rečeno, algoritam se sastoji od dvije komponente. Prva od njih je zadužena za pronalazak svih vatrozida na koje je potrebno dodati određeno pravilo. Konceptualno je jednostavno odrediti te vatrozide, to su svi vatrozidi koji se nalaze na najkraćoj putanji između izvorišnog i odredišnog računala. Takva metoda koja pronalazi putanju s najmanjim brojem vatrozida naziva se min konfiguracija [8]. S obzirom na to da ovaj algoritam koristi strukturu stabla za prikaz mrežne topologije, potrebno je osmisliti algoritam kojim će se odrediti min konfiguracija. Slika (Slika 2.1) prikazuje primjer mrežne topologije nad kojom će biti opisan navedeni postupak.



Slika 2.1 – Primjer mrežne topologije

Na slici svi čvorovi čije ime započinje s FW predstavljaju vatrozid, a čvorovi čije ime započinje s PC predstavljaju računala. Također važno je napomenuti da je ovo pojednostavljena mrežna topologija, u stvarnoj implementaciji te mreže PC1 i PC3 bili povezani na preklopnik koji bi bio dijete čvora FW2.

Za pronalazak vatrozida koji se nalaze između čvorova PC1 i PC4, algoritam prvo određuje potpune putanje do izvorišnog i odredišnog čvora. U ovom slučaju putanja za

izvorišni čvor bi bila: FW1,FW2,PC1, a za odredišni FW1,FW3,PC4. Naravno kako krajnja računala nisu vatrozidi, ona se uklanjaju iz tih putanja. Zatim je potrebno odrediti koja je putanja duža te po njoj krenuti iterirati u obrnutom redoslijedu, ako su putanje jednake duljine, može se odabrati bilo koja od te dvije putanje. U ovom slučaju odabire se putanja: FW1,FW2. Za svaki vatrozid V na toj putanji, provjerava se nalazi li se taj vatrozid u onoj drugoj putanji. Prvo se provjerava FW2, a zatim FW1. Ako je pronađen zajednički vatrozid, postupak se prekida te se konstruira skup vatrozida koje je potrebno konfigurirati zadanim pravilom. Ako zajednički vatrozid nije pronađen, znači da izvorišno ili odredišno računalo ne postoji u toj mreži. U ovom primjeru algoritam će pronaći FW1 kao zajednički vatrozid. Nakon što je zajednički vatrozid pronađen, potrebno je odabrati sve vatrozide koji se nalaze nakon njega u izvorišnoj i odredišnoj putanji. Sljedeći taj postupak, dobiva se: FW2, FW1, FW3 skup vatrozida.

Navedeni postupak koristi se u većini pravila, no postoje dva slučaja kad se taj postupak mora promijeniti.

- Prvi slučaj se javlja kad je izvorišno ili odredišno računalo uređaj s Interneta, u tom slučaju postoji samo jedna putanja te je potrebno podesiti sve vatrozide na toj putanji.
- Drugi slučaj odnosi se na računala koja se nalaze u istoj podmreži, kao što su PC1 i PC3. U toj situaciji mrežni promet uopće ne prolazi kroz vatrozid te nema razloga da takvo pravilo bude dodano u skup pravila nekog vatrozida.

Nakon što je pronađen skup vatrozida između izvorišnog i odredišnog uređaja, potrebno je na te vatrozide postaviti iptables pravila.

Postupak prevođenja pravila na visokoj razini apstrakcije u iptables pravila te podešavanje vatrozida tim pravilima vrlo je jednostavan postupak. Proces prevođenja pravila započinje čitanjem odgovarajućeg pravila na visokoj razini apstrakcije iz vatrozid_pravila.json datoteke. Nakon čitanja te datoteke, informacije koje su potrebne za stvaranje iptables pravila, a koje nedostaju pronalaze se u bazi znanja koja se nalazi u apps.csv datoteci. Nakon što se prikupe sve potrebne informacije, iptables pravilo konstruira se prema predlošku: iptables -A {chain} -p {transport} {src} {dst} {dst_port} {state} -j ACCEPT.

Prilikom konstruiranja svakog pravila, vodi se evidencija vatrozida na koje to pravilo mora biti postavljeno te nakon što se sva pravila konstruiraju, pokreće se skripta koja će za svaki vatrozid napisati njegovu konfiguracijsku datoteku. Primjer jedne takve datoteke prikazan

je u nastavku. Svrha tog primjera je prikazati strukturu datoteke koju aplikacija generira, nisu bitna sama pravila niti način kako su ta pravila nastala.

```
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

iptables -F INPUT
iptables -F OUTPUT
iptables -F FORWARD

iptables -A INPUT  -i lo -m state --state NEW  -j ACCEPT
iptables -A OUTPUT -o lo -m state --state NEW  -j ACCEPT

iptables -A FORWARD -s 10.0.0.21 -o eth0 -m state --state
NEW,RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 -d 198.51.100.10 --dport
22 -m state --state RELATED,ESTABLISHED -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 -d 198.51.100.11 --dport
80 -m state --state NEW -j ACCEPT
iptables -A FORWARD -p udp -i eth0 -d 198.51.100.11 --dport
80 -m state --state NEW -j ACCEPT
iptables -A FORWARD -p tcp -i eth0 -d 198.51.100.11 --dport
443 -m state --state NEW -j ACCEPT
iptables -A FORWARD -p udp -i eth0 -d 198.51.100.11 --dport
443 -m state --state NEW -j ACCEPT
iptables -A FORWARD -p tcp -s 198.51.100.11 -o eth0 --dport
54 -m state --state NEW -j ACCEPT
iptables -A FORWARD -p udp -s 198.51.100.11 -o eth0 --dport
54 -m state --state NEW -j ACCEPT
```

Kao što se može primijetiti, skripta će na početku dodati tri pravila koja nisu navedena od strane korisnika. To su pravila:

- `iptables -P input drop`
- `iptables -P output drop`
- `iptables -P forward drop`

Svrha tih pravila je odbacivanje svog mrežnog prometa koji nije eksplicitno definiran u vatrozid_pravila.json datoteci.

Nakon toga u datoteci se nalaze dodatna tri pravila:

- `iptables -F input`
- `iptables -F output`
- `iptables -F forward`

Ta pravila odbacuju sva prethodno definirana iptables pravila za svaki od navedena tri lanca. Naravno potrebno je napomenuti da ova pravila neće imati utjecaj na prethodna tri pravila.

Iduća vrsta iptables pravila su pravila koja omogućavaju dolazni i odlazni mrežni promet s *loopback* sučelja, na navedenoj slici to su pravila:

- `iptables -A INPUT  -i lo -m state --state NEW  -j ACCEPT`
- `iptables -A OUTPUT -o lo -m state --state NEW  -j ACCEPT`

Prethodno definirana iptables pravila bit će jednaka za sve vatrozide, neovisno u kojoj računalnoj mreži se nalaze. Aplikacija nikako ne utječe na ta pravila, već ih aplikacija samo zapisuju u svaku datoteku. Pravila koja su generirana aplikacijom prikazana su nakon *loopback* pravila.

3. Alternativni algoritam, nedostatci i poboljšanja

Završetkom poglavlja podešavanje vatrozida u mreži, opisana je cijela aplikacija razvijena u sklopu ovog diplomskog rada, te se u ovom, ali i ostalim poglavljima pažnja skreće na ostale aktivnosti koje je bilo potrebno provesti za završetak ovog rada.

Ovo poglavlje opisuje Rns algoritam [3], koji je poboljšanje Exp-Rns algoritma te opisuje neke nedostatke razvijene aplikacije kao i neka moguća poboljšanja. Za početak bit će opisani nedostatci i poboljšanja aplikacije, a na kraju poglavlja bit će detaljno opisan prethodno navedeni algoritam.

3.1. Nedostatci aplikacije

Prvi nedostatak odnosi se na problem loše definiranih pravila za vatrozid ili definiranje premalog broja pravila za vatrozid. Naime ako skup definiranih pravila u vatrozid_pravila.json datoteci nije optimalan, onda je sigurnost takve mreže ugrožena iz dva razloga. Prvu ugrozu predstavlja sam skup loše definiranih vatrozid pravila, a drugi problem se javlja jer takva pravila generiraju neoptimalnu mrežnu topologiju. Na primjer ako bi u nekoj mreži postojala četiri uređaja te ako bi svaki sa svakim komunicirao koristeći različiti protokol, takva mreža sastojala bi se samo od jednog vatrozida. Razlog za takvo ponašanje aplikacije leži u tome da Exp-Rns algoritam smješta vatrozide u mrežu na temelju sličnosti mrežnog prometa između računala. U prethodno navedenom primjeru postojala bi četiri računala i dvanaest protokola te ne bi postojala sličnost mrežnog prometa koja bi opravdavala dodavanje još jednog vatrozida u mrežnu topologiju. Slično bi se dogodilo kad bi osoba koja definira pravila za vatrozid, definirala premali broj pravila. Drugačije rečeno, ako pravila na visokoj razini apstrakcije nisu ispravno definirana, aplikacija će odrediti neoptimalnu mrežnu topologiju te će generirati neoptimalna iptables pravila. U takvoj situaciji ručno smještanje vatrozida bi prouzročilo manje sigurnosne rizike, iptables pravila bi u oba slučaja bila neispravna, ali bi barem mrežna topologija bila ispravna, naravno uz pretpostavku da osoba koja određuje tu topologiju neće napraviti pogrešku.

Idući nedostatak ne predstavlja sigurnosni problem, ali zahtjeva interakciju korisnika nakon stvaranja datoteke sa svim iptables pravilima za pojedini vatrozid. Problem je to da ova aplikacija ne može generirati iptables pravila koja omogućuju mrežni promet između računala i vatrozida. Takva pravila dodavala bi se u *input* i *output* iptables lance s obzirom da je u takvim pravilima vatrozid izvorišni ili odredišni uređaj. No takva pravila je nemoguće generirati ako u trenutku pisanja pravila na visokoj razini apstrakcije korisnik ne zna broj ni položaj vatrozida koji će se nalaziti u mreži. Primjer takvog pravila bio bi kad bi se neko računalo u mreži trebalo povezati s nekim vatrozidom koristeći SSH protokol. Moguće rješenje tog problema bilo bi podržavanje pravila na visokoj razini apstrakcije koje bi omogućilo komunikaciju između odabranog računala i svih vatrozida, no takvo rješenje bi uvodilo potencijalne sigurnosne rizike i stoga nije implementirano u ovoj aplikaciji.

Još jedan nedostatak aplikacije jest nepostojanje mogućnosti kojom bi se adresirale cijele podmreže u vatrozid pravilima. Ta mogućnost nije dodana u aplikaciju kako bi se zadržalo korištenje imena računala kao alternativa IP adresama. Takvom odlukom žrtvovano je lakše pisanje pravila na visokoj razini apstrakcije u zamjenu za čitljiviji ispis mrežne topologije.

3.2. Moguća poboljšanja aplikacije

Prethodno opisani problemi aplikacije također su mogli biti smješteni u ovo potpoglavlje, no ti nedostatci su teže rješivi ili proizlaze iz dizajnerskih odluka prilikom stvaranja aplikacije. S druge strane problemi aplikacije definirani u ovom potpoglavlju relativno lagano se mogu riješiti, ali u sklopu ovog diplomskog rada nisu implementirani zbog nekih tehničkih ograničenja, kao što su poteškoće s testiranjem aplikacije ili jer implementacija alternativnih rješenja ne bi nužno dala bolje rezultate, na primjer implementiranje različitih sličnosti za Exp-Rns algoritam.

Na početku treba spomenuti nftables [9] alat kao alternativu iptables alatu. *Nftables* je isto kao i iptables alat za podešavanje vatrozida na Linux sustavima, no nftables je moderniji alat te je izravni nasljednik iptablesa. Nftables je vrlo lako implementirati u postojeću aplikaciju, potrebno je dodati nftables predložak kao što je već prikazan iptables predložak u poglavlju – algoritam za podešavanje vatrozida. Jedini razlog zbog kojeg nftables nije implementiran je nemogućnost testiranja takvih pravila. Naime Imunes [10] okolina

korištena za testiranje aplikacije ne podržava nftables pravila. Ukoliko bi postojala okolina za testiranje nftables pravila, generiranje takvih pravila bilo bi skoro jednako kao i generiranje iptables pravila.

Druga funkcionalnost koju je moguće unaprijediti jest proširenje podržanih iptables lanaca. Kako je već spomenuto, samo su *input*, *output* i *forward* lanci podržani, a *prerouting* i *postrouting* nisu. Ta dva lanca koriste se za NAT preslikavanje adresa, koristeći te lance moguće je sakriti IP adrese računala u mreži. Također kad bi se implementirala pravila za navedene lance, aplikacija bi se daljnje mogla unaprijediti dodavanjem mogućnosti generiranja ostalih iptables tablica, u trenutačnoj verziji aplikacije implementirana su samo pravila koja pripadaju *filter* tablici.

Na kraju potrebno je spomenuti poboljšanje koje bi najviše unaprijedilo aplikaciju, a to je razvoj različitih metoda za određivanje sličnosti između računala i mrežnog prometa. Algoritam Exp-Rns definirao je metodu usporedbe koja uspoređuje: izvorišni uređaj, korištene *portove*, transportni protokol i iptables stanje, no to nije jedina moguća usporedba koja se može koristiti. Aplikacija bi se mogla proširiti različitim metodama za određivanje sličnosti između uređaja na mreži, moguće bi čak bilo definirati nove ulazne datoteke koje bi sadržavale dodatne informacije o uređajima kako bi se mogao dati kontekst u kojem će se uređaj koristiti. Na primjer to bi mogla biti datoteka koja bi definirala da će se uređaj A koristiti kao web posrednički poslužitelj, uređaj B kao računalo inženjera seniora i uređaj C kao računalo inženjera juniora. Takva dodatna datoteka bi onda pomogla aplikaciji da u jednu podmrežu postavi računala B i C, a računalo A u drugu mrežu te bi se između te dvije mreže nalazio vatrozid.

3.3. Alternativni RNS algoritam

U poglavlju smještanje vatrozida u mreži bilo je opisano da je jedan od glavnih problema Exp-Rns algoritma njegova složenost. Složenost compute-GCD funkcije jednaka je eksponencijalnoj složenosti te to svojstvo može predstavljati velik problem prilikom korištenja aplikacije nad mrežom s nešto većim brojem uređaja. Na primjer već za trideset računala na mreži, compute-GCD funkcija bi u jednom trenutku morala sadržavati nešto više od milijardu skupova. Takvo rješenje nije skalabilno te bi se moralo ponuditi bolje rješenje.

Rns algoritam [3] nudi znatno bolje performanse od običnog Exp-Rns algoritma. Za razliku od Exp-Rns algoritma, Rns algoritam ima svojstvo polinomijalne složenosti te ga to čini mnogo primjenjivijim za korištenje unutar organizacija koje sadrže računalne mreže s velikim brojem uređaja.

Jedina razlika Rns i Exp-Rns algoritma javlja se u compute-GCD funkciji, to je i očekivano s obzirom na to da je ta funkcija razlog eksponencijalne složenosti Exp-Rns algoritma. Cilj poboljšanog algoritma je stvoriti GCD skupove na manje zahtjevan način, složenost Exp-Rns algoritma proizlazi iz činjenice da je potrebno stvoriti 2^N skupova, gdje je N broj uređaja u mreži.

Rns algoritam stvara GCD skupove tako da se na početku odrede sva različita pravila za vatrozid te se svakom pravilu dodjeljuju svi uređaji za koje je to pravilo namijenjeno. Na taj način broj skupova koji se konstruiraju ovisit će o broju različitih pravila umjesto o broju uređaja. Važnija činjenica je da se više ne stvara partitivni skup, nego se stvara broj skupova koji je proporcionalan s brojem pravila zadanih na ulazu algoritma. Tom promjenom riješen je najveći problem Exp-Rns algoritma.

Nakon što su skupovi stvoreni, potrebno je napraviti još par izmjena kako bi rezultat novonastale funkcije bio istovjetan rezultatu compute-GCD funkcije iz originalnog algoritma.

Važno je primijetiti da nastali skupovi mogu sadržavati jedno ili više računala. Jednako kao i u compute-GCD funkciji, skupovi sa samo jednim elementom ne mogu biti razlog za dodavanje novog GCD objekta odnosno novog vatrozida. Takve element je potrebno ukloniti. Također potrebno je ukloniti sve duplikate skupova jer bi oni predstavljali isti vatrozid.

Ostale funkcije koje su definirane u Exp-Rns algoritmu nije potrebno mijenjati s obzirom na to da je rezultat novonastale funkcije jednak rezultatu compute-GCD funkcije.

4. Testiranje i demonstracija aplikacije

U ovom poglavlju bit će opisano kako je provedeno testiranje aplikacije napravljene u sklopu ovog diplomskog rada. Nakon opisa testiranja aplikacije također će biti prikazano izvođenje aplikacije u ključnim trenutcima kako bi se još jasnije prikazalo sve ono što je već objašnjeno u prethodnim poglavljima.

4.1. Testiranje aplikacije

Prilikom testiranja prve komponente aplikacije to jest algoritma za smještanje vatrozida u mrežu željeno stanje je bilo definirano s nekoliko uvjeta. Ako su svi uvjeti bili ispunjeni, u tom slučaju je algoritam ocijenjen ispravnim za taj ulazni skup pravila.

Uvjeti koje algoritam mora zadovoljiti su:

1) Ako postoje pravila gdje je izvorišno ili odredišno računalo Internet, tada mrežna topologija mora uključivati DMZ.
2) U generiranoj mrežnoj topologiji ne smiju postojati redundantni vatrozidi.
3) Računala čija pravila su slična, trebala bi imati isti roditeljski čvor kao roditelja. Drugim riječima, isti vatrozid bi trebao biti zadužen za zaštitu računala koja imaju mnogo zajedničkih pravila.

Tijekom testiranja aplikacije, svi navedeni uvjeti bili su zadovoljeni te je na temelju toga procijenjeno da komponenta aplikacije zadužena za smještaj vatrozida u mrežu radi ispravno.

Ostatak aplikacije odnosno komponenta zadužena za podešavanje vatrozida, većim djelom testirana je u IMUNES-u te koristeći Nmap. Za testiranje se koristi Nmap, a IMUNES je potreban kako bi se odabrana mrežna topologija postavila i emulirala. Kao i kod prethodne komponente i u ovom slučaju je bilo potrebno zadovoljiti nekoliko uvjeta kako bi se testiranje ocijenilo kao uspješno. Uvjeti koji su bili promatrani su:

1) Algoritam mora pronaći ispravan skup vatrozida koje treba podesiti određenim pravilom.

2) Konfiguracijska datoteka ne smije imati pogreške koje bi onemogućavale podešavanje vatrozida.
3) Sav mrežni promet koji je omogućen pravilima na visokoj razini apstrakcije mora biti propušten na svim vatrozidima koji se nalaze između izvorišnog i odredišnog računala.
4) Sav mrežni promet koji nije omogućen pravilima na visokoj razini apstrakcije ne smije biti propušten niti na jednom vatrozidu koji se nalazi između izvorišnog i odredišnog računala.

Prvi po redu uvjet je vrlo lako provjeriti usporedbom dobivenog skupa vatrozida i skupa vatrozida koji se nalaze između dva željena računala. Ovo je lako provjeriti promatranjem dobivene mrežne topologije i dobivenog skupa vatrozida.

Drugi uvjet je također lagano provjeriti. Naime ako se ne javljaju poruke o pogreškama prilikom postavljanja konfiguracije na vatrozide, onda je i ovaj uvjet zadovoljen.

Ispunjavanje četvrtog uvjeta može se provjeriti tek djelomično zato što nije moguće ispitati hoće li baš svaki paket čiji mrežni promet nije omogućen biti odbačen. Razlog za to je očit, dovoljno je u nekom pravilu promijeniti korišteni *port* i time dobivamo novo pravilo koje je potrebno testirati. Taj postupak može se ponavljati mnogo puta te bi takvo testiranje bilo besmisleno zbog vremena koje bi bilo potrebno utrošiti. Umjesto takvog testiranja, testiran je samo manji podskup svih pravila koja bi trebala biti odbačena. Testovi su provedeni skeniranjem servisa Nmapom. Ako je određeni servis bio dostupan, a nije trebao biti onda je algoritam neispravan za taj skup pravila, u suprotnom algoritam je uspješno prošao navedeni test.

Treći uvjet provjerava se gotovo jednako kao i četvrti, testiranje se obavlja provjerom svakog servisa koji bi prema pravilima visoke razine apstrakcije trebao biti dostupan. Ako je svaki servis definiran u pravilima dostupan onda je i ovaj testni uvjet uspješno obavljen. Testiranje Nmap-om može se provoditi pokretanjem jednostavne naredbe: `nmap -n -Pn <port> <ip adresa>`.

Jednako kao i za prvu komponentu i u ovom slučaju su svi uvjeti bili zadovoljeni te je zaključak da cijela aplikacija radi ispravno.

4.2. Demonstracija aplikacije

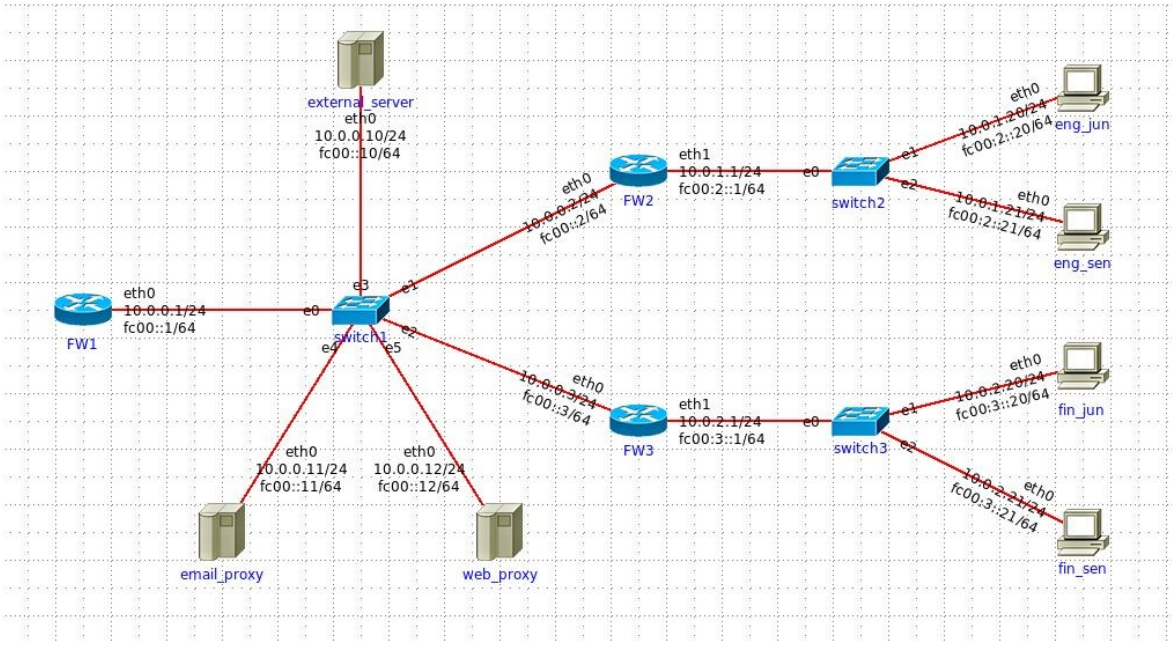
Kako je već najavljeno u uvodu ovog poglavlja, ovo potpoglavlje prikazat će neke ključne trenutke izvođenja aplikacije. Ova demonstracija uključivat će samo algoritam za smještanje vatrozida u mrežu jer je ta komponenta puno zahtjevnija za razumijevanje dok je algoritam za podešavanje vatrozida intuitivniji.

Sva pravila koja se koriste u sljedećim primjerima nalazit će se u vatrozid_pravila.json datoteci koja se predaje u sklopu ovog diplomskog rada te neće biti navedena u ovom tekstu zbog količine teksta kojeg bi pravila bespotrebno zauzela.

Sljedeća slika (Slika 4.1) u alatu IMUNES prikazuje konačnu mrežnu topologiju dobivenu korištenjem aplikacije. Ta mreža sastoji se od:

- Tri vatrozida: FW1, FW2, FW3
- Tri preklopnika
- Tri poslužitelja: email_proxy, web_proxy, external_server
- Četiri računala: eng_jun, eng_sen, fin_jun, fin_sen

U ovoj mrežnoj topologiji mogu se uočiti tri podmreže, jedna mreža sadrži računala *junior* i *senior* inženjera, ta računala označena su s eng_jun i eng_sen, druga podmreža sastoji se od fin_jun i fin_sen računala, koja predstavljaju financijski odjel organizacije.



Slika 4.1 – Generirana mrežna topologija

Zadnja podmreža sastoji se od poslužitelja, ta mreža predstavlja DMZ. Sva komunikacija računala iz prve dvije podmreže i računala na Internetu odvija se kroz DMZ. U ovoj podmreži nalaze se *email* i *web* posrednički poslužitelji, no izostavljeni su poslužitelji za te servise. To je napravljeno tako zbog jednostavnosti prikaza manjeg broja poslužitelja te jer je naglasak rada na sigurnosnim svojstvima računalne mreže. Naime posrednički poslužitelji potrebni su kako bi nadzirali i upravljali mrežnim prometom zaposlenika organizacije dok se obični poslužitelji mogu nalaziti i izvan ove mreže.

Sad kad je prikazano i objašnjeno krajnje željeno stanje mreže, vrijeme je za prikazati izvođenje prvog dijela aplikacije.

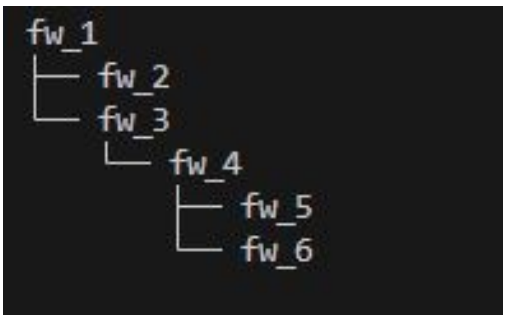
Prva funkcija koja se poziva je compute-GCD te je njezin rezultat skup skupova GCD objekata koji predstavljaju vatrozide. Na slici (Slika 4.2) prikazan je skup tih objekata. Odnosno prikazan je skup računala tog GCD skupa, GCD objekti osim skupa računala sadrže i skup pravila, no pravila nisu trenutno važna.

```
['eng_junior', 'eng_senior']
['fin_junior', 'fin_senior']
['fin_junior', 'fin_senior', 'eng_junior']
['fin_junior', 'fin_senior', 'eng_junior', 'eng_senior']
['web_proxy', 'email_proxy', 'fin_junior', 'fin_senior', 'eng_junior']
['web_proxy', 'email_proxy', 'external_server', 'fin_junior', 'fin_senior', 'eng_junior', 'eng_senior']
```

Slika 4.2 – Skup GCD objekata

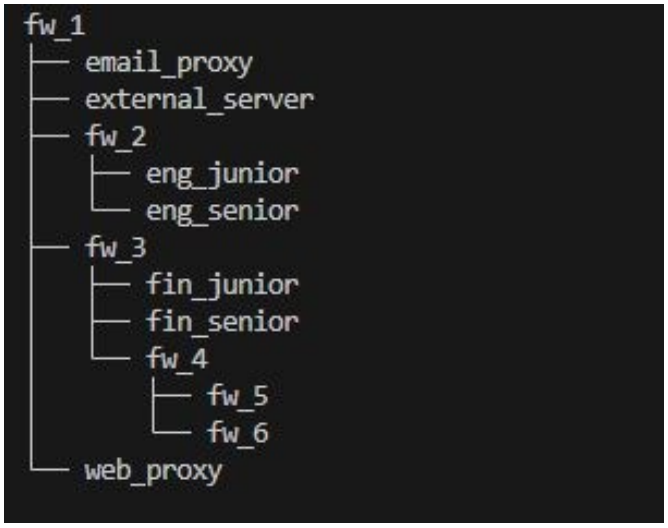
Lako je uočiti da je na slici prikazano šest GCD objekata, a krajnja mrežna topologija ima samo tri vatrozida. Razlog za to je taj da je dio vatrozida redundantan, to će biti jasno nakon prikaza optimize-network-graph funkcije.

Nakon završetka compute-GCD funkcije, sljedeća na redu je build-network-graph funkcija. Ta funkcija će u mrežu dodati sve GCD objekte iz prethodnog koraka, prikaz toga može se vidjeti na slici (Slika 4.3).



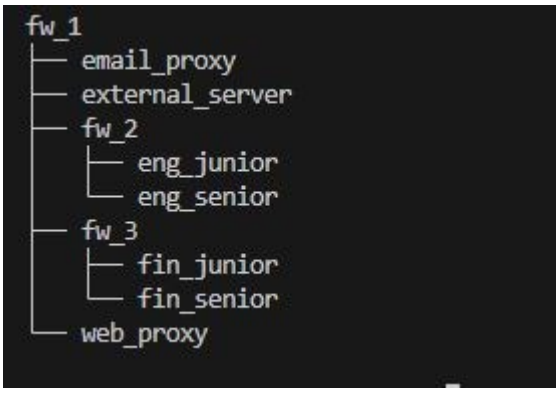
Slika 4.3 – Vatrozidi nakon pokretanja build-network-graph funckije

Trenutno se mreža sastoji samo od vatrozida, za dodavanje krajnjih računala zadužena je add-resources-to-net funkcija. Računala će u mrežu biti dodana kao čvorovi djeca u najnižu moguću razinu stabla kako bi na primjer računalo *eng_junior* bilo kao dijete FW2, a ne FW1 vatrozida. Rezultat toga prikazan je na (Slika 4.4)



Slika 4.4 – Vatrozidi nakon pokretanja add-resources-to-net funkcije

Očito je da nakon završetka prethodne funkcije, vatrozidi FW4, FW5 i FW6 nisu potrebni. Njihovo uklanjanje obavlja optimize-network-graph funkcija te je konačna mrežna topologija prikazana slikom (Slika 4.5).



Slika 4.5 – Konačna mrežna topologija nakon kraja algoritma

Ovim poglavljem završava opis razvijene aplikacije te će u idućem zaključnom poglavlju biti sažeto sve što je opisano u ovom diplomskom radu.

Zaključak

Tema ovog diplomskog rada bila je razvoj aplikacije za smještanje i podešavanje vatrozida u mreži na temelju politika visoke razine apstrakcije. Aplikacija se sastoji od dvije komponente. Prva komponenta zadužena je za smještanje vatrozida u mrežu dok je druga zadužena za određivanje vatrozida koje je potrebno podesiti određenim pravilom te podešavanje istih.

Smještanje vatrozida u mrežu temelji se na principu sličnosti mrežnog prometa skupa računala. Odnosno što je skup pravila za vatrozid jednog računala sličniji skupu pravila drugog računala, to je veća vjerojatnost da će isti vatrozid biti zadužen za filtriranje prometa tih računala. Ovaj dio aplikacije kao rezultat generira mrežnu topologiju strukture stabla koja se predaje komponenti za podešavanje vatrozida. Na temelju te strukture, komponenta za podešavanje vatrozida određuje koje vatrozide je potrebno podesiti za pojedino pravilo te prevodi pravila visoke razine apstrakcije u iptables pravila. Također na kraju rada, prikazana je demonstracija izvođenja algoritma za smještanje vatrozida u mrežu kako bi se intuitivno opisao rad algoritma.

Ovaj rad rezultirao je uspješnom implementacijom dvaju navedenih algoritama te vjerujem da razvijena aplikacija uvelike olakšava pisanje pravila za vatrozide. Velika prednost ove aplikacije je korištenje vatrozid pravila na visokoj razini apstrakcije jer su takva pravila mnogo kraća i intuitivnija te se tako olakšava podešavanje vatrozida i dolazi do manjeg broja pogrešaka.

Daljnji rad na aplikaciji može se usmjeriti na definiranje novih metoda za određivanje sličnosti mrežnog prometa te proširivanju podržanih vrsta pravila za vatrozid kao na primjer Nftables pravila.

Literatura

[1] „Report: Recent 10x Increase in Cyberattacks on Ukraine“, *krebsonsecurity.com*, Mar. 11, 2022. https://krebsonsecurity.com/2022/03/report-recent-10x-increase-in-cyberattacks-on-ukraine. [Accessed May. 27, 2023].

[2] W. R. Cheswick, S. M. Bellovin, and A. D. Rubin, Firewalls and Internet Security: Repelling the Wily Hacker. Addison-Wesley, 2003.

[3] N. Mhaskar, M. Alabbad, and R. Khedri, ‘A formal approach to network segmentation’, Computers & Security, vol. 103, p. 102162, 2021.

[4] “Welcome to treelib’s documentation! — treelib 1.5.5 documentation,” treelib.readthedocs.io. https://treelib.readthedocs.io/en/latest/ [Accessed May. 28, 2023].

[5] Python Software Foundation, “Welcome to Python.org,” *Python.org*, 2019. https://www.python.org/doc/ [Accessed May. 28, 2023].

[6] “netfilter/iptables project homepage - The netfilter.org ‘iptables’ project,” *Netfilter.org*, 2014. https://netfilter.org/projects/iptables/index.html [Accessed May. 29, 2023].

[7] “Iptables Tutorial: Ultimate Guide to Linux Firewall,” *Knowledge Base by phoenixNAP*, Jan. 28, 2020. https://phoenixnap.com/kb/iptables-tutorial-linux-firewall [Accessed May. 29, 2023].

[8] E. Karafili, F. Valenza, Y. Chen, and E. C. Lupu, ‘Towards a Framework for Automatic Firewalls Configuration via Argumentation Reasoning’, in *NOMS 2020 - 2020 IEEE/IFIP Network Operations and Management Symposium*, 2020, pp. 1–4.

[9] “The Netfilter.org ‘nftables’ project,” netfilter/iptables project homepage - The netfilter.org “nftables” project, https://netfilter.org/projects/nftables/ [Accessed May. 31, 2023].

[10] IMUNES - IP network emulator / simulator,” *imunes.net*. http://imunes.net/ [Accessed Jun. 05, 2023]

[11] nmap.org, “Chapter 15. Nmap Reference Guide | Nmap Network Scanning,” *Nmap.org*, 2019. https://nmap.org/book/man.html [Accessed Jun. 05, 2023]

[12] E. W. Dijkstra, ‘Guarded Commands, Nondeterminacy, and Formal Derivation of Programs’, in *Programming Methodology: A Collection of Articles by Members of IFIP WG2.3*, D. Gries, Ed. New York, NY: Springer New York, 1978, pp. 166–175.

[13] P. Höfner, R. Khedri, and B. Möller, ‘Algebraic View Reconciliation’, in *2008 Sixth IEEE International Conference on Software Engineering and Formal Methods*, 2008, pp. 85–94.

Smještanje i podešavanje vatrozida u mreži na temelju politika visoke razine apstrakcije

Sažetak

Velike organizacije mogu imati velik broj uređaja u svojoj računalnoj mreži te smještanje i podešavanje vatrozida u takvim mrežama može zahtijevati velik napor. Isto tako podešavanjem velikog broja vatrozida lako može doći do pogreške koja može rezultirati sigurnosnim rizikom. Iz navedenih razloga ovaj diplomski rad istražuje problem ručnog smještanja i podešavanja vatrozida u računalnim mrežama te predstavlja razvijenu aplikaciju za automatizaciju tih procesa.

Aplikacija je podijeljena u dva dijela, prvi dio aplikacije određuje raspored vatrozida u mreži temeljem sličnosti prometa između uređaja te određuje topologiju cijele mreže koja se koristi u drugom dijelu aplikacije. Drugi dio obavlja podešavanje vatrozida na temelju odabranih sigurnosnih pravila, prevodeći ih u iptables format.

Aplikacija uspješno rješava navedene probleme te demonstrira da ovo područje računarstva ima jako velik potencijal za daljnji razvoj.

Ključne riječi: Vatrozid, Iptables, Računalna mreža, Mrežna topologija

Network placement and configuration of firewalls based on high-level policies

Abstract

Large organizations can have a significant number of devices in their computer network, placing and configuring firewalls in such networks can require a substantial effort. Similarly, configuring a large number of firewalls can easily lead to errors that may result in security risks. For these reasons, this thesis explores the problem of manual firewall placement and configuration in computer networks and presents a developed application for automating these processes.

The application is divided into two parts. The first part determines the placement of firewalls in the network based on traffic similarities between devices and establishes the topology of the entire network, which is then used in the second part of the application. The second part handles the configuration of firewalls based on selected security rules, translating them into the iptables format.

The application successfully addresses the aforementioned issues and demonstrates that this area of computer science has great potential for further development.

Keywords: Firewall, Iptables, Computer network, Network topology